

Dov Dori

Modelové systémové inženýrství s využitím OPM a SysML

Modelové systémové inženýrství s OPM a SysML

Dov Dori

Modelové systémové inženýrství s OPM a SysML

Předmluva Edward Crawley



Springer

Dov Dori
Technion, Izraelský technologický institut
Haifa, Izrael

Massachusetts Institute of Technology
Cambridge, MA, USA

Prezentace kapitol a otázky a odpovědi na konci kapitol najdete na [adrese http://esml.iem.technion.ac.il/qanswer/](http://esml.iem.technion.ac.il/qanswer/)

ISBN 978-1-4939-3294-8 ISBN 978-1-4939-3295-5 (eBook)
DOI 10.1007/978-1-4939-3295-5

Kontrolní číslo Kongresové knihovny:

2015954175 Springer New York Heidelberg

Dordrecht Londýn

© Springer Science+Business Media New York 2016

Toto dílo podléhá autorským právům. Vydavatel si vyhrazuje všechna práva, ať už se jedná o celý materiál nebo jeho část, konkrétně práva na překlad, dotisk, opětovné použití ilustrací, recitaci, vysílání, reprodukci na mikrofilmtech nebo jakýmkoli jiným fyzickým způsobem a přenos nebo ukládání a vyhledávání informací, elektronickou adaptaci, počítačový software nebo podobnou či odlišnou metodikou, která je v současnosti známa nebo bude vyvinuta v budoucnu.

Použití obecných popisných názvů, registrovaných názvů, ochranných známek, servisních značek atd. v této publikaci neznámá, ani v případě absence konkrétního prohlášení, že tyto názvy jsou osvobozeny od příslušných ochranných zákonů a předpisů, a jsou tedy volně použitelné pro obecné účely.

Vydavatel, autoři a redaktori mohou s jistotou předpokládat, že rady a informace obsažené v této knize jsou k datu vydání považovány za pravdivé a přesné. Vydavatel, autoři ani redaktori neposkytují žádnou výslovnou ani implicitní záruku ohledně obsahu této knihy ani ohledně případných chyb nebo opomenutí, které se v ní mohou vyskytovat.

Vytištěno na bezkyselinovém papíře

Springer Science+Business Media LLC New York je součástí společnosti Springer Science+Business Media (www.springer.com)

Předmluva

Architektura a inženýrství velkých, komplexních sociálně-technických systémů, stejně jako hluboké porozumění stávajícím přírodním a umělým systémům, unikalo lidem po mnoho let. Naše myšlení o systémech a jejich roli při zlepšování kvality života lidí se za poslední dvě desetiletí vyvinulo. Nyní lépe chápeme, že úspěšné systémy nevznikají náhodně. Musí být pečlivě navrženy, stejně jako budovy, s ohledem na potřeby, přání a požadavky jejich zamýšlených uživatelů, a alternativní architektury – způsoby, jakými jsou tyto požadované funkce ztělesněny ve formě. Tyto rané rozhodnutí jsou pro budoucí systém zásadní, protože určují koncept, který bude následován, a následně celý směr vývoje systému a povahu konečného výsledku: jak dobře systém funguje z hlediska poskytování hodnoty, tj. přínosu v poměru k nákladům, při zachování dalších požadavků na bezpečnost, robustnost, snadné použití, šetrnost k životnímu prostředí a mnoho dalších.

Když jsem před asi 15 lety získával tyto poznatky, uvědomil jsem si, že bez ohledu na to, jak přesvědčivé jsou vaše nápady a jak pádné jsou argumenty, s kázáním a máváním rukama toho moc nezmůžete. Bylo mi jasné, že pokrok v oblasti architektury a inženýrství komplexních systémů závisí na pevných základech jazyka a metodiky. Náhodou se stalo, že v té době, kolem roku 2000, vešel Dov do mé kanceláře, kanceláře vedoucího katedry letectví a kosmonautiky na MIT, s návrhem své první knihy s názvem *Object-Process Methodology—A Holistic Systems Paradigm* (Springer, 2002). Když jsem si tento návrh přečetl v letadle, okamžitě jsem pochopil, že to, co držím v ruce, je přesně to, co jsem hledal.

Object-Process Methodology (OPM) je paradigma systémového modelování, které reprezentuje dvě věci vlastní systému: jeho objekty a procesy. OPM je v zásadě jednoduché; staví na minimálním souboru pojmů: objekty se stavem – věci, které existují, a procesy – věci, které se dějí a transformují objekty tím, že je vytvářejí nebo spotřebovávají nebo mění jejich stavy. Tato dualita je uznávána v celé komunitě, která studuje systémy, a někdy se označuje jako forma/funkce, struktura/funkce a funkční požadavky/parametry návrhu. Objekty jsou tím, čím systém nebo produkt je. Procesy jsou tím, co systém dělá. Je však pozoruhodné, že tak málo modelovacích rámců tuto dualitu explicitně uznává. Výsledkem je, že designéři a inženýři se snaží přejít od cílů systému (požadavků nebo „programu“) okamžitě k objektům. Seriózní teorie v tak odlišných oborech, jako je návrh softwaru, mechanický návrh a návrh civilní architektury, uznávají hodnotu uvažování o procesech souběžně s objekty. OPM nejenže reprezentuje objekty i procesy, ale také explicitně ukazuje souvislosti mezi nimi.

OPM má další zásadní výhodu – systém představuje současně ve formální grafice a v přirozeném jazyce. Oba jsou zcela zaměnitelné a sdělují přesně stejné informace. Výhoda tohoto přístupu spočívá v uznání lidských omezení v porozumění složitosti. Jak se systémy stávají složitějšími, hlavní překážkou úspěchu je schopnost lidských návrhářů a analytiků porozumět složitosti vzájemných vztahů. Zobrazením systému v textové i grafické podobě se zapojuje síla „obou stran mozku“ – vizuálního interpretátora a jazykového interpretátora. Jedná se o dvě nejsilnější zpracovatelské schopnosti, které jsou pevně zakotveny v lidském mozku. Jelikož je každý modelový fakt vyjádřen graficky i textově, v podskupině

přirozené angličtiny, je snadno přístupný i pro technicky neznalé zúčastněné strany, což jim umožňuje podílet se na raných, kritických fázích architektury a vývoje systému, ve kterých se přijímají nejdůležitější rozhodnutí.

OPM umožňuje jasné znázornění mnoha důležitých vlastností systému: jeho topologických spojení; jeho rozložení na prvky a podprvky; rozhraní mezi prvky; a vznik funkce z prvků. Tvůrce nebo prohlížeč modelu může zobrazit abstrakce nebo přiblížit detaily. Lze vidět, jak se specifikace přenáší do implementace. Tyto různé pohledy jsou neocenitelné při zvažování složitosti skutečného moderního produktového systému.

Sémantika OPM byla původně zaměřena na systémové inženýrství, protože dokáže modelovat informace, hardware, lidi a regulace. V posledních letech však OPM začala sloužit také výzkumníkům v molekulární biologii a přinesla hmatatelné nové poznatky týkající se životního cyklu mRNA. To je jasným důkazem univerzálnosti ontologie objektů a procesů: Ukazuje se, že tento minimální soubor pojmů lze použít k modelování systémů prakticky v jakékoli oblasti. Jedinou výjimkou je snad kvantová fyzika, kde se naše makro pojmy částice (objekt?) a vlna (proces?), stejně jako hmota (objekt?) a energie (proces?) stávají nejasnými, když se pokoušíme „prozkoumat“ subatomární částice, jako jsou elektrony. Pro jakýkoli systém od molekulární úrovně až po nejsložitější přírodní, sociotechnické a společenské systémy funguje paradigma objekt-proces velmi dobře. Modely OPM současně vysvětlují funkci (užitečnost), strukturu (formu) a chování (dynamiku) systémů v jediném soudržném modelu, který používá jeden druh diagramu na libovolném počtu úrovní detailů tím, že se podrobně zabývá detaily procesů ruku v ruce s objekty, které transformují. Sada těchto samopodobných diagramů objekt-proces je reprezentována nejen vizuálně, ale také textově, což vyhovuje duálnímu zpracování informací lidským mozkem, což je klíčový kognitivní předpoklad toho, jak zpracováváme informace a přeměňujeme je na využitelné znalosti.

Poté, co jsem si uvědomil hodnotu OPM pro systémovou architekturu a inženýrství, přijal jsem jej do svého myšlení a výuky a stal se důležitým základním kamenem kurzů, které učím v oblasti systémové architektury na MIT a jinde. OPM se zejména stal klíčovým prvkem výuky základních kurzů v postgraduálním programu Systémový design a management na MIT. OPM jsem použil v kurzu SDM System Architecture. Ukázalo se, že je neocenitelným nástrojem pro profesionální studenty při vývoji modelů složitých technických systémů, jako jsou automobily, kosmické lodě a softwarové systémy. Umožňuje explicitní reprezentaci duality formy a funkce a poskytuje prostředí, ve kterém lze zkoumat různé architektonické možnosti. Přidání OPM do mého předmětu zvýšilo míru přesnosti analýzy nezbytné k posunu studia architektury technických systémů směrem k inženýrské disciplíně.

OPM se také používá jako reprezentativní rámec v nové knize, na které jsem se podílel jako spoluautor, *System Architecture: Strategy and Product Development for Complex Systems* (Pearson, 2015), která rozvíjí přístup k architektuře a demonstruje jej na příkladech od čerpadel, obvodů a třídících algoritmů až po komplexní systémy v sítích a hybridních automobilech. Úkol navrhování a vývoje nového systému se skutečně stal složitějším kvůli rostoucímu počtu zapojených komponent, počtu různých oborů potřebných k provedení úkolu a rostoucí velikosti zapojených organizací. Navzdory společným zkušenostem, které sdílejí členové mnoha organizací, často postrádají společný slovník pro vývoj produktů nebo modelovací rámec. Takový rámec by měl být založen na systémové vědě, měl by být schopen reprezentovat všechny důležité interakce v systému

systému a být široce použitelný pro elektrické, informační, mechanické, optické, tepelné a lidské komponenty.

OPM takový rámec poskytuje. V roce 2008 pracovní skupina Mezinárodní rady pro systémové inženýrství (INCOSE) uznala OPM jako jednu ze šesti předních metodik systémového inženýrství založených na modelech. Při pohledu na historický vývoj inženýrských oborů je nyní vhodná doba pro vznik takového přísného rámce. Oborové disciplíny často procházejí postupným zráním. Na počátku historie intelektuální disciplíny nacházíme pozorování přírody nebo praxe, které se rychle vyvíjí přes období, ve kterém se věci klasifikují. Průlom často nastává, když jsou klasifikovaná pozorování abstrahována a kvantifikována. Tyto fáze charakterizují většinu práce dosud provedené v systémovém inženýrství a vývoji produktů. Zralé disciplíny, jako je mechanika, jsou již v éře symbolické manipulace a predikce. Zrající disciplíny, jako je genomika člověka, jsou ve fázi symbolické reprezentace.

OPM je paralelním vývojem v symbolické reprezentaci systémů. Během posledních dvou desetiletí se rozšířilo pochopení potřeby systematické modelovací schopnosti. V reakci na toto rostoucí uznání byl vyvinut OPM, stejně jako další přístupy. Nejvýznamnější z nich jsou UML, který zahrnuje 13 druhů diagramů zaměřených na softwarové inženýrství, a jeho derivát SysML, který zahrnuje devět druhů diagramů navržených obecněji pro systémové inženýrství. SysML i OPM jsou uvedeny jako přední standardy v příručce Guide to the Systems Engineering Body of Knowledge (SEBoK), což je online projekt, který společně sponzorují Systems Engineering Research Center (SERC), International Council on Systems Engineering (INCOSE) a Institute of Electrical and Electronics Engineers Computer Society (IEEE-CS).

SysML a OPM představují dva různé přístupy k modelování systémů. V SysML se používá až devět diagramů, které jsou odvozeny nezávisle na sobě a nemusí být zcela konzistentní. V OPM vzniká jeden hlavní diagram. Potřeba integrovat několik druhů diagramů může být *složitější*. Rozlišují mezi složitostí – inherentní skutečností, že systém obsahuje mnoho částí, které vzájemně interagují mnoha, často nevysvětlitelnými způsoby, a komplikovaností – způsobem, jakým je systémový model prezentován prostřednictvím určitého modelovacího jazyka a jak je vnímán uživatelem. I když nemůžeme udělat mnoho pro snížení inherentní složitosti systémů, můžeme a měli bychom se snažit snížit komplikovanost reprezentace na nezbytné minimum, aniž bychom obětovali přesnost a detaily. OPM s minimální ontologií stavových objektů a procesů na tuto výzvu reaguje příznivě.

Ačkoli se kniha zaměřuje na OPM, vzhledem k relativně širšímu rozšíření SysML zahrnul Dov do knihy také SysML s adekvátním popisem jeho syntaxe a sémantiky, stejně jako synergie s OPM, srovnání s OPM z hlediska faktorů, jako je délka standardní specifikace, a způsoby, jakými OPM může nahradit mnoho typů diagramů SysML jediným typem diagramu. Zahrnutí obou jazyků do jedné knihy je ojedinělé, protože knihy o modelovém systémovém inženýrství (MBSE) dosud většinou používaly SysML. Toto dvojí pokrytí OPM a SysML je velmi cenné, protože čtenář získá hlubší pohled na MBSE, který proniká pod povrch zvláštností konkrétního konceptuálního modelovacího jazyka.

Doporučuji použít tuto učebnici pro středně pokročilé nebo pokročilé kurzy modelového systémového inženýrství, vývoje produktů, inženýrského designu a softwarového inženýrství. Byla by ideální pro kurzy, které se snaží ukázat, jak se různé disciplíny spojují a tvoří multidisciplinární produkt. Nyní, když je OPM formálně uznán jako specifikace ISO, může tvořit páteř podnikového nebo

podnikového modelovacího rámce pro technické produkty a rozsáhlé sociotechnické systémy. Takové znázornění by bylo obzvláště cenné v koncepčním a předběžném návrhu, kdy je stanovena většina hodnoty, nákladů a rizik produktu, ale kdy je k dispozici jen málo jiných modelovacích rámců pro podporu rozhodování.¹

Profesor Edward Crawley

Massachusetts Institute of Technology, červenec 2015

¹Edward Crawley je Fordovým profesorem inženýrství a profesorem letectví a kosmonautiky na MIT. Je členem Americké národní akademie inženýrství a působí jako prezident Skolkovského institutu vědy a technologie v Moskvě. Prof. Crawley je prvním autorem dvou nedávno vydaných knih: „System Architecture: Strategy and Product Development for Complex Systems” (Architektura systému: Strategie a vývoj produktů pro komplexní systémy) a „Rethinking Engineering Education, the CDIO Approach” (Přehodnocení inženýrského vzdělávání, přístup CDIO).

Předmluva

Hledání jednoduchosti ve složitém světě zaměstnává myslitele již po tisíciletí. Jak konceptualizovat to, co lidé pozorují kolem sebe, a co chtějí navrhnout, aby zlepšili kvalitu života lidí, bylo jednou z hlavních hnacích sil pokroku civilizace. Příchod počítačů v polovině minulého století byl velkým impulsem k rozvoji myšlenek o tom, jak konceptuálně reprezentovat věci v reálném světě. Původní přijímaný způsob uvažování vedl k procedurálnímu programování, které do centra programování stavělo procedury, rutiny, funkce atd. Další úvahy vedly k myšlence, že základem programů by měly být objekty, které jsou svou povahou statictější. Po přechodu k objektově orientovanému (OO) paradigmatu programovacích jazyků, k němuž došlo v 80. a 90. letech, následovala myšlenka, že programování by mělo být předcházeno analýzou a návrhem programů, nebo obecněji řečeno systémů, které tyto programy reprezentují a kterým slouží. Přirozeně je tento přístup také objektově orientovaný.

Na počátku 90. let se objevila řada asi tří desítek objektově orientovaných metod analýzy a návrhu a notací, což vedlo k takzvané „válce metod“. V té době, v roce 1991, když jsem přešel z University of Kansas na Technion, Israel Institute of Technology, kde jsem měl za úkol vyučovat návrh softwaru, jsem se začal o tato témata zajímat. Netrvalo dlouho a uvědomil jsem si, že stejně jako procedurální přístup k softwaru byl nedostatečný, tak i „čistý“ objektově orientovaný přístup, který klade objekty jako jediné „prvotřídní“ prvky, s „metodami“ (nebo „službami“) jako jejich podřízenými procedurami druhé třídy. Nemohl jsem však přijít na to, co mi chybí.

Můj eureka moment nastal v roce 1993, když jsme se s kolegy z University of Washington pokoušeli modelovat systém pro automatickou transformaci ručně kreslených technických výkresů do CAD modelů, což bylo téma, na které se v té době zaměřoval můj výzkum. Když jsem kreslil objekty jako stavební bloky modelu a spojoval je na bílé tabuli, uvědomil jsem si, že ne všechny krabice v modelu jsou skutečně objekty; některé jsou věci, které se objektům dějí. Když jsem tyto věci zakroužkoval, objevil se vzor bipartitního grafu, kde uzly představující objekty – věci, které existují – byly zprostředkovány zakroužkovanými uzly, které jsem okamžitě nazval procesy. Byl to první objektově-procesní diagram (OPD), který kdy byl nakreslen. Tehdy jsem si uvědomil, že přechod od dříve přijímaného procedurálního softwaru k primárně statickému paradigmatu OO byl příliš drastický. Přestože posun od procedur k objektům jakožto středobod zájmu byl správným krokem, zašel příliš daleko, protože potlačil procedurální aspekt systémů, který je nezbytný pro věrný popis toho, jak se systémy v průběhu času mění.

Zákaz procesů, jako je pečení dortů nebo proplácení šeků, jako samostatných koncepčních entit a povolení jejich reprezentace pouze jako metod tříd objektů vede ke zkresleným modelům, ve kterých šek „vlastní“ metodu proplácení nebo dort vlastní proces pečení. Ve skutečném životě je však pečení vzorem transformace ingrediencí tvořících těsto, který vyžaduje pekaře, troubu a energii k přípravě těsta a jeho přeměně na dort. Podobně šek nemůže být proplacen sám; vyžaduje to vystavitele šeku s účtem s dostatečnými prostředky, proplácejícího šeku a bankovního úředníka nebo bankomat. Každý z objektů zapojených do těchto metod by mohl být stejně dobře vlastníkem metody. Modelování pečení a proplácení jako samostatných procesů – koncepčních věcí, které představují fyzické nebo informatické vzorce transformace objektů – otevírá dveře k vytváření modelů, které jsou mnohem věrnější způsobu, jakým chápeme realitu a předáváme ji ostatním.

Uznání procesů jako skutečných koncepčních stavebních kamenů vedle objektů, nikoli pod nimi, je základním kamenem metodiky objekt-proces (OPM). OPM je založena na univerzální minimální ontologii, podle které objekty existují, zatímco procesy je transformují. Transformace zahrnuje vytváření a spotřebu objektů, stejně jako změnu stavu objektu. Objekty OPM jsou proto stavové – mohou mít stavy. Stavové objekty a procesy, které je transformují, jsou tedy jedinými dvěma pojmy v univerzální minimální ontologii OPM. Dalšími dvěma základními kameny OPM jsou její bimodální graficko-textová reprezentace a její vestavěné mechanismy řízení složitosti zjemňování a abstrakce pomocí přiblížení a rozbalení jediného typu diagramu – OPD.

Když jsem se v roce 1993 pokusil publikovat článek s názvem „Object-Process Analysis: Maintaining the Balance between System Structure and Behavior” (Analýza objektů a procesů: udržování rovnováhy mezi strukturou a chováním systému) s náznaky těchto myšlenek, byl sériově odmítnut s tvrzením, že již bylo prokázáno, že to, co navrhuji, je nemožné, jako „míchání vody s olejem”. Nakonec jej přijal časopis *Journal of Logic and Computation*, možná proto, že byl spíše matematicky než softwarově orientovaný a byl tolerantnější k myšlenkám, které šly proti tehdy novému a slavnému paradigmatu OO.

Mezitím, v roce 1997, vyvrcholily „metodické války“ přijetím Unified Modeling Languages (UML) skupinou Object Management Group (OMG), čímž se stal de facto standardem pro návrh softwaru. UML 1 měl devět typů diagramů. V roce 2000, když jsem se zúčastnil technického setkání OMG, na kterém se uvažovalo o přechodu z verze 1 na verzi 2 UML, navrhl jsem, aby se UML rozšířil tak, aby nezahrnoval pouze softwarové systémy, ale systémy obecně. Tento návrh byl však většinou účastníků, kteří byli softwaroví odborníci, okamžitě zamítnut. Na základě iniciativy Mezinárodní rady pro systémové inženýrství (INCOSE) z roku 2001 však OMG v roce 2003 vydala výzvu k podávání návrhů na UML pro systémové inženýrství a v roce 2006 přijala specifikaci SysML (Systems Modeling Language) 1.0, která je založena na UML 2. Od té doby se SysML stal de facto standardem pro systémové inženýrství.

Mezitím byla vydána první kniha o OPM, *Object-Process Methodology—a Holistic Systems Paradigm* (Dori, 2002), a OPM byla úspěšně aplikována a publikována v mnoha různých oblastech, od sémantického webu přes obranu až po molekulární biologii. V prosinci 2015, po šesti letech práce, ISO přijalo a vydalo OPM jako ISO 19450 – *Automatizační systémy a integrace – Objektově-procesní metodika*.

V posledních letech nabývá na síle poznání a uznání, že modely mohou a měly by se stát ústředním artefaktem v životním cyklu systémů, což vedlo k vzniku modelového systémového inženýrství (MBSE) jako rozvíjejícího se oboru v oblasti systémového inženýrství. SysML a OPM slouží jako dva jazyky MBSE, ale protože SysML byl přijat jako standard asi osm let před OPM a je podporován špičkovými dodavateli, je jeho přijetí v současné době rozšířenější. OPM si však rychle získává uznání v akademické sféře a jeho aplikace se šíří v různých průmyslových odvětvích.

Tato učebnice, určená jak pro samostudium, tak pro vysokoškolské studium, poskytuje čtenářům hluboké porozumění myšlenkám, principům a aplikacím MBSE prostřednictvím modelování systémů pomocí OPM i SysML. Kniha se skládá ze tří částí, které zahrnují 24 kapitol. Každá kapitola končí shrnutím v bodech a sadou úloh. Řešení úloh je k dispozici na adrese <http://esml.iem.technion.ac.il/>.

Část I představuje OPM a SysML prostřednictvím postupného modelování automatického systému reakce na nehodu automobilu. **Kapitola 1** začíná popisem systému a jeho počátečního modelu OPM. V **kapitole 2** vylepšujeme model pomocí textu a animované simulace. **Kapitola 3** představuje odkazy, které propojují prvky v

v modelu. V kapitole 4 představujeme a používáme první tři diagramy SysML. Kapitola 5 představuje způsoby řízení složitosti systémů, zatímco dynamický aspekt systému je modelován v kapitolách 6 a 7. Mechanismy abstrakce a zdokonalování jako prostředky ke zvládnání složitosti jsou tématem kapitoly 8, poslední kapitoly části I.

Část II, *Základy modelového inženýrství systémů*, je formální, teoreticky podložená expozice OPM a SysML, která pojednává o ontologii MBSE, koncepčních modelovacích konstruktech a aplikacích. Kapitola 9 představuje a definuje koncepční modelování. Kapitola 10 představuje dva základní stavební kameny OPM – objekty a procesy, zatímco kapitola 11 se zabývá textovou modalitou OPM – OPL. V kapitole 12 se věnujeme systematickému studiu SysML s jeho čtyřmi pilíři a devíti druhy diagramů. Dynamický, časově závislý aspekt systémů je tématem kapitoly 13, na kterou navazuje studium strukturálního, časově nezávislého aspektu systému v kapitole 14. Po kapitole 15, která se zabývá omezeními účasti a rozvětvenými odkazy, představujeme v kapitole 16 čtyři základní strukturální vztahy.

V části III, *Struktura a chování: Ponoření se do hloubky*, se dostáváme k jádru koncepčního modelování a podrobně rozebíráme čtyři základní strukturální vztahy a aspekty celého systému, včetně řízení a kontroly složitosti. Kapitoly 17 a 18 se zabývají agregací-účasti a exhibicí-charakterizací. Kapitola 19 se zabývá stavy a hodnotami, pojmy, které jsou potřebné pro generalizaci-specializaci a klasifikaci-instanci, které jsou podrobněji rozvedeny v kapitole 20. Kapitola 21 se týká řízení složitosti a mechanismů zpřesňování-abstrahování OPM, jakož i řízení složitosti v SysML. Kapitola 22 se zabývá operační sémantikou OPM a kontrolními odkazy – způsobem, jakým je řízení spravováno během provádění systému. V kapitole 23 specifikujeme, jak modelovat logické operátory a pravděpodobnosti. Kapitola 24 je přehledem normy ISO 19450 – *Automatizační systémy a integrace – Metodika objekt-proces*, přijaté Mezinárodní organizací pro normalizaci v prosinci 2015.

S ohledem na OPM lze tuto knihu považovat za nadmnožinu normy ISO 19450. Zatímco OPM, jak je specifikováno v této knize, je v souladu s normou ISO 19450, kniha poskytuje hloubkovou motivaci, zdůvodnění a filozofické základy pro rozhodnutí učiněná během návrhu normy ISO 19450. Tyto informace nelze podrobně rozvést v normě, která je ze své podstaty stručná a rozhodná a obsahuje jen málo odůvodnění. Body OPM uvedené v knize, které nejsou zahrnuty v normě ISO 19450, lze považovat za volitelné nebo, v terminologii ISO, za *informativní*, na rozdíl od normativních specifikací ISO.

Tato kniha je výsledkem šestileté práce, během níž jsem vynaložil veškeré úsilí, aby byla přesná, konzistentní a formální, ale zároveň neztratila lidský rozměr a nezajímala budoucí čtenáře. Upřímně doufám, že kniha poslouží jako spolehlivý zdroj informací o MBSE obecně a o OPM a SysML konkrétně.



Při zkoumání výše uvedeného slovního oblaku této knihy (vytvořeného programem, který dovedně vyvinul [Jason Davies](#)) ²na základě téměř 140 000 slov obsažených v této knize můžeme vidět, že nejčastějšími slovy jsou *proces*, *objekt* a *odkaz*. To je skutečně nejvěrnější svědectví toho, že OPM se zaměřuje na to, jak *modelovat systémy* (další dvě nejčastější slova v oblaku) propojením *procesů* s *objekty* pomocí *odkazů*. *Vztah* je zde také, spolu s dalšími významnými slovy, včetně *diagram*, *atribut*, *strukturální*, *procedurální*, *sémantika*, *stav*, *kontrola*, *změna*, *efekt*, *agent*, *čas*, *omezení* a *funkce*. Samozřejmě je zde také *SysML* mezi *procesem* a *modelem*, poblíž *OPD* (Object-Process Diagram – grafická modalita OPM) a *OPL* (Object-Process Language – textová modalita OPM). Tento seznam dobře vystihuje, o čem tato kniha je.

Rád bych poděkoval svým třem spolupracovníkům z MIT, prof. Edovi Crawleymu a prof. Oli de Weckovi z divize inženýrských systémů a katedry letectví a kosmonautiky a Pat Haleovi, řediteli programu systémového designu a managementu. Zvláštní poděkování patří mému doktorandovi Yanivu Mordecaiovi, který poskytl podnětné komentáře k mnoha kapitolám této knihy. Děkuji Technionu, Izraelskému technologickému institutu, který mi poskytl prostředí pro vývoj OPM a sabatický rok 2013–2014, abych mohl tuto knihu dokončit. Na závěr bych chtěl poděkovat své milované ženě, prof. Judy Dori, která mi poskytla pedagogické vedení a morální podporu, díky nimž jsem mohl knihu dokončit.

Dov Dori

Massachusetts Institute of Technology, červenec 2015

² <https://www.jasondavies.com/wordcloud/>

Obsah

Hlavní symboly OPM v souladu s normou ISO 19450	xxi
---	-----

ČÁST I Úvod do modelového systémového inženýrství

<i>Kapitola 1 Jste připraveni začít modelovat?</i>	3
--	---

1.1 Automatický systém reakce na nehodu	3
1.2 Princip OPM „funkce jako semínko“	4
1.3 Identifikace funkce systému	5
1.4 Identifikace příjemce systému	6
1.5 Proces transformuje objekt	6
1.6 Shrnutí	7
1.7 Problémy	8

<i>Kapitola 2 Vylepšení textu a simulace</i>	11
--	----

2.1 OPL: Podskupina angličtiny	11
2.2 Stavby a animovaná simulace	12
2.3 Animované provedení modelu OPM	15
2.4 Shrnutí	16
2.5 Problémy	16

<i>Kapitola 3 Propojování věcí pomocí odkazů</i>	19
--	----

3.1 Procedurální odkazy vs. strukturální odkazy	19
3.2 Přidávání aktivátorů	19
3.3 Přidávání strukturálních odkazů	22
3.4 Fyzické vs. informatické věci	23
3.5 Modelové skutečnosti a odstavce OPL	24
3.6 Environmentální vs. systémové věci	24
3.7 Počáteční a konečné stavy	25
3.8 Spouštěcí stav a odkaz na událost	27
3.9 Shrnutí	27
3.10 Problémy	28

<i>Kapitola 4 SysML: Diagramy případů použití, bloků a stavových strojů</i>	29
---	----

4.1 Diagram případů použití SysML	29
4.2 Bloky SysML a diagram definice bloků	32
4.3 Diagram stavového stroje SysML	33
4.4 Shrnutí	35
4.5 Problémy	35

<i>Kapitola 5 Zpřesnění pomocí přiblížení</i>	<i>37</i>
5.1 Měření závažnosti nehody	37
5.2 In-Zooming: Zdokonalení procesu v novém OPD	38
5.3 Strom OPD	38
5.4 Modelové znázornění faktů Princip OPM	39
5.5 Atribut závažnosti nehody a jeho měření	40
5.6 Simulace systému: animovaný test provedení	41
5.7 Shrnutí	42
5.8 Problémy	43
<i>Kapitola 6 Dynamický aspekt systémů</i>	<i>45</i>
6.1 Ukončení v případě mírné závažnosti	45
6.2 Vytváření a odesílání zpráv	45
6.3 Pořadí provádění procesů: Princip časové osy OPM	47
6.4 Pomoc je na cestě!	48
6.5 Scénáře: Vlákna provádění	49
6.6 Shrnutí	50
6.7 Problémy	51
<i>Kapitola 7 Ovládání chování systému</i>	<i>53</i>
7.1 Větvení s booleovskými objekty	53
7.2 Podmíněné propojení vs. nástrojové propojení	55
7.3 Generalizace-specializace	56
7.4 Zvětšení měření závažnosti nehody	57
7.5 Omezení účasti	58
7.6 Logické operátory: OR vs. XOR	58
7.7 Zpřesnění měření závažnosti nehody	58
7.8 Rozsah věcí: signál jako dočasný objekt	59
7.9 Jak se provádí diagnostika?	60
7.10 Shrnutí	60
7.11 Problémy	61
<i>Kapitola 8 Abstrahování a zdokonalování</i>	<i>63</i>
8.1 Zvětšení: Zjemnění procesu v novém OPD	63
8.2 Zpracování zpráv Zvětšeno	65
8.3 Strukturální pohled na systém ACR	67
8.4 Shrnutí	68
8.5 Problémy	69
 <i>ČÁST II Základy OPM a SysML</i>	
<i>Kapitola 9 Koncepční modelování: účel a kontext</i>	<i>75</i>
9.1 Systémy, modelování a systémové inženýrství	75
9.2 Základní ontologie systémového inženýrství OPM	76
9.3 Definice objektu, stavu, transformace a procesu	82

9.4 Systém a související pojmy	83
9.5 Jazyk a modelování	91
9.6 Shrnutí	94
9.7 Problémy	95
<i>Kapitola 10 Věci: objekty a procesy</i>	97
10.1 Objektově orientovaný přístup vs. přístup založený na objektech a procesech	97
10.2 Existence, věci a transformace	98
10.3 Identita objektu	99
10.4 Syntaxe vs. sémantika	102
10.5 Princip jedinečnosti procesního propojení OPM	106
10.6 Procesní test	109
10.7 Pojmenování prvků OPM	111
10.8 Definovaná věc	113
10.9 Vlastnosti věcí OPM	114
10.10 Hraniční případy věcí	116
10.11 Operátor, operand a transformace	119
10.12 Shrnutí	120
10.13 Problémy	121
<i>Kapitola 11 Objektově-procesní jazyk: Text</i>	123
11.1 OPL: Textová modalita	123
11.2 Dvojitý účel OPL	124
11.3 Princip graficko-textové ekvivalence OPM	125
11.4 Metamodel struktury modelu OPM	125
11.5 Vyhrazené a nevyhrazené fráze OPL	127
11.6 Motivace pro bimodální vyjádření OPM	129
11.7 Tesperanto: Čitelný automaticky generovaný text	131
11.8 Shrnutí	132
11.9 Problémy	133
<i>Kapitola 12 SysML: Základy a diagramy</i>	135
12.1 UML: Jednotný modelovací jazyk	135
12.2 Piliře SysML	136
12.3 Diagram požadavků	136
12.4 Bloky a struktura	137
12.5 Diagram aktivit	138
12.6 Sekvenční diagram	141
12.7 Diagram požadavků	143
12.8 Parametrický diagram a bloky vlastností omezení	146
12.9 Porovnání SysML a OPM	148
12.10 Synergie SysML–OPM	155
12.11 Shrnutí	155
12.12 Problémy	156
<i>Kapitola 13 Aspekt dynamického systému</i>	157

13.1 Změna a účinek.....	157
13.2 Existence a transformace	158
13.3 Procedurální odkazy.....	162
13.4 Transformační odkazy	163
13.5 Aktivátory	165
13.6 Sady objektů předběžného a následného zpracování.....	169
13.7 Stavově specifikované procedurální odkazy	170
13.8 Stavově specifikované povolovací odkazy	171
13.9 Státem specifikované transformační odkazy	173
13.10 Státem specifikované odkazy na účinky	175
13.11 Shrnutí.....	179
13.12 Problémy	181
<i>Kapitola 14 Aspekt strukturálního systému</i>	<i>183</i>
14.1 Strukturální vztahy.....	183
14.2 Reciprocita a transitivita strukturálních vztahů	188
14.3 Strukturální vztahy jako procesy zachovávající stav	194
14.4 Shrnutí.....	194
14.5 Problémy.....	195
<i>Kapitola 15 Omezení účasti a rozvětvení.....</i>	<i>197</i>
15.1 Strukturální a procedurální omezení účasti	197
15.2 Strukturální omezení účasti	198
15.3 Zkratky a vyhrazené fráze	200
15.4 Kardinálnost.....	201
15.5 Omezení účastí na řízení.....	203
15.6 Distributivní zákon strukturálních vztahů	205
15.7 Vidlička, rukojeť a hrot	207
15.8 Sada hrotů	209
15.9 Shrnutí.....	213
15.10 Problémy	214
<i>Kapitola 16 Základní strukturální vztahy</i>	<i>215</i>
16.1 Symboly vztahů a účastníci	215
16.2 Názvy vztahů a věty OPL	216
16.3 Strukturální hierarchie, transitivita, symboly definované uživatelem	217
16.4 Shrnutí.....	218
16.5 Problémy	218
<i>ČÁST III Struktura a chování: Ponoření se do tématu</i>	
<i>Kapitola 17 Agregace – účast.....</i>	<i>221</i>
17.1 Základní pojmy.....	221
17.2 Agregace-účast jako rozdělení	223
17.3 Příklad sémantického webu	224

17.4 Pojmenování agregátu.....	226
17.5 Složené a sdílené agregace v UML a SysML.....	227
17.6 Vyjádření pořadí částí.....	229
17.7 Agregace a označené strukturální vztahy	230
17.8 Neúplná agregace.....	232
17.9 Mini-jazyk parametrizovaných omezení účasti	235
17.10 Shrnutí.....	236
17.11 Problémy.....	237
<i>Kapitola 18 Výstava – charakteristika.....</i>	<i>239</i>
18.1 Charakteristika a vystavovatel	239
18.2 Atribut a operace: dva druhy vlastností	241
18.3 Funkce v UML a SysML vs. OPM.....	242
18.4 Jedinečnost názvů věcí a prvků v OPM.....	243
18.5 Čtyři kombinace věcí a funkcí	244
18.6 Základní strukturální hierarchie.....	248
18.7 Problém pojmenování atributů	249
18.8 Vlastnosti prvků a odkazů	251
18.9 Shrnutí.....	254
18.10 Problémy.....	255
<i>Kapitola 19 Stavý a hodnoty.....</i>	<i>257</i>
19.1 Definice stavu	257
19.2 Potlačení a vyjádření stavu	259
19.3 Hodnota: specializace stavu.....	260
19.4 Přechod stavu: Když je proces aktivní 2.....	61
19.5 Označení cest a flip-flop.....	264
19.6 Model sebeorganizované kritičnosti mozku”	266
19.7 Stavově specifikované štítkované strukturální odkazy.....	268
19.8 Složené stavy a stavový prostor.....	271
19.9 Shrnutí.....	274
19.10 Problémy.....	274
<i>Kapitola 20 Zobecnění a instancování</i>	<i>277</i>
20.1 Zobecnění-specifikace: Úvod.....	277
20.2 Dědičnost	280
20.3 Specializace prostřednictvím rozlišovacího atributu	283
20.4 Odkaz na charakteristiku stanovenou státem	285
20.5 Klasifikace – instancování.....	286
20.6 Relativita instance.....	288
20.7 Omezení hodnot atributů	289

20.8 Případy procesu	290
20.9 Shrnutí.....	292
20.10 Problémy	292
<i>Kapitola 21 Řízení složitosti: Zjemnění a abstrakce.....</i>	<i>295</i>
21.1 Potřeba řízení složitosti.....	295
21.2 Tvzení o složitosti modelu	296
21.3 Rozklad založený na aspektech vs. rozklad založený na úrovni detailů	297
21.4 Kompromis mezi úplností a srozumitelností	298
21.5 Vyjádření stavu a potlačení stavu	299
21.6 Rozvíjení a skládání.....	300
21.7 Rozkládání v diagramu a novém diagramu	301
21.8 Skládání portů	302
21.9 Zvětšování a zmenšování.....	302
21.10 Synchronní vs. asynchronní zdokonalování procesů	305
21.11 Ekvivalence mezi přiblížením a rozbalením.....	306
21.12 Systémová mapa a konečný OPD.....	307
21.13 Strom objektů OPD a les	310
21.14 Zmenšení.....	311
21.15 Zjednodušení OPD.....	313
21.16 Abstrakční účty pro procedurální prioritu odkazů.....	315
21.17 Migrace odkazů při přiblížení.....	318
21.18 Vytváření zobrazení: čtvrtý mechanismus zdokonalení	318
21.19 Middle-Out jako de facto architektonická praxe	319
21.20 Navigace v rámci modelu systému OPM	320
21.21 Shrnutí.....	321
21.22 Problémy	324
<i>Kapitola 22 Provozní sémantika OPM a řídicí odkazy</i>	<i>327</i>
22.1 Mechanismus řízení událost-podmínka-akce.....	327
22.2 Sady objektů předběžných podmínek, předběžného zpracování a následného zpracování.....	328
22.3 Druhy řídicích vazeb	328
22.4 Událostní vazby	329
22.5 Podmínkové odkazy.....	333
22.6 Odkazy na výjimky	340
22.7 Míra transformace	342
22.8 Výpočty s OPM	344
22.9 Sady a iterace	346
22.10 Operační sémantika v kontextech procesů se zoomem	346
22.11 Transformace instancí zapojených objektových množin.....	353
22.12 Jazyk omezení objektů (OCL) v UML	354
22.13 Shrnutí.....	355
22.14 Problémy	356

<i>Kapitola 23 Logické operátory a pravděpodobnosti</i>	357
23.1 Logické a procedurální odkazy	357
23.2 Logické NOT	359
23.3 Logické XOR a OR spojky	360
23.4 Rozbíhající se a sbíhající se XOR a OR propojení	362
23.5 Kombinatorické XOR a kombinatorické OR	365
23.6 Stavově specifikované XOR a OR spojovací vějíře	367
23.7 Více ovládacích odkazů má sémantiku OR	368
23.8 Pravděpodobnosti spojení a pravděpodobnostní fanoušci spojení	370
23.9 Shrnutí	373
23.10 Problémy	373
<i>Kapitola 24 Přehled normy ISO 19450</i>	375
24.1 Úvod do normy ISO 19450	375
24.2 ISO 19450 Termíny, definice a symboly	377
24.3 Zásady a pojmy metodiky objekt-proces	378
24.4 Čtyři přílohy normy ISO 19450	380
Odkazy	387
Zásady OPM v kostce	393
Rejstřík	395

Hlavní symboly OPM v souladu s normou ISO 19450

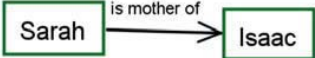
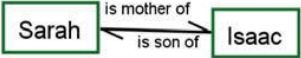
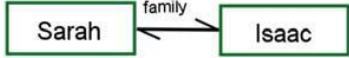
Věci: objekty a procesy se stavem

vlastno st věci	hodnota (označení)	věc		
		objekt	objekt se stavem	proces
podstata	Informatický (plochý)			
	Fyzický (stínova ný)			
příslušnost	Systémo vá (plná)			
	Environmentáln í (čárkovaná)			

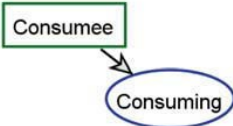
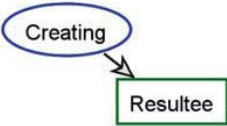
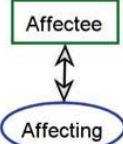
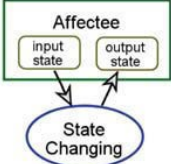
Základní strukturální vazby

modalita	agregace účást	výstava- charakterizace	generalizace- specializace	klasifikace- instancování
grafika – Objekt-proces diagram (OPD)				
Textový – Objektově- procesní jazyk (OPL)	Celek se skládá z částí.	Vystavovatel vystavuje Atributu.	Specializace je Obecná.	Instance je instance třídy.

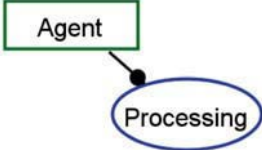
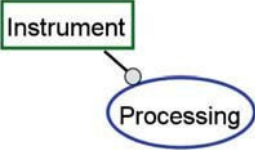
Označené strukturální odkazy

jednosměrný označený odkaz	obousměrný označený odkaz	Vzájemný označený odkaz
 Sarah je matkou Isaaca.	 Sarah je matkou Isaaca. Isaac je syn Sarah.	 Sarah a Isaac jsou rodina.

Procedurální transformační odkazy

spotřební odkaz	výsledkový odkaz	efektový odkaz	pár vstupních a výstupních odkazů
 Spotřeba spotřebovává Spotřebováváný.	 Vytváření výnosů Výsledek.	 Ovlivnění ovlivňuje Affectee.	 Změna stavu mění ovlivněného z vstupního stavu do výstupního stavu.

Procedurální povolení odkazů

agentní propojení	odkaz nástroje
 Agent zpracovává Zpracování.	 Zpracování vyžaduje Nástroj.

Part I

Model-Based Systems Engineering Introduced

Tato kniha se zaměřuje na koncepční modelování systémů pomocí OPM (Object-Process Methodology) a SysML (Systems Modeling Language). SysML je od roku 2006 uznávaným de facto standardem Object Management Group (OMG), zatímco OPM se v roce 2014 stalo veřejně dostupnou specifikací ISO 19450. Teoretické pozadí a diskuse jsou ponechány pro část II a podrobné technické specifikace pro část III. Tato první část představuje OPM a SysML prostřednictvím případové studie automatického systému reakce na nehodu automobilu, který modelujeme krok za krokem a postupně odhalujeme principy a postupy modelování. [Kapitola 1](#) začíná popisem systému, který má být modelován, a počátečním, jednoduchým modelem OPM. V [kapitole 2](#) vylepšujeme model pomocí textu a animované simulace. [Kapitola 3](#) představuje vazby, které propojují prvky v modelu. V [kapitole 4](#) přerušujeme modelování automatického systému reakce na nehodu pomocí OPM a přecházíme k představení a použití prvních tří diagramů SysML: diagramů použití, bloků a stavových strojů. V [kapitole 5](#) pokračujeme v modelování systému pomocí OPM a seznámíme se s přiblížením – nejvýkonnějším mechanismem zpřesnění, který umožňuje zvládat složitost systémů. Dynamický, behaviorální a časově závislý aspekt systému je tématem [kapitoly 6](#). V [kapitole 7](#) se seznámíme se specifiky řízení chování systému. Prohloubení našich znalostí o abstrakci a mechanismech zdokonalování jako prostředcích pro řízení složitosti je tématem [kapitoly 8](#), poslední kapitoly v části I.

Kapitola 1

Jste připraveni začít modelovat?

...všechny modely jsou špatné; praktickou otázkou je, jak špatné musí být, aby nebyly užitečné.

Box a Draper (1987)

U diagramů je význam zřejmý, protože jakmile pochopíte, jak základní prvky diagramů do sebe zapadají, význam vám doslova vyskočí do očí.

Steve Cook (1999)

Žijeme ve světě propojených systémů. Ve skutečnosti je každý z nás jako lidská bytost velmi složitým systémem, který žije v řadě sociálně-politických a technologických systémů, které nejsou o nic méně složité. Abychom mohli složité systémy pochopit a navrhovat, je nutné mít metodiku a jazyk pro vytváření modelů, které dokážou vyjádřit, co tyto systémy dělají, proč to dělají, jak to dělají a co k tomu potřebují. Zatímco vizuální a intuitivní povaha diagramů z nich učinila široce používaný prostředek pro vytváření modelů systémů, text v přirozeném jazyce je také důležitým způsobem, jak sdělovat složité myšlenky. Formální diagramy jsou grafickým jazykem v tom smyslu, že obsahují propojené symboly, které vyjadřují smysluplné skutečnosti a tvrzení o světě. Kombinace grafiky s textem posiluje naši schopnost specifikovat složité myšlenky ve vědě a technice.

1.1 Systém automatické reakce na nehodu ()

Představujeme koncepční modelování pomocí OPM a později SysML na příkladu specifikace systému GM OnStar Automatic Crash Response (ACR). Specifikace, kterou modelujeme níže, byla převzata téměř doslovně z rané verze popisu technologie OnStar na [webových stránkách společnosti OnStar](http://cms.cerritos.edu/auto/basic-its/ost.htm).¹

Bezpečnostní, ochranné a informační služby OnStar ve vozidle využívají satelitní technologii GPS (Global Positioning System) a mobilní technologii k propojení vozidla a řidiče s centrem OnStar. V centru OnStar poskytují poradci personalizovanou pomoc v reálném čase 24 hodin denně, 365 dní v roce. ...

Akcelerometr umístěný v modulu pro snímání a diagnostiku (SDM) měří závažnost nehody. V případě středně těžké až těžké čelní nebo boční srážky jsou data přenášena z postižených senzorů do SDM. Senzor SDM také dokáže identifikovat náraz zezadu dostatečné závažnosti.

¹ <http://cms.cerritos.edu/auto/basic-its/ost.htm>.

Bez ohledu na to, zda se airbagy aktivují, SDM přenáší informace o nehodě do modulu OnStar vozidla.

Během několika sekund po středně těžké až těžké nehodě modul OnStar odešle prostřednictvím mobilního připojení zprávu do call centra OnStar (OCC) a informuje poradce, že došlo k nehodě. Následně se naváže hlasové spojení mezi poradcem a cestujícími ve vozidle. Poradce může poté spojit dispečink tísňové linky 911 nebo veřejnou bezpečnostní službu (PSAP), která určí, zda jsou nutné záchranné služby. Pokud cestující nereagují, poradce může dispečerovi tísňové linky poskytnout informace o nehodě z SDM, které odhalí závažnost nehody. Dispečer může určit, jaké záchranné služby jsou vhodné. Pomocí satelitu globálního pozičního systému (GPS) mohou poradci OnStar sdělit záchranářům polohu vozidla.

Z popisu tohoto systému vyplývá, že systém ACR má za cíl poskytnout automatickou reakci v případě vážné dopravní nehody. V následujících částech metodicky modelujeme tento systém pomocí OPM a poté SysML.

1.2 Princip OPM () jako semínko funkce

Aby bylo možné spustit model OPM systému, je prvním krokem určení *funkce* systému. Funkce je hlavní proces systému, který je navržen tak, aby poskytoval hodnotu – přínos za cenu – příjemci systému. Příjemci systému jsou osoby, které získávají hodnotu z používání systému. Identifikace funkce systému je zásadní, protože vyjadřuje motivaci pro jeho vytvoření. Tato funkce bude nejvyšším procesem našeho modelu OPM. Určení funkce systému není jen důležité a doporučené, ale je také základním principem, známým jako *princip OPM „funkce jako semínko“*:⁽²⁾

Princip OPM „funkce jako semínko“

Modelování systému začíná definováním, pojmenováním a popisem funkce systému, která je zároveň jeho procesem nejvyšší úrovně.

Termín „funkce jako semínko“ podtrhuje ústřední význam zahájení procesu modelování způsobem, který se zaměřuje na *funkci* systému, tj. na hodnotu, kterou systém poskytuje svým uživatelům. Jak ukážou následující kapitoly, tato funkce je semínkem, ze kterého se postupně vyvíjí celý model. Tato zásada může být kontraintuitivní, protože mnoho inženýrů má tendenci začínat *formou* – objekty, látkami, z nichž se systém skládá – spíše než funkcí, která je procesem, díky němuž by příjemci systém vůbec používali. Funkce přináší hodnotu, zatímco forma s sebou nese náklady, které je třeba zaplatit, aby bylo možné dosáhnout funkce systému.

Vzhledem k ústřední roli funkce systému stojí za to se zamyslet nad tím, co tato funkce skutečně je a jak by se měla nejlépe nazývat, aby všichni, kdo se podílejí na modelování, byli na stejné vlně. Vhodná funkce objasňuje a zdůrazňuje ústřední cíl modelovaného systému.

² Toto je první z 13 principů OPM, které jsou uvedeny v rámečku v celé knize a také se objevují na konci knihy pro rychlou referenci po [kapitole 24](#) pod nadpisem „Principy OPM v kostce“.

Diskuse o funkci často vyvolává v této rané fázi debatu mezi členy týmu systémové architektury, ale je velmi cenná. Takové diskuse často odhalují rozdíly a často i mylné představy účastníků ohledně systému, který se chystají navrhnout, modelovat a konstruovat. Shoda na funkci systému a jeho nejvhodnějším názvu tedy zvyšuje pravděpodobnost, že výsledkem bude užitečný model.

1.3 Identifikace funkce systému „“

Z výše uvedeného popisu systému OnStar je zřejmé, že hlavní funkcí systému – jeho účelem a hodnotou, kterou přináší – je automatická reakce v případě dopravní nehody. Proto tuto funkci nazýváme **automatická reakce na nehodu** a jedná se o nejvyšší úroveň procesu systému, který se chystáme modelovat. OPM má pouze jeden typ diagramu, který se nazývá objektově-procesní diagram (OPD). Každý OPD je sestaven pomocí dvou stavebních bloků OPM: objektů a procesů.

Objekt je věc, která existuje nebo může existovat.

Zatímco objekty existují, procesy se odehrávají nebo nastávají a transformují objekty tím, že je generují, spotřebovávají nebo ovlivňují.

Proces je něco, co transformuje objekt.

Objekty a procesy se souhrnně nazývají věci.

Věc je objekt nebo proces.

Začneme modelováním *systémového diagramu* – *diagramu objektů a procesů* nejvyšší úrovně (OPD) – v našem modelu OPM. Symbol OPM pro proces je elipsa, ve které je zaznamenán název procesu.



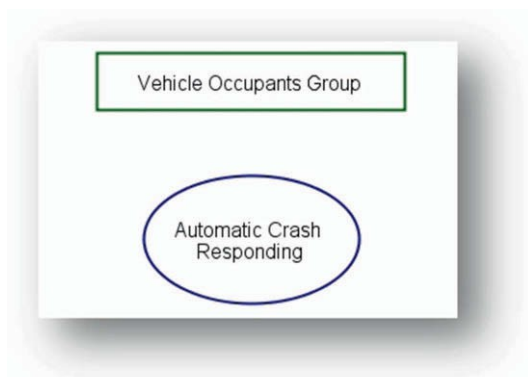
Obr. 1.1 **Automatická reakce na nehodu** modelovaná jako proces

Obrázek 1.1 popisuje proces **automatické reakce na nehodu** v notaci OPM pomocí OPCAT,³ modelovacího softwarového prostředí založeného na OPM, jako je OPCAT (Dori et al. 2003). Důrazně doporučujeme, aby si čtenář nainstaloval OPCAT a postupoval podle zde uvedených modelovacích aktivit.

Na základě definice procesu jako věci, která transformuje objekt, nemá žádný proces smysl, pokud netransformuje alespoň jeden objekt. Tento objekt je známý jako *transformovaný* objekt transformačního *procesu* nebo *operand* funkce systému.

1.4 Identifikace příjemce systému ()

Umělý systém vytvořený člověkem je navržen tak, aby přinášel prospěch alespoň některým ze svých *zainteresovaných stran*. Zainteresované strany, které mají prospěch, jsou *příjemci* systému. Příjemcem procesu **automatické reakce na nehodu**, který je v našem případě také transformovaným subjektem, je řidič a všichni další cestující, kteří se nacházejí v havarovaném vozidle. Tato skupina lidí je objektem „**Skupina cestujících ve vozidle**“. **Obrázek 1.2** ukazuje OPD z **obrázku 1.1** aktualizované o tento objekt. Symbol OPM pro objekt je obdélník s názvem objektu zaznamenaným uvnitř. Jedná se také o standardní symbol používaný v UML – Unified Modeling Language (OMG UML 2011I, 2011S) – a SysML, kde se označuje jako *blok*.



Obr. 1.2 **Skupina cestujících ve vozidle** je přidána jako objekt do procesu **automatické reakce na nehodu**

1.5 Proces transformuje objekt „ „

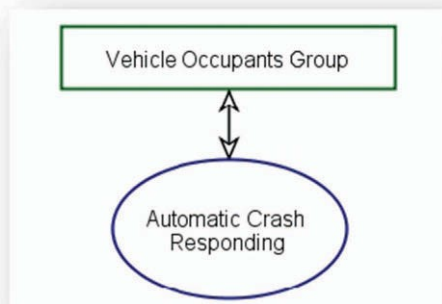
Objekt jsme definovali jako věc, která existuje nebo může existovat. Náš objekt, **skupina cestujících ve vozidle**, existuje, stejně jako před spuštěním procesu **automatické reakce na nehodu**. Co tedy

³ Objektově-procesní diagramy (OPD) v této knize byly nakresleny pomocí OPCAT, softwarového prostředí, které umožňuje modelování založené na OPM. OPCAT lze zdarma stáhnout a nainstalovat z webové stránky <http://esml.iem.technion.ac.il/>, která také obsahuje praktický tutoriál k OPCAT a mnoho článků o OPM. Tutoriál k OPCAT je také k dispozici na této stránce.

transformaci prochází **skupina cestujících ve vozidle**? Abychom mohli odpovědět na tuto otázku, podíváme se na následující definici transformace.

***Transformace** je vytvoření (generování, konstrukce) nebo spotřeba (eliminace, zničení) objektu nebo účinku (změna stavu) existujícího objektu.*

V našem případě se stav **skupiny cestujících ve vozidle** jasně změnil. Jinými slovy, **skupina cestujících ve vozidle** byla ovlivněna a následně měla prospěch z výskytu procesu **automatické reakce na nehodu**. Abychom vyjádřili skutečnost, že proces **automatické reakce na nehodu** ovlivňuje (mění stav) objektu **skupiny cestujících ve vozidle**, vložíme mezi proces a objekt odkaz. Propojení, znázorněné na [obr. 1.3](#), je propojení *účinku* – obousměrná šipka, $\leftrightarrow$ – mezi ovlivňujícím (stav měnícím) procesem a ovlivněným objektem, tj. objektem, jehož stav se změnil v důsledku výskytu procesu.



Obr. 1.3 Mezi procesem **automatické reakce na nehodu** a objektem **skupina cestujících ve vozidle** je přidáno efektové propojení, které označuje, že proces ovlivnil (změnil stav) objektu.

Náš model v současné době obsahuje tři prvky. Prvním je proces **automatické reakce na nehodu**, druhým je objekt **skupina cestujících ve vozidle** a třetím je spojení mezi procesem a objektem.

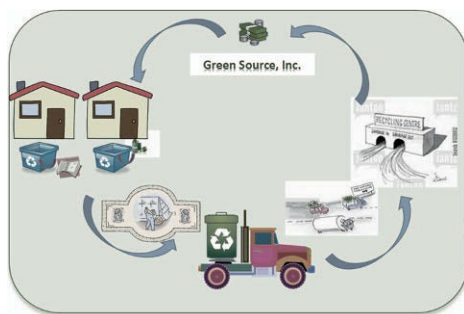
1.6 Shrnutí

- Začali jsme modelovat systém automatické reakce na nehodu pomocí OPM.
- OPM má jediný typ diagramu: diagram objekt-proces (OPD).
- OPM je tvořen objekty, které existují, a procesy, které objekty transformují.
- Transformace objektů je tvorba objektů, spotřeba objektů nebo změna objektů.

- Procesy – v našem příkladu **automatická reakce na havárii** – považujeme za samostatné stavební bloky OPM, které jsou oddělené od objektů.
- Objekty a procesy umožňují souběžné modelování struktury a chování systému ve stejném OPD.
- Transformace je vytvoření objektu, spotřeba objektu nebo změna stavu.

1.7 Problémy

Student technické univerzity dostal za úkol nakreslit grafické znázornění systému recyklace odpadu a vytvořil náčrt na [obr. 1.4](#).



Obr. 1.4 Recyklační systém – grafické znázornění

1. Které prvky ve skice představují objekty?
2. Které prvky ve skice představují procesy?
3. Které prvky ve skice představují vztahy?
4. Existují v náčrtu pojmy, které nespádají do žádné z výše uvedených kategorií? Pokud ano, jaké jsou to pojmy? Jak by se měly nazývat?

Případová studie systému manipulace se zavazadly, kterou začneme rozvíjet níže, bude sloužit jako základ pro problémy na konci každé kapitoly. Modelování můžete provést ručně, ale důrazně doporučujeme použít softwarový balík pro modelování OPM, například OPCAT (ke stažení na <http://esml.iem.technion.ac.il/>).

Cestující přilétající na letiště odevzdá svá zavazadla letecké společnosti, se kterou letí. Systém pro manipulaci se zavazadly řídí přepravu zavazadel do cíle cesty cestujícího.

5. Jaká je funkce systému? Vyjádřete ji jako název procesu OPM.
6. Nakreslete funkci jako hlavní proces v novém OPD.
7. Identifikujte hlavního příjemce systému.

8. Přidejte příjemce do OPD jako objekt a propojte jej s procesem definovaným jako funkce systému.
9. Identifikujte operand funkce systému.
10. Přidejte operand do OPD a propojte jej s procesem.
Identifikujte další hlavní objekty, na které má proces vliv, přidejte je do OPD a propojte je s procesem.

Kapitola 2

Vylepšení textu a simulace

Poté jsme se vydali do jazykové školy. ... Prvním projektem bylo zkrátit diskurz tím, že se víceslabičná slova zkrátila na jedno a vynechala se slovesa a přičestí, protože ve skutečnosti jsou všechny představitelné věci pouze podstatná jména. ... Mnoho z nejvzdělanějších a nejmoudřejších však lpí na novém způsobu vyjadřování pomocí věcí.

Jonathan Swift, *Gulliverovy cesty* (1726)

Winograd a Flores (1987) poznamenali, že „nic neexistuje mimo jazyk... Když řekneme, že nějaká „věc“ existuje (nebo že má nějakou vlastnost), přenesli jsme ji do oblasti artikulovaných objektů a vlastností, které existují v jazyce.“

Jazyk skutečně výrazně zlepšuje naši schopnost rozumět systémům a sdělovat naše porozumění ostatním. Tato kapitola představuje dvě vylepšení modelů OPM: textové znázornění modelu a animovanou simulaci modelu.

Představujeme jazyk objekt-proces (OPL) jako textovou modalitu OPM, která doplňuje grafické znázornění prostřednictvím OPD (diagramů objekt-proces). Ukazujeme ekvivalenci mezi touto grafickou specifikací a specifikací v přirozeném jazyce prostřednictvím OPL. Kapitola také ukazuje další důležitý prostředek pro zlepšení porozumění modelu: jeho animovanou simulaci.

2.1 OPL: podmnožina angličtiny ()

Na [obr. 2.1](#) byla přidána efektová vazba mezi procesem **automatické reakce na nehodu** a objektem skupina **cestujících ve vozidle**. Tato vazba vyjadřuje skutečnost, že proces ovlivnil objekt, tj. nějakým způsobem jej změnil. Jakmile je efektová vazba nakreslena, OPCAT (softwarové prostředí pro modelování OPM, které používáme) automaticky vygeneruje první větu v objektově-procesním jazyce (OPL):

Automatická reakce na nehodu ovlivňuje skupinu cestujících ve vozidle.

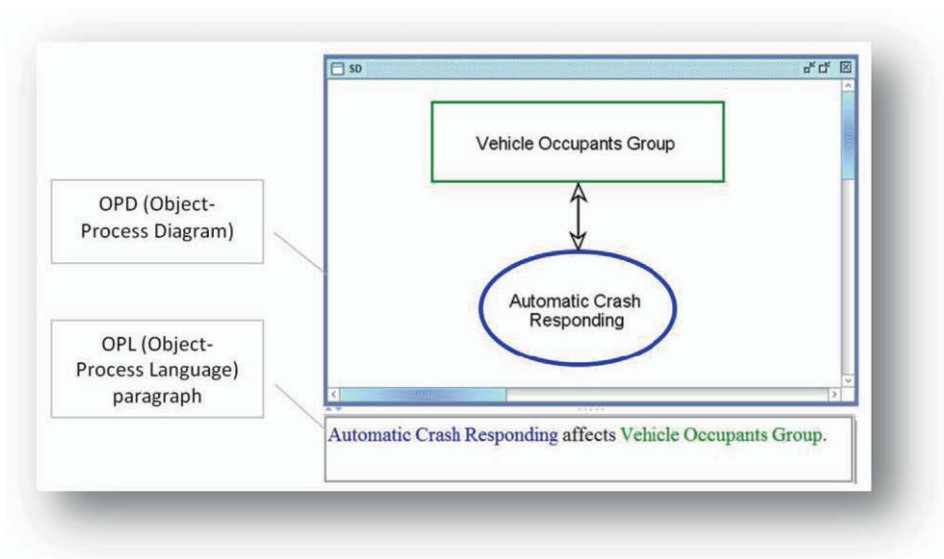
Jedná se o jasnou a jednoznačnou větu v běžné angličtině, která byla generována okamžitě v reakci na propojení těchto dvou věcí modelářem a vysvětluje, co OPCAT z této operace „pochopil“.

[Obrázek 2.1](#) ukazuje částečný snímek obrazovky OPCAT. Horní panel obsahuje OPD a spodní panel obsahuje větu OPL generovanou v reakci na vložení efektového odkazu modelářem.

V OPCAT je výchozí barva slov nebo frází (slovních sekvencí) OPL, které představují objekty, zelená; je to stejná barva jako odpovídající objekty v OPD. Stejně tak je výchozí barva frází představujících názvy procesů v OPL modrá, což je také barva odpovídajících procesních elips.

Převod grafiky na text je obousměrný proces: stejně jako bylo možné extrahovat výše uvedený odstavec OPL z OPD na [obr. 2.1](#), lze OPD rekonstruovat z jeho odstavce OPL, tj. ze souboru vět OPL, které v textu specifikují to, co OPD specifikuje v grafice. Jako cvičení lze použít

princip ekvivalence grafiky a textu lze ověřit rekonstrukcí [obr. 2.1](#) versa. Je snazší upravit grafiku a získat okamžitou textovou zpětnou vazbu.



Obr. 2.1 Snímek obrazovky OPCAT zobrazující OPD nahoře a první větu v objektově-procesním jazyce (OPL) generovanou v reakci na vložení efektového odkazu modelářem.

Schopnost přepínat mezi grafikou a textem znamená, že autoři specifikací a modeláři systému OPM jsou méně náchylní k nákladným chybám v návrhu. Navíc čtenáři textově specifikovaného modelu mají větší šanci plně porozumět systému a odhalit chyby nebo opomenutí. Čtenář specifikace může vyplnit mezery ve svém porozumění systému, které mohly vzniknout při zkoumání jedné modalit, tím, že se podívá na druhou. Tím si čtenář upevní znalost specifikace a může snáze odhalit chyby nebo opomenutí v návrhu.

2.2 Stavy a animovaná simulace OPM

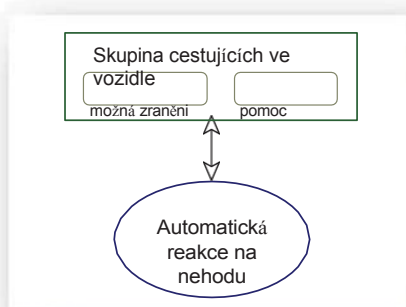
Stavy jsou důležitými entitami v koncepčním modelu. V této části představíme stavy a ukážeme, jak lze model OPM simulovat pomocí animace, aby se projevila jeho dynamika, zejména přechody mezi stavy. Dosud jsme modelovali skutečnost, že objekt **Skupina cestujících ve vozidle** byl ovlivněn procesem **Automatická reakce na nehodu**. Abychom identifikovali skutečnou povahu tohoto vlivu, musíme být konkrétnější. Za tímto účelem může být v kterémkoli okamžiku, kdy zkoumáme určitý objekt, tento objekt v určité situaci, kterou označujeme jako *stav*.

Stav je situace nebo pozice, ve které se objekt může nacházet, nebo hodnota, kterou může nabývat, po určitou dobu.

Jelikož stavy jsou možné situace nebo hodnoty *objektu*, nemají vlastní „život“; mají význam pouze v kontextu objektu, ke kterému patří a ve kterém se nacházejí.

2.2.1 Vliv procesu na objekt

Účinek, který má proces na objekt, je *změna stavu* tohoto objektu. Jinými slovy, proces transformuje objekt z jeho *vstupního stavu* (stav před procesem) do jeho *výstupního stavu* (stav po procesu). Proto je v daném okamžiku objekt se stavem v *jednom ze svých stavů* nebo v *přechodu mezi dvěma svými stavy*.



Obr. 2.2 Přidány jsou stavy skupiny „Osoby ve vozidle“: „možné zranění“ je vstupní stav a „poskytnutí pomoci“ je výstupní stav

Abychom mohli přesně popsat vliv procesu **automatické reakce na nehodu** na objekt „**Skupina cestujících ve vozidle**“, musíme nejprve vymyslet vhodné názvy pro stavy tohoto objektu před a po výskytu procesu. Bezprostředně po nehodě je **skupina cestujících ve vozidle možná zraněná**; po **automatické reakci na nehodu** je jim **poskytována pomoc**. Jedná se o odpovídající vstupní a výstupní stavy objektu **skupiny cestujících ve vozidle** ve vztahu k procesu **automatické reakce na nehodu**.

Obrázek 2.2 ukazuje OPD po přidání dvou stavů **skupiny cestujících ve vozidle**, přičemž **možný úraz** je vstupním stavem a **poskytnutí pomoci** je výstupním stavem. Symbol stavu je obdélník se zaoblenými rohy – zkráceně routangle¹ – který obklopuje název stavu, který začíná malým písmenem. Symbol stavu je nakreslen uvnitř obdélníku objektu, který „vlastní“ stav. Toto přidání stavů vytvořilo následující novou větu OPL:

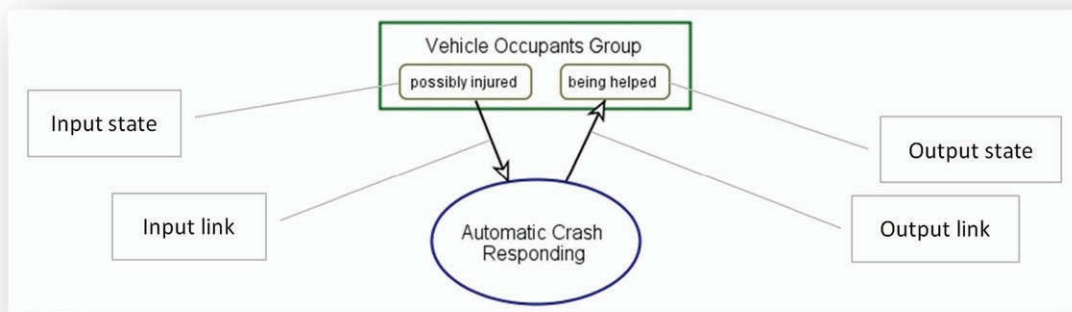
Skupina cestujících ve vozidle může být možná zraněná nebo jí může být poskytnuta pomoc.

¹ Tento termín vytvořil D. Harel.

Věty OPL v této knize jsou napsány **písmem Arial**. Vyhrazené fráze, jako například „může být“, jsou napsány běžným písmem, zatímco vše ostatní (včetně čárek a teček) je napsáno tučným písmem.

2.2.2 Od implicitního účinku k explicitní změně stavu

Po specifikaci dvou stavů objektu **Skupina cestujících ve vozidle** můžeme nyní být konkrétnější ohledně účinku – změny stavu –, který vyvolal výskyt procesu **Automatická reakce na nehodu**. To provedeme tak, že v **obr. 2.3** nahradíme jedinou obousměrnou vazbu účinku párem vstupně-výstupních vazeb (nazývaným také vstupně-výstupní specifikovaná vazba účinku), což je pár jednosměrných šipek, jedna ze vstupního stavu do procesu a druhá z procesu do výstupního stavu.



Obr. 2.3 Efektivé propojení na **obr. 2.2** je nahrazeno párem vstupně-výstupních propojení, která vysvětlují změnu stavu skupiny

skupiny cestujících ve vozidle ze stavu vstupu „možné zranění“ do stavu výstupu „poskytnutí pomoci“

Šipka ze vstupního stavu k procesu je *vstupní spojení* a šipka z procesu ke výstupnímu stavu je *výstupní spojení*. Společně tvoří tato dvě spojení *pár vstupně-výstupních spojení*. V reakci na tuto úpravu modelu je předchozí věta OPL „Automatická reakce na nehodu ovlivňuje skupinu cestujících ve vozidle“ nahrazena následující větou OPL:

Automatická reakce na nehodu mění skupinu cestujících ve vozidle z možného zranění na poskytnutí pomoci.

Díky specifikaci ovlivňujícího procesu a ovlivněného objektu s jeho vstupními a výstupními stavy tato věta přesně odráží podrobnosti dynamiky systému.

2.2.3 Pojmenování stavů

Názvy stavů musí odrážet různé relevantní situace, ve kterých se jejich „vlastníci“ objekt může v daném okamžiku nacházet. Například **vypnutý**, **pohotovostní** a **zapnutý** jsou tři možné stavy stroje, což by vedlo k následující větě OPL:

Stroj může být vypnutý, v pohotovostním režimu nebo zapnutý.

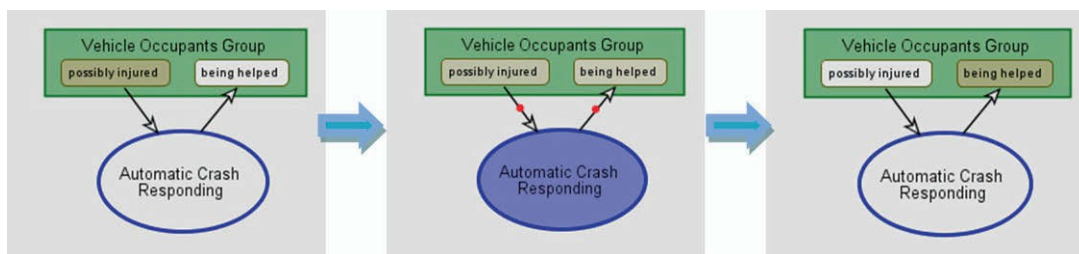
Jak již bylo uvedeno, na rozdíl od názvů věcí (objektů nebo procesů), které začínají velkým písmenem, se názvy států vždy píší malými písmeny. Je třeba uvést popisnější názvy,

jako „možná zraněný“ nebo „pomáháno“, jako v našem případovém studiu. Je důležité zkontrolovat, zda výsledná věta OPL dává smysl.

2.3 Animované provedení modelu OPM ()

Jednou z nejatraktivnějších a nejužitečnějších vlastností modelu OPM, která umožňuje jeho vizualizaci a testování, je jeho proveditelnost, tj. schopnost simulovat systém provedením jeho modelu prostřednictvím animace v řádně navrženém softwarovém prostředí.²

Obrázek 2.4 ukazuje tři fáze provádění modelu OPM na **obr. 2.3**. Snímek obrazovky na levé straně ukazuje systém před spuštěním procesu **automatické reakce na nehodu**. V této fázi je **skupina cestujících ve vozidle** ve vstupním stavu, **pravděpodobně zraněná**, což je označeno stavem, který je pevný (hnědá barva).



Obr. 2.4 Spuštění modelu OPM zobrazeného v OPD na **obr. 2.3**. Vlevo: systém před spuštěním procesu automatické reakce na nehodu. Uprostřed: proces v akci; objekt přechází ze vstupního stavu do výstupního stavu. Vpravo: systém po ukončení procesu automatické reakce na nehodu

Snímek obrazovky uprostřed **obr. 2.4** ukazuje proces v akci, označený jako plný (modrá barva). Během doby, kdy je proces **automatické reakce na nehodu** aktivní (tj. když se provádí), objekt **Skupina cestujících ve vozidle** přechází ze svého vstupního stavu, **kdy jsou cestující pravděpodobně zraněni**, do výstupního stavu, **kdy jim je poskytována pomoc**. To je označeno tím, že oba stavy jsou poloprůhledné (světle hnědé).

Při sledování animace v akci je patrné, že vstupní stav postupně mizí, zatímco výstupní stav se stává plným. Současně se dvě červené tečky, zobrazené uprostřed obou šipek, pohybují podél páru vstupně-výstupních spojů a označují „řízení“ systému, tj. kde se systém v daném časovém okamžiku nachází. Jedna červená tečka se pohybuje ze vstupního stavu k ovlivňujícímu procesu. Současně se druhá tečka pohybuje od procesu podél výstupního spojení k výstupnímu stavu. Nakonec snímek obrazovky vpravo ukazuje systém po ukončení procesu **automatické reakce na nehodu**. V této fázi je **skupina cestujících ve vozidle** ve výstupním stavu, **kdy jim je poskytována pomoc**.

Animované provedení modelu systému má několik výhod. Zprv se jedná o dynamickou vizualizační pomůcku, která pomáhá jak modeláři, tak cílové skupině sledovat a porozumět chování systému v čase. Zadruhé, podobně jako debugger programovacího jazyka, usnadňuje ověření dynamiky systému a odhalení logických chyb v jeho toku řízení.

² Pokud jste modelovali systém pomocí OPCAT, můžete si to vyzkoušet sami: Klikněte na ikonu filmu „Testovat systém“.

Proto důrazně doporučujeme, aby byl systémový model během jeho vytváření často animován, aby se chyby v návrhu nehromadily, ale byly opravovány ihned po jejich vzniku.

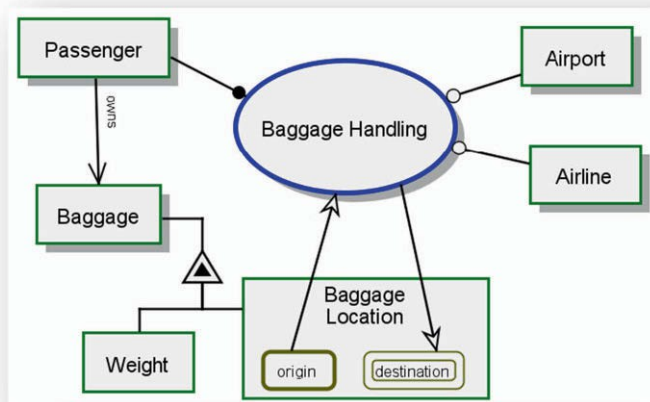
2.4 Shrnutí

- OPM má dvě ekvivalentní modalities reprezentace: grafický diagram objekt-proces (OPD) a textový jazyk objekt-proces (OPL).
- Věty OPL a OPD se vzájemně doplňují, protože apelují na paralelní vizuální a verbální kognitivní procesy v lidském mozku.
- Stav je situace, ve které se objekt může nacházet.
- Efektové propojení označuje určitou změnu stavu propojeného objektu propojeným procesem.
- Pár vstup-výstup naznačuje konkrétní stav, ze kterého a do kterého proces mění objekt.
- Model OPM je vhodný pro animovanou simulaci, která usnadňuje pochopení dynamického aspektu systému a testování jeho logického toku.

2.5 Problémy

Pokračujeme s modelem systému manipulace se zavazadly, který jste vytvořili v [kapitole 1](#), a předpokládáme, že aktuální model je znázorněn na [obr. 2.5](#).

1. Přiřaďte odpovídající větu OPL ke každému článku v OPD systému pro manipulaci se zavazadly na [obr. 2.5](#).
2. Přidejte stavy „vyloženo“ a „naloženo“ k objektu „Zavazadla“.
3. Nahraďte efektové propojení mezi procesem ve vašem modelu a **zavazadly** párem vstupně-výstupních propojení.



Obr. 2.5 Systémový diagram – diagram objekt-proces (OPD) nejvyšší úrovně systému pro manipulaci se zavazadly

4. Aktualizujte větu OPL tak, aby odrážela provedenou změnu.
5. Přidejte do svého modelu a propojte zbývající objekty podle výše uvedené specifikace.
6. Pomocí OPCAT simulujte systém jeho animací.

Kapitola 3

Propojování věcí pomocí spojovacích slov

... Spojovací slova nebo spojovací fráze jsou sada slov používaných ke spojení pojmů za účelem vyjádření vztahů mezi dvěma pojmy. ... Výběr vhodných spojovacích slov pro jasné vyjádření vztahu mezi dvěma pojmy je pravděpodobně nejobtížnějším úkolem při vytváření pojmových map.

Alberto J. Cañas, cmap.ihmc.us/docs/linkingwords.html (vyhledáno v roce 2014)

Odkazy jsou grafickým vyjádřením vztahů mezi věcmi. Odkazy OPM spojují procesy s objekty nebo jejich stavy a dodávají význam vztahům mezi nimi. Tato kapitola rozšiřuje použití odkazů v našem modelu a vysvětluje sémantiku různých druhů odkazů.

3.1 Procedurální odkazy versus strukturální odkazy

Odkazy, které jsme dosud používali – odkaz na efekt a odkazy na vstup a výstup – jsou procedurální odkazy.

Procedurální odkaz je odkaz, který specifikuje dynamický aspekt systému propojením objektu (nebo jednoho z jeho stavů) a procesu.

Procedurální vazby mohou být *transformační* nebo *umožňující*. Transformační vazby vyjadřují transformaci – generování, spotřebu nebo změnu stavu – objektu procesem, ke kterému je připojen. Umožňující vazby vyjadřují umožnění. Propojují proces s *umožňovačem* – objektem, který umožňuje výskyt tohoto procesu, ale není tímto procesem transformován.

Strukturální vazby modelují strukturu systému tím, že vyjadřují dlouhodobé vztahy mezi věcmi v modelu. Strukturální vazby zahrnují agregaci-účást (celek-část), generalizaci-specializaci a další dlouhodobé vztahy.

Strukturální propojení je propojení, které specifikuje statický aspekt systému tím, že propojuje objekt s jiným objektem nebo proces s jiným procesem.

3.2 Přidání podpurných prvků

Nejvyšší úroveň OPD, kterou jsme modelovali, se nazývá *systémový diagram (SD)*. SD, často nazývaný kontextový diagram, poskytuje „pohled z výšky 50 000 stop“ na systém. Umožňuje modeláři a všem zainteresovaným stranám, které mají zájem porozumět systému prostřednictvím jeho modelu OPM, rychle pochopit funkci – hlavní proces systému, kterým je v tomto případě **automatická reakce na nehodu**. SD také ukazuje příjemce.

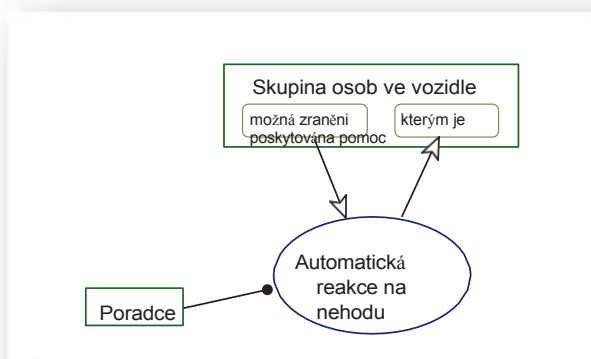
V našem případě je **skupina cestujících ve vozidle** také *operandem* systému; je to hlavní objekt transformovaný funkcí systému, s jeho vstupními a výstupními stavy.

Ještě jsme neskončili; funkce a operand jsou sice důležité, ale ani na této nejabstraktnější úrovni neposkytují úplný obraz systému. Kromě příjemce by měly být představeny také objekty, které tuto funkci *umožňují*. Mezi tyto *umožňující objekty* patří lidské i nelidské objekty, které se v OPM označují jako *agenti a nástroje*.

3.2.1 Přidání agenta a odkazu na agenta

Při čtení popisu našeho systému si všimneme, že poradkyně je významným lidským hráčem v našem systému, proto bychom ji rádi modelovali. Člověk, jako součást systému OPM, je objekt označovaný jako *agent*.

Agent procesu je člověk nebo skupina lidí, kteří interagují se systémem, aby tento proces umožnili a/nebo řídili, ale nejsou jím transformováni.



Obr. 3.1 **Poradce** je přidán jako agent **automatické reakce na nehodu** pomocí odkazu agenta z **Poradce** na **Automatic Crash Responding**. Volitelná figurka také naznačuje, že objekt je člověk

Propojení agentů, —●—, znázorněné na [obr. 3.1](#), je „černá lízátka“ – spojovací čára začínající u objektu a končící černým kruhem na konci procesu. Tento symbol označuje, že objekt propojený s procesem je člověk, jehož přítomnost je pro uskutečnění procesu nezbytná.

Odkaz na agenta naznačuje, že v procesu je „zapojen člověk“, což obvykle znamená, že je nutné rozhraní mezi člověkem – agentem – a systémem, aby agent mohl se systémem komunikovat. [Obr. 3.1](#) ukazuje **poradce** přidáného jako agenta do procesu **automatické reakce na nehodu**. Proces nemůže být spuštěn ani udržován bez agenta, ale proces agenta nemění: nevytváří ho ani nespotřebává, ani nemění jeho stav. Agent je tedy lidským *aktivátorem* procesu. Věta OPL, která je generována jako výsledek přidání odkazu agenta, je:

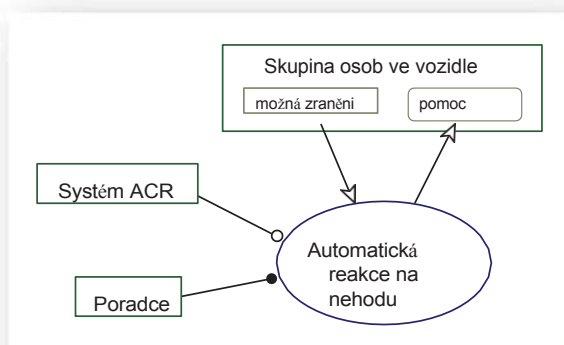
Poradce zpracovává **automatickou reakci na nehodu**.

Vyhrazené slovo OPL označuje potřebu agenta k umožnění procesu. Objekt „**Skupina cestujících ve vozidle**“ je sice skupinou lidí, ale nejde o agenta. Tito lidé jsou příjemci a operandy systému – mění se a jsou tedy transformováni funkcí systému a těží z ní.

3.2.2 Přidání nástroje a propojení nástrojů

Zatímco **Advisor** je agent (lidský aktivátor), náš systém má také neživý aktivátor – *nástroj*. Nástrojem, který umožňuje funkci systému, **Automatic Crash Responding**, je systém automatické reakce na nehodu (ACR), který budeme nazývat **systém ACR**. Nástroj je označen propojením nástroje – bílou lízátkovou tyčinkou,—○. Podobně jako propojení agenta je propojení nástroje čarou, která spojuje objekt s procesem, který tento nástroj vyžaduje. Stejně jako agent je nástroj potřebný pro uskutečnění procesu, ale není transformován (vytvořen, spotřebován nebo ovlivněn) výskytem tohoto procesu.

Nástroj procesu je nehumánní subjekt, který interaguje se systémem, aby umožnil a/nebo řídil tento proces, ale není jím transformován.



Obr. 3.2 **Systém ACR** je přidán jako nástroj **automatické reakce na nehodu** pomocí propojení nástroje ze **systému ACR** se **skupinou osob ve vozidle**.

Obrázek 3.2 ukazuje nástroj **ACR System** přidáný pomocí propojení nástroje z **Advisor** do **Vehicle Occupants Group**. Propojení nástroje je čára končící prázdným kruhem na konci procesu, což znamená, že objekt na počátku propojení je nástrojem pro tento proces. Nástroj je neživý, nehumánní prostředek umožňující proces; jinými slovy, proces nemůže být zahájen ani probíhat bez existence a dostupnosti nástroje po celou dobu trvání procesu. Stejně jako agent se ani nástroj v důsledku průběhu procesu nemění.

Věta OPL, kterou OPCAT vygeneroval v důsledku přidání propojení nástroje, zní:

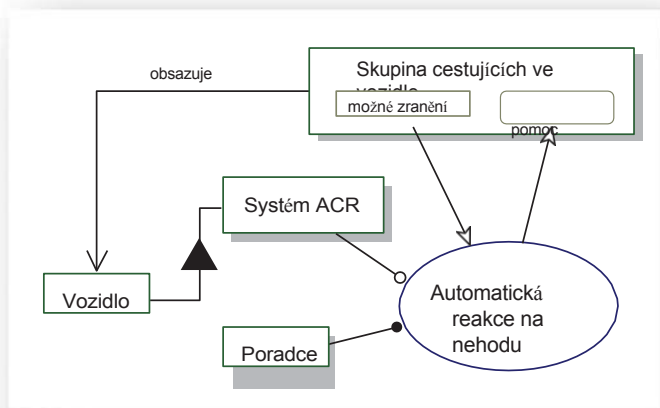
Automatická reakce na nehodu vyžaduje systém ACR.

Vyhrazené slovo OPL vyžaduje označuje potřebu nástroje, který umožní tento proces.

Enabler je agent nebo nástroj.

3.3 Přidání strukturálních odkazů typu „“

V této fázi modelování systému jsme již namodelovali část systémového diagramu (SD), což je OPD nejvyšší úrovně. Další věc (objekt nebo proces), kterou bychom měli zahrnout do SD, je **vozidlo**, protože se jedná o objekt, který obsazují řidič a cestující – **skupina osob ve vozidle** – a je také součástí **systému ACR**.



Obr. 3.3 **Vozidlo** je přidáno jako součást **systému ACR** a označený strukturální odkaz ze **skupiny „Osoby ve vozidle“** na **Vozidlo** je přidán se značkou „obsazuje“.

Obrázek 3.3 ukazuje naše SD s přidáním **vozidlem** a propojením s těmito dvěma objekty dvěma odkazy. Tyto dva odkazy se nazývají strukturální odkazy. První je agregační-participační odkaz, od **systému ACR** k **vozidlu**. Druhý odkaz je označený strukturální odkaz od **skupiny cestujících ve vozidle** k **vozidlu**. Dále budeme diskutovat každý z těchto strukturálních odkazů.

Vozidlo je propojeno agregačně-participačním (celek-část) odkazem jako součást **systému ACR**. Symbol agregačně-participačního odkazu je ▲, plný rovnostranný trojúhelník s vrcholem směřujícím nahoru a propojeným s celkem a základnou propojenou s částí nebo částmi. Tento grafický agregačně-participační odkaz je textově vyjádřen následující větou OPL:

Systém ACR se skládá z vozidla.

Rezervovaná fráze OPL sestává z označuje relaci agregace-účasť, přičemž celek (v našem případě **systém ACR**) předchází a část (v našem případě **vozidlo**) nebo části následují.

Vozidlo je spojeno se **skupinou osob ve vozidle** prostřednictvím druhého typu strukturálního spojení – označeného strukturálního spojení. Označené strukturální spojení je otevřená šipka, která směřuje z jednoho objektu na druhý. Označení je „uživatelé definovaná“ fráze – fráze, která je definována modelářem a zaznamenána podél spojení a vyjadřuje povahu strukturálního vztahu mezi dvěma propojenými objekty (nebo procesy). V našem modelu je značkou spojení „**occupies**“ (**obsazuje**). Je tučně vyznačena, protože je definována modelářem a nejedná se o frázi vyhrazenou pro OPL. Přidání značkování strukturálního spojení spustí generování následující věty OPL:

Skupina cestujících ve vozidle obsazuje vozidlo.

Značky ve značených strukturálních odkazech poskytují modeláři možnost vyjádřit sémantiku jakéhokoli strukturálního vztahu mezi libovolnými dvěma objekty nebo libovolnými dvěma procesy v systému. Jak ukazuje výše uvedená věta OPL, štítkované strukturální propojení vede k vytvoření věty OPL, ve které se nejprve objeví název objektu připojeného ke zdroji šipky propojení (v našem případě **Skupina cestujících ve vozidle**), následovaný názvem štítku (**occupies**) a názvem objektu připojeného k cíli šipky propojení (**Vozidlo**).

3.4 Fyzické versus informatické věci

Věci (objekty nebo procesy) se podle svého *podstatného* atributu dělí na dva druhy: *fyzické* věci a *informatické* věci. Všechny objekty v našem modelu byly dosud fyzické, jak [je](#) znázorněno na [obr. 3.3](#) stínem za každým objektem.

Výchozí hodnotu podstaty věci může určit modelář systému. Pokud se jedná o informační systém, je logické nastavit výchozí hodnotu podstaty věci jako informační, protože většina věcí v takovém systému bude informační. V tomto případě, pokud je věc informační a je již zmíněna alespoň v jedné větě OPL, není třeba žádná další věta OPL, aby bylo naznačeno, že tato věc je informační. Pokud je však věc fyzická, je to označeno ve speciální větě OPL. Například za předpokladu, že systém, který modelujeme, byl nastaven s informační hodnotou podstaty, byla přidána níže uvedená věta OPL, aby bylo naznačeno, že podstata **vozidla** je fyzická.

Vozidlo je fyzické.

Pokud jde o **automatickou reakci na nehodu**, která je naším hlavním procesem a funkcí systému, lze v tomto okamžiku říci, že je informatická, protože zahrnuje pouze předání informace, že vozidlo bylo zapojeno do nehody a že následně došlo k volání o pomoc pro jeho cestující. Skutečný proces pomoci, který je fyzický, je mimo rozsah tohoto systému. Podstatu procesu **automatické reakce na nehodu** lze později změnit na fyzickou, pokud si uvědomíme, že zahrnuje jeden nebo více fyzických dílčích procesů.

3.5 Modelové fakty a odstavce OPL ()

Jak jsme viděli, pokaždé, když jsme zavedli spojení mezi dvěma věcmi nebo změnili podstatu věci z informační na fyzickou, byla přidána nebo upravena alespoň jedna věta OPL. Při modelování se tedy fakta začnou hromadit a vyjadřovat v modelu.

Modelový fakt je vztah mezi věcmi nebo stavy v modelu.

Postupně jsme shromažďovali věty OPL, které společně tvoří *odstavec OPL* a textovou modalitu. Odstavec OPL popisuje srozumitelnou angličtinou přesně to, co grafická modalita OPD popisuje vizuálně.

V současné době zní odstavec OPL následovně.

Skupina osob ve vozidle je fyzická.

Skupina osob ve vozidle může být zraněna nebo jí může být poskytována pomoc. **Skupina osob ve vozidle** obsazuje vozidlo.

Poradce je fyzický.

Poradce se zabývá automatickou reakcí na nehodu. **Systém ACR** je fyzický.

Systém ACR se skládá z vozidla.

Vozidlo je fyzické.

Automatická reakce na nehodu vyžaduje **systém ACR**.

Automatická reakce na nehodu mění **skupinu osob ve vozidle** z možných zraněných na osoby, kterým je poskytována pomoc.

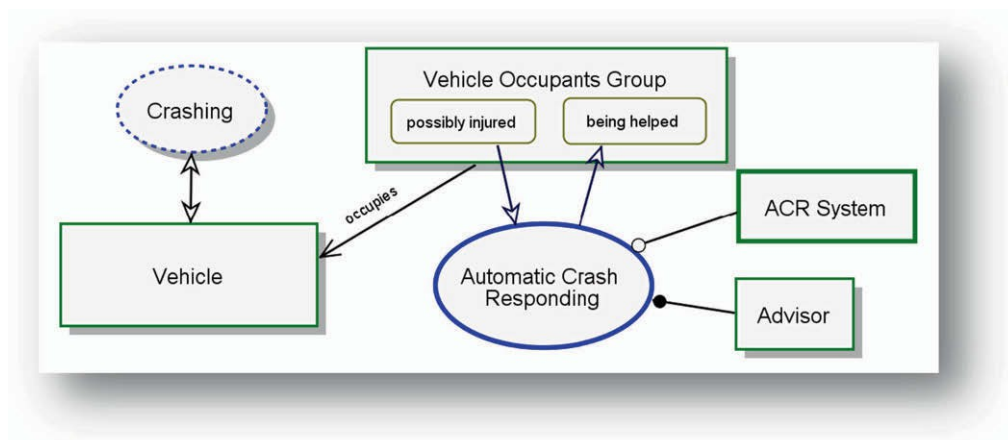
Abychom ušetřili místo, dovolili jsme si vynechat věty vyjadřující fyzickou podstatu věci ve většině následujících příkladů odstavců OPL, protože to je zřejmé ze stínování v OPD.

3.6 Environmentální versus systémové věci

Text, který jsme začali používat jako základ našeho modelu, není napsán způsobem, který usnadňuje proces modelování. Podrobnosti o struktuře a chování systému jsou roztroušeny po celém textu. Nejdříve se s nehodou setkáváme ve větě „*Akcelerometr ... měří závažnost nehody.*“ Později čteme: „*Během několika sekund po středně těžké až těžké nehodě ...*“

Spojením těchto specifikací s našimi předchozími osobními znalostmi o autonehodách si uvědomujeme, že autor specifikace chtěl vyjádřit skutečnost, že došlo k procesu **nárazu**. Tento proces není systémový, to znamená, že není součástí systému. Je spíše vnější vůči systému – odehrává se v prostředí systému: **Náraz** má nepříznivý vliv na **vozidlo** a případně na **skupinu cestujících ve vozidle** a **systém ACR** musí reagovat na výsledky tohoto nešťastného procesu.

Věci, které nejsou součástí systému, ale interagují s ním, se označují jako *environmentální*. Tyto environmentální věci jsou v kontrastu se *systémovými* věcmi – věcmi, které jsou součástí systému. Graficky jsou environmentální věci označeny přerušovanou konturou, na rozdíl od plné kontury systémových věcí. Na [obr. 3.4](#) lze **náraz** identifikovat jako environmentální proces podle jeho přerušované kontury a jako fyzický proces podle jeho stínování. To se odráží také v následující větě OPL:



Obr. 3.4 **Náraz** je přidán jako environmentální proces, který ovlivňuje **vozidlo**

Náraz je environmentální a fyzický.

Skupina osob ve vozidle může být také považována za environmentální, protože není součástí systému ACR, ale spíše příjemcem a operandem – objektem, na kterém systém působí, aby jej transformoval. V našem případě se jedná o transformaci ze stavu **možného zranění** do stavu, kdy **je osobě poskytnuta pomoc**. [Obrázek 3.4](#) zobrazuje **skupinu cestujících ve vozidle** jako fyzický objekt pomocí stínování. To se odráží také v následující větě OPL:

Skupina cestujících ve vozidle je fyzická.

Atribut věci, jehož hodnoty jsou *systémové* a *environmentální*, se nazývá *příslušnost*.

3.7 Počáteční a konečné stavy přidružení ()

Jakmile dojde k **nehodě**, je ovlivněno **vozidlo**. [Obrázek 3.4](#) to znázorňuje pomocí efektového propojení, které představuje obousměrnou šipku mezi **nehodou** a **vozidlem**. Přesná povaha efektu – změna stavu – však v modelu zatím není specifikována. Aby byl model srozumitelnější, prozatím jsme vynechali agregační-participační spojení mezi **systémem ACR** a **vozidlem**. Aby byla změna jasná, jsou vstupní a výstupní stavy **vozidla** na [obr. 3.5](#) specifikovány jako **nepoškozené** a **havarované**. Odpovídající věta OPL zní:

Vozidlo může být **nepoškozené** nebo **havarované**.

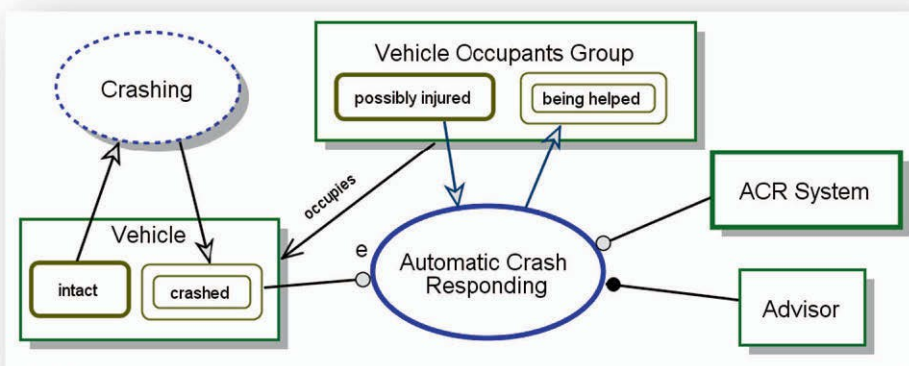
Vstupní stav, **nepoškozený**, je *počáteční stav*, tj. stav, ve kterém objekt začíná svůj životní cyklus po svém vytvoření. Graficky je to znázorněno silnou konturou kolem **nepoškozeného**. Výstupní stav,

havarované, je *konečným stavem*, tj. stavem, ze kterého objekt nemůže vystoupit. Graficky je to znázorněno dvojitou konturou kolem **havarované**. Textově odpovídající věta OPL specifikuje dva stavy:

Vozidlo je zpočátku nepoškozené a nakonec havarované.

Pomocí symbolů počátečního a konečného stavu jsou v [obr. 3.5](#) označeny stavy „možná zraněný“ a „pomoc“ jako počáteční a konečný stav skupiny „osoby ve vozidle“:

Skupina cestujících ve vozidle je zpočátku možná zraněná a nakonec jí je poskytována pomoc.



Obr. 3.5 Účinek **nárazu** je zřejmý díky nahrazení párem vstupně-výstupních odkazů, které specifikují vstupní a výstupní stavy **vozidla**. Odkaz události ze stavu **nárazu vozidla** spouští **automatickou reakci na náraz**.

Po specifikaci stavů **vozidla** nahrazujeme jediné efektové propojení mezi **Crashing** a **Vehicle** párem vstupně-výstupních propojení. Sémantiku této změny lze nejlépe pochopit zkoumáním vět OPL generovaných před a po této náhradě. Původně věta OPL, která odpovídala OPD na [obr. 3.4](#), zněla takto:

Crashing ovlivňuje Vehicle.

Po nahrazení efektového odkazu na [obr. 3.4](#) párem vstupních a výstupních odkazů na [obr. 3.5](#) zní věta OPL takto:

Náraz mění vozidlo z nepoškozeného na poškozené.

Druhá věta je zjevně informativnější, protože nám konkrétně říká, z jakého vstupního stavu do jakého výstupního stavu proces **Crashing** změnil **Vehicle**. Tento dodatečný detail však přichází na úkor zatížení OPD dvěma odkazy – vstupním a výstupním – namísto jediného odkazu efektu.

3.8 Spouštěcí stav a odkaz události

Jakmile **vozidlo** přejde do stavu **havárie**, spustí se funkce systému **ACR (Automatic Crash Responding, automatická reakce na havárii)**. Abychom to znázornili, nakreslíme *odkaz na událost nástroje* ze stavu **havárie** k procesu **Automatic Crash Responding**.

Jak ukazuje [obr. 3.5](#), odkaz na událost nástroje je procedurální odkaz, který je graficky podobný odkazu nástroje s přidavným *modifikátorem řízení* – písmenem e vedle kruhu. Sémantika tohoto odkazu je kombinací sémantiky odkazu nástroje a sémantiky spuštění události. V našem případě tento odkaz označuje skutečnost, že vstup **vozidla** do stavu **havárie** je událostí, která iniciuje proces, ke kterému je odkaz připojen. Jinými slovy, sémantika odkazů událostí je taková, že jakmile **vozidlo** vstoupí do stavu **havárie** (odkud odkaz události pochází), je spuštěn proces **automatické reakce na havárii** (na který odkaz události směřuje). Instrumentální složka odkazu naznačuje, že **vozidlo** není transformováno (ani spotřebováno, ani nemění svůj stav) procesem **automatické reakce na havárii**, který spouští. Věta OPL generovaná v reakci na vložení tohoto odkazu události je:

Havarované vozidlo spustí **automatickou reakci na nehodu**, která vyžaduje **havarované vozidlo**.

Tato věta OPL odráží kombinovanou sémantiku modifikátoru řízení události, která je vyjádřena vyhrazeným slovem OPL iniciuje, s sémantikou nástrojového odkazu, která je vyjádřena vyhrazeným slovem OPL vyžaduje. Jak ukazuje věta, **vozidlo** ve stavu **havárie** je prostě **havarované vozidlo**.

3.9 Shrnutí

- Enablers jsou nezbytné pro to, aby proces mohl proběhnout, ale nejsou ovlivněny výskytem tohoto procesu.
- Agent je lidský pomocník, zatímco nástroj je nelidský pomocník.
- Entity jsou navzájem propojeny dvěma typy vazeb: *strukturálními vazbami* a *procedurálními vazbami*.
 - Procedurální vazby, které existují mezi procesem a objektem nebo jedním z jeho stavů, vyjadřují chování systému; například vazba efektu.
 - Strukturální vazby vyjadřují trvalé, dlouhodobé vztahy mezi dvěma propojenými objekty nebo mezi dvěma propojenými procesy v systému; například agregační-participační vazba.
- Jelikož jsou strukturální a procedurální vazby vyjádřeny ve stejném diagramu, pomáhají integrovat strukturu a chování systému.
- Věci, tj. objekty a procesy, se klasifikují podle jejich
 - *Podstata fyzických a informačních věcí* a jejich
 - *příslušnosti* na *systémové věci* a *environmentální věci*.
- Proces je spuštěn odkazem na událost.

3.10 Problémy

Když cestující přijede na letiště se svým zavazadlem, odbaví ho. Letecká společnost naloží zavazadlo na palubu letadla.

1. Definujte tři stavy **zavazadel** na základě držitele nebo umístění zavazadel od okamžiku, kdy **cestující** dorazí na **letiště**, až po naložení **zavazadel** do **letadla**.
2. Přidejte tři stavy do OPD, které jste začali vytvářet v předchozí kapitole (nezapomeňte použít malá písmena).
3. Jaká je věta OPL, která popisuje stavy **zavazadel**?
4. Jaký proces mění **zavazadlo** z prvního stavu do druhého stavu?
5. Přidejte tento proces do OPD spolu se vstupním propojením z prvního stavu do procesu a z procesu do druhého stavu.
6. Jaká věta OPL popisuje změnu stavů **zavazadel**?
7. Opakujte úlohy 4–6 s procesem, který mění **zavazadla** z druhého stavu do třetího.
8. Proveďte animovanou simulaci svého modelu spuštěním každého z obou procesů a sledujte, zda se stavy mění podle očekávání. Poskytněte snímky obrazovky modelu v různých stavech.

Když letadlo přistane na cílovém letišti, zavazadla se vyloží a vrátí cestujícím.

9. Na základě výše uvedeného textu opakujte úlohy 7–8 s procesem, který mění **zavazadla** ze třetího stavu do prvního.

Kapitola 4

SysML: Diagramy případů použití, bloků a stavových strojů

SysML podporuje specifikaci, analýzu, návrh, ověřování a validaci široké škály komplexních systémů. Tyto systémy mohou zahrnovat hardware, software, informace, procesy, personál a zařízení.

OMG SysML, v1.3 str. 1 (přístup 20. června 2014)

Na chvíli opustíme OPM a začneme s paralelním modelem SysML. SysML je jazyk s více pohledy, kde každý pohled používá jiný typ diagramu. Celkem existuje devět typů diagramů SysML. V této kapitole se seznámíme se třemi typy diagramů: diagramem případů použití, diagramem definice bloků a diagramem stavového stroje. Diagram případů použití ukazuje kontext systému a jak je systém používán k přinášení hodnoty alespoň jednomu z jeho aktérů. Diagram definice bloků představuje bloky systému – hlavní entity, které jsou předmětem zájmu. Diagram stavového stroje ukazuje, jak se mění stavy bloků v systému. Porovnáním OPM a SysML již vidíme, že jejich přístupy jsou odlišné a vzájemně se doplňují. OPM používá jediný model, který kombinuje různé aspekty systému, zatímco SysML používá řadu typů diagramů, z nichž každý se zaměřuje na nějaký konkrétní aspekt systému.

4.1 ký diagram SysML

Náš model začínáme *diagramem use case*, protože tento pohled se používá k získání požadavků a k poskytnutí počátečního porozumění systému a jeho okolí.

Případ použití je způsob, jakým je systém používán, služba, kterou poskytuje alespoň jednomu ze svých uživatelů.

Podle standardu OMG SysML 1.3 (2012) diagram případů použití „popisuje použití systému (předmět) jeho aktéry (prostředí) k dosažení cíle, který je realizován předmětem poskytujícím sadu služeb vybraným aktérům“ (OMG SysML 1.3, 2012, s. 145).

Před nakreslením diagramů případů použití je třeba případy použití zapsat do textové podoby. Tento text může mít různé formáty. V závislosti na potřebách se případy použití zapisují v různé míře formálnosti. Mohou být

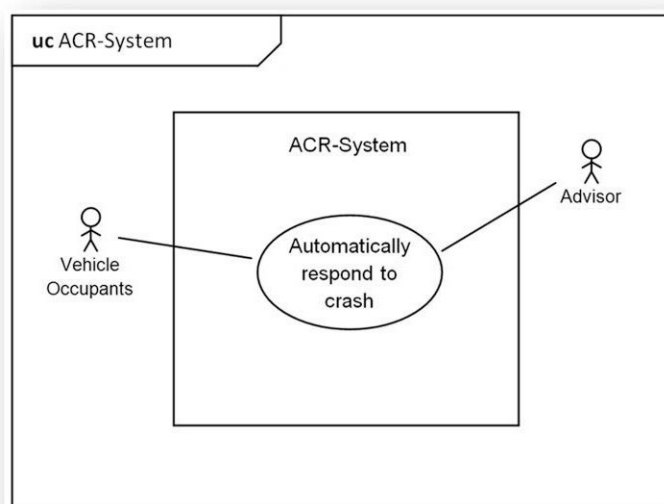
- stručné – krátké shrnutí v jednom odstavci, obvykle hlavního scénáře úspěchu;
- neformální – neformální formát odstavců, kde více odstavců popisuje různé scénáře; a
- plně propracované – nejpropracovanější úroveň, kde jsou podrobně popsány všechny kroky a varianty a kde jsou uvedeny podpůrné části, jako jsou předpoklady a záruky úspěchu.

Obrázek 4.1 je předběžný diagram případů použití systému automatické reakce na nehodu (ACR).

Název use case v našem modelu use case je „**Automaticky reagovat na havárii**“. Jak ukazuje [obr. 4.1](#), use case je znázorněn jako ovál s názvem uvnitř. Uživatelé systému se nazývají aktéři.

Aktér je externí entita, která interaguje se systémem a může od něj získávat služby.

Aktor je znázorněn buď jako lidská postavička, nebo jako stereotyp „aktor“; viz [tabulka 4.1](#). V diagramu případů použití na [obr. 4.1](#) se objevují dva aktéři: **cestující ve vozidle** a **poradce**. Aktor je podle definice externí entita. Na rozdíl od OPM nevyžaduje SysML, aby byl aktorem člověk; může to být cokoli, s čím systém interaguje.



Obr. 4.1 Předběžný diagram použití SysML systému automatické reakce na nehodu (ACR)

Osoby ve vozidle jsou nepochybně externí entitou, protože nejsou součástí systému, ale spíše jeho uživateli a příjemci. Případ **poradce** není tak jednoznačný, protože **poradce** lze považovat za součást systému a namísto toho, aby od systému získával služby, je to on, kdo služby poskytuje. Požadavek, aby aktér získával služby, však není povinný a jako člověk **poradce** se systémem interaguje. V tomto modelu vylučujeme lidi z úvahy o tom, že jsou součástí systému; **poradce** je tedy také aktérem. Každý z těchto dvou aktérů je propojen s use case prostřednictvím *komunikační cesty* – čáry mezi aktérem a use case.

Systém, který poskytuje požadovanou funkci v diagramu případů použití, se nazývá subjekt.

Subjekt v diagramu případů použití je systém, který poskytuje službu.

Subjekt případu použití je znázorněn jako obdélník s názvem subjektu v horní střední části obdélníku. Jak ukazuje [obr. 4.1](#), subjekt v našem diagramu případů použití se nazývá **ACR-System**.

Celý diagram použití je zobrazen v rámečku diagramu – obdélníku, který je vyžadován pro každý diagram SysML. V levém horním rohu rámečku diagramu se nachází štítek s názvem – obdélník se zúženým pravým dolním rohem – který obsahuje název záhlaví. Název záhlaví má následující syntaxi:

<diagramKind> [modelElementType] <modelElementName> [diagramName]

Pole diagramKind, které je tučně, a modelElementName jsou povinná. Každý diagramKind má zkratku složenou ze dvou nebo tří malých písmen. Jak je znázorněno na [obr. 4.1](#), diagramKind našeho případu použití je **uc**, zatímco diagramName je ACR-System. Další dva tokeny, modelElementType a diagramName, jsou volitelné a pokud se objeví, jsou uzavřeny v závorkách, což umožňuje čtenáři diagramu je odlišit.

[Tabulka 4.1](#) uvádí hlavní prvky diagramu případů použití, jejich sémantiku a symboly.

Tabulka 4.1 Hlavní prvky diagramu případů použití, jejich sémantika a symboly

Element:	Symbol
Semantics	
Use Case: A service the system (subject) provides to at least one actor	
Actor: An external entity that interacts with the system and can get services from it	
Subject: The system that provides the service(s)	
Communication path: A connection between an actor and a use case	

Guillemets, také známé jako symboly pro *zpětné přetáčení* («) a *rychlé přetáčení* (»), jsou úhlové uvozovky, jako ty, které obklopují následující slovo: «guillemets». V SysML slovo uvnitř dvojice guillemets označuje stereotyp – mechanismus rozšiřitelnosti, který umožňuje vytvářet nové prvky modelu.

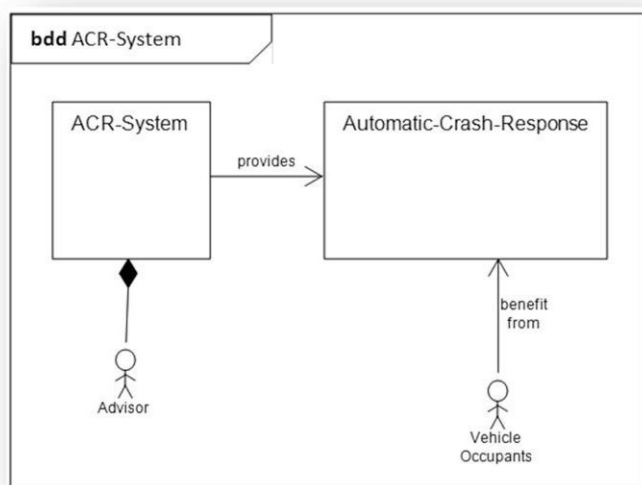
Stereotyp je znázorněn jako obdélníkový rámeček s názvem stereotypu, například „block“ v uvozovkách, «block», zaznamenaným v horní střední části rámečku, jako je tomu v případě «actor» v [tabulce 4.1](#). Jméno aktéra, ActorName, je zaznamenáno pod zápisem stereotypu «actor».

4.2 Bloky SysML a diagram definice bloku ()

Blok SysML je modulární komponenta, která definuje soubor vlastností popisujících část systému nebo jiný prvek zájmu. Blok SysML, který zhruba odpovídá třídě UML, může zahrnovat jak strukturální, tak behaviorální vlastnosti, jako jsou vlastnosti a operace. Blok může obsahovat vlastnosti pro specifikaci svých hodnot, částí a odkazů na jiné bloky.

Diagram definice bloků SysML (bdd) definuje vlastnosti bloků a vztahy mezi bloky, jako jsou asociace, generalizace a závislosti.

Diagram definice bloků zachycuje definici bloků z hlediska vlastností a operací a vztahů, jako je hierarchie systému nebo klasifikační strom systému. Souvisejícím diagramem SysML je *diagram vnitřního bloku (ibd)*, který zachycuje vnitřní strukturu bloku z hlediska vlastností a spojení mezi vlastnostmi.



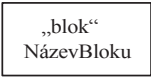
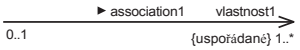
Obr. 4.2 Předběžný blokový diagram systému automatické reakce na nehodu (ACR)

Obrázek 4.2 je předběžný blokový diagram (bdd) systému automatické reakce na nehodu (ACR). DiagramKind, **bdd**, to označuje. Tento bdd vyjadřuje dva hlavní bloky systému a vztah mezi nimi, stejně jako hlavní aktéry a jejich vztahy k blokům.

Tyto dva bloky v bdd jsou ACR-System a Automatic-Crash-Response. Jsou propojeny pomocí ReferenceAssociation označené jako „provides“. Advisor je zobrazen jako aktér, který je součástí ACR-System. Tento vztah celku a části je vyjádřen černým kosočtvercem, symbolem SysML pro vztah celku a části.

Dalším aktérem jsou Vehicle Occupants (osoby ve vozidle). Jsou propojeni pomocí ReferenceAssociation s označením „benefit from“ (těží z) s blokem Automatic-Crash-Response (obr. 4.2).

Tabulka 4.2 Hlavní prvky diagramu definice bloku, jejich sémantika a symboly

Prvek:	Symbol
Sémantika	
Blok Modulární komponenta, která definuje soubor vlastností popisujících část systému nebo jiný prvek zájmu.	
Actor: Externí entita, která interaguje se systémem a může od něj získávat služby.	
ReferenceAssociation: Propojení mezi bloky, které označuje povahu jejich asociace	
PartAssociation: Propojení mezi bloky, které označuje, že blok propojený s diamantem je celý	
Generalizace: Spojení mezi dvěma bloky označující, že blok propojený s trojúhelníkem je obecný	

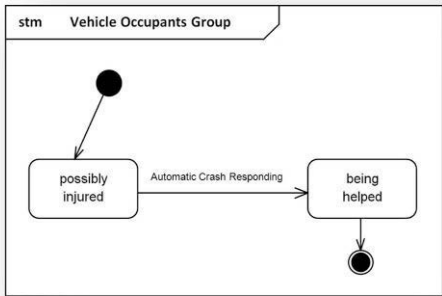
4.3 ový diagram SysML

SysML má typ diagramu, který je určený pro modelování stavů bloku a možných přechodů mezi nimi – diagram stavového stroje, zkráceně stm. V návaznosti na myšlenku původně představenou Harelem (1987, 1988) definuje balíček SysML State Machine sadu pojmů, které mohou modelovat diskrétní chování prostřednictvím přechodů mezi stavy. Stavový stroj může reprezentovat chování vyjádřené jako historie stavů objektu z hlediska jeho přechodů a stavů.

Obrázek 4.3 je diagram stavového stroje SysML (stm) skupiny cestujících ve vozidle. Je podobný diagramu OPD na obrázku 2.3 v tom, že oba obsahují stejné dva stavy pro skupinu cestujících ve vozidle. Symbol stm používaný k označení stavu je obdélník – stejný jako v OPM. Hlavní rozdíl mezi nimi spočívá v tom, že stm se nevztahuje na celý systém ACR. Vztahuje se pouze na blok skupiny cestujících ve vozidle. Proces OPM Automatická reakce na nehodu je ve stm vyjádřen jako *spouštěč* se stejným názvem, který způsobuje *přechod* ze stavu **možného zranění** do stavu **poskytování pomoci**.

Černý kruh na obr. 4.3 představuje počáteční stav. Tento stav se označuje jako *pseudostav*, protože nejde o skutečný stav, ale pouze o indikaci pro čtenáře diagramu, kde má začít. Je propojen s počátečním stavem, případně poškozeným, bloku, jehož stavový stroj je modelován, což je v našem případě skupina osob ve vozidle. Černý kruh s bílým okrajem na obr. 4.3 představuje konečný (pseudo)stav – na který odkazuje (skutečný) konečný stav – pomoc. Tyto dva symboly umožňují identifikaci počátečního a konečného stavu v diagramu stavového stroje. Jak uvidíme později, OPM označuje počáteční stav pomocí tučné čáry

stavového obdélníku a konečný stav pomocí dvojitého obdélníku. Tím se eliminuje potřeba dvou druhů pseudo stavů, které používá SysML.



Obr. 4.3 Diagram stavového stroje SysML (stm) systému ACR
Tabulka 4.3 Hlavní prvky diagramu stavového stroje, jejich sémantika a symboly

Element (node) name: Semantics	Symbol (Syntax)
State Machine Diagram: A diagram specifying possible states of a block, transitions between them, and conditions for those transitions.	
State: A situation a block can be at during its lifetime. Can be <i>simple</i> (atomic, non-decomposable) or <i>composite</i> .	
Composite State: A state enclosing lower level, possibly atomic (non-composite), states.	
Transition: A connection between two states specifying how states in a state machine change. Labeled by a trigger, which is associated with an event, optional [guard], which is a condition for the trigger, and 'activity', which is a possible action done during the transition.	

Tabulka 4.3 ukazuje hlavní prvky diagramu stavového stroje, jejich sémantiku a symboly. Jak tabulka ukazuje, stav může být složený a obsahovat vnitřní procesy nižší úrovně. Přejít může být označen kromě spouštěče také volitelným strážcem v závorkách a jednou nebo více volitelnými aktivitami, které syntakticky následují za symbolem zpětného lomítka (`\`), což jsou akce prováděné během přechodu.

4.4 Shrnutí

- SysML má devět typů diagramů, které modelují různé aspekty systému
- Diagram použití je často prvním diagramem, který se připravuje, protože poskytuje kontext systému a způsob, jakým s ním aktéři interagují.
- Blok je základní jednotka, podobná třídě v UML, používaná v diagramu definice bloků a interním diagramu bloků. Slouží k definování struktury systému.
- Diagram stavového stroje je diagram SysML, který specifikuje možné stavy relevantních bloků v systému a přechody mezi těmito stavy.

4.5 Problémy

1. Nakreslete diagram případů použití SysML pro systém popsaný níže.

Cestující přicházející na letiště odevzdá své zavazadlo letecké společnosti, se kterou letí. Systém pro manipulaci se zavazadly řídí přepravu zavazadel do cílového místa cestujícího.

2. Nakreslete blokové schéma systému popsaného výše.
3. **Poloha zavazadel** má stavy **cestující**, **letiště odletu**, **letadlo**, **letiště příletu**, **jiné**. Vytvořte model pomocí diagramu stavového stroje SysML a uveďte, co způsobuje přechody mezi stavy.
4. Porovnejte tři typy diagramů vytvořených v výše uvedených třech úlohách z hlediska jejich informačního obsahu.
5. Co lze o systému říci na základě jednotlivých diagramů?
6. Jak lze informace integrovat, aby bylo možné získat ucelený pohled na systém?

Kapitola 5

Zdokonalení pomocí přiblížení

Nejhlubší části oceánu jsou nám zcela neznámé... Co se děje v těch vzdálených hlubinách? Jaké tvory obývají nebo by mohly obývat tyto oblasti dvanáct nebo patnáct mil pod hladinou vody? To je téměř nemožné odhadnout.

Jules Verne, 20 000 mil pod mořem (1869)

V předchozích kapitolách jsme se seznámili se základními pojmy OPM, ale zatím jsme se dotkli pouze povrchu systému, který modelujeme. V této kapitole upřesníme další podrobnosti o systému a zároveň odhalíme některé další pojmy modelování OPM a to, jak je lze využít k podrobnějšímu znázornění našeho systému. Abychom prozkoumali text, který specifikuje systém, který modelujeme, vrátíme se k informacím z prvních vět:

„Akcelerometr ... měří závažnost nárazu.“

Tuto informaci kombinujeme s informací z věty v pokračování:

„Během několika sekund po středně těžké až těžké nehodě modul OnStar odešle zprávu ...“

Text opomíjí důležité informace, které musíme získat nepřímou. Již jsme modelovali skutečnost, že **vozidlo** je (ve stavu) **havarované**. Fráze „*středně těžká až těžká nehoda*“ naznačuje, že musíme modelovat závažnost nehody, protože ta určuje, zda bude odeslána zpráva. Implicitním předpokladem, který zde modelujeme, je, že pokud je nehoda lehká, je nepravděpodobné, že by způsobila zranění, takže systém by neměl být aktivován. V důsledku toho má smysl modelovat jako další proces **měření závažnosti nehody**.

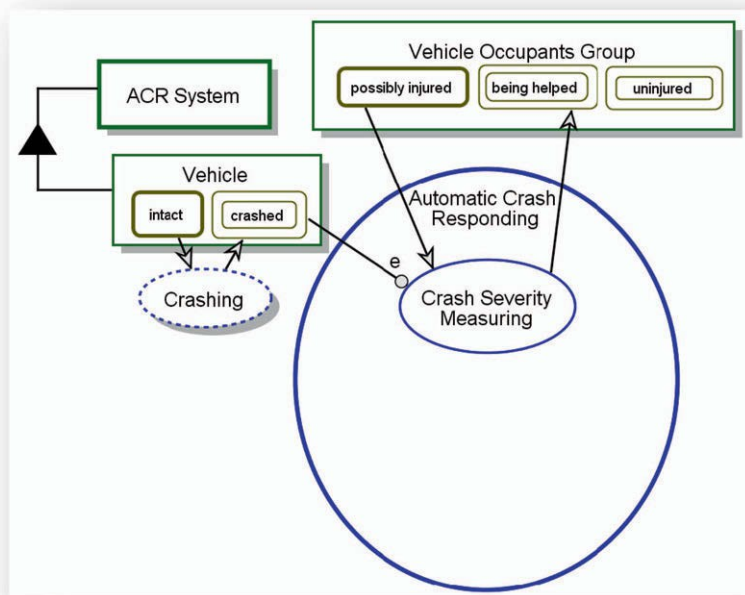
5.1 Měření závažnosti nehody ()

Již jsme určili, že **automatická reakce na nehodu** je funkcí našeho **systému ACR**. Jedná se o hlavní proces v diagramu systému, SD – OPD nejvyšší úrovně. Proces **měření závažnosti nehody**, který se chystáme modelovat, zjevně není na stejné úrovni důležitosti jako **automatická reakce na nehodu**. Jedná se spíše o *podproces automatické reakce na nehodu*. Navíc, jak systém nyní chápeme, bude to jeden (možná první) z několika podprocesů funkce **automatické reakce na nehodu**.

Mohli bychom zkusit modelovat **měření závažnosti nehody** podobným způsobem jako **havárii**. To však z několika důvodů pravděpodobně není dobrý nápad. Za prvé, **havárie** je environmentální proces. Za druhé, modelování **měření závažnosti nehody** na stejné úrovni jako **automatická reakce na nehodu** by mohlo být interpretováno tak, že tyto dva procesy mají stejnou důležitost, což samozřejmě není pravda. Za třetí, naše OPD již začíná být poněkud přeplněné a my bychom jej rádi udrželi jednoduché a srozumitelné.

5.2 In-Zooming: Vylepšení procesu v novém OPD ()

Zvětšení nového diagramu je proces modelování OPM, který vytváří nový potomský OPD, ve kterém lze specifikovat podrobnosti zvětšeného procesu – jeho podprocesy a objekty s nimi spojené. V našem případě je zvětšen **automatický reakce na nehodu**, což umožňuje tento proces zdokonalit modelováním jeho podprocesů a jejich interakcí s objekty nižší úrovně v novém OPD na úrovni pod úrovní SD. [Obrázek 5.1](#) ukazuje proces **automatické reakce na nehodu** po jeho zvětšení a po nakreslení jeho prvního podprocesu, **měření závažnosti nehody**, uvnitř něj v horní části elipsy obklopující proces **automatické reakce na nehodu**. Odkazy, které byly připojeny k **automatické reakci na nehodu**, byly přesunuty a připojeny k **měření závažnosti nehody**.

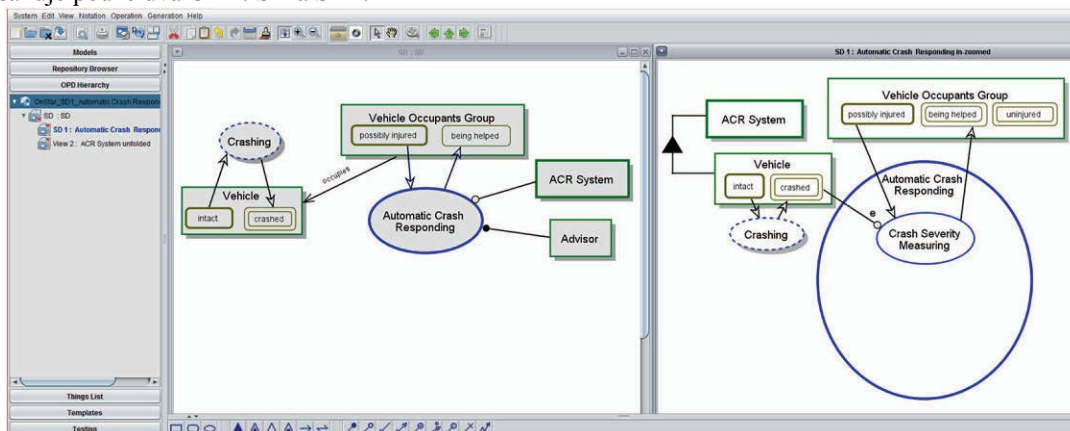


Obr. 5.1 Proces **automatické reakce na nehodu** je zvětšen a zobrazuje svůj první podproces, **měření závažnosti nehody**, vložený uvnitř něj.

5.3 Strom OPD ()

OPD na [obr. 5.1](#) nenahrazuje SD, na kterém jsme dosud pracovali. Jedná se o nový OPD, který doplňuje SD a je na nižší úrovni než SD. SD je vždy jediným OPD nejvyšší úrovně – je *kořenem stromu OPD*. [Obrázek 5.2](#), který je snímkem obrazovky OPCAT, zobrazuje OPD nejvyšší úrovně, *SD* (v levém

) a nový, nazvaný SD1 – *Automatická reakce na havárii* (vpravo od SD). Obrázek 5.2 také ukazuje v panelu hierarchie OPD na levé straně strom OPD procesů, který v současné době obsahuje pouze dva OPD: SD a SD1.



Obr. 5.2 Snímek obrazovky OPCAT, na kterém jsou současně zobrazeny dva OPD: SD (vlevo) a SD1 – Automatická reakce na nehodu v přiblíženém zobrazení (vpravo). Hierarchický strom OPD je zobrazen v levém panelu.

SD a SD1 jsou dva OPD, které v současné době tvoří *sadu OPD*, tj. sadu OPD uspořádaných do stromu procesů, které společně specifikují systém. Sada OPD se neustále rozšiřuje, protože se postupně vytvářejí další OPD, aby se model stále více zdokonaloval a byl konkrétnější. Schopnost přidat potomka, podřízený OPD, kdykoli ten, na kterém se právě pracuje, dosáhne svých limitů přetížení, umožňuje zabránit přeplnění jakéhokoli jednotlivého OPD.

Věta OPL, která propojuje odstavec OPL SD s odstavcem OPL SD1, zní:

Automatická reakce na havárii z SD přiblíží SD1 k měření závažnosti havárie.

Tento typ věty označuje hierarchické vztahy mezi dvěma odstavci OPL představujícími OPD z přilehlých hierarchických úrovní. V našem případě to znamená, že SD1 je podřízeným SD.

5.4 Modelové znázornění skutečnosti OPM ()

Označený strukturální odkaz v SD ze skupiny „Vozidlo – cestující“ na „Vozidlo“, který je na obr. 3.3 označen značkou „occupies“, se v SD1 (obr. 5.1) neopakuje. Toto vynechání je volbou prezentace založenou na následujícím principu OPM reprezentace modelových faktů.

Princip OPM pro reprezentaci modelových faktů

Modelový fakt OPM se musí objevit alespoň v jednom OPD, aby mohl být reprezentován v modelu.

Tento princip stanoví, že stačí, aby se modelová skutečnost objevila pouze jednou v jakémkoli OPD modelu OPM, aby byla platná pro celý model. Tento princip nevylučuje možnost reprezentovat jakoukoli modelovou skutečnost libovolný početkrát v tolika OPD, kolik si modelář přeje. Ačkoli však lze do jakéhokoli OPD zahrnout libovolný počet entit, z důvodu přehlednosti a zamezení nepořádku je často velmi žádoucí zahrnout pouze ty prvky, které jsou nezbytné pro pochopení určitého aspektu nebo pohledu na systém. V našem případě jsme se rozhodli nezahrnout označený strukturální odkaz do SD1, protože nepřispívá k pochopení bodu, který chceme v tomto OPD zdůraznit.

5.5 Atribut závažnosti nehody a jeho měření pomocí indexu závažnosti nehody ()

První dílčí proces **automatické reakce na nehodu**, **měření závažnosti nehody**, určuje **závažnost nehody**. **Závažnost nehody** je nový objekt, který dosud nebyl modelován. **Závažnost nehody** není jen nový objekt, ale popisuje **vozidlo**. Jinými slovy, je to **atribut vozidla**. Tento atribut se stává relevantním v důsledku procesu **nehody**.

Atribut je objekt, který charakterizuje věc.

V našem případě je **závažnost nehody** atributem, který charakterizuje objekt **vozidlo**. [Obrázek 5.3](#) ukazuje **závažnost nehody** propojenou s **vozidlem** prostřednictvím strukturálního vztahu *výstavy-charakterizace*. Symbol výstavy-charakterizace je rovnostranný černý trojúhelník uvnitř většího bílého trojúhelníku, jako tento: . Vrchol tohoto trojúhelníku je propojen s *vystavovatelem*, kterým je objekt **vozidlo**, a jeho základna je propojena s objektem **závažnost nehody**, který je atributem **vozidla**.

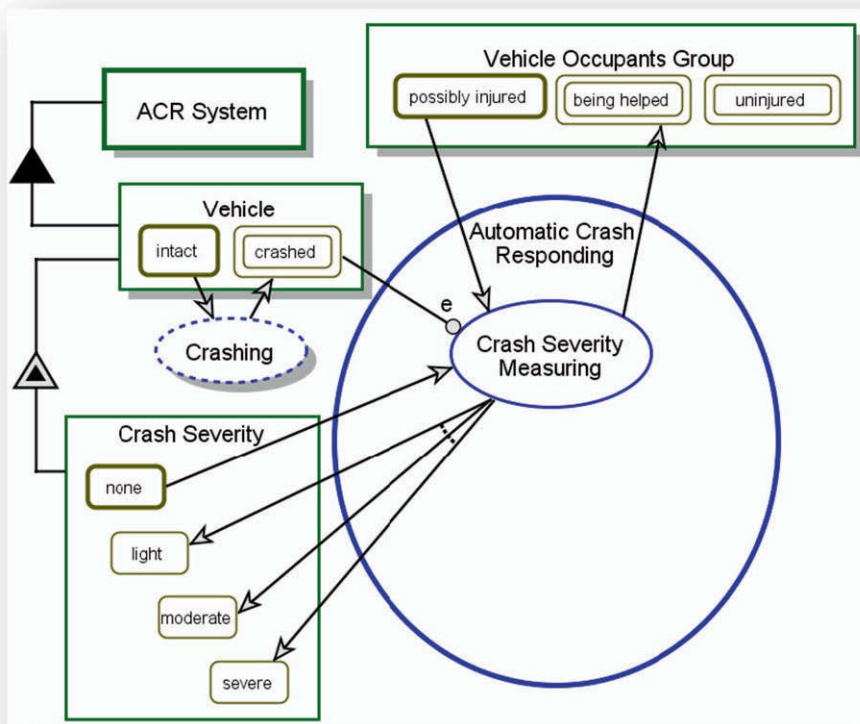
Jako samostatný objekt má tento atribut čtyři stavy. Přesněji řečeno, protože stavy atributu se nazývají *hodnoty*, má **Crash Severity** čtyři hodnoty: **none**, **light**, **moderate** a **severe**. Ty jsou znázorněny na [obr. 5.3](#) uvnitř **Crash Severity**. OPL věta, která tyto hodnoty vyjmenovává, je následující:

Závažnost nehody může být **žádná**, **lehká**, **střední** nebo **těžká**.

Jakmile dojde k **havárii**, stav **vozidla** se změní z původního **nepoškozeného** stavu na konečný stav **havárie**. Po vstupu **vozidla** do stavu **havárie** spustí tento dílčí proces odkaz na událost přístroje specifikovaný stavem z **havarovaného** stavu do **měření závažnosti havárie**. **Crash Severity Measuring** je prvním (a v současné době jediným) podprocesem přiblíženého procesu **Automatic Crash Responding**. **Crash Severity Measuring** mění **Crash Severity** z jeho počátečního stavu **none** na přesně jeden ze tří dalších stavů.

[Obrázek 5.3](#) to graficky vyjadřuje pomocí (1) vstupního propojení z hodnoty **žádná závažnost nehody** k procesu **měření závažnosti nehody** a (2) tří alternativních výstupních propojení vycházejících ze stejného bodu na elipse **měření závažnosti nehody** ke každé ze tří hodnot, **lehká**, **střední** a **těžká**, spojených přerušovanou křivkou. Tento přerušovaný oblouk označuje logický operátor XOR (exkluzivní OR) mezi odkazy. V OPCAT se zobrazuje automaticky pouze v případě, že odkazy XOR vycházejí ze společného bodu, jako je tomu v tomto případě, nebo směřují do společného bodu (jak je znázorněno na [obr. 6.2](#)).

Skutečnost, že tyto tři výstupní odkazy pocházejí ze stejného bodu a že je spojuje přerušovaná křivka, symbolizuje logický operátor XOR mezi odkazy: **Crash Severity Measuring** určuje, že **Crash Severity** může mít přesně jednu ze tří možných výstupních hodnot: **lehká**, **střední** nebo **závažná**, ale ne dvě nebo všechny tři současně. Věta OPL, která popisuje tuto změnu stavu, zní:

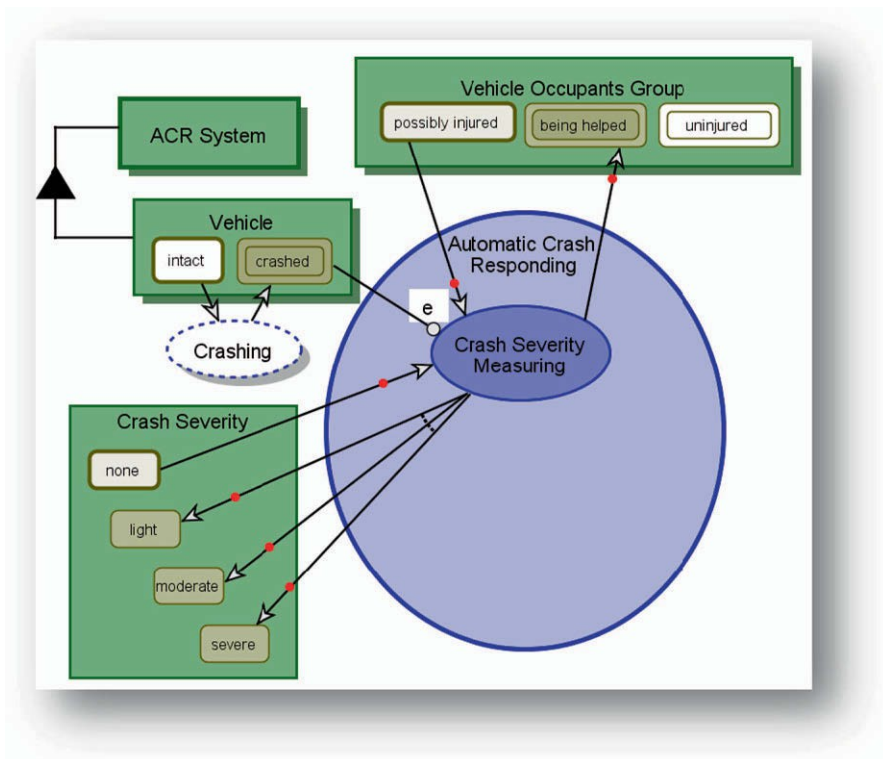


Obr. 5.3 **Závažnost nehody** se čtyřmi stavy je přidána jako atribut vozidla. Stav „žádný“ je počáteční a vstupní stav, zatímco „lehký“, „střední“ a „těžký“ jsou možné výstupní stavy procesu **měření závažnosti nehody**.

Měření závažnosti nehody mění **závažnost nehody** z **none** na přesně jeden ze stavů **light**, **moderate** nebo **severe**.

5.6 Simulace systému: Animované provedení testu „“

V tomto okamžiku je vhodné zahájit animované provádění systému v jeho aktuálním návrhu, aby bylo možné otestovat jeho koncepční fungování. [Obrázek 5.4](#) ukazuje systém poté, co (environmentální) proces **Crashing** změnil stav **vozidla** z **nepoškozeného** na **havarované**, což byla událost, která spustila proces **Automatic Crash Responding**. V rámci tohoto procesu se blíží dokončení procesu **Crash Severity Measuring**, který změní atribut **Crash Severity** vozidla z **none** na jeden ze stavů **light**, **moderate** nebo **severe**.



Obr. 5.4 Animované provedení systému v jeho současné podobě: **Měření závažnosti nehody** právě skončilo, změna **závažnost nehody se změnila z žádné na závažnou**

5.7 Shrnutí

- Přiblížení je mechanismus zdokonalení, který pomáhá řídit složitost systému.
- Přiblížení procesu vytvoří nový OPD s nafouknutým přiblíženým procesem.
- Procesy nižší úrovně (podprocesy) jsou vnořeny do tohoto přiblíženého procesu a mohou být propojeny s objekty nižší úrovně uvnitř nebo vně přiblíženého procesu.
- Rekursivní přiblížení vede k sadě OPD, která má stromovou strukturu, ve které OPD nižší úrovně modelují stále podrobnější informace o systému.
- Princip reprezentace faktů modelu OPM stanoví, že aby mohl být fakt modelu OPM reprezentován, musí se objevit alespoň v jednom OPD, aby mohl být reprezentován v modelu.
- Použití tohoto principu pomáhá snížit nepřehlednost diagramů, díky čemuž jsou jednotlivé diagramy dostatečně jednoduché, aby je bylo možné pochopit bez kognitivního přetížení.

5.8 Problémy

Pokračujeme s příkladem systému manipulace se zavazadly a předpokládáme, že aktuální model je znázorněn na [obr. 2.5](#).

1. Zkontrolujte, zda se všechny podpůrné prvky – agenti a nástroje – procesu **manipulace se zavazadly** objevují na [obr. 2.5](#). Pokud některé podpůrné prvky chybí, doplňte je.
2. Jaký je vhodný typ propojení mezi **leteckou společností** na jedné straně a **letadlem a personálem letecké společnosti** na straně druhé? Přidejte propojení do modelu.
3. Jaký je vhodný vztah mezi **cestujícím a zavazadlem**? Přidejte jej do modelu.
4. Které objekty jsou fyzické? Vyjádřete to v modelu.
5. Které objekty jsou environmentální? Vyjádřete to v modelu.
6. Které objekty v modelu by měly být stavové? Přidejte příslušné stavy ke každému takovému objektu.

Kapitola 6

Dynamický aspekt systémů

Odborník může při vysvětlování určité myšlenky nebo popisu chování náhle sáhnout po bloku a nakreslit náčrtky toho, co dělá, a říci: „Musí to vypadat takto“ nebo „Stačí se podívat na graf a hned vím, jestli je něco v nepořádku.“

Firlej a Helens (1991)

V návaznosti na modelování naší případové studie se v této kapitole dále zabýváme procesními otázkami, jako je pořadí provádění a způsob specifikace toho, zda jsou procesy sekvenční, souběžné nebo alternativní. Tyto otázky souvisejí s dynamickým aspektem systému a s jeho operační sémantikou.

6.1 Ukončení v případě mírné závažnosti ()

Připomeňme, že specifikace systému ACR stanoví:

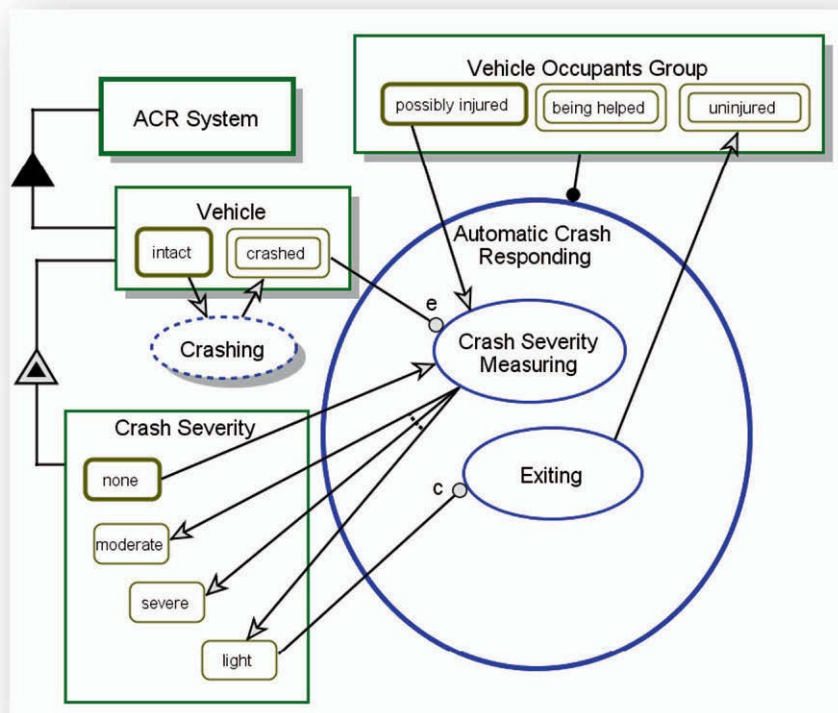
Během několika sekund po středně těžké až těžké nehodě modul OnStar odešle zprávu ...

Pokud je tedy **závažnost nehody nízká**, chceme modelovat, že proces **automatické reakce na nehodu** je ukončen a systém dokončil své provedení. K tomu v [obr. 6.1](#) přidáme do **skupiny cestujících ve vozidle** třetí stav, **nezraněný**, který je také konečný. Pomocí podmíněného propojení (propojení nástroje s modifikátorem řízení c vedle jeho kruhového konce) propojíme stav „**lehká**“ **závažnosti nehody** s novým podprocesem „**ukončení**“, který změní stav **skupiny „osoby ve vozidle“** na „**nezraněný**“. V tomto případě se provádění systému ukončí. Sémantika podmíněného propojení je taková, že pokud objekt, ke kterému je propojení připojeno, existuje, nebo pokud stav, ke kterému je propojení připojeno, je aktuálním stavem objektu, pak se proces provede, jinak se proces přeskočí. Sémantika podmíněného propojení nástrojů je slabší než sémantika (nepodmíněného) propojení nástrojů. Sémantika druhého je taková, že pokud propojený objekt neexistuje (nebo není v požadovaném stavu), pak se provádění systému zastaví a čeká, až nástroj začne existovat (nebo bude v požadovaném stavu).

6.2 Vytváření zpráv a odesílání zprávy „“

Pokračujeme v modelování toho, co se stane v případě, že **měření závažnosti havárie** změnilo **závažnost havárie** z **žádné** na **mírnou** nebo **závažnou**, na základě následujícího textu:

Během několika sekund po středně těžké až těžké nehodě modul OnStar odešle zprávu do call centra OnStar (OCC) prostřednictvím mobilního připojení a informuje poradce, že došlo k nehodě. Na základě přijaté zprávy poradce odešle potřebnou pomoc.



Obr. 6.1 Měření závažnosti nehody určilo, že **závažnost nehody** je **mírná**, takže výstup změnil stav skupiny cestujících ve vozidle na nezraněný.

Podle tohoto popisu je po **měření závažnosti nehody** v důsledku nehody, jejíž **závažnost** je **střední** nebo **vysoká**, vytvořena zpráva, která je poté odeslána prostřednictvím call centra OnStar poradci, který na základě zprávy vyšle pomoc. V souladu s tím, jak ukazuje [obr. 6.2](#), přidáváme tři následné dílčí procesy: **Vytvoření zprávy**, **Odeslání zprávy** a **Vyslání pomoci**. Následující věta OPL vyjadřuje vztah XOR mezi podmínkovými odkazy ze stavů **střední** a **těžké závažnosti nehody** k **vytvoření zprávy**.

Vytvoření zprávy nastane, pokud je **závažnost nehody** přesně **střední** nebo **závažná**.

Jak si vzpomínáme, princip modelového znázornění faktů OPM stanoví, že prvek OPM se musí objevit alespoň v jednom OPD, aby mohl být znázorněn. Na základě tohoto principu a za účelem zjednodušení OPD byl proces **Crashing**, který se objevil na [obr. 6.1](#), odstraněn z [obr. 6.2](#). To nám umožňuje přidat objekty zmíněné v textu, které jsou zde relevantní: **call centrum OnStar** a **mobilní systém**, které jsou součástí **systému ACR** (kromě **vozidla**), stejně jako **poradce** a **zpráva**.

Princip časové osy OPM se standardně dodržuje, pokud neexistuje indikace odchylky od časové osy. Indikace odchylky od časové osy shora dolů v rámci přiblíženého procesu zahrnují interní události v rámci procesu, které mohou způsobit smyčky.

Nejvyšší bod elipsy procesu slouží jako referenční bod, takže proces, jehož referenční bod je vyšší než jeho protějšek, začíná dříve. Pokud jsou referenční body dvou nebo více procesů ve stejné výšce (v rámci určité tolerance), tyto procesy začínají současně.

Podle principu časové osy OPM se nejprve provede **měření závažnosti selhání**, poté následuje **ukončení** (v případě **mírné závažnosti selhání**) nebo v případě **střední závažnosti selhání** nebo **závažné závažnosti selhání vytvoření zprávy**, následované **odesláním zprávy** a **odesláním nápovědy**.

6.4 Pomoc je na cestě ()!

Pokračujeme modelováním následujícího textu, který popisuje, co se stane, když **poradce** obdrží **zprávu**.

Mezi poradcem a cestujícími ve vozidle je navázáno hlasové spojení. Poradce poté může spojit dispečink tísňové linky 911 nebo veřejnou bezpečnostní službu (PSAP), která určí, zda jsou nutné záchranné služby, a pokud ano, zda se jedná o sanitku, vrtulník nebo obojí. Pokud cestující nereagují, může poradce poskytnout dispečerovi tísňové linky informace o nehodě z SDM, které odhalí závažnost nehody. Dispečer může určit, jaké záchranné služby jsou vhodné. Pomocí satelitu globálního pozičního systému (GPS) mohou poradci OnStar sdělit záchranářům polohu vozidla.

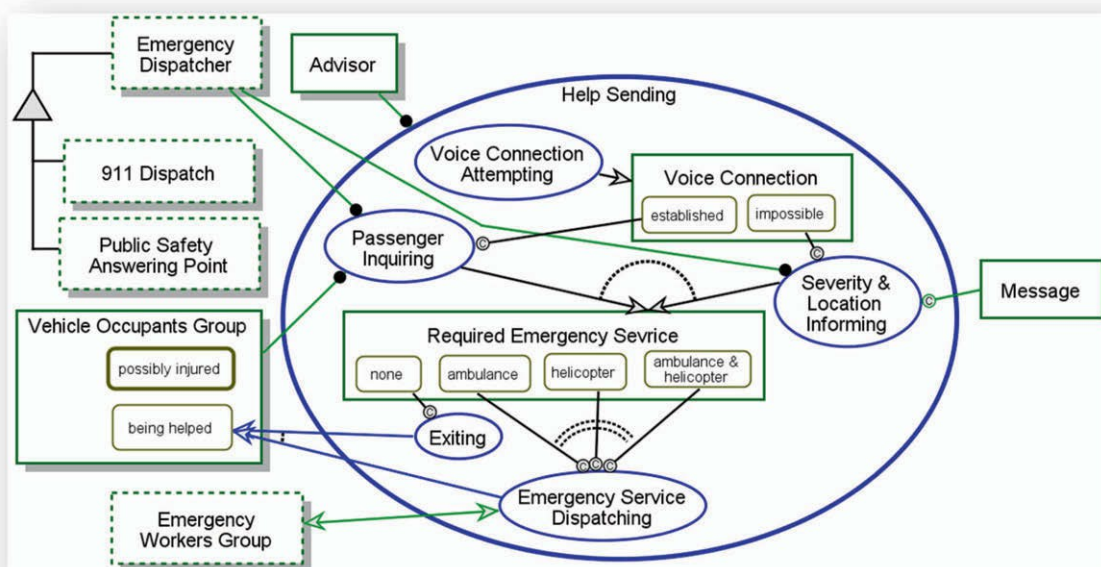
Tento popis pokrývá širokou oblast a zahrnuje řadu nových procesů, včetně **pokusů o hlasové spojení**, **konference veřejné pomoci**, **poskytování informací o nehodě** a **vyslání záchranné služby**.

Obrázek 6.3 ukazuje zvětšený výřez **funkce Help Sending**. **Pokus o hlasové spojení** vytvoří **hlasové spojení**, které může být **nemožné** (pokud cestující neodpovídají nebo není k dispozici mobilní připojení; není modelováno) nebo **navázané**. Pokud je **hlasové spojení nemožné**, **poradce** informuje **dispečera tísňové služby** o **závažnosti nehody** a poloze prostřednictvím procesu **informování o závažnosti a poloze**. **Dispečer tísňové služby** je zobecněním **dispečera tísňové linky 911** a **centra tísňového volání**.

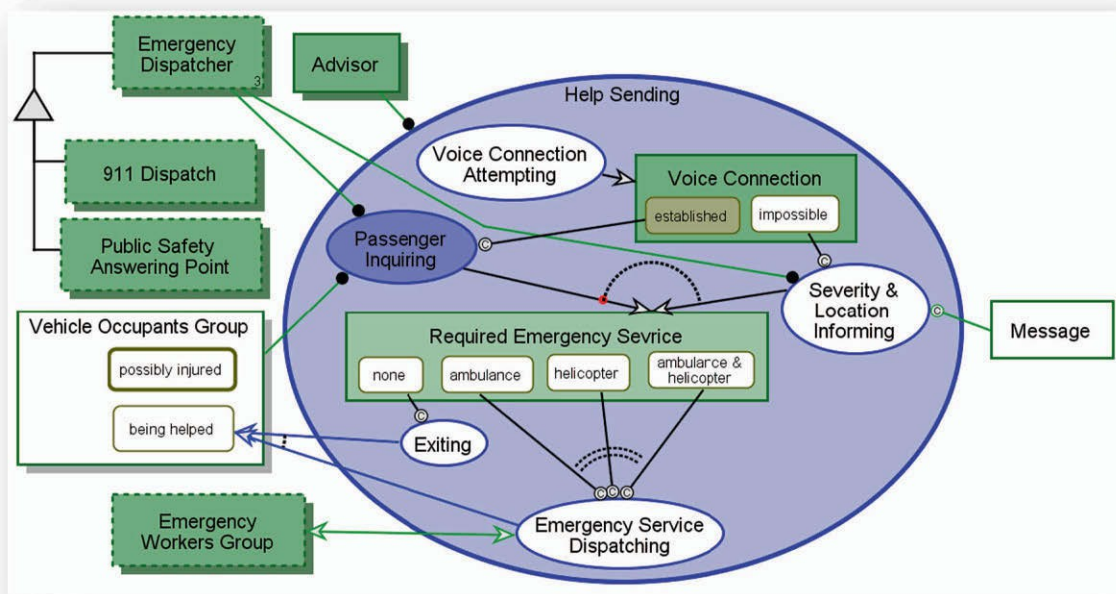
Pokud je **hlasové spojení navázané**, **poradce** a **dispečer tísňové služby** provedou konzultaci s **cestujícími**. Buď **dotazování cestujících**, nebo **informování o závažnosti a poloze** určuje **požadovanou záchrannou službu**, kterou může být **žádná**, **sanitka**, **vrtulník** nebo **sanitka a vrtulník**. Toto rozhodnutí se používá pro **dispečink záchranné služby**, který v případě potřeby vyšle příslušnou **skupinu záchranářů**, objekt prostředí, na cestu, aby pomohla, a změnil stav **skupiny cestujících ve vozidle** na „**pomoc poskytnuta**“.

6.5 Scénáře: Vlákna provádění

Obrázek 6.4 ukazuje konkrétní průběh provedení funkce „Help Sending“ (Odeslání pomoci), který lze sledovat podle stavu každého objektu. Funkce „Voice Connection Attempting“ (Pokus o hlasové spojení) vytvoří **hlasové spojení** ve stavu „established“ (navázáno), což vede k funkci „Passenger Inquiring“ (Dotazování cestujících). Pokud tento proces vytvoří funkci „Required Emergency Service“ (Požadovaná záchranná služba) ve stavu „none“ (žádná), dojde k funkci „Exiting“ (Ukončení), jinak dojde k funkci „Emergency Service Dispatching“ (Vyslání záchranné služby). V obou případech přejde skupina „Vehicle Occupants“ (Cestující ve vozidle) do stavu „



Obr. 6.3 je zvětšený pohled na proces „Help Sending“ (Odeslání pomoci), který odhaluje čtyři dílčí procesy, které vyvrcholí stavem „being helped“ (poskytnutí pomoci) skupiny „Vehicle Occupant Group“ (Skupina cestujících ve vozidle). (Poznámka: v tomto a v následujícím OPD je písmeno c podmínky nakresleno uvnitř kruhu)



Obr. 6.4 **Posílání pomoci** je provedeno a zobrazuje konkrétní vlákno provádění v průběhu. Lze jej sledovat podle stavu každého objektu.

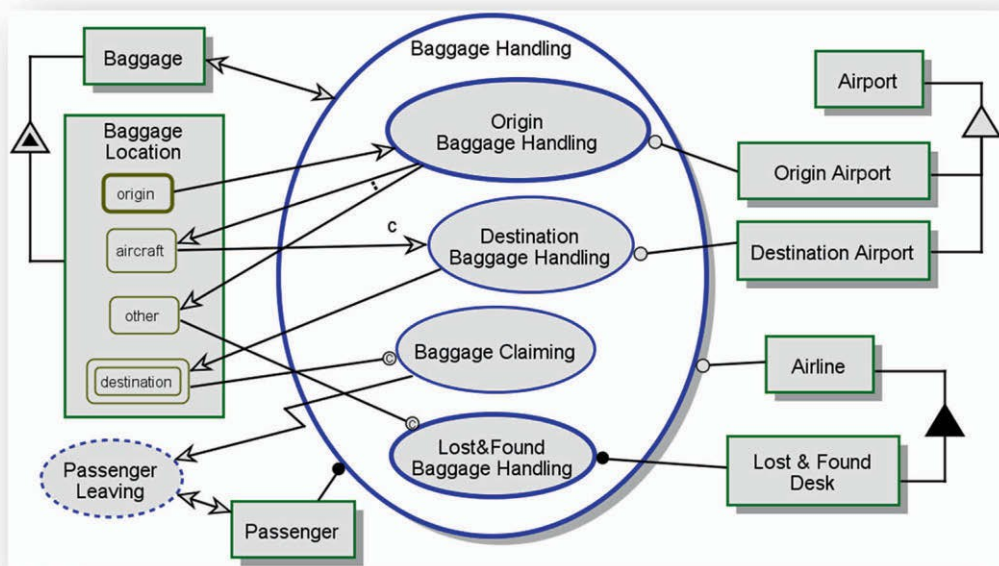
6.6 Shrnutí

- Sémantika podmíněného odkazu je taková, že pokud objekt, ke kterému je odkaz připojen, existuje, nebo pokud stav, ke kterému je odkaz připojen, je aktuálním stavem objektu, pak se proces provede, jinak se přeskočí.
- Vztah XOR mezi procedurálními odkazy naznačuje, že se uskuteční přesně jedna z možných interakcí označených těmito odkazy.
- XOR je graficky znázorněn společným bodem, ze kterého všechny odkazy XOR vycházejí nebo ve kterém končí, a přerušovanou křivkou procházející těmito odkazy, jejíž střed je společným bodem odkazů.
- Princip časové osy OPM stanoví, že časová osa v kontextu přiblíženého procesu je ve výchozím nastavení směřována od horní části přiblížené elipsy procesu k její spodní části.

- Pořadí provádění podprocesů je určeno výškou horních bodů elipsy podprocesů tak, že ten, který je nahoře, začíná jako první.
- Pokud je horní bod elipsy dvou nebo více podprocesů ve stejné výšce, v rámci předem definované tolerance, spustí se současně. Tímto způsobem se modeluje synchronizace procesů.

6.7 Problémy

Podívejme se na OPD na [obr. 6.5](#). Jedná se o OPD získané přiblížením procesu manipulace se zavazadly v SD na [obr. 2.5](#), nazvané „SD1 – manipulace se zavazadly přiblížená“.



Obr. 6.5 SD1 – Zvětšený proces manipulace se zavazadly, OPD získaný zvětšením procesu manipulace se zavazadly v SD na [obr. 2.5](#)

Manipulace se zavazadly zahrnuje (1) manipulaci se zavazadly v místě odletu – personál letecké společnosti odbavuje zavazadla cestujících a nakládá je do letadla na letišti odletu, (2) manipulaci se zavazadly v místě příletu – vykládání zavazadel na letišti příletu a (3) vrácení zavazadel cestujícím.

1. Modelujte všechny podpůrné prvky – činitele a nástroje – procesu **manipulace se zavazadly**.
2. Identifikujte tři dílčí procesy uvedené v rámečku výše. Který dílčí proces se objevuje v OPD, ale ne v popisu výše?

3. Jaké je vhodné strukturální propojení mezi **cestujícím** a **zavazadlem**, které by bylo v souladu s SD (obr. 2.5)? Přidejte jej do modelu.
4. **Umístění zavazadel** je atributem **zavazadel**, jehož hodnoty (stavy atributů) jsou různá možná umístění **zavazadel**. Jaké jsou čtyři hodnoty **umístění zavazadel** v modelu? Jaký je počáteční stav a jaký je konečný stav? Jak to můžete zjistit?
5. Přidejte k **zavazadlu** druhý atribut s názvem **Držitel zavazadla** se třemi hodnotami: **cestující**, **bezpečnost** a **letecká společnost**.
6. Ukažte, jak podprocesy **manipulace** se **zavazadly** mění hodnotu **držitel zavazadla**.
7. Jak **Zpracování zavazadel v místě původu** mění stav **Umístění zavazadel**?
8. Jaká je sémantika přerušované čáry mezi šipkami od **Zpracování zavazadel v místě původu** k **letadlem** a **ostatními**?

Kapitola 7

Řízení chování systému

Obrázek... odpovídá pojmu nebo paměťovému obrazu spojenému se slovy.

Schapiro (1996)

Řízení v kontextu koncepčního modelování je schopnost určovat tok procesů a způsob, jakým transformují objekty za různých podmínek a okolností. Několik řídicích struktur nám umožňuje určit, jak se systém bude chovat v průběhu času. Patří mezi ně booleovské objekty pro větvení a modifikátory řízení – indikátory podmínek a událostí, které se přidávají k procedurálním odkazům a rozšiřují jejich sémantiku. V této kapitole diskutujeme a ukazujeme, jak se řídicí struktury používají k modelování chování systému.

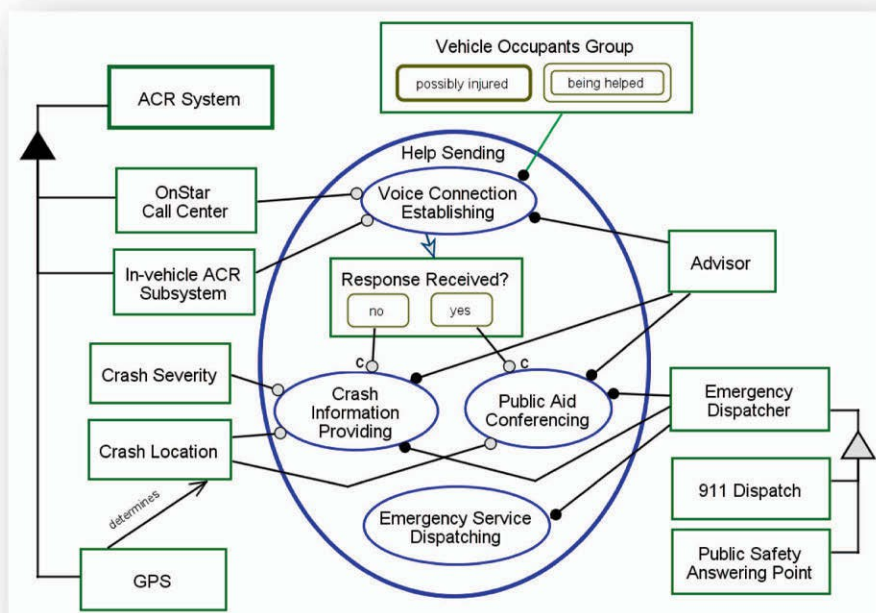
7.1 Větvení s booleovskými objekty typu „ „

Často potřebujeme specifikovat, že pokud platí určitá podmínka, uděláme něco, jinak uděláme něco jiného. V programovacích jazycích používáme příkazy „if ... then ... else“ nebo case. Například v našem příkladu ACR se zaměříme na první dílčí proces **Help Sending, Voice Connection Establishing**. Agenty tohoto procesu jsou **poradce** a **skupina cestujících ve vozidle**, zatímco jeho nástroje jsou **call centrum OnStar** a **mobilní systém**. Vrátime-li se k textu **Help Sending**, přečteme si znovu:

Je navázáno hlasové spojení mezi poradcem a cestujícími ve vozidle. Pokud cestující neodpovídají od pasažérů, poradce může poskytnout dispečerovi tísňové služby informace o nehodě...

To znamená, že musíme modelovat, zda **poradce** obdržel odpověď od **skupiny cestujících ve vozidle**, nebo ne. Potřebujeme rozvětvací mechanismus, jako je klauzule „if... then“. Řešením OPM pro tuto potřebu je *booleovský objekt*. Ve většině programovacích jazyků je *booleovská proměnná* logická proměnná, kterou lze reprezentovat jediným bitem, protože má přesně dvě hodnoty, obvykle nazývané *True (pravda)* a *False (nepravda)*. Podobně jako tento koncept je i *booleovský objekt* OPM informatickým objektem se dvěma stavy, který je generován procesem vyžadujícím rozhodování. Jinými slovy, booleovský objekt je objekt s dvěma stavy, který je generován jako výsledek procesu reagujícího na potřebu rozhodnout. Booleovský objekt je vytvořen v jednom ze dvou stavů. Tento stav je ten, který určuje rozhodnutí.

Booleovský objekt je rozhodovací objekt se dvěma stavy, který je generován jako výsledek procesu rozhodování o výběru přesně jednoho ze dvou možných stavů.



Obr. 7.1 Stavby booleovského objektu „**Response Received?**“ určují chování systému

Název booleovského objektu může končit otazníkem, v takovém případě se jeho OPL verze uvádí v uvozovkách, jako v případě „**Response Received?**“. Otázka by měla být formulována tak, aby v kombinaci s odpovědí pomocí odkazu na stav nebo odkazu na podmínku výsledná OPL věta dávala smysl, i když nemusí znít příliš přirozeně.

Jak ukazuje obr. 7.1, název našeho booleovského objektu je „**Response Received?**“ a má dva stavy, **ano** a **ne**. Každý z těchto stavů je propojen s podmíněným propojením, označeným písmenem c vedle kruhu (a ve starších verzích uvnitř) propojení nástroje. Každé podmíněné propojení směřuje k jednomu ze dvou alternativních procesů. Vzhledem k tomu, že podle definice stavu nemůže být objekt ve stejném časovém okamžiku ve více než jednom stavu, je „**Response Received?**“ buď **ano**, nebo **ne**, jak vyjadřuje následující věta OPL:

„**Odpověď přijata?**“ může být **ano** nebo **ne**.

Jelikož stavy „**ne**“ a „**ano**“ jsou spojeny s procesy „**Poskytování informací o nehodě**“ a „**Konzultace veřejné pomoci**“ pomocí podmíněných (nikoli instrumentálních) odkazů, může nastat přesně jeden z těchto dvou možných procesů: buď dojde ke **konzultaci veřejné pomoci** a **poskytování informací o nehodě** bude přeskočeno, nebo dojde k **poskytování informací o nehodě** a **konzultace veřejné pomoci** bude přeskočena. Takto jsou alternativní procesy modelovány v OPM. Na obr. 7.1 jsou tyto dva alternativní procesy ve stejné výšce, aby bylo zdůrazněno, že nastane jeden nebo druhý, ale ne oba. Jedná se o osvědčený postup, který usnadňuje

porozumění diagramu, ale tok řízení by byl stejný, i kdyby výšky elips alternativních procesů nebyly stejné.

Booleovský pár je pár inverzních názvů pro stavy nebo hodnoty booleovského objektu.

Existuje několik předdefinovaných booleovských párů, které lze začlenit do automatizovaného nástroje podporujícího OPM. Příklady zahrnují booleovské páry **true** a **false**, **pozitivní** a **negativní**, **černá** a **bílá**, **zapnuto** a **vypnuto**, **nahoru** a **dolů**, **černá** a **bílá**, **dobré** a **špatné**, **pravé** a **levé**, **nahore** a **dole**, **správné** a **nesprávné**, **správné** a **špatné**, **pravé** a **levé**, **sever** a **jih**, **vysoké** a **nizké**, **velké** a **malé**, **OK** a **ne OK**, **schváleno** a **zamítnuto**, **prošlo** a **neprošlo**, **větší než nebo rovno x** a **menší než x** (což lze také zapsat jako **>=x** a **<x**, kde x může být libovolné zadané číslo) a **větší než x** a **menší než nebo rovno x** (což lze také zapsat jako **>x** a **<=x**). Kromě těchto předdefinovaných booleovských párů může uživatel samozřejmě použít jakýkoli jiný booleovský pár, a to jednoduše zadáním názvů dvou stavů booleovského objektu.

Z metodologického hlediska není na booleovském objektu nic zvláštního, co by ho odlišovalo od jakéhokoli jiného informatického objektu, kromě toho, že má přesně dva stavy. V obecné rovině může mít objekt libovolný počet stavů (včetně nuly, v tom případě je bezstavový). Mít několik stavů je podobné jako „příkaz case“ v programovacích jazycích, protože každý stav může být nástrojem nebo podmínkou pro nějaký následný proces. Pokud od stavu vychází nástrojové propojení specifikované stavem, sémantika je „počkej, dokud objekt není ve specifikovaném stavu“. Pokud od stavu vychází podmíněné propojení, sémantika je „proved' proces, pokud je objekt ve specifikovaném stavu, jinak tento proces přeskoč“.

7.2 Podmíněné propojení versus nástrojové propojení

Dvě vazby pocházejí z booleovského objektu „**Response Received?**“ (Byla přijata odpověď?). Jedna je ze stavu „**ano**“ tohoto booleovského objektu k procesu **Public Aid Conferencing** (Konference o veřejné pomoci), zatímco druhá je ze stavu „**ne**“ tohoto objektu k procesu **Crash Information Providing** (Poskytování informací o nehodách). Tyto dvě vazby jsou vazby podmínky.

Jak ukazuje [obrázek 7.1](#), podmíněný odkaz je syntakticky (graficky) podobný odkazům na události, s tím rozdílem, že modifikátor řízení (písmeno uvnitř prázdného kruhu) je „c“ (pro podmínku) namísto písmene „e“, které je modifikátorem řízení odkazu na událost.

Sémanticky je podmíněné propojení podobné příkazu „if...then“: Pokud je objekt ve stavu, ze kterého podmíněné propojení vychází, pokusí se provést cílový proces, tj. proces, ke kterému je kruh propojení připojen. Pokud objekt není ve stavu propojeném s podmíněným propojením nebo pokud nejsou splněny všechny předpoklady pro výskyt tohoto procesu, proces se jednoduše přeskočí.

Předpoklady pro výskyt procesu zahrnují existenci všech spotřebitelů procesu (objektů, které proces potřebuje spotřebovat), ovlivněných objektů (objektů, jejichž stav proces mění), které musí být všechny ve svém vstupním stavu, a aktivátorů (agentů a nástrojů).

Tato sémantika je částečně vyjádřena dvěma větami OPL odpovídajícími dvěma podmínkovým odkazům:

Konference o veřejné pomoci se koná, pokud je odpověď na otázku „Byla přijata odpověď?“ kladná.

a

Poskytování informací o havárii nastane, pokud je „Odpověď přijata?“ ne.

Stejně jako nástrojové propojení může i podmíněné propojení pocházet spíše z bezstavového objektu (objektu bez stavů) než ze stavu objektu. V tomto případě, pokud objekt existuje (buď od začátku provádění systému, nebo proto, že byl vytvořen v určitém okamžiku během provádění systému), je proveden pokus o provedení cílového procesu, a pokud ne, je proces přeskočen.

Porovnáme-li podmíněné propojení s nástrojovým propojením, vidíme, že sémantika podmíněného propojení je slabší než sémantika nástrojového propojení. Je tomu tak proto, že v případě podmíněného propojení je spouštěný proces přeskočen, pokud nejsou splněny všechny předpoklady pro výskyt (spuštění) tohoto procesu. Naopak v případě nástrojového propojení se spuštění systému zastaví a čeká, dokud nejsou splněny všechny podmínky pro výskyt cílového procesu. V pojmech programovacího jazyka je podmíněný odkaz analogický k „if...then“, zatímco instrumentální odkaz je analogický k „wait until...“. Ve skutečnosti je vyhrazená fráze OPL pro podmíněný odkaz „**occurs if**“, zatímco pro instrumentální odkaz je to „**required**“.

7.3 Zobecnění-specializace

Podívejme se nyní na následující text:

*... Poradce se poté může spojit s **dispečinkem tísňové linky 911** nebo s **veřejným bezpečnostním dispečinkem (PSAP)**... Pokud od osob ve vozidle nedostane žádnou odpověď, může poradce poskytnout **dispečerovi tísňové linky** informace o nehodě.*

Zde jsou zmíněny tři pomocné entity: **dispečink tísňové linky 911**, **centrum tísňového volání** a **dispečer tísňové linky**. Z textu je zřejmé, že entita, která získává informace o nehodě, **dispečer tísňové linky**, je **zobecněním dispečinku tísňové linky 911** i **centra tísňového volání**. Naopak **dispečink tísňové linky 911** i **centrum tísňového volání** jsou **specializacemi dispečera tísňové linky**.

Zobecnění-specializace je silný strukturální vztah, který umožňuje abstrahovat libovolný počet objektů nebo tříd procesů do supertříd. Syntakticky je vztah zobecnění-specializace bílý trojúhelník, jehož vrchol je spojen se zobecňujícím odkazem a jehož základna se specializujícími odkazy. Na [obr. 7.1](#) je tento odkaz znázorněn jako spojení obecného **dispečera tísňové linky** se dvěma specializacemi, **dispečinkem tísňové linky 911** a **centrem tísňového volání**. Fráze OPL vyjadřující tento vztah je „je“ (nebo „je“). Následující věty OPL to vyjadřují:

911 Dispatch je **dispečer tísňové linky**.

Tísňové centrum veřejné bezpečnosti je **dispečer tísňové linky**.

Stručněji lze tyto dvě věty vyjádřit jednou větou:

911 Dispatch a **Public Safety Answering Point** jsou **dispečinky tísňových služeb**.

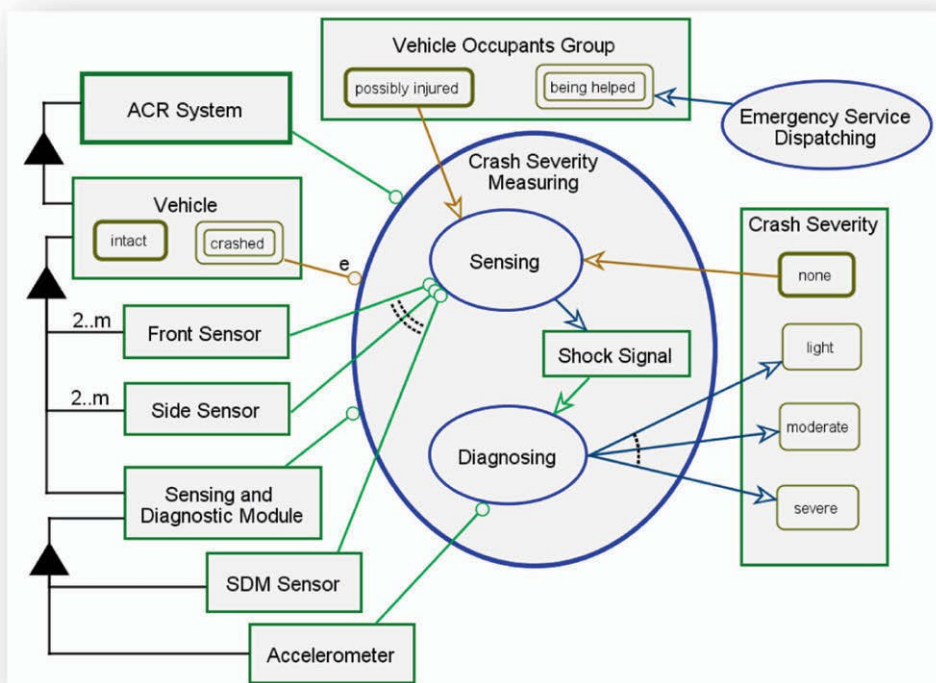
Sémanticky vede spojení generalizace-specializace k dědičnosti vlastností, stavů a spojení z generalizující nadřazené třídy – obecné – do jejích podtříd – specializací. Například jediné spojení agenta z **Emergency Dispatcher** do **Emergency Service Dispatching** na [obr. 7.1](#) je zděděno jak **911 Dispatch**, tak **Public Safety Answering Point**. Toto je příklad síly generalizace a dědičnosti, kterou vyvolává: namísto nakreslení šesti spojení agentů z **dispečinku tísňové linky 911** a **centra tísňového volání** do každého ze tří spodních podprocesů na [obr. 7.1](#) jsou nakreslena pouze tři, ale jsou interpretována jako šest.

7.4 Zvětšení měření závažnosti nehody ()

Některé specifikace systému jsme vynechali na začátku textu, takže tato část ještě není modelována. Vraťme se zpět a dokončíme model na základě toho, co jsme četli:

Systém ACR využívá přední a boční senzory, stejně jako snímání schopnosti samotného snímáního a diagnostického modulu (SDM). Akcelerometr umístěný v SDM měří závažnost nehody.

Obr. 7.2 Zvětšení měření závažnosti nárazu



Zde se zaměříme na použití senzorů k měření závažnosti nárazu s objekty, které jsme dosud do našeho modelu nezahrnuli. Proces **měření závažnosti nárazu** jsme již modelovali jako první dílčí proces **automatické reakce na náraz** (viz [obr. 6.3](#)) a **modul snímání a diagnostiky** jako nástroj tohoto procesu. Abychom tedy přidali podrobnosti, jako jsou různé senzory a jejich snímací procesy, musíme nyní přiblížit **měření závažnosti nárazu**.

Obrázek 7.2 je OPD, ve kterém je zvětšeno **měření závažnosti nehody**, a ukazuje, že se skládá ze dvou dílčích procesů: **snímání** a **diagnostika**, které se provádějí postupně v pořadí shora dolů podle jejich výskytu ve zvětšeném procesu: nejprve **snímání**, poté **diagnostika**. Není překvapením, že **modul snímání a diagnostiky** je nástrojem pro oba tyto dílčí procesy. Proto je propojení nástroje z

tohoto objektu se dotýká vnějšího procesu **měření závažnosti nehody** a funguje jako závorky v algebře, které označují, že se vztahuje na všechny dílčí procesy v něm.

7.5 Omezení účasti

Nástroji **snímání** jsou také přední a boční senzory, stejně jako senzor uvnitř **modulu snímání a diagnostiky**, který modelujeme jako **senzor SDM**. Protože počet předních a bočních senzorů není specifikován a víme pouze to, že je jich více než jeden každého druhu, modelujeme **vozidlo** (které je zase součástí **systému ACR**) tak, že má dva až mnoho objektů třídy **přední senzor** a dva až mnoho objektů třídy **boční senzor**. V OPD na [obr. 7.2](#) je to vyjádřeno *omezením účasti*

2..m se zobrazuje vedle **předního senzoru** a **bočního senzoru**. Odpovídající věta OPL je:

Vozidlo se skládá z **modulu pro snímání a diagnostiku**, **2** až mnoho **předních senzorů** a **2** až mnoho **bočních senzorů**.

7.6 Logické operátory: OR versus XOR

Proces **snímání** nepotřebuje všechny ani některé senzory k vygenerování **signálu nárazu** – stačí jeden. **Signál nárazu** však může vygenerovat více než jeden senzor. To je definice logického operátoru OR. OR je sémanticky volnější než jeho protějšek XOR, protože umožňuje jeden nebo více vstupů nebo výstupů namísto přesně jednoho. XOR vyžaduje, aby byla vybrána přesně jedna z několika alternativ. Například, jak ukazuje [obr. 7.2](#), logická operace XOR mezi třemi spojeními stejného typu od **diagnostiky** ke třem stavům – **mírný**, **střední** a **závažný** – je vyjádřena jediným přerušovaným obloukem, jehož střed je společným počátkem těchto spojení. Zatímco XOR je označen jednou přerušovanou křivkou, OR je označen dvěma přerušovanými křivkami, jak ukazuje tři propojení přístrojů končící ve stejném bodě na elipse procesu **snímání** na [obr. 7.2](#).

Věta OPL, která vyjadřuje operátor OR, je jednoduchá:

Sensing vyžaduje **SDM senzor**, **boční senzor** nebo **přední senzor**.

Porovnáme-li to s XOR ve stejném OPD, vidíme, že XOR je vyjádřeno vyhrazenou frází OPL „přesně jeden z“, jak je uvedeno v následující větě OPL.

Diagnostika mění **závažnost havárie** na přesně jednu z následujících: **lehká**, **střední** nebo **těžká**.

7.7 Míra závažnosti nehody Měřeno pomocí Vylepšeno

Při pečlivějším čtení specifikace si všimneme, že jsme nemodelovali následující větu:

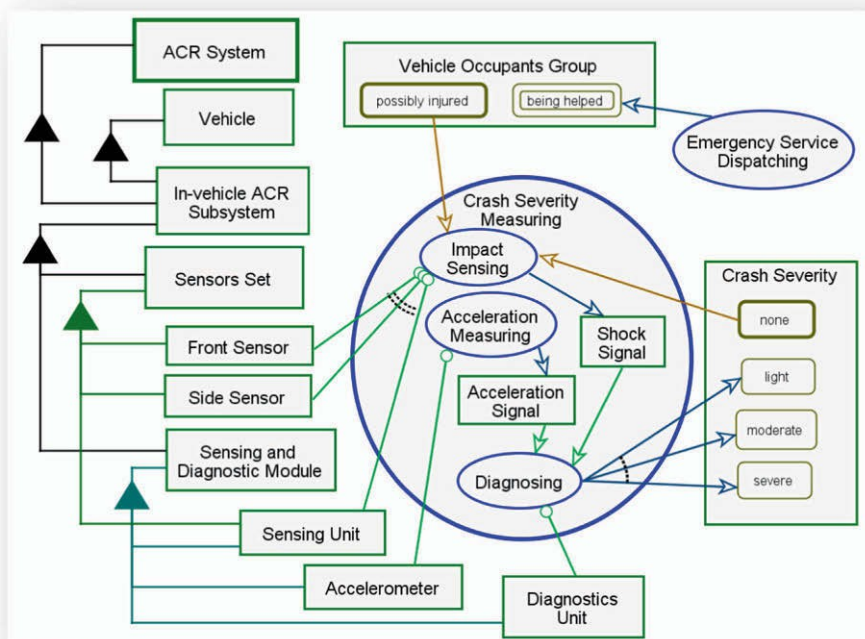
Akcelerometr umístěný v SDM měří závažnost nehody.

Místo toho jsme na [obr. 7.2](#) modelovali **akcelerometr** jako nástroj pro dílčí proces **diagnostiky**. Z jeho názvu usuzujeme, že funkci **akcelerometru** je měření zrychlení, takže i když to není

výslovně uvedeno, na [obr. 7.3](#) přidáváme proces **Měření zrychlení**, jehož nástrojem je **akcelerometr** a jehož výsledkem je **Měření zrychlení** – objekt, který je výsledkem tohoto procesu.

7.8 Rozsah věcí: signál jako dočasný objekt typu „“

Při prohlížení obsahu zvětšeného procesu **měření závažnosti nárazu** na [obr. 7.3](#) si uvědomíme, že kromě dvou procesů obsahuje také dva objekty, **signál nárazu** a **signál zrychlení**. To se odráží také v následujících třech odpovídajících větách OPL. První z nich je:



Obr. 7.3 Změření závažnosti nárazu upřesněno

Crash Severity Measuring se zaměřuje na **Impact Sensing**, **Acceleration Measuring** a **Diagnosing** v tomto pořadí, stejně jako na **Acceleration Signal** a **Shock Signal**.

Tato věta uvádí tři dílčí procesy, které jsou popsány v detailním **měření závažnosti nehody**: **snímání nárazu**, **měření zrychlení** a **diagnostika**. Vyhrazená fráze „v tomto pořadí“ označuje, že pořadí, v jakém jsou dílčí procesy uvedeny, je pořadím jejich provádění. Za tímto seznamem procesů následuje vyhrazená fráze OPL „stejně jako“, za kterou následuje seznam dvou obsažených objektů: **signál nárazu** a **signál zrychlení**. Vyhrazená fráze OPL „stejně jako“

odděluje seznam procesů a seznam objektů v přiblíženém procesu. Podprocesy jsou součástí přiblíženého procesu, zatímco objekty jsou atributy tohoto přiblíženého procesu. U zvětšeného objektu by pořadí bylo opačné: nejprve by byl uveden seznam objektů, poté fráze „as well as“ a nakonec seznam procesů. V tomto případě jsou interní objekty součástí zvětšeného objektu, zatímco interní procesy jsou operacemi tohoto zvětšeného objektu.

Dva informatické objekty **Shock Signal** a **Acceleration Signal** jsou vytvořeny uvnitř procesu **Crash Severity Measuring** dvěma jeho dílčími procesy. Poté jsou okamžitě spotřebovány třetím dílčím procesem **Diagnosing** a zmizí. Obecně platí, že objekty uvnitř přiblíženého procesu jsou dočasné: existují a jsou rozpoznávány výhradně v rámci tohoto procesu. To by platilo i v případě, že bychom místo dvou spotřebních odkazů použili dva nástrojové odkazy. Pokud chceme tyto objekty zachovat, musí se nacházet mimo přiblížený proces.

7.9 Jak se provádí diagnostik ?

Když jsme změnili určení **akcelerometru** na nástroj pro měření **zrychlení**, odstranili jsme **Diagnostika** mimo přístroj. Jak se tedy **diagnostika** provádí?

Když se zamyslíme nad názvem „**Senzorický a diagnostický modul**“ jako nad částí systému, která má senzorické schopnosti, a nad **akcelerometrem** jako jednou z jeho součástí, dojdeme k závěru, že **senzorický a diagnostický modul** musí obsahovat také část, která je zodpovědná za diagnostiku. Proto modelujeme **snímač a diagnostický modul** jako sestávající ze tří částí: **akcelerometru** a dvou dalších částí. Jednou z nich je **snímač** (který byl na [obr. 7.2](#) označen jako **SDM senzor**) a druhou je **diagnostická jednotka**. Tato struktura je modelována na [obr. 7.3](#) a je vyjádřena následujícím OPL příkazem:

Senzorický a diagnostický modul se skládá z **akcelerometru**, **senzorické jednotky** a **diagnostické jednotky**.

Při pohledu na [obr. 7.3](#) vidíme, že **snímací jednotka** je součástí nejen **snímacího a diagnostického modulu**, ale také nově zavedeného objektu **Sada senzorů**. Další věta OPL v našem odstavci OPL zní:

Snímač a diagnostický modul se skládá z **akcelerometru**, **snímače** a **diagnostické jednotky**.

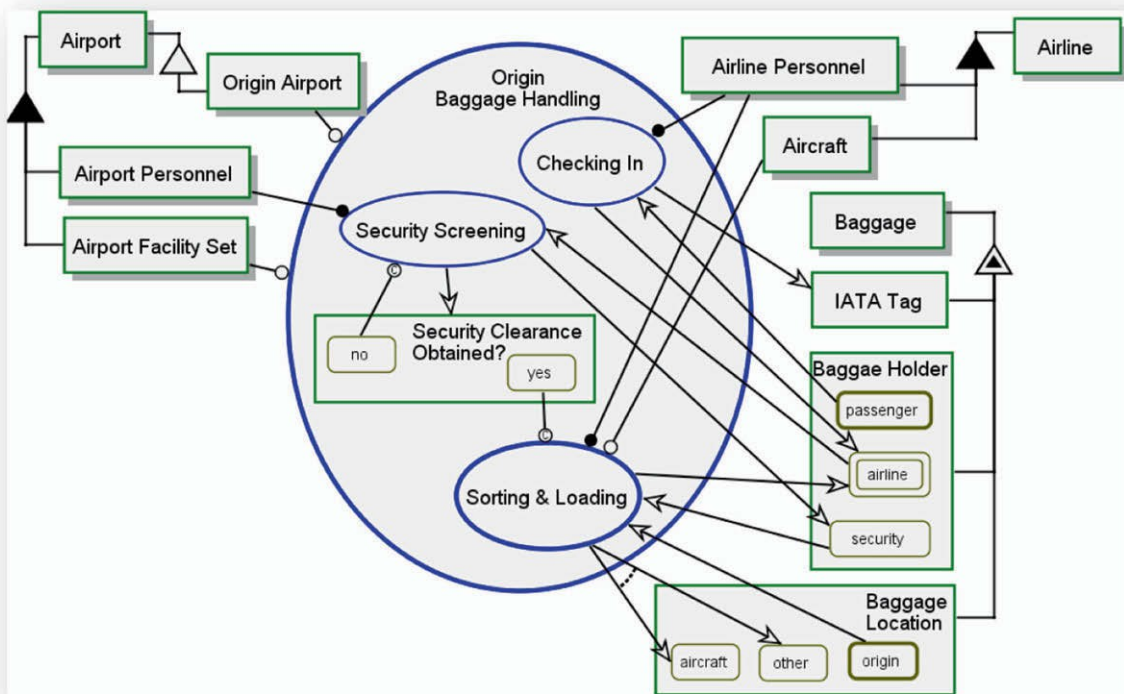
7.10 Shrnutí

- Booleovský objekt má dva stavy a používá se k modelování toku řízení.
- Podmínkový odkaz má k procedurálnímu odkazu přidán modifikátor řízení c, který rozšiřuje jeho sémantiku o význam přeskočení.
- Generalizace-specializace je základní strukturální vztah, který indukují dědičnost vlastností (atributů a operací), odkazů a stavů od obecné věci ke specializované věci.
- Omezení účasti umožňují specifikovat, kolik objektů stejné třídy se účastní vztahu.

- OR je uvolněná verze logického operátoru XOR, která umožňuje, aby byla aktivní libovolná podmnožina účastníků se odkazů najednou, namísto přesně jednoho z nich, jak to dělá XOR.
- Objekty v přiblíženém procesu jsou rozpoznány pouze v rámci tohoto procesu. Pokud jsou použity mimo tento rozsah, musí být umístěny mimo přiblížený proces.

7.11 Problémy

Obrázek 7.4 je SD1.1 – Zvětšený pohled na zpracování zavazadel v místě původu. Jedná se o OPD získaný zvětšením procesu **zpracování zavazadel v místě původu** v SD1 (obr. 6.5).



Obr. 7.4 SD1.1 – Zvětšený pohled na zpracování zavazadel v místě původu

1. **Při odbavení** se vytvoří (vznikne) objekt – co to je?
2. **Check-in** také ovlivňuje jiný objekt – co to je? Jaké jsou vstupní a výstupní stavy **Check-in**?
3. Kdo je agentem **bezpečnostní kontroly**?
4. Jaký objekt vytváří **bezpečnostní kontrola**?
5. O jaký typ objektu se jedná?
6. Vysvětlete, jak tento objekt řídí větvení.
7. Jaký podproces ovlivňuje **umístění zavazadel**?
8. Najdete v OPD smyčku? Popište ji.
9. Jak je možné, že na [obr. 6.5](#) proces „**Origin Baggage Handling**“ (**Zpracování zavazadel v místě odletu**) změnil „**Baggage Location**“ (**Umístění zavazadel**) z místa odletu na **letadlo** nebo jiné místo, zatímco zde, na [obr. 7.4](#), proces „**Sorting & Loading**“ mění stejný objekt, **Baggage Location**, také z **původního umístění** na **letadlo** nebo jiné?

Kapitola 8

Abstrakce a zdokonalování

Vše udělejte tak jednoduché, jak je to možné, ale ne jednodušší.
Albert Einstein

Dosud jsme vždy zvyšovali úroveň detailů našeho modelu a dělali jsme to pomocí přiblížení procesů. Existují však případy, kdy potřebujeme úroveň detailů snížit, nebo jinými slovy, model abstrahovat. K tomu může dojít, když si uvědomíme, že do jednoho diagramu je vtěsnáno příliš mnoho detailů, což ho činí příliš přeplněným a tudíž méně srozumitelným. Nechceme mazat detaily modelu, protože jsou důležité pro kompletní specifikaci systému. Chceme je však odstranit z konkrétního přeplněného diagramu. Toho dosáhneme vytvořením nového OPD na střední úrovni detailu, a to oddálením příliš detailního OPD a vytvořením nového na vyšší úrovni abstrakce. V této kapitole se zaměříme na tento proces abstrahování a poté budeme diskutovat a vylepšovat strukturální pohled na systém.

8.1 : Zpřesnění procesu v novém diagramu OPD

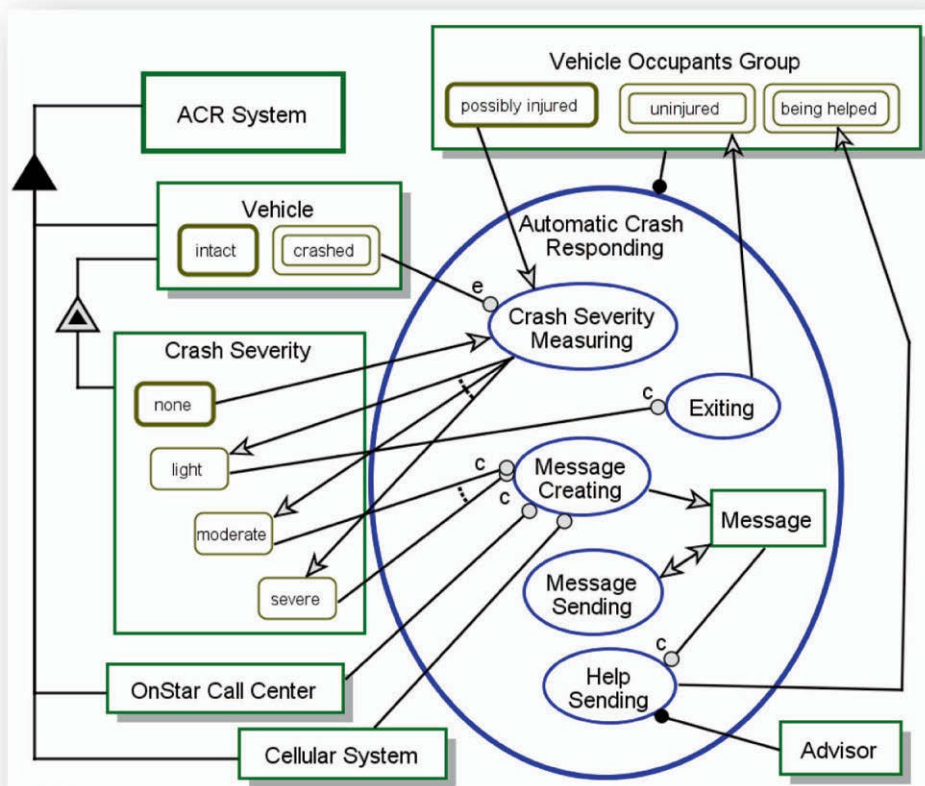
Přečtěte si pozorně větu:

Bez ohledu na to, zda se airbagy aktivují, SDM [Sensing and Diagnostic Module] přenáší informace o nehodě do modulu OnStar vozidla.

Vypadá to, že airbagy nejsou v našem modelu opravdu nezbytné. Při bližším prozkoumání věty však zjistíme, že v našem modelu chybí dílčí proces přenosu informací o nehodě z **modulu SDM** do **modulu OnStar**, který je zjevně další součástí **systému ACR** umístěného uvnitř **vozidla**, který jsme dosud nemodelovali.

Přirozeným místem pro přidání objektu **modulu OnStar** a procesu **přenosu informací o nehodě** je OPD na [obr. 6.2](#), který je pro větší přehlednost znovu zobrazen na [obr. 8.1](#). Jak vidíme, tento OPD je již přeplněný, takže přidání **modulu OnStar** jako objektu a **přenosu informací o nehodě** jako pátého dílčího procesu uvnitř **automatické reakce na nehodu** by jej ještě více zkomplikovalo a učinilo ještě méně srozumitelným. Důležitým cílem modelování OPM je udržet každý OPD dostatečně přehledný a čitelný, aby čtenář diagramu nebyl zahlcen. Musíme tedy přijít na způsob, jak přidat nové prvky, aniž bychom tento nebo jakýkoli jiný OPD příliš zkomplikovali.

Při zkoumání čtyř dílčích procesů na [obr. 8.1](#) si všimneme, že dva prostřední, **Vytváření zprávy** a **Odesílání zprávy**, mají podobnou povahu jako nový dílčí proces, který chceme zavést, a to **Přenos informací o havárii**. Řešením bude tedy sloučení **procesů Vytváření zprávy** a **Odesílání zprávy** do nového dílčího procesu, který nazveme **Zpracování zprávy**. Poté se zaměříme na tento nový proces v novém, samostatném OPD, kde odhalíme tři dílčí procesy: **Vytváření zprávy**, **Odesílání zprávy** a **Přenos informací o selhání**. Sloučení **procesů Vytváření zprávy** a **Odesílání zprávy** vede k *oddálení* procesu, při kterém jsou dva nebo více procesů abstrahovány do procesu vyšší úrovně.

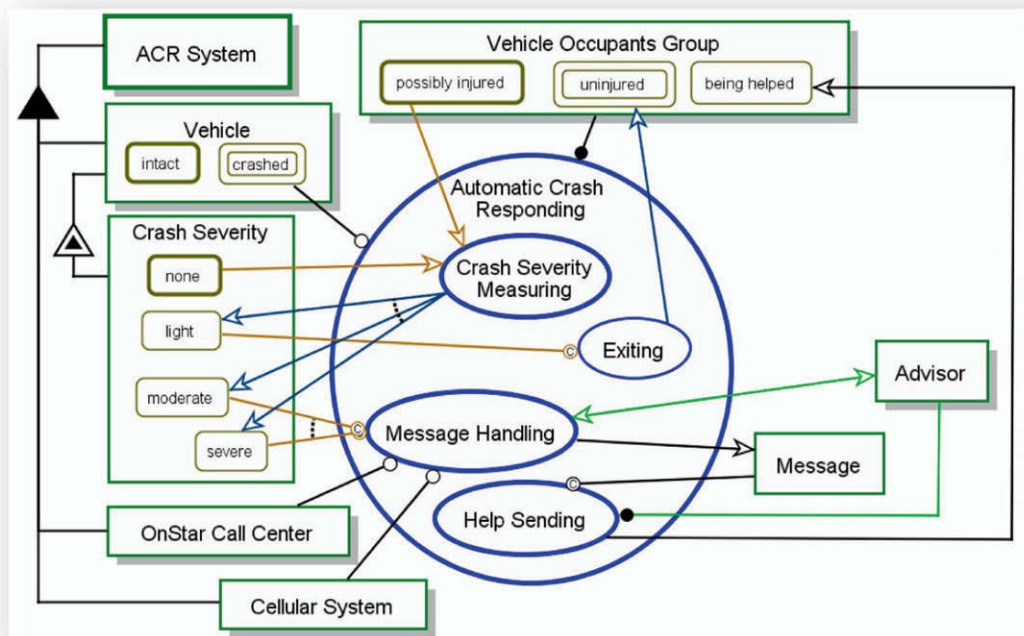


Obr. 8.1 Vytváření zpráv a odesílání zpráv budou brzy oddáleny a nahrazeny zpracováním zpráv

To má ještě jednu výhodu: na [obr. 8.2](#) definujeme agregovaný objekt nazvaný **Podsystém ACR ve vozidle** jako součást **Vozidla**. Díky tomu můžeme nyní modelovat pouze **In-vehicle ACR Subsystem** jako součást **ACR System**, místo modelování celého **Vehicle** jako součásti **ACR System**. Tento nový objekt **In-vehicle ACR Subsystem** se skládá z **OnStar Module** a všech ostatních objektů uvnitř **Vehicle**, které jsou součástí **ACR System**. Tato úprava dále zjednodušuje OPD. [Obrázek 8.2](#) skutečně vypadá jednodušeji než jeho předchozí verze na [obr. 8.1](#).

Tato zjednodušená verze nám umožňuje vysvětlit vztah mezi **poradcem** a **call centrem OnStar**, aniž by byl příliš komplikovaný. Přidáme strukturální odkaz s tagem **operates from**, čímž získáme následující větu v jazyce OPL:

Poradce pracuje z call centra OnStar.



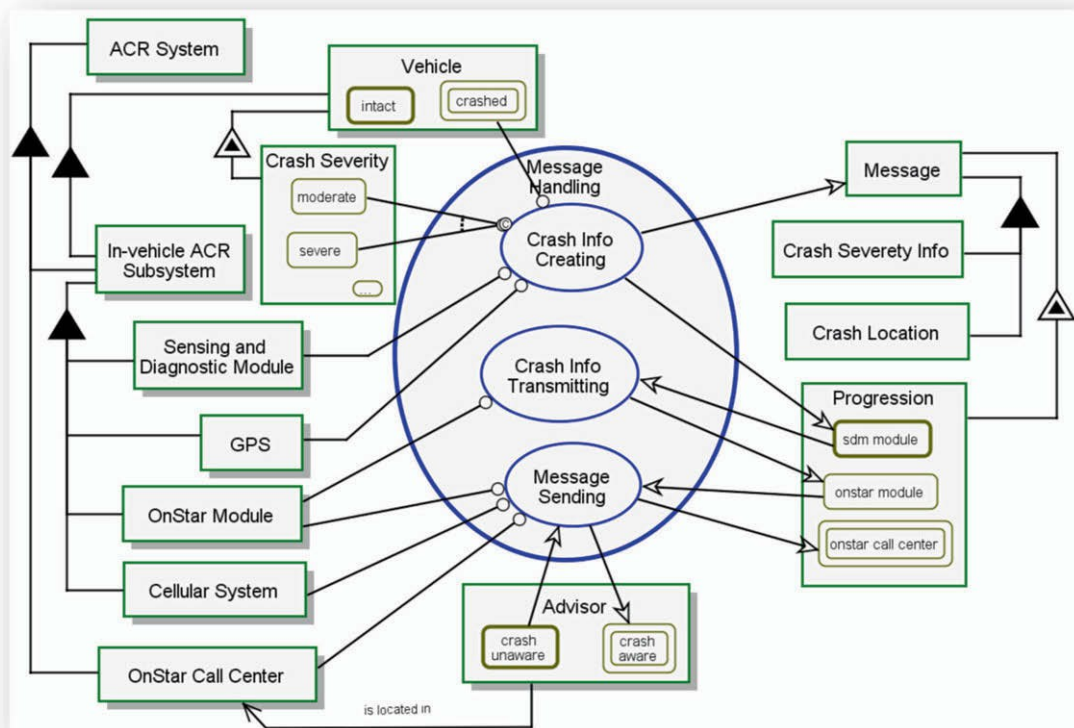
Obr. 8.2 Abstrahování vytváření zpráv a odesílání zpráv z obr. 8.1 do zpracování zpráv. Barvy odkazů usnadňují pochopení OPD a zvyrazňují odkazy c stavu nástroje uvnitř kruhu.

8.2 Zpracování zprávy v režimu In-

Obrázek 8.3 představuje OPD, ve kterém je přiblíženo zpracování zpráv. Přejmenujeme vytváření zpráv na vytváření informací o nehodě. Jako dva XOR odkazy událostí ze stavů střední a závažné závažnosti nehody je tento proces spuštěn buď střední, nebo závažnou nehodou.

Na obrázku 8.3 jsou znázorněny pouze dvě ze čtyř hodnot: střední a závažná. Aby čtenář diagramu věděl, že existují další hodnoty, které zde nejsou zobrazeny, je v dolní části závažnosti nehody přidán symbol „alespoň jeden další stav“ – malý symbol stavu s elipsou (tři tečky). Odpovídající věta OPL zní:

Závažnost nehody může být střední, závažná nebo alespoň jeden další stav.



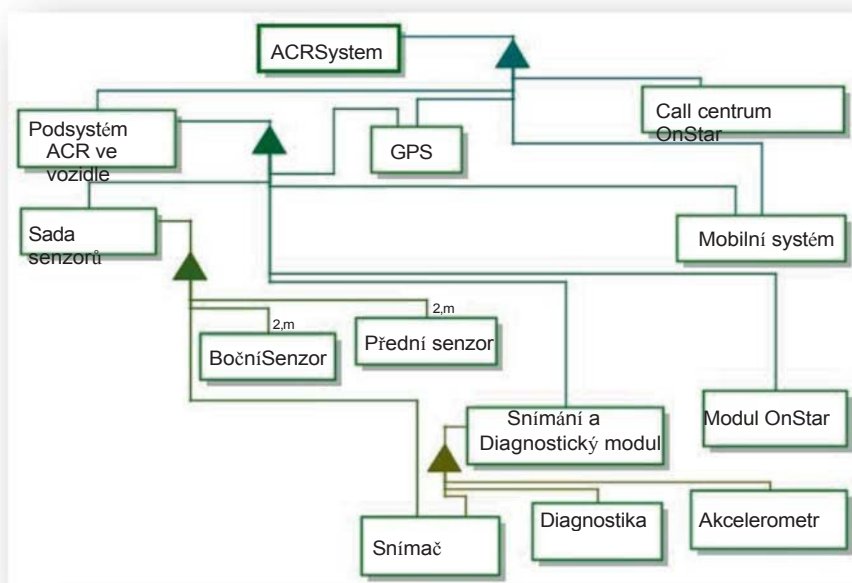
Obr. 8.3 Přiblížení zpracování zpráv

Když je proces, jako je **Zpracování zpráv**, přiblížen, zpočátku neexistují žádné vnitřní podprocesy, takže všechny procedurální odkazy, které začínají nebo končí v přiblíženém procesu, jsou umístěny podél obrysu elipsy tohoto procesu. Jak modelář specifikuje vnitřní podprocesy, každý z těchto odkazů musí být přesunut (v terminologii GUI je třeba přetáhnout jeho konec procesu) do příslušného podprocesu. Postupně se všechny odkazy obklopující nadřazený přiblížený proces přesouvají dovnitř, až žádný z nich není propojen s nadřazeným procesem, jak je znázorněno na [obr. 8.3](#). To by mělo být provedeno, pokud se odkaz nevztahuje na všechny podprocesy uvnitř přiblíženého procesu, v takovém případě by měl být ponechán na místě. Odkaz dotýkající se nadřazeného procesu by měl být propojen s každým z podprocesů uvnitř tohoto procesu. Příklad je uveden na [obr. 8.2](#), kde **havarované vozidlo** je nástrojem všech čtyř podprocesů uvnitř **automatické reakce na havárii**.

Proces **Message Creating** vytváří informační objekt **Message**, který se skládá ze dvou částí: **Crash Severity Info** a **Crash Location**. **Crash Severity Info** je vytvořen modulem **Sensing and Diagnostic Module**, zatímco **Crash Location** je vytvořen modulem **GPS**. Tyto dva objekty jsou proto modelovány jako nástroje **Crash Info Creating**. Tyto podrobnosti o tom, který modul vytváří kterou část, nejsou na této úrovni modelovány; budou zobrazeny na další úrovni níže, když bude přiblížen proces **Message Creating**.

8.3 Strukturální pohled na systém ACR ()

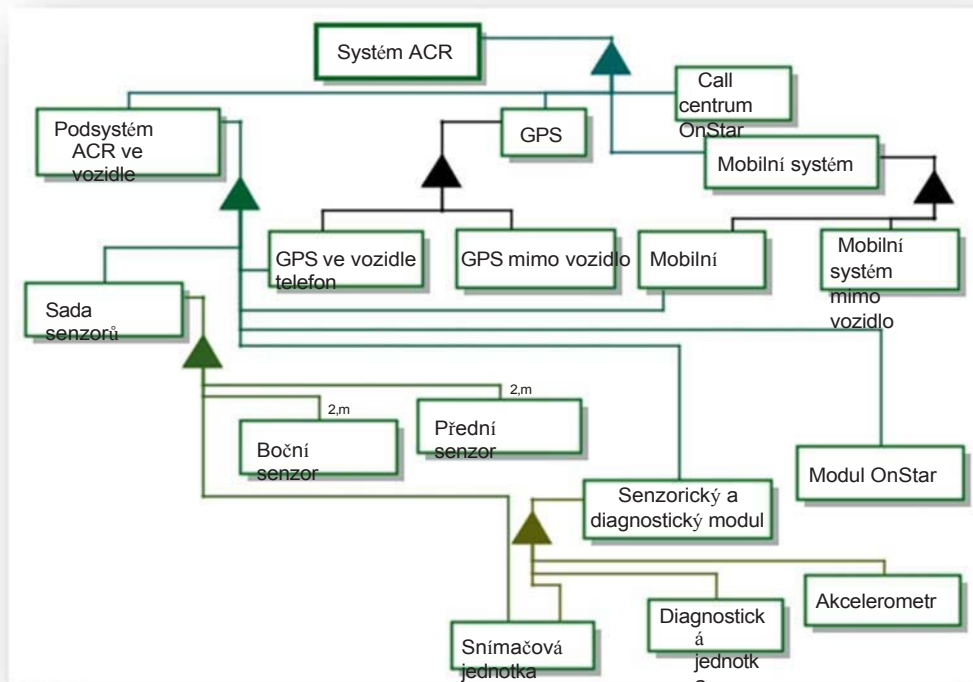
Jak ukazuje [obrázek 8.5](#), struktura **systému ACR** prošla několika změnami. Bylo by užitečné prozkoumat celou strukturu samostatně, bez dynamických aspektů procesů a přechodů stavů. OPCAT poskytuje takovou automatickou funkci. [Obrázek 8.4](#) ukazuje automaticky generovaný *strukturální pohled* na **systém ACR** po ručním přeskupení pro lepší čitelnost. Je zde zobrazena čtyřúrovňová hierarchie, která je také vyjádřena v následujících větách OPL, kde odsazení pomáhá realizovat hierarchii.



Systém ACR se skládá z call centra OnStar, mobilního systému, GPS a subsystému ACR ve vozidle. **Subsystém ACR ve vozidle** se skládá ze snímacího a diagnostického modulu, mobilního systému, GPS, modulu OnStar a sady senzorů. **Snímač a diagnostický modul** se skládá z akcelerometru, snímače a diagnostické jednotky. **Sada senzorů** se skládá z 2 až více předních senzorů, 2 až více bočních senzorů a snímače.

Obr. 8.4 Automaticky generovaná strukturální hierarchie systému ACR

Při zkoumání OPD a odpovídajícího OPL vyčnívají dva objekty, které je třeba přepracovat: **GPS** a **mobilní systém**. Důvodem je, že každý z těchto objektů je součástí jak **subsystému ACR**, tak **subsystému ACR ve vozidle**. **Podsystem ACR ve vozidle** je však také součástí **systému ACR**. I když to není rozpor, jedná se o nesrovnalost, protože **GPS** a **mobilní systém** jsou přímou i nepřímou součástí **systému ACR**. Jak víme, ani **GPS**, ani **mobilní systém** jako celek nejsou součástí **podsystemu ACR ve vozidle**; každý z nich má komponenty uvnitř i vně vozidla.



Obr. 8.5 Strukturně hierarchické uspořádání systému **ACR** po vyřešení nesrovnalostí s GPS a mobilním systémem

Řešením této nesrovnalosti, které je znázorněno na [obr. 8.5](#), je rozdělení každého z těchto dvou objektů na dvě části: **GPS** se rozdělí na **GPS ve vozidle** a **GPS mimo vozidlo**, zatímco **mobilní systém** se rozdělí na **mobilní telefon** a **mobilní systém mimo vozidlo**. **GPS ve vozidle** i **mobilní telefon** jsou součástí **subsystému ACR ve vozidle**, zatímco **GPS mimo vozidlo** a **mobilní systém mimo vozidlo** jsou součástí **systému ACR**, ale ne **subsystému ACR ve vozidle**.

8.4 Shrnutí

- Zatímco cílem modelování založeného na OPM je postupovat shora dolů a postupně zpřesňovat fakta modelu, aby se zabránilo nepřehlednosti diagramu, je někdy nutné abstrahovat dva nebo více procesů v přeplněném OPD a vytvořit nový OPD na prozatímní úrovni.
- Abstrakce může být dosažena procesním oddálením: Vytvořením abstraktního procesu, který po přiblížení bude zahrnovat oddálené podprocesy (a případně i další).
- Hned po přiblížení procesu zůstávají všechny procedurální odkazy k němu připojeny.

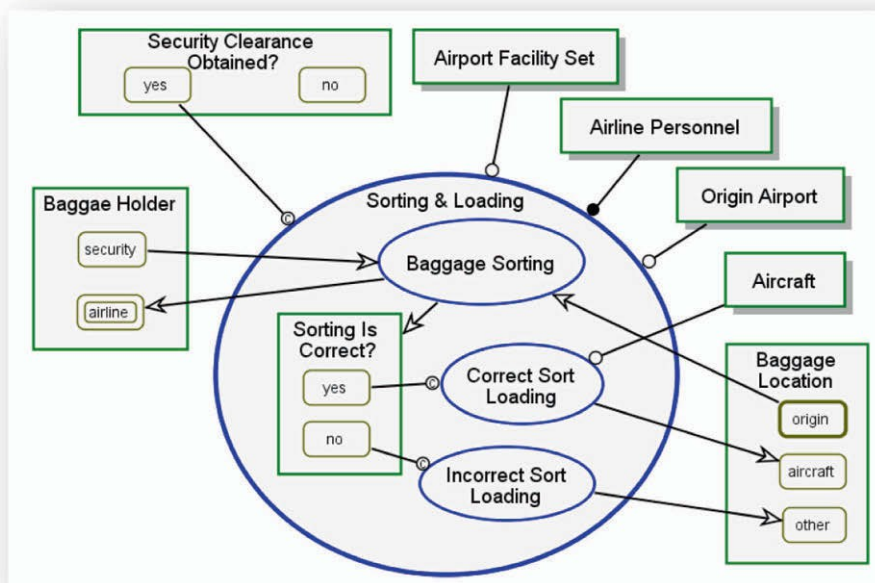
- Při přidávání podprocesů je třeba přetáhnout hrany procedurálních odkazů ze zvětšeného procesu do příslušných podprocesů.
- K zvětšenému procesu by měly zůstat připojeny pouze odkazy, které se vztahují ke všem podprocesům uvnitř zvětšeného procesu.
- Strukturální pohled se dosáhne odstraněním všech procesů a procedurálních odkazů z modelu.
- Strukturální pohled umožňuje soustředit se na strukturu systému a zkoumat možné strukturální vylepšení.

8.5 Problémy

Během **třídění a nakládání** provádí **personál letecké společnosti** **třídění zavazadel** a mění **držitel zavazadel** z **bezpečnostní služby** na **leteckou společnost**. Tento proces **třídění zavazadel** může vést ke správnému nebo nesprávnému třídění. Pokud je třídění správné, dojde k naložení **zavazadel** do správného **letadla**, takže se **umístění zavazadel** změní z **místa původu** na **letadlo**. V opačném případě se při nakládání nesprávně roztríděných zavazadel změní jejich **umístění** z **místa původu** na **jiné místo**.

V OPD SD 1.1.1 na [obr. 8.6](#) je zvětšeno **třídění a nakládání** z [obr. 7.4](#). S odkazem na [obr. 8.6](#) a [obr. 7.4](#) odpovězte na následující otázky.

1. Co je to „**správné třídění**“? K čemu se používá?
2. **Správné třídění a nakládání** je nad **nesprávným tříděním a nakládáním**. Znamená to, že první proces bude vždy proveden před druhým? Vysvětlete.
3. Mohou se správné třídění a načítání i nesprávné třídění a načítání vyskytnout ve stejném provedení sortování a načítání? Vysvětlete.
4. Je možné, že když je umístění zavazadel jiné, je držitel zavazadel bezpečnostní pracovník?
5. Za jakých podmínek probíhá proces **třídění a nakládání**? Vysvětlete.
6. Proč se v OPD zobrazuje pouze **letišť odletu** a ne **letišť příletu**?
7. Na [obr. 7.4](#) existuje vztah XOR mezi stavy **letadla** a **ostatními** stavy **umístění zavazadel**. Když **Třídění a nakládání** je zvětšeno na [obr. 8.6](#), tento vztah XOR se nezobrazuje. Je to v pořádku? Vysvětlete.
8. Proč je vztah XOR mezi stavy **letadla** a **ostatními** stavy **umístění zavazadel** potřebný na [obr. 7.4](#)?
9. Které dvě propojení přístrojů končí u **třídění a nakládání**? Vysvětlete význam tohoto jevu a proč je **letadlo** propojeno pouze se **správným tříděním a nakládáním**?

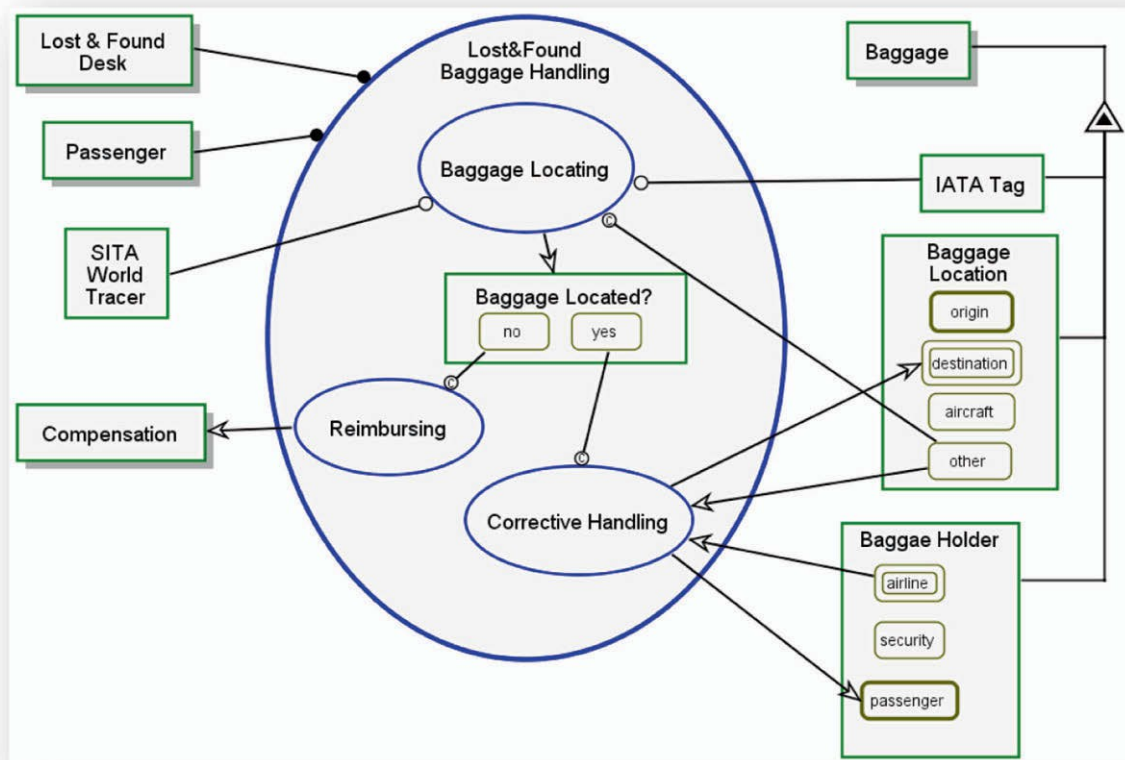


Obr. 8.6 SD1.1.1 – Třídění a nakládání v detailním zobrazení

Pokud se zavazadlo nenachází na letišti v místě určení (**místo určení zavazadla je jiné**), dojde k **vyřízení ztráty a nálezů**. **Pult ztrát a nálezů** používá k vyhledání zavazadla systém **SITA World Tracer** a **štítek IATA**. Pokud je zavazadlo nalezeno, dojde k **nápravnému vyřízení**, v opačném případě je cestujícímu vyplacena náhrada.

V OPD SD1.2 na [obr. 8.7](#) je přibliženo **zpracování ztracených a nalezených zavazadel**. S odkazem na [obr. 8.7](#) odpovězte na následující otázky.

10. Jaké jsou atributy **zavazadel**?
11. Za jakých podmínek dochází k **lokalizaci zavazadel**?
12. Jaké jsou nástroje pro **lokalizaci zavazadel**?
13. Jaký druh věcí se **lokalizuje**?
14. Mohou se při stejném provedení zpracování ztracených a nalezených zavazadel provést jak náhrada, tak nápravné opatření? Vysvětlete.
15. Jaké stavy jakých atributů **zavazadel** **nápravné opatření** mění? Z jakého stavu do jakého stavu?



Obr. 8.7 SD1.2 – Ztracená a nalezená zavazadla – přiblížení – OPD získané přiblížením procesu zpracování ztracených a nalezených zavazadel v SD na [obr. 6.5](#)

Part II

Foundations of OPM

and SysML

Část I byla neformálním úvodem do OPM a SysML, ve kterém jsme použili podrobnou případovou studii systému automatické reakce na nehody. Diskutovali jsme aspekty OPM a různé druhy diagramů SysML. Část II poskytuje formálnější a teoreticky podloženější výklad OPM a SysML. Systematicky pokrývá ontologii, konstrukce koncepčního modelování a aplikace. [Kapitola 9](#) představuje a definuje, co je koncepční modelování, jaký je jeho účel a kontext. [Kapitola 10](#) představuje dva základní stavební kameny OPM – objekty a procesy. Podobně jako je strukturována část I, [kapitola 11](#) se zabývá textovou modalitou OPM – OPL. V [kapitole 12](#) se věnujeme systematickému studiu SysML s jeho čtyřmi pilíři a devíti druhy diagramů. Dynamický, časově závislý aspekt systémů je tématem [kapitoly 13](#). Na to přirozeně navazuje studium strukturálního, časově nezávislého aspektu systému v [kapitole 14](#). Po [kapitole 15](#), která se zabývá omezeními účasti a odbočkami, představujeme v [kapitole 16](#) čtyři základní strukturální vztahy. Tím končí část II. V části III s názvem *Struktura a chování – ponoření se do hloubky* se věnujeme podrobnému rozboru každého ze čtyř základních strukturálních vztahů a pokračujeme celkovými aspekty systému, včetně řízení a kontroly složitosti.

Kapitola 9

Koncepční modelování: účel a kontext

Koncepční model je formální model, ve kterém má každá entita modelovaná v reálném světě transparentní a jednoznačnou korespondenci s objektem v modelu.

Simmons (1994)

Než se pustíme do formálních prezentací OPM a SysML jako koncepčních jazyků pro modelování systémů a OPM jako metodiky systémového inženýrství, budeme diskutovat teoretické aspekty, které tvoří základ rámce systémů, systémové architektury a systémového inženýrství, v němž je koncepční modelování cennou intelektuální činností.

9.1 Systémy, modelování a systémové inženýrství

Systémy jsou všude kolem nás. Přírodní systémy existují již po celé věky a biologické organismy se vyvinuly do extrémně komplexních systémů. Umělé, člověkem vytvořené systémy, produkty a služby se také stávají stále komplexnějšími. Systémy infrastrukturní povahy, jako je řízení letového provozu, internet a elektronická ekonomika, jsou o řád složitější než produkty, které běžně používají jednotlivci. Kombinace miniaturizace a výpočetního výkonu je tak všudypřítomná, že i běžné výrobky pro domácnost vykazují inteligentní funkce zabudované do stále menšího hardwaru podobného komoditám, což vede k vzniku internetu věcí – konglomerátu slabě propojených zařízení všeho druhu, které vytvářejí volně propojený mega systém systémů.

9.1.1 Věda a technika: společné rysy a rozdíly

Hlavní rozdíl mezi *vědou* a *inženýrstvím* spočívá v tom, že vědci se snaží zkoumat a porozumět pozorovatelným fyzikálním, informatickým (kybernetickým) a lidským jevům, zatímco inženýři, kteří vycházejí z vědeckých objevů, navrhují, konstruují, vyvíjejí, udržují a zdokonalují umělé systémy ve prospěch lidí. Někdy jsou inženýři nuceni provádět reverzní inženýrství, tj. zkoumání existujícího systému, jehož funkce, struktura, chování nebo principy fungování nejsou dostupné a jsou neznámé.

Vzhledem k tomuto průzkumnému charakteru reverzního inženýrství lze *vědu* považovat za *reverzní inženýrství přírody*. Když je systém navrhován (inženýry) nebo zkoumán (vědci), rychle se hromadí podrobnosti o něm. Shromážděné skutečnosti, ať už skutečné, předpokládané, uvažované nebo domnělé, se stávají tak objemnými, že je obtížné je zvládnout bez systematického způsobu, jak dát smysl tomu, co se odhaluje. Správa těchto skutečností je nezbytná, aby dávaly smysl jako celek. Vzhledem k rychlému vývoji složitosti systémů se stále více zvyšuje potřeba intuitivního, ale formálního způsobu dokumentace návrhů nových systémů nebo shromážděných informací o stávajících systémech.

zřejmé. To zase vyžaduje solidní infrastrukturu pro zaznamenávání, ukládání, organizování, vyhledávání a prezentování nashromážděných znalostí a kreativních nápadů, které na těchto znalostech staví.

9.1.2 Koncepční modelování a modelové systémové inženýrství

Proces reprezentace znalostí souvisejících se systémy v oblasti vědy i inženýrství se obvykle označuje jako *koncepční modelování* a výsledkem této činnosti je *koncepční model*. Následné kognitivní činnosti vyššího řádu, včetně porozumění, analýzy, návrhu, prezentace a komunikace výsledků analýzy a návrhových myšlenek, mohou být založeny na vyvíjejícím se koncepčním modelu.

Vize divize inženýrských systémů Massachusetts Institute of Technology (MIT ESD, 2015) je, že „základní principy a vlastnosti inženýrských systémů jsou dobře pochopeny, takže tyto systémy mohou být efektivně **modelovány, navrhovány a spravovány**“.

Koncepční modelování, které často předchází matematickému a fyzikálnímu modelování nebo se provádí souběžně s nimi, je základní činností nezbytnou pro pochopení, návrh a správu inženýrských systémů. Modelování je proces, na kterém je založeno **modelové systémové inženýrství** (MBSE), které je tématem této knihy. MBSE není jen o modelování, jak se někteří lidé mylně domnívají; je to systémové inženýrství (SE) založené na formálním modelování různých druhů – koncepčním, matematickým a fyzikálním. Koncepční model je komplexní základní plán – referenční artefakt, který představuje zdroj autority různých aspektů systému – požadavků, výkonu, funkčnosti, struktury, dynamiky a mnoha dalších fyzikálních a informatických (kybernetických) aspektů. MBSE proto vyžaduje přísnou metodiku koncepčního modelování, která zahrnuje univerzální ontologii, jazyk, soubor zásad a pokynů a podpůrné softwarové prostředí pro modelování.

Porozumění fyzickým, biologickým, umělým a sociálním systémům vyžaduje dobře podloženou, formální, ale intuitivní metodiku a jazyk, který je schopen modelovat složitosti inherentní těmto systémům koherentním a přímočarým způsobem. Stejný modelovací paradigmat, jádro metodiky, by měl sloužit jak pro navrhování nových systémů (inženýrství), tak pro studium (věda) a vylepšování těch stávajících. Měl by se vztahovat na umělé i přírodní systémy a reprezentovat je stejně věrně. Společný, jednotný koncepční modelovací rámec pro umělé i přírodní systémy je velmi důležitý, protože komplexní inženýrské systémy a fyzikální jevy se často vzájemně ovlivňují. Například pro modelování systému, jako je letadlo, satelit, systém protiraketové obrany nebo lékařské zařízení, je nutné porozumět relevantním mechanickým, elektrickým, chemickým, biologickým a fyzikálním principům, které řídí jak systém, tak prostředí, ve kterém funguje a se kterým interaguje.

9.2 Základní ontologie systémového inženýrství OPM ()

Ontologie je soubor pojmů a jejich vztahů v určité oblasti diskurzu.

Velikost ontologie je dána počtem pojmů a vztahů v ontologii. Systémová věda a inženýrství potřebují dobře definovanou základní, univerzální, obecnou, nezbytnou a dostačující ontologii, která by podpořila pojmy a termíny, které používají, aby byly přesné a jednoznačné. Následující princip *minimální ontologie* poskytuje dobrý výchozí bod pro naši diskusi.

Princip minimální ontologie

Pokud lze systém specifikovat se stejnou přesností a podrobností dvěma jazyky různé velikosti ontologie, pak je jazyk s menší velikostí preferován před jazykem s větší velikostí, za předpokladu, že srozumitelnost specifikace prvního je alespoň srovnatelná se srozumitelností druhého.

Tento princip nejenže dává dokonalý smysl, ale je také v souladu s dlouho přijímaným Ockhamovým břitvou (Ockham, 1495) – principem připisovaným logikovi a františkánskému mnichovi Williamovi z Ockhamu z 14. století, který říká, že „*entity by neměly být zbytečně znásobovány*“ (latinsky: „*Pluralitas non est ponenda sine necessitate*“). Tento princip, často nazývaný **principem úspornosti**, má tři užitečné varianty, které z Ockhamovy břitvy vyplývají.

- „*Pokud máte dvě konkurující si teorie, které vedou ke stejným předpovědím, je lepší ta jednodušší.*“
- „*Neměli bychom nad rámec nezbytného zvyšovat počet entit potřebných k vysvětlení čehokoli*“ (Hilghen 1997).
- „*Vždy by se mělo volit nejjednodušší vysvětlení jevu, které vyžaduje co nejméně logických skoků.*“

Důvodem pro doplnění principu minimální ontologie o podmínku „... za předpokladu, že srozumitelnost specifikace prvního je alespoň srovnatelná se srozumitelností druhého“ je to, že v extrémním případě lze argumentovat, že binární kód 0 a 1 je nejkratší, a tedy nejlepší. To platí pro počítače, pro které je skutečné lidské porozumění (stále?) bezvýznamné. Pro lidi je z sémantického hlediska binární specifikace jakéhokoli netriviálního systému (např. počítačový program v strojovém kódu, aby byl případ jasnější) zcela nerozlučitelná bez nepřiměřeného úsilí. Proto požadujeme, aby obě ontologie umožňovaly specifikaci (nebo modelování) systémů s přibližně stejnou úrovní srozumitelnosti, nebo ještě lépe, aby specifikace, která používá menší ontologii, byla srozumitelnější. Pokud je ontologie definována pečlivě a je založena na hlubokých filozofických základech, nemusí nutně existovat kompromis mezi velikostí ontologie a délkou specifikace nebo srozumitelností systému modelovaného na základě této ontologie.

Ockhamova břitva inspirovala také princip minimální délky popisu (MDL) (Rissanen 1978), metodu induktivní inference, která poskytuje obecné řešení problému výběru modelu, tj. jak se rozhodnout mezi konkurenčními vysvětleními dat při omezených pozorováních. MDL je založena na poznatku, že jakákoli pravidelnost v daném souboru dat může být použita ke kompresi dat tím, že je popíše méně symboly, než je počet symbolů potřebných k popisu původních dat. V podobném duchu formulujeme následující *princip minimálního konceptuálního modelovacího jazyka OPM*.

Princip minimálního konceptuálního modelovacího jazyka OPM

Symbolický systém – jazyk – který dokáže konceptuálně modelovat daný systém pomocí ontologie s menším počtem typů diagramů a menším počtem symbolů a vztahů mezi nimi, je lepší než větší jazyk s větším počtem typů diagramů a větším počtem symbolů a vztahů mezi nimi.

Použití menší ontologie klade menší kognitivní zátěž na lidského modeláře, díky čemuž je konceptuální model srozumitelnější a lépe sdělitelný všem zúčastněným stranám, aniž by byla ohrožena věrnost

a úrovně detailů modelu. Výše uvedený princip můžeme přeformulovat téměř opačně: Jazyk s menším počtem symbolů a méně druhy diagramů, který je založen na univerzální ontologii, může popsat jakýkoli systém s lepší srozumitelností než jazyk s více symboly a více druhy diagramů.

Snížení kognitivní zátěže člověka je velmi žádoucí, protože modelář musí čelit inherentní, neredukovatelné složitosti umělých systémů, které mají být vytvořeny (systémové inženýrství), nebo přírodních systémů, které mají být zkoumány (věda), takže snížení zbytečné složitosti (často nazývané komplikovanost) poskytnutím jednoduššího jazyka má obrovskou hodnotu.

9.2.1 Objekty existují, procesy se dějí? Několik podnětných otázek a odpovědí

Pokud přijmeme princip minimální ontologie, pak musíme najít *minimální univerzální* ontologii – ontologii, která je nezbytná a dostačující k modelování vesmíru a systémů v něm. Začneme tím, že nejprve tvrdíme, že cokoli ve vesmíru buď existuje, nebo se děje. Pokračujeme sérií otázek a odpovědí, které nás mají dovést k poznatkům o možné minimální univerzální ontologii.

Otázka 1: Za předpokladu, že vše ve vesmíru je věc, co mohou věci ve vesmíru „dělat“?

Odpověď 1: *Věci mohou existovat nebo se dít. Jakákoli věc může buď existovat, nebo se dít. O ničem nelze říci, že neexistuje ani se neděje, ať už ve skutečnosti nebo potenciálně, fyzicky nebo informačně.*

Otázka 2: Jaký by byl obecný název pro všechny věci ve vesmíru, které *existují* nebo by mohly existovat fyzicky nebo koncepčně?

Odpověď 2: *Objekty existují nebo by mohly existovat.*

Otázka 3: Jaké věci ve vesmíru se *dějí* nebo mohou existovat fyzicky nebo koncepčně?

Odpověď 3: *Procesy se dějí nebo mohou dít.*

Procesy nemohou probíhat ve vakuu, aniž by něco „dělaly“, což vede k další otázce.

Otázka 4: Na čem se procesy odehrávají?

Odpověď 4: *Procesy se odehrávají nebo mohou odehrávat na objektech.*

Otázka 5: Jak procesy ovlivňují objekty?

Odpověď 5: *Procesy transformují objekty.*

Otázka 6: Co znamená, že proces *transformuje* objekt?

Odpověď 6: *Transformace objektu procesem znamená jednu z následujících tří možností:*

1. *vytvoření (generování) objektu,*
2. *zničení (spotřeba) objektu nebo*
3. *ovlivnění (změna) objektu.*

Otázka 7: Co to znamená, když proces *ovlivňuje* objekt?

Odpověď 7: *Proces ovlivňuje objekt tím, že mění jeho stav.*

Objekty proto musí být stavové, tj. musí mít stavy.

Otázka 8: Jakým způsobem jsou věci sémanticky spojeny? Je to jediný způsob?

Odpověď 8: *Věci jsou sémanticky spojeny prostřednictvím vztahů. Vztahy jsou jediným způsobem, jak můžeme uvažovat o tom, jak se věci vzájemně vztahují, odkazují nebo jsou spojeny.*

Otázka 9: Existuje rozdíl mezi tím, jak jsou objekty a procesy vzájemně propojeny?

A9: *Objekty jsou spojeny s objekty (a procesy s procesy) prostřednictvím strukturálních (statických) vztahů, zatímco objekty a procesy jsou spojeny prostřednictvím časově závislých procedurálních (dynamických) vztahů.*

Otázka 10: Jaké jsou dva univerzální aspekty, tj. dva aspekty, z nichž lze věci ve vesmíru pozorovat, zvažovat a popisovat?

Odpověď 10: Dva univerzální aspekty jsou (1) *struktura – způsob, jakým se **objekty** vzájemně vztahují a procesy se vzájemně ovlivňují* – a (2) *chování – způsob, jakým **procesy** transformují objekty v čase*.

9.2.2 Věta o objektech a procesech

Odpovědi na výše uvedené otázky lze považovat za *univerzální axiomy*, protože ačkoli dávají smysl, je obtížné je dokázat. Pokud tyto axiomy přijmeme, dospějeme k závěru, že věci – objekty se stavem a procesy – a vztahy mezi nimi jsou jedinými třemi prvky potřebnými k popisu vesmíru! Univerzální axiomy můžeme použít k dokázání následujícího *objektově-procesního teorému*.

Věta o objektech a procesech

Objekty se stavem, procesy a vztahy mezi nimi tvoří minimální univerzální ontologii.

Důkaz:

Důkaz je založen na (1) nezbytnosti a (2) dostatečnosti objektů se stavem, procesů a vztahů mezi nimi jako jediných tří druhů prvků potřebných k vytvoření minimální univerzální ontologie. Důkaz je proto rozdělen do dvou částí: nezbytnost a dostatečnost.

Část 1 – nutnost: Objekty se stavem a procesy jsou nezbytné k určení dvou univerzálních aspektů, struktury a chování: Určení *strukturálního, statického* aspektu systému vyžaduje **objekty se stavem a vztahy** mezi nimi. Určení *procedurálního, dynamického* aspektu systému vyžaduje **procesy a vztahy** mezi nimi a objekty, které transformují.

Část 2 – dostatečnost: Věci mohou buď existovat (nazýváme je objekty se stavem) nebo se dít (nazýváme je procesy) a nic jiného. Věci mohou být navzájem spojeny pouze prostřednictvím vztahů. Proto jsou **věci** (objekty a procesy) a **vztahy** mezi nimi jedinými prvky potřebnými ke specifikaci faktů nebo myšlenek. **Q.E.D.**

9.2.3 Důsledek objekt-proces

Věta o objektech a procesech vede k následujícímu *důsledku*.

Důsledek objekt-proces

Pomocí stavových objektů, procesů a vztahů mezi nimi lze koncepčně modelovat jakýkoli systém v jakékoli doméně.

Vzhledem k tomu, že podle věty o objektech a procesech stavové objekty, procesy a vztahy mezi nimi tvoří minimální univerzální ontologii a vesmír je sjednocením všech domén, které zahrnuje, má toto tvrzení smysl. Jednou z možných výjimek je kvantová doména subatomárních částic, kde se naše makroskopické rozlišení mezi objekty a procesy stává nejasným. Například elektrony a fotony jsou popisovány jak jako částice (objekty), tak jako vlny (procesy). Jakmile vstoupíme na atomovou a molekulární úroveň, např. molekulární biologii (Somekh et al. 2014), stává se objektově-procesní korolář platným a OPM se stává životaschopným a atraktivním modelovacím paradigmatem.

Tato první verze objektově-procesního koroláře neříká nic o úrovni složitosti systémů, které lze modelovat pomocí stavových objektů, procesů a vztahů mezi nimi.

9.2.4 Objektově-procesní tvrzení: základ pro OPM

Spojením objektově-procesního koroláře s tvrzením o složitosti modelu dostaneme následující *Objektově-procesní tvrzení*.

Objektově-procesní tvrzení

Pomocí stavových objektů, procesů a vztahů mezi nimi, spolu s mechanismy zpřesňování a rozvíjení, lze konceptně modelovat systémy v jakékoli doméně a na jakékoli úrovni složitosti.

Kombinací objektově-procesního teorému, podle kterého stavové objekty, procesy a vztahy mezi nimi tvoří minimální univerzální ontologii, s principem minimální ontologie, musí optimální konceptuální modelovací jazyk mít pouze dva typy konceptů – *stavové objekty* a *procesy*, *souhrnně nazývané věci* – spolu s *vztahy* mezi nimi. Souhrnně jsou *věci* a *vztahy* jedinými dvěma *prvky* OPM.

Věci ve stejném systému musí být přímo nebo nepřímo propojeny. Graficky jsou tyto vztahy vyjádřeny pomocí *odkazů*. Věci a odkazy se souhrnně nazývají prvky, a proto je *prvek* nejvyšší úroveň konceptu OPM.

Prvek OPM je věc nebo odkaz.

9.2.5 Proč ne jen jeden druh věci? Graf s uzly a spojeními?

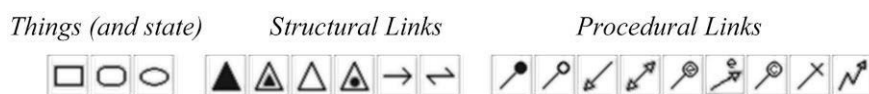
Lze argumentovat, že ještě minimalističtější reprezentace než tři druhy prvků – objekty, procesy a vztahy mezi nimi – by mohla být pouze dvě: věci a vztahy mezi nimi. Ve skutečnosti řada rámců pro reprezentaci znalostí přišla s touto myšlenkou reprezentace znalostí pomocí grafu s uzly pouze jednoho druhu a spojeními, která je propojují. Některé z těchto rámců, které se liší svou úrovní formálnosti, jsou popsány v Dori (2004). Patří mezi ně konceptuální mapy (Arnheim 1969), diagramy entit a vztahů (Chen 1976), sémantické sítě (Lehman 1999), konceptuální grafy (Chein a Mugnier 1992) a systemigramy (Blair et al. 2007). Při pohledu na příklady grafů vyjádřených v těchto přístupech rychle zjistíme, že jelikož existuje pouze jeden druh uzlu, neexistuje rozdíl mezi objektem a procesem, takže schopnost rozlišovat mezi strukturou a chováním – dvěma odlišnými aspekty, které musí být v každém modelu zastoupeny – je značně omezena nebo dokonce neexistuje. Za malou cenu zvýšení počtu prvků v ontologii ze dvou na tři získáme obrovskou schopnost současně modelovat strukturu i chování systému.

Objekty jsou skutečně věci, které *existují*. Vztahy mezi nimi tvoří *strukturu* systému. To je statický, strukturální aspekt systému. Abychom pochopili dynamický, procedurální aspekt systému, abychom věděli, *co se děje* s objekty v systému a jak funguje, aby poskytoval hodnotu, je zapotřebí druhý, doplňkový typ věci – *proces*. O existenci objektu víme, pokud jej můžeme pojmenovat a odkazovat se na jeho bezpodmínečnou, relativně stabilní existenci, ale bez procesů nemůžeme říci, jak je tento objekt vytvořen nebo zničen, ani jak se mění jeho stavy během jeho životnosti.

Objekt bez stavu je objekt, který nemá žádné stavy. *Objekt se stavem* je objekt, který má jeden nebo více stavů. Tyto stavy jsou *stabilní* v tom smyslu, že k přepnutí objektu z jednoho stavu do druhého je zapotřebí proces, a pokud na objekt nepůsobí žádný proces, zůstává objekt ve stejném stavu.

Obrázek 9.1 představuje hlavní symboly OPM. Symboly pro objekt, stav a proces jsou zobrazeny jako první (nejlevější) skupina symbolů. Zbytek symbolů jsou odkazy: *strukturální odkazy* jsou zobrazeny ve střední skupině a *procedurální odkazy* v nejvpravo skupině. Jejich názvy a sémantika byly zmíněny v části I a budou dále rozvedeny v dalším textu.

Objekty a procesy, souhrnně označované jako *věci* OPM, jsou dva typy univerzálních stavebních bloků OPM. OPM považuje objekty a procesy za rovnocenné, takže procesy nejsou nutně podřízené objektům ani jim nevlastní. Symetricky, objekty nejsou nutně podřízené procesům, ani procesy nejsou nutně vlastněné objekty.



Obr. 9.1 Tři skupiny symbolů prvků OPM

Stav je znázorněn na [obr. 9.1](#) mezi objektem a procesem. Jak bude podrobněji popsáno později, stav je situace, ve které se objekt může nacházet v určitém okamžiku během své životnosti.

9.2.6 Princip OPM „Důležitost věci“

V OO objekty „vlastní“ procesy, které se v žargonu OO nazývají operace, služby nebo metody. OPM zaujímá odlišný postoj: Hlavní procesy na úrovni systému mohou být stejně důležité nebo dokonce důležitější než objekty v systémovém modelu. Zejména jsme již poznamenali, že proces nejvyšší úrovně systému (nebo subsystému) je jeho *funkcí*, procesem nejvyšší úrovně poskytujícím hodnotu a sloužícím účelu, pro jehož výkon je systém vytvořen a používán. Proces proto musí být modelovatelný nezávisle na jakékoli konkrétní sadě objektů zapojených do jeho výskytu. OPM proto považuje *objekty* i *procesy* za *rovnocenné*. Stojí na stejné úrovni, ani jeden z nich nemá nad druhým převahu. Jejich význam spíše souvisí s hierarchií modelu, jak je vyjádřeno v následujícím *principu OPM týkajícího se významu věcí*.

Důležitost věci Princip OPM

Důležitost věci T v modelu OPM přímo souvisí s nejvyšší hodnotou OPD v hierarchii OPD, kde se T objevuje.

Například objekt **ACR System** a proces **Automatic Crash Responding** na [obr. 1.2](#) mají stejný relativní význam, protože se poprvé objevují v SD, systémovém diagramu, který je nejvyšší úrovní OPD. Objekt **ACR System** je skutečně nezbytný pro proces **Automatic Crash Responding**, takže nelze tvrdit, že objekt **ACR System** má přednost před procesem **Automatic Crash Responding** nebo naopak.

Schopnost rozlišit objekty a procesy a správně je použít v modelu je klíčem k modelování v OPM. Abychom definovali tyto základní pojmy a sdělili jejich sémantiku, budeme dále diskutovat pojmy *existence* a *transformace*.

9.3 Definice objektu, stavu, transformace a procesu

Jelikož objekty OPM mohou být fyzické nebo informatické (kybernetické), definujeme objekt jako něco, co zachycuje tyto dvě stránky, aniž by se zavazovalo k jedné z nich, a zároveň zahrnuje prvek „existence v čase“.

***Objekt** je věc, která existuje nebo může existovat fyzicky nebo informaticky.*

Existence objektu může být fyzická nebo informatická. Může to být něco tak jednoduchého, jako je kus ledu, slovo v knize nebo záznam v souboru, nebo něco tak složitého, jako je organizace, internet, lidský mozek nebo galaxie.

***Stav** je možná situace nebo pozice, ve které se objekt může nacházet po určitou dobu.*

Tato definice implikuje, že stav má význam pouze v rámci a v kontextu objektu. Stav nemá žádný význam mimo kontext objektu, ke kterému patří. Například stavy objektu **Organizace** mohou být **soukromé** nebo **veřejné** a stavy objektu **Záznam** mohou být **uzamčené** nebo **odemčené**. Stavy **soukromý** a **uzamčený** nemají žádný význam mimo kontext příslušných objektů, ke kterým patří.

***Transformace** je (1) vytvoření (generování, konstrukce), (2) spotřeba (eliminace, zničení) nebo (3) účinek – změna stavu objektu.*

Transformace trvá určitou dobu.

***Proces** je věc, která transformuje objekt.*

Podle této definice musí být proces spojen s alespoň jedním objektem: tím, který tento proces transformuje. Například **zmrazování** je proces, který mění stav **vody** z **kapaliny** na **led**. To je základ *principu transformace objektů procesem OPM*.

Princip transformace objektu procesem OPM

V kompletním modelu OPM musí být každý proces spojen s alespoň jedním objektem, který proces transformuje, nebo s jedním stavem objektu, který proces transformuje.

Netriviální synchronní proces (tj. proces, jehož podprocesy mají definované pořadí provedení) zahrnuje hierarchickou síť podprocesů. Na každé úrovni procesní hierarchie existuje časově indukované částečné pořadí procesů, tj. některé procesy musí skončit předtím, než jiné začnou, zatímco jiné mohou probíhat paralelně s jinými procesy nebo jako jejich alternativy.

9.4 Systém a související pojmy z oblasti

Odložíme formální definici systému na chvíli stranou a přidáme několik otázek k našim otázkám z oddílu 9.2.1:

Otázka 11: Jaké jsou dva hlavní aspekty, které mají všechny systémy společné?

A11: *Jako součást vesmíru lze všechny systémy nahlížet ze dvou hlavních hledisek: **struktury a chování**.*

***Struktura** je statický aspekt; souvisí s otázkou, z čeho je systém složen? Ze strukturálního hlediska je **systém** konečná **množina komponent** a jejich časově neměnných propojení.*

***Chování** je dynamický aspekt; souvisí s otázkou, **jak** se systém mění v čase?*

Otázka 12: Jaký je další významný aspekt, který se týká především předem promyšlených systémů vytvořených člověkem?

Odpověď 12: *Funkce – utilitární, subjektivní aspekt: **Proč** je systém vytvořen? **Pro koho?** **Kdo** jsou beneficianti, kteří mají prospěch z jeho provozu? **Jakou hodnotu** získávají tito beneficianti z provozu systému?*

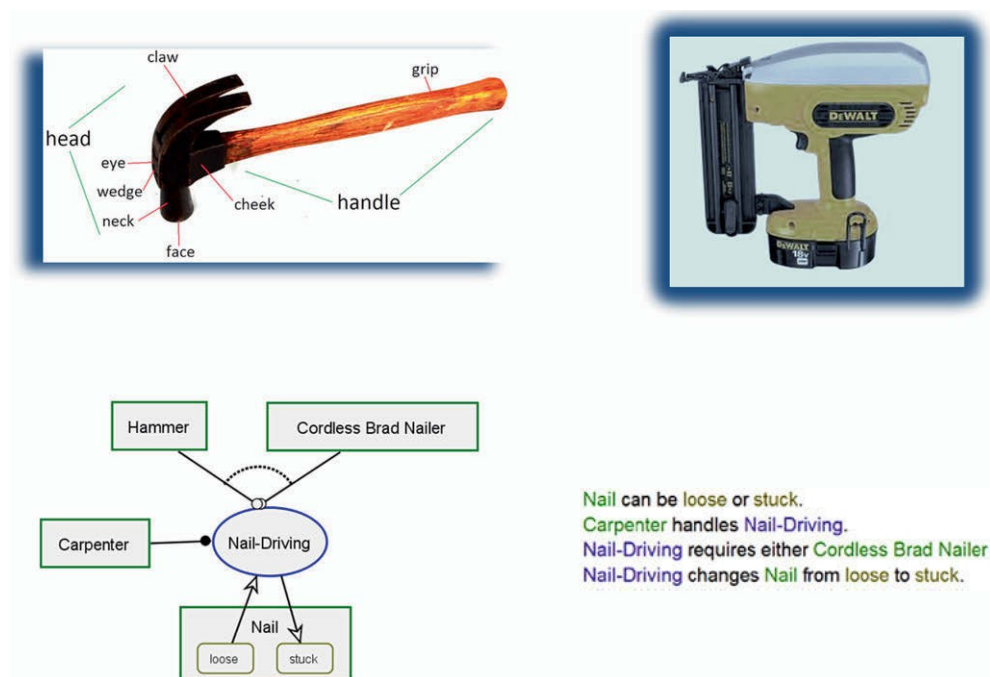
Do jisté míry lze tvrdit, že biologické organismy jsou systémy, které poskytují funkce, které prospívají jim samotným nebo jiným systémům, ale takové (často vzájemné, např. symbiotické) výhody jsou výsledkem evolučních procesů, nikoli předem promyšleného záměru, který je charakteristický pro člověka jako organismus „vytvářející nástroje“. Jak uvedli odborníci ze Smithsonian Institute (2015):

Za posledních 2,6 milionu let ... kamenné nástroje poskytují důkazy o technologiích, zručnosti, určitých druzích mentálních schopností a inovacích, které byly v dosahu raných lidských výrobců nástrojů...

Funkce je klíčovým pojmem v systémech vytvořených člověkem; je to proces, který poskytuje (funkční) hodnotu příjemci. Příjemcem je osoba nebo skupina osob a hodnotou je jejich *přínos v poměru k nákladům* – rozdíl mezi vnímaným přínosem systému a náklady systému. Na základě této definice funkce definujeme systém následovně.

Systém je objekt poskytující funkce.

Tato stručná definice je poměrně neortodoxní. Stojí za to porovnat tuto definici s definicí systému v normě ISO/IEC 15288. Podle normy ISO/IEC 15288 je *systém kombinací interagujících prvků uspořádaných tak, aby dosáhly jednoho nebo více stanovených účelů*. Standardní definice je kompatibilní s naší, protože obsahuje prvek účelu, který je podobný funkci – poskytování hodnoty nějakému příjemci. Naše definice systému je obecnější v tom, že nevyžaduje, aby byl systém složen z interagujících prvků. Ačkoli je tento popis obecně pravdivý, nevyjadřuje podstatu systému. V komplexních systémech, a ještě více v systémech systémů, jako je mezinárodní systém řízení letového provozu, jehož nově vznikající funkcí je regulace letecké dopravy po celém světě, existuje řada vzájemně interagujících fyzických a informačních částí, včetně letadel, letišť, komunikačních sítí a leteckých dopravců.



Obr. 9.2 Dva koncepty systému pro zatloukání hřebíků: Vlevo nahoře: kladivo; Vpravo nahoře: DEWALT 18-Volt 18-Gauge 2 in. Cordless Brad Nailer; Vlevo dole: OPD systému pro zatloukání hřebíků; Vpravo dole: Odpovídající OPL¹

V jednoduchém systému, jako je systém zatloukání hřebíků – kladivo zobrazené vlevo nahoře na obr. 9.2, které je kombinací dvou vzájemně spolupracujících prvků – hlavy a rukojeti – není počet vzájemně spolupracujících částí rozhodujícím prvkem. Důležité je, že kladivo je systém, který poskytuje funkci zatloukání hřebíků. Při bližším pohledu na toto kladivo lze rozlišit funkční prvky nižší úrovně, jako je dráp pro vytahování hřebíků, ale nejedná se o skutečně samostatné části, což dále zdůrazňuje funkční aspekty tohoto systému. Stejnou funkci zatloukání hřebíků může plnit i mnohem sofistikovanější systém, jako je ten, který je znázorněn v pravém horním rohu obr. 9.2. Ačkoli se jedná o mnohem složitější systém, poskytuje v zásadě stejnou funkci zatloukání hřebíků (a skutečně se nazývá „hřebíkovačka“).

Model OPM (OPD a odpovídající OPL) v dolní části obr. 9.2 zdůrazňuje společnou funkci těchto dvou systémů. Rozdíl mezi těmito dvěma systémy spočívá v několika výkonnostních metrikách, které lze odvodit z následujícího popisu uvedeného na webových stránkách tohoto produktu: „Akumulátorová hřebíkovačka DEWALT DC608K—18 Gauge 2 in. Cordless Brad Nailer zajišťuje konzistentní zatloukání hřebíků“

¹ Od tohoto bodu nejsou OPD stínovány, protože jsou doprovázeny odpovídajícími odstavci OPL. Barvy různých frází OPL v OPL zde jsou stejné jako v OPCAT. V následujících OPL jsou vyhrazené fráze OPL uvedeny tučným písmem Arial a nevyhrazené fráze **tučným písmem Arial**.

pronikání do měkkých i tvrdých spojů. Sekvenční provozní režim umožňuje přesné umístění a provozní režim s nárazem poskytuje uživateli rychlost výroby. Přímý zásobník pojme hřebíky o průměru 18 gauge v délkách od 5/8 palce do 2 palců. Jeho 12polohový ovladač umožňuje uživateli přecházet mezi aplikacemi, aniž by musel znovu nastavovat přesnou hloubku. Jak vidíme, funkce tohoto systému je popsána jako „penetrace hřebíků“, stejně jako u kladiva, i když možná s lepší rychlostí, silou a přesností. Podle naší definice systému jsou tedy jak kladivo, tak akumulátorová hřebíkovačka systémy pro zatlučení hřebíků.

Subsystém, známý také jako **komponenta** nebo **modul**, je částí systému, která sama o sobě neposkytuje funkci, kterou poskytuje systém.

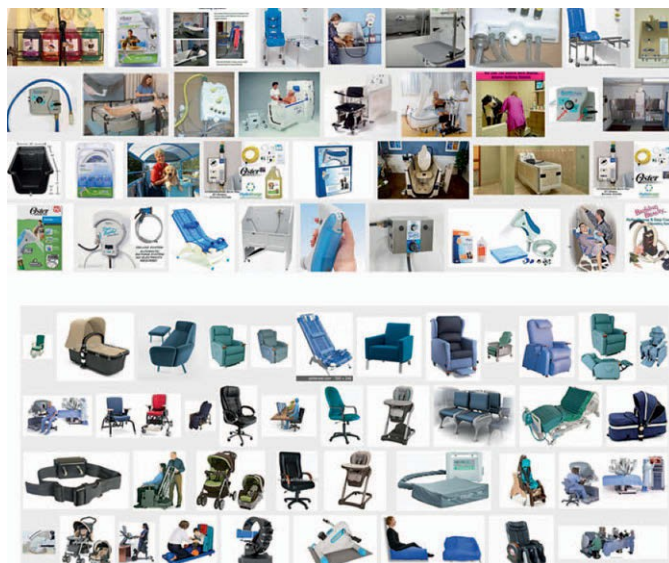
Systém se skládá z subsystémů, modulů nebo komponent – všech objektů –, které pouze ve spojení poskytují (vznikající) funkci, čímž tvoří systém. Toto je vhodné místo pro definici systému (SoS), který lze v níže vysvětleném smyslu považovat za „opak“ subsystému.

Systém systémů (SoS) je systém, jehož soubor subsystémů obsahuje alespoň dva systémy.

Tato definice SoS znamená, že SoS se skládá z alespoň dvou komponent, z nichž každá je samostatným systémem, a proto má podle definice svou vlastní funkci. Jinými slovy, pokud rozložíme SoS, získáme alespoň dvě funkční entity. Jelikož SoS je také systém, má kromě funkcí svých složek také vlastní emergentní funkci. Například globální systém řízení letového provozu je SoS, jehož funkcí je řízení letového provozu. Skládá se z mnoha systémů, jako jsou letiště, národní letecké úřady, národní a mezinárodní vzdušný prostor, Mezinárodní asociace leteckých dopravců (IATA, obchodní sdružení světových leteckých společností, které zastupuje přibližně 250 leteckých společností, tj. 84 % celkového leteckého provozu; IATA 2015), mezinárodní protokol letecké komunikace, nouzové předpisy, letecké dopravce, letadla, piloty, posádky, cestující a mnoho dalšího. Mnohé z těchto systémů, jako jsou letiště a letecké dopravce, jsou samy o sobě SoS. Naopak, ačkoli se jedná o vysoce komplexní systém, letadlo například není SoS, protože funguje samostatně a žádná z jeho součástí, jako jsou křídla, palivová nádrž nebo trup, nemůže plnit žádnou podstatnou funkci. V další části knihy se většina tvrzení o systému vztahuje také na subsystémy a SoS.

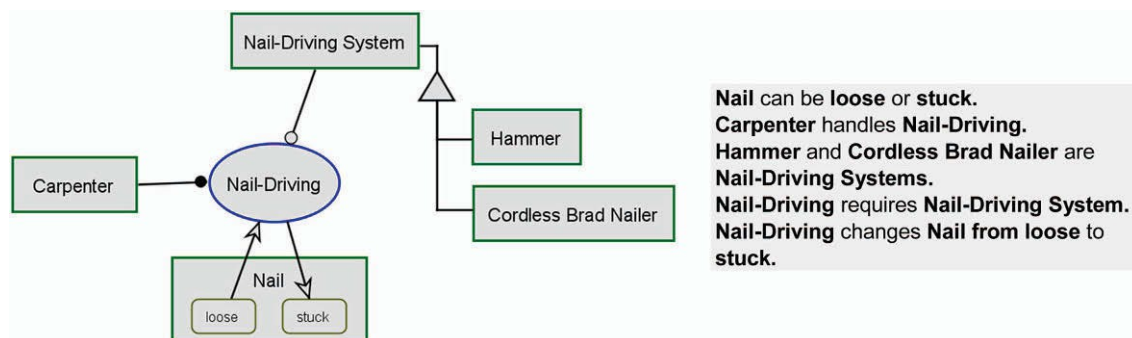
9.4.1 Výchozí pojmenování systému

V mluveném jazyce se jednoduché systémy, jako například kladivo, často nazývají nástroje, složitější systémy, jako například elektroměr, jsou přístroje, a ještě složitější systémy jsou „systémy“, ale všechny plní nějakou funkci – svůj stanovený účel – a rozdíl mezi nimi spočívá v jejich úrovni složitosti. Výchozí název systému je název funkce, kterou tento systém poskytuje, následovaný slovem „systém“. Například systém nazývaný tiskárna, jehož funkcí je tisk, lze nazvat „tiskovým systémem“. Nemocnice je systém zlepšující zdravotní stav, židle je systém pro sezení, dům je systém pro bydlení, vana je systém pro koupání a letadlo je systém pro létání. Pokud vyhledáte na webu obrázky „systému koupání“ a „systému sezení“, získáte neuvěřitelnou škálu výsledků, z nichž některé jsou uvedeny na [obr. 9.3](#), které lidé označují jako systémy koupání (nahore) a sezení (dole). Společnou funkcí prvních je jejich schopnost očistit nebo uklidnit lidi a druhých – poskytnout lidem určitou úroveň pohodlí při sezení.



Obr. 9.3 Výsledky vyhledávání obrázků „systému koupání“ (nahore) a „systému sezení“ (dole)

Obrázek 9.4 je model OPM funkce **zatloukání hřebíků** a systému **zatloukání hřebíků** – nástroje pro dosažení této funkce. Metoda používaná u každého druhu **systému zatloukání hřebíků** může být v diagramu zachycena jako specializace **zatloukání hřebíků**. **Kladivo** a **akumulátorová hřebíkovačka** jsou dvě takové specializace; jsou ztělesněním dvou různých *konceptů* pro dosažení funkce systému: **kladivo**, které je základní, a **akumulátorová hřebíkovačka**, která je složitější (a tudíž dražší).



Obr. 9.4 Model OPM funkce zatloukání hřebíků a systému zatloukání hřebíků se specializacemi kladivo a akumulátorová hřebíkovačka

9.4.2 Zúčastnění lidé: zainteresovaná strana, příjemce, zákazník, uživatel, dodavatel

Zúčastněné strany systému jsou subjekty, které se systémem souvisejí.

Zainteresaná strana je jednotlivec, organizace nebo skupina lidí, kteří mají zájem o systém nebo mohou být systémem ovlivněni.

Níže definujeme hlavní typy zainteresovaných stran. Jedna nebo více zainteresovaných stran systému je příjemce – účastník, který ze systému získává hodnotu a výhody.

Beneficient je zúčastněná strana, která ze systému získává hodnotu a výhody.

Zákazníci (skuteční nebo potenciální) jsou klíčovými zainteresovanými stranami.

Zákazník je zainteresovaná strana, která objednává systém a sponzoruje jeho vývoj, implementaci, nasazení a podporu, nebo kupuje produkt, který je součástí systému.

První typ zákazníka v výše uvedené definici je obvykle organizace, která potřebuje speciálně navržený systém a objednává jej od dodavatele (definovaného níže). Druhým typem je obvykle jednotlivec, který nakupuje spotřební zboží, které bylo navrženo a vyrobeno dodavatelem na základě předpokladu, že lidé budou ochotni za něj zaplatit, protože zákazník předvídá hodnotu, kterou tento systém (v tomto případě produkt, definovaný níže) přinese. Ať tak či onak, bez zákazníků je těžké si představit, proč by měl být systém vůbec vyvíjen.

Uživatel je zúčastněná strana, která systém provozuje nebo s ním přímo interaguje.

U relativně jednoduchých systémů, jako jsou výrobky pro domácnost, jsou zákazník a uživatel totožní. Například majitel automobilu, který jej řídí, je zákazníkem, uživatelem a příjemcem, zatímco ostatní cestující jsou pouze příjemci. Příjemci národního systému protiraketové obrany jsou občané dané země, ačkoli nejsou ani uživateli, ani zákazníky. Dodavatel je dalším klíčovým zainteresovaným subjektem.

Dodavatel je zainteresovaná strana, která dohlíží na vývoj, podporu a údržbu systému nebo produktu.

Mezi další zainteresované strany mohou patřit regulační orgány, soudní systém, veřejnost a subjekty, které mohou být systémem ovlivněny.

9.4.3 Zdroj systému: přírodní nebo umělý

Systémy pocházejí ze *zdroje*, který může být přírodní nebo umělý (vytvořený člověkem). Zdroj určuje mechanismus, prostřednictvím kterého se systém stal funkčním. V přírodních systémech je to výsledek činností řízených fyzikálními zákony. V biologických systémech, které jsou podskupinou přírodních systémů, hrají další důležitou roli principy evoluce.

Jelikož se tato kniha zaměřuje na inženýrství umělých systémů, bude se od tohoto bodu termín „systém“ vztahovat na *umělý* systém, pokud není uvedeno jinak. Mechanismus, prostřednictvím kterého je systém vytvořen a stává se funkčním, zahrnuje určitou úroveň intelektuálního a fyzického úsilí člověka, ať už je jakkoli primitivní a základní. Když se toto úsilí stane značným a překročí určitou hranici složitosti a zároveň vykazuje známky plánování a koordinace, nazýváme to inženýrstvím a

v poslední době také *systémovým inženýrstvím*. Netriviální systémy, které jsou předmětem zájmu systémového inženýrství, zahrnují značné množství *procesů*, které transformují velké množství propojených *objektů* (komponent systému) tak, aby bylo možné dosáhnout *funkce* systému.

Sociálně-technický systém, známý také jako inženýrský systém, je systém, který integruje technologii, lidi a služby a kombinuje perspektivy inženýrství, managementu a sociálních věd.

Produkty jsou navrhovány a vyráběny komerčními subjekty a prodávány jiným subjektům za účelem zisku.

Produkt je komerčně životaschopný systém.

Jelikož systém je objekt, produkt je komerčně životaschopný *objekt*. Analogicky, služba je *proces*, který je prodáván komerční entitou za účelem zisku.

Služba je komerčně životaschopný proces.

Zde máme na mysli obchodní službu. Ve světě softwaru je služba podobná metodě nebo operaci. Konkrétněji řečeno, v architektuře orientované na služby zahrnuje pojem služby jakékoli rozhraní poskytované komponentou v systému jiným komponentám a systémem jiným systémům.

9.4.4 Upřesnění definic funkce, struktury a chování

Po definování příjemce můžeme nyní upřesnit naši definici funkce.

Funkce umělého systému je jeho proces poskytování hodnoty na nejvyšší úrovni, jak jej vnímá příjemce.

Například funkcí kladiva je zatloukání hřebíků, funkcí tiskárny je tisk, funkcí židle je sezení, funkcí klíče a zámku je zamykání a odemykání, funkcí okna je větrání a osvětlení, funkcí ledničky je prodloužení trvanlivosti potravin, funkcí požárního alarmu je varování před požárem. Složitější systémy mají vyšší úroveň a abstraktnější funkce. Funkcí systému zvaného nemocnice je tedy *zlepšení zdravotního stavu* pacientů. Každý pacient je příjemcem tohoto systému, zákazníkem může být vláda nebo soukromý subjekt a zdravotnický personál tvoří skupinu uživatelů. Dalším příkladem je funkce protiraketového obranného systému, která spočívá v obraně země před raketovým útokem. Zákazníkem systému je vláda dané země, uživatelem je její armáda a příjemcem jsou lidé žijící v dané zemi.

Na nižších úrovních subsystému nebo komponenty může funkce subsystému také prospívat funkci systému na vyšší úrovni nebo jiným systémům či subsystémům. Například Dictionary.com uvádí následující podstatná jména pro kormidlo:

1. Námořnictví: svislá lopatka na zádi plavidla, kterou lze otáčet vodorovně, aby se změnil směr plavidla za jízdy.
2. Letectví: pohyblivá ovládací plocha připevněná k vertikálnímu stabilizátoru, umístěná v zadní části letadla a používaná spolu s křídélky k otáčení letadla.

Jak v námořnictví, tak v letectví je kormidlo subsystémem vozidla – plavidla a letadla, s funkcí změny kurzu nebo otáčení či navigace vozidla. Tato funkce kormidla je součástí funkce vozidla, kterou je přeprava osob a zboží, a která vyžaduje také pohon, který zajišťuje pohonný subsystém vozidla.

Struktura je statický, časově nezávislý aspekt systému:

Struktura systému je jeho forma – soustava jeho fyzických a informačních komponent spolu s dlouhodobými vztahy mezi nimi.

Chování je proměnlivý, časově závislý aspekt systému:

Chování systému je jeho dynamika – způsob, jakým se systém mění v čase transformací systémových (vnitřních) a/nebo environmentálních (vnějších) objektů.

9.4.5 Potřeba souběžného modelování struktury a chování

Během analýzy a návrhu se shromažďují a zaznamenávají fakta a nápady týkající se objektů v systému a jeho prostředí a procesů, které je transformují. U téměř každého procesu, který je objeven nebo zvažován, se první otázky týkají objektů zapojených do tohoto procesu. Podobně u každého objektu identifikovaného v systému je klíčovou otázkou, na jakých procesech se tento objekt podílí. Jakmile je do systému zaveden nový objekt, je třeba modelovat také proces, který jej transformuje nebo je jím umožněn.

Existuje tedy úzká soudržnost dvou klíčových aspektů systému: struktury (objektů a vztahů mezi nimi) a chování (procesů a jejich vztahů k objektům). Vzhledem k této komplementaritě struktury a chování mají systémoví analytici a architekti intuitivní a oprávněnou tendenci modelovat strukturu a chování systému *současně*.

Díky svému jednotnému, sjednocujícímu modelu objekt-proces OPM splňuje tento požadavek na souběžné modelování struktury a chování. Umožňuje modelovat tyto dva hlavní aspekty systému současně v rámci stejného modelu, aniž by bylo nutné neustále přepínat mezi různými typy diagramů.

U zkoumaného (na rozdíl od navrženého) systému se výzkumník snaží pochopit smysl shromážděných pozorování a porozumět jejich příčinným a následným vztahům. V jistém smyslu se pokouší o reverzní inženýrství studovaného systému, což je úkolem vědců. Jak v případě architektonicky navrženého, tak i zkoumaného systému jde struktura a chování systému ruku v ruce a je velmi obtížné pochopit jedno bez druhého, takže má smysl prezentovat obojí v jednom diagramu.

9.4.6 Architektura systému

S porozuměním toho, co je struktura a chování, můžeme definovat architekturu systému.

Architektura systému je kombinací struktury a chování systému, která mu umožňuje plnit jeho funkci.

Může být zajímavé porovnat naši definici architektury s definicí používanou v rámci architektury amerického ministerstva obrany (DoDAF 2007), která je založena na normě IEEE STD 610.12:

Architektura: *struktura komponent, jejich vztahy a zásady a pokyny, které řídí jejich návrh a vývoj v čase.*

TOGAF (2011) poskytuje podobnou definici v odpovědi na otázku „Co je architektura?“

Architektura je základní organizace něčeho, ztělesněná v jejích komponentách, jejich vzájemných vztazích a vztazích k prostředí a principech, které řídí její návrh a vývoj.

Společným prvkem obou definic i naší definice architektury je struktura systému. Definice DoDAF a TOGAF však postrádají integraci struktury s chováním, které zajišťuje funkci. Na druhou stranu definice DoDAF zahrnuje „zásady a pokyny upravující návrh a vývoj komponent systému v průběhu času“. Ty však zřejmě nejsou součástí architektury systému. Principy a pokyny spíše upravují *proces* navrhování architektury, jehož výsledkem je architektura systému. Zajímavé je, že DoDAF Architecture Framework Version 2.02, Change 1 (DoDAF 2015), verze z ledna 2015, neobsahuje žádnou jasnou definici architektury (stejně jako vydání z roku 2009)!

9.4.7 Systémové prostředí a příslušnost věcí

V posledních letech termín „*prostředí*“ stále častěji nabývá významu ekosystému planety Země, ve kterém všichni žijeme a který je neustále ohrožován kumulativními účinky velkých systémů vytvořených člověkem (například elektráren) a velkého počtu menších systémů vytvořených člověkem (například automobilů a letadel). Naše definice systémového prostředí je s tímto poznatkem skutečně kompatibilní, protože počítá s možností, že se prostředí může měnit v důsledku fungování systému.

Prostředí systému je soubor věcí, které se nacházejí mimo systém, ale s ním interagují.

Interakce systému s jeho prostředím způsobuje změnu systému a případně i jeho prostředí. Aby byla zajištěna udržitelnost, musí systémoví inženýři zabránit této nepříznivé změně nebo ji zvrátit, zejména pokud se týká možných nevratných škodlivých účinků současných a plánovaných systémů na globální oteplování a vyčerpávání přírodních zdrojů. Nejedná se pouze o morální nebo etickou povinnost – jde o zajištění udržitelného života všech organismů na Zemi, včetně lidí, i po dalších několika desetiletích...

Věc, která je součástí systému, je *systémová*, zatímco věc, která je součástí prostředí systému, je *environmentální*. Atributem věci OPM, jehož hodnoty jsou systémové a environmentální, je *příslušnost*. Rozlišování mezi systémovými a environmentálními věcmi je při modelování velmi důležité, protože ukazuje, nad čím má architekt kontrolu a co by mělo být považováno za dané. Například při navrhování čerpací stanice, je auto systémové nebo environmentální? Je zřejmé, že auta a jejich řidiči budou s čerpací stanicí interagovat, ale architekt čerpací stanice nemá kontrolu nad velikostí aut a umístěním otvorů pro palivové nádrže – ty jsou dané a musí se s nimi počítat. Auto je tedy pro čerpací stanici environmentální.

9.4.8 Funkce versus chování

Výše uvedené definice vedou k závěru, že funkcí systému je jeho proces nejvyšší úrovně. Navíc architektura systému, konkrétně kombinace jeho struktury a chování, umožňuje systému provádět proces nejvyšší úrovně, a tím plnit svou funkci a přinášet hodnotu jeho příjemci.

Hodnota funkce pro příjemce je často implicitní; je vyjádřena v pojmech procesu, které zdůrazňují *to, co se děje*, spíše než *účel*, pro který se proces nejvyšší úrovně děje. Toto implicitní vyjádření funkce může vysvětlit, proč je funkce systému často zaměňována s chováním nebo dynamikou systému. Je však zásadní jasně a jednoznačně rozlišovat mezi těmito dvěma pojmy, tedy mezi funkcí a chováním. Chování je to, jak se systém mění v časové dimenzi. Funkce je hodnota, kterou systém poskytuje svému příjemci prostřednictvím svého provozu. Chování je tedy objektivní – je to způsob, jakým se systém mění, bez ohledu na to, kdo tuto změnu popisuje, zatímco funkce je subjektivní – je to hodnota získaná z pohledu příjemce. Toto rozlišení mezi funkcí a chováním je nanejvýš důležité, protože v mnoha případech lze funkci systému dosáhnout různými architekturami, tj. různými kombinacemi procesů (chování systému) a objektů (struktura systému).

Vezměme si například systém, který umožňuje lidem překonat řeku se svými vozidly. Dvě zřejmé architektury jsou trajekt a most. Zatímco funkce a proces nejvyšší úrovně obou systémů – překonání řeky – jsou identické, jejich struktura a chování se dramaticky liší. Neuznání tohoto rozdílu mezi funkcí a chováním může vést k předčasnému výběru suboptimální architektury. V výše uvedeném příkladu by to mohlo znamenat rozhodnutí postavit most, aniž by se vůbec zvažila možnost trajektu.

9.5 Jazyk a modelování

Nyní se zaměříme na definice týkající se jazyka a modelování.

Jazyk je prostředek komunikace mezi lidmi a případně také stroji, který slouží k vyjádření pojmů, myšlenek, procesů a metod.

Jazyk se skládá ze dvou složek: syntaxe a sémantiky.

Syntaxe je soubor symbolů a pravidel jazyka, které určují, jak lze symboly kombinovat, aby vznikly syntakticky správné konstrukce.

Ne každá syntakticky správná konstrukce v jazyce má význam.

Sémantika je význam, který vyjadřuje podmnožina syntakticky správných konstrukcí jazyka.

9.5.1 Model a modelování

Jazyky nejen umožňují komunikaci mezi lidmi a stroji, ale jsou také prostředkem k vytváření modelů.

***Model** je abstrakcí určité části pojaté reality (systém „takový, jaký je“) nebo uvažovaného systému (systém „takový, jaký má být“) vyjádřená v nějakém jazyce.*

Například dostatečně podrobný textový popis součásti stroje v běžné angličtině lze považovat za model této součásti. Tento model však není formální, protože je vyjádřen v angličtině, což je přirozený, neformální jazyk. Proto jej nelze, alespoň s pomocí současné technologie, automaticky sestavit nebo analyzovat, ale je k tomu zapotřebí lidský zásah.

***Modelovací jazyk** je jazyk pro konstrukci modelů v určité oblasti.*

***Formální modelovací jazyk** je modelovací jazyk, který má matematicky podloženou definici syntaxe, což umožňuje jeho automatickou analýzu, kontrolu a syntézu.*

Například strojní výkresy mechanických součástí využívají formální modelovací jazyk, kreslení, ve kterém symboly vyjadřují formální syntaxi s dohodnutou sémantikou, kterou strojní inženýři rozumějí a sdílejí. Tak například čárkovaná čára vyjadřuje osu symetrie, rozměr stanovený šipkami, vodítka a textovým polem vyjadřuje rozměr součásti atd.

Formální modelovací jazyk je vyjádřen pomocí jedné nebo dvou *modalit*, tj. způsobů vyjádření. Dvě významné modalities pro vyjádření modelů jsou *grafika* a *text*. OPM je jedinečný v tom, že je jediným známým modelovacím jazykem, který tyto dvě modalities používá zaměnitelně a společně.

***Modelování** je proces vytváření modelu v určité oblasti pomocí modelovacího jazyka, který je pro danou oblast vhodný.*

Modelování je základní inženýrská činnost. Výsledný model je ústřední infrastrukturní entita, která podporuje vývoj systému po celou dobu jeho životního cyklu v kontextu „založeném na modelu“ nebo „řízeném modelem“.

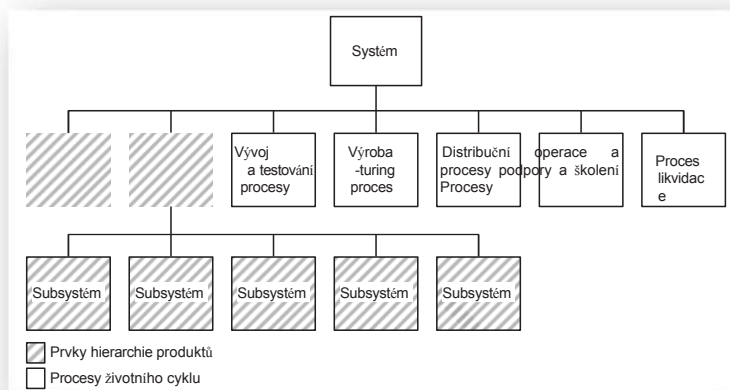
9.5.2 Neformální versus formální modely

Lidé jsou zvyklí volně kreslit neformální modely systémů. Ad hoc symboly v takových modelech jsou nekonzistentní a nelze je škálovat, což umožňuje vyjádřit pouze jednoduché systémové myšlenky. Příklad takového neformálního modelu je uveden na [obr. 9.5](#). Jak nám říká legenda, šrafování rámečků rozlišuje mezi procesy životního cyklu a „hierarchií produktů“. Tento ad hoc model ponechává mnoho otázek nezodpovězených. Například, jaká je sémantika implikované hierarchie? Jedná se o agregaci? Specializaci? Proč systém obsahuje kromě produktů také procesy životního cyklu? Proč jeden produkt sestává z pěti subsystémů a druhý z žádného? Zajímavé je, že tento model se objevuje v mezinárodní normě (ISO/IEC 26702 IEEE Std. 1220-2005), která by ze všech dokumentů měla zachovávat nejvyšší možnou úroveň formálnosti. Je zřejmé, že tomuto modelu chybí formalita a jeho prezentace jako součásti mezinárodní normy může být zavádějící více než jeho vynechání.

***Formální model** je model vyjádřený formálním modelovacím jazykem.*

Pokračujeme-li v příkladu strojního výkresu, výkres součásti je formálním trojrozměrným modelem této součásti. Systém CAD/CAM, který je navržen tak, aby „rozuměl“ tomuto jazyku, může automaticky generovat

skutečnou součástí z tohoto modelu. Dalším příkladem je Newtonův druhý zákon, $F = m \times a$, který je formálním modelem vztahu mezi silou, hmotností a zrychlením tuhého tělesa, vyjádřeným matematickou rovnicí. Zajímavé však je, že tuhé těleso, kterého se tento model týká a jehož atributy jsou hmotnost a zrychlení, se v tomto modelu nikde nevyskytuje. Je spíše implicitní, že je předmětem tohoto modelu. To stále odpovídá naší definici modelu jako abstrakce určité části pojaté reality.



Obr. 9.5 Příklad neformálního modelu – základní stavební bloky systému (ISO/IEC 26702 IEEE Std. 1220-2005)

Koncepční model je formální model systému, který vyjadřuje jeho architekturu tím, že popisuje jeho strukturu a chování do takové míry detailu, která je dostatečná pro jeho následný podrobný návrh a případnou realizaci.

Část OPM, která specifikuje, jak konstruovat diagramy objektů a procesů (OPD) spolu s jejich textovými reprezentacemi v OPL, je příkladem koncepčního modelovacího jazyka. Dalším příkladem je SysML.

Koncepční modelovací jazyk je formální modelovací jazyk pro konstrukci koncepčních modelů systému.

9.5.3 Řízení složitosti

V dalších kapitolách podrobně diskutujeme, jak OPM řeší řízení složitosti. Stručně řečeno, každá věc (objekt nebo proces) může projít dvěma mechanismy zpřesnění: přiblížením a rozvinutím. Přiblížení procesů specifikuje podprocesy procesu a jejich časové pořadí. Přiblížení objektů specifikuje části objektu a (zhruba, v relevantním a možném rozsahu) také jejich prostorové pořadí (v současné době pouze

schematicky a ve dvou rozměrech). Rozbalení věcí (objektů nebo procesů) odhaluje jejich části, vlastnosti (atributy nebo operace), specializace nebo instance. Oba procesy zpřesňování – přiblížení a rozbalení – lze provést ve stejném OPD nebo v novém OPD. Nové zpřesnění OPD (přiblížení nebo rozbalení) vytvoří nové OPD, ve kterém je zpřesněná věc rozpracována tak, aby vyjadřovala více detailů.

9.6 Shrnutí

- Vědu lze považovat za reverzní inženýrství přírody.
- Princip minimální ontologie říká, že pokud lze systém specifikovat se stejnou přesností a podrobností dvěma jazyky různé ontologické velikosti, pak je preferován jazyk s menší velikostí před jazykem s větší velikostí.
- Objekty existují, procesy se odehrávají.
- Ontologie je soubor pojmů a jejich vztahů v určité oblasti diskurzu.
- Minimální univerzální ontologie je ontologie, která je nezbytná a dostačující pro modelování vesmíru a systémů v něm.
- Věta o objektech a procesech: Objekty se stavem, procesy a vztahy mezi nimi tvoří minimální univerzální ontologii.
- Objektově-procesní tvrzení: Pomocí objektů se stavem, procesů a vztahů mezi nimi, spolu s mechanismy zjemňování a rozvíjení, lze koncepčně modelovat systémy v jakékoli doméně a na jakékoli úrovni složitosti.
- Princip důležitosti věci OPM: Důležitost věci T v modelu OPM přímo souvisí s nejvyšší OPD v hierarchii OPD, kde se T objevuje.
- Objekt je věc, která existuje nebo může existovat fyzicky nebo informačně.
- Stav je možná situace nebo pozice, ve které se objekt může nacházet po určitou dobu.
- Transformace objektu je (1) vytvoření (generování, konstrukce), (2) spotřeba (eliminace, destrukce) nebo (3) účinek – změna stavu daného objektu.
- Proces je něco, co transformuje objekt.
- Princip transformace objektu procesem OPM: V kompletním modelu OPM musí být každý proces propojen s alespoň jedním objektem, který proces transformuje, nebo s jedním stavem objektu, který proces transformuje.
- Systém je objekt poskytující funkce.
- Zainteresovaná strana je jednotlivec, organizace nebo skupina lidí, kteří mají zájem o systém, který se zvažuje, vyvíjí nebo nasazuje, nebo kteří mohou být tímto systémem ovlivněni.
- Příjemce je zainteresovaná strana, která ze systému získává hodnotu a výhody.
- Zákazník je zainteresovaná strana, která objednává systém a sponzoruje jeho vývoj, implementaci, nasazení a podporu.

- Uživatel je zainteresovaná strana, která systém provozuje nebo s ním přímo komunikuje.
- Dodavatel je zúčastněná strana, která dohlíží na vývoj, podporu a údržbu systému nebo produktu.
- Funkce umělého systému je jeho proces poskytování hodnot na nejvyšší úrovni, jak jej vnímá příjemce.
- Struktura systému je jeho forma – souhrn jeho fyzických a informačních komponent spolu s dlouhodobými vztahy mezi nimi.
- Chování systému je jeho dynamika – způsob, jakým se systém mění v čase transformací systémových (vnitřních) a/nebo environmentálních (vnějších) objektů.
- Architektura systému je kombinací struktury a chování systému, která mu umožňuje plnit jeho funkci.
- Prostředí systému je soubor objektů, které se nacházejí mimo systém, ale interagují s ním, což způsobuje změny systému a případně i jeho prostředí.
- Rozdíl mezi funkcí a chováním: Chování je způsob, jakým se systém mění v čase, zatímco funkce je hodnota, kterou systém poskytuje svému příjemci prostřednictvím svého provozu.
- Jazyk je prostředek komunikace mezi lidmi a případně také stroji, který slouží k vyjádření pojmů, myšlenek, procesů a metod.
- Syntaxe je soubor symbolů a pravidel jazyka, které určují, jak lze symboly kombinovat, aby vznikly syntakticky správné konstrukce.
- Sémantika je význam, který vyjadřuje podmnožina syntakticky správných konstrukcí jazyka.
- Model je abstrakcí určité části pojaté reality nebo uvažovaného systému vyjádřeného v určitém jazyce.
- Modelovací jazyk je jazyk pro konstrukci modelů v určité oblasti.
- Formální modelovací jazyk je modelovací jazyk, který má matematicky podloženou definici syntaxe, což umožňuje jeho automatickou analýzu, kontrolu a syntézu.
- Formální model je model vyjádřený ve formálním modelovacím jazyce.
- Koncepční model je formální model systému, který vyjadřuje jeho architekturu tím, že popisuje jeho strukturu a chování do takové míry detailu, která je dostatečná pro jeho následný podrobný návrh a případnou realizaci.
- Koncepční modelovací jazyk je formální modelovací jazyk pro konstrukci koncepčních modelů systémů.

9.7 Problémy

1. S odkazem na OPD na [obr. 7.4](#) najděte:
 - Proces, který je důležitější než objekt,

- objekt, který je důležitější než proces,
 - objekt a proces stejného významu,
 - dva objekty stejného významu a
 - dva objekty stejného významu.
2. Vysvětlíte, proč odstranění stavových objektů, procesů nebo vztahů mezi nimi z minimální univerzální ontologie způsobuje, že je nepoužitelná.
 3. Vytvořte model malého OPD, který je syntakticky správný, ale sémanticky nesprávný.
 4. Vysvětlíte souvislost mezi transformací objektu procesem OPM a definicí procesu.
 5. Definujte dvě architektury pro každý ze systémů, které poskytují následující
 - a. Překročení řeky
 - b. Zobrazení denní doby
 - c. Prodloužení trvanlivosti potravin
 - d. Přeprava osob
 - e. Sledování filmů

Kapitola 10

Věci: předměty a procesy

Každé konvexní zrcadlo musí mít ... na spodním okraji odrazové plochy zrcadla ... nápis „Objekty ve zrcadle jsou blíže, než se zdá“.

USA, ČÁST 571 Federální bezpečnostní normy pro motorová vozidla, [oddíl 571.111](#) S5.4.2 (2004)

Immanuel Kant řekl, že „objekty jsou naším způsobem poznávání“. I když je to samozřejmě pravda, není to celá pravda, ale pouze její polovina. Objekty jsou naším způsobem poznávání toho, co *existuje*, nebo jinými slovy, *struktury* systémů. Abychom věděli, co *se děje*, abychom pochopili chování systémů, potřebujeme druhý, doplňkový typ věcí – procesy. O existenci objektu víme, pokud jej můžeme pojmenovat a odkazovat se na jeho bezpodmínečnou, relativně stabilní existenci, ale bez procesů nemůžeme říci, jak se tento objekt transformuje – jak vzniká, jak se jeho stavy mění v čase a jak mizí. Tyto dva základní pojmy – objekty a procesy, zobecněné jako věci – jsou předmětem této kapitoly.

10.1 Objektově orientovaný versus objektově-procesní přístup

Jak jsme viděli v oddíle 9.2.5, objekty a procesy jsou dva typy univerzálních stavebních bloků OPM a procesy jsou modelovány jako „prvotřídní občané“, kteří nejsou podřízeni objektům. Tato orientace na objekty a procesy je zásadním odklonem od objektově orientovaného (OO) softwarového paradigmatu, které klade objekty jako jediné hlavní aktéry. Objekty „vlastní“ procesy, které se v OO nomenklatuře často nazývají „operace“, „služby“ nebo „metody“.

Hlavní procesy na úrovni systému mohou být stejně důležité nebo dokonce důležitější než objekty v systémovém modelu. Proto musí být procesy modelovatelné nezávisle na konkrétní třídě objektů. To je v souladu s principem důležitosti věcí OPM, který byl představen v předchozí kapitole a který uvádí, že důležitost věci T v modelu OPM přímo souvisí s nejvyšší OPD v hierarchii OPD, kde se T objevuje. Tento paradigmat rovnosti objektů a procesů umožňuje OPM koncepčně modelovat reálné systémy v grafice a textu.

Schopnost rozlišit objekty a procesy a správně je používat v modelu je klíčem k zvládnutí OPM. Abychom definovali tyto základní pojmy a sdělili jejich sémantiku, nejprve budeme diskutovat o „existenci“ a „změně“, čímž položíme základ pro definici objektů a procesů a jejich rozlišení. Poté představíme „podstatu věcí“ a prozkoumáme rozdíl mezi „fyzickými“ a „informatickými“ věcmi. Slovo *informatický* nebo *kybernetický* odkazuje na zobecnění souvislosti s daty, informacemi, znalostmi, odborností nebo vynalézavostí bez jakéhokoli odkazu na jejich fyzickou podobu.

10.2 Existence, věci a transformace

Websterův nový slovník (1997) definuje *existenci* jako podstatné jméno odvozené od slovesa existovat, což znamená *být, mít bytí, pokračovat v bytí*. Existovat znamená vyniknout, ukázat se a mít identifikovatelnou, zřetelnou jedinečnost ve fyzické nebo mentální sféře. Věc, která existuje ve fyzické realitě, má „hmatatelnou existenci“ na určitém místě a v určitém čase. Protože vyniká a ukazuje se, můžeme na ni ukázat a říci: „Tady je.“

Vyniknout znamená prezentovat stabilní formu na pozadí něčeho jiného, co existuje. Pojem „pozadí“ je zásadní, protože kdyby nic jiného neexistovalo, nemohl by existovat kontrast jedné věci, která vyniká a odlišuje se od pozadí věcí, které existují spolu s ní. Stabilní forma, kterou musí existující věc vykazovat, je „podstatně neměnná“ po dostatečně dlouhou dobu (vzhledem k typické rychlosti změny pozadí), aby mohla být rozpoznána jako „vychýlující“. To, co nemůžeme identifikovat ani nějakým způsobem odvodit jeho identitu, pro nás nemůže existovat. Jinými slovy, „vychýlívát“ vyžaduje nepřetržitou identifikovatelnost po přiměřenou dobu, ať už fyzicky nebo informačně.

Zvažujeme-li existenci v časové dimenzi, existují dva způsoby „vynikání“ nebo existence věcí. V prvním případě dochází k „vynikání“ během pozitivního, relativně podstatného časového období. Toto „vynikání“ musí být pozorovatelné v podstatě neměnné, stabilní nebo trvalé formě. To, co v tomto způsobu vyniká, nazýváme *objektem*. Websterův slovník (1997) definuje objekt jako *hmotnou věc; to, na co je zaměřen pocit nebo akce; cíl nebo záměr; slovo závislé na slovesu nebo předložce*. Sloveso, na kterém objekt „závisí“, je syntaktickým projevem procesu. Slovník Dictionary.com definuje sloveso jako „klíčové slovo ve většině vět, slovo, které odhaluje, co se děje“. Vzor v našich myslích toho, „co se děje“, je proces.

10.2.1 Objekt upřesněn

Starší verze Websterova slovníku (1984) uvádí dvě odlišné relevantní definice objektu:

- *Cokoli, co je viditelné nebo hmatatelné a má stabilní formu.*
- *Cokoli, co lze pochopit intelektuálně.*

Tyto dvě definice odpovídají našim představám o *fyzických a informatických* (nebo *kybernetických*) objektech. První definice je ta, kterou obvykle máme na mysli, když používáme termín objekt v běžném jazyce. Druhá definice se týká informatické, koncepční, kybernetické, logické a nehmotné stránky objektů. Informatické objekty se liší od svých fyzických protějšků v tom, že nemají fyzickou existenci, a proto nepodléhají zákonům fyziky. Nositelem informatického objektu je však fyzický objekt; *existence* informatických objektů závisí na tom, zda jsou symbolicky zaznamenány, vepsány, vyraženy nebo vyryty na nějakém fyzickém médiu: kameni, papýru, papíru, elektromagnetickém médiu nebo skupině neuronů v mozku. Zde se fyzické a informační aspekty informačního objektu protínají, což vede ke vzniku pojmů, jako je šum a souvislost mezi [statistickou mechanikou](#) a [informační teorií](#) prostřednictvím entropie (Shannon a Weaver 1949).

Jelikož objekty OPM jsou fyzické nebo informatické, definujeme objekt jako něco, co zachycuje tyto dvě stránky, aniž by se zavazovalo k jedné z nich, a zároveň zahrnuje prvek „existence v čase“.

Objekt je věc, která existuje nebo má potenciál fyzické nebo informační existence.

Tato definice se značně liší od klasických definic objektu v literatuře o objektově orientovaném programování, které lze formulovat jako „*Objekt je abstrakce atributů a operací, která má pro systém význam.*“ Například v elektronické knize [Object-Oriented Programming Basics with Java \(Základy objektově orientovaného programování v jazyce Java\)](#) je objekt definován jako „*zapouzdřený, zcela specifikovaný datový agregát obsahující atributy a chování.*“

10.2.2 Objekty a lidské vzpomínky

Kvalifikovat lidský mozek jako hmotné médium, které může ukládat nehmotné věci, se některým čtenářům může zdát cynické nebo nevhodné. Zaslouží si proto zvláštní diskusi a odůvodnění. Centrální nervový systém, jehož hlavní součástí je mozek, je informačním systémem lidí a jiných organismů. Řídí a reguluje celý organismus. Lidská paměť nebo mentální záznam věci je stále převážně záhadným způsobem, jakým se věc vryje do mysli, ale v porozumění struktuře a funkci mozku se neustále dosahuje pokroku (např. Kostovic a Rakic 1990).

Mezi mnoha dalšími, radostnějšími schopnostmi inteligence a emocí, je úžasná schopnost lidského mozku pamatovat si věci, která z něj činí vynikající záznamové médium. Lidský mozek ukládá obrovské množství dat, informací a znalostí různých forem, které jsou nezbytným základem inteligence, včetně dedukce, predikce, rozhodování a chování. Lidské vzpomínky nejsou jen řadou objektů představujících fakta, obrazy, tváře, jména, tvary, postavy, formy a symboly. Zahrnují také strukturální a behaviorální vztahy, které mezi těmito objekty existují, a pravidla, která je řídí. Cokoli, co je zaznamenáno v lidském mozku, je informatický objekt. Tento informatický objekt může být záznamem nějaké (hmotné a/nebo nehmotné) sady objektů a procesů, kterými objekt v sadě procházejí.

10.3 Identita objektu

Identita objektů je důležitá, ale těžko uchopitelná. Fyzické objekty musí být zacházeno jinak než s informatickými objekty. Vzhledem k tomu, že fyzický objekt je tvořen hmotou (nebo energií, která podle Einsteinových učením může být přeměněna na hmotu a naopak), jsou dvě instance fyzického objektu identické, pouze pokud zabírají stejný prostor ve stejném čase. To je možné pouze tehdy, pokud se jedná o skutečně stejný objekt, což znamená, že žádné dvě odlišně identifikovatelné instance fyzického objektu nejsou stejné. Dvě nové identické automobily modelu X, které právě sjely z montážní linky, jsou tedy různými instancemi stejné třídy objektů.

10.3.1 Identita informatických objektů

Situace s identitou objektů je odlišná, pokud jde o informatické objekty, protože zde je podstatnou vlastností objektu myšlenka, koncept, vzor nebo symbol, který reprezentuje, spíše než fyzická hmota, která jej dokumentuje. Z hlediska fyzického média je každá instance informatického objektu, jako například kopie stejné knihy, odlišná, stejně jako dva automobily sjíždějící z montážní linky. Z

informatického hlediska jsou však všechny fyzické kopie určitého informatického objektu stejné. Dvě kopie stejné knihy jsou identické, pokud jde o jejich informatický obsah (sémantiku). Jsou vytištěny na samostatných stránkách a svázané jako dvě odlišné fyzické instance objektu. I když je jedna kopie papírová a druhá elektronická, z informatického hlediska jsou stále stejné.

Z informatického hlediska jsou dva identické (papírové nebo počítačové) soubory obsahující plány a výrobní pokyny pro automobil modelu X stejným objektem, protože informatický obsah, který přenášejí, je identický. Fyzicky se média, na kterých je tento fyzický objekt zaznamenán, liší, protože se jedná o fyzickou hmotu, která podléhá přírodním zákonům. Stejně tak se dvě kopie stejného souboru fyzicky liší, protože zabírají odlišné adresní prostory v primární nebo sekundární paměti počítače. Z informatického hlediska jsou však identické.

10.3.2 Proces jako metafora transformace

Zaznamenali jsme, že existují dva způsoby, jak vyniknout. První je v prostoru, druhý v čase. V časovém režimu „vynikání“ se stále jedná o vynikání objektu, ale tentokrát k němu dochází „měněním se způsobem“ na pozadí, které je v podstatě stabilní. Protože objekt, který vyniká, prochází transformací, může mít před a po transformaci různá jména. Je vhodné si představit věc, která způsobila transformaci, jako nějaký nosič, který je za tuto transformaci „odpovědný“.

Když máme sklon uvažovat tímto způsobem, ve skutečnosti přemýšlíme o vzorové změně, o sérii transformací, kterými prochází jeden nebo více objektů. Pro usnadnění jazyka nebo uvažování spojujeme tuto vzorovou změnu s „nosičem“, kterému mentálně přiřazujeme „odpovědnost“. *Transformaci* definujeme jako zobecnění změny, vzniku a zničení objektu.

Transformace je generování (konstrukce, tvorba) nebo spotřeba (destrukce, eliminace) nebo změna (účinek, přechod stavu) objektu.

10.3.3 Upřesnění definice procesu

Podle Websterova slovníku (1997) je proces „stav pokračování, série akcí a změn, způsob fungování, působení zákona, důsledek“. American Heritage Dictionary (1996) definuje proces jako „sérii akcí, změn nebo funkcí, které vedou k výsledku“. V Dictionary.com je sloveso, které je zhruba syntaktickým analogem procesu, definováno jako „klíčové slovo ve většině vět, slovo, které odhaluje, co se děje“.

Nositel, který způsobuje transformaci, nazýváme *procesem* a říkáme, že proces je „to, co způsobilo transformaci“ objektu. Tento nositel je však pouze metaforou, protože proces nemůžeme „uchopit“ ani se ho dotknout, i když může být zcela fyzický, protože zahrnuje transformaci jednoho nebo více fyzických objektů. Jediné, čeho se můžeme dotknout, co můžeme vidět nebo vnímat jiným způsobem, je transformovaný objekt. Můžeme měřit jednu nebo více *hodnot atributů* objektu v určitých časových bodech nebo v průběhu procesu transformace tohoto objektu. Můžeme například měřit hodnoty (v stupních Celsia) atributu teploty železných tyčí v průběhu procesu zahřívání, nebo se jich dotknout a cítit, jak se zahřívají ve srovnání s nějakým minulým časovým bodem, ale nemůžeme se dotknout procesu zahřívání.

V jakémkoli daném časovém bodě před, během nebo po výskytu procesu se pozorovaný objekt může potenciálně lišit od toho, jaký byl v předchozím časovém bodě. Pomocí naší lidské paměti získáváme pocit procesu porovnáním současné podoby transformovaného objektu s jeho minulou podobou. **Proces tedy *existuje pouze jako koncept, mentální konstrukt v lidské mysli***. Procesům dáváme jména, abychom odkazovali na měnící se vzorce objektů. Zaměřujeme-li se na transformaci, přijímáme následující definici.

Dříve jsme řekli, že objekty existují a procesy se dějí. Zde jsme právě řekli, že proces existuje, ale pouze jako mentální konstrukt. V tomto ohledu bychom mohli o procesech uvažovat také jako o (mentálních) objektech a navrhnout modelovací paradigma, které je založeno pouze na objektech jako „prvotřídních občanech“ a má pravděpodobně ještě kompaktnější univerzální ontologii než OPM. Jedná se skutečně o objektově orientovaný přístup. Jak však ukážeme v celé knize, hodnota přidání procesu jako konceptu do univerzální ontologie, který je oddělený od objektu, daleko převyšuje cenu přidání dalšího konceptu do této ontologie.

Proces je mentální konstrukt představující vzorec transformace objektu.

Tato definice procesu působícího na objekt okamžitě implikuje, že žádný proces nemá význam, pokud není spojen s alespoň jedním objektem – tím, který proces transformuje. Transformace objektu (objektů) je nezbytným a nevyhnutelným výsledkem provedení procesu. Toto je první případ, kdy dochází k narušení symetrie mezi objekty a procesy. Zatímco jsme definovali a mohli odkazovat na objekt, aniž bychom nutně používali termín „proces“, schopnost definovat a uvažovat o procesu, včetně jeho transformace, závisí na existenci alespoň jednoho objektu, který je tímto procesem transformován.

S odkazem na syntaktický význam objektu poskytuje Dictionary.com následující definici:

Gramatika. (v mnoha jazycích, jako je angličtina) podstatné jméno, podstatná fráze nebo náhrada podstatného jména, která svou syntaktickou pozici představuje buď cíl akce slovesa, nebo cíl předložky v předložkové frázi, jako například míč ve větě John hit the ball...

Zde, stejně jako v naší definici, je vytvořena vazba mezi objektem a slovesem, které je procesem. Ve větě *John hit the ball* je hitting procesem a ball objektem. Tento příklad ukazuje, že *syntaktický* termín objekt – to, na co je akce zaměřena – se často shoduje se *sémantickým* termínem objekt. Sémanticky je však John také objektem (instancí objektu osoba), zatímco syntakticky je subjektem. Syntaktický termín *sloveso* je často analogický se *sémantickým* termínem *proces*. Podrobněji se tím zabýváme v oddíle 10.6, kde diskutujeme test procesu.

10.3.4 Definice transformovaného prvku

Když říkáme, že proces vedl ke vzniku objektu, máme na mysli, že objekt, který před vznikem procesu neexistoval, nyní existuje – je identifikovatelný na pozadí svého okolí. Analogicky, když řekneme, že proces vedl k *eliminaci* objektu, máme na mysli, že objekt, který dříve vynikal, nelze identifikovat, takže již neexistuje. Tyto radikální změny vzniku a eliminace jsou extrémními verzemi transformace. Méně radikální transformací je změna stavu objektů. Objekt, který proces transformuje, se nazývá *transformát*.

Transformátorem procesu P je objekt, který P transformuje.

Přípona „ee“, jako v případě slova employee (zaměstnanec), se zde a v několika dalších případech, které budou brzy definovány, používá k vytvoření nového slova, které označuje objekt, na který působí proces (sloveso) X. Zde X = Transformovat. Brzy se setkáme také s pojmy Consume (spotřeba), Result (výsledek) a Affect (ovlivnění).

V teoretickém, zmrazeném, statickém vesmíru při absolutní nule neexistují žádné procesy a nedochází k žádné transformaci. Bez procesů můžeme popsat pouze statické, trvalé strukturální vztahy mezi objekty. V reálných pozemských podmínkách mají procesy a objekty srovnatelný význam jako stavební kameny při popisu a porozumění přírodním systémům a vesmíru jako celku (což je posláním vědy) a při navrhování umělých systémů (což je posláním inženýrství).

10.3.5 Příčina a následek

Jedním z poznatků vyplývajících z výzkumu časového vztahu je *příčina a následek*. Určité objekty, jsou-li uvedeny do správného prostorového a časového vztahu (např. jsou na „stejném“ místě ve „stejný“ čas), umožňují, aby došlo k procesu, který způsobí transformaci alespoň jednoho objektu: Když proces skončí, alespoň jeden z objektů, které se ho účastnily (jako vstup, výstup nebo obojí), je transformován (spotřebován, vygenerován nebo změněn). „Příčina“ v idiomu „příčina a následek“ je spouštěcí událost, ke které dochází při souběžné nebo jinak časově koordinované přítomnosti souboru objektů, z nichž některé mohou vyžadovat určitý stav. „Následek“ v tomto idiomu „příčina a následek“ je transformace, kterou prochází jeden nebo více z těchto objektů.

Například chod spalovacího motoru závisí na přítomnosti objektů směsi vzduchu a benzinových par uvnitř válce objektu při správném tlaku a teplotě (atributy směsi). Spouštěcí událostí je okamžik, kdy jiskra (vytvořená předchozím časovaným procesem) zapálí směs. V důsledku tohoto procesu se směs benzínu spotřebuje a hodnota kinetické energie pístu se zvýší. Ve zpětné vazbě jsou příčina a účinek kruhové: Účinek v daném okamžiku je příčinou pozdější změny.

10.4 Syntaxe versus sémantika

Aby bylo možné odkazovat na věci (objekty a procesy) a rozlišovat mezi nimi, přirozené jazyky vyvinuté lidmi za účelem komunikace přiřazují věcem jména. Jméno věci představuje primární identifikační symbol této věci, díky čemuž je možné na ni odkazovat a komunikovat o ní. Tato jména věcí se nazývají *podstatná jména*. Podstatné jméno je však jako součást řeči syntaktickým termínem, zatímco objekty a procesy jsou sémantickými termíny. Tuto otázku podrobněji rozvedeme dále.

10.4.1 Jsou objekty a procesy sémantickými analogy podstatných jmen a sloves?

V přirozených jazycích jsou objekty téměř vždy syntakticky reprezentovány jako podstatná jména. Procesy jsou syntakticky často reprezentovány jako slovesa, ale mohou být také podstatnými jmény. Například slovo „cihla“ je syntakticky podstatné jméno a sémanticky objekt, zatímco slovo „stavět“ je sloveso a proces. Slovo „stavba“ v kontextu „procesu stavby“ je však také podstatné jméno, i když sémanticky je stejné jako „proces stavění“. Pro ilustraci uvádíme, že výraz „*proces stavění*“ je přijatelný, zatímco výraz „*proces cihly*“ přijatelný není. Stejně tak je přijatelný výraz „*cihlový objekt*“, zatímco výraz „*stavební objekt*“ (kde *objekt* není synonymem pro *cíl*) je mnohem méně přijatelný. Ještě více

je objekt budování (podstatné jméno), který je výsledkem procesu budování (stavění). Píše se a vyslovuje se stejně jako proces budování (sloveso). Pouze z kontextu ve větě lze tyto dvě sémanticky odlišné slova rozlišit.

Běžnou strategií návrhu softwaru je strategie návrhu orientovaná na podstatná jména/přídavná jména/slovesa (MacIntyre 2010). Ve svém blogu MacIntyre napsal:

... Poté jsem se naučil C++, objektově orientované programování, a seznámil se se svatým grálem rad pro objektově orientovaný design, který zněl asi takto: Vezměte své požadavky a zakroužkujte všechna podstatná jména, ta jsou vašimi třídami. Poté podtrhněte všechna přídavná jména, ta jsou vašimi vlastnostmi. Poté zvýrazněte všechna slovesa, ta jsou vašimi metodami.

Tato strategie návrhu podstatných jmen/přídavných jmen/sloves se zdála být nejgeniálnějším programovacím moudrem, jaké kdy bylo vysloveno... ale vedla nás na scestí. Je to sloveso, které je špatně pochopeno. **Sloveso by mělo být jinou třídou, ne metodou.** Mělo by to být procesní třída. Jako programovací koncept je proces stejně tak „věcí“ jako jakýkoli objekt v reálném světě. Sloveso by mělo být třídou, která přijímá podstatné jméno jako vstup, který má být zpracován.

Zajímavé je, že MacIntyre intuitivně dospěl k závěru, že sloveso je „*další* třída“ (zdůraznění v zdroji). Uvědomil si, že proces není méně důležitý než objekt, a proto by neměl být metodou vlastněnou objektem, ale „třídou procesu“ samou o sobě.

Výše uvedené příklady ukazují, že lákavé tvrzení, že objekt a proces jsou sémantickými analogy syntaktických pojmů podstatné jméno a sloveso, je v nejlepším případě hrubé a nepřesné. Místo toho, abychom se spoléhali na syntaktické pojmy slovních druhů, musíme proto vytvořit sémantický, obsahově založený způsob analýzy slov ve větě, který nám umožní rozlišit objekty od procesů. To nám umožní překonat úskalí a zvláštnosti přirozených jazyků.

10.4.2 Syntaktická versus sémantická analýza vět

Obtížnost, s níž se často setkáváme při nezbytném a dostatečném rozlišování mezi objekty a procesy, má kořeny v našem vzdělání: jako studenti střední školy jsme byli vedeni k tomu, abychom uvažovali a analyzovali věty z hlediska syntaktických slovních druhů – podstatných jmen, sloves, přídavných jmen a příslovčí – spíše než z hlediska sémantických, hlubších významových pojmů – objektů, procesů, atributů a operací. To pravděpodobně platí pro jakýkoli přirozený jazyk, který studujeme a používáme, ať už je to náš mateřský jazyk nebo cizí jazyk.

Stejnou myšlenku lze velmi často vyjádřit více než jednou větou, což vede k různému přiřazení slovních druhů. *Sémantická analýza vět* je rozbor věty podle její sémantiky, nikoli syntaxe. Pouze prostřednictvím sémantické analýzy věty můžeme překonat povrchní rozdíly ve vyjádření a dostat se k záměru autora nebo mluvčího určitého textu. Nicméně myšlenka sémantické analýzy vět, ve které hledáme hluboký význam věty pod jejím povrchem, je pravděpodobně relativně méně přijímaná.

Aby bylo možné OPM užitečně aplikovat, je třeba umět analyzovat věty sémanticky. To v první řadě znamená rozlišovat mezi objektem a procesem. Jak to systematicky provádět, je tématem následujících částí. Nejprve definujeme tři sady objektů s ohledem na jejich účast a roli v procesu.

10.4.3 Sada objektů předběžného zpracování

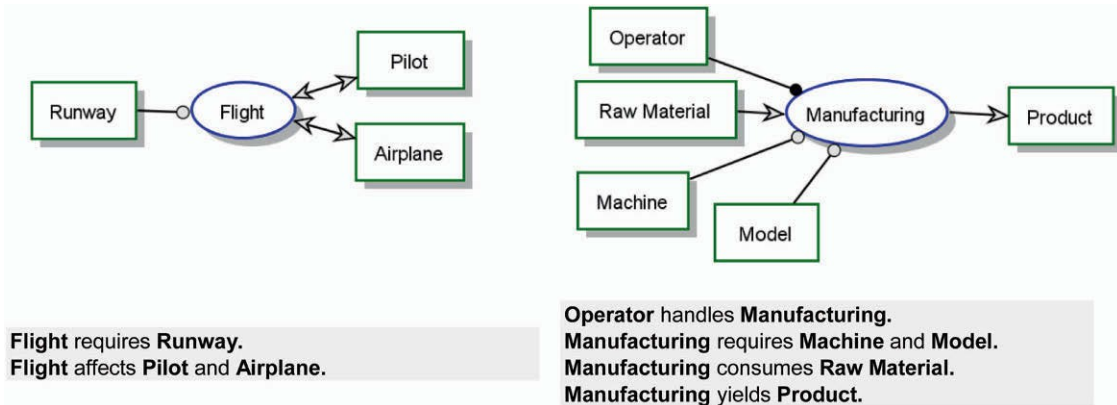
Aby proces mohl začít, musí být spuštěn. Toto spuštění může být vnější, například tím, že objekt začne existovat nebo je k dispozici, nebo tím, že objekt vstoupí do určitého stavu, nebo vnitřní, například událostí, která označuje konec předchozího procesu v kontextu vyššího, přibližného procesu. Jakmile je proces spuštěn, „pokouší se“ fungovat (vyskytnout se, nastat nebo provést se). Za tímto účelem musí zkontrolovat existenci sady objektů – sady objektů předběžného zpracování –, která by umožnila jeho provedení.

Sada předzpracovávaných objektů procesu P , $Pre(P)$, je sada objektů, které musí existovat, případně v určitých stavech, aby mohl P začít provádět, jakmile byl spuštěn.

Samotný spouštěcí objekt není součástí sady objektů předběžného zpracování. Existence sady objektů předběžného zpracování je předpokladem procesu – podmínkou pro jeho spuštění. Jako proces neexistuje podstatné jméno, které by jej reprezentovalo, ale spíše se vyskytuje, děje se, funguje, provádí se, transformuje, mění nebo upravuje alespoň jedno další podstatné jméno, které by bylo objektem.

Uvažujme dva příklady procesů: **Let** a **Výroba**, znázorněné na [obr. 10.1](#). V příkladu **Let** (OPD vlevo) jsou **Letadlo**, **Pilot** a **Přistávací dráha** objekty v sadě objektů předprocesu, protože bez nich nemůže **Let** nastat. V notaci sady: $Pre(\text{Let}) = \{\text{Letadlo}, \text{Pilot}, \text{Přistávací dráha}\}$.

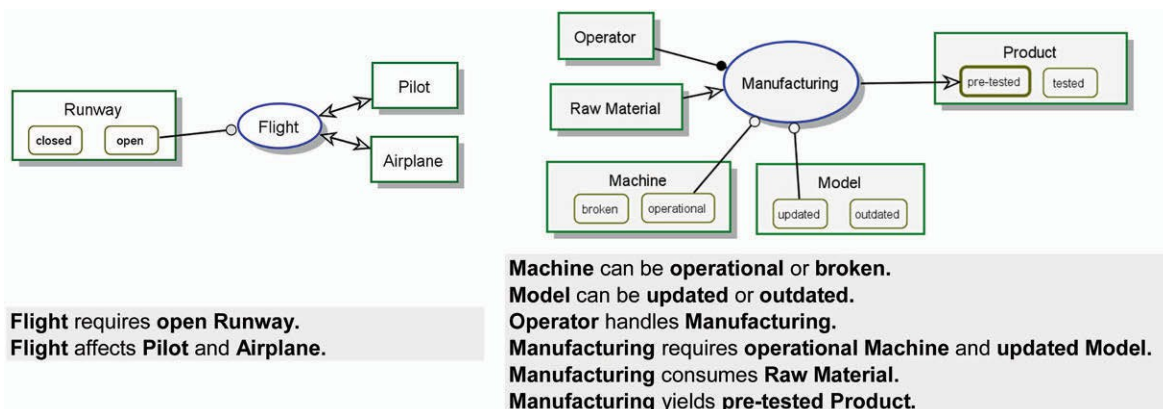
V případě **výroby** (OPD vpravo) se sada objektů předběžného zpracování skládá ze **surovin**, **operátora**, **stroje** a **modelu**: $Pre(\text{výroba}) = \{\text{suroviny}, \text{operátor}, \text{stroj}, \text{model}\}$. **Produkt** v této sadě není, protože ještě neexistuje a není potřebný pro zahájení procesu.



Obr. 10.1 Příklady sad objektů před a po zpracování

Na stavy některých objektů v sadě objektů předběžného zpracování mohou být kladeny požadavky. Například, jak ukazuje OPD vlevo na [obr. 10.2](#), aby **letadlo** mohlo vzlétnout, je nutné, aby byla **dráha otevřená**. V notaci sady: $Pre(\text{Flight}) = \{\text{Airplane}, \text{Pilot}, \text{open Runway}\}$. Jinými slovy, toto je vyjádřeno v odpovídající větě OPL:

Flight vyžaduje **otevřenou Runway**.



Obr. 10.2 Příklad objektu se specifikovaným stavem, **otevřená Runway**, v sadě objektů předzpracování **Flight**

Podobně, jak ukazuje OPD vpravo na [obr. 10.2](#), aby mohla proběhnout **výroba**, je nutné, aby byl **stroj funkční** a **model aktualizován**. V tomto případě bude výsledkem **předem otestovaný produkt**. V notaci množin: $Pre(\text{výroba}) = \{\text{operátor}, \text{surovina}, \text{funkční stroj}, \text{aktualizovaný model}\}$. To je vyjádřeno ve třech odpovídajících větách OPL:

Operátor provádí výrobu.

Výroba vyžaduje funkční stroj a aktualizovaný model. Výroba spotřebovává suroviny.

10.4.4 Sada objektů postprocesu

Sada objektů postprocesu je definována analogicky k sadě objektů předprocesu následovně.

Sada objektů po zpracování procesu P , $Post(P)$, je sada jednoho nebo více objektů, které existují, případně v určitých stavech, po dokončení provádění P .

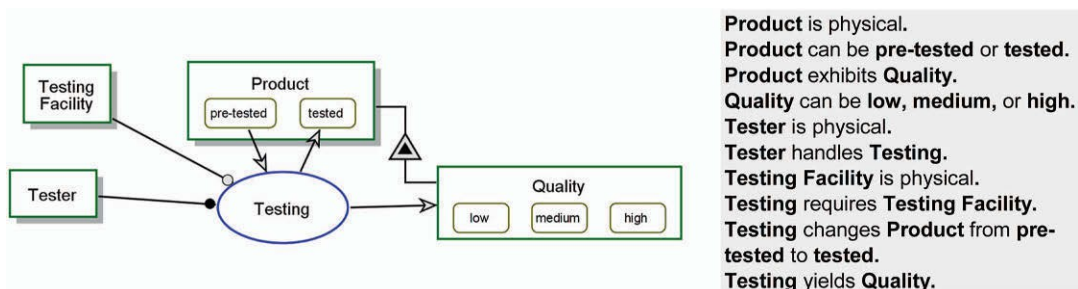
Existence všech objektů v sadě objektů postprocesu, některé případně v určených stavech, je *postpodmínka* tohoto procesu.

Sada objektů před zpracováním a sada objektů po zpracování nemusí být nutně disjunktní; mohou se alespoň částečně překrývat. V příkladu **letu** na [obr. 10.1](#) jsou všechny tři objekty v sadě objektů před zpracováním, **letadlo**, **pilot** a **přistávací dráha**, také v sadě objektů po zpracování: $Post(\text{Flight}) = \{\text{Airplane}, \text{Pilot}, \text{Runway}\}$. Měli bychom však poznamenat, že transformovány jsou pouze **objekty Airplane a Pilot**: jejich atribut **Location** se mění z **původního místa** na **místo určení**. Na [obr. 10.2](#) to není modelováno explicitně, pouze implicitně, s uvedením, že objekty **Airplane a Pilot** procházejí určitou změnou stavu.

V příkladu **výroby** na [obr. 10.1](#) jsou **suroviny, operátor, stroj a model** v sadě objektů před zpracováním, zatímco **operátor, stroj, model a produkt** jsou v sadě objektů po zpracování: $Post$

(Výroba) = {Operátor, Stroj, Model, Produkt}. Surovina se transformuje tím, že se spotřebuje, zatímco produkt se transformuje tím, že se vytvoří.

Pokud proces ovlivňuje objekt, pak vstupní stav – stav ovlivněného objektu před výskytem procesu – se liší od výstupního stavu – stavu ovlivněného objektu po výskytu procesu. V tomto případě je stejný objekt sice v sadě objektů před procesem i v sadě objektů po procesu, ale nachází se v různých stavech. To je znázorněno na obr. 10.3, kde je **předem testovaný produkt** v sadě objektů před procesem, zatímco **testovaný produkt** je v sadě objektů po procesu.



Obr. 10.3 **Předem testovaný produkt** je v sadě objektů před zpracováním, zatímco **testovaný produkt** je v sadě objektů po zpracování

10.4.5 Sada zapojených objektů

Sada zapojených objektů je definována následovně.

Sada zapojených objektů procesu P , $Inv(P)$, je sjednocením sady objektů před zpracováním P a sady objektů po zpracování.

V sadové notaci: $Inv(P) = Pre(P) \cup Post(P)$.

V příkladech na obr. 10.1, $Inv(\text{Flight}) = \{\text{Runway, Pilot, Airplane}\}$ a $Inv(\text{Manufacturing}) = \{\text{Operátor, Stroj, Model, Surovina, Produkt}\}$.

10.5 Princip jedinečnosti procedurálního propojení OPM

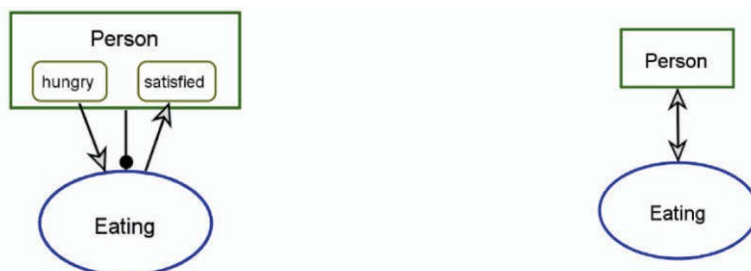
Podle definice procesu proces transformuje alespoň jeden objekt, takže v kompletním modelu OPM musí být proces propojen s alespoň jedním objektem nebo s některým z jeho stavů prostřednictvím transformačního propojení, a to buď přímo, nebo nepřímo. Proces a objekt mohou být propojeny pouze prostřednictvím procedurálního propojení, s výjimkou výstavy-charakterizace, která je strukturálním propojením. Jakékoli procedurální propojení, s výjimkou propojení vyvolání a výjimky, propojuje proces s objektem.

Objekt má určitou roli ve vztahu k procesu. Může to být agent, nástroj nebo transformovaný objekt. Proto objekt nebo stav objektu a proces nemohou být spojeny více než jedním procedurálním propojením. To je důvodem následujícího principu jedinečnosti procedurálního propojení OPM.

Princip jedinečnosti procedurálního spojení OPM

Na jakékoli úrovni detailu může být objekt a proces propojeny nejvýše jedním procedurálním propojením, které jednoznačně určuje roli objektu ve vztahu k procesu.

Důvodem pro omezení tohoto principu na danou úroveň abstrakce je to, že na různých úrovních abstrakce může být objekt modelován odlišně. Role objektu se může měnit s úrovní detailu. Jedinečnost procedurálního propojení vede modeláře k zachování sémanticky nejvýznamnějšího modelu na dané úrovni detailu.



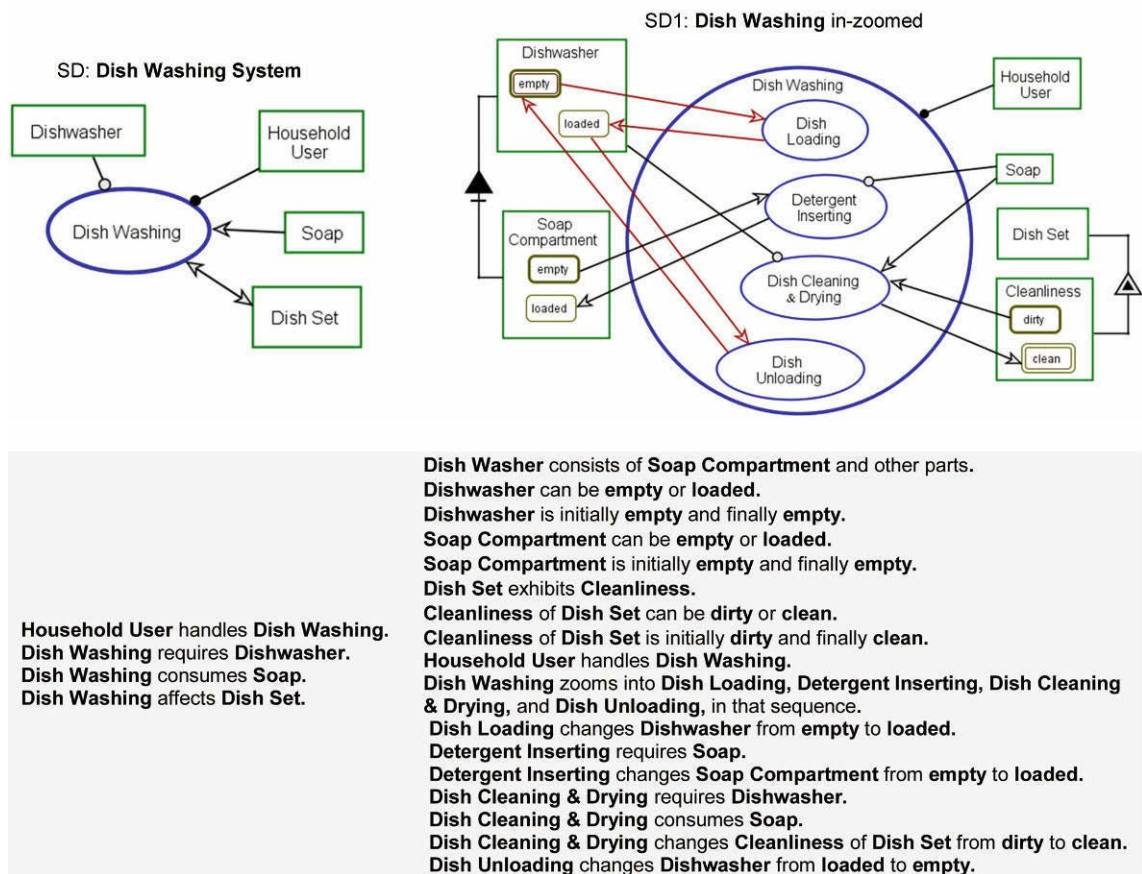
Person can be **hungry** or **satisfied**.
Person handles **Eating**.
Eating affects **Person**.
Eating changes **Person** from **hungry** to **satisfied**.

Eating affects **Person**.

Obr. 10.4 Demonstrace principu jedinečnosti procedurálního propojení OPM Vlevo: Vyjádření **osoby** jako aktéra i subjektu ovlivněného **jídlem** je možné pomoci vyjádření stavu. Vpravo: Když jsou stavy potlačeny, zůstává pouze propojení účinku.

Například v OPD vlevo na [obr. 10.4](#), když **osoba** vykonává činnost „**jíst**“, je **osoba** jak činitelem, protože **osoba** vykonává činnost „**jíst**“, tak i subjektem tohoto procesu, protože činnost „**jíst**“ mění **stav osoby** z **hladové** na **nasycenou**. To je možné, protože jsou vyjádřeny stavy „**hladová**“ a „**nasycená**“ **osoby**. Když jsou stavy potlačeny (vpravo), nemůžeme mít mezi **osobou** a **jídlem** jak spojení činitel, tak spojení účinek, takže musíme učinit volbu. Jak formálně definujeme a podrobněji vysvětlujeme v oddíle [21.13](#), volba spojení je založena na přednosti procedurálních spojení. Jelikož transformační (v našem případě efektové) spojení je sémanticky silnější než umožňující spojení (v našem případě agentové), převládá efektové spojení. Stále můžeme použít obě spojení, pokud se zaměříme na **Jídlo** a odhalíme jeho tři dílčí procesy: **Výběr jídla**, **Polykání jídla** a **Trávení jídla**. Pouze poslední dílčí proces ovlivňuje **osobu**, takže nyní může být **osoba** spojena agentovým spojem s **výběrem jídla** a **polykáním jídla** a efektovým spojem s **trávením jídla**. Při oddálení od **jídla** a potlačení stavů **osoby** budou **osoba** a **jídlo** opět spojeny efektovým spojem, protože celkově se **stav osoby** změnil, v souladu s prioritou spojů.

Dalším příkladem je **nákladní automobil**, který je zjevně nástrojem pro **přepravu**. **Přeprava** se zaměřuje na **nakládku**, **přepravu** a **vykládku**. **Nakládka** mění **nákladní automobil** z **nenaloženého** na **naložený**, takže nákladní automobil je zjevně ovlivněn. Po skončení **přepravy** však **vykládka** mění **nákladní automobil** zpět z **naloženého** na **nenaložený**, takže celkově lze **nákladní automobil** z hlediska přepravy považovat za neovlivněný a lze jej tedy modelovat jako nástroj **přepravy**, nikoli jako subjekt, na který má přeprava vliv.



Obr. 10.5 Role abstrakce s rozděleními stavovými transformačními vazbami

Objekt může mít roli nástroje v abstraktním OPD a transformovaného objektu v jiném potomkovi, podrobnějším a konkrétnějším OPD. V abstraktním OPD se zdá, že proces nemá na objekt žádný vliv, protože počáteční stav objektu je stejný jako jeho konečný stav. Proto je v abstraktním OPD objekt nástrojem, jak naznačuje propojení nástroje. V potomku, konkrétnějším OPD, však stejný proces mění stav tohoto objektu z počátečního stavu a poté zpět do počátečního stavu.

Jako poslední příklad, na obr. 10.5, levý OPD (SD: **Systém mytí nádobí**), objekt **Myčka nádobí** je nástrojem pro proces **mytí nádobí**, protože v tomto rozsahu abstrakce není viditelná žádná změna stavu **myčky nádobí**. V podřízeném OPD (SD1: **Dish Washing in-zoomed**) se **Dish Washing** přiblíží na **Loading** (naložení špinavého nádobí), **Cleaning** (čištění, které změní nádobí ze špinavého na čisté) a **Unloading** (vyložení čistého nádobí). **Nakládání** mění stav **myčky nádobí** z **prázdné** na **naloženou**, zatímco **vykládání** ji mění zpět z **naložené** na **prázdnou**, takže **prázdná** je jak počáteční, tak konečný stav. Zatímco **myčka nádobí** je nástroj v SD, systémovém diagramu, v potomském, podrobnějším OPD je **myčka nádobí**

ovlivněným prvkem – stane se **naplněnou** a poté opět **prázdnou**. Jediným viditelným účinkem v systémovém diagramu je účinek na **sadu nádobí**.

10.6 Test procesu

Jak již bylo uvedeno, zatímco základním principem OPM je rozlišení mezi objekty a procesy, někdy je obtížné odlišit objekt od procesu, zejména pokud jsou oba podstatná jména. Problém rozlišení mezi objektem a procesem lze jednoduše vyjádřit takto:

Jak můžeme u daného podstatného jména rozlišit, zda se jedná o objekt nebo proces?

Procesní test, specifikovaný v této sekci, je formální postup pro řešení problému rozlišení mezi objektem a procesem. Umožňuje identifikovat podstatná jména, která jsou spíše procesy než objekty, což je předpoklad pro úspěšnou analýzu a návrh systému.

Ve výchozím nastavení je podstatné jméno objektem. Aby bylo podstatné jméno procesem, musí splňovat všechna tři následující

kritéria testu procesu: (1) *transformace objektu*, (2) *časová asociace* a (3) *asociace slovesa*.

Pokud výsledek stále není jasný, je samozřejmě nejlepší použít zdravý rozum.

10.6.1 Kritérium transformace objektu

Kritérium testu transformace *objektu* stanoví, že proces musí transformovat (spotřebovat, vytvořit nebo změnit stav) alespoň jeden z objektů v dané sadě objektů.

Kritérium transformace objektu je splněno, pokud dané podstatné jméno transformuje alespoň jeden z objektů v dané sadě objektů.

Členství transformovaného objektu B v P se určuje následovně.

- Pokud P *spotřebovává* B , pak $B \in \text{Pre}(P)$: B je pouze v sadě objektů předzpracování P .
- Pokud P *vytváří* (*generuje*) B , pak $B \in \text{Post}(P)$: B je pouze v sadě objektů postprocesu P .
- Pokud P *ovlivňuje* (*mění stav*) B , pak B je členem $\text{Inv}(P)$. $\text{Inv}(P)$: B je v sadě zapojených objektů, tj. jak v sadě objektů před zpracováním, tak v sadě objektů po zpracování.

Enablers (agenti nebo nástroje) jsou také členy $\text{Inv}(P)$, protože jejich přítomnost je vyžadována po celou dobu trvání procesu.

V návaznosti na předchozí příklady proces **Let** transformuje **Letadlo** (změnou jeho atributu **Poloha** z **místa odletu** na **místo přiletu**). Proto $\text{Letadlo} \in \text{Inv}(\text{Let})$. **Výroba** transformuje dva objekty: spotřebovává **Suroviny** a vytváří **Produkt**, proto $\text{Suroviny} \in \text{Pre}(\text{Výroba})$, zatímco

$\text{Product} \in \text{Post}(\text{Manufacturing})$. Nakonec $\text{Machine} \in \text{Inv}(\text{Manufacturing})$, protože **Machine** je nástroj pro výrobu.

10.6.2 Kritérium časové asociace

Kritérium *asociace s časovým procesem* vyžaduje, aby dané podstatné jméno představovalo nějakou událost, výskyt, akci, postup, rutinu, provedení, operaci nebo činnost, která zabírá určitý čas na časové ose.

Kritérium časové asociace je splněno, pokud lze dané podstatné jméno považovat za událost, která se odehrává v čase.

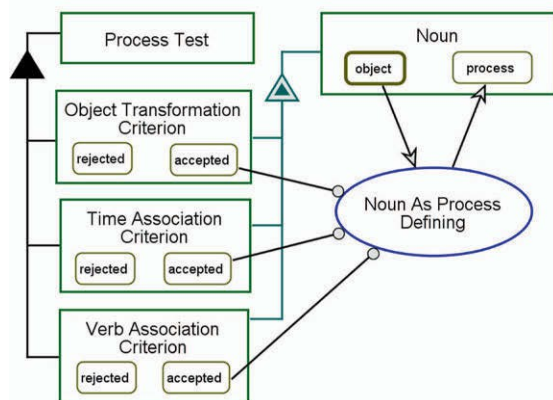
V našem příkladu začínají jak **let**, tak **výroba** v určitém časovém bodě a trvají určitou dobu. Čas i trvání jsou velmi relevantními vlastnostmi těchto dvou podstatných jmen.

10.6.3 Kritérium asociace se slovesem

Kritérium *asociace se slovesem* vyžaduje, aby byl proces asociován se slovesem.

Kritérium asociace se slovesem je splněno, pokud lze dané podstatné jméno odvodit od slovesa, má s ním společný kořen nebo má synonymum, které je slovesem.

Létání je sloveso spojené s **pojmem let**. Věta „Letadlo letí“ je krátkým způsobem, jak vyjádřit skutečnost, že **letadlo** se nachází v procesu **letu**. Podobně sloveso vyrábět (produkovat, vytvářet, generovat) je slovesem asociovaným s **výrobou**. Věta „Operátor vyrábí produkt ze surovin pomocí stroje a modelu“ je přirozeným jazykovým zkrácením odstavce OPL vpravo na [obr. 10.1](#).



Noun can be **process** or **object**.

Noun exhibits **Object Transformation Criterion**, **Time Association Criterion**, and **Verb Association Criterion**. The state **object** is initial.

Process Test consists of **Object Transformation Criterion**, **Time Association Criterion**, and **Verb Association Criterion**.

Object Transformation Criterion of **Noun** can be **rejected** or **accepted**.

Time Association Criterion of **Noun** can be **rejected** or **accepted**.

Verb Association Criterion of **Noun** can be **rejected** or **accepted**.

Noun As Process Defining requires **accepted Verb Association Criterion** of **Noun**, **accepted Time Association Criterion** of **Noun**, and **accepted Object Transformation Criterion** of **Noun**.

Noun As Process Defining changes **Noun** from **object** to **process**.

Obr. 10.6 Model OPM systému **testování procesů**

Zde se spoléháme na sloveso – syntaktickou konstrukci, ale není nutné, aby sloveso bylo syntakticky ze stejného kořene jako název procesu; může to být synonymum, pokud je sémantika stejná. Například „**Marrying**“ (**svatba**) je proces, který je spojen se slovesem „to marry“ (vdát se). „To wed“ (vdát se) je také právní sloveso, i když se používá méně často. Alternativně bychom mohli použít „**Wedding**“ (**svatba**), aby to odpovídalo slovesu „wed“. Mnoho objektů, jako například

Apple a **Airplane**, nejsou spojeny s žádným slovesem, takže nesplňují toto kritérium procesu. Je snadné ověřit, že **Apple** ani **Airplane** nesplňují ani ostatní kritéria testu procesu. Existují hraniční případy, jak je popsáno v oddíle 10.10 s příklady.

10.6.4 OPM model systému testování procesu

Obrázek 10.6 je model OPM systému testování procesů.

Dané **podstatné jméno** je zpočátku definováno jako **objekt**. **Procesní test** se skládá ze tří kritérií: **kritéria transformace objektu**, **kritéria časové asociace** a **kritéria slovesné asociace**, z nichž každé může být **přijato** nebo **odmítnuto**. Tři nástrojové odkazy ze tří **přijatých** stavů těchto tří kritérií naznačují, že pouze v případě, že jsou přijata všechna tři kritéria, je proces **definující podstatné jméno jako** proces aktivován, čímž se **podstatné jméno** změní z **objektu** na **proces**. Samo-vysvětlující odstavec OPL tohoto systému je také zaznamenán na [obr. 10.6](#).

10.7 Pojmenování prvků OPM

Výběr vhodných názvů pro objekty, procesy a stavy OPM je velmi důležitý, protože názvy ovlivňují, jak snadno a jak dobře je náš model sdělován a chápán cílovým publikem. Konvence pojmenování procesů a objektů pomáhají lidem je odlišit. Navíc, protože tyto modelářem definované názvy jsou také zabudovány do automaticky generovaných vět OPL, budou tyto věty dávat smysl pouze do té míry, do jaké jsou názvy entit v nich smysluplné a vedou ke správně formulovaným větám OPL. Například v odstavci OPL výše předpokládáme, že jsme tento proces nazvali jednoduše **Testování procesu**. To by vedlo k následujícím větám OPL:

Testování procesu vyžaduje **přijaté kritérium asociace slovesa podstatného jména**, **přijaté kritérium asociace času podstatného jména** a **přijaté kritérium transformace objektu podstatného jména**. **Testování procesu** mění **podstatné jméno z objektu na proces**.

Po změně názvu procesu jsou vytvořeny následující, přesnější věty OPL.

Podstatné jméno jako proces Definování vyžaduje **přijaté kritérium asociace slovesa podstatného jména**, **přijaté kritérium asociace času podstatného jména** a **přijaté kritérium transformace objektu podstatného jména**.

Podstatné jméno jako proces definující mění **podstatné jméno z objektu na proces**.

10.7.1 Velká písmena, tučné písmo, fráze a pojmenování věcí

Konvence OPM pro velká písmena stanoví, že první písmeno každého slova v názvu věci (objektu nebo procesu) se píše velkým písmenem, zatímco stavy se píší malými písmeny (bez velkých písmen). **Skupina možných zraněných osob ve vozidle** tedy označuje objekt **Skupina osob ve vozidle** v **možném zraněném stavu**.



Obr. 10.7 Model OPM krabíčky s jedním (vlevo) a šesti (vpravo) tužkami

Značky označených strukturálních vztahů také nejsou psány velkými písmeny, jako v OPL větě „**Krabice obsahuje** tužku“, která je textovou modalitou OPD vlevo na [obr. 10.7](#). Značka obsahuje podél šipky od **Krabice** k **Tužce** je psána malými písmeny.

***Fráze** je soubor jednoho nebo více slov, která netvoří větu.*

Pojmenování objektů je jednoduché – jedná se o podstatné jméno psané velkým písmenem. Názvy objektů mohou být fráze s více než jedním slovem, jako například **Apple Cake** (jablečný koláč) nebo **Insurance Claim** (pojistná událost).

10.7.2 Princip OPM pro singulární názvy

Důležitým principem OPM, který je třeba dodržovat při pojmenovávání objektu nebo procesu, je *princip jednotného názvu OPM*:

Princip jednotného názvu OPM

Název věci OPM musí být v jednotném čísle. Množné číslo musí být převedeno do jednotného čísla přidáním slova „Set“ (sada) pro neživé věci nebo „Group“ (skupina) pro lidi.

Pro definování tohoto principu existují dva důvody. Za prvé, automatizovaný nástroj se postará o převod jednotného čísla na množné číslo podle potřeby. Například v OPD vpravo na [obr. 10.7](#), když se přidá omezení účasti (definované později) „6“, věta OPL nyní zní: **Box obsahuje 6 tužek**. Za druhé, chceme být schopni specifikovat části, atributy nebo specializace věci v jednotném čísle.

Co tedy máme dělat, pokud chceme modelovat více než jednu instanci? Převedeme množné číslo objektu v modelu OPM na jednotné číslo přidáním slova „Set“ (sada) pro neživé věci nebo „Group“ (skupina) pro lidi. Objekt „Ingredients“ (například ingredience do dortu) se tak stane „**Ingredient Set**“ (sada ingrediencí), proces „Modifications“ (úpravy) se stane „**Modification Set**“ (sada úprav) a „Customers“ (zákazníci) se stane „**Customer Group**“ (skupina zákazníků).

10.7.3 Pojmenování procesů

Pokud to v angličtině nedává smysl, konvence pojmenování procesů OPM spočívá v pojmenování procesu tak, že jeho poslední slovo je gerundium, tj. kořen slovesa následovaný příponou „ing“, jako v případě **Igniting**. Tento *způsob pojmenování procesů* nazýváme *gerundovým*. Pokud existuje několik možností, jako například **Construction** vs. **Constructing**, je vhodnější použít druhou možnost, pokud odborníci v dané oblasti neuvedou, že v dané oblasti se běžně používá a rozumí se forma bez gerundia.

Tato konvence pojmenování objasňuje dynamickou povahu procesu jako dynamické věci, věci, která *se odehrává* v časové dimenzi, spíše než statické věci, která *existuje* beze změny. Pro zvýšení srozumitelnosti a zřejmosti funkce procesu může být před gerundiem uvedeno primární objekt, který proces transformuje, jako v případě **Engine Igniting**. Název objektu, který může předcházet gerundiu, kvalifikuje proces a činí jej specializací původního procesu. Například **Wall Painting** (malování stěn) a **Car Painting** (lakování aut) jsou dva různé (ale podobné) procesy, které specializují **Painting** (malování). Oba transformují malovaný objekt změnou hodnoty atributu barvy ovlivněného objektu (operandu) – malovaného objektu. Jelikož jsou však malované objekty odlišné, liší se nástroje a techniky každého druhu malování.

Název procesu v příkladu v části I této knihy, **Automatická reakce na nehodu**, by mohl být jednoduše **Reakce**, ale to by mohlo působit příliš obecně, protože nespecifikuje, na co je reakce zaměřena. Ani samotný název **Reakce na nehodu** není zcela dostačující, protože by se dal provést i bez automatizovaného systému. Také se vyhýbáme pojmenování tohoto procesu „**Reakce**“, protože tento název nesleduje způsob pojmenování procesů gerundiem a lze jej oprávněně považovat za objekt – výsledek procesu reakce.

Doporučený způsob pojmenování procesů pomocí gerundia má několik verzí s rostoucí délkou a informačním obsahem:

1. Verze s *transformujícím* slovesem: název procesu (syntakticky gerundová forma slovesa, tj. + *ing*), jako v **Making** nebo **Responding**.
2. Verze *transformující objekt*: spojení objektu OPM (syntakticky podstatné jméno) s názvem procesu (syntakticky gerundium slovesa), jako v **Cake Making (výroba dortů)** nebo **Crash Responding (reakce na havárii)**. Toto je ve většině případů doporučený způsob pojmenování.
3. *Kvalifikovaná* verze *transformace*: spojení hodnoty atributu (syntakticky přídavné jméno) s názvem procesu, jako v **Quick Making (rychlé pečení)** nebo **Automated Responding (automatická reakce)**.
4. *Kvalifikovaná* verze *transformace objektu*: spojení hodnoty atributu s objektem a gerundiem. Hodnota atributu může kvalifikovat proces, jako v **Quick Cake Making (Rychlá výroba dortů)** nebo **Automatic Crash Responding (Automatická reakce na havárii)**, nebo může kvalifikovat objekt, jako v **Sweet Cake Making (Výroba sladkých dortů)** nebo **Fatal Crash Responding (Reakce na smrtelnou havárii)**.

Druhou možností pojmenování procesu, kterou modeláři často používají, je *imperativní režim pojmenování procesu*, jako v „respond“ (reagovat) nebo konkrétněji „respond to crash“ (reagovat na havárii) nebo „automatically respond to crash“ (automaticky reagovat na havárii). OPM tento režim nedoporučuje, protože je méně kompaktní a méně elegantní a věty OPL vytvořené pomocí tohoto režimu v aktuální verzi OPM 19450 jsou neohrabané. Modelovací jazyky obvykle takové konvence pojmenování nepředepisují. Modeláři si proto nejsou vědomi nuancí, jako je rozdíl mezi gerundiem a imperativním režimem pojmenování procesu. Funkční analýza doporučuje pojmenovávat funkce imperativně: „Start Engine“ (Spustit motor). „Launch Missile“ (Odpalit raketu), „Turn Left“ (Zahnout doleva), ale nezdá se, že by to byl předem promyšlený a povinný způsob, pouze krátký a někdy pohodlný způsob vyjádření. V důsledku toho mnoho modelářů používá gerundium i imperativ jako název procesu zaměnitelně nebo smíšeně, což činí model méně soudržným a zbytečně náročnějším na kognitivní schopnosti.

10.8 Definice věci

Viděli jsme, že objekty a procesy jsou dva typy úzce propojených a komplementárních *věcí*. Objekty nelze transformovat (generovat, ovlivňovat nebo eliminovat) bez procesů, zatímco procesy nemají žádný význam bez objektů, které transformují, a často také bez objektů, které umožňují jejich výskyt. Rozsah tohoto propojení je tak intenzivní, že pokud chceme být schopni analyzovat a navrhovat systémy v jakékoli oblasti co nejintuitivněji a nejpřirozeněji, musíme uvažovat o objektech a procesech současně. Objekty *existují* jako relativně trvalé, statické věci, zatímco procesy *se vyskytují* jako přechodné, dynamické věci.

Míra propojení objektů a procesů je ještě větší; musíme být schopni specifikovat, v jakém stavu se objekt nacházel předtím, než na něj proces působil, které objekty byly spotřebovány a které byly generovány. Zároveň musíme být schopni ukázat, jak části, vlastnosti a specializace (diskutované později) těchto objektů hrají roli v podprocesech procesu vyšší úrovně.

Jak uvidíme, objekty a procesy mají mnoho společného, pokud jde o specifikaci prostřednictvím strukturálních vztahů, jako je agregace, generalizace a charakterizace. Potřeba hovořit o těchto dvou pojmech obecně, aniž bychom opakovali „objekt nebo proces“ znovu a znovu, vyžaduje zavedení ještě abstraktnějšího termínu. Nazýváme jej jednoduše „věc“.

Věc je zobecněním objektu a procesu.

Pojem „věc“ nám umožňuje uvažovat a vyjadřovat se v pojmech této abstrakce a odkazovat se na ni, aniž bychom museli opakovat slova „objekt nebo proces“. Na základě ontologie Bungeho (1987, 1989) používají Wand a Weber (1989, 1993) termín *věc* jako synonymum pro to, co označujeme jako objekt. Jejich první premisou je, že *svět je tvořen věcmi, které mají vlastnosti*. Podle této definice se *věc* jeví jako synonymum *objektu*. Během posledních dvou desetiletí se však termín *objekt* hluboce zakořenil, alespoň v komunitě softwarových inženýrů. V SysML a UML byl pojem objekt nahrazen pojmy *blok* a *třída*. Zajímavé je, že vznik pojmu „internet věcí“ (IoT; Weber a Weber 2010) je v souladu s pojmem *věc* jako zobecněním objektu a procesu, protože IoT se týká procesů probíhajících mezi fyzicky propojenými objekty.

10.9 Vlastnosti věcí OPM

Vlastnost je atribut na úrovni metamodelu. Vlastnost lze považovat za meta-atribut – atribut prvku v metamodelu OPM.

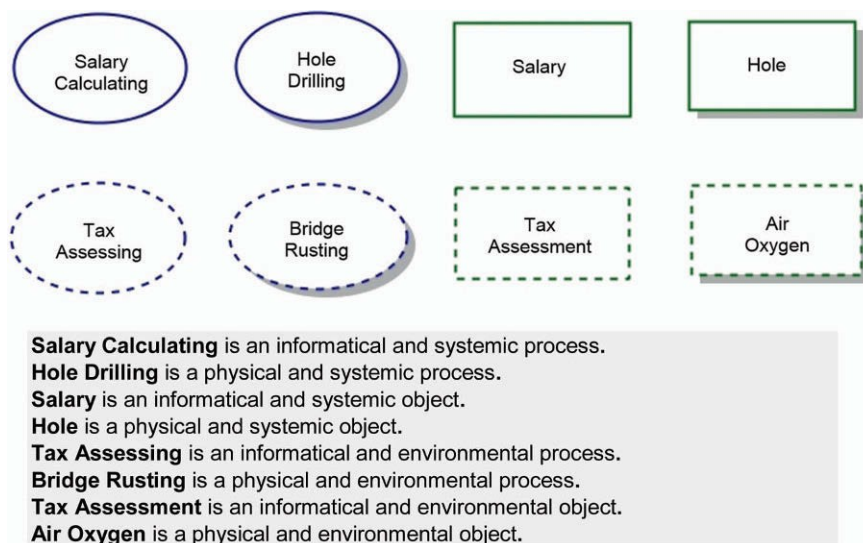
Vlastnost je atributem prvku modelu OPM.

Na rozdíl od „běžného“ atributu, jehož hodnoty se mohou během provádění modelu OPM měnit, je hodnota vlastnosti jakéhokoli prvku v modelu OPM pevně daná. Příklad uvidíme na konci této části. Všechny věci OPM mají následující tři vlastnosti:

- **Vytrvalost**, která se týká trvalosti věci a označuje, zda je věc **statická** (trvalá), tj. **objekt**, nebo **dynamická** (přechodná), tj. **proces**. Existují příklady hraničních statických, trvalých procesů a dynamických, přechodných objektů, jak bude popsáno později v této kapitole. Na základě hodnoty **vytrvalosti** rozlišuje tato vlastnost **věci** mezi **objektem** a **procesem**. Na úrovni modelu nazýváme takové atributy rozlišujícími atributy, jak je popsáno v pozdější kapitole.
- **Podstata**, která se týká povahy věci a označuje, zda je věc **fyzická** nebo **informační**.
- **Příslušnost**, která se týká rozsahu věci a označuje, zda je věc **systémová**, tj. součástí systému, nebo **environmentální**, tj. součástí prostředí systému.

Graficky, jak je znázorněno na [obr. 10.8](#), stínované efekty označují fyzické věci OPM a přerušované čáry označují environmentální věci OPM. Může se vyskytnout všech osm kombinací vlastností **Perseverance-Essence-Affiliation** věci OPM znázorněných na [obr. 10.8](#). Spodní část [obr. 10.8](#) vyjadřuje zleva doprava a shora dolů věty OPL odpovídající grafickým prvkům.

Zaznamenali jsme, že hodnota vlastnosti jakéhokoli prvku v modelu OPM je pevně daná. Pokud se podíváme na příklad **Perseverance**, vlastnost OPM **Thing**, pokud je hodnota určitého **Thing** v modelu OPM nastavena jako **statická** (tj. **Thing** je **Object**), pak je tato hodnota pevně daná a **Object** se nemůže stát **Process**.



Obr. 10.8 Kombinace generických atributů OPM thing

10.9.1 Výchozí hodnoty generických vlastností Thing

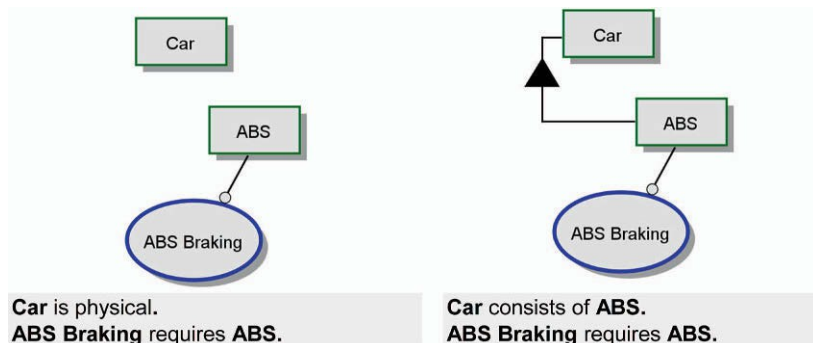
Vlastnost „**Příslušnost**“ věci je ve výchozím nastavení **systémová**. S ohledem na **podstatu** poznamenáváme, že většina věcí v netriviálních systémech má tendenci mít stejnou hodnotu vlastnosti: buď je většina věcí v systému **fyzická**, nebo je většina z nich **informatická**. Například systémy zpracování dat jsou informatické, i když mají fyzické komponenty. Dopravní systémy, jako je železniční systém nebo letecký systém, jsou fyzické, i když mají informatické komponenty.

***Primární podstatou** systému je hodnota podstaty většiny věcí v systému.*

Výchozí hodnota podstaty věci je primární podstatou systému. Motivací pro definování primární podstaty, založenou na zkušenostech, je ušetřit modeláři nutnost označovat drtivou většinu věcí v systému jako informační nebo fyzické. Podpůrný nástroj by proto měl modeláři poskytnout možnost specifikovat primární podstatu systému jako prostředek ke snížení množství věcí, pro které modelář musí specifikovat jejich podstatu.

Odstavec OPL odpovídající OPD by neměl obsahovat větu OPL označující hodnotu podstaty nebo příslušnosti věci, pokud je to výchozí hodnota, ledaže by věc byla izolovaná – během procesu modelování ještě nebyla propojena s žádnou jinou věcí. Důvodem je potřeba vyhnout se porušení principu OPM pro grafiku a text. Předpokládáme, že výchozí podstatou OPD na [obr. 10.9](#) je fyzická podstata. Po nakreslení fyzického objektu **Auto** a před jeho propojením s čímkoli se objeví věta OPL „Auto je fyzické“, jak je znázorněno v OPD vlevo, jinak by v OPD byla zobrazena věc (**Auto**), která není zmíněna v OPL, což by porušovalo princip OPM pro grafiku a text. Jakmile se však

jakmile se izolovaná věc propojí s jinou věcí, jak je znázorněno v OPD vpravo, věta OPL věnovaná specifikaci výchozí podstaty nebo příslušnosti věci bude odstraněna.



Obr. 10.9 Primární podstata protiblokovacího brzdového systému (ABS) automobilu je fyzikální, proto jakmile je **automobil** připojen k **systému ABS**, první věta je odstraněna z věty OPL

10.10 Hraniční případy věcí

Zatímco objekty jsou trvalé a procesy jsou přechodné, existují hraniční případy procesů zachovávajících stav (trvalých) a přechodných objektů. Ty jsou popsány v této části.

10.10.1 Procesy zachovávající stav

Proces jsme definovali jako věc, která transformuje objekt. Existují případy, kdy změnu stavu objektu způsobuje spíše *absence* procesu než jeho výskyt. Jedním z příkladů je podpora: Jakýkoli objekt na planetě Zemi (nebo na jakékoli jiné planetě) je udržován ve vertikální poloze procesem **podpory**, který zabráňuje jeho volnému pádu. Existuje celá řada takových procesů *zachovávajících stav*, které mají statický význam, protože působí tak, aby udržovaly stav objektu, nikoli aby jej měnily.

***Proces zachovávající stav** je proces, který působí tak, aby udržoval stabilní stav nebo status quo objektu, spíše než aby jej měnil.*

Nejvýraznějším příkladem je proces **existence**, který popisuje situaci, kdy objekt „existuje“, aniž by docházelo k jakékoli změně tohoto objektu. U biologických objektů existence zahrnuje udržování nezbytných životních procesů, takže rozhodně nejsou statické. Ne biologické systémy, jako je sluneční soustava nebo globální systém řízení letového provozu, také existují, ale neustále se mění.

Mezi členy této rodiny *procesů zachovávajících stav* patří procesy jako **podporovat, držet, udržovat, zachovávat, zůstávat, čekat, prodlužovat, odkládat, zabírat, přetrvávat, zahrnovat, obsahovat, pokračovat, uzavírat, upevňovat, spojovat, odkládat, táhnout, ukládat, vlastnit, omezovat, přitahovat** a **zůstávat**. Sémantika těchto sloves spíše než vyvolání skutečné změny znamená ponechání aktuálního stavu objektu tak, jak je, ve svém status quo, po určitou dobu.

Každý z těchto procesů lze považovat za proces *bránící změně* – proces, který působí proti nějaké „síle“, která by jinak změnila operand – objekt, na který se operace vztahuje. Například **podepření notebooku** lze přeformulovat jako **zabránění pádu**, **držení mince** lze přeformulovat jako **zabránění její ztrátě** a **držení rukojmí** lze přeformulovat jako **zabránění útěku** tohoto **rukojmí**. Vzhledem k jejich povaze zachovávající stav lze tyto „pseudoproceny“ spíše modelovat pomocí označených strukturálních vztahů mezi dvěma objekty. Tuto otázku diskutujeme v kontextu strukturálních vztahů.

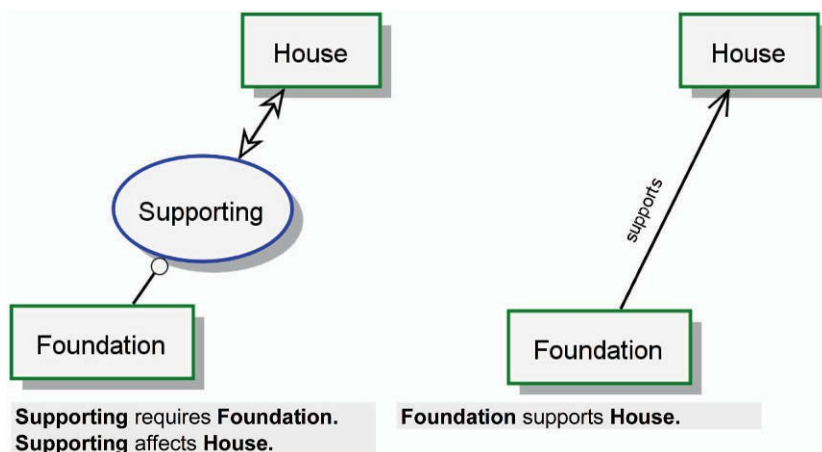
10.10.2 Jak modelovat procesy zachovávající stav pomocí označených strukturálních vazeb

Mnoho sloves zachovávajících stav lze považovat za působící proti určité „síle“, která by jinak změnila nějaký objekt. Například **podstavec** podpírající **sochu** působí proti gravitaci, takže můžeme **podpírat** považovat za proces „zabraňující pádu“, bez kterého by se stav **sochy** změnil ze **stabilního** na **padlý**. Proces **podpory** začíná, jakmile je **socha** umístěna, a pokračuje, dokud se něco v systému nezmění, např. **podstavec** neprojde procesem **rozbití**, čímž se jeho stav změní z **neporušeného** na **rozbitý**. Modernějším příkladem je **autopilot**, systém navržený k udržování a stabilizaci **letadla** na jeho kurzu, který působí proti vztlaku, odporu, gravitaci a odstředivé síle. Jakmile proces udržování stavu skončí, stav se změní, takže je třeba tento proces zachytit jako opakující se – ať už prostřednictvím sebevyvolání, jak je uvedeno v oddíle 22.4.6, nebo řízené reakce na vnější spouštěč.

Statická povaha procesů zachovávajících stav je v rozporu s definicí procesu, která vyžaduje, aby *transformoval* nějaký objekt. V takových případech je často možné, a dokonce žádoucí, modelovat vztah mezi dvěma příslušnými objekty pomocí označeného strukturálního odkazu namísto procesu. Tento přístup k modelování trvalých procesů je znázorněn na [obr. 10.10](#), který ukazuje **Supporting** jako proces zachovávající stav. Na levé straně je dynamická verze modelu, ve které je **Supporting** explicitním procesem, prezentovaným s odpovídajícím odstavcem OPL. Na pravé straně je statická verze modelu, ve které označená strukturální relace **supports** vyjadřuje časově neměnný vztah mezi **Foundation** a **House**, což vede k odpovídajícímu kompaktnějšímu a výstižnějšímu odstavci OPL o jedné větě: **Foundation supports House**.

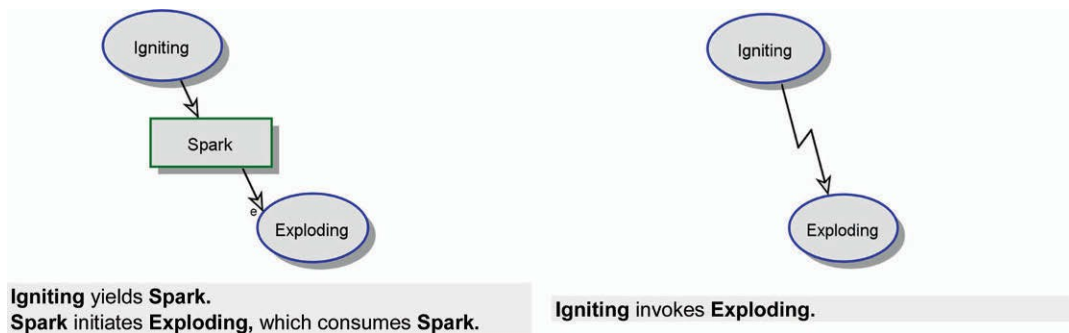
10.10.3 Přechodné objekty a jejich vyvolání Náhrada odkazu

Přechodné objekty jsou analogickými protějšky trvalých procesů. *Přechodný objekt* je fyzický nebo informatický objekt s krátkou životností. Příklady přechodných objektů jsou nestabilní materiály, jako je například dočasná sloučenina s krátkou životností v chemické reakci nebo atom v excitovaném stavu, který spontánně roztáhá do základního stavu emisí rentgenového záření a fluorescenčního záření. Dalším příkladem přechodného objektu je paket v telekomunikační síti. Takový paket může na své cestě krátce setrvat v nějakém směrovači a po přijetí cílovým uzlem nezanechat žádné stopy.



Obr. 10.10 Podpora jako proces zachovávající stav

V modelu OPM lze přechodný objekt, který je vytvořen procesem a okamžitě spotřebován dalším procesem, přeskočit pomocí *invokačního odkazu*, což je bleskovitý procedurální odkaz, který přímo spojuje oba procesy. **Obrázek 10.11** demonstruje pojmy přechodný objekt a odkaz na vyvolání. Na levé straně je model, ve kterém je **Spark** explicitním objektem vytvořeným **Igniting**. Přítomnost **Spark** je událostí, která iniciuje (spouští) **Exploding**, jak je označeno písmenem e vedle šipky směřující k **Exploding**. **Exploding** okamžitě spotřebuje **Spark**, takže **Spark** je přechodný a krátkodobý. Na pravé straně je alternativní, kompaktnější model, ve kterém je přechodný **Spark** potlačen odkazem na vyvolání. Sémantika odkazu na vyvolání spočívá v tom, že konec **Igniting** je událostí, která spouští **Exploding**. Odstavec OPL je v tomto případě také kompaktnější.



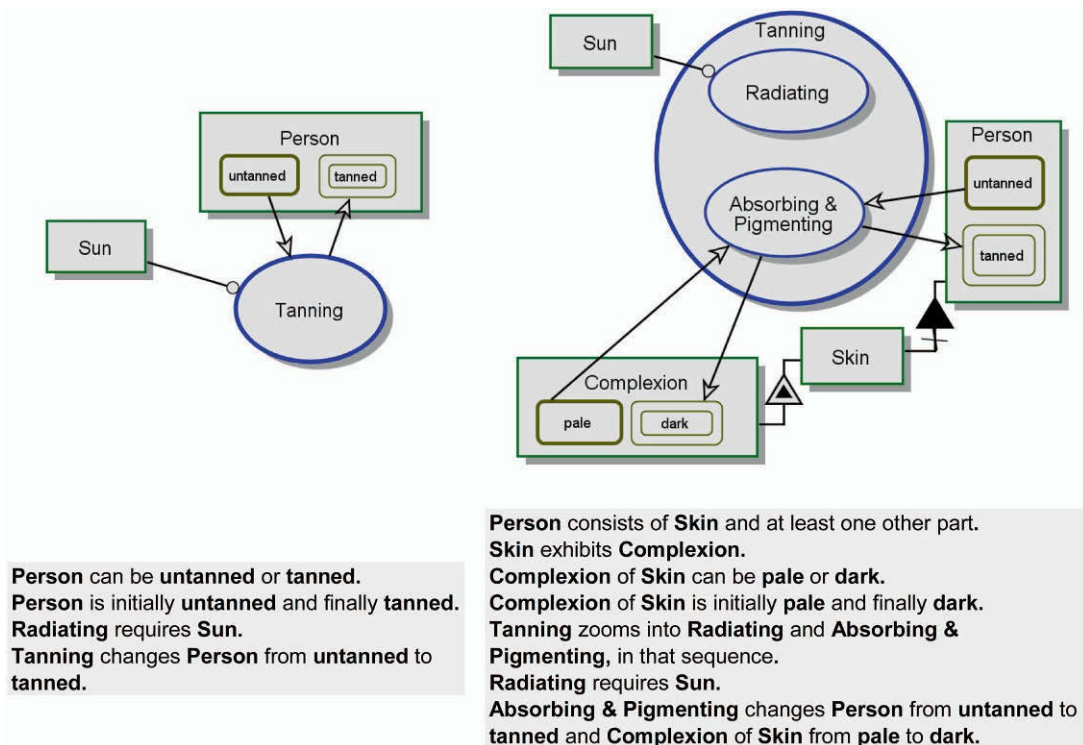
Obr. 10.11 Jiskra jako přechodný objekt (vlevo) a modelování bez něj pomocí odkazu na vyvolání (vpravo)

Při pohledu zpět na **obr. 10.10** a porovnání s **obr. 10.11** můžeme vidět vzorec: Použití volacího odkazu jako kratší verze modelování generování a okamžité spotřeby přechodného objektu je analogické s použitím značeného strukturálního odkazu jako kratší verze modelování trvalého procesu. Dalším příkladem je **Signaling** a přechodný objekt **Signal**.

10.11 Operátor, operand a transformace

Než uzavřeme tuto kapitolu o dynamice systémů, může být zajímavé porovnat ontologii OPM s definicemi Ashbyho (2001) týkajícími se operandu, operátoru a transformace:

Uvažujme jednoduchý příklad, ve kterém se pod vlivem slunečního záření bledá kůže změní na tmavou. Něco, „světlá kůže“, je ovlivněno faktorem, „slunečním zářením“, a změní se na tmavou kůži. To, co je ovlivněno, světlá kůže, bude nazýváno OPERANDEM, [způsobující] faktor bude nazýván OPERÁTOREM a to, na co se operand změnil, bude nazýváno TRANSFORMACÍ.



Obr. 10.12 Opalování na nejvyšší úrovni (vlevo) a přiblížený pohled (vpravo)

V ontologii OPM je **kůže** objektem, zatímco **tmavá** a **bledá** jsou stavy atributu objektu **kůže** nazývaného **pleť**. **Kůže** je jednou z částí **osoby**. **Opalování** je proces a **slunce** je nástroj, který umožňuje proces **opalování**, jehož účinkem je změna **pleti** z **bledé** na **tmavou**. Tato terminologie a model OPM na obr. 10.12 se zdají být intuitivnější a vhodnější pro nematematické systémy než ontologie operandů, operátorů a transformací. „Faktor slunečního svitu“ je poněkud problematický k popisu. Není jasné, zda se vztahuje k procesu svitu slunce, nebo k objektu, který agreguje fotony energie vyzařované sluncem, které kůže absorbuje.

V OPM bychom modelovali **vyzařování** jako první dílčí proces **opalování**. **Vyzařování vyžaduje** (tj. je umožněno nástrojem) **slunce**. **Vyzařování** zase produkuje objekt **sluneční energie**, který je absorbován **kůží** prostřednictvím druhého dílčího procesu, **absorpce a pigmentace**, který mění **barvu pleti** z **bledé** na **tmavou**. Stručně řečeno, operátor je proces (**opalování**). Operand je ovlivněná veličina ve stavu před procesem (**kůže** ve stavu **bledé pleti**), zatímco transformace je její stav po procesu (**kůže** ve stavu **tmavé pleti**).

10.12 Shrnutí

- **Vlastnost** je atribut metamodelu prvku OPM. Hodnota vlastnosti každého prvku v modelu OPM zůstává neměnná.
- Přístup OPM považuje procesy za „prvotřídní občany“ vedle objektů, nikoli za objekty nižšího řádu.
- **Objekt** je věc, která existuje nebo má potenciál fyzické nebo informační existence.
- Dvě instance fyzického objektu jsou identické, pokud a pouze pokud zabírají stejný prostor ve stejném čase.
- Z informatického hlediska jsou všechny fyzické kopie určitého informatického objektu stejné.
- **Transformace** je generování (konstrukce, vytvoření) nebo spotřeba (zničení, eliminace) nebo změna (účinek, přechod stavu) objektu.
- **Proces** je mentální konstrukt představující vzorec transformace objektu.
- V analýze „příčina a následek“ je **příčina** spouštěcí událostí, která se pokouší způsobit zahájení provádění procesu.
- **Účinek** v analýze „příčina a následek“ je transformace, kterou prochází jeden nebo více objektů spojených s prováděným procesem.
- Slovní druhy (podstatné jméno, sloveso, přídavné jméno, příslovce...) jsou *syntaktické* konstrukce, zatímco prvky OPM (objekt a proces) jsou *sémantické* konstrukce.
- **Sada objektů předprocesu** procesu P , $Pre(P)$, je sada objektů, které musí existovat, případně v určitých stavech, aby mohl P začít provádět, jakmile byl spuštěn.
- **Sada objektů po zpracování** procesu P , $Post(P)$, je sada jednoho nebo více objektů, které existují, případně v určitých stavech, po dokončení provádění P .
- **Sada spojených objektů** procesu P , $Inv(P)$, je sjednocením sady objektů před zpracováním P a sady objektů po zpracování: $Inv(P) = Pre(P) \cup Post(P)$.
- **Problém rozlišení objektu a procesu** je problémem určení, zda dané podstatné jméno je objektem nebo procesem.
- **Procesní test** je formální postup pro řešení problému rozlišení mezi objektem a procesem.
- Procesní test předpokládá, že podstatné jméno je ve výchozím nastavení objektem, takže aby se jednalo o proces, musí splňovat tři kritéria: (1) *transformace objektu*, (2) *časová asociace* a (3) *slovesná asociace*.

- **Kritérium transformace objektu** je splněno, pokud dané podstatné jméno transformuje alespoň jeden z objektů v dané sadě objektů.
- **Kritérium časové asociace** je splněno, pokud lze dané podstatné jméno považovat za něco, co se odehrává v čase.
- **Kritérium asociace se slovesem** je splněno, pokud dané podstatné jméno lze odvodit od slovesa, má s ním společný kořen nebo má synonymum, které je slovesem.
- **Konvence OPM pro velká písmena** stanoví, že první písmeno každého slova v názvu věci se píše velkým písmenem, zatímco stavy se píší malými písmeny.
- **Princip jednotného čísla názvu OPM** stanoví, že název věci OPM musí být v jednotném čísle.
- **Konvence pojmenování procesů OPM** spočívá v pojmenování procesu tak, že jeho poslední slovo je gerundium, pokud je to možné, přijatelné a dává smysl v nomenklatuře dané oblasti.
- **Věc** je zobecněním objektu a procesu.
- **Proces zachovávající stav** je proces, který udržuje stabilní stav status quo a může být potlačen nahrazením označeným strukturálním vztahem.
- **Přechodný objekt** je objekt s krátkou životností a lze jej potlačit nahrazením odkazem na vyvolání.

10.13 Problémy

1. Uveďte příklad vědeckého objevu a vysvětlete, jak jej lze považovat za reverzní inženýrství přírody.
2. Proč je nemožné se dotknout procesu, i když je fyzický?
3. Proč je proces v OPD, ke kterému není připojen žádný transformační odkaz, bezvýznamný?
4. Kdo jsou „hráči“ v analýze příčin a následků? Jaká je role každého z nich?
5. Uveďte příklad dvou vět, které vyjadřují stejnou skutečnost, ale mají odlišné slovní druhy.
6. Jaké jsou objekty a procesy v první větě výše? A ve druhé?
7. Sestavte model OPM systému popsaného v předchozí otázce.
8. Jaké jsou v modelu OPM testovacího systému procesu na [obr. 10.6](#) členové v sadě objektů předběžného zpracování, v sadě objektů následného zpracování a v sadě zapojených objektů?
9. Vyberte dvě položky z OPD na [obr. 10.3](#) a proveďte procesní test na každé z nich.
10. Jaký je preferovaný způsob modelování trvalých procesů?
11. Jaká je možná zkratka pro modelování přechodných objektů?
12. Modelujte následující specifikaci: Provoz spalovacího motoru závisí na přítomnosti objektů směsi vzduchu a benzínových par uvnitř objektu válce při správném tlaku a teplotě (atributy směsi). Spouštěcí událostí je okamžik, kdy jiskra (vytvořená předchozím časovaným procesem) zapálí směs. Výsledkem tohoto procesu je spotřeba benzínové směsi a zvýšení hodnoty kinetické energie pístu.

Kapitola 11

Jazyk objektově-procesní: Text

Mezi univerzálními modelovacími jazyky dominují grafické jazyky, jako je UML; textové modelovací jazyky nejsou tak populární, i když mají velký potenciál.

Mazanec a Macek (2012)

OPM je bimodální: využívá jak vizuální (grafickou) modalitu – OPD, tak verbální (textovou) modalitu – OPL. Textová reprezentace OPL modelu OPM má jak cíle orientované na člověka, tak cíle orientované na stroj. Tato kapitola je věnována představení OPL a diskusi o jeho výhodách.

11.1 OPL: Textová modalita

Abychom zvýšili expresivní sílu OPM, spojujeme s každým OPD soubor vět v Object-Process Language (OPL) jako textovou, přirozenou interpretaci grafického znázornění OPD.

Object-Process Language (OPL) je podmnožina angličtiny, která textově vyjadřuje model OPM, který sada OPD vyjadřuje graficky.

OPL je textová obdoba grafické specifikace systému OPM. Je extrahován z diagramového popisu v sadě OPD. Pomocí nástroje, jako je OPCAT, je OPL automaticky generovaným textovým popisem systému v podskupině přirozené angličtiny. Bez zvláštností a nadměrných tajemných detailů, které charakterizují programovací jazyky, jsou věty OPL srozumitelné i pro lidi bez technických nebo programovacích zkušeností.

Modelový fakt je vztah mezi dvěma nebo více věcmi v modelu OPM.

Každý modelový fakt je v modelu OPM vyjádřen ve dvou modalitách: v grafické modalitě v jednom nebo více OPD a v textové modalitě ve větě OPM pro každý grafický výraz tohoto modelového faktu.

Každý prvek OPD (věc nebo odkaz) má grafický symbol. *Konstrukce OPD* je syntakticky platná kombinace grafických symbolů OPM, která vyjadřuje *modelovou skutečnost*. Tato modelová skutečnost je ekvivalentně vyjádřena větou nebo částí věty v textu *Object-Process Language (OPL)*. To je shrnuto v následující sadě definic.

Prvek OPD je grafickým vyjádřením věci nebo odkazu.

Konstrukce OPD je soubor propojených prvků OPD.

Modelový fakt je graficky vyjádřen konstrukcí OPD a textově ekvivalentní větou nebo částí věty OPL.

11.2 Dvojitý účel OPL

OPL slouží dvěma cílům, orientovaným na dva směry: lidi a stroje.

11.2.1 Cíl OPL orientovaný na člověka

Cílem OPL orientovaným na člověka je převést sadu OPD tvořících model OPM do textu v přirozeném jazyce, který lze použít k vyjádření a sdělení výsledků analýzy a návrhu mezi různými zúčastněnými stranami zapojenými do systému, který se buduje. Mezi uživatele patří odborníci v dané oblasti a jejich vedoucí pracovníci na straně zákazníka systému, který se vyvíjí, jakož i architekti a modeláři na straně dodavatele téhož systému.

OPL umožňuje zapojit zainteresované strany na straně zákazníka, které často nemají technické znalosti, do zjišťování požadavků a počátečního koncepčního modelování systému, který se vyvíjí. Zapojení těchto zainteresovaných stran jako aktivních účastníků pomáhá zefektivnit požadavky, získat souhlas zainteresovaných stran a odhalit chyby brzy po jejich neúmyslném zavedení.

Obvykle tito zainteresovaní nemají znalosti programovacích jazyků a není realistické očekávat, že budou číst diagramy v jazyce koncepčního modelování, natož programový kód. Jsou zvyklí číst text (nebo prohlížet prezentace na vysoké úrovni) spíše než se zabývat diagramy, a proto oni (nebo jejich inženýři či právníci) pravděpodobně upřednostní čtení textu před zkoumáním a interpretací OPD. Pro ně slouží OPL k ověření a validaci požadavků, které jsou obvykle nejprve poskytnuty v textové podobě a poté modelovány v OPM. Tento model požadavků pomáhá identifikovat mezery a nesrovnalosti, takže požadavky mohou být vylepšeny a přijatelné pro obě strany – zákazníka i dodavatele nebo vývojáře.

Pro systémové architektky a modeláře na straně dodavatele je bimodální reprezentace modelu OPM zásadní pro získání okamžité zpětné vazby ke každé grafické editační operaci, což jim umožňuje odhalit chyby modelování ihned po jejich vzniku, než se rozšíří a začnou způsobovat škody, jejichž rozsah se exponenciálně zvyšuje s prodlevou při detekci chyby. Navíc mohou začínající uživatelé OPM rychle získat znalosti o sémantice grafické modalitě OPM tím, že si prohlédnou text a odpovídající grafiku současně.

Vzhledem k tomu, že textové dokumenty jsou stále nejčastějším způsobem komunikace požadavků a specifikací systémů mezi stranami, má formální textová modalita, která je generována „zdarma“ a vždy odpovídá grafické specifikaci, velkou hodnotu. Kromě textu existují i různé jiné způsoby definování a specifikování požadavků, včetně storyboardů a maket, které si získávají popularitu v softwarovém průmyslu, kde lidé mají stále méně trpělivosti jak s únavnými textovými dokumenty, tak se složitými koncepčními modely. Formální model OPM může sloužit jako základ pro generování takových populárních prostředků. Práce v tomto směru již byla zahájena vytvořením animovaného filmu ze simulované animace modelů OPM v OPCAT (Bolshchikov et al. 2015).

11.2.2 Cíl OPL orientovaný na stroje

Cíl OPL orientovaný na stroje souvisí s jeho formalitou. OPL poskytuje pevný základ pro automatické generování navržené aplikace – infrastruktury potřebné pro pokračování vývoje aplikace.

OPL je formálně definován pomocí kontextově nezávislé gramatiky, takže textový soubor OPL může sloužit jako základ pro generování artefaktů aplikace, které zahrnují spustitelný kód a schéma databáze. Tento přístup umožňuje round-trip engineering, při kterém se změny v analýze, návrhu a specifikaci téměř automaticky promítají do finální aplikace. Tyto vlastnosti činí kombinaci graficky orientovaného OPD a jeho ekvivalentního textového protějšku OPL ideální infrastrukturou pro specifikaci systémů.

11.3 Princip graficko-textové ekvivalence OPM

Výchozí OPL je angličtina, ale jako základ pro OPL může sloužit jakýkoli přirozený jazyk. Vzhledem k tomu, že OPD je založeno na grafice a ikonických symbolech, může sloužit jako společná platforma pro překlad mezi OPL v různých přirozených jazycích.

Odstavec OPL v OPD je souborem vět OPL, které textově vyjadřují stejné modelové skutečnosti, které tento OPD vyjadřuje graficky.

V každém okamžiku během modelování (když v modelu nejsou žádné nespojené věci) lze OPD přesně rekonstruovat z jeho odstavce OPL a naopak. To je vyjádřeno v následujícím *principu OPM ekvivalence grafiky a textu*.

Princip graficko-textové ekvivalence OPM

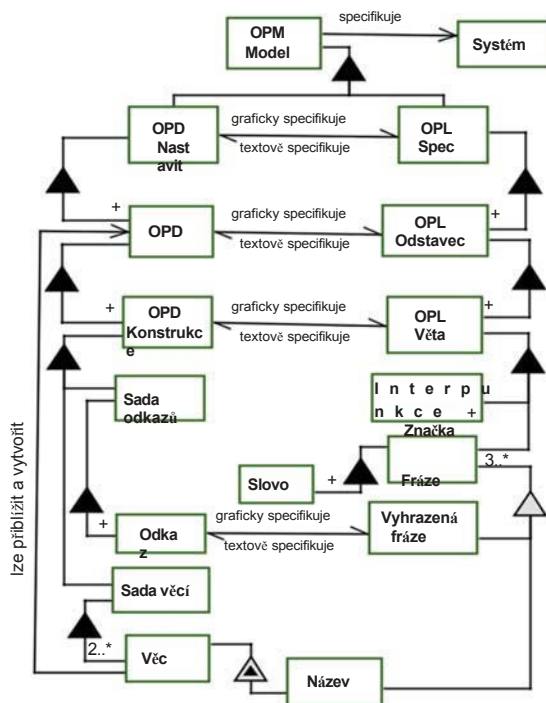
Jakýkoli modelový fakt vyjádřený graficky v OPD je také vyjádřen textově v odpovídajícím odstavci OPL.

Sada OPD je kompletní grafické znázornění modelu OPM. Jedná se o sadu (hierarchicky uspořádaných) OPD, které společně specifikují všechny modelové skutečnosti v modelu OPM.

Specifikace OPL (OPL Spec) modelu OPM je soubor všech jedinečných vět OPL, které textově vyjadřují všechny modelové skutečnosti, které graficky vyjadřuje sada OPD.

11.4 Metamodel struktury modelu OPM

Zatímco komplexní metamodel OPM je uveden v příloze normy ISO 19450 (viz [kap. 24](#)), na [obr. 11.1](#) uvádíme obecný model struktury modelu OPM, který uvádí výše uvedené definice do kontextu. Model modelu je metamodel. Tento model OPM je tedy metamodel. Pomocí OPM k určení struktury modelu OPM systému znázorňuje koncepční aspekty OPM jako paralelní hierarchie grafických a textových modalit OPM a jejich korespondenci za účelem vytvoření ekvivalentních modelových výrazů. Tento OPD je systémový diagram (SD nebo SD0) – diagram nejvyšší úrovně (úroveň nula) celého metamodelu OPM.



Model OPM specifikuje systém.

Model OPM se skládá z OPD Set a OPL Spec. OPL Spec se skládá z alespoň jednoho OPL Paragraph. OPD Set se skládá z alespoň jednoho OPD.

Sada OPD graficky specifikuje specifikaci OPL.

Specifikace OPL textově specifikuje sadu OPD.

OPD se skládá z alespoň jednoho OPD konstrukt.

Odstavec OPL se skládá z alespoň jedné věty OPL. OPD

graficky specifikuje odstavec OPL.

OPL odstavec textově specifikuje OPD.

OPD konstrukce graficky specifikuje OPL větu. OPL věta

textově specifikuje OPD konstrukci.

Konstrukce OPD se skládá ze sady věcí a sady

odkazů. Sada věcí se skládá z 2 až mnoha věcí.

Sada odkazů se skládá z alespoň jednoho

odkazu. Věc má název.

OPL věta se skládá z 3 až mnoha frází a alespoň jednoho interpunkčního znaménka.

Fráze se skládá z alespoň jednoho slova.

OPL Reserved Phrase a Name of Thing jsou fráze. Link

graficky specifikuje Reserved Phrase.

Rezervovaná fráze textově specifikuje odkaz.

Věc lze přiblížit a vytvořit OPD

Obr. 11.1 Metamodel (model OPM) struktury modelu OPM

Dva objekty v horní části OPD na [obr. 11.1](#) jsou **OPM Model** a **System**, propojené jednosměrným šitkováným strukturálním odkazem z prvního na druhý, což vede k větě OPL **OPM Model specifikuje System**. Dále se **OPM Model** skládá z **OPD Set** a **OPL Spec**. Jedná se o dvě doplňkové modalitty – grafickou a textovou. Od tohoto bodu OPD zobrazuje dvě paralelní hierarchie – grafickou a textovou –, kde sestupování znamená zvýšenou úroveň detailů.

Grafická hierarchie je **OPD Set**, **OPD**, **OPD Construct** a (na stejné úrovni) **Link Set** a **Thing Set**. Textová hierarchie, která je paralelní s grafickým **OPD Set** a **OPD**, je **OPL Paragraph** a **OPL Sentence**. **OPD** a odpovídající **OPL Paragraph** jsou sbírky modelových faktů, které modelář umístí do stejného diagramu – stejného modelového kontextu. Na další úrovni zpřesnění v této hierarchii je **OPD Construct** grafickým protějškem odpovídající textové **OPL Sentence** a opět oba vyjadřují stejný modelový fakt. Poté je grafický prvek **Link** (Odkaz) paralelní s **Reserved OPL Phrase** (Vyhrazená fráze OPL), protože ta textově specifikuje ten první, jako v případě vyhrazené fráze OPL, která se skládá z textového protějšku symbolu agregace-účasti, ▲, a v případě vlivů, které jsou textovým protějškem efektového odkazu, ↗.

11.5 Vyhrazené a nevyhrazené fráze OPL

Ačkoli OPL je podmnožinou angličtiny, je formální. Formální syntaxe OPL je vyjádřena kontextově nezávislou gramatikou v rozšířené Backus-Naurově formě (EBNF) v příloze A veřejně dostupné specifikace ISO 19450 (viz oddíl 24.4.1). Specifikace EBNF OPL zahrnuje asi 400 produkčních pravidel na 12 stranách. Pomocí EBNF sada produkčních pravidel jednoznačně definuje, jak mají být věty OPL konstruovány a analyzovány. [Obrázek 11.2](#) představuje tři produkční pravidla jako příklady vyjádření syntaxe OPL v EBNF. Každé produkční pravidlo má pravou a levou stranu, oddělené znakem

=. První produkční pravidlo určuje, že odstavec OPL se skládá z jedné nebo více vět OPL, oddělených symbolem „nový řádek“. Druhé produkční pravidlo určuje, že věty OPL se skládají z formální věty OPL následované symbolem tečky („.“). Třetí produkční pravidlo určuje, že formální věta OPL může být jednoho ze čtyř typů: věta popisující věc, procedurální věta, strukturální věta nebo věta pro správu kontextu.

V programovacích jazycích jsou analogiemi slov tokeny – atomové jednotky vzniklé lexikální analýzou. Ve většině programovacích jazyků jsou tokeny odděleny mezerami. Tokeny jsou zadávány do dalšího procesu, který se nazývá parsování. Věty OPL jsou samozřejmě mnohem čitelnější než skript v jakémkoli programovacím jazyce. Tyto věty jsou pečlivě navrženy s použitím podskupiny angličtiny, aby sdělovaly jasný a přímý význam prostřednictvím dobře formulovaných a lidsky srozumitelných konstrukcí. Přesto, pomocí formální definice syntaxe OPL založené na EBNF, mohou být věty OPL analyzovány stejně jako příkazy nebo řádky v programovacím jazyce. Stejně jako v programovacích jazycích, analýza věty OPL vede k frázím.

Fráze je kombinace jednoho nebo více slov oddělených mezerami, která tvoří logickou entitu, ale není úplnou větou. Fráze OPL mohou být rezervované a nerezervované. Jakákoli věta OPL se skládá z nerezervovaných frází OPL – slov nebo slovních kombinací specifických pro doménu nebo systém – které používá architekt systému nebo modelář, a rezervovaných frází OPL, které propojují nerezervované fráze a umožňují vytvoření věty v přirozeném jazyce.

A.4 OPL Syntax

A.4.1 OPL document structure

(* Region OPL document *)

```
OPL paragraph = OPL sentence, { new line, OPL sentence};
OPL sentence = OPL formal sentence, " ";
OPL formal sentence = thing description sentence
                      | procedural sentence
                      | structural sentence
                      | context management sentence;
```

Obr. 11.2 Tři příkladové produkční pravidla vyjadřující syntaxi OPL v EBNF

Fráze OPL je sekvence jednoho nebo více slov ve větě OPL.

Nerezervovaná fráze OPL je modelářem definovaná fráze OPL, která vyjadřuje název entity nebo relace modelu OPM specifický pro systém nebo doménu.

Nerezervované fráze OPL jsou názvy objektů, procesů a stavů OPM, které modelář přiřazuje při vytváření OPD, které tvoří model OPM. Nerezervované fráze OPL zahrnují také méně často používané fráze, jako jsou (uživatelé definované) označené strukturální vztahy a omezení účasti.

Vyhrazená fráze OPL je fráze OPL zabudovaná do definice syntaxe OPL EBNF, která spojuje dvě nebo více nevyhrazených frází OPL.

Vyhrazené fráze OPL jsou části syntaxe věty, které vyjadřují vztahy nebo spojení mezi nevyhrazenými frázemi OPL nebo omezení na nich. Příklady vyhrazených frází OPL jsou „vyžaduje“, „vydává“, „spotřebovává“, „a“, „nebo“, „ovlivňuje“, „přesně jeden z“, „alespoň jeden“ a „skládá se z“. Tyto definice vyhrazených a nevyhrazených frází stanoví, že první z nich jsou pojivem, které „spojuje“ a smysluplně drží pohromadě stavební bloky modelu – nevyhrazené fráze, které vyjadřují termíny specifické pro daný systém.

Následující konvence tučného písma OPL pomáhá rozlišit mezi těmito dvěma druhy frází OPL.

Konvence tučného písma OPL

Nerezervované fráze OPL se zobrazují tučným písmem **Arial**, zatímco rezervované fráze OPL se zobrazují písmem Arial.
. Interpunkční znaménka jsou tučně.

Například fráze OPL „**Automatic Crash Responding**“ (**Automatická reakce na nehodu**) ve větě OPL „**Automatic Crash Responding** affects **Vehicle Occupants Group**.“ (Automatická reakce na nehodu ovlivňuje skupinu osob ve vozidle) není vyhrazená, a proto se zobrazuje tučným písmem Arial. Fráze, které nejsou tučné, jako „and“ (a), „or“ (nebo), „affects“ (ovlivňuje), „exactly one of“ (přesně jeden z) a „consists of“ (skládá se z), jsou vyhrazené fráze OPL.

Implementace nástroje CASE musí automaticky překládat modelové skutečnosti vyjádřené konstrukcemi OPD do vět OPL. Aby bylo možné lépe rozlišovat mezi jednotlivými věcmi, měly by tyto nástroje používat barvy písma frází, které odpovídají jejich barvám v OPD. Pro objekty je výchozí barva v OPCAT zelená, pro procesy modrá, pro stavy hnědá a pro nevyhrazené fráze OPL černá. Pokud vaše verze knihy

umožňuje zobrazení barev, [obr. 10.6](#), který je modelem OPM generovaným OPCAT pro testovací systém procesu, ilustruje tuto konvenci barev.

11.6 Motivace pro bimodální vyjádření OPM

Legitimní otázkou, kterou lze v souvislosti s OPL položit, je, proč je kromě diagramu potřebný také text, pokud máme dobré grafické znázornění našeho modelu? Člověk se skutečně může divit, proč jsou potřeba dvě modalitty. Podle *principu ekvivalence grafiky a textu v OPM* vyjadřují text a grafika stejný obsah, takže z hlediska informačního obsahu dochází ke 100% redundanci! Nemá to plýtvání zdroji? Nebylo by rozumnější držet se pouze jedné modalitty – buď grafiky, nebo textu – a druhou vynechat?

11.6.1 Předpoklad dvou kanálů

Ekvivalence grafiky a textu je hlavním zdrojem expresivní síly OPM. Text OPL doplňuje grafiku OPD. Tato dualita implementuje *předpoklad dvoukanalového vnímání* (Clark a Paivio 1991; Baddeley 1992). Jedná se o jeden ze tří hlavních kognitivních předpokladů podložených výzkumem (Mayer 2003; Mayer a Moreno 2003), který stanoví, že lidé disponují oddělenými kanály a mechanismy pro zpracování vizuálních a verbálních reprezentací. Kombinace OPD a OPL přímo odpovídá tomuto předpokladu dvou kanálů (Dori 2008). Někteří lidé jsou více vizuálně orientovaní, zatímco jiní jsou více orientovaní na text. Text a grafika se navzájem posilují, zatímco tvůrce modelu nebo čtenáři modelu se snaží pochopit sémantiku, kterou prvky modelu vyjadřují v různých kombinacích.

Kognitivně-fyziologickým základem tohoto principu je skutečnost, že lidská mysl je uzpůsobena k přijímání jak vizuálně-obrazových-grafických signálů, tak audio-verbálních-psaných signálů. Grafika a text aktivují různé oblasti mozku. V běžném jazyce se to často označuje jako funkce levé/pravé hemisféry mozku. Levá hemisféra skutečně dominuje v jazyce, zpracovává to, co člověk slyší, a zajišťuje většinu úkolů souvisejících s mluvením. Pravá hemisféra je zodpovědná hlavně za prostorové schopnosti, rozpoznávání tváří, porozumění vizuálním obrazům a pochopení toho, co vidíme. Proto je pravděpodobnější, že se poselství – koncepční model – podaří sdělit, pokud se osloví „obě strany mozku“ prostřednictvím jazyka a obrázků. V souladu s tím je model, který lze prezentovat bimodálně v grafické i textové podobě, upřednostňován před modelem, který lze prezentovat pouze v jedné z těchto modalit. Téměř všechny jazyky konceptuálního modelování jsou buď textové, nebo grafické, ale ne obojí. OPM je první, který kombinuje obě modalitty (USPTO 7,099,809, 2006).

11.6.2 Výhody bimodální reprezentace

Jednotlivci mají různé preference, pokud jde o způsob, jakým čtou a píší specifikace. Obvykle lidé orientovaní na techniku (někdy považovaní za „levoprávé mozky“) preferují diagramy, zatímco lidé orientovaní na obchod („pravoprávé mozky“) upřednostňují text. Navíc i pro stejného jedince může být obsah někdy jasnější při pohledu na jednu modalitu, zatímco jindy je užitečnější doplňková modalita. Skutečnost, že OPL je podmnožinou angličtiny, *lingua franca* přirozeného jazyka, činí jej čitelným a srozumitelným pro lidi, aniž by se museli učit jakýkoli programovací nebo pseudokódový jazyk

. Syntaxe a sémantika OPL jsou dobře definovány, což eliminuje nejednoznačnost, která je často vlastní přirozeným jazykům.

Syntaxe OPL je navržena tak, aby výsledný text tvořil jednoduché přirozené, i když syntakticky omezené, anglické věty. Proto bimodální graficko-textové znázornění modelu OPM pomáhá zapojit netechnické zúčastněné strany do zjišťování požadavků a počátečního koncepčního modelování vyvíjeného systému. Toto zapojení takových zúčastněných stran je motivuje k aktivní účasti a pomáhá odhalit chyby brzy po jejich neúmyslném zavedení.

Předpokládejme například, že místo použití obousměrné šipky $\longleftrightarrow$, která je efektním propojením, by modelář z **obr. 1.3** omylem použil jednosměrnou šipku $\rightarrow$, která je výsledkovým propojením, z procesu **Automatická reakce na nehodu** k objektu **Skupina cestujících ve vozidle**, aby vyjádřil skutečnost, že **Automatická reakce na nehodu** ovlivňuje **Skupinu cestujících ve vozidle**. V tomto případě by byla vytvořena následující věta OPL:

Automatická reakce na nehodu vede ke skupině cestujících ve vozidle.

Je zřejmé, že tato věta, ačkoli je syntakticky správná, nedává smysl. Modelář nebo zástupce zákazníka účastníci se modelovacího sezení by to pravděpodobně okamžitě zjistil. Zjištěnou chybu lze pak okamžitě opravit a opravu lze ověřit pouhým přečtením nově vytvořené věty OPL.

Jako cílový jazyk, do kterého se konstrukce OPD převádějí, lze vybrat jakýkoli přirozený (i umělý) jazyk. Grafické znázornění je navíc jazykově neutrální, a může proto sloužit jako prostředek pro překlad z jednoho jazyka do druhého.

11.6.3 Zapojení zákazníka: sociální aspekt

Pomocí softwarového produktu podporujícího OPM se věty OPL vytvářejí automaticky v reálném čase v reakci na zadávání grafických symbolů OPD na obrazovce. Tato schopnost každého člena týmu poskytovat okamžitou zpětnou vazbu o skutečnostech modelovaných během procesu modelování je nanejvýš důležitá, protože umožňuje okamžitou interpretaci záměrů lidského vývojáře systémem.

Jednoduchost a přímočarost této reakce v reálném čase na grafický vstup modeláře ve formě podskupiny angličtiny je velmi cenná; nejenže umožňuje zachytit chyby ihned po jejich vzniku, ale také umožňuje aktivní účast zákazníka systému na modelovací relaci, kde může poskytovat okamžitou zpětnou vazbu v průběhu modelování. Hodnota takové účasti tedy přesahuje pouhé odhalování chyb při jejich vzniku; jedná se o sociální proces, který zapojuje zainteresované strany na straně zákazníka již v rané fázi návrhu, což jim oprávněně dává pocit, že jsou součástí rozhodovacího procesu, a zmírňuje odpor ke změnám, což je obecně známá lidská vlastnost. Specifikace OPL systému vyplývající ze sady OPD je tak přístupná k přezkoumání, úpravám a konečnému potvrzení zákazníkem nebo odborníky v dané oblasti, kteří jednají jeho jménem a nemusí být odborníky na software.

Zapojení zástupců zákazníka, kteří pracují bok po boku s vývojáři, zvyšuje pravděpodobnost odhalení a zachycení chyb v návrhu ihned po jejich vzniku, což vede k významné úspoře času, peněz a problémů v budoucnu. Tato zpětná vazba v reálném čase je nepostradatelná nejen při odhalování chyb, ale také při jejich opravě v rané fázi životního cyklu systému

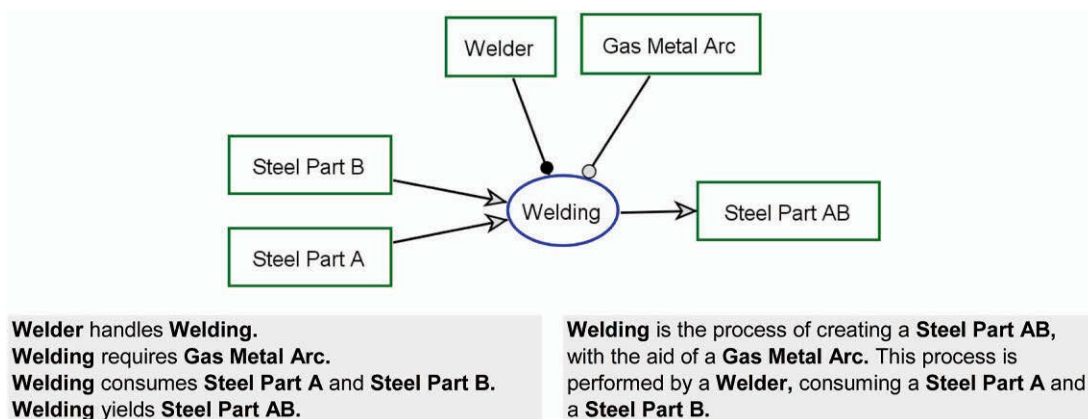
životního cyklu, než se stačí rozšířit a způsobit nákladné škody. Jakákoli grafická úprava (přidání nebo odstranění prvku) mění skript OPL. Změny lze provádět, dokud není dosaženo uspokojivého výsledku a zákazník může „podepsat“ model jako plán systému, který má být vyvinut.

11.6.4 Překlenutí propasti mezi požadavky a návrhem

Schopnost přímo a přesně převést výsledky analýzy a návrhu do podskupiny přirozeného jazyka má obrovskou výhodu. Jak již bylo uvedeno, potenciální uživatelé a zákazníci mohou být více zvyklí na čtení textu než na interpretaci OPD, natož na dešifrování programového kódu. Tímto způsobem text OPL a jeho grafický ekvivalent OPD pomáhají překlenout mezeru mezi původní specifikací požadavků, která je v současné době stále vyjádřena volnou prózou, a skutečnou specifikací systému vyjádřenou výsledným modelem OPM. Zatímco věty OPL jsou pro člověka snadno srozumitelné a dokumentují tak systém „zdarma“, schopnost je analyzovat poskytuje pevný základ pro automatizované úkoly, jako je generování spustitelného kódu, simulace, generování počátečního uživatelského rozhraní a definice schématu databáze.

11.7 Tesperanto: Člověkem čitelný automaticky generovaný text

OPL se skládá z krátkých, často nesouvislých vět. Ačkoli každá věta OPL je syntakticky a sémanticky správná anglická věta, nedostatek plynulosti mezi jednotlivými větami OPL brání tomu, aby se OPL stal důstojnou náhradou volného textu, který dominuje reálným požadavkům a dalším technickým specifikacím, jako jsou mezinárodní normy. Dlouhý text OPL, který je mechanický a repetitivní a postrádá plynulost, není pro lidské čtení přirozený. To vedlo k vývoji Tesperanta (Blekhnman a Dori 2013) jako další úrovně automatického generování textu z grafiky na základě modelu, který doplňuje nebo nahrazuje OPL.



Obr. 11.3 Model OPM **svařování v ochranné atmosféře**. Nahoře: OPD. Dole vlevo: OPL. Dole vpravo: Tesperanto

Tesperanto je vylepšením OPL, které se řídí principy postupného prezentování OPM, které zohledňují omezenou kognitivní kapacitu člověka. Zahrnuje heuristiku pro úpravy délky vět, synonyma, pořadí slov, kontrolu opakování frází a další algoritmy, jejichž cílem je, aby text

Tesperanto méně mechanickým a čitelnějším pro člověka. [Obrázek 11.3](#) je model OPM **svařování** v ochranné atmosféře. OPD nahoře je automaticky přeloženo do OPL a Tesperanta vlevo a vpravo dole. Tento jednoduchý příklad demonstruje rozdíly v plynulosti čtení OPL a Tesperanta. Zatímco oba překlady textu z grafiky věrně odrážejí formální a ověřený grafický model OPM, Tesperanto je čitelnější a méně nudné, repetitivní a mechanické. Například zatímco v OPL se proces **svařování** opakuje čtyřikrát, jednou pro každý druh procedurálního vztahu, v překladu Tesperanto se objevuje pouze jednou. Vzhledem k tomu, že Tesperanto se jako předmět výzkumu stále vyvíjí, není v této knize dále používáno.

11.8 Shrnutí

- *Object-Process Language (OPL)* je podmnožina angličtiny, která textově vyjadřuje model OPM, který graficky vyjadřuje sada OPD.
- Formální syntaxe pro OPL je vyjádřena kontextově nezávislou gramatikou v rozšířené Backus-Naurově formě (EBNF) v příloze A veřejné specifikace ISO 19450 (PAS).
- *Modelový fakt* je vztah mezi dvěma nebo více věcmi v modelu OPM.
- *OPD prvek* je grafickým vyjádřením věci nebo odkazu.
- *Konstrukce OPD* je soubor propojených prvků OPD.
- OPL slouží dvěma cílům, orientovaným na dva směry: lidi a stroje.
- Cílem OPL orientovaným na člověka je převést sadu OPD tvořících model OPM do textu v přirozeném jazyce.
- Cílem OPL orientovaným na stroje je poskytnout pevný základ pro automatické generování infrastruktury pro vývoj aplikací.
- *Odstavec OPL* v OPD je soubor vět OPL, které textově vyjadřují stejné modelové skutečnosti, které tento OPD vyjadřuje graficky.
- *Princip graficko-textové ekvivalence OPM*: Jakýkoli modelový fakt vyjádřený graficky v OPD je také vyjádřen textově v odpovídajícím odstavci OPL.
- *Metamodel* je model modelu.
- Metamodel struktury modelu systému OPM zobrazuje dvě paralelní hierarchie – hierarchii grafických objektů a odpovídající hierarchii textových objektů.
- *Specifikace OPL* modelu OPM je soubor vět OPL, které textově vyjadřují všechny skutečnosti modelu, které graficky vyjadřuje sada OPD.
- *OPL fráze* je sekvence jednoho nebo více slov.
- *Nerezervovaná fráze OPL* je fráze OPL definovaná modelářem, která vyjadřuje název entity nebo relace modelu OPM specifický pro systém nebo doménu.
- *Rezervovaná fráze OPL* je fráze OPL zabudovaná do definice syntaxe OPL EBNF, která spojuje dvě nebo více nerezervovaných frází OPL.

- *Předpoklad dvoukanálového* vnímání spočívá v tom, že lidé disponují oddělenými systémy pro zpracování vizuálních a verbálních reprezentací.
- Syntaxe a sémantika OPL jsou definovány jako podmnožina angličtiny, což eliminuje nejednoznačnost, která je často vlastní přirozeným jazykům.
- Tesperanto je další generací OPL.

11.9 Problémy

1. Jaké jsou výhody a nevýhody textové specifikace systémového modelu vedle grafické modality?
2. Kdybyste měli navrhnout nový modelovací jazyk s omezením, že může používat pouze jednu modalitu, kterou byste si vybrali? Proč?
3. Která z následujících tří definic slova „meta“ převzatých ze slovníku dictionary.com odpovídá metamodelu?
4. Předpona vyskytující se v přejatých slovech z řečtiny, s významy „po“, „spolu s“, „za“, „mezi“, „za“ a produktivní v angličtině podle řeckého vzoru: metacarpus; metagenesis.
5. Předpona připojená k názvu předmětu a označující jiný předmět, který analyzuje původní předmět, ale na abstraktnější, vyšší úrovni: metaphilosophy; metalinguistics.
6. Předpona připojená k názvu něčeho, co vědomě odkazuje na svůj vlastní předmět nebo vlastnosti nebo je komentuje: meta-obraz umělce malujícího plátno, metakognice; metaanalýza.
7. Zkopírujte tři věty OPL z této kapitoly a obraťte jejich tučné písmo, tj. změnit tučné písmo na normální a naopak. Kterou verzi preferujete – původní nebo obrácenou? Proč?

Kapitola 12

SysML: Základy a diagramy

Ať už se jedná o pokročilé vojenské letadlo, hybridní vozidlo, mobilní telefon nebo distribuovaný informační systém, od těchto systémů se očekává výkon na úrovni, o které se před jednou generací ani nesnilo.

Friedenthal, Moore a Steiner (2012)

SysML (Systems Modeling Language) je profilem jednotného modelovacího jazyka (UML), tj. přizpůsobenou verzí určenou pro aplikace v oblasti systémového inženýrství. Presentaci SysML zahájíme stručným popisem UML, následovaným přehledem SysML a jeho různých typů diagramů s odkazem na OPM. Připomeňme, že zatímco OPM používá jediný model, který kombinuje různé aspekty systému a prezentuje je v grafické a textové podobě, SysML používá devět druhů diagramů, z nichž každý se zaměřuje na nějaký konkrétní aspekt systému. Zaměřujeme se na typy diagramů SysML, které dosud nebyly diskutovány: sekvenční diagram, diagram aktivit, diagram požadavků a diagram parametrů. *Sekvenční diagram* zobrazuje časový průběh a výměnu zpráv mezi bloky. *Diagram aktivit* představuje aktivity prováděné systémem, jejich pořadí a řízení. *Diagram požadavků* představuje uživatelské a odvozené požadavky, které musí systém splňovat. A konečně, *diagram parametrů* modeluje výpočty, které se provádějí v systému.

12.1 UML: Unified Modeling Language

Unified Modeling Language (UML) je standardizovaný vizuální specifikační jazyk pro modelování softwarových systémů. Specifikace UML (OMG UML 2011I, 2011S) je definována a udržována organizací Object Management Group, OMG, neziskovým konsorciem pro specifikace v počítačovém průmyslu, které poprvé přizpůsobilo specifikaci UML v listopadu 1997.

UML je vizuální modelovací jazyk. Jako takový specifikuje grafickou notaci spolu s odpovídající sémantikou, které se společně používají k vytvoření abstraktního modelu systému. UML byl navržen jako jazyk pro vývoj softwarových systémů, které jsou implementovány pomocí objektově orientovaného programovacího jazyka. UML specifikuje několik grafických pohledů oddělených podle aspektů, celkem 13, které reprezentují model systému. Z 13 typů diagramů UML je šest typů strukturálních diagramů a sedm typů behaviorálních diagramů.

Od svého zavedení se UML stal dominantním modelovacím jazykem v softwarovém průmyslu. UML, který je používán velkou částí softwarové inženýrské komunity a podporován mnoha komerčními a open-source softwarovými modelovacími nástroji, prošel několika menšími verzemi (1.x) a jednou větší revizí (UML 2).

Ačkoli má UML kořeny v oblasti softwarového inženýrství, byly podniknuty některé pokusy, např. (Holt 2004), o jeho aplikaci v obecnější oblasti systémového inženýrství. Tyto pokusy však

byly uznány jako problematické, především proto, že UML je zaměřeno na software, takže jeho ontologie a taxonomie je omezena na softwarové artefakty. Například fyzické vlastnosti a komponenty systému jsou v diagramech UML vyjádřeny jen stěží. Kromě toho UML dostatečně nepodporuje modelování hierarchií v rámci systémového modelu, což je v systémovém inženýrství zásadní problém.

Snaha přizpůsobit UML aplikacím systémového inženýrství vedla k založení OMG Systems Engineering Domain Special Interest Group (SE DSIG). Tato skupina OMG, podporovaná Mezinárodní radou pro systémové inženýrství (INCOSE) a pracovní skupinou ISO AP 233, spolupracovala na požadavcích na modelovací jazyk. Výsledkem byla žádost o návrh UML pro systémové inženýrství, zkráceně UML for SE RFP (OMG UML 2003), vydaná OMG v březnu 2003. SysML byla jedinou odpovědí na tuto žádost. Tým SysML se skládal z uživatelů z průmyslu, prodejců nástrojů, vládních agentur, profesních organizací a akademické sféry. Čtyři a půl roku po zveřejnění RFP byla v září 2007 organizací OMG oficiálně vydána verze 1.0 specifikace SysML jako specifikace OMG.

12.2 Pilíře SysML

SysML je univerzální modelovací jazyk pro systémové inženýrství, který je určen k podpoře specifikace, analýzy, návrhu, ověřování a validace složitých systémů. Systémy mohou být velmi rozmanité a mohou zahrnovat hardware, software, data, personál, postupy, zařízení a další. SysML znovu využívá podmnožinu UML 2 a poskytuje další rozšíření, aby splnil požadavky RFP. Jako vizuální modelovací jazyk nabízí SysML několik druhů diagramů, které mohou odrážet různé aspekty systému.

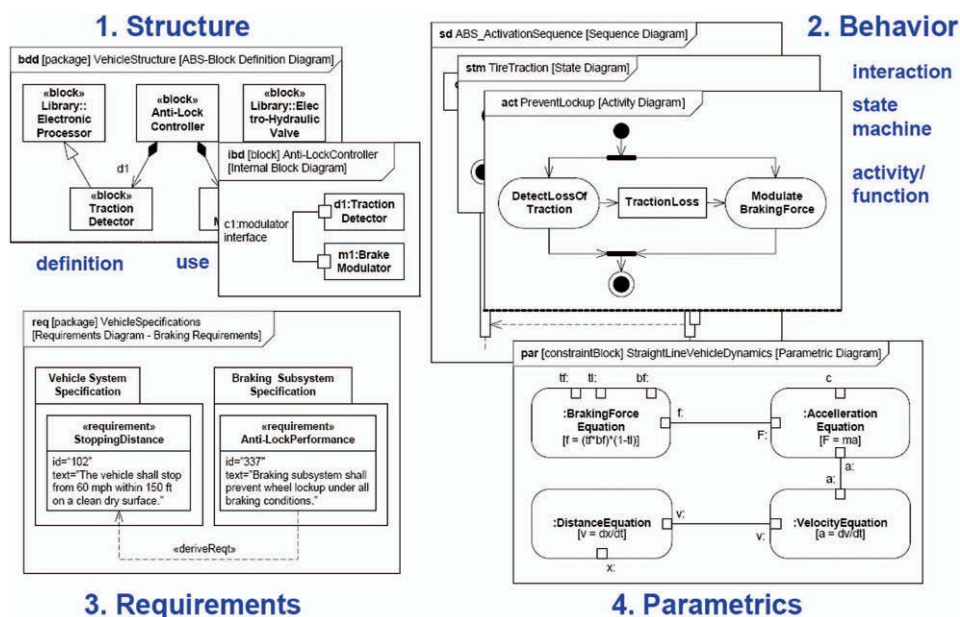
Diagramy SysML se obvykle dělí do čtyř „pilířů“ – struktura, chování, požadavky a parametrické vztahy. Kromě toho SysML poskytuje prostředky pro propojení různých prvků modelu. [Obrázek 12.1](#), upravený podle Friedenthal et al. (2012), ukazuje příklady klíčových typů diagramů SysML. Celkově SysML zahrnuje devět typů diagramů: čtyři typy strukturálních diagramů, čtyři typy diagramů chování a diagram požadavků.

Taxonomie diagramů SysML je uvedena na [obr. 12.2](#). Pomocí notace OPM [obr. 12.2](#) ukazuje, které diagramy byly převzaty z UML beze změny, které diagramy byly převzaty z UML se změnami a které diagramy jsou nové. Čtyři diagramy SysML jsou stejné jako jejich protějšky v UML: diagram užití, diagram balíčku, diagram sekvence a diagram stavového stroje. Tři jsou upraveny z UML 2: diagram definice bloku, diagram vnitřního bloku a diagram aktivity. Nakonec jsou přidány dva nové diagramy: diagram požadavků a diagram parametrů. Následně jsou popsány všechny čtyři pilíře SysML.

12.3 Diagram požadavků

Požadavky jsou primárním vstupem každého systému nebo produktu. SysML poskytuje prostředky pro reprezentaci textových požadavků a jejich propojení s jinými prvky modelu. Základní požadavek, reprezentovaný pomocí stereotypu „požadavek“, se skládá z jedinečného identifikátoru a textových vlastností. Je také možné jej rozšířit o další vlastnosti (např. vlastnost stavu ověření). Diagram požadavků lze zobrazit v různých formátech: grafickém, tabulkovém nebo ve formě stromové struktury. Požadavky mohou být také součástí jiných

diagramů, které odrážejí vztah k jiným modelovým konstrukcím. Obecně platí, že konstrukce požadavků SysML nemají nahradit externí nástroje pro správu požadavků, ale spíše lépe integrovat systémové požadavky s jinými částmi modelu.

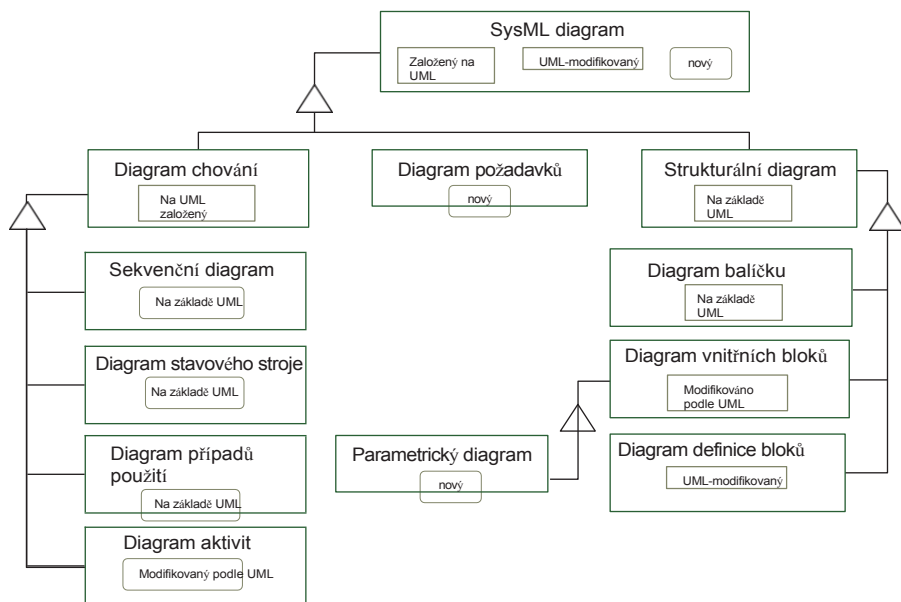


Obr. 12.1 Čtyři pilíře SysML (Friedenthal et al. 2012)

Specifikace SysML poskytuje několik vztahů mezi požadavky. Hierarchie požadavků, které se skládají ze složených požadavků a dílčích požadavků, lze popsat pomocí mechanismu obsahu jmenného prostoru UML. Závislost „deriveReqt“ popisuje vztah mezi odvozeným požadavkem a jeho zdrojovým požadavkem. Obvykle se požadavek na úrovni systému odvozuje do více požadavků na úrovni subsystému. Vztah „satisfy“ se používá k zobrazení toho, jak jeden nebo více požadavků splňuje návrh modelu. Dalšími vztahy jsou závislost „verify“, která propojuje požadavky a testovací případy, a závislost „refine“, která určuje, že prvek modelu SysML je upřesněním textového požadavku.

12.4 Bloky a struktura

Základním strukturálním prvkem v SysML je *blok*. Lze jej použít k popisu fyzických nebo logických prvků systému, jako je hardware, software, data nebo osoby. Bloky mohou popisovat jakoukoli úroveň hierarchie systému, od jednotlivých komponent až po systém nejvyšší úrovně. Blok v SysML je analogický třídě v UML, ale nejedná se o konstrukci specifickou pro software, nýbrž o strukturální prvek pro všeobecné použití.

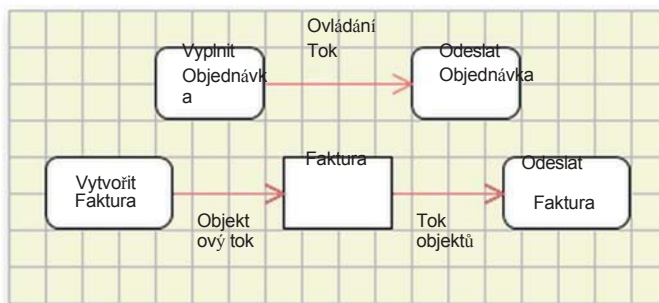


Obr. 12.2 Taxonomie diagramů SysML vyjádřená jako model OPM

Existují dva typy strukturálních diagramů pro znázornění bloků: diagram definice bloků a diagram vnitřních bloků. Diagram definice bloků (BDD) popisuje vztahy mezi bloky, jako jsou asociace, závislosti a generalizace. Specifikuje hierarchii systému, propojení částí a klasifikace. Interní diagram bloků (IBD) představuje vnitřní strukturu bloku pomocí vlastností bloků a spojení mezi vlastnostmi. Další strukturální diagram SysML, diagram balíčků UML, se používá k organizaci modelu seskupením prvků modelu.

12.5 Diagram aktivit

Jak je znázorněno na [obr. 12.2](#), SysML specifikuje čtyři typy behaviorálních diagramů: diagram aktivit, diagram sekvencí, diagram stavového stroje a diagram případů použití.



Obr. 12.3 Jednoduchý diagram aktivit SysML s blokovými a akčními uzly a kontrolními a objektovými toky

Aktivita je základním prvkem chování v různých behaviorálních diagramech SysML (s výjimkou diagramu případů použití). Úlohou diagramu aktivit (viz [obr. 12.3](#)) je znázornit tok vstupů a výstupů a tok řízení mezi akcemi. Za tímto účelem diagram aktivit zahrnuje sekvence a podmínky pro koordinaci aktivit. Aktivita a diagramy aktivit existují také v UML, ale SysML poskytuje několik rozšíření (Bock 2006), včetně prostředků na podporu modelování „kontinuálního“ toku, jako jsou omezení rychlosti. Do diagramů aktivit SysML byla přidána podpora pravděpodobnosti a rozšíření řízení (známá jako „řízení jako data“). Kromě toho, aby bylo možné hladce sladit SysML s široce používaným klasickým diagramem chování systémového inženýrství (známým jako EFFBD – Enhanced Functional Flow Block Diagrams; Bock, 2005), je specifikován stereotyp „effbd“. Pokud je tento stereotyp aplikován na aktivitu, znamená to, že aktivita musí vyhovovat omezením nezbytným pro EFFBD.

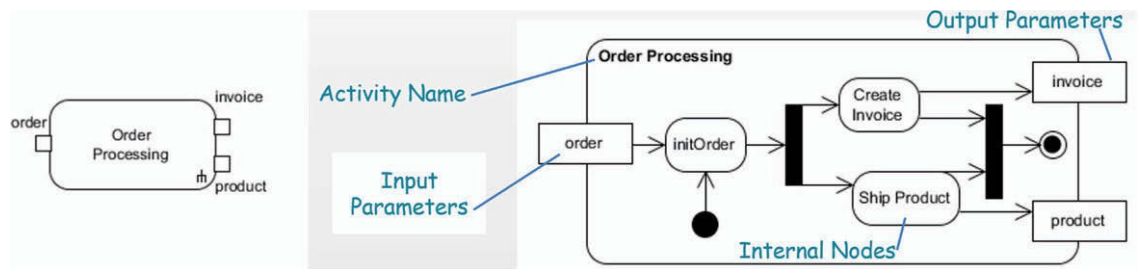
Diagram použití je určen k popisu základních funkcí na vysoké úrovni tím, že specifikuje použití systému jeho aktéry k dosažení cíle. Často se jedná o první typ diagramu, který se používá k poloformální specifikaci s zákazníkem za účelem definování funkce a rozsahu systému, který má být vyvinut.

Diagram aktivit je jediný druh behaviorálního diagramu, který je v SysML rozšířen ve vztahu k UML 2, zatímco ostatní tři druhy behaviorálních diagramů SysML zůstávají nezměněny nebo byly odstraněny. Sekvenční diagram se používá k znázornění toku řízení založeného na zprávách mezi interagujícími entitami, kterými mohou být aktéři, systémy nebo části systému. Diagram stavového stroje modeluje chování založené na stavech pomocí stavů objektů a přechodů.

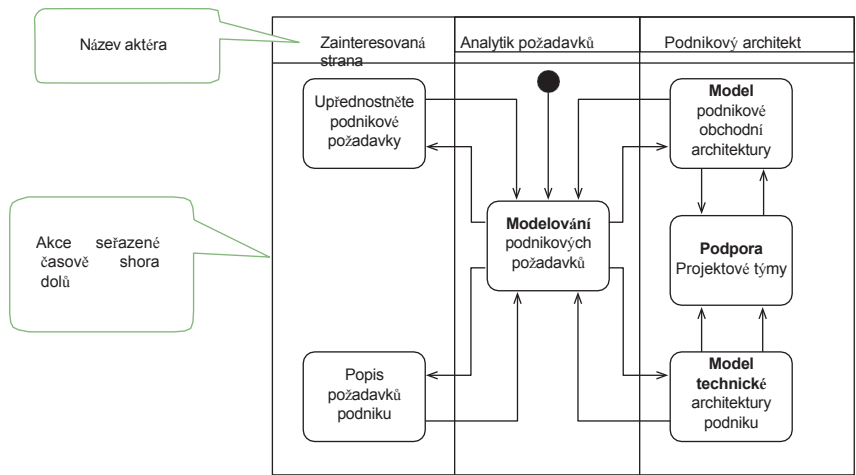
Akce (označená obdélníkem) je základní (obvykle atomická) jednotka procesu v diagramu aktivit. Jak ukazuje [obr. 12.3](#), diagram aktivit se skládá z uzlů a hran, kde uzel může být akcí nebo blokem (označeným obdélníkem) a hrana může být *tokem řízení*, pokud se nachází mezi akcemi, nebo *tokem objektů* (nebo *tokem bloků*), pokud se nachází mezi blokem a akcí.

12.5.1 Zpracování akce do aktivity

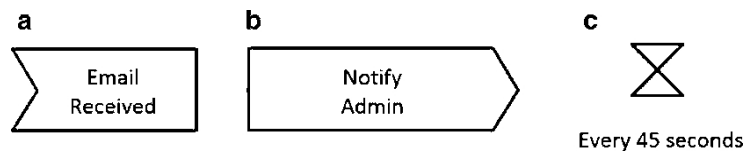
Pokud má akce, například **Zpracování objednávky** vlevo na [obr. 12.4](#), v pravém dolním rohu symbol malého hrábě (nebo trojzubce), znamená to, že se jedná o *volací akci*, která je rozvedena do aktivity s vlastním diagramem ([obr. 12.4](#), vpravo). Jedná se o podobný princip jako proces přiblížení v OPM. Bloky **Objednávka**, **Faktura** a **Produkt** jsou označeny jako kolíky – slouží jako vstupní a výstupní parametry.



Obr. 12.4 Akce Zpracování objednávky (vlevo) má malý symbol trojzubce, který označuje, že je rozvedena do aktivity (vpravo). Bloky Objednávka, Faktura a Produkt jsou označeny jako kolíky – vstupní a výstupní parametry



Obr. 12.5 Příklad diagramu aktivit s plavečnými drahami



Obr. 12.6 Tři speciální akční notace: (a) přijmout událost (b) odeslat signál (c) časová událost

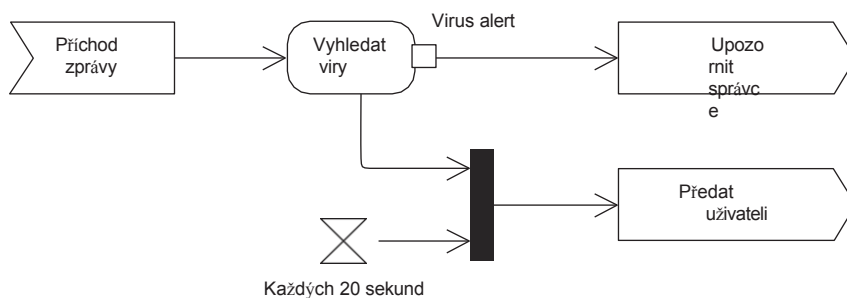
Diagram aktivity **zpracování objednávky** má *počáteční* a *konečné pseudovrcholy* – černé a černé kruhy na bílém pozadí – které označují začátek a konec aktivity. Má také dva synchronizační vrcholy: *rozvětvený* vrchol – tlustá svislá čára od akce **initOrder** k akcím **Create Invoice** a **Ship Order**, a *spojovací* vrchol – tlustá svislá čára od těchto dvou akcí k konečnému pseudovrcholu. Rozvětvený uzel označuje souběžný začátek akcí, které z něj vycházejí, zatímco spojovací uzel označuje ukončení všech akcí, které do něj vstupují.

Plavecká dráha je druh diagramu aktivit, který umožňuje seskupit aktivity prováděné stejným aktérem nebo seskupit aktivity v jednom vlákně. [Obrázek 12.5](#), upravený z Agile Modeling (2015), je příkladem diagramu aktivit plavecké dráhy. Aktéři jsou označeni ve svislých plaveckých drahách a časová osa diagramu běží shora dolů s vodorovnými odkazy překračujícími hranice plaveckých drah, kde je to nutné.

12.5.2 Uzly akcí přijetí, odeslání a časové události

Tři speciální akce mají specifické značky (viz [obr. 12.6](#)): (a) **přijmout událost**, která čeká na výskyt události (signálu), (b) **odeslat signál**, který při aktivaci vytvoří a odešle signál, a (c) **časová událost**, která čeká na určitý okamžik v čase nebo na konkrétní (případně periodickou) dobu trvání.

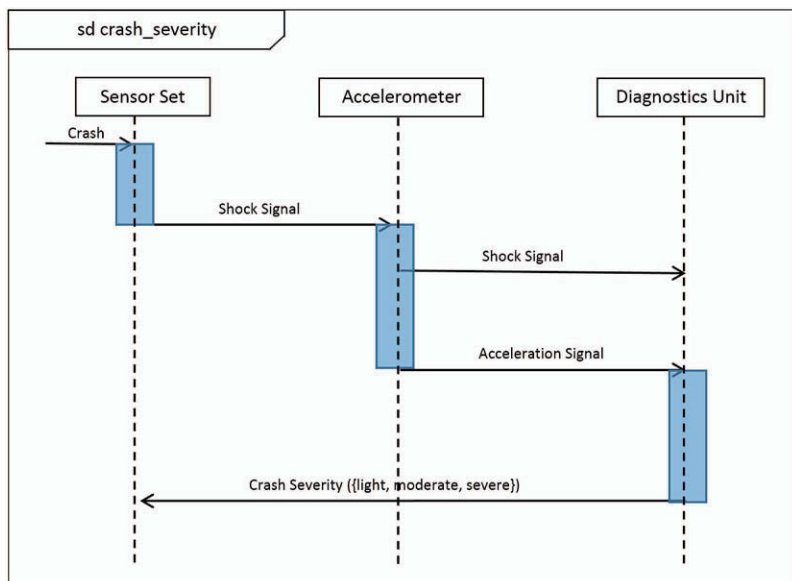
[Obrázek 12.7](#) ukazuje diagram aktivit s akcí přijetí události, odeslání signálu, časovou událostí, pinem (typu „Virus alert“) a spojovacím uzlem, což znamená, že naskenované zprávy bez detekovaných virů jsou předávány uživateli každých 20 sekund.



Obr. 12.7 Diagram aktivity s událostí přijetí, odesláním signálu, časovou událostí, pinem a spojovacím uzlem

12.6 Sekvenční diagram

Sekvenční diagram popisuje tok řízení mezi aktéry a bloky. Tento diagram představuje odesílání a přijímání zpráv mezi interagujícími entitami nazývanými lifelines.

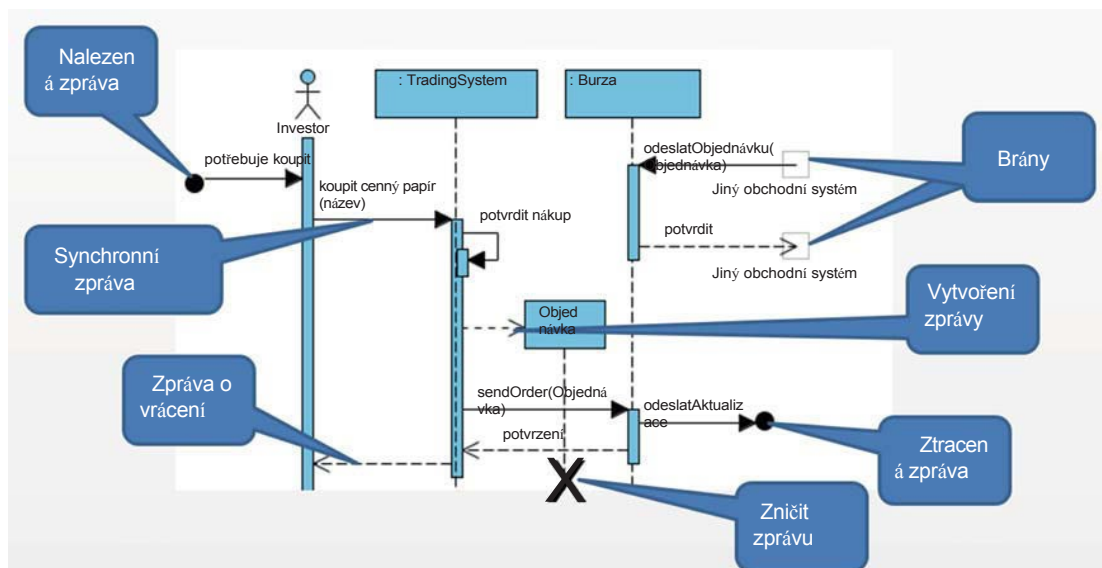


Obr. 12.8 Sekvenční diagram SysML závažnosti nehody

Čas je znázorněn na svislé ose shora dolů, podobně jako plavecké dráhy v diagramu aktivit. Jak je specifikováno v SysML 1.3, sekvenční diagramy mohou znázorňovat vysoce komplexní interakce se speciálními konstrukcemi, které představují různé typy řídicí logiky, odkazují na interakce v jiných sekvenčních diagramech a rozkládají životní linie na jejich jednotlivé části. Zde uvádíme pouze základní symboly a sestavujeme relativně jednoduchý sekvenční diagram.

Obrázek 12.8 je sekvenční diagram závažnosti nehody. Nahoře vidíme tři bloky: **Sada senzorů**, **Akcelerometr** a **Diagnostická jednotka**. Odpovídající *životní linie*, označená přerušovanou čarou, vede svisle dolů z každého bloku. Vodorovné šipky označují zprávy mezi bloky. Nejprve **senzorový set** přijme zprávu **Crash (Náraz)**, načež provede určitou operaci, nazývanou *provedení události* a označenou širokou čarou nebo protáhlým obdélníkem podél životní linie, jako je ten přímo pod **senzorovým setem**. Po dokončení provedení **senzorový set** odešle zprávu **Shock Signal (Signál nárazu)** **akcelerometru**. **Akcelerometr** začne pracovat a odešle **signál nárazu** dále do **diagnostické jednotky**, která následně provede svou operaci a odešle zpět do **sady senzorů** zprávu s hodnotou **závažnosti havárie**, která může být **lehká**, **střední** nebo **závažná**.

Jak ukazuje **obr. 12.9**, zprávy mohou být různých typů. Mohou být *synchronní* nebo *asynchronní* a mohou poskytovat *návratové hodnoty*. Zprávy mohou začínat od výskytů provedení, externího zdroje (bran) nebo neznámých zdrojů (nalezené zprávy). Mohou končit u výskytů provedení, externích cílů (bran) nebo neznámého cíle (ztracené zprávy). Bloky, jako je **Order** na **obr. 12.9**, mohou být vytvořeny a/nebo zničeny. Šipka zprávy může být nakloněna dolů namísto horizontální polohy, aby označila skutečnost, že předání zprávy samotné trvá určitý čas, a kvantifikovala latenci. Interakce (zprávy) mohou začínat nebo končit na branách od jiných bloků nebo systémů.



Obr. 12.9 Druhy zpráv v sekvenčním diagramu SysML a jejich symboly

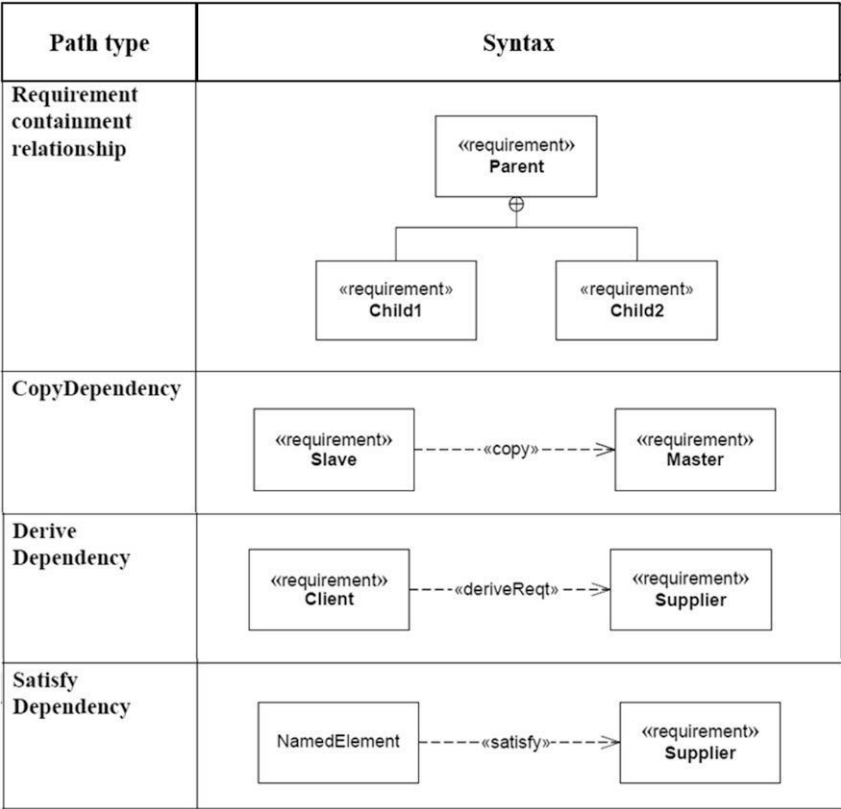
12.7 Diagram požadavků

Diagram požadavků a parametrický diagram jsou dva zcela nové druhy diagramů, které SysML přidalo do UML a které nejsou součástí UML. Diagramy požadavků propojují typické nástroje pro správu požadavků a systémový model. Jak uvádí oficiální stránka OMG SysML, diagram požadavků SysML je grafická konstrukce pro znázornění textových požadavků a jejich propojení s jinými prvky modelu.

Diagram požadavků zachycuje hierarchie a odvození požadavků. Lze jej použít k ověření vztahů mezi požadavky a jejich implementací tím, že modelář může propojit požadavek s prvkem modelu, který tento požadavek splňuje nebo ověřuje.

Hlavní symboly diagramu požadavků SysML jsou uvedeny na [obr. 12.10](#). Požadavek je znázorněn jako blok s vyhrazeným slovem „požadavek“ v horní části. V dokumentu SysML 1.3 se to označuje jako stereotyp třídy UML, který podléhá sadě omezení.

Vztah obsahu, znázorněný jako přeškrtnutý kruh (jako ten pod rodičem na [obr. 12.10](#)), označuje, že požadavek připojený ke kruhu obsahuje ty, které jsou s ním propojeny. To umožňuje vytvářet hierarchie požadavků. Tři další hlavní závislé vztahy mezi bloky, označené přerušovanými šipkami, jsou znázorněny na [obr. 12.10](#). Stereotyp „copy“ označuje, že „Slave“ je kopií „Master“ (na který ukazuje šipka). Stereotyp „deriveReq“ označuje, že požadavek „Client“ je odvozen od „Supplier“ (na který ukazuje šipka). Stereotyp „satisfy“ označuje, že blok „namedElement“ je systémová komponenta, která splňuje požadavek, na který šipka ukazuje.



Obr. 12.10 Hlavní symboly diagramu požadavků SysML

Tři další závislosti podobné povahy jsou „verify“, „refine“ a „trace“. Závislost „verify“ je mezi požadavkem a blokem, který poskytuje způsob jeho ověření. Může to být blok se stereotypem „testCase“. Závislost „refine“ označuje, že nějaký propracovaný požadavek upřesňuje obecnější požadavek, např. požadavek klienta. Závislost „sledovat“ označuje, že nějaký blok poskytuje způsob, jak sledovat požadavek. Závislost „sledovat“ poskytuje způsob, jak sledovat, kde jsou požadavky v systému splněny, protože často bývá obtížné požadavky sledovat nebo není jasné, proč byla některá komponenta do modelu zahrnuta.

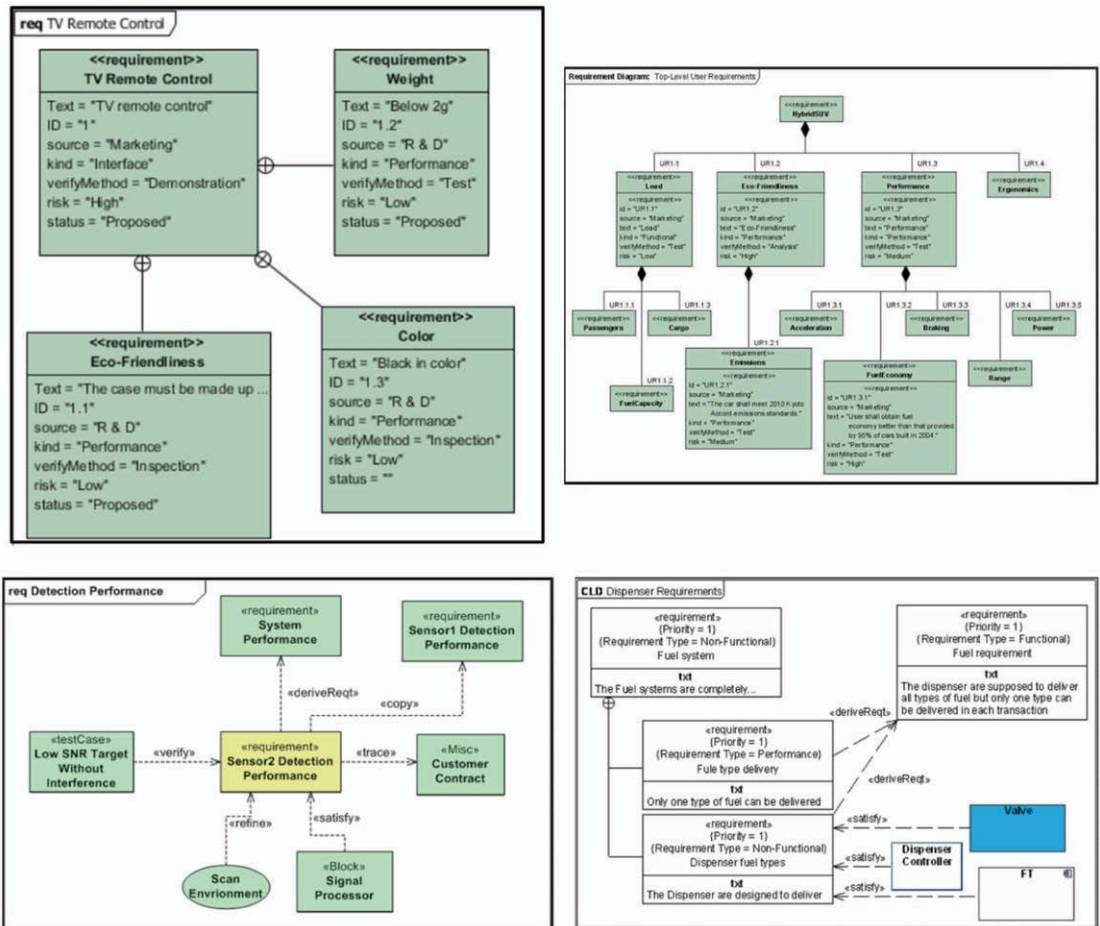
Standardní požadavek zahrnuje vlastnosti pro specifikaci jeho jedinečného identifikátoru a textového požadavku. Uživatel může specifikovat další vlastnosti, jako je stav ověření. Diagramy požadavků jsou skutečně znázorněny v široké škále forem a stylů.

Obrázek 12.11 představuje čtyři příklady nalezené na webu, které demonstrují tuto variabilitu stylů.

Diagram v levém horním rohu má název **req TV Remote Control** (požadavek na dálkové ovládání televizoru). Obsahuje čtyři požadavky, z nichž hlavní se také nazývá TV Remote Control (dálkové ovládání televizoru). Ostatní tři, nazvané Weight (hmotnost), Color (barva) a Eco-Friendliness (ekologická šetrnost), jsou „podřízené“ – požadavky nižší úrovně, které jsou podřízené nadřazenému požadavku TV Remote Control (dálkové ovládání televizoru).

To je označeno čarami s překříženými kruhy na koncích, v souladu se standardní specifikací na [obr. 12.10](#). Kromě atributů text a ID má každý požadavek také atributy source, kind, verifyMethod, risk a status.

Diagram v pravém horním rohu má název **Diagram požadavků** – uživatelské požadavky nejvyšší úrovně. Norma stanoví, že k označení diagramu požadavků se používá **req**. Použití černého kosočtverce, který je symbolem z diagramu definice bloků (bdd), je další nestandardní aplikací. Místo toho by se měl použít symbol obsahu.



Obr. 12.11 Příklady velké rozmanitosti diagramů požadavků SysML

Diagram v levém dolním rohu s názvem „**Req** Detection Performance“ (Výkon detekce požadavků) ukazuje použití všech šesti výše popsanych typů závislosti. Například „testCase“ Nízký SNR cíl bez rušení „verify“ „requirement“ Výkon detekce senzoru 2. Dalším příkladem je „block“ Signálový procesor „satisfy“ „requirement“ Výkon detekce senzoru 2. Žádný z požadavků v tomto diagramu nemá

kompartment vlastností s minimální sadou vlastností – text a ID, ale norma (sekce 16.3.1.2) umožňuje tento kompartment vynechat (nechat mimo).

Zajímavé je, že prostředí skenování, které je případem použití, je původcem závislosti „upřesnit“. To se nezdá být standardem povoleno, protože podle OMG SysML v1.3 oddíl 16.3.1.1 (str. 144) „Diagram požadavků může zobrazovat pouze požadavky, balíčky, jiné klasifikátory, testovací případy a odůvodnění.“ Toto propojení mezi případy použití a požadavky však může být užitečné. Při snaze obhájit legálnost míchání symbolů z různých diagramů jsme zjistili, že informativní příloha A – Diagramy standardu SysML v1.3 (str. 168) uvádí:

„Ačkoli taxonomie poskytuje logickou organizaci pro různé hlavní druhy diagramů, nevylučuje pečlivé míchání různých typů diagramů, jak by se mohlo stát při kombinování strukturálních a behaviorálních prvků (např. zobrazení stavového stroje vnořeného do oddílu bloku). Je však zásadní, aby typy prvků diagramu, které se mohou objevit v konkrétním druhu diagramu, byly omezené a dobře specifikované. Tabulky prvků diagramu v každé klauzuli popisují, jaké symboly se mohou v diagramu objevit, ale nespecifikují různé kombinace symbolů, které lze použít.“

Tento odstavec v podstatě dává modelářům SysML úplnou svobodu kombinovat a spojovat jakékoli symboly z jakéhokoli typu diagramu SysML s jakýmkoli jinými symboly. Stačí pouze „pečlivě kombinovat“ a zajistit, aby „typy prvků diagramu, které se mohou objevit v konkrétním typu diagramu, byly omezené a dobře specifikované“. Co však „omezené a dobře specifikované“ znamená, není samo o sobě specifikováno, což ponechává prostor pro libovolnou interpretaci modeláře.

Nakonec diagram v pravém dolním rohu s názvem Požadavky na výdejní stojan **CLD** ukazuje v horní části každého požadavku kromě názvu (např. Dodávka typu paliva) také prioritu požadavku (např. {Requirement Type=Non-Functional}) a typ (např. {Type=1}). Pole vlastností obsahuje relativně podrobný text (např. „Lze dodávat pouze jeden typ paliva“), ale neobsahuje ID. Tři bloky (Ventil, Řídící jednotka výdejního zařízení a FT) splňují požadavek na typ paliva výdejního zařízení, který je odvozen od jiného, komplexnějšího požadavku.

12.8 Parametrický diagram a bloky vlastností omezení

Parametrický diagram SysML umožňuje vyjádřit omezení mezi vlastnostmi, čímž umožňuje integraci matematických výpočtů nebo technických analýz, jako jsou modely výkonu a spolehlivosti, s návrhovými modely SysML. Bloky vlastností omezení mohou také specifikovat síť kvantitativních omezení vyplývajících z matematických výrazů fyzikálních vlastností systému. Omezení jsou zachycena v blocích vlastností omezení – konstrukcích ConstraintBlock, vyjádřených jako rovnice, které zahrnují základní parametry. Například ConstraintBlock může mít parametry F , m a a a omezení $\{F=m \cdot a\}$. Dalším příkladem je rovnice kinetické energie $E=mv^2/2$. Parametry výkonu a jejich vztahy k jiným parametrům lze sledovat po celou dobu životního cyklu systému.

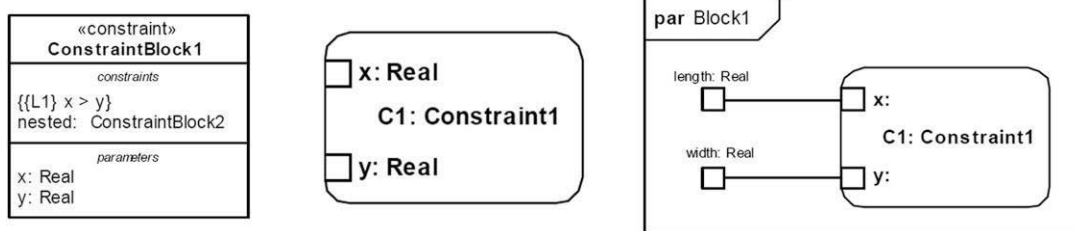
Bloky vlastností omezení SysML umožňují integraci inženýrské analýzy, jako jsou modely výkonu a spolehlivosti, s jinými modely SysML. Blok omezení (viz [obr. 12.12](#) vlevo) zahrnuje omezení, obvykle v podobě matematické rovnice, a parametry omezení, jako jsou E , m a v pro energii, hmotnost a rychlost.

Pro účely opětovného použití jsou bloky omezení definovány v diagramu definice bloků. Parametrické diagramy používají bloky omezení k omezení vlastností hodnot jiných bloků. Bloky omezení lze tak znovu použít v diagramech definice bloků a zabalit do knihoven modelů pro všeobecné použití nebo pro konkrétní domény. Toto omezení se provádí vazbou parametrů omezení (například m v příkladu výše) na konkrétní vlastnosti skutečných hodnot bloku (například hmotnost vozidla). Příklad parametrického diagramu je uveden na [obr. 12.13](#). [Obrázek 12.14](#) je OPD reprezentace tohoto parametrického diagramu.⁽¹⁾

Parametrický diagram (viz [obr. 12.12](#) vpravo) používá jeden nebo více bloků vlastností omezení k omezení vlastností jednoho nebo více jiných bloků svázáním parametrů prostřednictvím matematického vztahu. Vlastnost omezení může být zobrazena na parametrickém diagramu pomocí standardního tvaru obdélníku vnitřní vlastnosti s klíčovým slovem „constraint” (zkratka pro constraintProperty) před jejím názvem (viz [obr. 12.12](#) vlevo). Omezení vlastnosti však může být v parametrickém diagramu zobrazeno také pomocí obdélníku (viz [obr. 12.12](#) uprostřed). Stejně jako u standardního obdélníku mohou být v obdélníku zobrazeny oddíly a vnitřní vlastnosti. Použití tohoto tvaru umožňuje vyhnout se nutnosti explicitně zaznamenávat klíčové slovo „constraint“.

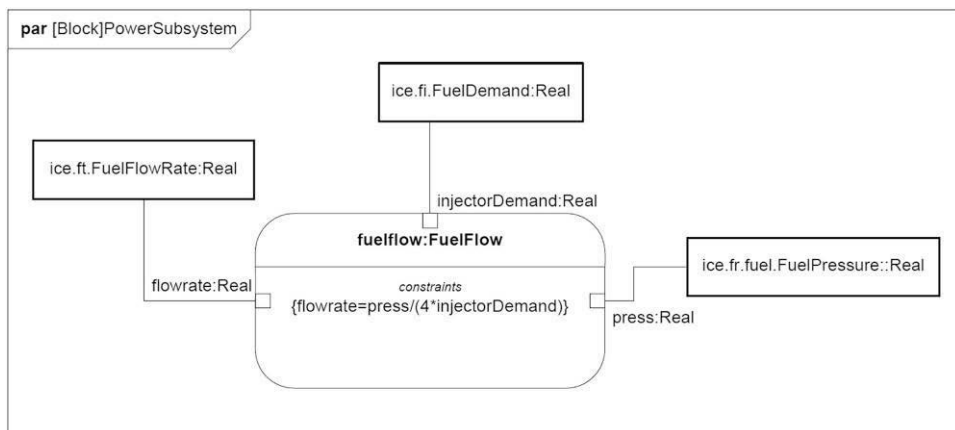
[Obrázek 12.13](#) představuje parametrický diagram, který omezuje průtok paliva na spotřebu paliva automobilu a tlak paliva prostřednictvím vztahu $\text{FuelFlow} = \text{Pressure}/(4 * (\text{InjectorDemand}))$, pravděpodobně kvůli tomu, že se jedná o čtyřválcový motor.

Jakákoli matematická operace, od základních čtyř aritmetických operací po nejsložitější výpočty, může být v OPM zobrazena jako výpočetní (informatický) proces, který používá (ale nespotebovává) jeden nebo více vstupních parametrů k vytvoření výstupu. S ohledem na to lze jakýkoli parametrický diagram SysML prezentovat jako OPD, kde omezení je **výpočetní** proces předcházející matematickému výrazu, který váže a omezuje vstupní parametry. Vstupní parametry jsou nástroje, pokud nechceme specifikovat, že po skončení **výpočetního** procesu nebudou zachovány.



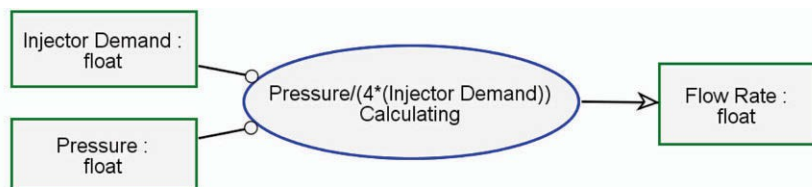
Obr. 12.12 Obecný blok omezení (vlevo) a parametrický diagram (vpravo). Zdroj: OMG SysML v1.3, str. 84

⁽¹⁾ Další, složitější příklad výpočtů je uveden na [obr. 22.5](#) a [obr. 22.6](#).



Obr. 12.13 Parametrický diagram, který omezuje průtok paliva na spotřebu paliva a tlak paliva (zdroj: OMG SysML v1.3, str. 199)

Při použití této jednoduché transformace konceptu je [obr. 12.14](#) OPD reprezentací parametrického diagramu na [obr. 12.13](#). Kromě této kompaktní grafické reprezentace získáme „zdarma“ textovou reprezentaci OPL, kterou lze snadno převést do kódu v jakémkoli programovacím jazyce nebo dokonce přímo spustit pro výpočet **Pressure/(4*(Injector Demand))**. Další, složitější příklad výpočtů v OPM je uveden na [obr. 22.5](#) a [22.6](#).



Flow Rate, Injector Demand, and Pressure are of type float.
Pressure/(4*(Injector Demand)) Calculating requires **Pressure** and **Injector Demand**.
Pressure/(4*(Injector Demand)) Calculating yields **Flow Rate**.

Obr. 12.14 OPD znázornění parametrického diagramu na [obr. 12.13](#)

12.9 Porovnání SysML a OPM

V této části nejprve porovnáme SysML s OPM z hlediska vztahu k jednotlivým jazykům jako celku a poté diskutujeme a ukážeme, jak lze OPM použít k modelování několika typů diagramů SysML. [Tabulka 12.1](#) poskytuje stručné srovnání SysML a OPM z hlediska různých atributů.

Tabulka 12.1 Porovnání atributů SysML a OPM

Funkce	SysML	OPM
Teoretický základ	UML; Objektově orientovaný paradigm	Minimální univerzální ontologie; Objektově-procesní věta
Standardní dokumentace počet stran	~1670=700 (UML Infrastruktura) + 700 (UML Superstruktura)+ 270 (OMG SysML)	~180=100 (hlavní standard ISO 19450 standard)+ 80 (přílohy)
Normalizační orgán	OMG	ISO
Počet druhů diagramů	9	1
Koncept nejvyšší úrovně	Blok (třída objektů UML)	Věc (objekt nebo proces)
Zásada řízení složitosti	Rozklad podle aspektů	Rozklad na úrovni detailů
Hierarchická dekompozice	V některých typech diagramů	Ano
Počet symbolů	~120	~20
Grafická modalita	Ano	Ano
Textová modalita	Ne ²	Ano
Vestavěné fyzicko-informatické rozlišení	Ne	Ano
Rozlišení mezi systémem a prostředím	Částečné (s využitím hranic)	Ano
Logické vztahy (OR, XOR, AND)	Ne	Ano
Modelování pravděpodobnosti	Ne	Ano
Schopnost provádění, animované simulace, validace a ověřování	Částečná (v některých nástrojích pro některé typy diagramů)	Ano
Dostupnost nástrojů	Mnoho, některé zdarma	V současné době jeden bezplatný (OPCAT) od http://esml.iem.technion.ac.il/ Cloudový nástroj ve vývoji

² Je možné generovat textovou modalitu pomocí modulů pro vytváření zpráv a dokumentace některých komerčních nástrojů SysML, ale to není součástí jazykového standardu a neexistuje předdefinovaná syntaxe pro formální věty.

12.9.1 Procesy OPM jako prvotřídní občané

OPM vychází z filozofie, která stanoví, že pro věrné a přirozené modelování, analýzu a návrh systémů v jakékoli oblasti je třeba procesy uznat jako „prvotřídní občany“. Stejně jako objekty jsou i procesy OPM považovány za bona fide, samostatné „věci“, nikoli za součásti objektů, jak to prosazuje objektově orientovaný (OO) přístup. Nedostatek přímého uznání procesu jako základního ontologického konceptu vedle objektu vede k množství termínů a symbolů pro proces v UML a SysML: use case, activity, action, method a sequence. Všechny jsou procesy, ale každý z nich má určité nuance nebo konotace, které nejsou explicitně definovány.

Objekty v OPM jsou věci, které přetrvávají, zatímco procesy jsou přechodné věci, které transformují objekty. Procesy transformují objekty jedním ze tří způsobů: (1) ovlivňují jejich stav, (2) generují nové objekty nebo

(3) spotřebovávají existující objekty.

12.9.2 Fyzické a informatické věci

OPM, které je od počátku zaměřeno na systémové inženýrství a zachází se softwarovými systémy jako se specializacemi obecných systémů, nemá žádnou inherentní „softwarově orientovanou“ sémantiku jazyka. Například objekty a procesy OPM mohou být *informatické* nebo *kybernetické*, které mohou existovat v modelech softwarových systémů i jiných obecných systémů, nebo *fyzické*, které jsou atypické pro čisté softwarové systémy, ale zjevně nezbytné pro systémy obecné.

Objekty i procesy mohou být fyzické nebo informatické. Objekty a procesy mohou být nejen fyzické nebo informační, ale také systémové (součást systému) nebo environmentální (součást prostředí interagujícího se systémem).

12.9.3 Množství modelů versus jedinečnost modelu

Hlavním rozdílem mezi SysML a OPM je počet pohledů (typů diagramů) používaných v každém jazyce. Zatímco OPM je založeno na jediném typu diagramu – diagramu objekt-proces (OPD), SysML zdědilo mnohost modelů UML (Peleg a Dori 2000), tj. prezentuje každý aspekt systému v jiném pohledu pomocí jiného typu diagramu. SysML zahrnuje podmnožinu diagramů UML a také dva nové typy diagramů pro systémové inženýrství: diagram požadavků a parametrický diagram.

Sada vzájemně souvisejících diagramů objekt-proces (OPD), které zobrazují části systému v různých úrovních detailu, tvoří grafický, vizuální formalismus OPM. OPD, jediný typ diagramu OPM, může postrádat některé prvky, které jsou důležité pro systémové inženýrství, jako jsou parametrická omezení SysML, i když tyto prvky lze v OPM zpracovat jako atributy s hodnotami, které jsou manipulovány procesy, které jsou matematickými operacemi.

Oba jazyky podporují hierarchické znázornění modelu. Na rozdíl od SysML, kde je model znázorněn v samostatných pohledech s částečnou podporou hierarchie, je v OPM celý model systému založen na dobře definované hierarchii OPD. To je jen několik z mnoha rozdílů mezi těmito jazyky, které činí studium a srovnání jejich rozdílů zajímavým.

12.9.4 Grafika versus bimodální kombinace grafiky a textu

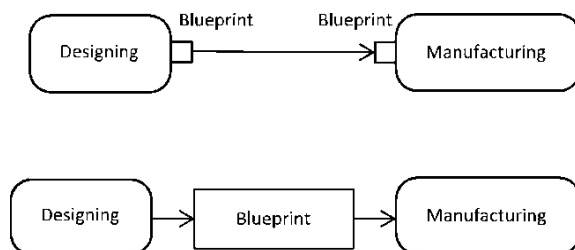
OPM kombinuje matematicky podložený formální, ale jednoduchý grafický jazyk s větami v přirozeném jazyce, aby vyjádřil funkci, strukturu a chování systémů v integrovaném, jednotném modelu. Dvě sémanticky ekvivalentní modalita, jedna grafická a druhá textová, společně vyjadřují stejný model OPM. Ačkoli jsou vizuálně-grafická a verbálně-textová modalita sémanticky ekvivalentní, oslovují dva různé kanály zpracování informací v mozku, vizuální a jazykový.

OPM je hlavním nástrojem pro provádění úkolů souvisejících s vývojem systému. Dělá to přímočarým, přátelským a jednoznačným způsobem. Jedním z důležitých důvodů je to, že design OPM nebyl ovlivněn tím, co současné programovací jazyky mohou nebo nemohou dělat, ale spíše tím, co dává největší smysl při pokusu o co nejlepší reprezentaci a koncepční modelování systémů.

Díky výsledné intuitivnosti je OPM srozumitelný jak technickým, tak netechnickým zúčastněným stranám vyvíjeného systému, včetně kolegů, zákazníků a implementátorů. Současně je OPM díky své formálnosti vhodný pro počítačovou manipulaci za účelem automatického nebo poloautomatického generování velkých částí koncipovaného systému, zejména programového kódu a schématu databáze.

12.9.5 Srovnání diagramů aktivit s OPD

Jak již bylo uvedeno, toky v diagramu aktivit mohou být dvou druhů: tok řízení a tok objektů (nebo bloků). Tok řízení označuje tok z jedné akce do druhé bez výslovného uvedení objektu. [Obrázek 12.7](#) je příkladem, kde jsou všechny toky toky řízení; zpráva předávaná z jedné akce do druhé je implicitní. Naproti tomu tok objektu má konkrétní objekt, který je výstupem jedné akce a vstupem do další. Existují dva způsoby modelování toku objektu, oba jsou znázorněny na [obr. 12.15](#). Ten nahoře je pinová notace: **Blueprint** je jak výstupem na pinu akce **Designing**, tak vstupem na pinu akce **Manufacturing**. Typ vstupu a výstupu musí být stejný. Druhý způsob modelování toku objektů, znázorněný ve spodní části [obr. 12.15](#), je explicitní objektová notace: Objekt (nebo blok) je znázorněn jako výstup i vstup.



Obr. 12.15 Dva způsoby zobrazení toku objektů. Nahoře: pinová notace. Dole: objektová notace

Aktivita diagramu ve spodní části [obr. 12.15](#) vypadá velmi podobně jako OPD. Vypadá to, že stačí nahradit symbol aktivity – obdélník – symbolem OPD pro proces – elipsou, a získáme sémanticky ekvivalentní model. Tímto způsobem vznikne OPD na [obr. 12.16](#). Oba modely vypadají opravdu izomorfne: stačí nahradit tvar a hotovo! Abychom však získali přehled o přesné sémantice tohoto OPD, měli bychom si přečíst jeho odstavec OPL:



Obr. 12.16 První pokus o modelování OPM diagramu aktivit v dolní části [obr. 12.15](#)

Návrh vede k vytvoření **plánu**. **Výroba** spotřebovává **plán**.

Je to opravdu to, co jsme chtěli modelovat? Je **Blueprint** opravdu spotřebováván **výrobou**? To, co opravdu chceme modelovat, je to, že jakmile je **Blueprint** připraven, umožňuje **výrobu**. **Výroba** však **Blueprint** nespotřebovává, ale spíše na něj odkazuje, takže **Blueprint** je umožňujícím prvkem, nebo přesněji řečeno nástrojem: je vyžadován **výrobním** procesem, ale není jím zničen. Navíc, zatímco **Blueprint** je informatický objekt, **výroba** je fyzický proces.

Odstavec OPL tohoto OPD je skutečně výmluvnější a potvrzuje náš vylepšený grafický model ([obr. 12.17](#)):

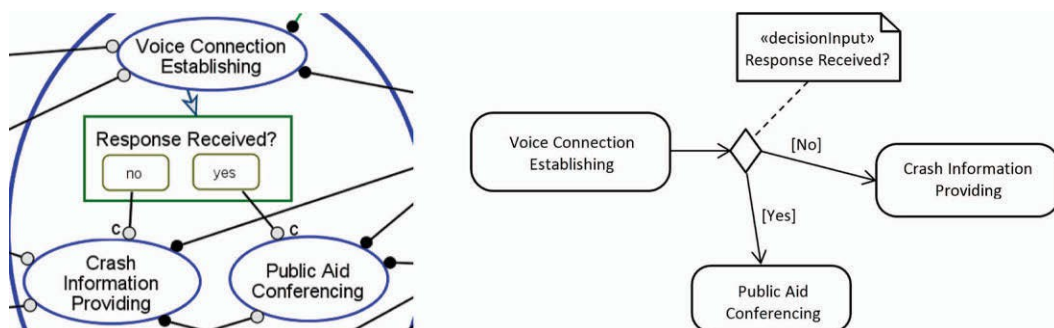
Návrh vede k **Blueprintu**. **Výroba** je fyzický proces. **Výroba** vyžaduje **Blueprint**.

Obr. 12.17 Vylepšený model OPM diagramu aktivit v dolní části [obr. 12.15](#)



Při zamyšlení nad myšlenkovým procesem, který tato úloha zahrnovala, si uvědomujeme, že sémantika diagramu aktivit je méně expresivní než sémantika OPD. Šipky mezi aktivitou a objektem v diagramu aktivit mají sémantiku *toku*, zatímco v OPM mají sémantiku transformace – vytvoření, spotřebu nebo změnu stavu. **Blueprint** v diagramu aktivit jednoduše „proudí“ mezi dvěma akcemi a implicitně mění svou logickou polohu, ale není specifikováno, zda je **Blueprint** spotřebováván **výrobou**, nebo ji pouze umožňuje. Naopak šipky OPM nemají sémantiku toku – neznamenají, že dotčený objekt mění svou polohu, ale pouze že prochází určitou transformací.

Diagram aktivit nemůže rozlišovat mezi nástrojem a spotřebitelem (stejně jako žádný jiný typ diagramu SysML, alespoň ne přímo). Rozdíl mezi informační podstatou návrhu a blueprintu na jedné straně a fyzickou podstatou výroby na straně druhé nelze také modelovat. Ani tento rozdíl v podstatě, ani rozdíl mezi aktivátorem a transformátorem nelze přímo modelovat žádným z devíti typů diagramů SysML.



Obr. 12.18 Příklad rozhodovacího uzlu v diagramu aktivit. Vlevo: relevantní část OPD. Vpravo: ekvivalentní diagram aktivit

12.9.6 Tok řízení v diagramu aktivit versus OPD

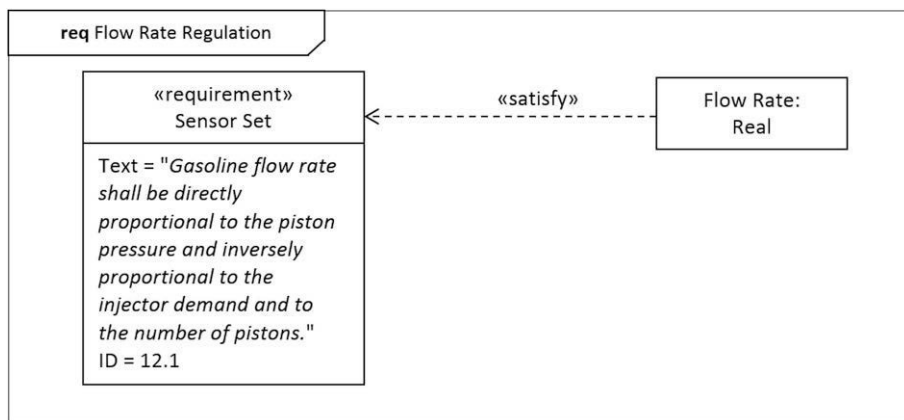
Tok řízení v diagramu aktivit je dosažen pomocí *rozhodovacích uzlů*, které jsou uzly ve tvaru kosočtverce, ze kterých vycházejí dvě nebo více linií toku řízení, jak je znázorněno na [obr. 12.18](#). Na levé straně [obr. 12.18](#) je relevantní část OPD, zatímco na pravé straně je ekvivalentní diagram aktivit.

Při zkoumání těchto dvou modelů vidíme, že OPM nevyžaduje speciální symbol rozhodnutí – diamant v diagramu aktivit. Místo toho se používá objekt se dvěma stavy – booleovský objekt. Kromě toho symbol rozhodnutí často vyžaduje, jako je tomu v tomto případě, poznámku (jejíž symbolem je kus papíru s ohnutým pravým horním rohem; viz [obr. 12.18](#)) s vyhrazeným slovem „decisionInput“, aby bylo možné specifikovat, co se v uzlu rozhodnutí rozhoduje. Poznámky jsou neformální anotace, které brání automatizaci provádění modelu. Kromě toho je rozhodovací proměnná často sama o sobě objektem nebo částí objektu (např. zprávou nebo výsledkem funkce), případně se stavy, atributy a dalšími detaily, ale použití rozhodovacího uzlu neumožňuje modelování tohoto objektu.

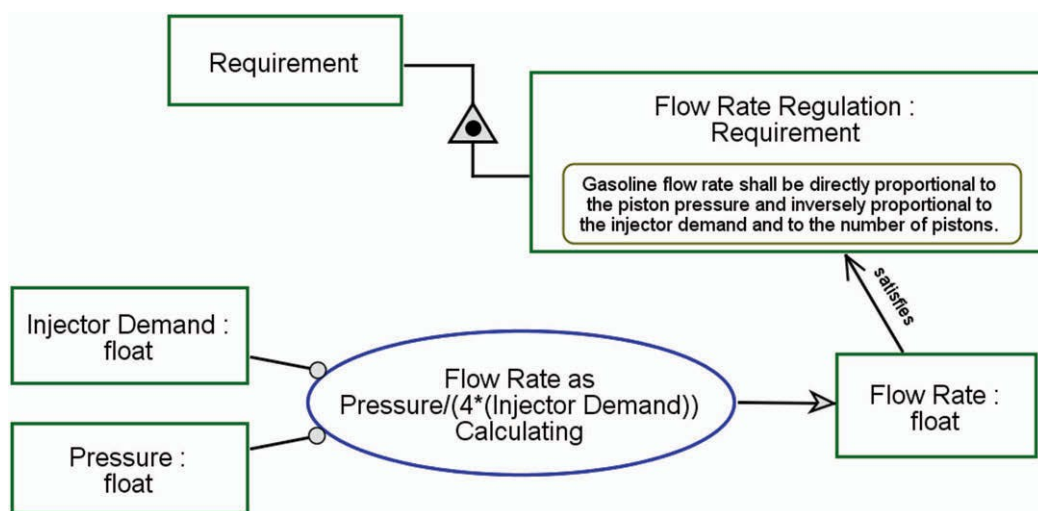
12.9.7 Implementace diagramu požadavků v OPM

Stejně jako u jiných diagramů SysML, které jsme viděli, umožňuje OPM modelovat požadavky bez nutnosti použití specializované sady symbolů. Uvažujme například následující požadavek nazvaný Regulace průtoku:

„Průtok benzínu musí být přímo úměrný tlaku pístu a nepřímo úměrný požadavku vstřikovače a počtu pístů.“



Obr. 12.19 Schéma požadavků na regulaci průtoku



Obr. 12.20 Požadavek na průtok z obr. 12.19 vyjádřený v modelu OPM

Tento požadavek je znázorněn v diagramu požadavků na [obr. 12.19](#). V OPM definujeme informatickou třídu objektů **Požadavek**, která má instanci pro **regulaci průtoku**. Jelikož **průtok** splňuje tento požadavek, rozšiřujeme OPD na [obr. 12.14](#) a vyjadřujeme tento vztah pomocí štítkovaného strukturálního odkazu OPM **splňuje** na [obr. 12.20](#).

Dalším přístupem je formální modelování požadavku a odvození textové specifikace požadavku z výsledného textu OPL, namísto psaní volných požadavků. Pomocí vztahu výstava-charakteristika můžeme modelovat **požadavek** na [obr. 12.20](#) jako výstavu několika částí, včetně **volného textu klienta**, **vitality**, **naléhavosti**, **stavu uspokojení** a **odvozeného požadavku**. Hodnota atributu **Volný text klienta** instance **Regulace průtoku** bude „Průtok benzínu musí být

přímo úměrný tlaku pístu a nepřímo úměrný požadavku vstřikovače a počtu pístů“.

Pro zvýšení obecnosti můžeme modelovat objekt Sada pístů, jehož hodnota atributu Velikost je parametr namísto čísla 4 v matematickém výrazu, který je součástí názvu procesu „**Průtok jako tlak/(4*(požadavek vstřikovače)) výpočet**“. Místo zahrnutí tohoto výrazu do názvu procesu můžeme navíc přiblížit a modelovat, jak se výsledek počítá krok za krokem pomocí parametrů. Tímto způsobem, pokud se změní některý z parametrů nebo dokonce výraz, není nutné měnit název procesu.

12.10 Synergie SysML–OPM

Grobshtein a Dori (2011) vyhodnotili aspekty SysML a OPM na základě konkrétního vzorového problému, ve kterém bylo modelováno více aspektů systému jak v SysML, tak v OPM. OPM se ukázalo jako výhodné při prezentaci různých hierarchických úrovní systému a kombinování struktury s chováním, zatímco SysML se ukázalo jako vhodnější pro modelování detailních pohledů na některé aspekty. Tento závěr byl potvrzen v pozdější empirické práci, která poukázala na to, že pro zodpovězení konkrétních zaměřených otázek může určitý pohled SysML, který byl automaticky generován z modelu OPM, poskytnout lepší odpověď rychleji. Existuje tedy zřejmá potenciální synergie kombinace výhod těchto dvou jazyků.

12.11 Shrnutí

- Aktivita diagramy jsou ilustrace pracovních postupů, které popisují tok mezi akcemi a jsou sémanticky nejbližší OPD.
- Akce je základní jednotkou v diagramu aktivit, ale pomocí symbolu rake může být rozvedena do celého diagramu aktivit sama o sobě, což poskytuje mechanismus zpřesnění podobný OPM při přiblížení.
- Toky v diagramu aktivit mohou být dvou druhů: tok *řízení* a tok *objektů*.
- *Uzly akcí typu přijmout, odeslat a časová událost* mají speciální syntaxi a sémantiku.
- *Uzly spojení a rozvětvení* se používají pro synchronizaci akcí.
- Šipky mezi aktivitou a objektem v diagramu aktivit mají sémantiku *toku*, zatímco v OPM mají sémantiku *transformace* – vytvoření, spotřebu nebo změnu stavu.
- Tok řízení v diagramu aktivit je dosažen pomocí *uzlů rozhodnutí*, které jsou uzly ve tvaru diamantu, ke kterým jsou často připojeny *poznámky o vstupu rozhodnutí*. V OPM je tok řízení částečně založen na booleovských objektech.
- Diagramy požadavků propojují typické nástroje pro správu požadavků a model systému.
- Parametrické diagramy používají *bloky vlastností omezení* k propojení systémových parametrů mezi sebou pomocí matematických výrazů. V OPM lze toho dosáhnout vyjádřením matematického vzorce jako výpočetního procesu.

- Požadavky a parametrické diagramy (stejně jako všechny ostatní typy diagramů SysML) lze v OPM modelovat bez speciálních symbolů.

12.12 Problémy

1. Nakreslete diagram aktivit OPD na [obr. 7.4](#).
2. Nakreslete diagram aktivit OPD na [obr. 8.6](#).
3. Pomocí symbolu hrábě spojte dva diagramy aktivit z úloh 1 a 2.
4. U každého z obou diagramů aktivit z úloh 1 a 2 označte *kontrolní* toky červeně a každý tok *objektů* zeleně.
5. Vysvětlete rozdíl v sémantice šipek mezi diagramem aktivit a OPD.
6. Pomocí níže uvedené specifikace nakreslete diagram požadavků.

Cestující přilétající na letiště odevzdá svá zavazadla letecké společnosti, se kterou letí. Systém pro manipulaci se zavazadly řídí přepravu zavazadel do cíle cesty cestujícího.

7. Porovnejte diagram požadavků z úlohy 6 s OPD na [obr. 2.5](#).
8. Nakreslete parametrický diagram pro vzorec $E = mc^2$.
9. Nakreslete OPD pro vzorec $E = mc^2$.
10. Porovnejte parametrický diagram z úlohy 8 s OPD z úlohy 9.

Kapitola 13

Aspekt dynamického systému

Každý den se setkáváme se systémy, které mají přirozenou tendenci se měnit. Příkladem může být počasí, burza nebo ekonomická situace.

Meinhardt (1995)

Systémy se v průběhu času mění. Důležitou motivací při vývoji OPM bylo dosažení potřebné rovnováhy v koncepčním modelu systému mezi strukturálními, statickými a procedurálními, dynamickými aspekty systému. Dynamický aspekt systému určuje, jak systém funguje, aby dosáhl své funkce, a doplňuje tak jeho statický aspekt. OPM je minimálně stejně orientován na procesy jako na objekty. Modely OPM skutečně sjednocují strukturu a chování v jednom soudržném referenčním rámci, přičemž čas je základním podkladovým konceptem. Tato kapitola se zabývá modelováním dynamického aspektu systému.

13.1 Změna a účinek

Procesy a dynamika systému jsou úzce spojeny s pojmem *změny*. Změna je tak známý a základní pojem, že jeho definování se jeví jako obtížné a zbytečné. Když však hovoříme o změně v OPM, musíme přesně specifikovat, co změna znamená.

Změna objektu je změna stavu tohoto objektu.

Konkrétněji řečeno, změna objektu znamená nahrazení jeho aktuálního stavu jiným stavem. Jedinou věcí, která může tuto změnu způsobit, je proces. Proces způsobuje změnu tím, že jako vstup přijímá objekt v určitém stavu – vstupním stavu – a jako výstup jej vydává v jiném stavu – výstupním stavu. Změna objektu tedy znamená *změnu stavu*, ve kterém se objekt nachází.

Objekty se stavem mohou být ovlivněny, tj. jejich stavy se mohou měnit. Tento mechanismus změny podtrhuje úzké, neoddělitelné propojení mezi objekty a procesy. Tuto změnu stavu nazýváme *účinkem* procesu na objekt.

Účinek je změna stavu objektu, kterou proces způsobuje svým výskytem.

Ačkoli se termíny „změna“ a „efekt“ zdají být téměř synonymní, existuje mezi nimi jemný rozdíl v použití. *Termín efekt* používáme k označení toho, co proces s objektem dělá, a *termín změna* k označení toho, co se s objektem stane v důsledku výskytu procesu. Později v této části upřesníme výše uvedenou definici efektu pomocí pojmů vstupní a výstupní vazby.

13.2 Existence a transformace

V oddíle 10.3.2 jsme definovali transformaci objektu procesem jako zobecnění konstrukce, účinku a spotřeby. *Konstrukce* je synonymem pro vytvoření, generování nebo produkci. *Účinek* je synonymem pro změnu nebo přepnutí a *spotřeba* je synonymem pro eliminaci, ukončení, zničení nebo destrukci. *Účinkem* procesu na objekt je změna stavu objektu z jednoho do druhého, ale objekt stále existuje a zachovává si identitu, kterou měl před procesem. Konstrukce a spotřeba mění samotnou existenci objektu, a proto jsou hlubšími transformacemi než účinek. Když říkáme, že proces *konstruuje* (vytváří, generuje, vytváří nebo vede k) objektu, máme na mysli, že objekt, který předtím neexistoval, prošel radikální transformací a je nyní novou, samostatnou entitou. Když říkáme, že proces *spotřebovává* (eliminuje nebo ničí) objekt, máme na mysli, že objekt, který předtím existoval, prošel radikální transformací, takže již v systému neexistuje.

13.2.1 Konstrukce a spotřeba: extrémní změny objektu

Když uvažujeme o existenci a neexistenci jako o stavech objektu, konstrukce a spotřeba se stávají extrémními případy změn stavu objektu, jak ukazuje

obr. 13.1 pomocí devíti OPD. Řádky v této matici 3×3 představují tři fáze transformace vývoje tří objektů: **konstruovaný objekt**, **existující objekt** a **spotřebovaný objekt**. Horizontální osa (tři sloupce) představuje druh transformace objektů, od konstruktivní na levé straně po destruktivní na pravé straně. V souladu s tím existují tři odpovídající procesy: **konstrukce** v levém sloupci, **změna** uprostřed a **spotřeba** vpravo.

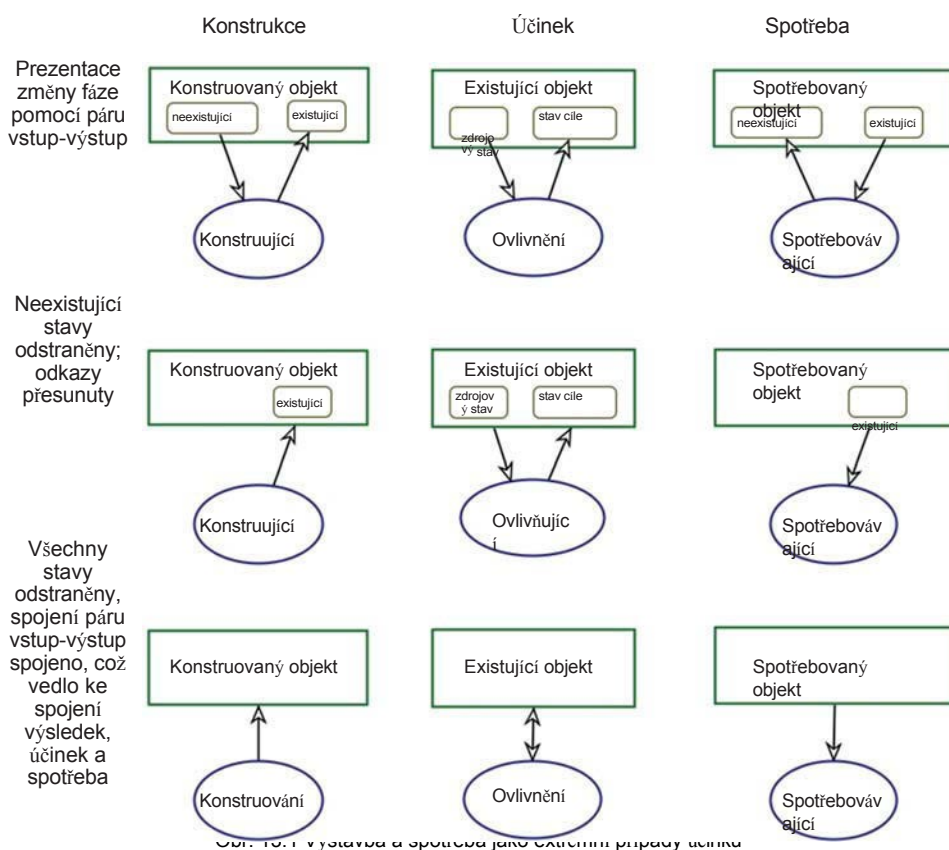
Vertikální osa (tři řádky) představuje úroveň detailů, od nejdetailnějšího v horním řádku po nejabstraktnější v dolním řádku. V horním řádku jsou vyjádřeny všechny stavy všech tří objektů a páry vstup-výstup pocházejí z těchto stavů a do těchto stavů směřují. Ve středním řádku, protože stav **neexistujícího konstruovaného objektu** a stav **neexistujícího spotřebovaného objektu** neexistují, jsou odstraněny spolu s odkazy, které je spojují. Také hrany zbývajících transformačních odkazů ve všech třech sloupcích byly přesunuty ze stavů do obrysu objektového pole. Jedná se o prozatímní fázi, jejímž cílem je ukázat vývoj odkazů. Nakonec jsou ve spodním řádku potlačeny všechny zbývající stavy a jsou zobrazeny tři konečné transformační vazby: **výsledek**, **účinek** a **spotřeba**. Vazba účinku (spodní střed) je abstrakcí *páru vstupních a výstupních vazeb* (horní střed), ve kterém jsou stavy potlačeny tak, že sémantika vazby účinku je změna stavů objektu z nějakého nespecifikovaného vstupního stavu do jiného nespecifikovaného výstupního stavu.

Použití **neexistujících** a **existujících** stavů objektu je užitečné, když chceme explicitně modelovat, že objekt je v systému přítomen nebo v něm chybí. Použili jsme jej například k modelování pojmů molekulární biologie, jako je odstranění faktoru nebo potlačení genu (Somekh et al. 2014).

13.2.2 Změna stavu nebo změna identity?

Během svého života mohou objekty procházet řadou transformací. Transformace objektu může podle naší definice nastat pouze tehdy, když na objekt působí nějaký proces. Tato transformace generuje, ovlivňuje nebo eliminuje objekt. Rozsah změny se může lišit od velmi malého po velmi velký. Pokud je změna

malá, jako například změna umístění objektu nebo jeho barvy, máme tendenci říkat, že objekt byl změněn z jednoho stavu do druhého, přičemž si zachoval svou identitu. S rostoucím rozsahem účinku roste i rozdíl mezi objektem před zahájením procesu a po jeho skončení. V určitém okamžiku se oba stávají koncepčně tak odlišné, že modelář má sklon považovat objekt vzniklý v důsledku procesu za nově vytvořený objekt. Objekt, který existoval před zahájením procesu, mohl být odstraněn nebo alespoň radikálně změněn. Jak ukážeme níže, otázka, zda objekt změnil pouze svůj stav nebo celou svou identitu, je podobná v přírodních i umělých systémech.

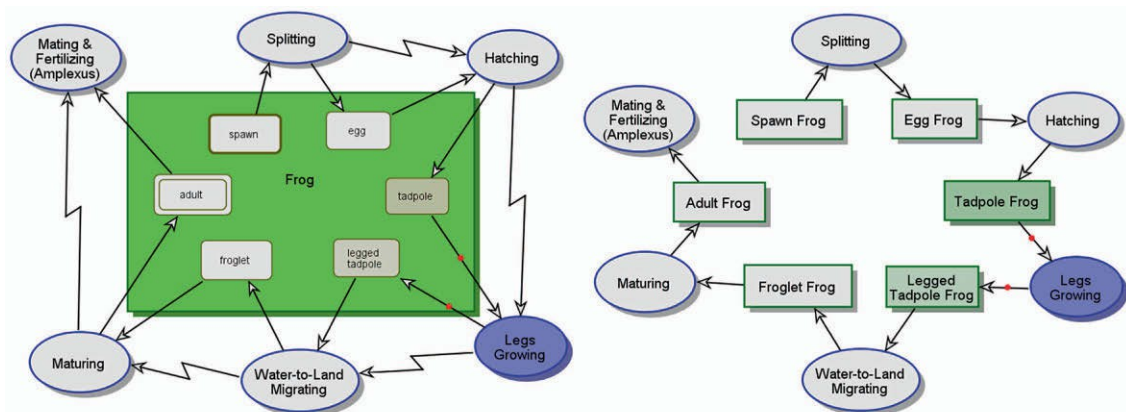


Obt. 10.1 Vytváření a spotřeba jako extrémní případy účinku

13.2.3 Transformace v živých organismech

V přírodě procházejí živé organismy pozoruhodnou škálou proměn. Některé z těchto proměn jsou považovány pouze za změnu stavu, zatímco jiné jsou považovány za změnu identity organismu. Proměna lvičete v dospělého lva je považována za změnu stavu lva z mladého na dospělého. Podobně je růst dítěte v dospělého považován za změnu stavu člověka.

Na druhé straně má bource morušový čtyři odlišné formy existence. Proměňuje se z vajíčka na larvu (červa nebo housenku) a pak na kuklu, larva prochází kompletní proměnou uvnitř ochranného kokonu nebo ztvrdlého pouzdra, až se stane motýlem, který zase klade vajíčka další generace bource morušového. Každá transformace vede k objektu, který se velmi liší od svého předchůdce tvarem a funkcí. Rozdíl je tak zásadní, že každá taková transformace se nazývá metamorfóza. Máme sklon vnímat každou reinkarnaci jako samostatný objekt, nikoli jako pouhou změnu stavu stejného objektu. Žába, stejně jako ostatní obojživelníci, se mění z jiker na vajíčka, z vajíček na pulce, z pulců na žabky a z žabek na dospělé žáby, což je podobný příklad jako u bource morušového.



Frog can be **spawn**, **egg**, **tadpole**, **legged tadpole**, **froglet**, or **adult**.
Frog is initially **spawn** and finally **Adult**.
Mating & Fertilizing (Amplexus) is physical.
Mating & Fertilizing (Amplexus) changes **Frog** from **adult** to **spawn**.
Mating & Fertilizing (Amplexus) invokes **Splitting**.
Splitting changes **Frog** from **spawn** to **egg**.
Splitting invokes **Hatching**.
Hatching changes **Frog** from **egg** to **tadpole**.
Hatching invokes **Legs Growing**.
Legs Growing changes **Frog** from **tadpole** to **legged tadpole**.
Legs Growing invokes **Water-to-Land Migrating**.
Maturing changes **Frog** from **froglet** to **adult**.
Maturing invokes **Mating & Fertilizing (Amplexus)**.
Water-to-Land Migrating changes **Frog** from **legged tadpole** to **froglet**.
Water-to-Land Migrating invokes **Maturing**.

Mating & Fertilizing (Amplexus) consumes **Adult Frog**.
Splitting consumes **Spawn Frog**.
Splitting yields **Egg Frog**.
Hatching consumes **Egg Frog**.
Hatching yields **Tadpole Frog**.
Legs Growing consumes **Tadpole Frog**.
Legs Growing yields **Legged Tadpole Frog**.
Maturing consumes **Froglet Frog**.
Maturing yields **Adult Frog**.
Water-to-Land Migrating consumes **Legged Tadpole Frog**.
Water-to-Land Migrating yields **Froglet Frog**.

Obr. 13.2 Dva souběžně simulované modely životního cyklu žáby. Vlevo: Změna stavu objektu. Vpravo: Změna identity objektu

Obrázek 13.2 ukazuje snímek obrazovky OPCAT se dvěma modely OPM, které jsou simulovány současně. Model vlevo ukazuje **žabu** jako jeden objekt se šesti stavy: **jikra**, **vajíčko**, **pulc**, **pulc s nohama**, **žabka** a **dospělý jedinec**. Díky odkazům na vyvolání, jakmile je vyvolán externí aktivací **rozdělení**, tento model dokončí celý cyklus od **jikry** po **dospělého jedince**.

Model vpravo ukazuje šest různých bezstavových objektů různých inkarnací žáby: **Spawn Frog**, **Egg Frog**, **Tadpole Frog**, **Legged Tadpole Frog**, **Froglet Frog** a **Adult Frog** (zde je každý proces

musí být vyvolán samostatně; tomu se lze vyhnout, pokud nahradíme každý spotřební odkaz, např. odkaz z **Egg Frog** do **Hatching**, odkazem na spotřebu události). V tomto bodě simulace je aktivní proces **Legs Growing** (jak ukazuje tmavá barva a body podél šipek). Příslušná věta OPL pro model vlevo je:

Růst nohou mění žábu z pulce na pulce s nohama.

Příslušné věty OPL pro model vpravo jsou:

Růst nohou spotřebovává žabí pulce. Růst nohou produkuje žabí pulce s nohama.

Příklady žaby a bource morušového lze pohodlně považovat za změny objektu, i když geneticky se jedná o stejný organismus, protože různé podoby těchto tvorů se od sebe výrazně liší jak vzhledem, tak chováním. Příklady člověka a lva lze naopak přirozeněji modelovat jako změnu stavu objektu.

13.2.4 Transformace umělých objektů

Situace s transformacemi umělých objektů je podobná jako u přírodních objektů: pokud je změna zásadní, objekty mění svou identitu, jinak stejný objekt pouze mění svůj stav. Co je „zásadní“ transformace, je subjektivní a závisí na kontextu. Uvažujme například dva procesy z oblasti výroby: **lisování** a **testování**. **Formování** působí na objekt **Surovina** (např. plast) a přeměňuje jej na jiný objekt, který nazýváme **Produkt**. Identita **Suroviny** se v důsledku procesu **Formování** změnila do té míry, že musíme výsledek procesu označit jiným názvem. Objekt **Surovina** byl tedy odstraněn nebo spotřebován, zatímco nový objekt, nazývaný **Produkt**, byl vytvořen (nebo zkonstruován, nebo vygenerován). Vztah mezi těmito dvěma objekty můžeme modelovat například přidáním označené strukturální relace od **produktu** k **surovině** s označením „je vyroben z“, což povede k větě OPL **Produkt je vyroben ze suroviny**.

Předpokládejme, že **produkt** nyní prochází procesem **testování**, při kterém se měří jeho pevnost v tahu. Pokud test dopadne úspěšně, **produkt** je **schválen**, v opačném případě je **zamítnut**. Na rozdíl od procesu **formování**, který změnil identitu zpracovaného objektu ze **suroviny** na **produkt**, **testování** nemění **produkt** do té míry, že bychom mohli říci, že ztratil svou identitu. Jediným účinkem procesu **testování** je změna stavu **produktu** (z **netestovaného** na **testovaný**). I když existuje rozdíl mezi **produktem** před a po **testování** (protože po testu máme informace o pevnosti produktu, které jsme předtím neměli), tento rozdíl není natolik zásadní, aby odůvodňoval změnu identity. Způsobuje však změnu stavu. Transformaci lze tedy považovat za obecný pojem, který zahrnuje vytvoření, účinek (změnu stavu) a odstranění objektu. Podrobněji se tím budeme zabývat při diskusi o systémové dynamice.

Kritériem pro to, zda proces mění stav objektu nebo identitu objektu, je to, zda je možné a smysluplné vytvořit atribut, který daný objekt vykazuje se stejnými hodnotami jako stavy objektu. V našem případě by atributem **produktu** byl objekt „stav testování“ se stejnými hodnotami, **před testováním** a **po testování**. Pokud je to možné, jako je tomu v tomto případě, pak se změna týká pouze stavu objektu, ale ne jeho identity. Pokud tomu tak není, jako je tomu v příkladu **suroviny** a **produktu**, změna se týká identity objektu.

Zobecněním přírodních a umělých příkladů, když změna není zásadní nebo drastická, máme sklon si myslet, že objekt pouze mění svůj stav, zatímco si zachovává svou identitu. Když je transformace extrémní, dochází ke změně identity objektu. Stejně jako v podobných situacích není hranice mezi „drastickým“ a „nedrastickým“ jasně definována. Při analýze stejného systému mohou různí modeláři poskytnout různé názory na to, zda by konkrétní objekt měl ztratit svou identitu a stát se novým objektem. Ve skutečnosti uvidíme příklady, kdy má smysl modelovat změny v objektech buď jako změnu jejich stavu (nebo hodnoty atributu), nebo jako změnu jejich identity, a obě verze by byly přijatelné.

13.3 Procedurální odkazy

Procedurální odkazy jsou nepostradatelným „spojovacím článkem“ mezi procesy a objekty nebo jejich stavy. Zajišťují integraci struktury a chování systému v rámci jediného modelu. Procedurální odkazy mají zásadní význam.

Procedurální odkaz je odkaz mezi procesem a objektem nebo jeho stavem, případně mezi dvěma procesy.

Většina procedurálních odkazů je mezi procesem a objektem nebo jeho stavem. Jediné tři druhy procedurálních odkazů mezi dvěma procesy jsou odkaz vyvolání a odkazy výjimek přesčasů a podčasů, které jsou popsány v kapitole 22. Jak je popsáno v oddíle 10.10.3, odkaz vyvolání může nahradit přechodný, krátkodobý fyzický nebo informační objekt, který zdrojový proces vytvoří k zahájení cílového procesu, který okamžitě spotřebuje přechodný objekt. To platí také pro výjimekové vazby (kde objektem může být zpráva). Proto často vynecháváme poslední část definice procedurální vazby a říkáme, že procedurální vazba je vazba mezi procesem a objektem nebo jeho stavem.

Integrace struktury a chování, kterou procedurální odkazy poskytují, je jednou z nejdůležitějších vlastností OPM. Tato integrace v rámci jediného modelu v první řadě eliminuje problém inherentní mnohosti diagramů (Peleg a Dori 2000), který je charakteristický pro objektově orientované metody, jako jsou UML a SysML, jejichž ontologie zdaleka není minimální.

13.3.1 Transformační versus umožňující procedurální vazby

Definice procesu OPM vyžaduje, aby proces transformoval alespoň jeden objekt. Kromě transformovaných objektů může proces také vyžadovat jeden nebo více objektů, které tento proces *umožňují*, ale nejsou jím transformovány. Z pohledu daného procesu tedy OPM rozlišuje mezi dvěma typy objektů: transformovaným objektem – objektem, který proces transformuje (generuje, ovlivňuje nebo spotřebovává), a umožňujícím objektem – objektem, který proces umožňuje, ale není tímto procesem transformován. V souladu s tím existují dva typy procedurálních vazeb: *transformující vazby* a *umožňující vazby*.

13.3.2 Transformované objekty

Definovali jsme, že transformát procesu P je objekt B , který P transformuje v důsledku svého výskytu. Transformací může být konstrukce, účinek (změna stavu) nebo spotřeba. Transformát je *role*, kterou objekt B přebírá ve vztahu k určitému procesu P . B tedy může být transformátem ve vztahu

nějakého procesu P_1 a aktivátorem ve vztahu k jinému procesu P_2 . Transformace může být jedním ze tří typů definovaných níže.

***Spotřebitel** procesu P je transformátorem P , který P spotřebovává v důsledku výskytu P .*

***Výsledek** procesu P je transformovaný objekt P , který P vytvoří v důsledku výskytu P .*

***Osoba ovlivněná procesem** P je transformovaná osoba P , na kterou má P vliv v důsledku výskytu P .*

V dolní části [obr. 13.1](#) jsou **spotřebovaný objekt**, **vytvořený objekt** a **existující objekt** spotřebovaným, výsledkem a ovlivněným objektem. Spotřebovaný objekt lze považovat za *vstup* do procesu, protože proces jej spotřebovává, a výsledek za *výstup* procesu, protože proces jej vytváří. Ovlivněný objekt je jak *vstupem*, tak *výstupem*: proces jej přijímá ve vstupním stavu a vydává ve výstupním stavu.

Tyto analogie rozhodně platí pro fyzické objekty. Informační objekt však může sloužit jako vstup do procesu, aniž by byl spotřebován jako celek nebo jeho část. Předpokládejme například, že na [obr. 13.3](#) je **soubor** v databázi vymazán procesem **Erasing**, pak je **soubor** spotřebovaným objektem i vstupem do **Erasing**. Pokud je **soubor** vytvořen procesem **Creating**, pak je **soubor** výsledkem i výstupem procesu **Creating**. Pokud je **soubor** upraven (tak, že jsou do něj přidána, změněna nebo odstraněna data) procesem **Editing**, pak je **soubor** ovlivněným prvkem **procesu Editing**, stejně jako vstupem i výstupem.

Pokud je **soubor** čten prostřednictvím procesu **čtení**, pak je **soubor** vstupem do tohoto procesu, ale není nijak spotřebován ani změněn – slouží jako nástroj pro **čtení**, jak popíšeme níže. Fyzický objekt, jako je například zdroj, který je vstupem do procesu, je tímto procesem alespoň částečně spotřebován. Existuje mnoho fyzických objektů, jako je například **kladivo** – nástroj pro proces **zatloukání hřebíků**, které jsou nástroji a v podstatě se procesem, který umožňují, nemění.

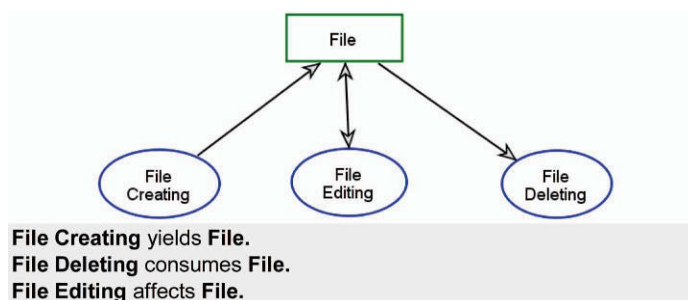
13.4 Transformační vazby

Transformační vazby jsou jednosměrné nebo obousměrné šipky spojující transformovaný objekt s procesem, který jej transformuje.

***Transformační spojení** je procedurální spojení, které spojuje proces s transformovaným objektem tohoto procesu.*

[Obrázek 13.3](#) ukazuje vazby mezi **souborem** na jedné straně a **vytvářením souboru**, **úpravou souboru** a **mazáním souboru** na straně druhé jako příklady vazeb výsledku, účinku a spotřeby. Jedná se o specializace transformační vazby, jak je definováno níže.

Na [obr. 13.4](#) je **Zpracování** propojeno se třemi specializacemi **transformovaného objektu**: **Spotřebitel**, **Ovlivněný** a **výsledkem** prostřednictvím odpovídajících transformačních vazeb – vazeb spotřeby, účinku a výsledku.

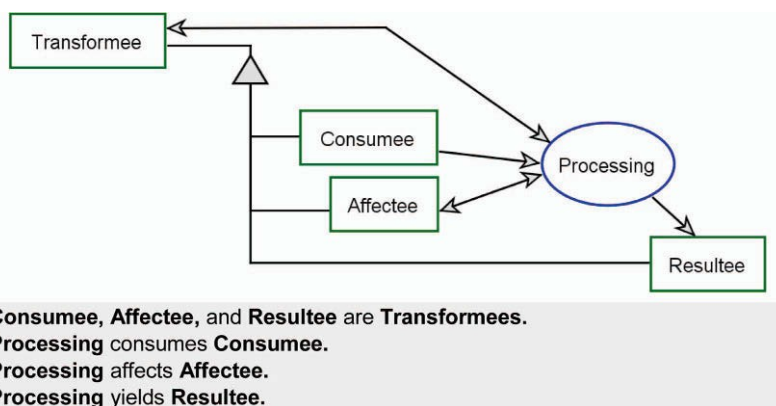


Obr. 13.3 Vztah mezi výsledkem, účinkem a spotřebou mezi souborem a vytvářením souboru, úpravou souboru a mazáním souboru

***Vztah výsledku** je jednosměrný transformační vztah mezi procesem a výsledkem, který tento proces vytváří.*

***Vazba účinku** je obousměrná transformační vazba, která spojuje proces s objektem, na který má tento proces vliv.*

***Spotřební vazba** je jednosměrná transformační vazba od spotřebitele k procesu, který jej spotřebovává.*



Obr. 13.4 Zpracování propojilo tři typy transformovaných prvků pomocí odpovídajících transformačních odkazů. Enablers

13.4.1 Načasování spotřeby a výsledku

Existence spotřebitele je předpokladem nebo součástí předpokladu pro aktivaci procesu. Pokud v době zahájení procesu neexistuje požadovaný počet instancí spotřebitele (obvykle 1), pak aktivace procesu počká, až tento počet instancí spotřebitele existuje. Spotřeba instance spotřebovaného objektu je okamžitá po aktivaci procesu, pokud model nevyjadřuje spotřebu objektu v čase, v takovém případě se použije míra spotřeby, specializace míry transformace, jak je vysvětleno níže.

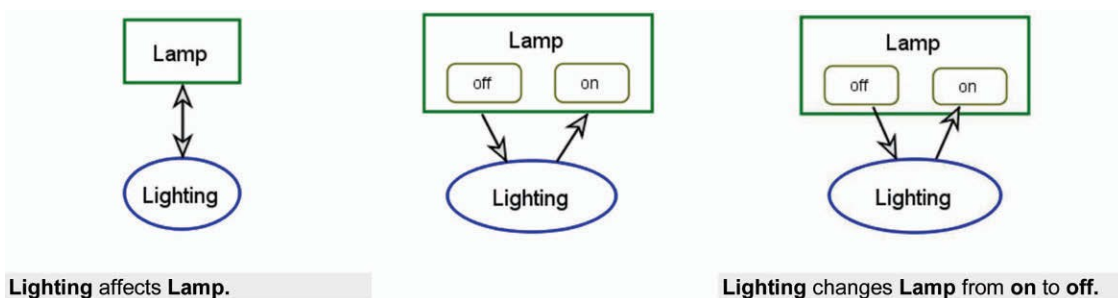


Obr. 13.5 Spotřeba a načasování výsledku: **Ocelová tyč** se spotřebuje a zmizí, jakmile začne **obrábění**. **Hřídel** se vytvoří až po skončení **obrábění**.

Na obr. 13.5 je **ocelová tyč** spotřebním materiálem pro proces **obrábění**, který generuje výsledný **hřídel**. Jakmile je **obrábění** zahájeno, spotřebuje **ocelovou tyč**. **Hřídel** je však považován za vytvořený až po ukončení **obrábění**. Během procesu již **ocelová tyč** neexistuje, ale ani **hřídel**.

13.4.2 Vývoj efektové vazby

Výslovné vyjádření stavů objektu v diagramu často vede k příliš podrobnému, přeplněnému nebo nepřehlednému OPD, které je obtížné číst. Jedná se o projev kompromisu mezi srozumitelností a přehledností: tyto dvě žádoucí vlastnosti komplexních systémových modelů jsou v neustálém konfliktu.



Obr. 13.6 Vývoj efektového propojení

Obrázek 13.6 ukazuje potlačení stavů a vývoj efektového propojení, podobně jako prostřední sloupec na **obrázku 13.1**. Ve středním OPD se vstupní a výstupní propojení, která jsou na pravém OPD připojena ke stavovým obdélníkům, přesouvají na hranici rámečku objektu **Lamp**. Nyní propojují proces a objekt přímo, a to od a k samotnému objektu, nikoli od a k jeho stavům. Toto prozatímní znázornění není v OPM platné. Aby se snížila grafická nepřehlednost, byly vstupní a výstupní vazby, označené dvěma protilehlými jednosměrnými šipkami, překryty spojením do jedné obousměrné šipky, čímž vznikl symbol vazby účinku. Nakonec byly vlevo potlačeny stavy **Lamp**, protože již nejsou důležité, jelikož k nim nejsou připojeny žádné vazby.

13.5 Enablers

Předpokládejme, že se chcete přestěhovat ze svého bytu do bytu v jiném městě. K tomu potřebujete stěhovací vůz, který si pronajmete od stěhovací společnosti. Vůz vrátíte na stejné místo, kde jste si ho vyzvedli, a se stejným množstvím benzínu, jaké jste měli při převzetí. Pokud tedy pomineme amortizaci vozu, nic se v něm nezměnilo. Bez vozu byste však nemohli stěhování provést. Říkáme, že vůz je aktivátorem procesu stěhování. Navíc, protože některé vaše nábytky jsou velmi těžké, potřebujete přítele jako druhého aktivátora procesu stěhování.

stěhování. Říkáme, že **nákladní vůz** je *podpůrným* prostředkem procesu **stěhování**. Navíc, protože některé vaše nábytky jsou velmi těžké, potřebujete **přítele** jako druhý podpůrný prostředek procesu **stěhování**.

Enabler procesu je objekt, který umožňuje provedení procesu. Jeho přítomnost je nutná po celou dobu trvání procesu, ale po skončení procesu existuje enabler ve stejném stavu, v jakém byl na začátku procesu. Jinými slovy, enabler procesu je objekt, který musí být přítomen po celou dobu trvání procesu, aby tento proces mohl proběhnout a úspěšně skončit, ale není transformován v důsledku výskytu procesu.

***Aktivátor E** procesu **P** je objekt, který musí existovat a být k dispozici, aby mohl **P** začít, a musí zůstat přítomen po celou dobu trvání **P**, aby **P** mohl normálně skončit, přičemž **E** zůstane nakonec nedotčen.*

Enabler může během procesu projít změnou stavu, ale jak uvádí definice enableru, po skončení procesu, který enabler spustil, se enabler vrátí do stejného stavu, ve kterém byl na začátku. Například enabler **Oven (trouba)** na [obr. 13.7](#) změní stav z **vypnutého** na **zapnutý** na začátku procesu **Baking (pečení)** a těsně před koncem procesu **Baking** se vrátí **ze zapnutého stavu zpět do vypnutého stavu**.

Jak ukazuje příklad **Moving**, některé aktivátory jsou lidské, zatímco jiné jsou neživé. Aktivátor má tedy dvě specializace: agent nebo nástroj, jak je definováno níže.

13.5.1 Agent: lidský aktivátor

Termín *agent* je vyhrazen pro lidského aktivátora.

***Agent** je aktivátor, který je člověkem nebo skupinou lidí.*

Agent je inteligentní aktivátor, který může řídit proces, který aktivuje, pomocí zdravého rozumu nebo úvah zaměřených na cíl, což znamená, že musí sestávat z jednoho nebo více lidí. Obvykle se jedná o jednu osobu – uživatele nebo příjemce systému. Agentem může být také organizace nebo jednotka v rámci umělé organizace, jako je oddělení, městská rada, vláda, skupina, tým atd.

Pojem agent je důležitý, protože umožňuje modelovat „člověka v smyčce“, tj. jak lidé interagují se systémem. To je pro návrháře systému jasným ukazatelem bodů interakce se systémem, kde je třeba vyvinout *lidské rozhraní*. Hierarchie procesů, do nichž je agent zapojen, navíc poskytuje vynikající vodítko pro uspořádání přátelského grafického uživatelského rozhraní a vytvoření takového rozhraní lze na základě tohoto modelu do jisté míry dokonce automatizovat.

Ve světě softwaru a vestavěných systémů se roboti často označují jako agenti a softwaroví agenti jsou běžní na internetu, kde využívají rozvíjející se technologie agentů. V OPM, které je zaměřeno na modelování všech druhů systémů, včetně komplexních sociotechnických systémů a systémů, kde jsou uživateli a příjemci lidé, jsou lidé (jako jednotlivci nebo skupiny) privilegováni a odlišováni od všech ostatních neživých činitelů, takže termín *agent* je vyhrazen pouze pro lidi. ⁽¹⁾ To umožňuje soustředit pozornost systémových architektů a designérů na péči o bezpečnost lidí a jejich speciální potřeby a přání.

¹ Robot lze stále nazývat *agentem s vestavěným softwarem* a programy působící na internetu jménem lidí lze stále nazývat *softwarovými agenty*. *Agent* bez jakékoli kvalifikace je vyhrazen pro jednotlivce nebo skupiny lidí.

při interakci s ostatními částmi systému – použitelnost systému a zkušenosti a potěšení uživatelů z používání dobře navrženého, uživatelsky přívětivého a přizpůsobivého systému.

Odkaz na agenta je do jisté míry analogický s aktérem – „panáčkem“ v diagramu případů použití UML nebo SysML. V OPM však není potřeba žádný samostatný druh diagramu, protože modelování uživatele je začleněno do jediného modelu OPM. Případy použití v notaci SysML lze automaticky extrahovat z modelu OPM, stejně jako jiné modely SysML (Grobshtein a Dori 2011).

Ne každý člověk nebo organizace je nutně pouze agentem. Například pokud se **student** věnuje **studiu**, změní se jeho atribut **úroveň znalostí**, řekněme z **povrchní** na **hlubokou**. V tomto případě není **student** pouze agentem, ale také transformovaným subjektem. Podobně, pokud oddělení v podniku prochází reorganizací obchodních procesů, mění se v důsledku tohoto procesu jeho struktura a/nebo chování, takže kromě toho, že je agentem, je také transformovaným subjektem.

Princip jedinečnosti procedurálního propojení OPM stanoví, že na jakékoli úrovni detailu může být objekt a proces propojen maximálně jedním procedurálním propojením. Sémantická síla a priorita propojení jsou definovány a podrobně popsány v kapitole 21. Zde pouze poznamenáváme, že transformační propojení jsou sémanticky silnější než umožňující propojení, protože transformační propojení označují vytvoření, spotřebu nebo změnu propojeného objektu, zatímco umožňující propojení označují pouze umožnění. Transformační vazba má přednost před umožňující vazbou, jak je znázorněno na obr. 21.15, proto pokud musíme volit mezi vazbou agenta a vazbou účinku, jako v příkladech výše, zvolíme vazbu účinku.

13.5.2 Nástroj: Nehumánní aktivátor

Nástrojem procesu je jakýkoli nehumánní, fyzický nebo inženýrský objekt, který se v důsledku provedení procesu nemění.

Nástroj je nehumánní aktivátor.

Příklady nástrojů zahrnují stroje, nástroje, počítače, roboty, ovladače, hardware, software, dokumenty, objednávky, recepty, algoritmy, předpisy, soubory, příkazy, informace a data. Algoritmy a recepty jsou typickými příklady informačních nástrojů, které lze používat opakovaně, ideálně bez opotřebení (v praxi můžeme být svědky také „amortizace softwaru“...).

Fyzické nástroje se při umožňování procesu obvykle do určité míry mění. Zejména se mohou opotřebovat nebo degradovat, protože jsou používány jako prostředky umožňující proces. Z pohledu vyvíjeného systému by však takové objekty byly stále považovány za nástroje, protože tyto změny nejsou buď dostatečně významné, aby byly zohledněny, nebo jsou mimo rozsah systému.

V jiných případech je třeba zohlednit opotřebení. Například při vývoji **výrobního systému** může být od systémového architekta vyžadováno, aby zohlednil **údržbu stroje**, který se opotřebovává v důsledku procesu **řezání kovů**, který umožňuje. V tomto případě by **stroji** neměla být přiřazena role nástroje. Bude spíše modelován jako ovlivněný subjekt. Atributem **stroje**, který se mění v důsledku jeho provozu, může být například jeho **úroveň amortizace** nebo počet provozních hodin od poslední generální opravy. Tento proces **opotřebení stroje** budeme muset zohlednit, pokud náš systém zahrnuje aspekt údržby **stroje**. Rozlišení mezi dvěma typy aktivátorů – agenty a nástroji – v OPD je možné díky jejich propojení s procesem, který aktivují, prostřednictvím různých aktivujících odkazů, které jsou definovány dále.

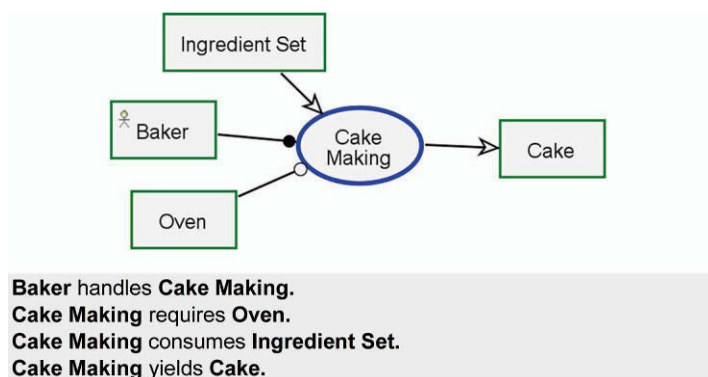
13.5.3 Aktivační vazby: vazby agentů a nástrojů

Enablers jsou propojeny s procesy prostřednictvím enablujících odkazů.

Aktivační propojení je procedurální propojení, které spojuje proces s aktivátorem tohoto procesu.

Propojení agenta je propojení, které spojuje proces s agentem tohoto procesu.

Instrumentální propojení je procedurální propojení, které spojuje proces s jeho aktivátorem.



Obr. 13.7 Příklad aktivujících vazeb: vazba agenta od pekaře a vazba nástroje od pece

Graficky, jak ukazuje [obr. 13.7](#), je aktivující odkaz „lízátko“, čára vedoucí od aktivátoru (**pekař**) k procesu (**výroba dortu**), který aktivuje, a končí kruhem dotýkajícím se strany procesu. Pokud je aktivátorem člověk nebo skupina lidí, je aktivující odkaz odkazem agenta, označeným jako „černá lízátka“, tj. jeho koncový kruh je vyplněn (černě).

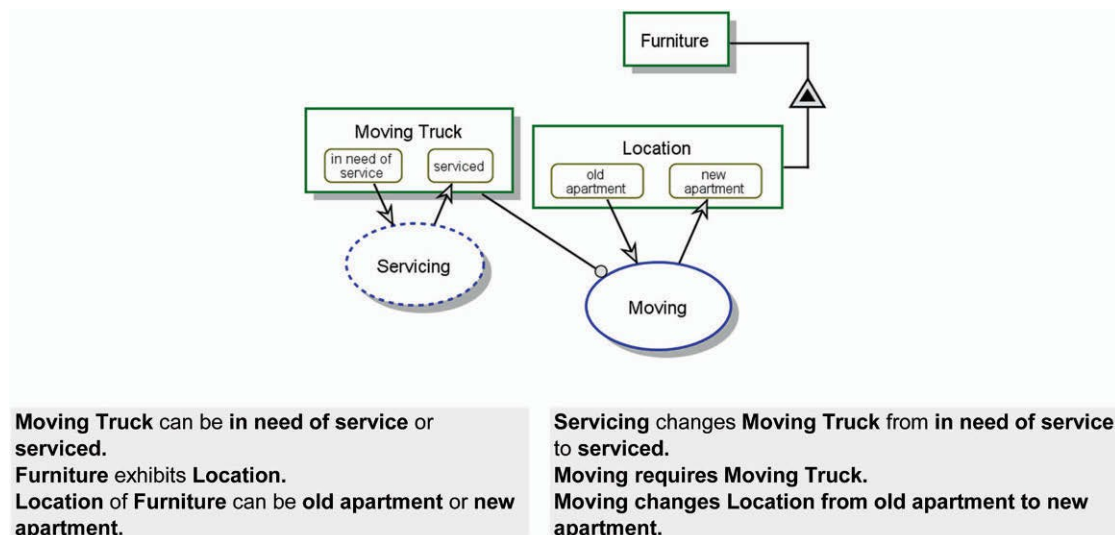
Rozdíl mezi lidským a nelidským aktivátorem je důležitý, protože pro interakci lidí se systémem je třeba navrhnout speciální rozhraní. Proto lze do levého horního rohu symbolu objektu agenta přidat volitelnou figurku, jak je znázorněno na [obr. 13.7](#). Tato volitelná figurka je obzvláště užitečná, když je člověk v modelu ovlivněnou osobou, tj. je ovlivněn procesem, ke kterému je propojen, v takovém případě musíme použít efektové propojení namísto propojení agenta. V tomto případě figurka zachovává informaci, že je do procesu zapojen člověk.

Pokud je aktivátor nástrojem, je aktivující spojení „bílá lízátka“, tj. jeho koncový kruh je prázdný (bílý). Dvě věty OPL spojené s těmito spojeními jsou:

Agent zpracovává zpracování.
Zpracování vyžaduje nástroj.

Syntaxe OPL první věty (agenta) je navržena tak, že agent se objeví jako první, následuje vyhrazená fráze OPL zpracovává a poté název procesu. U věty o nástroji je syntaxe OPL taková, že nejprve se objeví název procesu, za ním následuje vyhrazená fráze OPL vyžaduje a za ní název nástroje. Tento rozdíl v frázích OPL i v pořadí aktivátorů ve větách podtrhuje, že agenti jsou jako lidé důležitější než nástroje.

Všechny aktivátory procesu musí být přítomny po celou dobu provádění procesu, který aktivují. Například na obr. 13.7 musí být agent **Baker** i nástroj **Oven** přítomny po celou dobu procesu **pečení dortu**.



Obr. 13.8 Stejný objekt hrající roli nástroje a ovlivněného subjektu: **Stěhovací vůz** je nástrojem **stěhování** a ovlivněným subjektem **servisu**.

13.5.4 Enabler versus Affectee

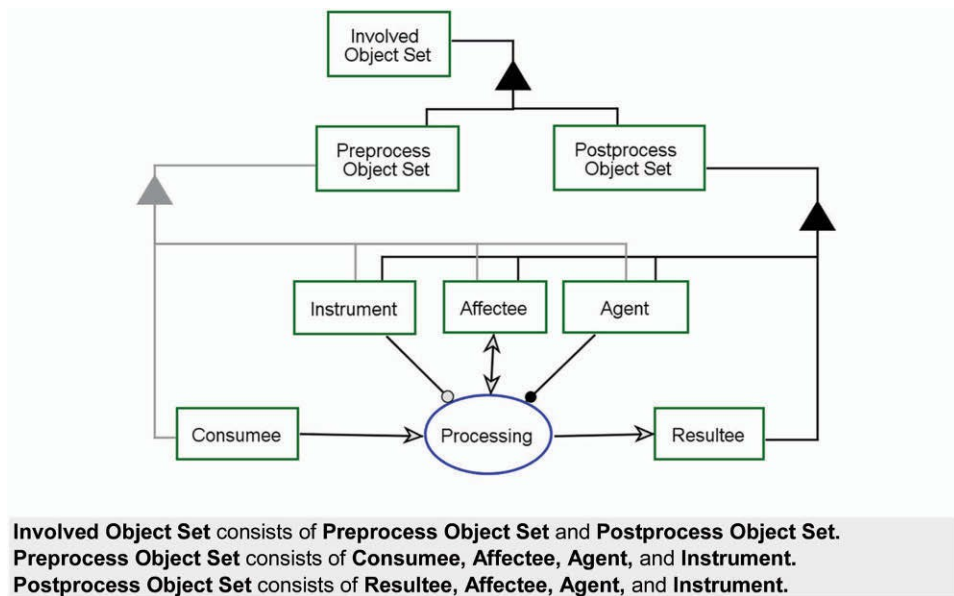
Enabler a affectee jsou možné role, které objekt hraje v souvislosti s některými procesy. Stejný objekt může být enablerem pro jeden proces, ale ne pro jiný, nebo může být enablerem pro jeden proces a affectee pro jiný. Například (environmentální) proces **Servicing** na obr. 13.8, který stěhovací společnost pravidelně aplikuje na svůj **Moving Truck**, mění stav **Moving Truck** z „vyžaduje servis“ na „proveden servis“, proto je **Moving Truck** ovlivňovaným objektem **procesu Servicing**. V souvislosti se (systémovým) procesem **Stěhování** je však **Stěhovací vůz** aktivátorem – nástrojem pro **Stěhování**, zatímco **Umístění nábytku** je ovlivňovaným objektem, protože **Stěhování** mění hodnotu (stav atributu) atributu **Umístění nábytku** ze **starého bytu** na **nový byt**.

13.6 Sady objektů před zpracováním a po zpracování

Připomeňme, že sada zapojených objektů je sjednocením sady objektů před zpracováním a sady objektů po zpracování. Jak ukazuje obr. 13.9, pokud sada zapojených objektů obsahuje aktivátory (agenty a/nebo nástroje), jsou společné pro sady objektů před zpracováním a po zpracování, protože jejich přítomnost je vyžadována po celou dobu trvání procesu, který aktivují. Každý proces má svou vlastní sadu zapojených objektů, sadu objektů před zpracováním a sadu objektů po zpracování, přičemž každá z nich může obsahovat libovolný počet objektů.

Ovlivněné objekty jsou také společné pro sady objektů před zpracováním a po zpracování, protože existovaly před zahájením ovlivňujícího procesu a zůstávají existující i po skončení tohoto procesu. Spotřebované objekty zmizí, takže

patří pouze do sady objektů před zpracováním, zatímco výsledné objekty jsou vytvořeny, takže patří pouze do sady objektů po zpracování.



Obr. 13.9 Sada zapojených objektů rozdělená na sadu objektů před zpracováním a sadu objektů po zpracování

Sada objektů před zpracováním a **sada objektů po zpracování** nemusí být nutně disjunktní – mohou se překrývat. Ve skutečnosti jsou na obr. 13.9 překrývajícími se členy dva aktivátory – **Agent** a **Nástroj** – a jeden transformovaný objekt – **Ovlivněný objekt**. **Agent** a **nástroj** mohou patřit do obou sad objektů, protože podle své definice jsou aktivátory a jsou vyžadovány v průběhu celého procesu (a neměly by se měnit v důsledku výskytu procesu, který aktivují). **Ovlivněný subjekt** patří do sady objektů před zpracováním i do sady objektů po zpracování, protože po skončení procesu nadále existuje, i když v jiném stavu. **Spotřebitel** je jediným zapojeným objektem, který není v **sadě objektů po zpracování**, protože ho proces **zpracování** spotřeboval, takže po ukončení **zpracování** neexistuje. Antisymetrickým způsobem je **výsledek** jediným zapojeným objektem, který není v **sadě objektů před zpracováním**, protože ho **proces zpracování** vygeneroval, takže před začátkem **zpracování** neexistoval. Procedurální vazby jsou shrnuty v tabulce 13.1.

13.7 Státem specifikované procedurální vazby

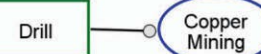
Často se stává, že v našem modelu chceme specifikovat nejen to, že objekt je transformován nebo že umožňuje proces, ale také to, v jakém stavu musí být aktivátor, aby mohl proces umožnit. Můžeme také chtít specifikovat nejen objekt, který proces generuje, ale také konkrétní stav, ve kterém je tento objekt generován v důsledku výskytu procesu. Stejně tak můžeme chtít

specifikovat nejen to, jaký objekt proces spotřebovává, ale také konkrétní stav, ve kterém musí být objekt, aby jej proces mohl spotřebovat. To umožňují procedurální vazby specifikované stavem.

Procedurální odkaz specifikovaný stavem je procedurální odkaz, který spojuje proces se stavem objektu.

Pro každé procedurální propojení existuje verze specifikovaná stavem. Procedurální propojení specifikovaná stavem se liší od svých verzí nespecifikovaných stavem v tom, že místo propojení (transformujícího nebo aktivovaného) procesu s příslušným objektem (transformovaným nebo aktivujícím) propojují proces s jedním ze stavů příslušného objektu. Procedurální propojení specifikovaná stavem jsou tedy propracovanějšími verzemi svých běžných procedurálních protějšků.

Tabulka 13.1 Procedurální odkazy, jejich sémantika, symboly, zdroj a cíl

Link Type	Name	Semantics	Sample OPD & OPL	Source	Destination
Transforming link	Consumption	The process consumes the object.	 Eating consumes Food.	Consume	Process
	Result	The process generates the object.	 Mining yields Copper.	Process	Result
	Effect	The process affects the object by changing it from one unspecified state to another.	 Purifying affects Copper.	Object and Process are both source and destination.	
Enabling link	Agent	The agent—a human—handles the process and is required to be there throughout the process, but it is not transformed.	 Miner handles Copper Mining.	Agent	Process
	Instrument	The process requires the instrument—a non-human object—throughout the process, but not transformed.	 Copper Mining requires Drill.	Instrument	Process

13.8 Stavově specifikované aktivační vazby

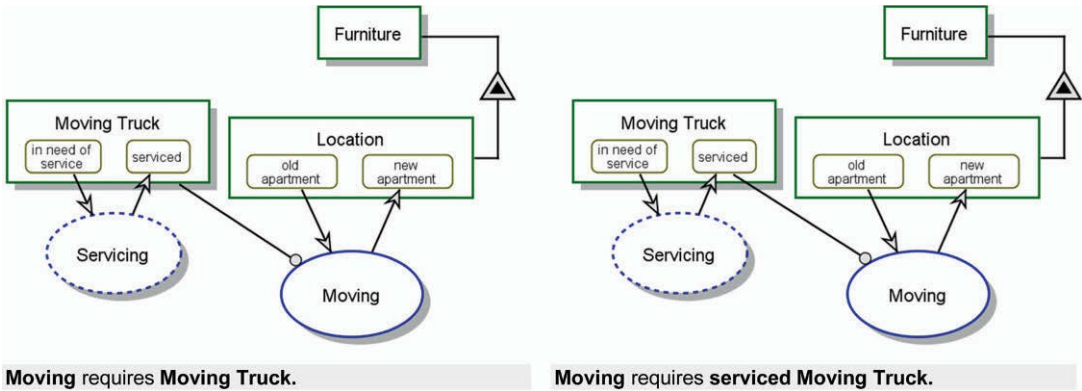
Státem specifikované aktivační odkazy – odkaz agenta a odkaz nástroje – jsou definovány následovně.

Stavově specifikovaný odkaz agenta je odkaz agenta, který pochází z konkrétního stavu s agenta **G** pro zpracování **P** a označuje, že aby **G** mohl zpracovat **P**, musí být **G** po celou dobu trvání **P** ve stavu **s**.

Stejně jako jeho protějšky, odkaz na spotřebu specifikovaný stavem a odkaz na výsledek, odkaz na nástroj specifikovaný stavem pochází z konkrétního stavu a končí v procesu. Sémantika tohoto propojení spočívá v tom, že proces je povolen pouze v případě, že objekt existuje a nachází se ve stavu, ze kterého propojení pochází. To je v kontrastu s „běžným“ propojením nástroje, které pochází z povolujícího nástroje, ale ne z konkrétního stavu tohoto nástroje. Například **pilot** musí být **střízlivý**, aby se mohl kvalifikovat jako agent pro proces **létání letadlem**. V OPL: **Střízlivý pilot** zpracovává **létání**.

*Státem specifikovaný odkaz na nástroj je odkaz na nástroj, který pochází z konkrétního stavu s nástroje **I** pro zpracování **P**, což znamená, že aby bylo možné provést **P**, musí být **I** po celou dobu trvání **P** ve stavu **s**.*

Rozdíl mezi těmito dvěma typy propojení nástrojů je znázorněn na [obr. 13.10](#), kde na levé straně je objekt **Moving Truck** nástrojem pro **Moving**, což znamená, že stav, ve kterém se tento **Moving Truck** nachází, není důležitý. Na pravé straně propojení nástrojů pochází ze stavu **serviced** (servisovaný) **Moving Truck**, což znamená, že pouze pokud je **Moving Truck** servisovaný, může dojít k **Moving**.



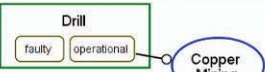
Obr. 13.10 Propojení nástrojů vs. propojení nástrojů specifikované státem: Vlevo: Propojení nástrojů – **Moving Truck** je nástroj **stěhování**. Vpravo: Odkaz na nástroj specifikovaný stavem – opravený **stěhovací vůz** je nástrojem **stěhování**

[Tabulka 13.2](#) shrnuje sémantiku, symboly, zdroj a cíl dvou odkazů umožňujících specifikovaných stavem.

13.9 Stavem specifikované transformační vazby

Stavově specifikované transformační vazby se liší od svých odpovídajících běžných, nestavově specifikovaných verzí v tom, že namísto propojení transformovaného subjektu (spotřebitele, ovlivněného subjektu nebo výsledného subjektu) s transformačním procesem nebo od něj propojují jeden ze stavů transformovaného subjektu s tímto procesem nebo od něj.

Tabulka 13.2 Stavby specifikující aktivující vazby: sémantika, symboly, zdroj a cíl

Name	Semantics	Sample OPD & OPL	Source	Desti-nation
State-specified Agent Link	The human agent enables the process provided she is at the specified state.	 Healthy Miner handles Copper Mining.	Agent state	Process
State-specified Instrument Link	The process requires the instrument at the specified state.	 Copper Mining requires operational Drill.	Instrument state	Process

Každý ze tří transformačních odkazů – spotřeba, účinek a výsledek – má verzi specifikovanou stavem, jak je definováno níže. Tři transformované objekty – spotřebitel, transformovaný objekt a výsledek – jsou také rolemi ve vztahu k odpovídajícím procesům, které jsou s nimi spojeny, stejně jako agent a nástroj. Podobně termíny „vstupní stav“ a „výstupní stav“ odkazují na role dvou stavů ovlivněného objektu ve vztahu k ovlivňujícímu procesu. Vstupní stav je stav těsně před zahájením ovlivňujícího procesu, zatímco výstupní stav je stav, ve kterém se objekt nachází v okamžiku ukončení tohoto procesu.

<i>Vstupní stav objektu B je stav s_i objektu B, ve kterém se B nachází v okamžiku spuštění procesu P. Vstupní spojení je spojení ze stavu s_i k procesu P.</i>
<i>Výstupní stav objektu B je stav s_o objektu B, ve kterém se B nachází při ukončení procesu P. Výstupní spojení je spojení z procesu P do stavu s_o.</i>
<i>Stavově specifikovaná spotřební vazba je spotřební vazba, která pochází ze vstupního stavu s_i spotřebitele C a končí v procesu P, což znamená, že aby mohl být C spotřebován procesem P, musí být ve stavu s_i.</i>

Stavově specifikované spotřební spojení vyjadřuje skutečnost, že spotřebitel je spotřebován procesem tehdy a jen tehdy, pokud je spotřebitel ve specifikovaném stavu – v tom, ke kterému je spotřební spojení připojeno.

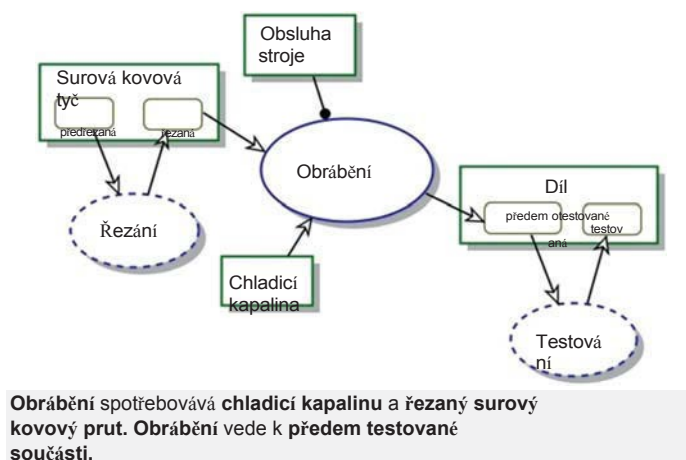
Stavově specifikovaný výsledkový odkaz je výsledkový odkaz, který pochází z procesu **P** a končí ve stavu **s** výsledku **R**, což znamená, že když **P** skončí, vytvoří **R** ve stavu **s**.

Stavově specifikované výsledkové propojení vyjadřuje skutečnost, že výsledek je generován procesem pouze ve specifikovaném stavu – v tom, ke kterému je výsledkové propojení připojeno.

Státem specifikovaný efektový odkaz je pár vstupních a výstupních odkazů, jehož vstupní odkaz pochází ze vstupního stavu s_i ovlivněného subjektu **A** a končí v procesu **P**, a jehož výstupní odkaz pochází z **P** a končí ve výstupním stavu s_o subjektu **A**, což znamená, že aby byl subjekt **A** ovlivněn procesem **P**, **A** musí být v $s(i)$, v takovém případě bude **A** po ukončení **P** v $s(o)$.

Obrázek 13.11 ukazuje dva příklady odkazů na spotřebu a výsledek specifikovaných stavem. **Obrábění** může spotřebovávat **surový kovový prut** pouze ve stavu **řezání** a generovat **díl** ve stavu **předběžného testování**. Následují odpovídající věty OPL. Syntaxe OPL pro objekt specifikovaný stavem je „název stavu“ následovaný „název objektu“. Tato syntaxe je demonstrována ve dvou větách OPL na obr. 15.13 pomocí **řezané surové kovové tyče** a **předem testované součásti**. Při pojmenovávání stavu je proto třeba otestovat jeho výstižnost tím, že se vyhodnotí, zda fráze, která je výsledkem tohoto spojení, dává smysl a dobře se čte ve větách OPL, kde se objevuje.

Vzhledem k tomu, že funkcí tohoto systému je **obrábění**, jsou **řezání** a **testování** procesy prostředí.² **Řezání** musí předcházet **obrábění**, aby se **surová kovová tyč** změnila z **předřezaného** stavu na **řezaný** stav, zatímco **testování** změní **díl** z **předtestovaného** na **testovaný**. Další příklady transformačních odkazů specifikovaných stavem jsou uvedeny v tabulce 13.3.



Obr. 13.11 Stavy specifikované spotřebou a výslednými odkazy: **Obrábění** může spotřebovávat pouze **surové kovové tyče** ve stavu **řezání** a generovat **díly** ve stavu **předběžného testování**.

² V systému s větším rozsahem **výroby** by všechny tři procesy, **řezání**, **obrábění** a **testování**, v tomto pořadí, byly systémovými dílčími procesy **výroby**.

13.9.1 Změna stavu versus spotřeba a generování objektů

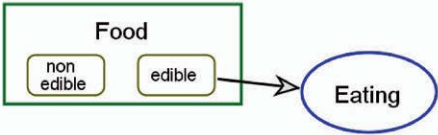
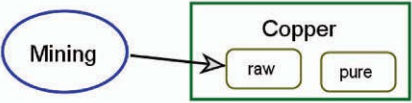
Zjistili jsme, že spotřebu a generování objektů lze považovat za extrémní případy změny stavu, kdy stavy implicitně neexistují a implicitně existují. Například na [obr. 13.11](#) proces **obrábění** spotřebovává objekt – **surovou tyč** – a generuje jiný **objekt – díl**. Procesy **řezání** a **testování** však mění pouze stavy **surové tyče** a **dílu**, nikoli jejich identitu. Důvodem je to, že proces **obrábění** je drastickější – zásadně mění vstupní objekt tak, že ztrácí svou identitu jako **surová tyč** a vzniká nový objekt, **díl**.

V důsledku procesu **obrábění** změnil **surový materiál** svůj stav z **existujícího** na **neexistující**. Jinými slovy, byl spotřebován. **Díl** změnil svůj stav z **neexistujícího** na **existující**. Jinými slovy, byl vytvořen. Naopak **testování díl** nespotebovává. Pouze přidává informace o dílu, které udávají, zda může postoupit do další fáze výroby. **Řezání** není tak jednoznačným případem, protože lze oprávněně argumentovat, že tento proces bere jako vstup dlouhý objekt **suroviny** a jako výstup několik kratších objektů **segmentů suroviny**, z nichž každý je samostatně vložen do **obrábění**.

13.10 Stavově specifikované efektové vazby

Každé z pěti procedurálních propojení uvedených v [tabulce 13.1](#) má protějšek specifikovaný stavem, který je uveden v [tabulce 13.3](#). Jediné propojení bez stavu z [tabulky 13.1](#) vede ke třem druhům párů propojení specifikovaných stavem, které jsou uvedeny v [tabulce 13.4](#).

Tabulka 13.3 Spotřeba a výsledek stavově specifikovaných procedurálních vazeb: sémantika, symboly, zdroj a cíl

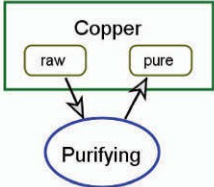
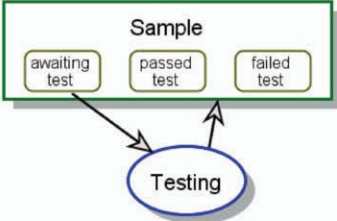
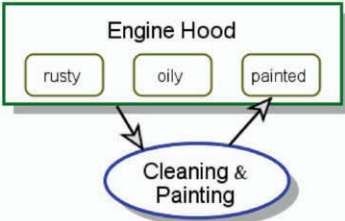
Name	Semantics	Sample OPD & OPL	Source	Destination
State-specified consumption link	The process consumes the object if and only if the object is in the specified state.	 Eating consumes edible Food.	consumee state	consuming process
State-specified result link	The process generates the object in the specified state.	 Mining yields raw Copper.	generating process	resultee state

Namísto jediného efektového propojení v [tabulce 13.1](#), kdy stavy nebyly přítomny, existují v [tabulce 13.4](#) tři typy párů efektových propojení specifikovaných stavem. Každý pár propojení se skládá ze vstupního propojení a výstupního propojení.

Rozdíl mezi nimi vyplývá z původu vstupního odkazu a cíle výstupního odkazu. Slovo *in-out* používáme jako zkratku pro vstup-výstup.

Pár efektových spojů s určeným vstupem a výstupem procesu P je pár spojů sestávající ze vstupního spoje ze vstupního stavu s_{in} objektu B do P a výstupního spoje z P do výstupního stavu s_o .

Tabulka 13.4 Procedurální odkazy specifikované stavem vstupu a výstupem: sémantika, symboly, zdroj a cíl

Name	Semantics	Sample OPD & OPL	Source	Destination
In-out-specified effect link pair (consisting of one state-specified <i>input link</i> and one state-specified <i>output link</i>)	The process changes the object from a specified input state via the <i>input link</i> to a specified output state via the <i>output link</i> .	 Purifying changes Copper from raw to pure.	affectee input (source) state	affecting process
			affecting process	affectee output (destination) state
Input-specified effect link pair (consisting of one state-specified <i>input link</i> and one state-unspecified <i>output link</i>)	The process changes the object from a specified input state to any output state.	 Testing changes Sample from awaiting test.	affectee input (source) state	affecting process
			affecting process	affectee
Output-specified effect link pair (consisting of one state-unspecified <i>input link</i> and one state-specified <i>output link</i>)	The process changes the object from any input state to a specified output state.	 Cleaning & Painting changes Engine Hood to painted.	affectee	affecting process
			affecting process	affectee output (destination) state

V příkladu pro efektové propojení specifikované vstupem a výstupem v [tabulce 13.4](#) je věta OPL:

Purifying mění **Copper** ze **surového** na **čistý**.

Zde jsou **surová** a **čistá** vstupní (zdrojový) a výstupní (cílový) stav **mědi**.

Pár efektových vazeb specifikovaných vstupem procesu **P** je pár vazeb sestávající ze vstupní vazby ze vstupního stavu s_{in} objektu **B** do **P** a výstupní vazby z **P** do **B**.

V příkladu pro výstupem specifikovanou efektovou vazbu v [tabulce 13.4](#) je věta OPL:

Testování změn Vzorek z čekajícího testu.

Zde je **čekající test** vstupním stavem **vzorku**. Výstupní stav **vzorku** není specifikován, což znamená, že (v závislosti na výsledku **testování**) může být kterýmkoli ze tří stavů **vzorku**.

***Pár efektových vazeb s určeným výstupem** procesu **P** je pár vazeb sestávající z vstupní vazby z objektu **B** do **P** a výstupní vazby z **P** do výstupního stavu s_{out} objektu **B**.*

V příkladu pro výstupně specifikované efektové propojení v [tabulce 13.3](#) je věta OPL:

Čištění a lakování mění stav **kapoty motoru** na **lakovaný**.

Zde je **lakovaný** výstupním stavem **kapoty motoru**. Vstupní stav **kapoty motoru** není specifikován, což znamená, že může být kterýmkoli ze tří stavů **kapoty motoru**.

13.10.1 Procedurální odkazy specifikované hodnotou

***Procedurální odkaz specifikovaný hodnotou** je odkaz mezi procesem a jednou nebo dvěma hodnotami atributu, který proces mění.*

Každý stavově specifikovaný procedurální odkaz v [tabulce 13.4](#) má svůj protějšek v podobě hodnotově specifikovaného procedurálního odkazu. Tři hodnotově specifikované procedurální odkazy jsou znázorněny v [tabulce 13.5](#). Hodnoty jsou stavy atributů, takže sémantika a syntaxe hodnotově specifikovaných procedurálních odkazů se poněkud liší od jejich stavově specifikovaných protějšků, jak je uvedeno v [tabulce 13.5](#) a definováno níže.

***Odkaz nastavení hodnoty** je jednosměrný procedurální odkaz specifikovaný hodnotou z procesu na hodnotu atributu, který nastavuje tuto hodnotu bez ohledu na to, jaká byla dříve.*

Odkaz pro nastavení hodnoty je protějškem odkazu pro výsledek specifikovaného stavem objektu, který není atributem. Rozdíl spočívá v tom, že zatímco odkaz pro výsledek specifikovaný stavem vytvoří objekt ve specifikovaném stavu, atribut není vytvořen, protože existuje společně se svým vystavovatelem. Jeho úkolem je specifikovat hodnotu tohoto atributu.

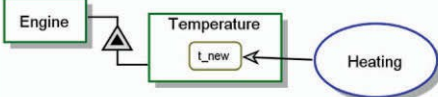
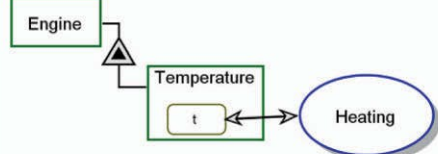
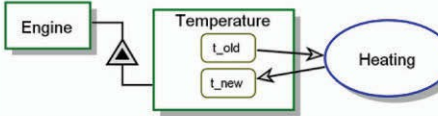
***Odkaz na hodnotový efekt** je obousměrný procedurální odkaz specifikovaný hodnotou z procesu na hodnotu atributu a zpět, který mění tuto hodnotu z nějaké nespecifikované hodnoty na jinou.*

Hodnotu lze snadno odlišit od stavu kontrolou objektu, který daný stav „vlastní“: Pokud je tento vlastnický objekt atributem, pak je stav hodnotou, a pokud ne, pak je stav pouze stavem.

Odkaz na hodnotový efekt je protějškem odkazu na efekt specifikovaný stavem objektu, který není atributem. Rozdíl spočívá v tom, že zatímco odkaz na efekt specifikovaný stavem mění objekt z jednoho nespecifikovaného stavu na jiný, odkaz na hodnotový efekt mění hodnotu z nějaké nespecifikované hodnoty na jinou.

Pár efektových odkazů s určenou hodnotou vstupu a výstupu je pár odkazu s určenou hodnotou vstupu a odkazu s určenou hodnotou výstupu, které mění hodnotu atributu ze vstupní hodnoty na výstupní hodnotu.

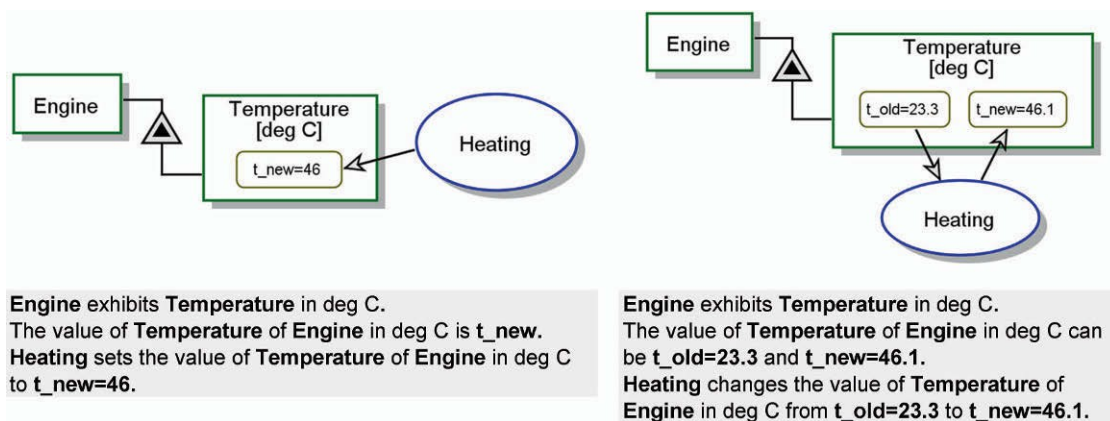
Tabulka 13.5 Procedurální vazby s určenou hodnotou: sémantika, symboly, zdroj a cíl

Name	Semantics	Sample OPD & OPL	Source	Destination
Value Setting Link	The process sets the value of an attribute.	 Engine exhibits Temperature. The value of Temperature of Engine is t_new. Heating sets the value of Temperature of Engine to t_new.	process	attribute value
Value Effect Link	The process affects the value of an attribute by changing it from an unspecified input value to an unspecified output value.	 Engine exhibits Temperature. The value of Temperature of Engine is t. Heating affects the value t of Temperature of Engine.	attribute value and process	attribute value and process
In-out-specified value effect link pair (consisting of one value-specified input link and one value-specified output link)	The process changes the value of an attribute by changing it from an input value to an output value.	 Engine exhibits Temperature. The value of Temperature of Engine can be t_old and t_new. Heating changes the value of Temperature of Engine from t_old to t_new.	input value process	Process output value

Pár odkazů s efektem hodnoty specifikované vstupem a výstupem je protějškem páru odkazů s efektem specifikované vstupem a výstupem objektu, který není atributem. Rozdíl spočívá v tom, že zatímco pár odkazů s efektem specifikované vstupem a výstupem mění

objekt z jednoho specifikovaného stavu do jiného specifikovaného stavu, hodnotový efektový odkaz mění hodnotu z některé specifikované hodnoty na jinou specifikovanou hodnotu.

Názvy hodnot jsou názvy parametrů. Například **t_new** je nová hodnota **teploty motoru** nastavená **topením**. Parametrům můžeme také přiřadit skutečná čísla, jak je znázorněno na [obr. 13.12](#).



Obr. 13.12 Procedurální odkazy s parametry a skutečnými číselnými hodnotami

13.11 Shrnutí

- **Změna** objektu je změna stavu tohoto objektu.
- **Účinek** je změna stavu objektu, kterou způsobuje proces.
- **Konstrukce** je extrémním případem účinku objektu, kdy vstupní stav objektu neexistuje a výstupní stav existuje.
- **Spotřeba** je extrémním případem efektu objektu, kdy vstupní stav objektu existuje a výstupní stav neexistuje.
- Při **extrémní** transformaci dochází ke **změně identity objektu**.
- Pokud změna není zásadní nebo drastická, objekt pouze mění svůj stav, ale zachovává si svou identitu.
- **Transformátorem** procesu *P* je objekt *B*, který *P* transformuje v důsledku výskytu *P*.
 - **Spotřebitel** procesu *P* je transformovaný objekt *P*, který *P* spotřebovává v důsledku výskytu *P*.
 - **Výsledkem** procesu *P* je transformát *P*, který *P* vytvoří v důsledku výskytu *P*.
 - Ovlivněný proces *P* je transformátem *P*, který *P* ovlivňuje v důsledku výskytu *P*.

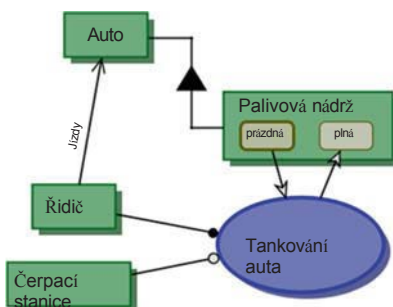
- **Transformační vazba** je procedurální vazba, která spojuje proces s transformovaným prvkem tohoto procesu.
 - **Výsledkový odkaz** je jednosměrný transformační odkaz z procesu k výsledku, který tento proces vytváří.
 - **Efektové propojení** je obousměrné transformační propojení, které spojuje proces s objektem, na který má tento proces vliv.
 - **Spotřební odkaz** je jednosměrný transformační odkaz od spotřebitele k procesu, který jej spotřebovává.
- **Enabler** E procesu P je objekt, který musí existovat a být k dispozici, aby mohl P začít, a musí zůstat přítomen po celou dobu trvání P , aby P mohl normálně skončit, přičemž E zůstane nakonec nedotčen.
 - **Agent** je aktivátor, který je člověkem nebo skupinou lidí.
 - **Nástroj** je nehumánní aktivátor.
- **Enabling link** je procedurální propojení, které spojuje proces s enablerem tohoto procesu.
 - **Agent link** je aktivní odkaz, který propojuje proces s agentem tohoto procesu.
 - **Odkaz nástroje** je procedurální odkaz, který spojuje proces s aktivátorem tohoto procesu.
- **Vstupní stav** objektu B je stav s_i objektu B , ve kterém se B nachází v okamžiku spuštění procesu P .
- **Vstupní odkaz** je odkaz ze stavu s_i k procesu P .
- **Výstupní stav** objektu B je stav s_o objektu B , ve kterém se B nachází po skončení procesu P .
- **Výstupní spojení** je spojení z procesu P do stavu s_o .
- **Stavově specifikovaná spotřební vazba** je spotřební vazba, která vychází ze vstupního stavu s_i spotřebitele C a končí v procesu P , což znamená, že aby mohl být C spotřebován procesem P , musí být ve stavu s_i .
- **Výsledkové spojení specifikované stavem** je výsledkové spojení, které pochází z procesu P a končí ve stavu s výsledku R , což znamená, že když P skončí, vytvoří R ve stavu s .
- **Stavově specifikované efektové propojení** je pár vstupně-výstupních propojení, jehož vstupní propojení pochází ze vstupního stavu s_i ovlivněného subjektu A a končí v procesu P , a jehož výstupní propojení pochází z P a končí ve výstupním stavu s_o subjektu A , což znamená, že aby byl subjekt A ovlivněn procesem P , musí být A ve stavu $s(i)$, v takovém případě bude A po ukončení P ve stavu $s(o)$.
 - **Pár efektových spojů** procesu P **specifikovaný vstupem a výstupem** je pár spojů sestávající ze vstupního spoje ze vstupního stavu s_{in} objektu B do P a výstupního spoje z P do výstupního stavu s_{out} objektu B .
 - **Pár efektových vazeb specifikovaných vstupem** procesu P je pár vazeb sestávající ze vstupní vazby ze vstupního stavu s_{in} objektu B do P a výstupní vazby z P do B .
 - **Pár efektových vazeb specifikovaných výstupem** procesu P je pár vazeb sestávající z vstupní vazby z objektu B do P a výstupní vazby z P do výstupního stavu s_{out} objektu B .
- **Odkaz měnící hodnotu** je odkaz mezi procesem a nespecifikovanou hodnotou atributu, kterou proces mění.

13.12 Problémy

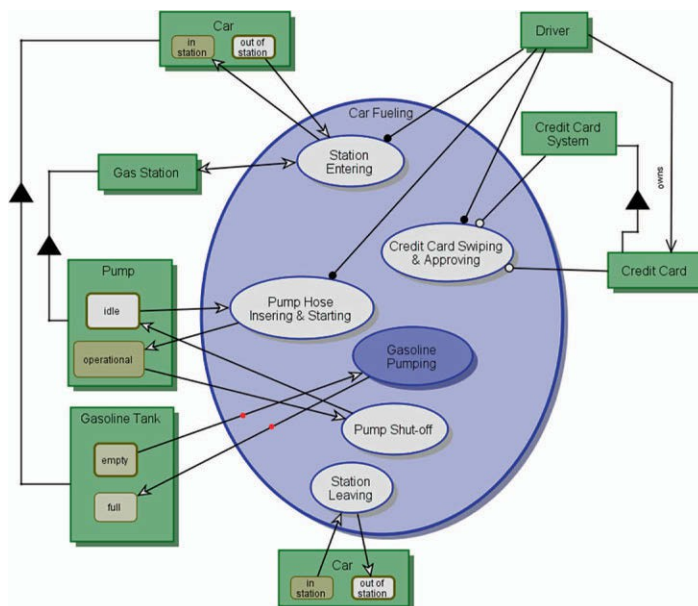
1. Uveďte dva příklady každého z následujících případů, uveďte jejich modely OPM a ověřte, zda výsledný OPL odpovídá vašemu původnímu záměru.
2. Změna objektů.
3. Spotřeba objektů.
4. Vytvoření objektů.

Následující otázky se vztahují k [obrázkům 13.13](#) a [13.14](#), které popisují systém simulovaný animací.

5. Jaký systém je popsán v tomto OPD?
6. Co je ovlivněný subjekt, činitel a nástroj?
7. Jaká je věta OPL, která popisuje účinek?
8. Jaký je vztah mezi **řidičem** a **autem**? Jaký odkaz se používá k vyjádření tohoto vztahu?
9. Jaký je vztah mezi **palivovou nádrží** a **automobilem**? Jaký odkaz se používá k vyjádření tohoto vztahu?
10. Jaký proces probíhá v tomto okamžiku při **tankování automobilu**?
11. Jaký objekt se transformuje a jakým způsobem? Jaká věta OPL to popisuje?
12. Jaký odkaz chybí mezi tímto procesem a **čerpací stanicí**? Jaká věta OPL by vznikla, pokud byste tento odkaz přidali?
13. Kterých pět subjektů je v tomto OPD ovlivněno? Který z nich se liší od ostatních čtyř a proč?
14. Jaký spojovací článek chybí?
15. Popište změny stavu **čerpání**.
16. Co můžete poznat podle barev stavů různých objektů? Odkazujte se konkrétně na **benzinové nádrže**.
17. Jaký efektní odkaz chybí? Nápoděda: Podívejte se na ten, který je přítomen.
18. Předpokládejme, že **auto** je jediným vozidlem v **čerpací stanici** během celého procesu **tankování**. Kam byste umístili stavy nazvané „**auto přítomno**“ a „**auto nepřítomno**“?
19. Jak by se v důsledku toho změnilo odpovídající propojení?
20. Jaký proces vrací **čerpací stanici** do původního stavu?



Obr. 13.13 OPD pro problémy v kapitole



Obr. 13.14 OPD tankování automobilu z obr. 13.13 – zvětšené

21. Na základě vašich odpovědí na předchozí dvě otázky vysvětlíte, proč je v [obr. 13.13](#) čerpací stanice je nástrojem, zatímco na [obr. 13.14](#) je ovlivňovaným subjektem?

Kapitola 14

Strukturální systémový aspekt

Prasátko žilo ve velmi velkém domě uprostřed buku, a buk byl uprostřed lesa, a prasátko žilo uprostřed domu. Vedle jeho domu ležel kus rozbité desky, na které bylo napsáno: „TRESPASSERS W“ (Vstup zakázán).

Medvídek Pú, A. A. Milne

Struktura se týká relativně pevných, netr transientních, dlouhodobých vztahů, které existují mezi objekty – komponenty nebo částmi systému. Alternativně lze strukturu vnímat jako momentku – obraz obecně dynamického systému nebo jeho části v určitém časovém okamžiku. Tato momentka zachycuje celý systém v určitém stavu, kdy každý stavový objekt je v určitém stavu nebo v přechodu mezi dvěma svými stavy a mezi objekty existují specifické vztahy. Struktura je v kontrastu s komplementárním dynamickým aspektem systému, neboli jeho chováním, které souvisí se změnami, kterými systém v průběhu času prochází, spolu s příčinami a důsledky těchto změn. Jinými slovy, struktura se týká statického aspektu systému, zatímco chování se týká jeho dynamického aspektu. Tato kapitola je věnována diskusi o struktuře systémů a jejím vyjádření prostřednictvím OPM.

14.1 Strukturální vztahy

Základním pojmem, který je nezbytný pro diskusi o struktuře, je strukturální vztah.

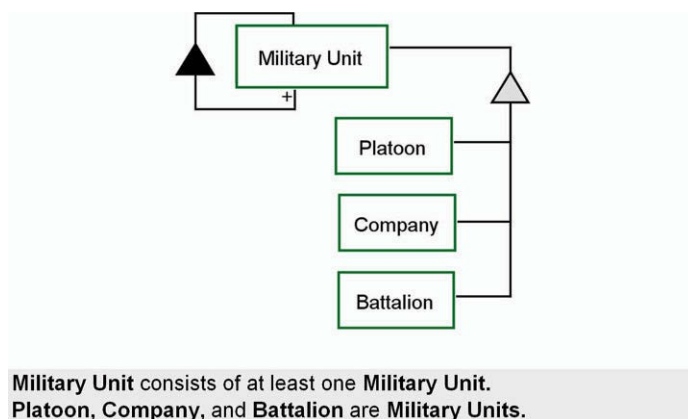
Strukturální vztah je propojení, spojení nebo asociace mezi dvěma objekty nebo mezi dvěma procesy, které v systému trvá alespoň po určitou dobu.

Strukturální vztah v systému není závislý na podmínkách, které jsou časově závislé. Zatímco strukturální vztahy obvykle existují mezi dvěma objekty a méně často se používají mezi dvěma procesy, existují procesy, které vykazují strukturální vztahy. Například podprocesy uvnitř přiblíženého procesu jsou součástí tohoto procesu, takže mezi nimi existuje strukturální vztah, konkrétně vztah agregace-účasti (celek-část), přičemž přiblížený proces je celek a podprocesy jsou části. Jak diskutujeme níže, strukturální vazba je grafickým vyjádřením strukturálního vztahu.

14.1.1 Binární vztahy v centru pozornosti

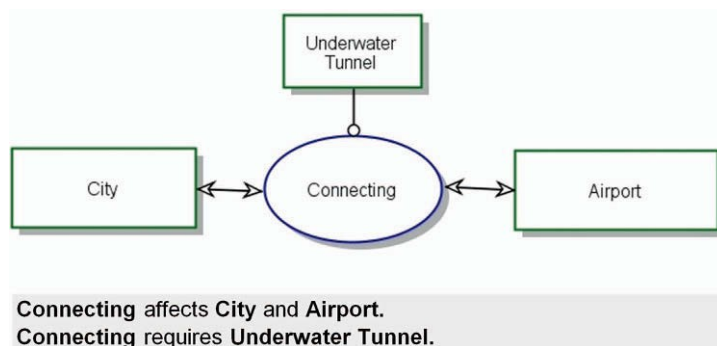
Strukturální vztah je ze své podstaty multilaterální, protože každá věc, která se účastní asociace, má nějaký vztah k ostatním věcem. Počet věcí zapojených do strukturálního vztahu určuje *aritu* vztahu: vztah věci k sobě samé je *unární* strukturální vztah, vztah mezi dvěma věcmi je *binární* strukturální vztah, vztah mezi třemi věcmi je *ternární* strukturální vztah atd.

Jednoduchý strukturální vztah je možný s ohledem na agregační účast, která je strukturálním vztahem. Například, jak ukazuje OPD na [obr. 14.1](#), **vojenská jednotka** se skládá z jedné nebo více (menších) **vojenských jednotek**.



Obr. 14.1 Modelování unární strukturální vazby jako vazby od objektu k sobě samému

Jako příklad modelování ternárních nebo vyššího řádu strukturálních vztahů uvažujme modelování následující věty: „Podvodní tunel spojuje město s letištěm.“ Vztah „spojuje“ lze považovat za ternární vztah, protože zahrnuje tři objekty v této větě. Pomocí procesu zachovávajícího stav **Spojování** můžeme tento systém modelovat v OPM pomocí tří procedurálních vazeb, jak je znázorněno na [obr. 14.2](#).



Obr. 14.2 Modelování ternárního strukturálního odkazu jako tří procedurálních odkazů pomocí procesu zachovávajícího stav **Connecting**

Jak ukazuje tento příklad, n -ární relace s $n \geq 3$ lze analyzovat jako soubor strukturálních nebo procedurálních binárních odkazů. [Obrázek 14.3](#) ukazuje systém z [obrázku 14.2](#), který používá pouze dva strukturální odkazy a žádný proces. Navíc, jak uvidíme, vzhledem k tomu, že relace označená **jako propojená** je tranzitivní, je možné dojít k závěru, že „**město a letiště** jsou **propojené**“.



Obr. 14.3 Systém na obr. 14.2 specifikovaný pouze pomocí struktury

S ohledem na tuto pozorování se zaměříme na binární vztahy. Binární strukturální vztahy existují mezi dvěma věcmi se stejnou vytrvalostí, tj. buď mezi dvěma objekty – věcmi, jejichž hodnota vytrvalosti je trvalá nebo statická, nebo mezi dvěma procesy – věcmi, jejichž hodnota vytrvalosti je přechodná nebo dynamická. Například agregace-účast je základní strukturální vztah, který je použitelný jak pro procesy, tak pro objekty. Obecně však strukturální vztahy převládají spíše u objektů než u procesů, proto se v této kapitole zaměříme na binární strukturální vztahy mezi objekty.

14.1.2 Přímé a zpětné strukturální vztahy

Binární strukturální vztah je obousměrný: pokud věc T_1 souvisí s věcí T_2 prostřednictvím vztahu A , pak platí také, že T_2 souvisí s T_1 prostřednictvím jiného vztahu A' , a naopak.

Symbolicky lze vyjádřit jako $T_1 \ A \ T_2$ a $T_2 \ A' \ T_1$.

A je symbol, který představuje *dopředný strukturální vztah*, tj. strukturální vztah, jak jej vnímá T_1 při odkazování na T_2 . Naopak A' je *zpětný strukturální vztah*, tj. vztah, jak jej vnímá T_2 při odkazování na T_1 . A a A' tvoří *pár strukturálních vztahů*.

A a A' mohou být sémanticky identické, v takovém případě jsou symetrické. Symbolicky $A = A'$. Například, pokud $A =$ „dotýká se“, pak $A' =$ „dotýká se“. Pokud tedy „**A se dotýká B**“, pak „**B se dotýká A**“. Jak diskutujeme v pokračování, pár symetrických vztahů lze převést na jeden vzájemný vztah.

A a A' mohou být vzájemně inverzní, neboli antisymetrické, takže existence vztahu v jednom směru vyžaduje existenci opačného vztahu v opačném směru. Symbolicky, pokud jsou A a A' vzájemně inverzní, pak $(A')' = A$. Například, pokud $A =$ „je rodičem“, pak $A' =$ „je dítětem“. Pokud tedy „**A je rodičem B**“, pak „**B je dítětem A**“. Ne každá dvojice vztahů je symetrická nebo antisymetrická. Například pokud A **ukazuje na B**, nemusí nutně platit, že B **ukazuje na A**.

14.1.3 Strukturální vazby versus strukturální vztahy

Odkazy graficky vyjadřují sémantiku vztahů.

Odkaz je grafickým znázorněním vztahu.

Tato definice odkazu platí jak pro procedurální, tak pro strukturální vztahy. Zatímco strukturální vztah modeluje asociace mezi dvěma věcmi (obvykle objekty), které jsou v systému významné, *strukturální odkaz* graficky znázorňuje strukturální vztah.

Strukturální odkaz je šipka s otevřenou hlavičkou, která představuje binární strukturální vztah v OPD od zdrojového objektu k cílovému objektu.

Vztah mezi strukturální relací a strukturálním propojením je stejný jako vztah mezi procedurální relací a procedurálním propojením: v obou případech propojení graficky vyjadřuje to, co relace vyjadřuje verbálně. Například procedurální (instrumentální) spojení na [obr. 14.2](#) mezi **podvodním tunelem** a **propojením** je grafickým vyjádřením instrumentálního vztahu mezi těmito dvěma věcmi. Stejně tak strukturální (obousměrné označené) spojení na [obr. 14.3](#) mezi **podvodním tunelem** a **městem** je grafickým vyjádřením strukturálního vztahu mezi těmito dvěma objekty, označeným jako **propojené**.

Otevřená šipka, $\longrightarrow$, která symbolizuje strukturální spojení, kontrastuje s uzavřenými trojúhelníkovými šipkami transformačních spojení. Připomeňme, že transformační odkazy zahrnují čtyři jednosměrné odkazy symbolizované jako $\longrightarrow$ s různými zdroji a cíli: (1) odkaz spotřeby (z objektu do procesu), (2) odkaz výsledku (z procesu do objektu), (3) odkaz vstupu (ze stavu do procesu) a (4) odkaz výstupu (z procesu do stavu). Jak vidíme, tyto čtyři vazby sdílejí stejný grafický symbol a jediným rozdílem mezi nimi jsou jejich různé zdroje a cíle. Jedinou obousměrnou transformační vazbou je vazba účinku, $\longleftrightarrow$, spojující proces a objekt, která označuje nespecifikovanou změnu stavu (když budeme diskutovat o hodnotách, které jsou stavy atributů, uvidíme další použití tohoto symbolu, stejně jako pro vstupní a výstupní vazby).

14.1.4 Strukturální značka a označené strukturální spojení

Otevřená šipka strukturálního odkazu směřuje z jednoho objektu na druhý. Frázi jsme definovali jako kombinaci jednoho nebo více slov oddělených mezerami, která tvoří logickou entitu, ale ne úplnou větu. Významný *strukturální tag* lze zaznamenat podél strukturálního odkazu, čímž se stane *tagovaným strukturálním odkazem*.

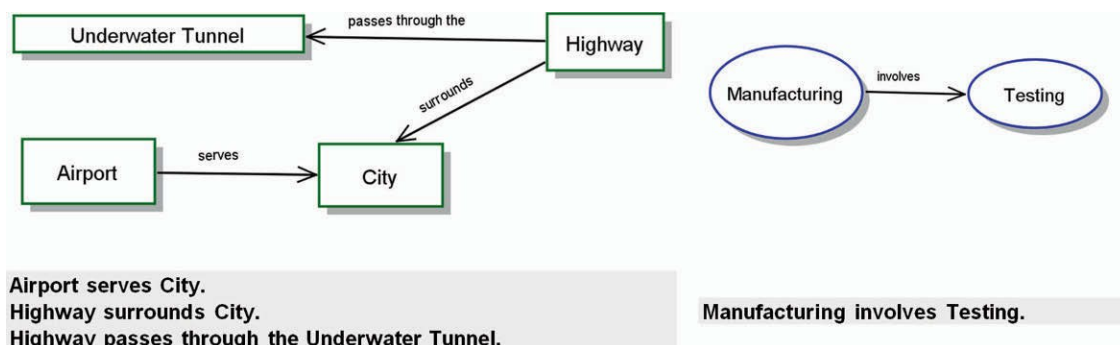
Strukturální značka je fráze, která vyjadřuje sémantiku – povahu, význam nebo obsah – strukturálního vztahu mezi dvěma věcmi, které se na tomto vztahu podílejí.

Příklady strukturálních značek (nebo zkráceně jen značek) jsou: **podobá se**, **vlastní**, **je vedle**, **podobný**, **rozšiřuje**, **omezuje**, **nachází se v**, **hraničí s**, **eliminuje potřebu**, **představuje**, **je ekvivalentní** a **obsahuje**.

Štítkovaný strukturální odkaz je strukturální odkaz se strukturálním štítkem zaznamenaným podél odkazu.

Vlevo na [obr. 14.4](#) je příklad OPD se čtyřmi objekty – **letišťe**, **město**, **dálnice** a **podmořský tunel** – a třemi značenými strukturálními odkazy. Dva jsou jednosměrné, se značkami „**slouží**“ a „**obklopuje**“. Třetí je obousměrný značený strukturální odkaz se značkami „**prochází**“ a „**umožňuje zkrácení**“.

OPD na pravé straně [obr. 14.4](#) ukazuje, že strukturální vztah může existovat také mezi dvěma procesy. Jak ukazují věty OPL na [obr. 14.4](#), syntaxe označené strukturální věty OPL je jednoduchým zřetěžením zdrojové věci následované značkou a následované cílovou věcí. Značka není psána velkými písmeny a je **tučně** vyznačena písmem **Arial**, protože je definována uživatelem a nejedná se o vyhrazenou frázi.



Obr. 14.4 Příklady jednosměrných označených strukturálních vztahů

14.1.5 Obousměrný označený strukturální odkaz

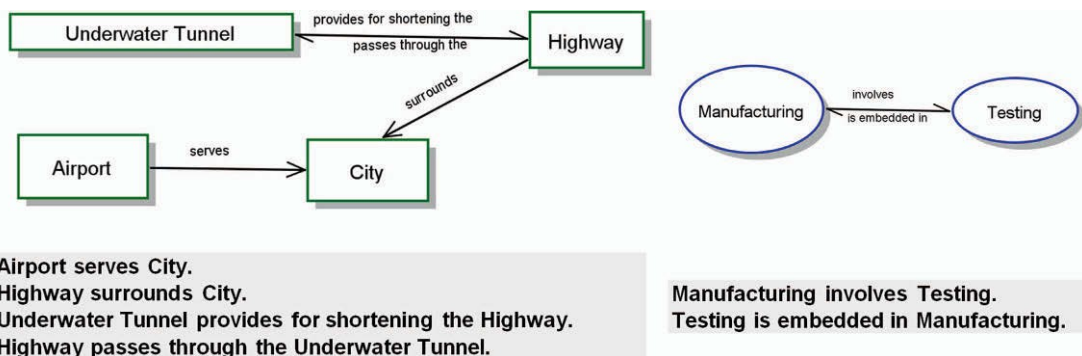
Dosud definované a ilustrované značkové strukturální vztahy a odkazy byly jednosměrné. Značka je tedy *přední značka* – značka od zdrojové věci k cílové věci. Jak již bylo uvedeno, pro každý značkový strukturální vztah (a jeho grafické znázornění jako odkaz) existuje odpovídající značkový strukturální vztah (a odkaz) v opačném směru, vyjádřený *zadní značkou* – značkou od cílové věci ke zdrojové věci. Tyto dva vztahy můžeme vyjádřit v jediném obousměrném označeném strukturálním odkazu.

Obousměrný štítkový strukturální odkaz je kombinací dvou štítkových strukturálních odkazů v opačných směrech.

Obousměrný strukturální odkaz je znázorněn jako šipka ve tvaru harpuny, $\longleftrightarrow$, která spojuje dvě věci. Značky vztahu v jednom směru a v druhém směru jsou zaznamenány tak, že hrany harpuny vyčnívající z hrotů šipek jednoznačně určují směr, ve kterém každý vztah platí, jak ukazuje [obr. 14.5](#). Harpuna směřuje k cílovému objektu. Zvažte OPD na levé straně [obr. 14.5](#), kde jsou objekty **Dálnice** a **Podvodní tunel** propojeny obousměrným označeným strukturálním vztahem. Značka z **Dálnice** do **Podvodního tunelu** je „prochází“, zatímco značka v opačném směru je „umožňuje zkrácení“. Podobně v OPD na pravé straně [obr. 14.5](#) jsou procesy **Výroba** a **Testování** propojeny obousměrnou strukturovanou relací, kde jsou dvě značky „zahrnuje“ a „je zabudováno do“.

V OPD vlevo na [obr. 14.5](#) je inverzní vztah z **města** na **letišti** „je obsluhován“ a vztah z **města** na **dálnici** je „je obklopen“. Jelikož specifikováním těchto pasivních vztahů nejsou přidány žádné informace, vyhýbáme se tomu. Musíme však mít na paměti, že v zásadě je každý strukturální vztah obousměrný.

UML a SysML používají asociační odkaz jako něco analogického k označenému strukturálnímu odkazu OPM (jednosměrnému i obousměrnému), ale tyto odkazy jsou použitelné pouze pro objekty (v UML) nebo pro jejich protějšky v SysML, nazývané bloky. Pokud jde o procesy, je to možné pouze v diagramech případů použití, kde značky mohou označovat vztahy mezi případy použití.



Obr. 14.5 Dálnice a podvodní tunel jsou propojeny obousměrným označeným strukturálním vztahem

14.2 Reciprocita a transitivita strukturálních vztahů

Dvě vlastnosti (atributy prvků modelu OPM) strukturálních vztahů jsou užitečné pro odvozování a určování příčin a následků: reciprocita a transitivita. Tyto vlastnosti jsou popsány a definovány níže.

14.2.1 Atribut reciprocit strukturalního vztahu

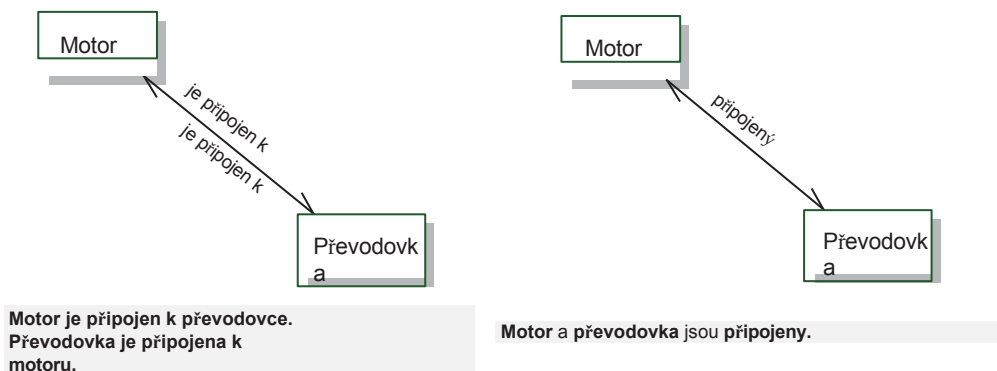
V příkladech, které jsme viděli, se antisymetrické dopředné a zpětné strukturální vztahy mezi zdrojovými a cílovými objekty liší názvem a sémantikou. Existují však případy, kdy jsou dopředné a zpětné strukturální vztahy identické. Uvažujme příklad v OPD na levé straně [obr. 14.6](#) a odpovídající dvě věty OPL.

Obousměrný označený strukturální vztah je reciproční, pokud je strukturální vztah mezi objekty **A** a **B** v obou směrech stejný. Abychom se vyhnuli opakování stejné značky dvakrát v OPD a získání dvou samostatných vět v odstavci OPL, změníme značku tak, aby po spojení s větou „A a B jsou ...“ vznikla smysluplná věta. Například na [obr. 14.6](#) potřebujeme značku k dokončení věty „**Motor a převodovka** jsou ...“. V OPD na pravé straně [obr. 14.6](#) používáme jedinou reciproční strukturální relaci, která je sémanticky ekvivalentní OPD na levé straně.

Reciproční strukturální relace je obousměrná tagovaná strukturální relace, ve které jsou dopředné a zpětné strukturální relace identické.

Algebraicky, necht' A je strukturální vztah z A do B tak, že $A A B$, a necht' A' je strukturální vztah z B do A tak, že $B A' A$. Pokud $A = A'$, pak A je reciproční. Znamenko rovnosti se zde interpretuje jako „jsou sémanticky stejné“.

Reciproční strukturální relace A je strukturální relace, pro kterou platí, že pokud $A B$ a $B A' A$, pak $A = A'$.



Obr. 14.6 Příklad vzájemnosti. Vlevo: Vztah mezi motorem a převodovkou je vzájemný, protože obě značky v obousměrném strukturálním vztahu „je připojen k“ jsou identické. Vpravo: Ekvivalentní, stručnější OPD s připojenou jako jedinou značkou vzájemnosti

Reciproční strukturální odkaz – grafické vyjádření recipročního strukturálního vztahu – je definován následovně.

Reciproční strukturální odkaz je obousměrný strukturální odkaz s tagy, ve kterém jsou identické přední a zadní tagy nahrazeny jediným recipročním tagem.

Další příklady vzájemných strukturálních vztahů zahrnují **vzájemně se vylučující**, **v kontaktu**, **sourozenci**, **příbuzní**, **propojení**, **ekvivalentní**, **sousedící**, **protínající se**, **překrývající se**, **zasnoubení**, **příbuzní**, **spojení**, **v souladu** a **zamilování do sebe**. Přidání předpony „vzájemně“ nebo přípony „se sebou“ může pomoci vyjádřit nebo objasnit vzájemnost mnoha vztahů. Jak ukazuje příklad na [obr. 14.6](#), věta OPL, která vyjadřuje vzájemný strukturální vztah, je vytvořena spojením dvou objektů, spojovaných vyhrazeným slovem and mezi nimi, následovaným vyhrazeným slovem are a vzájemnou strukturální vztahovou značkou.

Reciprocita je vlastnost strukturálního vztahu, která označuje, zda jeho dopředné a zpětné strukturální vztahy mají stejnou sémantiku.

V metamodelu **strukturálního odkazu** jsou tři možné hodnoty jeho vlastnosti **reciprocity**: **pozitivní**, **neutrální** a **negativní**. Hodnota **reciprocity** je **pozitivní**, pokud je sémantika značky odkazu v předním i zadním směru stejná, **negativní**, pokud je sémantika značky v předním směru opačná než v zadním směru, a **neutrální**, pokud sémantika značky v předním směru není ani stejná, ani opačná než v zadním směru. Výchozí hodnota **reciprocity strukturálního odkazu** je **neutrální**.

Obousměrný strukturální odkaz s pozitivní reciprocitou lze zkráceně nazvat *recipročním strukturálním odkazem*, odkaz s negativní reciprocitou je *anti-reciproční strukturální odkaz* a odkaz s neutrální reciprocitou je *nereciproční strukturální odkaz*.

14.2.2 Atribut transitivity strukturálního vztahu

Transitivita strukturálního vztahu má následující definici a sémantiku, která je stejná jako definice transitivity v algebře:

Transitivní strukturální vztah A je strukturální vztah, pro který platí, že pokud $A B$ a $B A C$, pak $A A C$.

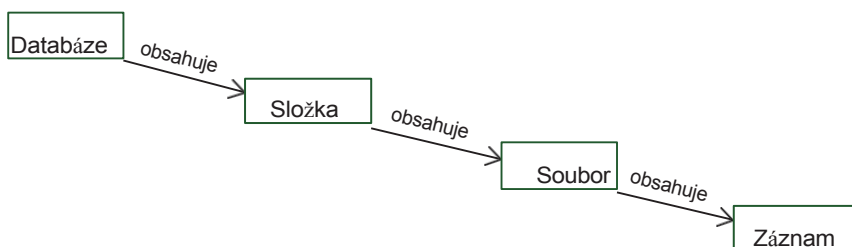
Obrázek 14.7 ilustruje transitivní atribut strukturálního odkazu. Vztah **obsahuje** je transitivní, a proto jej lze rozšířit z jakéhokoli objektu, ze kterého odkaz pochází, na jakýkoli objekt, na který odkaz ukazuje. Vzhledem k tomu, že strukturální vztah **obsahuje** je transitivní, lze z OPD na **obr. 14.7** odvodit další vztahy, například:

Databáze obsahuje soubor.

Složka obsahuje záznam.

Databáze obsahuje záznam.

Transitivní strukturální vztahy vedou k hierarchiím. **Obrázek 14.7** je příkladem hierarchie *obsahování*. Tuto hierarchii můžeme rozšířit například tím, že určíme, že **záznam obsahuje pole**, **pole obsahuje znak** atd. Nemůžeme však vytvořit podobnou hierarchii pro strukturální vztah **připojený**, protože strukturální vztah **připojený** není nutně transitivní, takže jeho transitivity je neutrální.



Databáze obsahuje složku. Složka

obsahuje soubor.

Soubor obsahuje záznam.

Obr. 14.7 Příklad transitivity. Vztah obsahuje je transitivní, a proto může být rozšířen z jakéhokoli objektu, ze kterého odkaz pochází, na jakýkoli objekt, na který odkaz ukazuje.

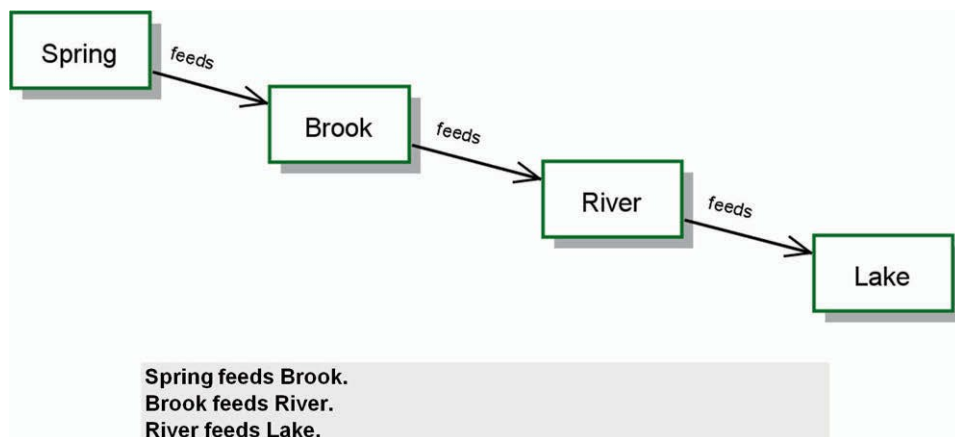
Obrázek 14.8 je dalším příkladem transitivního vztahu. Díky pozitivní transivitě označených strukturálních vztahů je na základě modelu možné odvodit například, že „**pramen napájí jezero**“.

Transitivita je vlastnost strukturálního vztahu, která určuje, zda je strukturální vztah transitivní.

Stejně jako recipocita může být hodnota vlastnosti transitivity strukturálního vztahu *pozitivní*, *neutrální* nebo *negativní*, podle sémantiky značky. Když říkáme, že vztah je transitivní, máme na mysli, že hodnota transitivity strukturálního odkazu je pozitivní.

Vztah **obsažený** na **obr. 14.7** je příkladem strukturálního vztahu, jehož hodnota transitivity je kladná: vždy platí, že pokud **A obsahuje B** a **B obsahuje C**, pak **A obsahuje C**. Jelikož

strukturální relace **obsahuje** je transitivní, říkáme, že hodnota atributu transitivní relace **obsahuje** je kladná. Strukturální relace **uprostřed**, která se několikrát objevuje v úryvku z Medvídka Pů na začátku této kapitoly, je také transitivní.



Obr. 14.8 Symbol transitivní znázorněný na příkladu transitivního vztahu „feeds“

Neutrální transitivní strukturální vztah je strukturální vztah, který může, ale nemusí být transitivní. Jako příklad neutrálního transitivního strukturálního vztahu uvažujme vztah **je přítelem**. Může platit, že pokud **AI je přítelem Bena** a **Ben je přítelem Chena**, pak může platit, že **AI je přítelem Chena**, ale to není zaručeno. Stejně tak, pokud A, B a C jsou uzavřené tvary v rovině, **A se dotýká B** a **B se dotýká C**, může to znamenat, že **A se dotýká C**, ale to není zaručeno. Proto **je** hodnota transitivní **jak je přítelem**, tak se dotýká neutrální.

Příkladem strukturálního vztahu s negativní transitivitou je „**přímo obsahuje**“: Pokud v [obr. 14.7](#) nahradíme vztah „obsahuje“ vztahem „**přímo obsahuje**“, pak pokud „**databáze přímo obsahuje složku**“ a „**složka přímo obsahuje soubor**“, pak není pravda, že „**databáze přímo obsahuje soubor**“. Podobný příklad lze uvést pro [obr. 14.8](#), pokud změníme „feeds“ na „**directly feeds**“.

Jako další příklad uvažujme, že **Jack je otcem Jima** a **Jim je otcem Jill**, pak **Jack není otcem Jill**. Jako poslední příklad z rovinné geometrie uvažujme tři body **A, B, C** a **D** ležící na přímce tak, že **AB, BC** a **CD** jsou tři úsečky. Pokud **se AB dotýká BC** a **BC se dotýká CD**, pak věta **AB se dotýká CD** je vždy nepravdivá. V tomto systému je tedy transitivita strukturálního vztahu **dotýká se** mezi dvěma úsečkami na přímce negativní.

V metamodelu **strukturálního odkazu** jsou tři možné hodnoty jeho vlastnosti **transitivní kladné, neutrální a záporné**. Hodnota **transitivní** je **kladná**, pokud je sémantika značky odkazu transitivní, **záporná**, pokud je sémantika značky taková, že vztah není nikdy transitivní, a **neutrální**, pokud sémantika značky neumožňuje určit, zda je transitivní kladná nebo záporná. Výchozí hodnota **transitivní strukturálního odkazu** je **neutrální**.

Tabulka 14.1 Příklady tří hodnot reciprocity a transitivity

Hodnota			
Atribut strukturálního vztahu	pozitivní	neutrální	negativní
Reciprocita	dotýká se, je propojený, sousedí, je odpojený, je ekvivalentní, je shodný, souvisí, je smíšený	má rád, nemá rád, je lhostejný	je otcem, je nad, obklopuje, nad, nosí, skládá se z
Transitivita	živí, obsahuje, je předkem, obklopuje, nad, smíšený, skládá se z	sousedí, je vedle, je přítelem, drží za ruku	je otcem, přímo obsahuje, přímo krmí

Znalost hodnot reciprocity a transitivity odkazů v modelu může pomoci odvodit vztahy, které mohou přesahovat cíl odkazu, a to lze automatizovat tak, aby se z komplexního modelu generovaly nové znalosti, kde vzdálené vztahy, které mohou přesahovat několik OPD, nejsou zřejmé pouhým vizuálním prozkoumáním modelu. [Tabulka 14.1](#) uvádí několik příkladů pro každou ze tří hodnot dvou vlastností strukturálních vztahů – **reciprocity** a **transitivity**.

Pro zvýšení analytické síly modelu OPM lze hodnoty reciprocity a transitivity odkazu, které nejsou výchozí, uložit do modelu pomocí nástroje pro modelování OPM jako vlastnost (meta-atribut) odkazu, ke které lze získat přístup (např. dvojitým kliknutím na odkaz). Pokud například model OPM obsahuje označený strukturální vztah se značkou „obsahuje“, můžeme uložit skutečnost, že hodnota vlastnosti **transitivity** tohoto vztahu je **kladná**, tj. že „obsahuje“ je transitivní vztah. To umožňuje modelu odvodit, že pokud **A obsahuje B** a **B obsahuje C**, pak **A obsahuje C**.

14.2.3 Nulové značky, nulové strukturální odkazy a jejich výchozí fráze OPL

Značka v jednosměrných i obousměrných označených strukturálních odkazech může být *nulová značka*, tj. prázdná značka. Modely OPM na [obr. 14.9](#) ukazují **svobodu** a **spravedlnost** propojené jednosměrnými (nahore) a obousměrnými (dole) nulovými strukturálními odkazy – strukturálními odkazy s nulovými značkami.

Výchozí nulové tagy jsou nejobecnější; zahrnují *vztah* mezi slovy, jak je definováno níže.

Jednosměrný výchozí nulový tag je výchozí nulový tag pro jednosměrný strukturální odkaz a jeho přidružená fráze vyhrazená OPL je „relates to“ (vztahuje se k).

Obousměrná výchozí značka null je výchozí značka null pro obousměrné strukturální propojení a její přidružená fráze vyhrazená OPL je „související“.

Pro OPD na [obr. 14.9](#) vedou tyto výchozí nulové tagy k OPL větám napsaným na pravé straně [obr. 14.9](#).



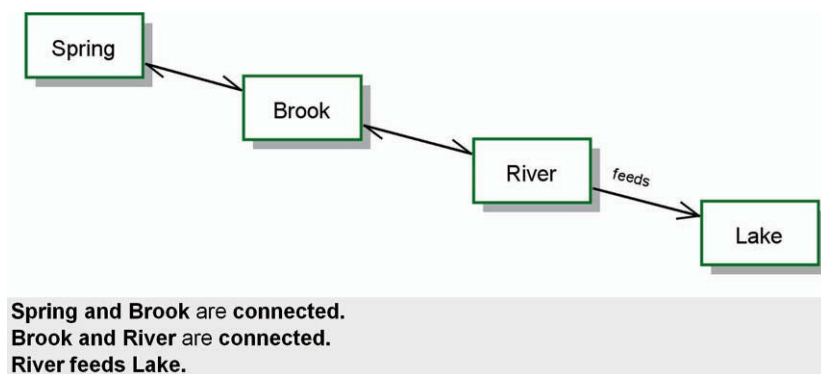
Obr. 14.9 Svoboda a spravedlnost propojené jednosměrným (nahore) a obousměrným (dole) nulovým strukturálním propojením

14.2.4 Modelově specifické nulové tagy

Výchozí nulové tagy může systémový modelář změnit pro každý systémový model zvlášť nebo pro celou společnost či celou doménu.

Štítky null specifické pro model jsou štítky definované modelářem systému, které přepisují výchozí štítky null pro konkrétní systémový model, podnik nebo doménu.

Jelikož existují dva výchozí nulové tagy, jednosměrný výchozí nulový tag a obousměrný výchozí nulový tag, existují také dva odpovídající uživatelsky definované nulové tagy: *jednosměrný nulový tag specifický pro model* a *obousměrný nulový tag specifický pro model*.



Obr. 14.10 Příklad použití obousměrné uživatelsky definované nulové značky connected

Předpokládejme například, že pro systém znázorněný na [obr. 14.9](#) nastaví modelář systému uživatelsky definovanou jednosměrnou nulovou značku na „vede k“ a uživatelsky definovanou obousměrnou nulovou značku na „ekvivalentní“. Dvě věty OPL odpovídající dvěma OPD na [obr. 14.9](#) by pak byly:

Svoboda vede ke spravedlnosti.

Svoboda a spravedlnost jsou rovnocenné.

Jelikož uživatelsky definované nulové tagy jsou ze své podstaty definovány uživatelem, jsou ve výchozím nastavení tučné, ale uživatel je může definovat jako netučné. Minimální rozsah uživatelsky definovaných tagů je celý model OPM, ale jak již bylo uvedeno, lze jej rozšířit na skupinu modelů, podnik nebo celou doménu.

Jedním z nejužitečnějších obousměrných uživatelsky definovaných nulových tagů je **connected**. Předpokládejme například, že v modelu OPM na [obr. 14.10](#) byl obousměrný uživatelsky definovaný nulový tag definován jako **connected**, odstavec OPL by se odpovídajícím způsobem změnil na první dvě věty OPL na [obr. 14.10](#).

14.3 Strukturální vztahy jako procesy zachovávající stav

Mnohé ze značek strukturálních vztahů jsou některé slovesné tvary *procesů zachovávajících stav*, které byly popsány v kapitole 13 výše. Některé příklady jsou **surrounds**, **precedes**, **follows**, **contains**, **holds**, **maintains**, **remains**, **supports**, **owns**, **possesses**, **connects**, **surrounds**, **fastens**, **aggregates** a **comprises**. Jaký je tedy rozdíl mezi těmito slovesy a skutečnými procesy? Rozdíl spočívá v jejich sémantice: procesy zachovávající stav vyjadřují kontinuitu, stabilitu, oddělení od času nebo ustálený stav. Taková slovesa totiž nesplňují test procesu, který stanoví tři podmínky, aby se něco mohlo kvalifikovat jako proces:

(1) *Transformace objektu*: Proces musí transformovat alespoň jeden z objektů v sadě objektů předzpracování.

(2) *Spojení s časem*: Proces musí představovat nějakou událost, výskyt, akci, postup nebo činnost, která se odehrává v časové ose.

(3) *Spojení se slovesem*: Proces musí být spojen se slovesem.

Z těchto podmínek procesy zachovávající stav nesplňují první a druhou podmínku. Sloveso v tagu strukturální relace neprovádí (a nemělo by provádět) žádnou transformaci objektu, a proto se v časové ose neodehrává žádná událost, výskyt, akce, postup nebo činnost. Sloveso představující strukturální relaci má spíše pojem statického, stabilního stavu, který platí, dokud žádný proces nepůsobí na žádný z dotčených objektů.

Proces zachování stavu můžeme chápat jako proces, který působí proti síle, která se snaží změnit status quo. Například podpora fyzického objektu na Zemi je proces, který udržuje objekt na jeho aktuálním místě. Bez této podpory by objekt spadl kvůli gravitaci. Podobně zadržování kapaliny v nádobě zabraňuje jejímu rozlití. Naopak, pokud musí proběhnout nějaký netriviální proces, aby se objekt udržel v klidu proti gravitaci nebo jiné síle, jako je například pohon vrtulníku, aby se udržel na jednom místě, měl by být tento proces výslovně definován.

Stejně sloveso může sloužit jako proces i jako strukturální vztah. Rozlišení mezi skutečným procesem a strukturálním vztahem vyjádřeným slovesem vyžaduje hluboké pochopení sémantiky modelu. Například ve větě OPL „**Dálnice obklopuje město**“ je slovo **obklopuje** strukturálním vztahem. Jakmile je dálnice postavena, je po celou dobu své existence statická a obklopuje město bez ohledu na časový prvek. Ve větě „**Police surrounds House**“ (Policie **obklopuje dům**) je však fráze „**Surrounding**“ (**obklopuje**) procesem, který mění objekt „**House**“ (**dům**) z **neobklopeného** na **obklopený**. Splňuje všechny podmínky testu procesu: vyžaduje činnost **policie**, která se odehrává v časové ose.

14.4 Shrnutí

- *Strukturální vztah* je propojení, spojení nebo asociace mezi dvěma objekty nebo mezi dvěma procesy, které v systému trvá alespoň po určitou dobu.
- Binární strukturální vztah je obousměrný.

- **Strukturální spojení** je šipka s otevřenou hlavou, která představuje binární strukturální vztah v OPD od zdrojového objektu k cílovému objektu.
 - **Strukturální značka** je fráze, která vyjadřuje sémantiku – povahu, význam nebo obsah – strukturálního vztahu mezi dvěma věcmi, které se na tomto vztahu podílejí.
 - **Označený strukturální odkaz** je strukturální odkaz se strukturální značkou zaznamenanou podél odkazu.
 - **Obousměrný značený strukturální odkaz** je kombinací dvou značených strukturálních odkazů v opačných směrech.
 - **Vlastnost** je atribut prvku modelu OPM.
 - **Reciproční strukturální vztah** A je strukturální vztah, pro který platí, že pokud $A \text{ } A \text{ } B$ a $B \text{ } A' \text{ } A$, pak $A = A'$.
 - **Reciproční strukturální odkaz** je obousměrný značkováný strukturální odkaz, ve kterém jsou identické značky vpřed a vzad nahrazeny jedinou značkou reciprocit.
 - **Reciprocita** je vlastnost strukturálního vztahu, která označuje, zda mají jeho dopředné a zpětné strukturální vztahy stejnou sémantiku.
 - **Transitivní strukturální vztah** A je strukturální vztah, pro který platí, že pokud $A \text{ } A \text{ } B$ a $B \text{ } A \text{ } C$, pak platí $A \text{ } A \text{ } C$.
 - **Transitivita** je vlastnost strukturálního vztahu, která určuje, zda je strukturální vztah transitivní.
 - **Hodnoty** vlastností reciprocit a transitivity strukturálního vztahu mohou být *pozitivní*, *neutrální* nebo *negativní*.
 - **Jednosměrný výchozí nulový tag** je výchozí nulový tag pro jednosměrný strukturální odkaz a jeho přidružená fráze vyhrazená OPL je „související s“.
 - **Obousměrná výchozí nulová značka** je výchozí nulová značka pro obousměrné strukturální propojení a její přidružená fráze vyhrazená pro OPL je „related“ (související).
 - **Uživatelé definovaná nulová značka** je značka definovaná modelářem systému, která přepisuje výchozí nulové značky pro konkrétní systémový model, podnik nebo doménu.
- Procesy zachovávající stav** vyjadřují kontinuitu, stabilitu, oddělení od času nebo stabilní stav.

14.5 Problémy

1. Na [obr. 14.2](#) je proces zachovávající stav propojením? Vysvětlete.
2. Jaký je rozdíl mezi strukturální relací a strukturálním propojením?
3. Na [obr. 14.5](#) změňte každý z dvou jednosměrných strukturálních vztahů s tagem na obousměrný.
4. Aktualizujte OPL tak, aby odrážel změny, které jste provedli v odpovědi na předchozí otázku.
5. Do tabulky 15 přidejte tři příklady do každé ze šesti buněk.

6. Uved'te tři příklady procesů zachovávajících stav, které nejsou uvedeny v této kapitole.
7. Napište šest anglických vět, které vyjadřují binární strukturální vztahy. Pro každou větu nakreslete odpovídající OPD a napište věty OPL.
8. Nakreslete tři OPD, které vyjadřují jednosměrné binární strukturální vztahy, a tři OPD, které vyjadřují obousměrné binární strukturální vztahy. Pro každý OPD napište odpovídající OPL věty.
9. Vezměte si knihu, noviny nebo časopis, který máte po ruce, a otevřete jej na náhodné stránce.
 - a. Najděte na této stránce 7–10 strukturálních vztahů.
 - b. Napište jejich OPL věty.
 - c. Nakreslete jejich OPD.
10. Nakreslete OPD skutečného produktu ve vaší domácnosti nebo produktu, který dobře znáte a který má alespoň tři úrovně agregace. Napište odpovídající odstavec OPL (věty OPL, které jsou ekvivalentní OPD).
11. Najděte příklady tří vztahů, jeden s pozitivní transitivity, jeden s neutrální a jeden s negativní transitivity. Nakreslete jejich OPD a napište jejich OPL věty.
12. Když Don zapne svůj notebook pomocí tlačítka napájení, jeho internetový prohlížeč se otevře s URL adresou jeho oblíbeného komiksového webu, který čte každý den. Tlačítko v horní části prohlížeče otevírá jeho e-mailový účet. Po přečtení obou zavře počítač.
 - a. Použijte OPM k modelování tohoto popisu třemi různými způsoby:
 - (1) Používání pouze objektů a strukturálních vztahů;
 - (2) Používání co nejvíce procesů a stavů a pouze procedurálních vztahů; a
 - (3) Používání kombinace obou výše uvedených způsobů.
 - b. Zamyslete se nad třemi modely, které jste vytvořili, a odpovězte na následující otázky.
 - (1) Který model nejméně přesně popisoval systém?
 - (2) Který model obsahuje nejvíce detailů?
 - (3) Který model nejlépe zachycuje to, co Don dělal? Proč?
 - (4) Který model preferujete? Vysvětlete a uveďte, co se vám na každém z nich líbí a nelíbí.

Kapitola 15

Omezení účasti a rozcestí

Rozcestí: místo nebo část, kde se něco, například řeka nebo silnice, rozděluje na více větví.

Dictionary.com

Ve všech dosavadních příkladech a diskusích jsme mlčky předpokládali, že každá věc, ať už objekt nebo proces, se účastní vztahu samostatně, tj. v množství přesně 1. Ve skutečnosti je v OPD zvykem, že pokud není vedle strukturálního odkazu výslovně zaznamenáno žádné množství, považuje se za 1, což je výchozí hodnota. Obecně však můžeme chtít specifikovat určitý počet nebo rozsah čísel instancí stejné třídy věcí, které se účastní vztahu. Podobně naše dosavadní modely mlčky předpokládaly, že proces zahrnuje jednu instanci objektu z každé třídy objektů, ke které je propojen. To je skutečně výchozí nastavení. Někdy je však nutné modelovat skutečnost, že se procesu účastní více než jeden objekt. Omezení účasti v procesu a kardinality vazeb jsou navrženy tak, aby tuto skutečnost zohledňovaly. Poté se obrátíme k další užitečné notaci – vidlici –, která vychází z pozorování, že strukturální vztahy jsou distributivní v smyslu analogickém k distributivnímu zákonu v algebře. To je graficky znázorněno pomocí vidlic, jak je definováno, diskutováno a demonstrováno v této kapitole.

15.1 Strukturální a procedurální omezení účasti

Pokud se na vztahu podílí více než jeden objekt, je třeba to označit pomocí omezení účasti.

Omezení účasti je vlastnost odkazu vyjadřující počet nebo matematický výraz zaznamenaný podél odkazu vedle objektu, který označuje multiplicitu (počet opakování) tohoto objektu v daném vztahu.

Vzhledem k tomu, že vztah a odkaz, který jej označuje, mohou být strukturální nebo procedurální, existují dva odpovídající druhy omezení účasti: strukturální a procedurální.

Strukturální omezení účasti je omezení účasti zaznamenané podél strukturálního propojení.

Procedurální omezení účasti je omezení účasti zaznamenané podél procedurálního odkazu.

Výchozí omezení účasti je 1 a je implicitní. Pokud se tedy na vztahu podílí přesně jedna věc, není třeba specifikovat žádné omezení účasti. Když je omezení účasti na

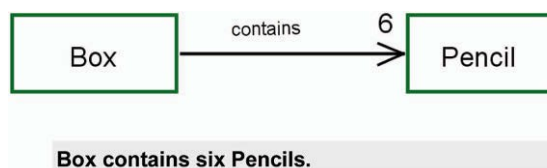
Pokud je cílová strana strukturálního propojení jiná než 1, musí být výslovně uvedena, jak je znázorněno na [obr. 15.1](#) pro strukturální vztah.

15.2 Strukturální omezení účasti

Strukturální omezení účasti mohou být jednostranná nebo oboustranná.

***Jednostranné omezení účasti** je omezení účasti na straně zdrojového nebo cílového spojení.*

Omezení účasti zdroje je jednostranné omezení účasti na straně zdroje spojení. Omezení účasti cíle je jednostranné omezení účasti na straně cíle spojení.



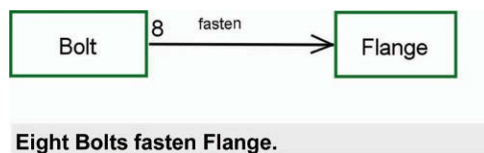
Obr. 15.1 Příklad omezení účasti na straně cíle

OPD na [obr. 15.1](#) je příkladem omezení účasti na straně cíle – označené strukturální spojení, u kterého je omezení účasti na straně cílového objektu (cíle spojení). V tomto příkladu je vyjádřeno konkrétním číslem 6. Cílový objekt **Pencil** v OPD na [obr. 15.1](#) má omezení účasti **6**, zatímco objekt **Box** má implicitní výchozí omezení účasti, které je 1. Pokud je omezení účasti explicitní, jako je tomu u **objektu Pencil** ve větě OPL „**Box contains six Pencils**“ (**Krabice obsahuje šest tužek**), znamená to, že omezení účasti je větší než 1. V tomto případě se při generování věty OPL z OPD číselná nebo symbolická hodnota nebo matematický výraz omezení účasti umístí před název objektu a název objektu se změní na množné číslo.⁽¹⁾

Aby bylo zachováno souladu s anglickou gramatikou, musí sloveso pro jakýkoli tag, včetně tagu null, odpovídat množnému číslu zdrojových a cílových věcí ve větě. Například pokud má zdroj **Bedroom** jednosměrného tagu null omezení účasti 3 a cílem je **Apartment**, věta OPL zní: „**Three Bedrooms** relate to **Apartment**.“ Aby bylo dodrženo pravidlo „psát malá čísla slovy“, číslice (symboly) čísel od nuly do devíti by měly být psány písmeny nebo jako číslice (čísla), takže je vhodnější „**Tři ložnice** souvisejí s **bytem**.“ Proto věta OPL na [obr. 15.1](#) obsahuje **šest** místo 6.

Pokud bychom ignorovali omezení účasti na [obr. 15.2](#), věta OPL by zněla jednoduše „**Šroub upevňuje přírubu**“. Jelikož zdrojový objekt **Šroub** má omezení účasti **8**, zatímco cílový objekt **Příruba** má implicitní výchozí omezení účasti, dostaneme větu OPL na [obr. 15.2](#).

¹ Obvykle to znamená zřetězení písmene **s**, ale program, který generuje věty OPL z OPD, by měl také zohlednit výjimky při převodu podstatného jména z jednotného čísla do množného čísla. OPCAT skutečně zpracovává většinu nepravidelností spojených s množným číslem.



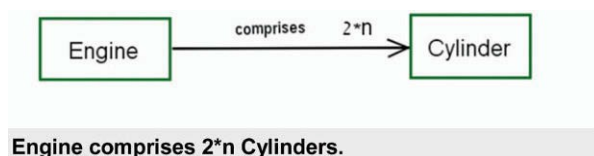
Obr. 15.2 Příklad zdrojového účastnického omezení

15.2.1 Parametrizované strukturální účastnické omezení

Ve výchozím nastavení je účastnické omezení *numerické*, tj. je specifikováno jako číslo, obvykle celé číslo, jak je znázorněno v OPD na [obr. 15.1](#) a [15.2](#). Účastnické omezení však může být také *parametrizované*, tj. může být matematickým výrazem obsahujícím jeden nebo více symbolů.

***Parametrizované omezení účasti** je omezení účasti, které je matematickým výrazem s jedním nebo více parametry.*

[Obrázek 15.3](#) je příkladem parametrizovaného omezení účasti. Zde je n přirozené číslo a modelář vyjadřuje skutečnost, že počet válců v motoru je sudý.



Obr. 15.3 Příklad parametrizovaného účastnického omezení

Pokud jsou čísla, i malá, součástí věty s parametry, jako na [obr. 15.3](#), nebo v rozsahu (jako na [obr. 15.4](#), viz další část), pak jsou čísla vyjádřena jako číslice a ne jako písmena. Syntaxe výrazů omezení účasti a podrobnější příklad parametrizovaných omezení účasti jsou uvedeny v [kapitole 17](#) níže o agregaci-účasti.

15.2.2 Omezení rozsahu účasti

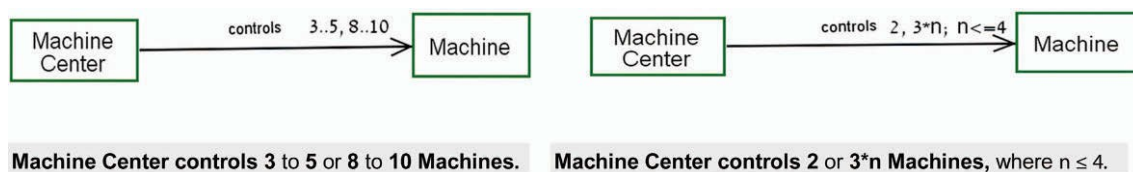
Omezení účasti může být více než jen jedno číslo nebo jeden výraz. Může to být také rozsah.

***Omezení účasti rozsahu** je omezení účasti s dolní a horní hranicí, z nichž každá může být výrazem, na počet možných objektů, které se mohou účastnit relace.*

Složené omezení účasti může být libovolná kombinace čísel, výrazů a rozsahů. Rozsah se označuje jako „ $q_{min}..q_{max}$ “. Jedno číslo nebo parametr lze považovat za zvláštní případ rozsahu s $q_{min} = q_{max}$.

Dvě složené účastnické omezení jsou znázorněny na [obr. 15.4](#). V levém OPD se složené účastnické omezení skládá ze dvou rozsahů. V prvním rozsahu je $q_{min} = 3$ dolní mez a $q_{max} = 5$ horní mez. Tyto dvě veličiny jsou odděleny dvěma po sobě jdoucími tečkami. Druhý rozsah je 8..10. V pravém OPD na [obr. 15.4](#) se složený účastnický omezení skládá z jednoho čísla, 2, a jednoho parametrizovaného rozsahu, $3*n$, kde $n \leq 4$.

Často je q_{min} malé číslo, například 0, 1 nebo 2, zatímco q_{max} je symbol *, který znamená **mnoho**. Symbol * je v omezení účasti „vyhrazeným symbolem“, což znamená, že přesná hodnota „mnoho“ není pevně stanovena jako v algebraické rovnici. Písmeno představuje parametr – konkrétní, ale nespecifikované číslo.



Obr. 15.4 Jednostranná kardinalita s omezením účasti v rozsahu 3..5

15.3 Zkrácené zápisy a vyhrazené fráze

Vyhrazená fráze „ q_{min} až q_{max} “ může být použita pro jakékoli omezení účasti, kde q_{min} i q_{max} mohou být libovolná reálná čísla. Často však dává větší smysl použít jiné fráze, které vyjadřují omezení účasti přirozeněji.

Stejně jako v UML a SysML symbol hvězdička * znamená „mnoho“, takže „0..*“ znamená nula nebo více, nebo jinými slovy „volitelné“, zkráceně *. Rozsah „1..*“, zkráceně +, znamená jeden nebo více a jako vyhrazená fráze OPL: „alespoň jeden“.

Čtyři zkrácené symboly omezení účasti jsou:

- „?“ pro 0..1,
- „*“ pro 0..*,
- „+“ pro 1..1 a
- „+“ pro 1..*.

Každá taková zkratka má odpovídající vyhrazenou frázi OPL. Zkrácené symboly účastnických omezení, jejich meze, vyhrazené fráze OPL a vzorové OPD s odpovídajícími větami OPL jsou uvedeny v [tabulce 15.1](#).

Je také povoleno kombinovat konkrétní hodnoty. Například účastnické omezení „?, 3..*“ je platné a v OPL se překládá jako „**volitelné** nebo alespoň 3“. Ačkoli se všechny dosavadní příklady týkaly objektů, lze je použít i na procesy.

Tabulka 15.1 Zkrácené symboly omezení účasti, jejich meze, fráze a vzorové OPD s odpovídajícími větami OPL

Lower & Upper Bounds $q_{min} .. q_{max}$	Symbol	OPL Phrase	OPD Example & OPL Sentence
0..1	?	an optional	 Car has an optional Sunroof.
0..*	*	optional (+ plural)	 Car is equipped with optional Airbags.
1..1	(none)	(none)	 Car is steered by Steering Wheel.
1..*	+	at least one	 Car carries at least one Spare Tire.

15.4 Kardinálnost

Ve strukturálním vztahu může mít každá hrana spojení – jedna na straně zdroje a druhá na straně cíle – omezení účasti, které je obecně nezávislé na omezení účasti na druhé hraně.

Omezení účasti zdroje je omezení účasti na straně zdroje (strukturálního nebo procedurálního) propojení.

Omezení účasti na straně cíle je omezení účasti na straně cíle (strukturálního nebo procedurálního) propojení.

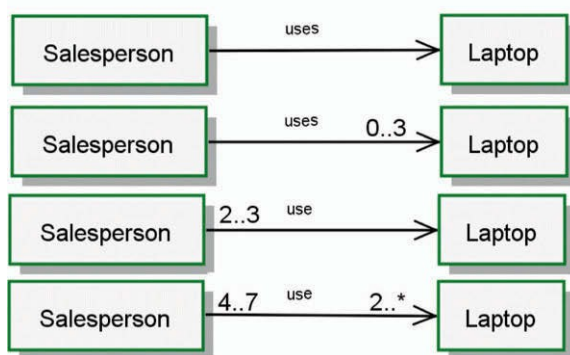
Definice se vztahuje stejně na strukturální i procedurální vazby. Kombinace těchto dvou omezení účasti je *kardinalita* vazby, která se rovněž vztahuje na strukturální i procedurální vazby.

Kardinálnost je vlastnost spojení, jejíž hodnota závisí na kombinaci omezení účasti zdroje a cíle strukturálního spojení.

Kardinalitu označujeme jako $[q_{\min} .. q_{\max}, q'_{\min} .. q'_{\max}]$, kde $q_{\min}$ a $q_{\max}$ jsou dolní a horní mez omezení účasti na zdrojové straně odkazu, zatímco $q'_{\min}$ a $q'_{\max}$ jsou odpovídající parametry na cílové straně odkazu.

15.4.1 Čtyři běžné druhy kardinality

Kardinálnost je důležitým faktorem při návrhu schématu databáze, který se provádí během fáze návrhu vývoje informačních systémů. Různé druhy omezení účasti na obou strukturálních hranách spojení vedou k řadě kombinací. Tradičně se mělo za to, že tyto kombinace vedou ke čtyřem možným druhům kardinálnosti: jedna k jedné, jedna k mnoha, mnoho k jedné a mnoho k mnoha. Ty jsou znázorněny na [obr. 15.5](#).



One-to-one cardinality:

Salesperson uses Laptop.

One-to-many cardinality:

Salesperson uses 0 to 3 Laptops.

Many-to-one cardinality:

2 to 3 Salespersons use Laptop.

Many-to-many cardinality:

4 to 7 Salespersons use 2 to many Laptops.

Obr. 15.5 Čtyři druhy kardinality znázorněné na příkladu

Jak ukazuje horní OPD na [obr. 15.5](#), kardinálnost jedna ku jedné existuje, když není zaznamenáno žádné omezení účasti na žádné straně strukturálního spojení, v takovém případě je oběma stranám přiřazena výchozí hodnota 1. Kardinálnost jeden k mnoha existuje, pokud existuje explicitní omezení účasti s $q_{(\min)} > 0$ a $q_{(\max)} > 1$ přesně na jedné straně strukturálního propojení a 1 na druhé straně. To je znázorněno na druhém OPD na [obr. 15.5](#), zatímco třetí je příkladem kardinálnosti mnoho k jednomu. Konečně, kardinálnost mnoho-k-mnoha existuje, když jsou omezení účasti na obou stranách strukturálního propojení explicitní a v obou případech $q_{(\min)} > 1$, jak je znázorněno v dolním OPD na [obr. 15.5](#).

15.4.2 16 druhů kardinality

Kombinací dvojic čtyř symbolů „?“, „*“, „1“ a „+“ získáme 16 druhů kardinality. Ty jsou uvedeny v [tabulce 15.2](#) v matici 4×4. Buňky pole se čtyřmi obvyklými kardinálnostmi, [1, 1], což je „jeden k jednomu“, [1, +], což je „jeden k mnoha“, [+1], což je „mnoho k jednomu“, a [+ +], což je „mnoho k mnoha“, jsou v pravé dolní části tabulky šedé. Tyto druhy kardinality jsou uznávány v

diagramy vztahů entit (ERD) navržené Chenem (1976), které se používají k navrhování databází. Zde vidíme, že tvoří čtvrtinu z 16 možných kombinací.

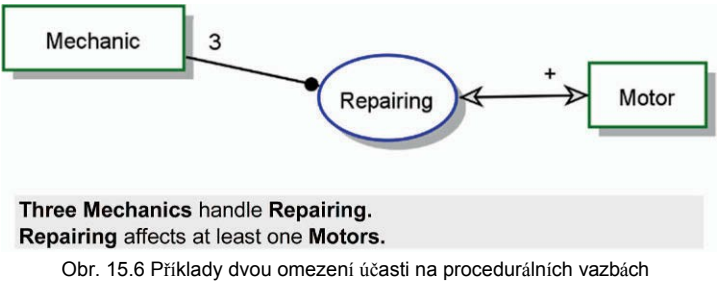
Tabulka 15.2 16 typů kardinality získaných kombinacemi dvojic čtyř typů účastnických omezení

Source symbol Destination symbol	?	*	1	+
?	[?, ?]	[?, *]	[?, 1]	[?, +]
*	[*, ?]	[*, *]	[*, 1]	[*, +]
1	[1, ?]	[1, *]	[1, 1]	[1, +]
+	[+, ?]	[+, *]	[+, 1]	[+, +]

15.5 Procedurální účastnická omezení

Podle definice musí proces transformovat *alespoň* jeden objekt, ale může jich být i více. Navíc chceme být schopni modelovat skutečnost, že aktivátor, ať už je to agent nebo nástroj, je volitelný nebo je vyžadován v určitém množství. Procedurální omezení účasti, definované výše jako číslo nebo výraz zaznamenaný podél procedurálního odkazu vedle zdrojového nebo cílového objektu, označuje mnohost tohoto objektu v daném procedurálním vztahu. Množství procesů se vždy předpokládá jako jedno, takže *vedle konce procesu procedurálního odkazu není žádné omezení účasti*.

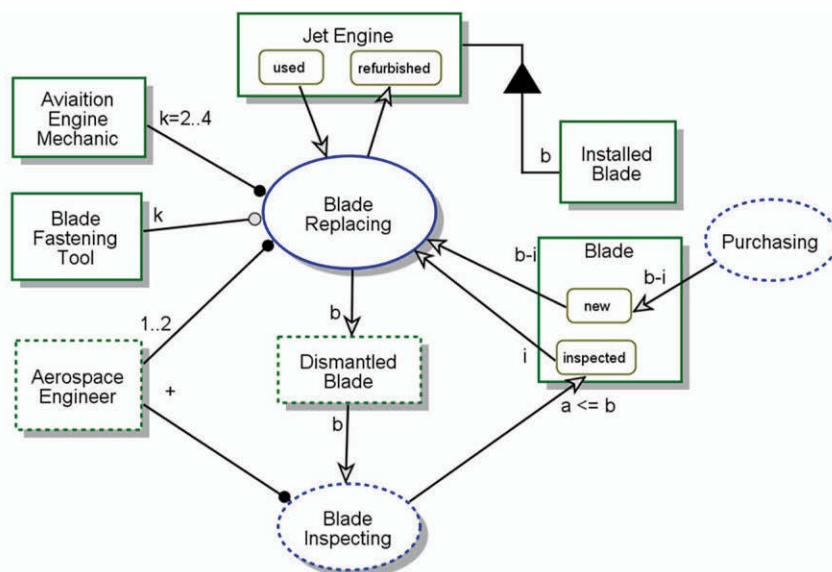
Obrázek 15.6 ukazuje dvě procedurální omezení účasti, jedno na odkazu agenta a druhé na odkazu účinku. Stejně jako u strukturálních omezení účasti symbol+ (plus) znamená „alespoň jeden“, zatímco symbol * (hvězdička) znamená „volitelný“. V souladu s gramatickým pravidlem, že čísla do deseti by měla být v textu psána slovy, nikoli číslicemi, je číslo 3 v OPD v odpovídající větě OPL napsáno jako „tři“. Jedná se o volitelnou konvenci, která se nepoužívá ve spojení s parametry, jak ukazuje **obr. 15.7**.



Obr. 15.6 Příklady dvou omezení účasti na procedurálních vazbách

15.5.1 Parametrizovaná procedurální omezení účasti

Obr. 15.7 ukazuje použití různých omezení účasti v procedurálních vazbách s parametry, rozsahy a omezeními parametrů.



k=2 to 4 Aviation Engine Mechanics handle Blade Replacing.
Jet Engine can be **used** or **refurbished**.
Jet Engine consists of **b** **Installed Blades**.
1 to 2 Aerospace Engineers handle Blade Replacing.
 An optional **Aerospace Engineer** handles **Blade Inspecting**.
Blade can be **inspected** or **new**.

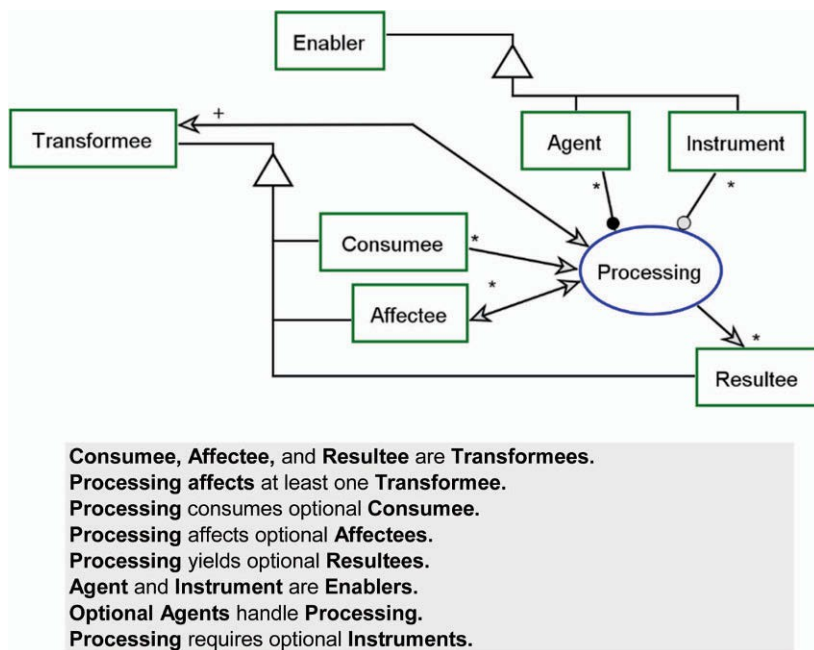
Blade Replacing requires **k** **Blade Fastening Tools**.
Blade Replacing changes **Jet Engine** from **used** to **refurbished**.
Blade Replacing consumes **i** **inspected Blades** and **b-i** **new Blades**.
Blade Replacing yields **b** **Dismantled Blades**.
Blade Inspecting consumes **b** **Dismantled Blades**.
Blade Inspecting yields **a<=b** **inspected Blades**.
Purchasing yields **b-i** **new Blades**.

Obr. 15.7 Omezení účasti v procedurálních vazbách s parametry, rozsahy a omezeními parametrů

Jak uvádí OPD i OPL, v tomto systému **výměny lopatek** má **proudový motor nainstalováno b lopatek**. Proces zajišťují dva až čtyři (počet nastavený na k) **mechanici leteckých motorů**, kteří k tomu používají k **nástrojů pro upevnění lopatek**. Proces **výměny lopatek** zajišťuje také jeden nebo dva **leteckí inženýři**. Tento proces vede k b **demontovaným lopatkám**, které procházejí **kontrolou lopatek**, což je environmentální proces, který vede k počtu a (maximálně b) **zkontrolovaných lopatek**. Proces spotřebuje celkem b **lopatky**, z nichž i je **zkontrolováno** a b-i je **nových**. Toto je počet **nových lopatek** získaných jejich **nákupem**. Tento příklad ukazuje nejen to, jak se parametrizované účastnické omezení používají s procedurálními vazbami, ale také to, jak mohou sloužit k vyjádření *parametrických* omezení – omezení mezi parametry. Lze přidat další omezení. Můžeme například specifikovat, že $i \leq b$, aby se zabránilo získání záporného čísla pro b-i.

15.5.2 Omezení účasti aktivátoru a transformovaného objektu

Proces musí obsahovat alespoň jeden transformátor a může mít jeden nebo více aktivátorů. To je vyjádřeno v OPD na [obr. 15.8](#), který ukazuje metamodel typů objektů zapojených do **zpracování**, rozdělených do **aktivátorů** a **transformátorů**.



Obr. 15.8 Omezení účasti na **aktivátorech** a **transformátorech**

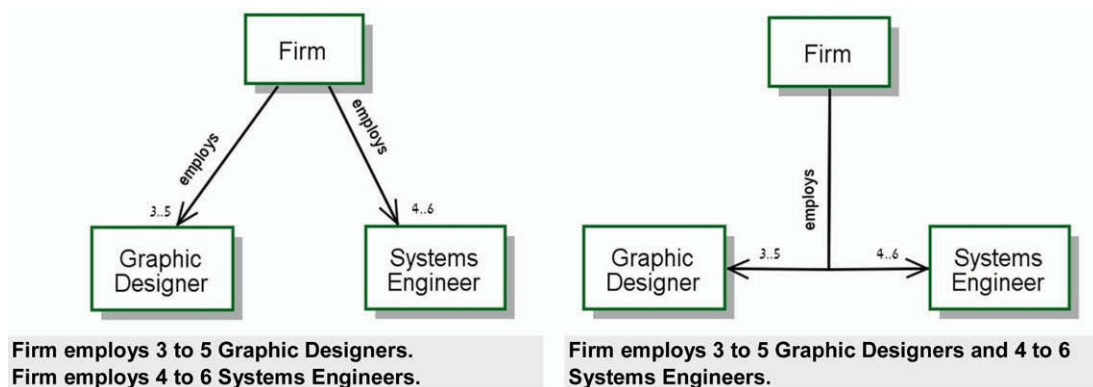
15.6 Distributivní zákon strukturálních vztahů

V algebře, když máme výraz ve tvaru $ab + ac$, můžeme jej rozložit a zapsat jako $a(b+c)$. Podobně platí pro distributivní zákon strukturálních vztahů následující pravidlo.

Pokud A , B a C jsou všechny objekty nebo všechny procesy a A je strukturální vztah, pak $A \ A \ B, A \ A \ C \sqsubseteq A \ A \ (B, C)$.

Nejedná se pouze o zákon v matematice a v OPM, ale jak uvidíme dále, stejný princip lze aplikovat i v přirozených jazycích. Dva OPD na [obr. 15.9](#) poskytují příklad grafické aplikace distributivního zákona strukturálních vztahů. V OPD na levé straně [obr. 15.9](#) jsou dvě disjunktní označené strukturální vazby, obě nesoucí stejnou značku **zaměstnává**. Jedna značka **zaměstnává** je zaznamenána podél vazby od **firmy** k **grafickému designérovi** a druhá podél vazby od **firmy** k **systémovému inženýrovi**. Tento OPD

má přesně dvě grafické věty, z nichž každá vede k jedné větě OPL. Označíme-li vztah **zaměstnává** jako A , **firmu** jako A , **grafického designéra** jako B a **systémového inženýra** jako C , bez ohledu na přidání omezení účasti, je to jako napsat $A A B$, $A A C$.



Obr. 15.9 Distribuční zákon strukturálních vztahů aplikovaný v OPD. Vlevo: disjunktní vazby. Vpravo: spojné vazby

Nejedná se pouze o zákon v matematice a v OPM, ale jak uvidíme dále, stejný princip lze aplikovat i v přirozených jazycích. Dva OPD na obr. 15.9 poskytují příklad grafické aplikace distributivního zákona strukturálních vztahů. V OPD na levé straně obr. 15.9 jsou dvě disjunktní označené strukturální vazby, obě nesoucí stejnou značku **employs**. Jedna značka **employs** je zaznamenána podél odkazu od **firmy** k **grafickému designérovi** a druhá podél odkazu od **firmy** k **systémovému inženýrovi**. Tento OPD má přesně dvě grafické věty, z nichž každá vede k jedné větě OPL. Označíme-li vztah **employs** jako A , **firmu** jako A , **grafického designéra** jako B a **systémového inženýra** jako C , bez ohledu na přidání omezení účasti, je to jako napsat $A A B$, $A A C$.

V OPD na pravé straně obr. 15.9 jsou dva strukturální odkazy označené tagem **employs** spojeny ve svém původu a rozdělují se (odcházejí) někde podél odkazu. Jelikož nyní vychází ze zdrojového objektu pouze jeden strukturální odkaz, stávají se dvě věty OPL jednou. Použijeme-li naši notaci a opět ignorujeme přidání omezení účasti, je to jako napsat $A A (B, C)$. Výrazy představující levý a pravý OPD jsou stejné jako výrazy v algebraickém vzorci distributivního zákona výše (s přidáním omezení účasti). Ve skutečnosti jsou sémanticky ekvivalentní jak graficky, tak textově. Procesy mohou být také spojeny strukturálními vztahy, které jsou distributivní, ale protože použití strukturálních vztahů je mnohem častější u objektů, zaměříme se na objekty.

Graficky má spojení počátku dvou strukturálních odkazů na obr. 15.9, které mají stejnou značku **employs**, stejnou funkci jako algebraické závorky. Závorky v distributivním zákoně $AA(B, C) \sqsubseteq AAB, AAC$ umožňují použít A pouze jednou. Analogicky, společný označený odkaz umožňuje použít **employs** pouze jednou v OPD i v odpovídající větě OPL. A konečně, rezervované slovo OPL and je analogické čárce v distribučním zákonu. Spojení strukturálních vztahů se stejnou značkou vede k rozvětvení, o kterém bude řeč dále.

15.7 Rozvětvení, rukojeť a hrot

V algebře je distributivní zákon $AA(B, C) \sqsubseteq AAB, AAC$ rozšiřitelný na libovolný počet n prvků. Tedy $AA(B_1, B_2, \dots, B_n) \sqsubseteq AAB_1, AAB_2, \dots, AAB_n$. Totéž platí pro OPM a přirozené jazyky. Abychom to vyjádřili v OPM, definujeme níže vidlice.

Fork je kombinace dvou nebo více strukturálních odkazů se stejnou sémantikou vyjádřenou stejnou značkou.

Rozvětvení má společnou spojovací hranu na straně původu odkazu, nazývanou *rukojeť*, která se rozděluje na dvě nebo více hran na straně cíle odkazu, z nichž každá je *hrotem*.

Rukojeť je společná spojovací hrana vidlice na straně původu.

Hrot je rozdělená hrana vidlice na straně cíle.

Rukojeť je věc propojená s rukojetí vidlice.

Tine thing je věc propojená s hroty odbočky.

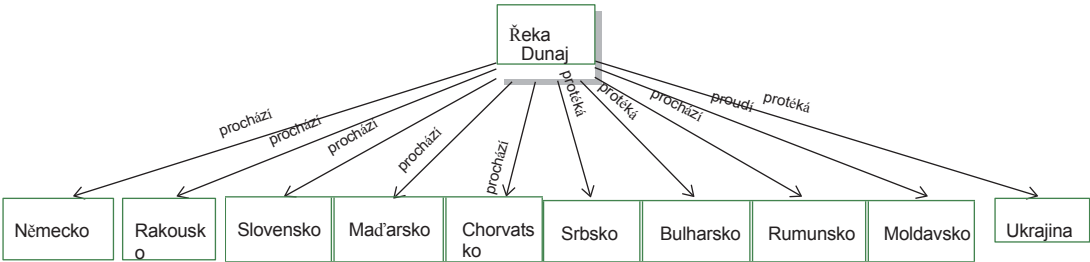
Objektový fork je sada objektů propojených pomocí forku.

Procesní vidlice je sada procesů propojených vidlicí.

Jelikož strukturální vztah existuje mezi objekty nebo mezi procesy, pokud je handle thing objektem, jsou všechny tine things také objekty, a totéž platí pro procesy. Dva OPD na [obrázcích 15.10](#) a [15.11](#) ilustrují hodnotu použití fork vztahů. OPD na [obrázku 15.10](#) obsahuje 10 samostatných strukturálních odkazů, všechny s tagem **passes through**. Je tedy ekvivalentní odstavci OPL na [obr. 15.10](#), který obsahuje 10 vět OPL. Tento odstavec OPL odráží redundanci odkazů v odpovídajícím OPD. Ačkoli je syntakticky a sémanticky správný, odstavec s 10 větami je mechanický, repetitivní a nevhodný pro lidské čtení.

Použití distributivního zákona umožňuje agregovat deset odkazů do vidlice. Použitím výrazu $AAB_1, AAB_2, \dots, AAB_n \sqsubseteq AA(B_1, B_2, \dots, B_n)$ a nahrazením $A = \text{Dunaj}$, $B_1 = \text{Německo}$, $B_2 = \text{Rakousko}$ atd., je výsledek znázorněn na [obr. 15.11](#), kde pouze jeden strukturální odkaz, označený jako **prochází**, vychází z **Dunaje** a rozvětňuje se do deseti vidlic. Odstavec OPL tohoto OPD se zmenší z deseti vět na pouhou jednu větu OPL vidlice – jedinou dokonalou a lidsky čitelnější větu v angličtině. Rukojeť vidlice na [obr. 15.11](#) je segment vycházející z objektu rukojeti **Dunaj**, zatímco každá z 10 vidlic je úsečka s hrotem šipky směřujícím k objektu vidlice (v tomto příkladu pole země) a segment spojující tuto úsečku s rukojetí.

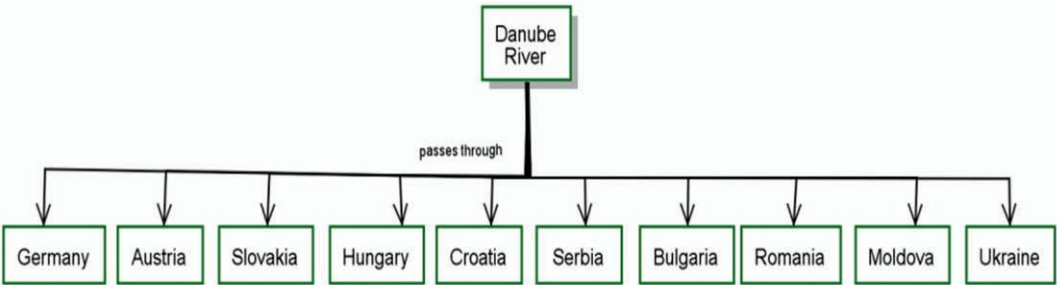
Omezení účasti na různých hrotech se mohou lišit od výchozí hodnoty 1 pro každý objekt hrotu zvlášť. Protože rukojeť je společná pro všechny hroty, je společné i její omezení účasti. Pokud je pro některé spojení vyžadováno jiné omezení účasti na straně rukojeti, musí být toto spojení odděleno od vidlice.



Řeka Dunaj protéká Moldavskem. Řeka Dunaj protéká Rumunskem. Řeka Dunaj protéká Srbskem. Řeka Dunaj protéká Slovenskem. Řeka Dunaj protéká Ukrajinou.

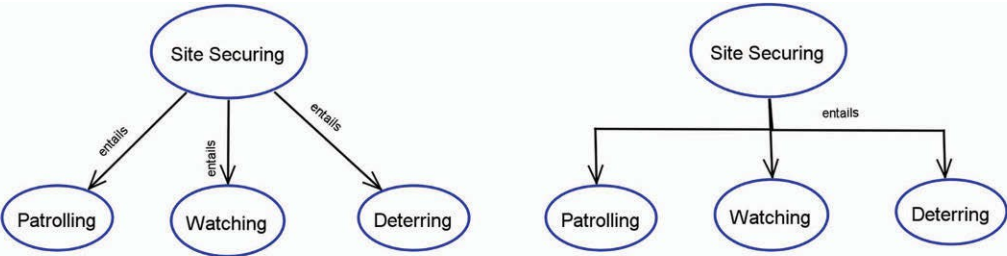
Řeka Dunaj protéká Německem. Řeka Dunaj protéká Maďarskem. Řeka Dunaj protéká Rakouskem. Řeka Dunaj protéká Bulharskem. Řeka Dunaj protéká Chorvatskem.

Obr. 15.10 10 zemí, kterými protéká řeka Dunaj



Danube River passes through Germany, Austria, Slovakia, Hungary, Croatia, Serbia, Bulgaria, Romania, Moldova, and Ukraine.

Obr. 15.11 Deset odkazů označených jako „protéká“ na obr. 15.10 je nahrazeno jednou vidlicí se stejným označením



Site Securing entails Patrolling.
Site Securing entails Watching.
Site Securing entails Deterring.

Site Securing entails Patrolling, Watching, and Deterring.

Obr. 15.12 Příklad rozvětvení s procesy

Obrázek 15.12 je příkladem vidlice, ve které jsou všechny propojené věci procesy, což ukazuje, že všechny věci propojené vidlicí mají stejnou perzistenci: buď jsou všechny objekty, nebo jsou všechny procesy. Je tomu tak proto, že strukturální vztahy existují mezi věcmi stejné perzistence, tj. mezi dvěma objekty nebo mezi dvěma procesy. Pokud je tedy věc rukojetí objektem, jsou i všechny věci hroty objekty a naopak.

15.8 Sada věcí Tine

Množina je abstraktní soubor věcí (nazývaných také *prvky* nebo *členy*). Každá věc v množině je jedinečná.

Sada věcí spojených s vidličkou je sada všech věcí spojených s hroty vidličky.

Objektová množina vidlice je množina všech objektů spojených s hroty vidlice, zatímco *procesní množina* vidlice je množina všech procesů spojených s hroty vidlice.

Sada objektů vidlice označené jako „zaměstnává“ v OPD na [obr. 15.9](#) zahrnuje dva typy povolání, které firma zaměstnává. Sada objektů vidlice označené jako „zaměstnává“ v OPD na [obr. 15.9](#) je {grafický designér, systémový inženýr}. Sada objektů vidlice označené jako „prochází“ v OPD na [obr. 15.11](#) je {Německo, Rakousko, Slovensko, Maďarsko, Chorvatsko, Srbsko, Bulharsko, Rumunsko, Moldavsko, Ukrajina}.

Často se stává, že zobrazení všech větví přetěžuje OPD jak graficky, tak mentálně, jak je tomu v [případě obr. 15.11](#). Pokud je množina objektů výrazně větší než 2, jako v [obr. 15.11](#), může být vhodné vynechat některé objekty v množině objektů, které nejsou relevantní pro to, co má daný OPD sdělit. Vzpomeňte si, že princip OPM pro reprezentaci modelových faktů stanoví, že modelový fakt OPM se musí objevit alespoň v jednom OPD, aby mohl být reprezentován v modelu; objekty, které nejsou relevantní v konkrétním OPD, nemusí být v něm zobrazeny. Podle tohoto principu nemusí každý OPD v sadě OPD systému, který obsahuje věc rukojetí, obsahovat všechny věci v sadě věcí hrotů. Stačí, aby se každý z předmětů typu tine objevil jednou ve vztahu A k předmětu typu handle, aby byl součástí sady předmětů typu tine. To ilustrujeme a rozvedeme, když budeme definovat a diskutovat vlastnosti stupně vidličky a komplexnosti. Jedna OPD může obsahovat jednu podmnožinu sady věcí typu tine, zatímco v jiných OPD, které patří do stejné sady OPD, mohou být jiné podmnožiny věcí spojených s tine skryty, aby se snížila kognitivní zátěž a zlepšila čitelnost diagramu.

Tři vlastnosti vidlice pomáhají vylepšit model OPM: stupeň, komplexnost a uspořadatelnost. Tyto vlastnosti jsou popsány dále v této části.

15.8.1 Stupeň rozvětvení

Velikost (počet prvků) sady objektů hrotů je rovna stupni vidlice.

Stupeň vidlice je vlastnost vidlice, jejíž hodnota je velikost sady objektů hrotů.

Například stupeň vidlice na [obr. 15.9](#), jejíž rukojeť je **pevná**, je 2. Stupeň vidlice z **řeky Dunaj** na [obr. 15.11](#) je 10, protože sada objektů hrotů vidlice označené **prochází** v OPD na [obr. 15.11](#) zahrnuje všech deset zemí, kterými **řeka Dunaj** protéká.²

***Sada hrotů** vidlice je sjednocením sad hrotů vycházejících ze stejné rukojeti a majících stejnou značku ve všech OPD v sadě OPD.*

Předpokládejme například, že jiný OPD v modelu OPM, ke kterému patří OPD na [obr. 15.9](#), má následující sadu objektů o velikosti 4: {**Systémový inženýr**, **Programátor**, **Softwarový inženýr**, **Vedoucí projektu**}. Předpokládejme také, že se jedná o jediné dva OPD v sadě OPD tohoto modelu OPM, kde se objekt **Firma** objevuje se značkou strukturálního odkazu označenou jako **zaměstnává**. Sada objektů tine vidlice označené **jako zaměstnává** by pak byla: *Sada objektů tine (Firma zaměstnává) = {Grafický designér, Systémový inženýr}* u {**Systémový inženýr**, **Programátor**, **Softwarový inženýr**, **Vedoucí projektu**} = {**Grafický designér**, **Systémový inženýr**, **Programátor**, **Softwarový inženýr**, **Vedoucí projektu**}. Stupeň vidlice OPD, který zobrazuje všechna povolání, která firma zaměstnává, je 5 – velikost sady objektů tine.

15.8.2 Komplexnost rozvětvení

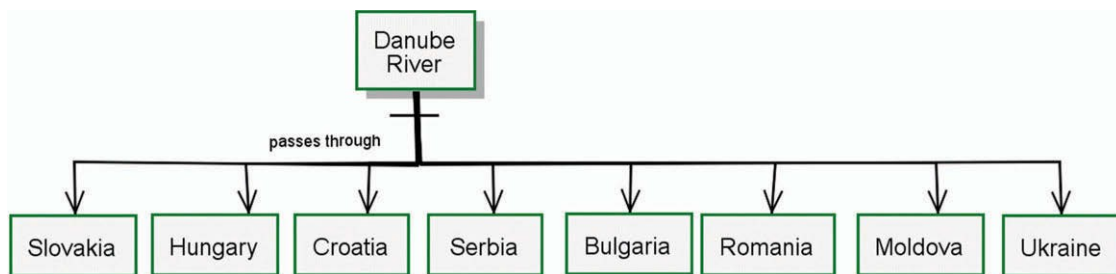
Zatímco vynechání irelevantních věcí pomáhá eliminovat nadměrný nepořádek, který se často vyskytuje v OPD reálných systémů, může také uvést čtenáře jednotlivých OPD v omyl a ten si může myslet, že sada věcí prezentovaná v daném OPD je komplexní, tj. že všechny věci, které lze propojit s rukojetí, jsou skutečně propojeny. Aby se předešlo takovému zmatku, je důležité uvést, zda jsou všechny věci v sadě drobných věcí, které lze propojit s hlavní věcí, skutečně propojeny. Za tímto účelem definujeme hodnotu vlastnosti *úplnosti* vidlice následovně.

***Komplexnost vidličky** je booleovská vlastnost vidličky, která je kladná, pokud jsou všechny věci v sadě hrotů připevněny k rukojeti vidličky, a v opačném případě záporná.*

Jako booleovská vlastnost má **komplexnost** dvě hodnoty: **kladnou**, pokud je vidlička komplexní, tj. všechny věci v sadě věcí jsou připojeny k rukojeti vidličky, a **zápornou** v opačném případě. Pomocí vlastnosti komplexnosti vidličky lze určit, zda je struktura implikovaná vidličkou komplexní nebo nekomplexní. Důležitost komplexnosti vidlice spočívá v tom, že čtenáři diagramu sděluje, zda jsou všechny věci, které mohou být potenciálně spojeny s objektem rukojeti, skutečně spojeny. Nekomplexní vidlice je označena krátkou čarou kolmou k vidlici v blízkosti rukojeti.

Pokračujeme-li v příkladu na [obr. 15.11](#), předpokládejme, že v nějakém OPD chceme zobrazit pouze ty země nebo oblasti, které byly historicky „za železnou oponou“. Při zkoumání OPD na [obr. 15.13](#) vidíme, že **Německo** a **Rakousko** byly odstraněny. Graficky je neúplnost této vidlice označena symbolem neúplné vidlice – krátkou tyčí kolmo k vidlici poblíž rukojeti. Tento symbol neúplné vidlice vyjadřuje skutečnost, že ne všechny země, kterými protéká **řeka Dunaj**, jsou v tomto OPD zastoupeny. Výraz OPL, který vyjadřuje skutečnost, že vidlice je neúplná, je „a další“, který je připojen na konec seznamu objektů vidlice, jak ukazuje věta OPL na [obr. 15.13](#).

² Pro milovníky zajímavostí: Dunaj protéká nejvíce státními hranicemi (askville.amazon.com).



Danube River passes through Slovakia, Hungary, Croatia, Serbia, Bulgaria, Romania, Moldova, Ukraine, and at least one more.

Obr. 15.13 Neúplná vidlice je označena krátkou čarou kolmo k vidlici poblíž objektu rukojeti.

Výchozí hodnota vlastnosti „**Comprehensiveness**“ (Komplexnost) vidlice je **kladná**, což znamená, že vidlice je komplexní a všechny předměty v sadě hrotů vidlice jsou připevněny k rukojeti vidlice. V tomto výchozím případě nebude rukojeť označena symbolem nekomplexní vidlice. Druhá hodnota vlastnosti **Comprehensiveness** je **záporná**, takže vidlice je nekomplexní, což znamená, že sada hrotů je neúplná, protože chybí alespoň jeden hrot. To vyjadřuje vyhrazená fráze OPL „a alespoň jeden další“ na konci věty OPL na [obr. 15.13](#). Neúplná vidlička může být doplněna o chybějící vidličky, čímž se symbol neúplné vidličky odstraní a stav **úplnosti** se změní z **negativního** na **pozitivní**.

15.8.3 Řaditelnost vidlice

Prvky sady obecně a zejména prvky sady hrotů vidlice lze *seřadit* nebo *neuspořádané*. To je určeno vlastností *uspořádatelnosti* vidlice.

Řaditelnost je booleovská vlastnost sady věcí vidlice, která je kladná, pokud jsou věci v sadě věcí vidlice seřazeny, a záporná v opačném případě.

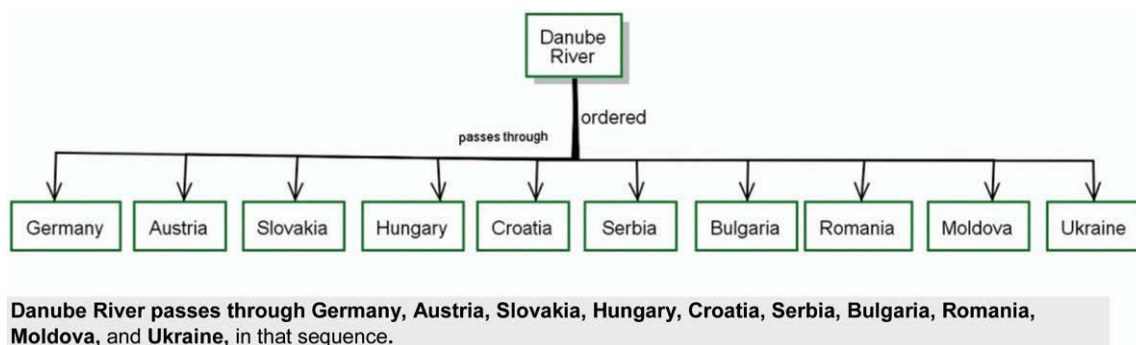
Stejně jako **úplnost** je i **uspořádatelnost** booleovským atributem **sady hrotů vidlice**, jehož hodnoty jsou **kladné** a **záporné**. **Sada hrotů** se **zápornou uspořádatelností** je **neuspořádaná sada hrotů**, což je výchozí nastavení, takže nevyžaduje žádné zvláštní označení.

Pro sadu věcí s kladnou uspořádatelností často (ale ne vždy) existuje nějaký logický vztah A mezi věcmi v sadě věcí $\{T_1 \dots T_N\}$, takže $T^{(i)} A T^{(i+1)}$ pro každé $T^{(i)} \in \{T^{(1)}, T^{(2)}, \dots, T^{(N)}\}$. Například pokud $T_i ; 1 \leq i < N$ je sada N přirozených čísel a A je symbol nerovnosti $<$, pak je uspořádatelnost sady věcí pozitivní. Pokud je sada věcí částmi vědeckého článku {záhlaví, tělo, zápatí}, neexistuje žádné A , které by určovalo toto pořadí.

Sada hrotů s pozitivní uspořádatelností je **uspořádaná sada hrotů**. Pro označení, že sada hrotů vidlice je uspořádaná, se vedle rukojeti vidlice objevuje slovo „uspořádaná“, jak je znázorněno v OPD na [obr. 15.14](#).

15.14. Slovo „uspořádané“ je spíše grafickým symbolem než vyhrazenou frází OPL, protože je součástí OPD stejně jako symbol vidličky, který není úplný.

Jak ukazuje [obr. 15.14](#), vyhrazená fráze OPL pro označení, že sada hrotů je uspořádaná, je „v tomto pořadí“, která se přidává za čárkou na konci věty. Pro neúplnou a uspořádanou vidličku je fráze OPL „a alespoň jeden další, v tomto pořadí“.



Obr. 15.14 Uspořádaná sada věcí vidlice je označena slovem „uspořádaná“ vedle rukojeti vidlice

Aby bylo možné pořadí vyjádřit graficky, musí být věci v sadě věci s hroty uspořádány buď horizontálně zleva doprava, jak je znázorněno na [obr. 15.15](#), nebo vertikálně, shora dolů. Objekty nemusí být uspořádány pěkně, i když je uspořadatelnost sady věcí s hroty pozitivní.

Aby se vyřešila tato potenciální nejednoznačnost, algoritmus řazení uspořádá objekty podle pořadí zleva doprava podle jejich levé strany rámečku objektu (rostoucí souřadnice x) a u těch, které mají stejnou souřadnici levé strany, podle pořadí shora dolů podle horní strany rámečku objektu (klesající souřadnice y , nebo rostoucí, pokud uvažujeme souřadnice pixelů na monitoru). Totéž platí pro procesy, kde rámeček je ten, který obklopuje elipsu procesu.

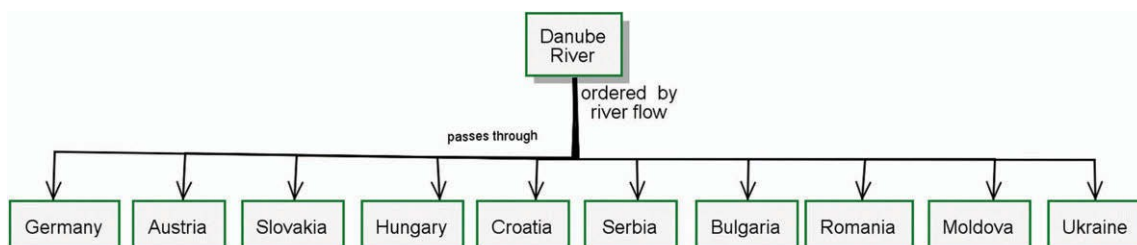
15.8.4 Pravidlo pořadí sady Tine Thing

Pořadí prvků v uspořádané množině může být založeno na určitém pravidle.

Pravidlo řazení je vlastností uspořádané množiny věcí, která v OPD textově specifikuje pravidlo nebo kritérium, podle kterého jsou věci v dané množině uspořádány.

Pravidlo řazení může být nulové, což je výchozí nastavení, nebo jakákoli jiná fráze napsaná malými písmeny. **Pravidlo řazení**, jehož hodnota je nulová, znamená, že neexistuje žádné kritérium řazení, a vedle identifikátoru není napsáno nic (pokud neexistuje žádné řazení) nebo „řazeno“ (pokud existuje řazení, ale pravidlo je triviální, jako například řazení dnů v týdnu).

Pokud je třeba specifikovat pravidlo řazení, použije se v OPD vedle rozvětvení výraz „řazeno podle“ namísto „řazeno“ a pod ním se zaznamená samotné kritérium řazení. Například OPD na [obr. 15.15](#) označuje řazenou sadu vidlic s pravidlem řazení „tok řeky“, což znamená, že země jsou řazeny podle toku řeky Dunaje.



Danube River passes through Germany, Austria, Slovakia, Hungary, Croatia, Serbia, Bulgaria, Romania, Moldova, and Ukraine, ordered by river flow.

Obr. 15.15 Kritérium řazení označené frází „řazeno podle“, pod níž je uvedeno kritérium řazení „tok řeky“

15.9 Shrnutí

- **Omezení účasti** je číslo nebo matematický výraz zaznamenaný podél odkazu vedle objektu, který označuje multiplicitu (počet opakování) tohoto objektu v daném vztahu.
- **Strukturální omezení účasti** je omezení účasti zaznamenané podél strukturálního propojení.
- **Procedurální omezení účasti** je omezení účasti zaznamenané podél procedurálního odkazu.
- **Výchozí omezení účasti** je 1 a je implicitní.
- **Parametrizované omezení účasti** je omezení účasti, které je matematickým výrazem s jedním nebo více parametry.
- **Omezení účasti rozsahu** je omezení účasti s dolní a horní hranicí, z nichž každá může být výrazem, na počet možných objektů, které se mohou účastnit relace.
- **Omezení účasti zdroje** je omezení účasti na straně zdroje (strukturálního nebo procedurálního) spojení.
- **Omezení účasti** na straně cíle je omezení účasti na straně cíle (strukturálního nebo procedurálního) propojení.
- **Kardinálnost** je vlastnost odkazu, jejíž hodnota závisí na kombinaci zdrojových a cílových účastnických omezení strukturálního odkazu.
- **Distributivní zákon strukturálních vztahů**: Pokud A, B a C jsou všechny objekty nebo všechny procesy a A je strukturální vztah, pak platí $A \text{ A } B, A \text{ A } C \square A \text{ A } (B, C)$.
- **Rozvětvení** je kombinace dvou nebo více strukturálních odkazů se stejnou sémantikou vyjádřenou stejnou značkou.
- **Rukojeť** je společná hrana vidlice na straně původu.
- **Hrot** je rozdělená hrana vidlice na straně cíle.
- **Rukojeť** je věc spojená s rukojetí vidlice.

- **Tine thing** je věc spojená s hroty vidlice.
- **Objekt fork** je sada objektů propojených vidličkou.
- **Procesní vidlice** je sada procesů propojených vidlicí.
- **Sada tine thing** vidlice je sada všech věcí spojených s hroty vidlice.
- **Fork degree** je vlastnost vidlice, která určuje velikost sady objektů tine.
- **Komplexnost vidlice** je booleovská vlastnost vidlice, která je kladná, pokud jsou všechny věci v sadě věci spojených s hroty připojeny k rukojeti vidlice, a záporná v opačném případě.
- **Objednatelnost** je booleovská vlastnost vidlice, která je kladná, pokud jsou věci v sadě věci objednány, a záporná v opačném případě.
- **Kritérium řazení** je vlastnost uspořádané sady věcí, která v OPD textově specifikuje kritérium, podle kterého jsou věci v dané sadě věci uspořádány.

15.10 Problémy

1. Modelujte systém, ve kterém se pomocí tří jeřábů zvedá výtah na vrchol nové budovy.
2. Změňte objekty, značku v označených strukturálních vztazích a omezení účasti v každém ze čtyř OPD na [obr. 15.5](#) tak, aby byly získány smysluplné věty.
3. Vyberte z [tabulky 15.2.3](#) z 16 typů kardinality. Pro každý z nich vytvořte OPD, které jej demonstruje.
4. Modelujte knihovnu složenou z n polic, z nichž každá pojme až 20 knih.
5. Pro knihovnu z předchozí otázky vytvořte model procesu **Maximální počet knih Výpočet**, který dělá to, co naznačuje jeho název.
6. Vytvořte model dvou objektových větví s objekty a dvou procesních větví a napište jejich OPL odstavce.
7. Pro každou větev v předchozím problému nakreslete OPD za předpokladu, že distribuční zákon strukturálních vztahů neexistuje. Která možnost je kompaktnější? Proč?
8. Určete sadu tine thing a stupeň vidlice pro každou ze čtyř vidlic v předchozím problému.
9. Pro jednu vidlici objektu nebo jeden proces z předchozí otázky přidejte novou vidlici, jejíž množina hrotů má neprázdný průnik se starou vidlicí.
10. Přidejte symbol neúplné vidlice na příslušná místa ve vidlicích z předchozí otázky.
11. Nakreslete komplexní vidličku dvou vidliček z předchozí otázky.
12. Existuje nějaké potenciální kritérium pořadí v některé ze čtyř vidliček z první otázky? Pokud ano, vyberte jedno a přidejte k němu kritérium řádu. Pokud ne, navrhněte novou uspořádanou vidličku a specifikujte její kritérium řádu.
13. Napište věty OPL pro všechny OPD ve svých odpovědích na otázky v této kapitole.

Kapitola 16

Základní strukturální vztahy

Čtyři strukturální vztahy jsou nejčastější a hrají obzvláště důležitou roli při specifikaci a porozumění systémům. Tyto vztahy, nazývané **základní strukturální vztahy**, jsou:

- **Agregace-účast**, která označuje vztah mezi celkem a jeho částmi,
- **Exhibice-charakterizace**, která označuje vztah mezi exhibitorem – věcí, která vykazuje jednu nebo více vlastností (atributů a/nebo operací) – a věcmi, které exhibitora charakterizují,
- **Generalizace-specializace**, která označuje vztah mezi obecnou věcí a jejími specializacemi, což vede k dědičnosti, a
- **Klasifikace-instancování**, což označuje vztah mezi třídou věcí a instancí této třídy.

Tato kapitola je věnována diskusi o těchto strukturálních vztazích, zatímco následující kapitoly se zabývají každým z nich samostatně.

16.1 Symboly vztahů a účastníci

Vzhledem k převaze základních strukturálních vztahů, aby se zabránilo opakovanému psaní jejich značek a aby byly snadno graficky identifikovatelné, je každému ze čtyř základních strukturálních vztahů přiřazen jedinečný trojúhelníkový symbol. [Tabulka 16.1](#) uvádí základní strukturální vztahy s jejich příslušnými trojúhelníkovými symboly, jak se objevují propojené v OPD, a větu OPL, která odpovídá každému OPD. Zatímco všechny příklady OPD jsou objekty propojené s objekty (s výjimkou **operace B**), jelikož se jedná o strukturální vztahy, čtyři základní strukturální vztahy existují mezi procesy a lze je znázornit také propojením procesů. Nejprve definujeme pojmy „refineable“ a „refinee“.

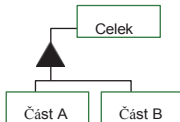
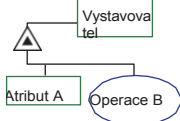
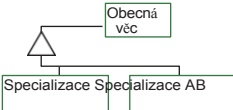
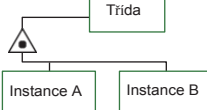
Zdokonalitelné je něco, co lze zdokonalit prostřednictvím základního strukturálního vztahu.

Každá **rafinovatelná věc** je předkem (rodičem) dvouúrovňové hierarchie vyvozené základním strukturálním vztahem. Jak ukazuje [tabulka 16.1](#) v závorkách v levém sloupci, **rafinovatelná věc** může být **celek**, **vystavovatel**, **obecný** pojem nebo **třída**. Každá ze čtyř rafinovatelných věcí odpovídá jednomu ze čtyř základních strukturálních vztahů.

Zjemnitelná věc je věc, která zjemňuje zjemnitelnou věc.

Každý **Refinee** je potomkem (dítětem) dvouúrovňové hierarchie odvozené od základního strukturálního vztahu. [Tabulka 16.1](#) uvádí v závorkách ve druhém sloupci zleva čtyři **Refinees** odpovídající refineables ve strukturálních vztazích: **Part**, **Feature**, **Specialization** a **Instance**. Jak budeme diskutovat později, **Feature** se dále specializuje na **Attribute** (strukturální vlastnost) a **Operation** (procedurální vlastnost).

Tabulka 16.1 Základní názvy strukturálních vztahů, symboly OPD a věty OPL

Název strukturálního vztahu [Název účastníka]		Grafický symbol s použitím OPD	OPL věty	
Vpřed [Vylepšitelné]	Zpět [Vylepšeno]		Vpřed	Zpět
Souhrn [Celek]	Účast [Část]		Celek se skládá z části A a části B.	
Výstava [Vystavovatel]	Charakteristika [Vlastnost: Atribut nebo operace]		Vystavovatel vystavuje atribut A i operaci B.	
Zobecnění [Obecné]	Specializace [Specializace]			Specializace A a specializace B jsou obecné věci.
Klasifikace [Třída]	Instanciacce [Instance]			Instance A a instance B jsou instance třídy.

16.2 Názvy vztahů a věty OPL

Název každé základní strukturální relace se skládá z dvojice slov oddělených pomlčkou.¹ Jak ukazuje [tabulka 16.1](#), první slovo v každé takové dvojici je *název přímé relace*, tj. název relace z pohledu věci výše v hierarchii – předka nebo rodiče – při pohledu dolů v hierarchii. Druhé slovo je *název zpětného (nebo reverzního) vztahu*, tj. název vztahu z pohledu prvku níže v hierarchii – potomka nebo dítěte – tohoto vztahu při pohledu nahoru v hierarchii.

První základní strukturální vztah, agregace-účast, označuje vztah mezi celým objektem a jeho částmi. Výstava-charakterizace označuje vztah mezi objektem a jeho vlastnostmi (atributy a operacemi). Generalizace-specializace označuje vztah mezi obecným objektem a jeho specializacemi. A konečně klasifikace-instancování označuje vztah mezi třídou objektů a instancemi této třídy.

Jelikož jsou úplné názvy těchto vztahů poměrně dlouhé, má každý z nich zkrácenou verzi, která je buď pouze názvem dopředného, nebo zpětného strukturálního vztahu. Zkrácený název, označený v [tabulce 16.1](#) tučným písmem,

¹ Dvojice slov „oddělená pomlčkou“ je dvojice slov oddělených pomlčkou (slovní hříčka).

je vybráno jako významnější z obou: **agregace**, **charakterizace**, **generalizace** a **klasifikace**.

Jak ukazuje [tabulka 16.1](#), všechny čtyři základní symboly strukturálních vztahů jsou rovnostranné trojúhelníky spojené ortonormálními polyliniemi, tj. čarami, jejichž segmenty jsou rovnoběžné s jednou z os diagramu (nazývané také manhattanské čáry). Špička trojúhelníku je spojena ortonormální polylinií s kořenem hierarchického stromu – v našem případě agregátem nebo celkem (například **Celek** v prvním řádku [tabulky 16.1](#)). Základna trojúhelníku je spojena dalšími ortonormálními polyliniemi s každou z částí agregátu (v našem příkladu **část A** a **část B**). Skutečnost, že spojení základních strukturálních vztahů probíhá horizontálně nebo vertikálně, ale ne diagonálně (jako všechna procedurální spojení), pomáhá je vizuálně odlišit od procedurálních spojení. Použití různých barev pro různá spojení, která se kříží (čemuž by se mělo co nejvíce vyhýbat), je také užitečné v přeplněných OPD.

Věty OPL základních strukturálních vztahů jsou také buď v předním, nebo zadním směru. Směr byl podobně určen podle toho, jak přirozeně věta zní v běžné angličtině. Přední směr se používá pro agregaci a charakterizaci:

Celek se skládá z **části A** a **části B**.

Vystavovatel vystavuje **atribut A**, stejně jako **operaci B**.

Zpětný směr se používá pro generalizaci a klasifikaci:

Specializace A a **specializace B** jsou **obecné věci**. **Instance A** a **instance B** jsou instance **třídy**.

Jako obvykle jsou více verzemi těchto dvou vět OPL, které zahrnují tři nebo více upřesnění, následující:

Specializace A, **specializace B** a **specializace C** jsou **obecné věci**.

Instance A, **instance B** a **instance C** jsou instance **třídy**.

16.3 Strukturální hierarchie, transitivita, symboly definované uživatelem

Speciální grafické symboly přiřazené ke čtyřem základním strukturálním vztahům nejsou nijak zvlášť výjimečné, jejich výhodou je však snadné znázornění, absence duplicitních značek a přehlednost diagramů. První tři ze čtyř vztahů mají společnou hierarchii a transitivitu (příklady jsou uvedeny v následujících kapitolách, kde jsou jednotlivé vztahy popsány samostatně):

V **agregaci** může být část celku tvořena ještě menšími částmi, čímž vzniká hierarchie agregace-účasti. Tato hierarchie je transitivní: pokud A sestává z B (a dalších částí) a B sestává z C (a dalších částí), pak A (nepřímě) sestává z C (a dalších částí).

V **charakterizaci** může být vlastnost (atribut nebo operace) projevem vlastností nižší úrovně, čímž vzniká hierarchie projevu a charakterizace. Tato hierarchie je transitivní: pokud A projevuje B a B projevuje C, pak A (nepřímě) projevuje C.

V **generalizaci** může specializace generalizovat specializace nižší úrovně, čímž vzniká hierarchie generalizace-specializace. Tato hierarchie je tranzitivní: pokud A generalizuje B (a případně další specializace) a B se skládá z C (a případně dalších specializací), pak A (nepřímě) generalizuje C (a případně další specializace). Pokud jde o **klasifikaci**, jak je vysvětleno v [kapitole 20](#), instance

může být pouze listem v hierarchii generalizace-specializace. Proto vztah klasifikace-instancování nemůže být přenosný.

Složitější hierarchie lze vytvořit kombinací těchto čtyř vztahů. V souladu s myšlenkou označovat často používané vztahy speciálním symbolem je možné přidat symbol pro jeden nebo více strukturálních vztahů, které jsou široce používány v určité specializované oblasti. Uvažujme příklad z oblasti chemických laboratorních testů průmyslových šarží. V této oblasti je fráze „**je** vzorkem“ velmi častým a užitečným strukturálním vztahem mezi vzorkem a šarží, ze které byl odebrán. V této oblasti lze zavést speciální grafický symbol a odpovídající vyhrazenou frázi „**je** vzorkem“, aby bylo možné rychlejší a snazší modelování. Symbol vybraný v reálném případě při práci prováděné ve společnosti ISCAR Ltd. – podniku působícím v oblasti výroby řezných nástrojů na kov pomocí technologie slinování – byl kousek vyříznutý z dortu, symbolizující, že chuť kousku dortu – vzorku – je stejná jako chuť celého dortu – šarže, ze které byl vzorek odebrán. Čtyři základní strukturální vztahy jsou pro koncepční modelování tak zásadní, že následující kapitoly jsou věnovány diskusi o každém z nich.

16.4 Shrnutí

- Čtyři strukturální vztahy jsou základní, a proto jim jsou přiřazeny grafické symboly.
 - **Zlepšitelný** je něco, co lze zlepšit prostřednictvím základního strukturálního vztahu.
 - **Refínee** je věc, která zušlechťuje zušlechtitelné.
- Čtyři základní strukturální vztahy jsou:
 - Agregace-účast;
 - Exhibice-charakterizace;
 - Generalizace-specializace; a
 - Klasifikace-instancování.
- Každý základní strukturální vztah má jedinečný trojúhelníkový symbol.
- Symbol nahrazuje značku, díky čemuž je OPD graficky přehlednější a rychleji srozumitelný.
- Každý základní strukturální vztah vyvolává hierarchii.
- Kombinací těchto čtyř vztahů lze vytvořit složité hierarchie.

V určitých oblastech mohou být zásadní další strukturální vztahy, pro které lze přiřadit speciální symboly definované uživatelem.

16.5 Problémy

1. U každé položky v [tabulce 16.1](#) uveďte, zda se jedná o refineable nebo refinee.
2. Pro každý OPD v [tabulce 16.1](#) nakreslete alternativní OPD bez použití základního vztahu.
3. Pro každou větu OPL v oddíle [16.2](#) uveďte konkrétní příklad OPL a odpovídající OPD.

Part III

Structure and Behavior:

Diving In

Poté, co jsme v části II stanovili základy a principy modelového systémového inženýrství v OPM i SysML, část III se zabývá jádrem koncepčního modelování. V prvních čtyřech kapitolách této části se podrobně zabýváme podrobnostmi a použitím každé ze čtyř základních strukturálních relací. [Kapitola 17](#) a [18](#) se zabývají agregací-účastí a exhibicí-charakterizací. [Kapitola 19](#) se zabývá stavy a hodnotami, pojmy, které jsou potřebné pro zbývající dvě základní strukturální vztahy – generalizaci-specializaci a klasifikaci-instanci, které jsou podrobně popsány v [kapitole 20](#). [Kapitola 21](#) se týká řízení složitosti. Definuje a popisuje čtyři mechanismy zpřesnění a abstrakce OPM a zároveň se zabývá řízením složitosti v SysML. [Kapitola 22](#) se zabývá operační sémantikou OPM a kontrolními odkazy – způsobem, jakým je řízení spravováno během provádění systému. V [kapitole 23](#) specifikujeme, jak modelovat logické operátory a pravděpodobnosti. [Kapitola 24](#) je přehledem veřejně dostupné specifikace ISO 19450 (PAS) – *Automatizační systémy a integrace – Metodika objekt-proces*, přijaté Mezinárodní organizací pro normalizaci v roce 2014.

Kapitola 17

Agregace – Účast

Celek je více než součet jeho částí.

Aristoteles, *Metafyzika*

Tento velký čtyřkolový vůz ... se skládá z několika částí spojených koženými řemeny a dřevěnými hřebíky. ... Každé ze čtyř velkých kol má 34 paprsků ...

Popis dřevěného a koženého vozu,
východní Altaj, Rusko (Státní
muzeum Ermitáž, 2001)

Tato kapitola pojednává o prvním základním strukturálním vztahu, který je pravděpodobně nejdůležitější: agregace-účast – vztah mezi celkem a jeho částmi. Jakýkoli zajímavý systém lze popsat jako celek rozložený na části. Systém jako celek a kterákoli z jeho částí lze pak popsat samostatně pomocí adjektiv v přirozeném jazyce, která přiřazují atributové hodnoty objektům, a adverbia, která přiřazují atributové hodnoty procesům. Bez schopnosti mentálně rozložit věci na části a prozkoumat jejich vlastnosti by naše schopnost studovat systémy byla značně omezena. Agregace-účast je také známá jako celek-část (Coad a Yourdon 1991), kompozice (Kilov a Simmonds 1996) nebo vztah části-celku (Fowler 1996).

17.1 Základní pojmy

Agregace-účast je základní strukturální vztah, který označuje skutečnost, že rafinovatelná věc – relativně vysoká úroveň, předek, rodič (objekt nebo proces) agreguje (tj. sestává z, skládá se z, obsahuje nebo zahrnuje) jednu nebo více rafinovaných věcí – nižší úroveň, potomek, dítě. Věc vyšší úrovně se nazývá *celek* nebo *agregát*, zatímco věci nižší úrovně, které jej tvoří, jsou *části*. Tento vztah je velmi důležitý v koncepčním modelování a alespoň na povrchní úrovni je relativně snadno pochopitelný. Agregace-účast je prostředkem k popisu složení každé netriviální věci výčtem jejích částí v hierarchii celek-část.

17.1.1 Gestaltova teorie

V souvislosti se slavným výrokiem připisovaným Aristotelovi, který je uveden výše, že „*celek je více než součet jeho částí*“, Koffka (1935, s. 176) přeformuloval tuto pozorování takto: „*celek je něco jiného než součet jeho částí*“ a argumentoval, že operace sčítání je často bezvýznamná, ale co je v celku vždy významné, jsou jeho vztahy s jeho částmi.

V roce 1924 Wertheimer a Reizler (1944) představili gestaltovou teorii, která v zásadě tvrdí, že „*to, co se děje v celku, nelze odvodit z charakteristik jednotlivých částí*“ a že to, co se děje s částmi celku, je určeno zákony týkajícími se struktury tohoto celku. Konfigurace nebo vzorec prvků v jakékoli oblasti je sjednocen jako celek natolik, že jeho vlastnosti nelze odvodit z jednoduchého součtu jeho částí. V psychologii poskytli Rescher a Oppenheim (1995) koncepční rámec pro přesné vysvětlení gestaltového pojmu „celek“ a shrnuli intuitivní požadavky nebo podmínky pro hovoření o celku a jeho částech:

Celek musí mít nějakou vlastnost, která vyplývá z jeho statusu jako celku, vlastnost, která je mu vlastní a charakteristická pro něj jako celek. Části celku musí být ve vzájemném zvláštním a charakteristickém vztahu závislosti; musí splňovat nějakou zvláštní podmínku, která vyplývá z jejich statusu jako částí celku.

17.1.2 Holismus a emergentismus

Abychom mohli specifikovat pojem části, je nutné hned na začátku stanovit podmínky, za kterých lze nějaký objekt považovat za část jiného celku. Specifikace konkrétního vztahu mezi částí a celkem tedy určuje pro danou věc, celek, které věci jsou jejími částmi. (Latimer a Stevens 1997). Z pohledu systému je „zvláštní podmínkou“, kterou musí věci jako části celku splňovat, *holismus*. Holismus je zase podmínkou *emergentnosti* – emergentní funkce systému, která vyplývá z konkrétních vztahů mezi celkem a částmi a ze způsobu, jakým jsou části agregovány, a kterou žádná z částí sama o sobě nevykazuje.

Máme tendenci vnímat agregaci jako vztah mezi celým objektem a jeho částmi. Ve skutečnosti se jedná o obvyklý kontext. Na rozdíl od většiny myšlenkových proudů, které holismus připisují částem objektu a celek agregovanému objektu, se v OPM používá stejný vztah s analogickou sémantikou pro procesy i pro objekty: Proces může sestávat z částí, které jsou podprocesy celého procesu, a výsledek agregovaného procesu není pouhým součtem výsledků jeho podprocesních částí, ale procesem s emergentní transformací objektu, kterou by žádný z jeho podprocesů samostatně ani jejich jednoduchý „aritmický“ součet nemohl poskytnout.

Objekt OPM tedy může sestávat z jiných objektů nižší úrovně (a vykazovat, ale nesestávat z procesů, které jsou jeho operacemi, jak diskutujeme v kapitole 18). Analogicky může proces OPM sestávat z jiných procesů nižší úrovně (a vykazovat, ale nesestávat z objektů, které jsou jeho atributy, jak diskutujeme v kapitole 18). Tuto myšlenku rozvedeme při diskusi o zpřesnění v kapitole 21.

17.1.3 Hloubka dekompozice

Otázka, která se při modelování často objevuje, zní: Jak daleko by měla dekompozice sahát? Jak hluboko by měla pokračovat? Přirozeně lze většinu věcí rozložit dále, než je nejhlubší dekompozice specifikovaná v modelu systému. Zejména fyzické objekty lze rozložit až na molekulární a atomovou nebo dokonce subatomovou úroveň. Specifikace ještě hlubších úrovní hierarchie účasti by však měla vždy skončit v bodě, který modelář, architekt, inženýr nebo analytik systému považuje za dostatečný pro účely specifikace systému, který se vyvíjí nebo studuje. Tato úroveň detailů musí být dostatečná k vysvětlení funkce, struktury a chování studovaného systému (jak je to typické ve vědě) nebo k předepsání postupu při jeho podrobném návrhu (typicky v inženýrství).

17.1.4 Proč používat „skládá se z“ a ne „má“?

Některé rané objektově orientované (OO) metody označovaly relaci agregace-účast jako relaci „má“ (na rozdíl od relace „je“, která se týká relace generalizace-specializace, což je předmětem kapitoly 20). Může se skutečně zdát přirozené použít nějakou formu slovesa „mít“ k označení vztahu mezi celkem a jeho částmi, jako v „*Auto má karoserii, motor a čtyři kola.*“ Vyhýbáme se však použití tohoto slovesa k označení agregace, protože je přetíženým slovem a může mít¹⁾ různé výklady. Abychom to pochopili, stačí se podívat na příklady vět „*Dave má nevlastní matku*“, „*Jack má žlutou motorku*“, „*Vedeme diskusi*“, „*Mám potíže s porozuměním*“, „*Pacient má nachlazení*“ a „*Objekt má atribut*“. Volba OPM pro vyhrazenou frázi označující vztah agregace-účast je vysvětlena níže.

17.2 Agregace-účast jako rozcestí

Stejně jako všechny strukturální vztahy je i agregace-účast dvojicí dopředných a zpětných strukturálních vztahů. *Agregace* je dopředný strukturální vztah – vztah viděný z pohledu agregátu, celku nebo předka, když odkazuje na své části – potomky. Zpětný strukturální vztah, tj. vztah viděný z pohledu každé části, je *účast*.

Agregace a účast jsou inverzní vztahy: agregaci lze považovat za proces vytváření celku z jeho částí, zatímco účast je jednou z částí, které tvoří agregát. Jak jsme však poznamenali v diskusi o strukturálních vztazích v oddíle 14.3, *agregace* je proces zachovávající stav. Jeho sémantika spočívá v tom, že části jsou drženy pohromadě, aby vytvořily celek, přičemž čas má pro tento vztah malý nebo žádný význam.

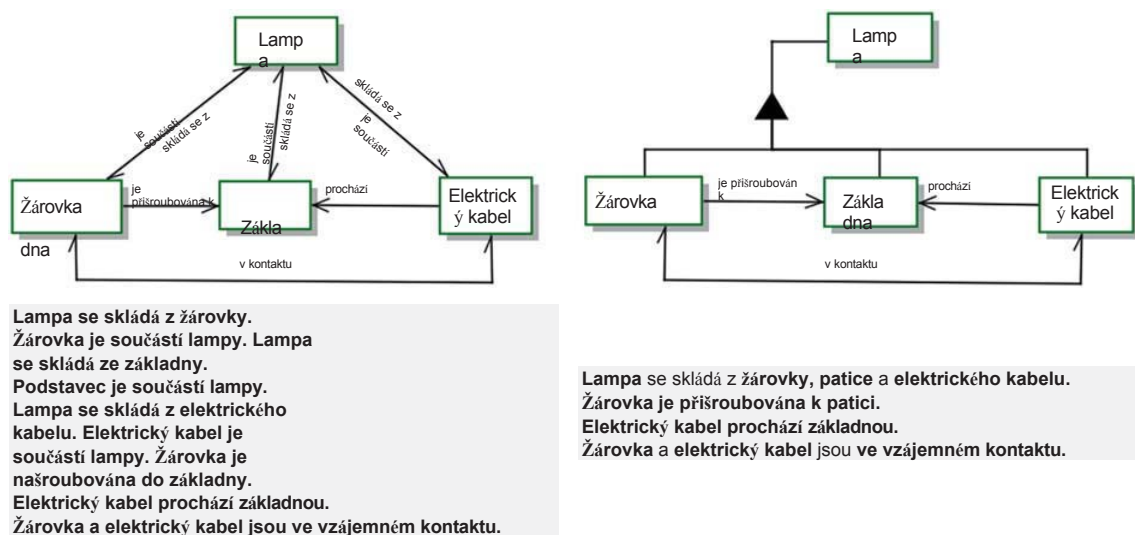
Směr vpřed (nebo hierarchicky dolů) vztahu agregace-účast, od celku k jeho částem, je směr *agregace*. Vyhrazený výraz používaný k vyjádření směru vpřed tohoto vztahu je „skládá se z“. Zpětný (nebo vzhůru směřující, nebo opačný) směr, od jednotlivých částí k celku, je směrem *účasti*. Fráze používaná k vyjádření zpětného směru vztahu je „je součástí“, ale nejedná se o frázi vyhrazenou pro OPL.

Dvě OPD na obr. 17.1 ilustrují, jak vztah agregace-účast nahrazuje označený strukturální vztah. V OPD vlevo jsou vztahy mezi **lampou** a jejími třemi částmi vyjádřeny pomocí tří obousměrných označených strukturálních odkazů, jak jsme dosud používali. Všechny tři přední značky jsou „skládá se z“, přičemž zdrojový objekt **Lampa** je celek a cílové objekty – **Podstavec**, **Žárovka** a **Elektrický kabel** – jsou části. Značka v opačném směru pro každé z těchto tří spojení je „je součástí“. Získáme tak šest vět OPL – jednu vpřed a jednu vzad pro každé ze tří spojení.

Fráze „skládá se z“ v odstavci OPL na levé straně obr. 17.1 je tučně vyznačena, protože není vyhrazená – pochází z uživatelsky definované značky, umístěné na obousměrném značkováném strukturálním odkazu, nikoli ze speciálního agregačního symbolu. Opak platí pro vyhrazenou frázi „skládá se z“ v OPL

¹⁾ Již v této větě, stejně jako v této poznámce pod čarou, máme zabudovaný příklad, který ukazuje více použití „mít“.

odstavec na pravé straně [obr. 17.1](#), ve kterém jedna věta OPL „Lampa se skládá z žárovky, patice a elektrického kabelu“ nahrazuje šest vět OPL na levé straně.



Obr. 17.1 Agregace vyjádřená třemi označenými strukturálními odkazy (vlevo) a symbolem agregace-účasti (vpravo)

Černý plný trojúhelník▲ — symbol agregace-účasti — nahrazuje dvojici dopředných a zpětných textových značek obousměrného strukturálního odkazu, které textově vyjadřují vztah agregace-účasti. Stejně jako ostatní základní symboly strukturálních vztahů je i symbol agregace-účasti užitečnou zkratkou pro tento důležitý a široce používaný strukturální vztah. Symbol pomáhá snadno identifikovat vztah v OPD, čímž šetří grafický nepořádek a nadměrné psaní a čtení textu.

Jelikož se jedná o strukturální vztah, vztah agregace-účasti se řídí distributivním zákonem, dva nebo více strukturálních odkazů lze znázornit jako rozvětvení. V OPD vpravo na [obr. 17.1](#) jsou vztahy mezi **lampou** a jejími třemi částmi vyjádřeny pomocí specifického symbolu určeného pro vztah agregace-účasti, černého rovnostranného trojúhelníku▲, jehož základna je vodorovná. Celek je propojen s vrcholem trojúhelníku a částí s jeho základnou. To umožňuje nahradit prvních šest vět OPL vlevo jednou jedinou větou – první vpravo. Na rozdíl od značky „**skládá se z**“ na [obr. 17.1](#), která je tučně vyznačena, protože je definována uživatelem, fráze „**skládá se z**“ na [obr. 17.1](#) je vyhrazená fráze OPL, a proto není tučně vyznačena.

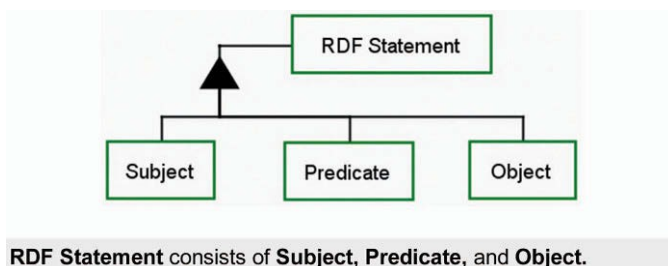
17.3 Příklad sémantického webu

RDF, Resource Description Framework (W3C Consortium 2014), integruje řadu aplikací od knihovních katalogů a celosvětových adresářů po syndikaci a agregaci zpráv, softwaru a obsahu až po osobní sbírky hudby, fotografií a událostí pomocí XML jako syntaxe pro výměnu dat. Specifikace RDF poskytuje lehký ontologický systém na podporu výměny znalostí na webu.

Příklad použití základního strukturálního vztahu agregace-účast lze najít v následujícím výňatku z oddílu 2.2 příručky RDF Primer (Manola a Miller 2004):

„...každé tvrzení se skládá z předmětu, přísudku a objektu.“

Použití fráze „*skládá se z*“ jasně naznačuje existenci vztahu celku a části, resp. agregace a účasti mezi celkem a jeho částí. Jak jsme viděli, OPM tuto frázi používá jako vyhrazený výraz pro označení tohoto vztahu. OPD, které je ekvivalentní této větě OPL (a mělo by být z ní generováno jakýmkoli nástrojem podporujícím OPM, jako je OPCAT), je znázorněno na [obr. 17.2](#). Tato věta OPL je skutečně téměř identická s původní větou výše.



Obr. 17.2 OPD věty „RDF Statement consists of Subject, Predicate, and Object“ (RDF prohlášení se skládá z předmětu, predikátu a objektu)

Černý trojúhelník, který označuje základní strukturální vztah agregace-účast, má svůj vrchol spojený s agregátem (celkem, kterým je objekt **RDF Statement**), zatímco jeho (vždy vodorovná) základna je spojena se třemi částmi: **Subject**, **Predicate** a **Object**. Jedná se o vidličku, ve které je **RDF Statement** objektem rukojeti a množina {**Subject**, **Predicate**, **Object**} je množinou objektů hrotů. Kdyby vidlice neexistovaly, OPD by vyžadovalo tři samostatné agregační odkazy, každý s vlastním symbolem černého trojúhelníku. Jak brzy uvidíme na [obr. 17.4](#), protože UML a SysML nemají pojem vidlice, potřebovali bychom skutečně tři samostatné agregace (symboly diamantu), abychom vyjádřili stejné tři modelové skutečnosti.

17.3.1 Různé fráze, stejná sémantika

V případě výše analyzovaného **RDF prohlášení** jsme měli štěstí, že jsme zjistili, že fráze v přirozeném jazyce „*...každé prohlášení se skládá z předmětu, přísudku a objektu*“ obsahovala vyhrazenou frázi OPL „*skládá se z*“. Díky tomu bylo snadné odvodit, že mezi objektem **RDF prohlášení** a sadou dalších objektů existuje vztah celku a části. Existuje však mnoho dalších syntaktických výrazů se stejnou sémantikou celku a části. Mezi ně patří „*má části*“, „*sestává z*“, „*je tvořen*“ a „*zahrnuje*“. Jiné výrazy, jako „*je rozdělen na*“, „*tvoří*“ nebo „*obsahuje*“, mohou být při určitém výkladu také považovány za výrazy se stejnou sémantikou vztahu celku a části, zatímco v jiném kontextu mohou mít poněkud odlišný význam. Uvažujme například následující definici trojice RDF, kterou najdeme v oddíle 3.1 navrhované doporučení W3C *Resource Description Framework (RDF): Concepts and Abstract Syntax* (Klyne et al. 2004):

„Každá trojice má tři části: subjekt, objekt a predikát (nazývaný také vlastnost), který označuje vztah.“

V oddíle 6.1 stejného dokumentu najdeme:

*„RDF trojice obsahuje tři složky: **subjekt**, který je *odkazem RDF URI* nebo *prázdným uzlem*, **predikát**, který je *odkazem RDF URI*, a **objekt**, který je *odkazem RDF URI*, *literálem* nebo *prázdným uzlem*.“*

Porovnáním těchto dvou úryvků ze stejného dokumentu musíme dojít k závěru, že „*má tři části*“ má přesně stejný význam jako „*obsahuje tři složky*“, protože oba se vztahují ke složení nebo struktuře trojice RDF. Pokud navíc přijmeme, že sémantika sloves „obsahuje“ a „*má části*“ v tomto kontextu je stejná jako „*skládá se z*“, můžeme výše uvedené dvě citace shrnout do následující věty v OPL:

RDF trojice se skládá z subjektu, predikátu a objektu.

Problém více slov, idiomů nebo frází, které mají stejnou nebo téměř stejnou sémantiku, jak je zde demonstrováno, je významnou otázkou v oblasti zpracování a porozumění přirozenému jazyku (NLP). Lidé díky své přirozené inteligenci obvykle nemají problém přiřadit stejnou sémantiku tak odlišným syntaktickým entitám a pochopit jemné rozdíly, pokud existují a jsou relevantní. Výše uvedený příklad ukazuje, že i ve velmi formálních dokumentech, jako je dokument definující sémantický web, v němž je sémantika předmětem diskuse, se volně používají ekvivalentní idiomy a fráze, přičemž se oprávněně spoléhá na lidskou inteligenci, která tento problém vyřeší.

Lidé skutečně neustále bez námahy interpretují smysluplné věty, aniž by si vůbec všímali, že k vyjádření stejného významu byla použita jiná slova a zcela odlišná syntaxe. V případě technik NLP se tato otázka stává velmi důležitou a je třeba ji pečlivě řešit. OPL tento problém řeší tím, že je podmnožinou angličtiny, která je formálně definována pomocí kontextově nezávislé gramatiky. Budoucí vývoj v oblasti automatizovaného sémantického porozumění větám může být klíčový pro modelování evoluce modelů OPM založených na skutečných datech a ověřených lidmi, jako je například model vyvinutý Somekhem et al. (2014) pro životní cyklus mRNA.

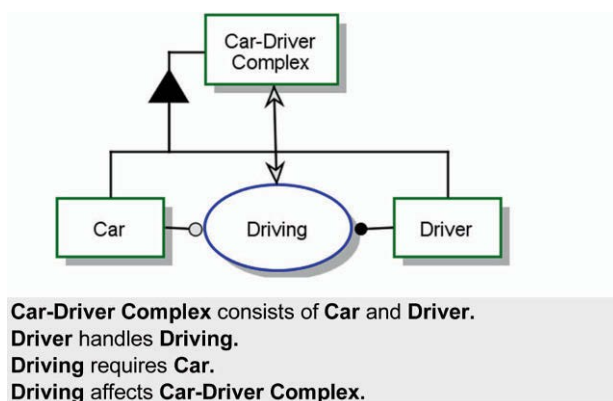
17.4 Pojmenování agregátů

Během analýzy se často setkáváme se situacemi, kdy potřebujeme pojmenovat agregát, který nemá v přirozeném jazyce žádné jediné slovo. Pro ilustraci pojmenování agregátu a významu generování vhodných frází uvažujme vývojový tým v oblasti dopravy, stavebnictví a systémového inženýrství, jehož úkolem je zlepšit dopravu ve městě. Po určité úvaze a diskusi se tým shodne, že základním objektem v systému je soustava automobilu a osoby, která jej řídí v ulicích města. Tento objekt je pro systém mnohem důležitější než samotný automobil nebo samotný řidič.

Role automobilu bez řidiče se omezuje na otázky parkování, zatímco řidič bez automobilu by měl být považován za chodce. Nicméně i když se tým shodl, že automobil spolu s řidičem je hlavním objektem, který je třeba v systému zohlednit, stále mu chybí elegantní způsob, jak jej označit. Jelikož v angličtině (a pravděpodobně ani v žádném jiném přirozeném jazyce) neexistuje pro tento objekt jediné slovo, tým přišel s názvem **Car-Driver Complex (komplex automobil-řidič)**, jak je znázorněno na [obr. 17.3](#). Jak uvidíme, tyto situace nejsou specifické pouze pro agregáty; vyskytují se také v řadě jiných okolností, jako například

jako pojmenování atributu, když jsou explicitní pouze názvy jeho hodnot.² V takových případech musíme uplatnit svou kreativitu a vytvořit vhodnou frázi, která nejlépe vystihuje podstatu toho, co chceme vyjádřit.

Schopnost vymýšlet smysluplná jména nebo vytvářet výstižné fráze je velmi důležitou součástí procesu analýzy. Poskytuje nám schopnost abstrahovat do celku soubor věcí, které by jinak bylo velmi obtížné vnímat a vztáhnout k sobě jako k jednotě. Vzpomeňte si, že první princip OPM – princip OPM „funkce jako semínko“ – vyžaduje zahájit proces modelování systému definováním, pojmenováním a popisem funkce systému. Název funkce musí vyjadřovat, k čemu je systém určen a jakou hodnotu jeho uživatelé získají jeho používáním.



Obr. 17.3 Pojmenování agregátu, který nemá v přirozeném jazyce žádné jediné slovo

17.5 Složené a sdílené agregace v UML a SysML

SysML převzal z UML 2 všechny definice související s diagramem tříd (a několika dalšími druhy diagramů) „tak, jak jsou“. Blokový diagram SysML zdědil stejnou sémantiku jako diagram tříd UML 2. V diagramech tříd UML 2 a SysML proto existují dva typy agregace: *kompozitní agregace* a *sdílená agregace* (Object Management Group 2010, s. 39).

- **Složené agregace**, znázorněné jako černý kosočtverec vedle celého konce spojení (viz [obr. 17.4](#)), „označují, že složený objekt má odpovědnost za existenci a ukládání složených objektů (částí)“. Složené agregace, také označované jako *silné agregace*, *kompoziční vztahy*, *standardní složené agregace* nebo *nesdílené asociace*, jsou považovány za „silnou“ formu obsahu nebo agregace: Část může patřit pouze k jednomu agregátu a pokud je agregát spotřebován, jsou spolu s ním spotřebovány i všechny jeho části. Původně definované pro UML, *odpovědnost* a *ukládání* v definici kompozitní agregace jsou pojmy související se softwarem. SysML, který má pojmut systémy jakéhokoli druhu, nejen software, převzal tuto definici, stejně jako mnoho jiných definic.

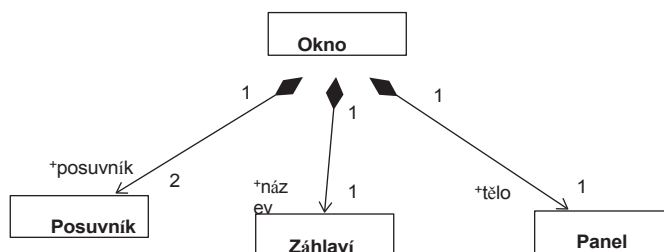
² Jak se například nazývá atribut, jehož hodnoty jsou **široké** a **úzké**? **Šířka**? **Úzkost**? Něco mezi tím? Takové neutrální slovo neexistuje. Podrobná diskuse na toto téma je uvedena v oddíle [18.7](#).

- **Sdílená agregace**, nazývaná také jednoduše *agregace*, označená bílým (prázdným) kosočtvercem vedle celého konce odkazu, je volný, „slabý“ typ vztahu celku a části. Na rozdíl od kompozitní agregace má v sdílené agregaci část „vlastní život“ a může být součástí více než jednoho celku. Podle Object Management Group (2010, s. 39) „*přesná sémantika sdílené agregace se liší podle oblasti použití a modeláře*“. Ačkoli v sdílené agregaci může každá část obvykle existovat nezávisle na celku, ponechání sémantiky vztahu nejasné není zpočátku dobrý nápad. Označená strukturální relace v OPM je definována uživatelem, což je lepší způsob, jak vyjádřit specifickou sémantiku podle oblasti použití nebo modeláře, než nechat sémantiku jazykového symbolu otevřenou různým interpretacím různých modelářů, i když se všichni zabývají stejným systémovým modelem.

Spojovací čáry agregačního vztahu v UML nemusí být ortonormální a jsou obvykle diagonálně rovné, jak ukazuje [obr. 17.4](#). UML a SysML nemají konstrukci vidlice, takže jak ukazuje [obr. 17.4](#), musí být každá část v diagramu třídy UML (a SysML) spojena speciálním symbolem agregace.

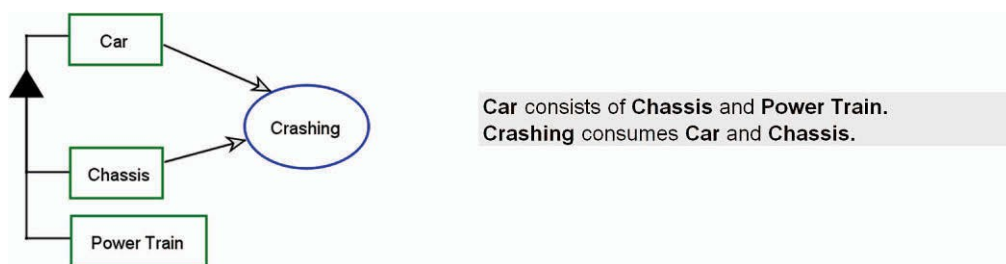
Kompozice je silnější než agregace v tom, že celek je „odpovědný“ za své části, takže když je spotřebován celek, jsou spotřebovány i všechny objekty, z nichž se skládá. Část proto nemůže být vlastněna více než jedním celkem. Zde je, co dokument UML 2.0 Superstructure v 2.2 (2005) říká o kompozitní agregaci (str. 41):

„Asociace může představovat kompozitní agregaci (tj. vztah celek/část). ... Kompozitní agregace je silná forma agregace, která vyžaduje, aby instance části byla zahrnuta nejvýše do jedné kompozice najednou. Pokud je kompozice odstraněna, všechny její části jsou obvykle odstraněny s ní. Upozorňujeme, že část může být (pokud je to povoleno) odstraněna z kompozice předtím, než je kompozice odstraněna, a tedy nemusí být odstraněna jako součást kompozice.“



Obr. 17.4 Symbol kompozitní agregace v UML a SysML

V OPM není nutné rozlišovat mezi kompozitní a sdílenou agregací, protože lze přesně modelovat, která část nebo části jsou spotřebovány, když je spotřebován celek, a které části zůstávají, jak ukazuje model OPM na [obr. 17.5](#): Po **havárii** zmizí celé **auto** i jeho **podvozek**, ale **pohonná jednotka** zůstává (a může být znovu použita).

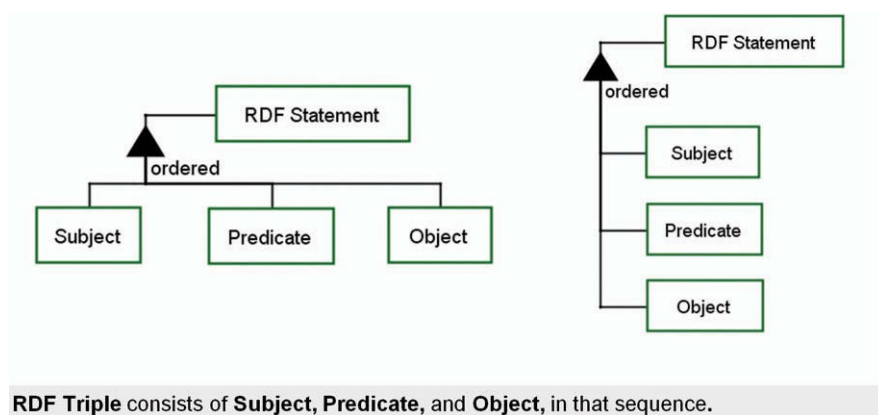


Obr. 17.5 Model OPM demonstrující, jak lze modelovat sdílenou a kompozitní agregaci v UML/SysML společně

17.6 Vyjádření pořadí částí

Někdy je pořadí částí, které tvoří celek, významné. Množiny jsou abstraktní sbírky věcí, které se skládají z *prvků* nebo *členů*. Množinu lze proto považovat za agregát (celek) a její prvky za části. Každý prvek v množině je jedinečný. Jelikož agregace-účast je strukturální vztah, vše, co platí pro vidličku, platí i pro agregaci-účast, včetně způsobu, jakým je označena řaditelnost. Jako vidlička vykazuje vztah **agregace-účast** vlastnost booleovské **uspořadatelnosti**, která označuje, zda je množina částí uspořádána. Dvě hodnoty **uspořadatelnosti** jsou uspořádaná a neuspořádaná, přičemž výchozí hodnota je neuspořádaná. Znovu se podívejme na případ trojice RDF (Klyne et al. 2004):

RDF trojice se obvykle zapisuje v pořadí subjekt, predikát, objekt.



Obr. 17.6 Označení OPD „ordered“ a vyhrazená fráze OPL „in that sequence“ označují pořadí částí trojice RDF zleva doprava (v OPD vlevo) nebo shora dolů (v OPD vpravo).

Graficky znázorňujeme skutečnost, že tři prvky trojice RDF jsou seřazeny přidáním štítku „ordered“ vedle černého trojúhelníku symbolizujícího vztah agregace-účast, jak je znázorněno v OPD na [obr. 17.6](#). Části mohou být seřazeny bez sémantického rozdílu buď zleva doprava, jak ukazuje OPD vlevo

vlevo, nebo shora dolů, jak ukazuje OPD vpravo. Odpovídající fráze OPL je „v tomto pořadí“, která následuje za čárkou za názvem poslední části v uspořádaném seznamu.

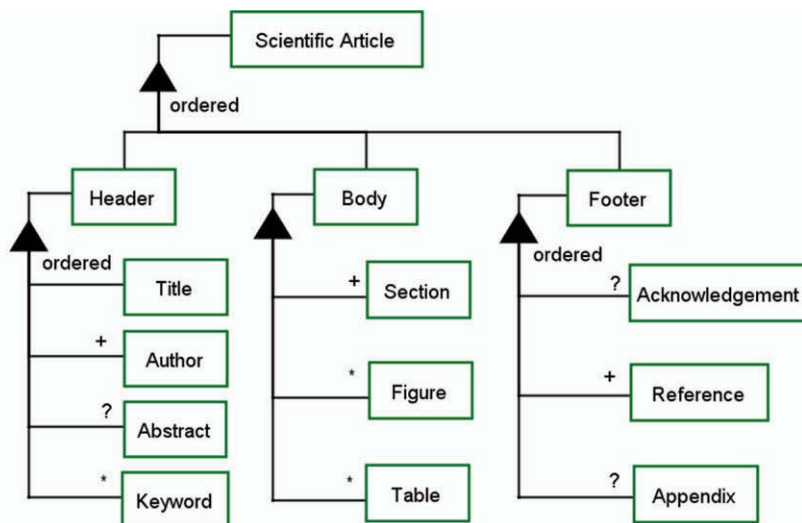
OPD na [obr. 17.7](#) je příkladem agregační hierarchie, která určuje pořadí čtení vědeckého článku, tj. pořadí, v jakém by měly být části článku čteny, s omezeními účasti, která jsou popsána v [kapitole 15](#).

Při práci s procesy je pořadí úzce spojeno s časovou osou shora dolů v rámci přibližného procesu, která určuje pořadí provádění procesu. Podrobněji se tím zabýváme v [kapitole 21](#) při diskusi o řízení složitosti.

17.7 Agregace a označené strukturální vztahy

V následujícím příkladu ilustrujeme model OPM, který kombinuje agregaci-účast s označenými strukturálními vztahy. Uvažujme větu extrahovanou z RDF Primer (Manola a Miller 2003):

RDF modeluje výroky jako uzly a oblouky v grafu.



Scientific Paper consists of **Header**, **Body**, and **Footer**, in that sequence.

Header consists of **Title**, at least one **Author**, an optional **Abstract**, and optional **Keywords**, in that sequence.

Body consists of at least one **Section**, optional **Figures**, and optional **Tables**.

Footer consists of an optional **Acknowledgement**, at least one **Reference**, and an optional **Appendix**, in that sequence.

Obr. 17.7 Uspořádaná agregační hierarchie vědeckého článku s omezeními účasti

Abychom tuto větu modelovali v OPM, využijeme naše předchozí znalosti o grafech a za předpokladu, že graf má alespoň dva uzly a jednu hranu (což je případ grafů RDF), rozdělíme výše uvedenou větu na následující tři jednodušší a explicitnější věty:

1. Graf se skládá z alespoň dvou uzlů a jedné hrany.
2. RDF graf je graf.
3. RDF graf modeluje alespoň jedno RDF tvrzení.

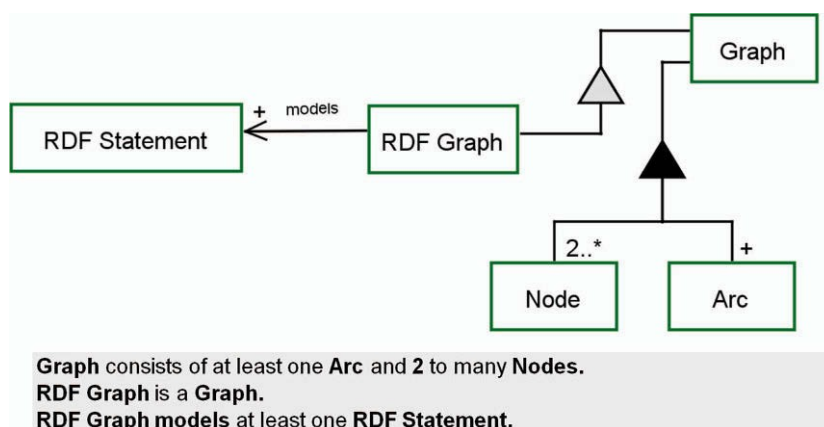
Věta (1) výše je modelována na [obr. 17.8](#). Stejně jako v předchozím příkladu černý trojúhelník označuje agregaci, kde objekt **Graph** je celek, zatímco **Node** a **Arc** jsou části. Symbol plus (+) nad **Arc** označuje účastnické omezení „alespoň jeden“ (+), zatímco symbol „2..*“ nad **Node** označuje účastnické omezení „2 až mnoho“.

Ve druhé větě OPL byla přidána skutečnost, že **RDF Graph** je (specializací) **Graph**. Jako takový zdědil strukturu **Graph**. K vyjádření skutečnosti, že **RDF Graph** modeluje alespoň jedno **RDF Statement**, se používá jednosměrná strukturovaná relace s tagem „models“.

Nakonec jsme získali dvě podobné věty OPL, které pocházejí ze dvou doporučení navržených W3C:

„RDF Statement se skládá z Subject, Predicate a Object.“ (Manola a Miller 2003) a

„RDF Triple se skládá z předmětu, predikátu a objektu.“ (Klyne et al. 2004)



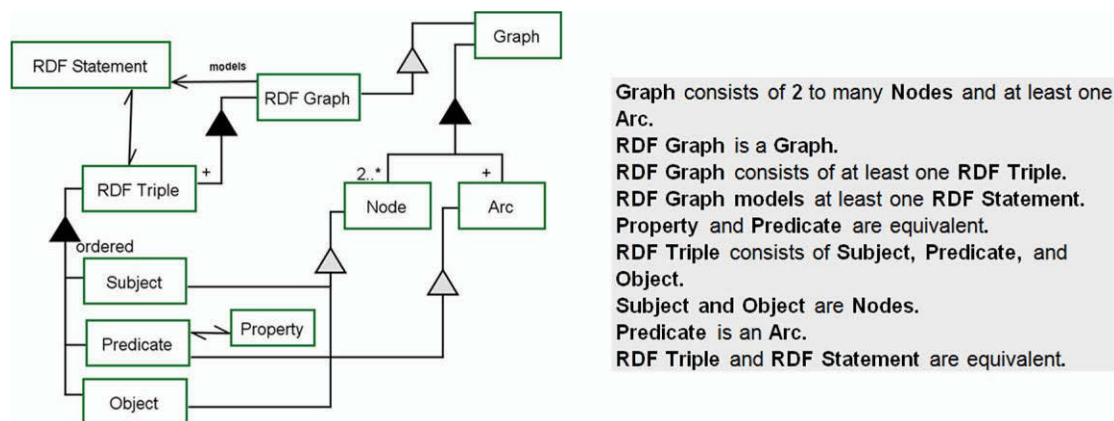
Obr. 17.8 Model OPM grafu sestávajícího z alespoň jednoho uzlu a volitelných oblouků

Za předpokladu, že pokud se dvě věci skládají z přesně stejné sady částí nebo komponent, jsou ekvivalentní (pokud nejsou stejné), lze odvodit, že **RDF Triple** a **RDF Statement** jsou ekvivalentní. Toto tvrzení je vyjádřeno v modelu OPM znázorněném na [obr. 17.9](#) pomocí (vertikální) nulové značky obousměrného strukturálního propojení mezi těmito dvěma objekty, které kombinuje modelové skutečnosti z [obr. 17.6](#) a [17.8](#). Tento OPD také vyjadřuje, že **subjekt** a **objekt** v **grafu RDF** jsou **uzly** v obecném **grafu** a že **predikát** v **grafu RDF** je **oblouk** v **grafu**.

Dalším příkladem použití nulového tagu obousměrného strukturálního vztahu je modelování věty z oddílu 6.1 (Klyne et al. 2004).

Predikát je také známý jako vlastnost trojice.

To je vyjádřeno v OPD na [obr. 17.9](#), kde je **vlastnost** propojena s **predikátem** pomocí nulového tagu obousměrného strukturálního propojení, aby se naznačilo, že jsou ekvivalentní, za předpokladu, že výchozí hodnota nulového tagu je „ekvivalentní“. To se překládá do věty OPL „**RDF trojice** a **RDF tvrzení** jsou ekvivalentní“.



Obr. 17.9 Model OPM demonstrující obousměrné strukturované propojení s jednou značkou

17.8 Neúplná agregace

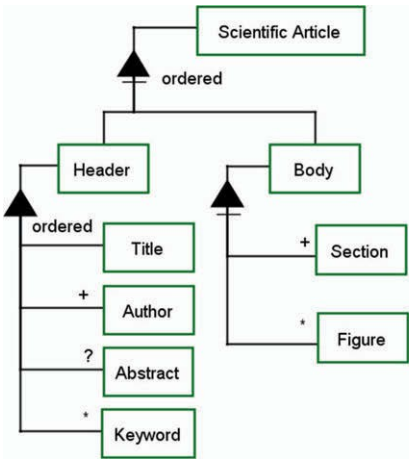
Jako specializace rozvětvení agregace zdědí vlastnost booleovské **úplnosti** stejně jako vlastnost booleovské **řazitelnosti**. Výchozí hodnota **úplnosti** agregace je **úplná**: předpokládáme, že pokud není nic uvedeno, jsou všechny části specifikovány v modelu. Pokud chceme označit, že agregace je **neúplná**, přidáme symbol neúplnosti – krátkou vodorovnou čárku pod černým trojúhelníkovým symbolem agregace, jak je znázorněno na [obr. 17.10](#) a [17.11](#). Odpovídající fráze OPL je „a alespoň jedna další část“, použitá v poslední větě OPL na [obr. 17.10](#).

Pokud je symbol agregace jak uspořádaný, tak neúplný, fráze OPL pro neúplnost předchází frázi pro uspořádanost. Například pokud by na [obr. 17.10](#) byl symbol agregace připojený k části **Body**, která je neúplná, také uspořádaný, výsledná věta OPL by byla:

Body se skládá z alespoň jedné **sekcce**, volitelných **obrázků** a alespoň jedné další části, v tomto pořadí.

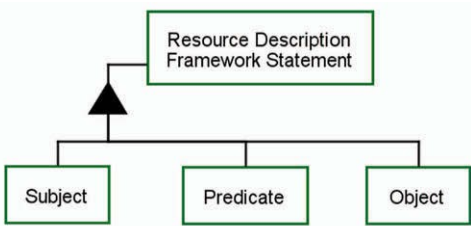
17.8.1 Částečná spotřeba agregace

Pokud chceme specifikovat, že je spotřebováno celé a konkrétní podmnožina jeho částí, můžeme to stručně modelovat pomocí částečné agregace spotřeby, jak je znázorněno na [obr. 17.12](#). V modelu OPM na levé straně [obr. 17.12](#) proces **Spotřeba** spotřebovává **Celek** spolu s jeho **Částí B** a **Částí D**, zatímco **Část A** a **Část C** zůstávají neporušené jako samostatné objekty. To je podobné jako v příkladu s autonehodou na [obr. 17.5](#). V modelu OPM na pravé straně [obr. 17.12](#) stručná verze používající částečnou agregaci ukazuje, že proces **spotřeby** spotřebovává **celek** a pouze **část B** a **část D**, zatímco všechny ostatní části **celku**, které nejsou zobrazeny v částečné agregaci, zůstávají jako samostatné, nezměněné objekty.

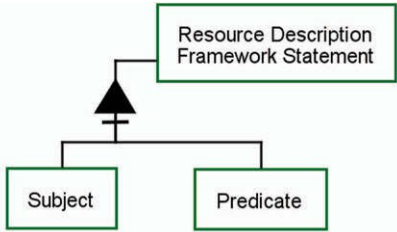


Scientific Paper consists of **Header**, **Body**, and at least one other part, in that sequence.
Header consists of **Title**, at least one **Author**, an optional **Abstract**, and optional **Keywords**, in that sequence.
Body consists of at least one **Section**, optional **Figures**, and at least one other part.

Obr. 17.10 Symbol neúplné agregace je krátká svislá čára pod agregačním trojúhelníkem, která vyjadřuje, že nejsou zobrazeny všechny části.

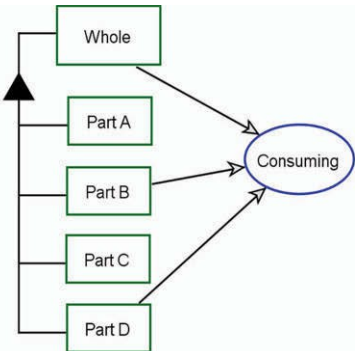


Resource Description Framework Statement consists of **Subject**, **Predicate**, and **Object**.

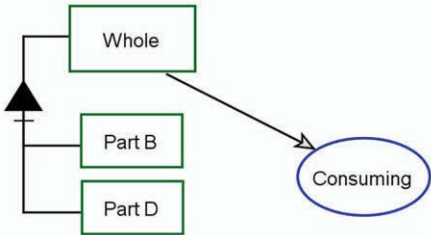


Resource Description Framework Statement consists of **Subject**, **Predicate**, and at least one other part.

Obr. 17.11 Použití neúplné agregace v prohlášení o rámci popisu zdrojů

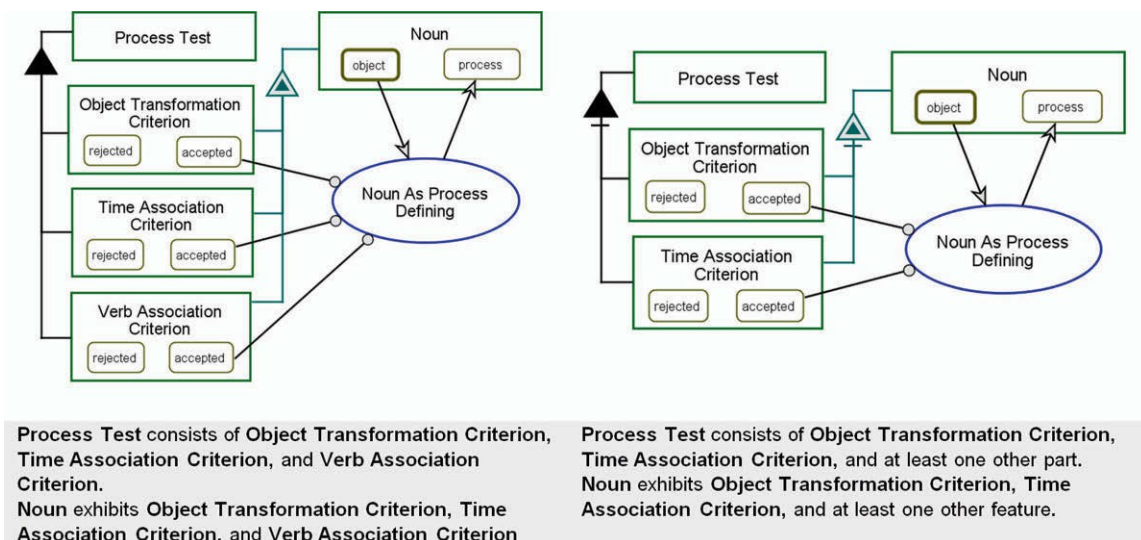


Whole consists of **Part A**, **Part B**, **Part C**, and **Part D**.
Consuming consumes **Whole**, **Part B**, and **Part D**.



Whole consists of **Part A**, **Part D**, and at least one other part.
Consuming consumes **Whole**, **Part B**, and **Part D**.

Obr. 17.12 Příklad částečné agregace spotřeby



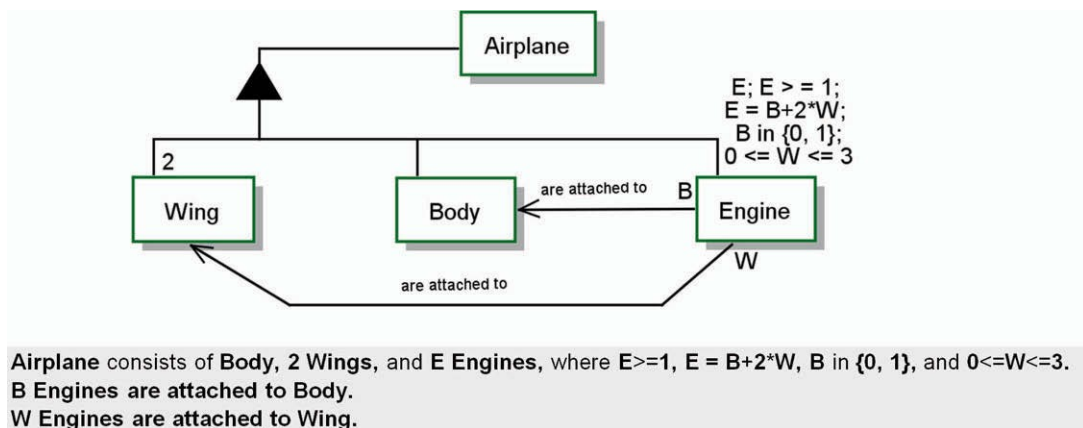
Obr. 17.13 Symbol neúplnosti demonstrováný pro agregaci a charakterizaci. Vlevo: Původní procesní testovací model. Vpravo: Aktualizovaný model po odstranění kritéria slovního spojení

Jelikož se jedná o vlastnost rozvětvení, symbol neúplnosti lze použít nejen pro agregaci, ale také pro každou z dalších tří základních strukturálních vztahů – exhibici, specializaci a klasifikaci. Aby bylo možné symbol neúplnosti správně použít, musí modelovací nástroj OPM sledovat sadu zjemněních pro každé zjemnitelné a upravovat symbol a odpovídající věty OPL podle toho, jak modelář mění sbírku zjemněních. To je demonstrováno na obr. 17.13, kde znovu používáme model OPM procesu testování z obr. 10.6, tentokrát však poskytujeme pouze dvě relevantní věty OPL. Vlevo je původní model, zatímco vpravo bylo odstraněno **kritérium asociace slovesa** objektu, které je součástí **testu procesu** i atributem **podstatného jména**, což způsobilo automatickou aktualizaci OPD tak, aby zahrnovala symbol neúplnosti jak pro agregaci, tak pro výstavu. Věty OPL byly také aktualizovány.

Jak vidíme, fráze OPL pro neúplnou charakterizaci výstavy je „a alespoň jedna další vlastnost“. Používáme vlastnost namísto atributu, protože, jak jsme diskutovali v kapitole 16 a podrobněji vysvětlíme v kapitole 18, vlastnost může být atribut (objekt) nebo operace (proces), přičemž oba mohou být připojeny k základu stejného symbolu výstavy, ▲. Podobně je fráze OPL pro neúplnou generalizaci-specializaci „a alespoň jedna další specializace“ a fráze OPL pro neúplnou klasifikaci-instanci je „a alespoň jedna další instance“.

17.9 Mini-jazyk parametrizovaných omezení účasti

Použití omezení účasti ve vztahu agregace-účast je podobné jejich použití v obecné značkové strukturální relaci. K každé části v sadě celku lze připojit jiné omezení účasti. Stejně jako u obecného označeného strukturálního vztahu je implicitní výchozí hodnota pro počet částí celku 1. Omezení účasti jiné než 1 se zaznamenává mimo část vedle bodu spojujícího část s ortonormální přímkou od základny pevného trojúhelníku.



Obr. 17.14 Parametrizované omezení účasti aplikované na agregční-účastnické vazby

OPD na obr. 17.14 a následující OPL jsou toho příkladem. Jelikož **letadlo** se skládá ze dvou **křidel**, je omezení účasti **2** zaznamenáno vedle objektu **Křídlo**. **Letadlo** se také skládá z určitého počtu **motorů**, jejichž přesný počet je určen několika parametry. Příklad na obr. 17.14 používá tři parametry, E , B a W , k vyjádření počtu **motorů** v **letadle**, počtu **motorů** připojených k **trupu** a počtu **motorů** připojených k **křídlu**.

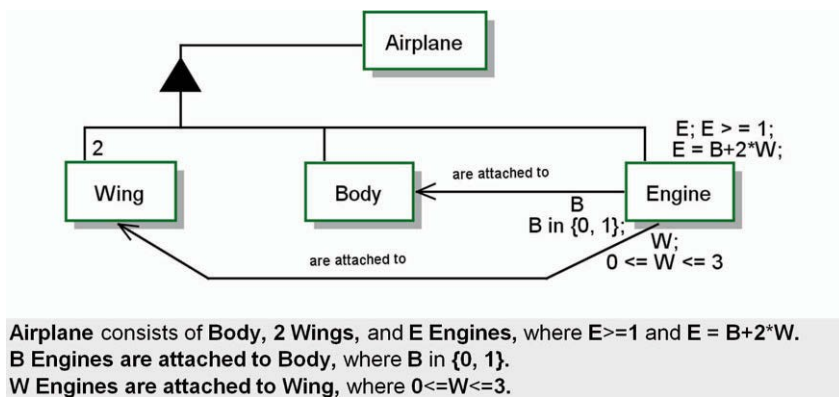
Jak je znázorněno na obr. 17.14, existuje specifická syntaxe parametrizovaných účastnických omezení, jak jsou zaznamenána v OPD. Tato syntaxe definuje jazyk s malou syntaxí, nazývaný mini-jazyk parametrizovaných účastnických omezení (PPC). Vychází ze syntaxe aritmetiky a zápisu množin v konvenčních programovacích jazycích třetí generace a OO, jako jsou C, C++ a Java, a je jim podobná. Tato syntaxe, neformálně specifikovaná v následujícím příkladu, nesmí být zaměňována s mnohem složitější syntaxí OPL, která je formálně prezentována v EBNF v OPM ISO 19450 PAS (viz kap. 24). Mini-jazyk PPC také nesmí být zaměňován s OCL v UML, který je rovněž navržen jako doplněk k UML pro specifikaci omezení, která nelze v UML vyjádřit graficky, jak je stručně popsáno v oddíle 22.10.

Pro demonstraci syntaxe mini-jazyka PPC se podívejme na příklad na obr. 17.14. Sada čtyř omezení, z nichž každé je vyjádřeno v řádku textu na obr. 17.14 nad objektem **Engine**, představuje *sadu omezení parametru E* – sadu čtyř omezení pro E , kde E je parametr pro počet **motorů** v **letadle**. Parametr (v našem případě E) se objevuje jako první, následuje středník a poté nula nebo

více (v našem případě čtyř) omezení oddělených středníky. Každé omezení je rovnost nebo nerovnost nebo zápis členství v sadě. Levá strana je název parametru, pravá strana je matematický výraz a obě strany jsou odděleny jedním ze symbolů rovnosti nebo nerovnosti $=$, $\neq$ (nebo $!=$, pokud je k dispozici pouze znaková sada ASCII), $<$, $>$, $\leq$ (nebo $<=$), $\geq$ (nebo $>=$) nebo znaky členství $\in$ (nebo „in“) nebo $\notin$ (nebo „not in“).

Jak již bylo uvedeno, symboly a syntaxe používané ve výrazech omezení vycházejí z běžných konvencí programovacích jazyků. Například násobení se označuje hvězdičkou, jako v $E = B + 2 * W$. Vyhrazená fráze in je symbol teorie množin $\in$, takže „ b in $\{0, 1\}$ “ je totéž jako „ $b \in \{0, 1\}$ “. V našem příkladu první omezení, $E \geq 1$, omezuje počet motorů v letadle na alespoň jeden. Druhé omezení, $E = B + 2 * W$, je celkový počet motorů, který se rovná počtu motorů v trupu (který může být 0 nebo 1), a W je počet motorů v každém křídle (který může být 0, 1, 2 nebo 3).

Obr. 17.15 Parametrizované omezení účasti z obr. 17.14 vyjádřené jinak



Jak ukazuje tento příklad, syntaxe OPL pro sadu parametrizovaných omezení je taková, že hlavní parametr předchází názvu objektu, ke kterému se vztahuje, následuje čárka a vyhrazená fráze where, za kterou následuje seznam omezení oddělených čárkami s vyhrazenou frází a předcházející poslednímu omezení.

Obrázek 17.15 představuje další způsob specifikace parametrizovaných omezení účasti, který se liší od způsobu na obrázku 17.14, ale používá stejnou syntaxi parametrizovaných omezení a má stejnou sémantiku. Mini-jazyk PPC je v oddíle 22.10 porovnán s jazykem Object Constraints Language (OCL), který rozšiřuje UML.

17.10 Shrnutí

- Agregace-účast je základní strukturální vztah, který označuje skutečnost, že zdokonalitelná entita – celek – agreguje jednu nebo více zdokonalitelných entit – části.
- Agregace-účast je dvojice dopředných a zpětných strukturálních vztahů.

- Černý trojúhelník, ▲ , je symbol vztahu agregace-účast. Nahrazuje dvojici dopředných a zpětných textových značek vztahu agregace-účast.
- Agregace je proces vytváření celku z jeho částí, zatímco participace je výčet částí, které tvoří agregát.
- V UML 2 a SysML existují dva typy agregace v diagramu tříd: sdílená – slabá agregace, označená bílým kosočtvercem, a složená – silná agregace, označená černým kosočtvercem.
- V OPM není rozlišení mezi kompozitní a sdílenou agregací nutné, protože lze přesně modelovat, která část nebo části jsou spotřebovány, když je spotřebován celek, a které části zůstávají.
- Řaditelnost je booleovská vlastnost agregací vztahu, zděděná z rozvětvení.
- Abychom označili, že agregace je uspořádaná, přidáme symbol `ordered` vedle trojúhelníku agregace.
- Komplexnost je další booleovská vlastnost agregací vztahu, zděděná z fork.
- Abychom označili, že agregace není komplexní, přidáme pod agregací trojúhelník krátkou vodorovnou čárou.
- Mini-jazyk parametrizovaných účastnických omezení (PPC) má malou syntaxi, která určuje, jak formulovat sadu omezení pro parametr v účastnickém omezení.

17.11 Úlohy

1. Nakreslete dva OPD dvoupatrového domu a jeho hlavních částí, jeden bez a jeden s agregacím účastnickým odkazem.
2. Který OPD bylo snazší nakreslit? Proč?
3. Použijte druhý OPD z prvního problému k demonstraci použití řaditelnosti z hlediska vertikálního umístění částí domu, přičemž nejvyšší je první. Přidejte části podle potřeby.
4. Prokažte neúplnost odstraněním jedné nebo více částí z OPD v předchozí otázce.
5. Přidejte alespoň dvě omezení účasti do OPD z jedné z předchozích otázek.
6. Nakreslete OPD popisující dva objekty, u nichž jsou části seřazeny, a dva, u nichž nejsou.
7. Nakreslete dva OPD pro objekt skládající se z alespoň osmi různých částí na první úrovni účasti s neúplným agregováním. Podmnožina částí by se měla objevit v jednom OPD a další podmnožina v druhém OPD tak, aby sjednocení podmnožin bylo úplné.
8. Podle obrázků 17.13 a 17.14, jaký je možný počet motorů v **letadle**?

9. Pomocí parametrizovaných omezení účasti vytvořte agregační hierarchii výškové budovy. Budova má určitý počet podlaží, z nichž každé má dva typy bytů, standardní a luxusní. Na každém patře od 4. patra výše jsou tři standardní a dva luxusní byty. Na prvních třech patrech je jedna malá a dvě velké kanceláře. Rozhodněte, kolik pater je a kolik kohoutků je potřeba pro každou jednotku, a vytvořte příslušný OPD s omezeními účasti. Doplňte podrobnosti podle svého uvážení. Pomocí svého OPD vypočítejte počet kohoutků, které musí dodavatel objednat pro 22patrovou budovu.

Kapitola 18

Výstava – Charakterizace

Musím být schopen přiřadit vlastnosti k objektům.

Kant (1787)

K definování a popisu věcí ve světě používají přirozené jazyky adjektiva a adverbia. Bez těchto typů slov, která popisují objekty a jsou také zaměnitelně nazývána atributy, vlastnosti, kvality, charakteristiky nebo vlastnosti, nelze objekty ani procesy adekvátně rozlišit a pochopit. Charakterizace výstavy je základním strukturálním vztahem, který spojuje zdokonalitelný objekt nebo proces (*vystavovatel*) se zdokonalovaným objektem nebo procesem (*vlastnost*), který charakterizuje vystavovatele.

18.1 Vlastnost a vystavovatel

Charakterizace výstavy je základní strukturální vztah. Stejně jako každý binární strukturální vztah zahrnuje dvě věci: vystavovatele a vlastnost.

***Vlastnost** je rafinovaný objekt, který charakterizuje (popisuje) věc.*

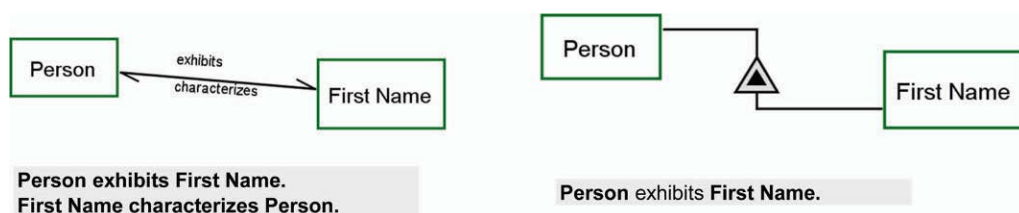
***Vystavovatel** je rafinovatelný objekt, který vystavuje (je charakterizován) vlastností.*

***Výstava-Charakteristika** je základní strukturální vztah, který označuje skutečnost, že určitá vlastnost charakterizuje vystavovatele (a naopak, vystavovatel vystavuje tuto vlastnost).*

Aby byl zachován soulad s konvencí pojmenování základních strukturálních vztahů, první slovo v páru vztahů výstava-charakterizace popisuje přímý směr vztahu, od vystavovatele k vlastnosti, zatímco opačný směr vede od vystavovatele k vlastnosti.

Vztah mezi vlastností a vystavovatelem ve vztahu vystavení-charakterizace je analogický vztahu mezi částí a celkem ve vztahu agregace-účast: Část je rafinát, který zahrnuje rafinovatelný produkt – celek, který agreguje části. Stejně jako agregace-účast je i vystavení-charakterizace přenosná, což vede k vytvoření hierarchie vystavení. Směr vpřed je tedy také směrem dolů: od věci vyšší v hierarchii – vystavovatele – k jedné nebo více věcem nižším v hierarchii – vlastnostem.

Směr vpřed (dolů) vztahu mezi výstavou a charakteristikou, od vystavovatele k jeho vlastnostem, je směrem *výstavy*, zatímco opačný směr (nahoru), od každé vlastnosti k vystavovateli, je směrem *charakteristiky*. Výše uvedená definice předpokládá směr vpřed vztahu mezi výstavou a charakteristikou. Při pohledu v opačném směru se říká, že vlastnost *charakterizuje* vystavovatele. **Obrázek 18.1** vyjadřuje vlevo vztah mezi výstavou a charakterizací jako obousměrné strukturální propojení s označením, což vede ke dvěma větám OPL, zatímco vpravo je použit symbol určený pro tento vztah, což vede k jedné větě OPL s vyhrazeným slovem OPL (nevyznačeným tučně) exhibits.



Obr. 18.1 Vztah mezi výstavou a charakterizací vyjádřený jako obousměrný značkový strukturální odkaz (vlevo) a s určeným symbolem vztahu (vpravo)

Slovo pro zpětný vztah, *charakterizace*, se v kontextu tohoto vztahu používá mnohem častěji než jeho dopředný protějšek, *výstava*. *Charakterizace* je proto zkráceným názvem tohoto vztahu. Na základě toho můžeme občas vynechat část názvu tohoto základního strukturálního vztahu „výstava“ a zkrátit jej na *charakterizace*, přičemž je třeba mít na paměti, že se jedná o směr nahoru v hierarchii.¹

18.1.1 Primární a sekundární vlastnosti

Mnoho filozofů, kteří se zabývali epistemologií a metafyzikou, včetně Galilea, Descarta a Locka, provedlo koncepční rozlišení mezi primárními a sekundárními vlastnostmi (nebo atributy). Primární vlastnosti jsou „nezávislé“ vlastnosti objektů, jako je tvar nebo hmotnost, které vyjadřují fakta o věci a nespolehají se na subjektivní úsudky. Sekundární vlastnosti jsou vlastnosti jako barva, chuť, vůně a zvuk, které závisí na vjemu pozorovatele a vyvolávají v něm pocity, ale neposkytují objektivní fakta o věcech.

Jak napsal Galileo (1623) v díle *The Assayer*, „... *chutě, vůně, barvy a tak dále nejsou ničím jiným než pouhými jmény, pokud jde o objekt, ve kterém je nacházíme, a že se nacházejí ve vědomí.*“ Descartes (1647) dále napsal o sekundárních vlastnostech, že „*nejsme si vědomi toho, že jsou něčím jiným než různými uspořádáními velikosti, tvaru a pohybů částí těchto objektů, které umožňují našim nervům pohybovat se různými způsoby a vzbuzovat v naší duši všechny různé pocity, které*

¹ Tento ústepek ilustruje druh kompromisních rozhodnutí, která je třeba učinit při vymýšlení názvů OPM. Na jedné straně je nezbytná konzistence a přehlednost, na druhé straně se však jasnost a expresivita zvyšují, když je jazyk co nejpřirozenější a nejstručnější. Jelikož angličtina, stejně jako všechny přirozené jazyky, má své zvláštnosti, je často nutné učinit kompromisy, jako je tento, po zvážení kladů a záporů každé alternativy. Může se také zdát poněkud podivné, že vynaládáme tolik intelektuálního úsilí na výběr vhodných názvů pro abstraktní pojmy. Významný název však může znamenat velký rozdíl mezi dobře pochopeným a správně použitým pojmem a pojmem, který kvůli formálně správnému, ale špatně srozumitelnému termínu postrádá smysl.

kteřé tam produkuje.“ Podobně pozorování učinil Newton (v *Optica*, 1721) ohledně barvy paprsků a Leibnitz (v *Discourse on Metaphysics*, 1686) ohledně velikosti, tvaru a pohybu.

Toto rozlišení kritizoval Berkeley (1710) ve své teorii „nematerialismu“, která popírala existenci hmotné substance jako takové. Podle Berkeley jsou známé objekty, jako například stůl, pouze idejemi v mysli lidského vnímatele a nemohou existovat, aniž by byly vnímány. Ideje vytvořené vjemy jsou vše, co lidé mohou s jistotou vědět. Když je objekt zbaven všech svých sekundárních vlastností, myšlenka, že nějaký objekt existuje, nemá žádnou oporu, protože bez vlastností nelze myšlence existence objektu dát žádný obsah. Kant (1783) se také postavil proti tomuto rozlišení a tvrdil, že jak primární, tak sekundární vlastnosti jsou subjektivní, protože se nacházejí v mozku vnímajícího pozorovatele. Tato diskuse doplňuje naše předchozí pojednání v oddíle 10.3 o identitě objektů.

18.2 Atribut a operace: dva druhy vlastností

Vytrvalost je vlastnost věci, která má dvě hodnoty: statickou a dynamickou. Vytrvalost je vlastnost, která umožňuje rozlišovat mezi objektem a procesem. Určuje, že věc je objektem, pokud je hodnota vytrvalosti trvalá (statická), a procesem, pokud je hodnota vytrvalosti přechodná (dynamická). Funkce – vlastnost, která charakterizuje věc – se také dělí na dva typy podle toho, zda je její hodnota vytrvalosti statická nebo dynamická.

Atribut je vlastnost, jejíž hodnota vytrvalosti je statická.

Operace je vlastnost, jejíž hodnota vytrvalosti je dynamická.

Atribut je statická vlastnost – *objekt*, který charakterizuje věc, zatímco operace je dynamická vlastnost – *proces*, který charakterizuje věc. Jako objekt má atribut *trvalou* hodnotu (statickou). Jako proces má operace *přechodnou* hodnotu (dynamickou).

Věta OPL, která spojuje **vystavovatele** se dvěma vlastnostmi, **vlastností 1** a **vlastností 2**, zní:

Vystavovatel vystavuje **prvek 1** a **prvek 2**.

Věta OPL, která spojuje **vystavovatele** se třemi vlastnostmi, **vlastností 1**, **vlastností 2** a **vlastností 3**, zní:

Vystavovatel vystavuje **vlastnost 1**, **vlastnost 2** a **vlastnost 3**.

Všechny vlastnosti na seznamu musí být stejného charakteru, tj. všechny jsou atributy (vlastnosti objektu) nebo všechny jsou operace (vlastnosti procesu). Pokud jsou některé vlastnosti atributy a jiné operace, rozdělíme vlastnosti do dvou seznamů, jeden pro atributy a druhý pro operace. Pokud je vystavovatel objekt, pak první seznam vlastností obsahuje (jeden, dva nebo více) atributů a druhý seznam obsahuje operace. Seznam atributů je propojen se seznamem operací pomocí vyhrazené fráze OPL „stejně jako“. Jako příklad uvádíme následující větu OPL, která specifikuje **objektový** exponent se třemi atributy a dvěma operacemi:

Objekt vykazuje **atribut 1**, **atribut 2** a **atribut 3**, stejně jako **operaci 1** a **operaci 2**.

Pokud je vystavenou věcí **proces**, seznam operací předchází seznamu atributů:

Proces vykazuje **operaci 1** a **operaci 2**, stejně jako **atribut 1**, **atribut 2** a **atribut 3**.

18.3 Funkce v UML a SysML versus OPM

Atributy a operace jsou pojmy, které existují také v objektově orientovaném (OO) přístupu. V terminologii OO se atribut také označuje jako *datový člen*, zatímco operace se také označuje jako *metoda* nebo *služba*. Všechna tato slova mají vyjádřit „něco, co objekt může dělat“ nebo „způsob, jakým se objekt chová“. V tradičních procedurálních programovacích jazycích třetí generace se operace označuje také jako *funkce*, *procedura* nebo *rutina*. [Tabulka 18.1](#) shrnuje definice atributu a operace jako specializace vlastnosti spolu s podobnými pojmy v OO a tradičních programovacích jazycích.

Tabulka 18.1 Specializace věci a vlastnosti podle vytrvalosti a podobné pojmy v OO a tradičních programovacích jazycích

Hodnota vytrvalosti	Věc	Vlastnost	Podobné pojmy v OO	Tradiční podobné pojmy
trvalý (statický)	Objekt	Atribut	Datový člen	Proměnná, parametr
přechodný (dynamický)	Proces	Operace	Metoda, služba	Postup, rutina, podprogram, funkce,

OPM zachází s vlastnostmi jako s věcmi, které mají své vlastní právo na existenci, bez ohledu na to, že mohou také charakterizovat věci vyšší úrovně. Zatímco agregace-účast a generalizace-specializace jsou v SysML (stejně jako v UML) uznávané vztahy a mají své vlastní symboly (černý nebo bílý kosočtverec pro první, bílý trojúhelník pro druhý), výstava-charakterizace není explicitním vztahem a nemá symbol. Atribut je spíše uznáván jako takový v UML podle svého umístění ve druhém ze tří vertikálně uspořádaných oddílů, které tvoří symbol třídy objektů UML. V SysML může být v bloku libovolný počet oddílů, takže každý oddíl musí být označen. Například na [obr. 18.2](#) je označení „hodnoty“.

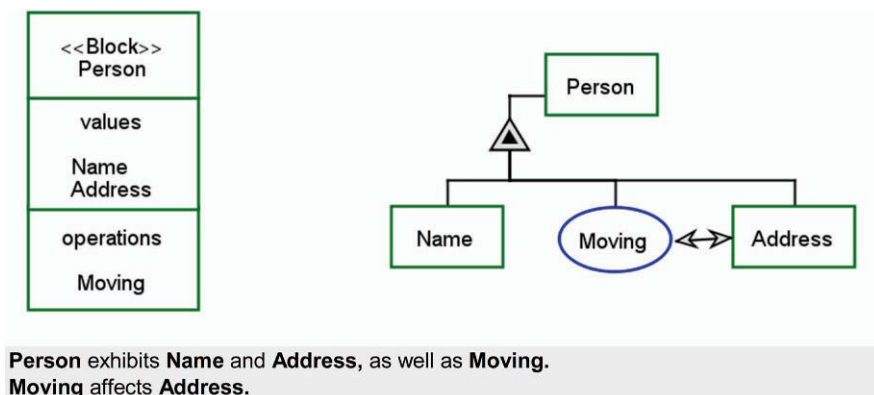
Paradoxně, ačkoli se OPM nesnaží být „čistě“ objektově orientované, je ve svém zacházení s charakterizací více objektově orientované než paradigma OO. V OO jsou atributy a metody zapouzdřeny nebo vloženy do objektů. Nejsou atributy objekty, ale spíše „jiná zvířata“, která se nacházejí uvnitř objektu? Pokud atribut není objekt, co tedy je? Skládá se svět nejen z objektů, ale také z atributů (a metod)? OPM se s tímto dilematem nesetkává, protože definuje vlastnost obecně jako věc, která popisuje věc, a jako věc, která se specializuje na atribut – objekt – a operaci – proces.

Pro ilustraci problému způsobeného tím, že atributy nejsou považovány za objekty, uvažujme „klasický“ příklad **jména** a **adresy** jako atributů objektové třídy **Osoba** a **stěhování** jako operace **Osoba**.² Jak ukazuje [obrázek 18.2](#) vlevo, v SysML se to provádí přiřazením názvu každému oddílu. Horní oddíl má stereotypní název „Block“, který je analogický k Object v UML a OPM, s názvem bloku Person pod ním. Pod tímto horním oddílem se nachází oddíl „hodnot“ (atributů) s hodnotami Name a Address a ve spodní části jsou operace

² Zde předpokládáme, že **osoba** je schopna **se pohybovat** bez nutnosti vnějších objektů, takže **pohyb** lze považovat za operaci **osoby**.

s operací Moving. V UML a mnoha jeho předchůdcích, jako je Object Modeling Technique, OMT (Rumbaugh et al. 1991), jsou atributy a operace vždy uvedeny ve druhé a třetí části pole třídy, takže nejsou potřeba žádné názvy.

Na pravé straně [obr. 18.5](#) je zobrazena odpovídající notace OPM: **Jméno** a **adresa** jsou samostatné objekty a **pohyb** je proces. Jelikož **jméno** a **adresa** jsou spojeny s **osobou** symbolem charakterizace výstavy, jsou také atributy **osoby**. Ze stejného důvodu je **pohyb** operací **osoby**. Vedlejším přínosem této notace je, že můžeme spojit **stěhování** s **adresou** pomocí efektového odkazu, abychom označili skutečnost, že **stěhování** má vliv na **adresu osoby**, čímž již v tomto jednoduchém OPD kombinujeme strukturu a chování.



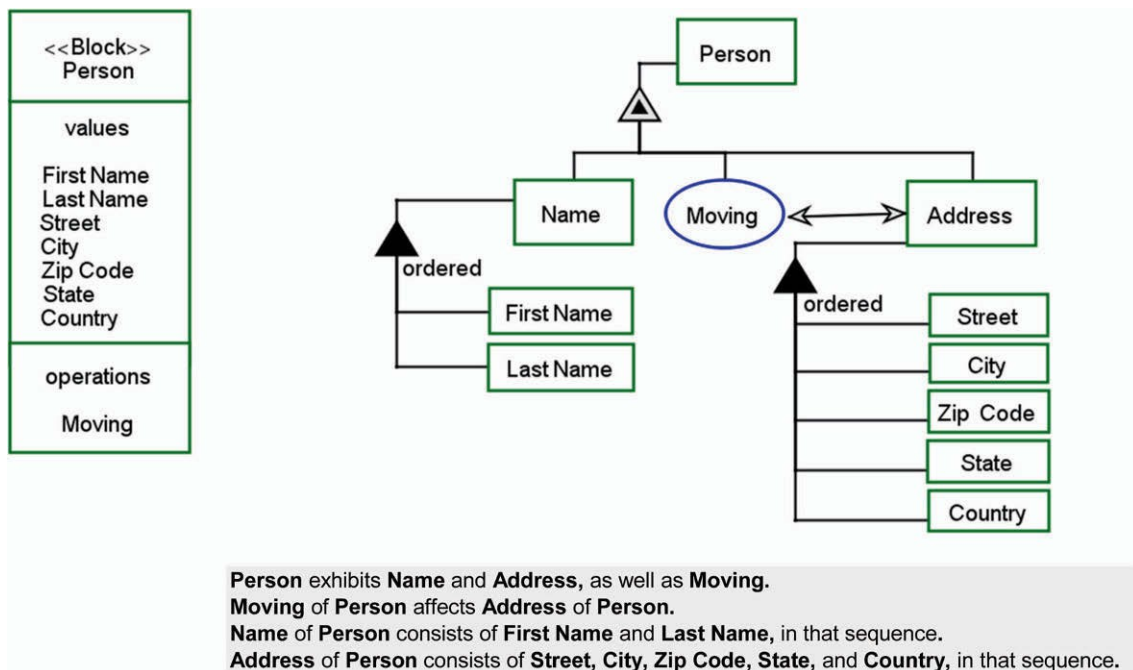
Obr. 18.2 Vyjádření atributů (hodnot) a operací v SysML (vlevo) a v OPM (vpravo)

Mimo kontext **osoby** jsou **jméno** i **adresa** samostatnými objekty. Jak je znázorněno na [obr. 18.3](#), každý z nich se skládá z několika částí: **jméno** se skládá z **křestního jména** a **příjmení**, **adresa** se skládá z **ulice**, **města**, **PSČ**, **státu** a **země**, v tomto pořadí.

18.4 Jedinečnost názvů věcí a vlastností v OPM

Různé věci v modelu OPM musí mít různá jména, aby byly rozlišitelné a nedocházelo k záměně. Pokud však jde o vlastnosti, které popisují věci, je obtížné vymyslet pro každou vlastnost jiné jméno. Například na [obr. 18.3](#) je atribut **osoby** nazvaný **Jméno**, ale **ulice** a **město** mohou mít také atribut nazvaný **Jméno**. Proto mohou vlastnosti věcí mít stejný název jako vlastnosti jiných věcí.

Jedinečnost vlastností je zachována přidáním „of Exhibitor“, kde of je vyhrazená fráze OPL (v tomto případě slovo) a **Exhibitor** je název věci, která vlastnost vykazuje. Vlastnost vlastnosti tedy musí mít dvě vyhrazená slova OPL „of“, jako v případě **Length of Name of Person**. Následující *princip OPM týkající se jedinečnosti názvů* to shrnuje.



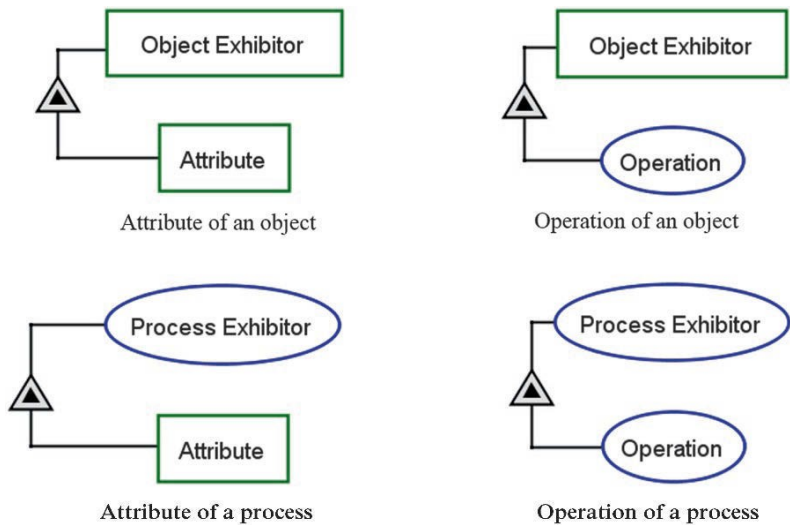
Obr. 18.3 Vyjádření části atributů v SysML (vlevo) a v OPM (vpravo)

Princip jedinečnosti názvu věci OPM

Různé prvky v modelu OPM, které nejsou funkcemi, musí mít odlišné názvy. Funkce lze rozlišit přidáním vyhrazeného slova „of“ a názvu jejich vystavovatele.

18.5 Čtyři kombinace věcí a funkcí

Charakterizace výstavy je mezi strukturálními vztahy jedinečná v tom, že jako jediná umožňuje propojit objekty s procesy a procesy s objekty. Všechny ostatní strukturální vztahy, včetně zejména zbývajících tří základních strukturálních vztahů, umožňují propojit pouze věci se stejnou hodnotou vytrvalosti: objekty (věci, jejichž hodnota vytrvalosti je trvalá nebo statická) lze propojit pouze s objekty a procesy – (věci, jejichž hodnota vytrvalosti je přechodná nebo dynamická) pouze s procesy. Objekty tedy mohou být pouze částmi, specializacemi nebo instancemi objektů a procesy mohou být pouze částmi, specializacemi nebo instancemi procesů. Pokud však jde o charakterizaci výstavy, jsou možné všechny čtyři kombinace objekt-proces (vystavovatel-vlastnost). Jinými slovy, jak je také znázorněno na [obr. 18.4](#), protože jak věc, tak její vlastnost mohou být objektem nebo procesem, 2x 2 kartézský součin dává stavový prostor čtyř různých kombinací věci a vlastnosti, která ji charakterizuje, a to zleva doprava a shora dolů na [obr. 18.4](#): (1) atribut objektu, (2) operace objektu, (3) atribut procesu a (4) operace procesu.

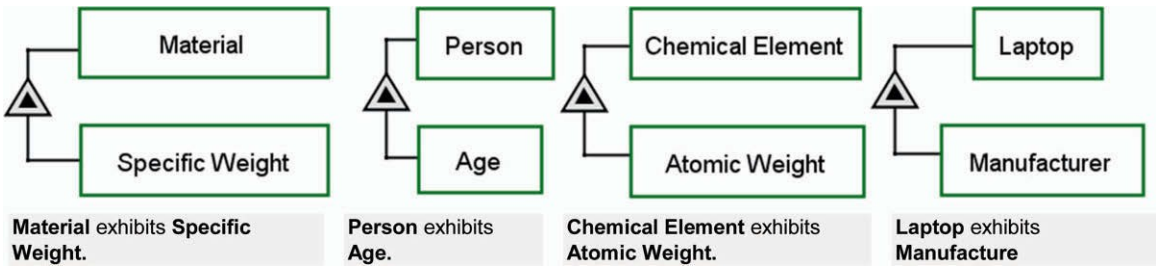


Obr. 18.4 Čtyři kombinace objekt-vlastnost

Jako příklad kombinace objektu a atributu lze uvést **adresu**, která je sama o sobě objektem, ale zároveň je také atributem **osoby**, protože je jednou z věcí, které ji charakterizují. Jako příklad kombinace objektu a operace lze uvést **tisk**, který je procesem, ale zároveň také operací **tiskárny**, protože je věcí, která charakterizuje schopnosti **tiskárny** – její funkci. Všechny čtyři kombinace jsou popsány a dále demonstrovány v této části. V následujících podkapitolách se budeme podrobněji zabývat každou z těchto kombinací.

18.5.1 Kombinace objekt-atribut

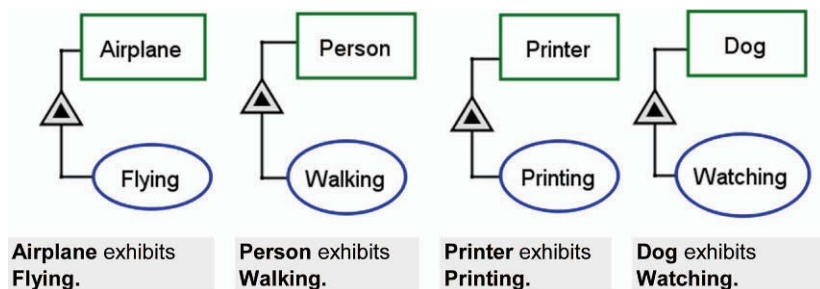
První kombinace věci a vlastnosti – objekt a jeho atribut – je obvyklým atributem klasických OO přístupů. Zde odkazujeme na objekt B_2 – atribut – který *charakterizuje* (popisuje) objekt vyšší úrovně B_1 . Naopak říkáme, že B_1 *vykazuje* B_2 . Několik příkladů takových párů objektů a jejich atributů jsou **materiál – specifická hmotnost**, **osoba – věk**, **chemický prvek – atomová hmotnost**, **notebook – výrobce**, **kniha – autor**, **důstojník – hodnost** a **pes – plemeno**. První čtyři z těchto příkladů jsou znázorněny ve čtyřech modelech OPM na [obr. 18.5](#).



Obr. 18.5 Příklady atributů objektů

18.5.2 Kombinace objekt-operace

Druhou kombinací vlastnosti a funkce je objekt a jeho operace. Jak již bylo uvedeno, v objektově orientovaných přístupech se operace také nazývá *metoda* nebo *služba* (viz [tabulka 18.1](#)). Zde odkazujeme na proces P_1 —operace—, který charakterizuje objekt vyšší úrovně B_1 . Naopak říkáme, že B_1 vykazuje operaci P_1 .



Obr. 18.6 Příklady operací objektů

Operace objektu je proces, který je vnitřní pro objekt: může být prováděn objektem nebo jeho částí (částmi) a ovlivňuje pouze objekty, které jsou částmi, vlastnostmi nebo specializacemi tohoto objektu. Jinými slovy, operace objektu B_1 nemá žádný vedlejší účinek na objekty mimo B_1 ani žádné takové objekty nevyžaduje. Za této podmínky lze operaci identifikovat jako „vlastněnou“ B_1 OO přístup, a následně i UML a SysML, považují *všechny* procesy za operace, které jsou zapouzdřeny v objektech a jsou jimi vlastněny. Toto zapouzdření je hlavním zdrojem nejasností a překážkou věrného modelování systému. V OPM je zapouzdření platné pouze tehdy, když je proces interní pro objekt. V takových případech je proces definován jako operace zapouzdřujícího objektu.

Několik příkladů párů objektu a jeho operace jsou **Letadlo – Let**, **Osoba – Chůze**, **Tiskárna – Tisk**, **Důstojník – Velení** a **Pes – Hlídkání**. [Obrázek 18.6](#) představuje čtyři modely OPM, které odpovídají těmto dvojicím. Jak ukazují tyto příklady, operace je specializací procesu. Jako taková by měla být operace pojmenována gerundiem, tj. slovesnou formou končící příponou „ing“.

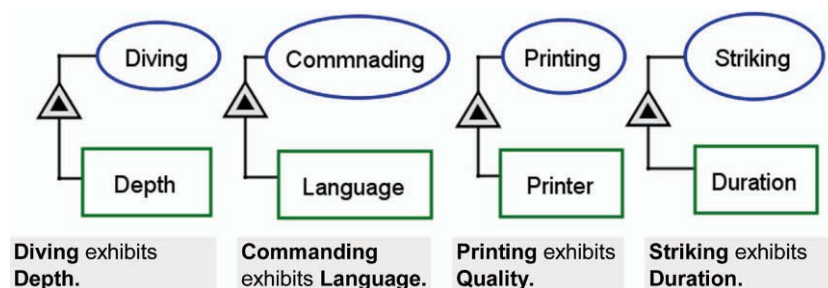
Mnoho objektů, zejména fyzických a umělých, vykazuje hlavní operaci, která vyjadřuje hlavní *funkci*, pro kterou je objekt navržen; *službu*, kterou má poskytovat. Takové objekty jsou systémy. Systém (který je umělý) poskytuje hodnotu příjemci systému. Například funkcí, kterou poskytuje objekt **Tiskárna**, je **tisk**, funkcí **Letadla** je **létání**, funkcí **Jeřábu** je **zvedání** a funkcí **Sušičky** je **sušení**. To je v souladu s naší definicí umělého systému jako objektu, který vykonává určitou funkci.

18.5.3 Kombinace procesu a atributu

Stejně jako objekty vyžadují i procesy adekvátní reprezentaci v modelu jakéhokoli systému. Stejně jako objekty mohou i procesy vyžadovat atributy – objekty, které je popisují. Myšlenka atributů pro procesy je přirozeným rozšířením atributů pro objekty a nepředstavuje žádnou zvláštní koncepční obtíž.

Dosud jsme viděli, že první a druhá kombinace věci a vlastnosti – objekt popisující objekt a proces popisující objekt – jsou odpovídajícími objektově orientovanými pojmy pro atribut a operaci (nebo službu, nebo metodu). Třetí kombinace věci a vlastnosti – objekt

popisující proces – není v OO přístupu explicitně definován. Zde odkazujeme na objekt B_1 – atribut – který charakterizuje proces vyšší úrovně P_1 . Naopak říkáme, že proces P_1 *vykazuje* atribut B_1 . Několik příkladů dvojic procesu a jeho atributu jsou: **potápění – hloubka**, **velení – jazyk**, **tisk – kvalita**, **úder – trvání**, **výroba – množství**, **sledování – účinnost**, **zpěv – hlasitost**, **lyžování – místo** a **létání – rychlost**.



Obr. 18.7 Příklady atributů procesů

Obrázek 18.7 představuje modely OPM, které odpovídají prvním čtyřem párům proces-atribut. Každý z těchto párů proces-atribut může být vložen do věty v přirozeném jazyce. Zde jsou možné příklady, kde jsou procesy tučně a jejich atributy kurzívou:

- (1) **Potápění** v *hloubce* 30 metrů nebo více vyžaduje, aby potápeč provedl dekompresní zastávky.
- (2) *Jazyk*, který kancelář používala k **vydávání příkazů**, byl cizí a podivný.
- (3) **Tisk** tohoto zařízení je *nekvalitní*.
- (4) Zaměstnanci **stávkují** již více než dva týdny.

Ačkoli všechny procesy v těchto příkladech jsou podstatná jména v gerundiu, lze je snadno převést do vět, kde jsou procesy slovesy, se stejným významem jako předtím:

- (1) Potápeč, který **se potápí** do *hloubky* 30 metrů nebo více, je povinen provést dekompresní zastávky.
- (2) Důstojník **vydává rozkazy** v *cizím jazyce*.
- (3) Toto zařízení **tiskne** v nízké *kvalitě*.
- (4) Zaměstnanci **stávkují** a tato situace trvá již více než dva týdny.

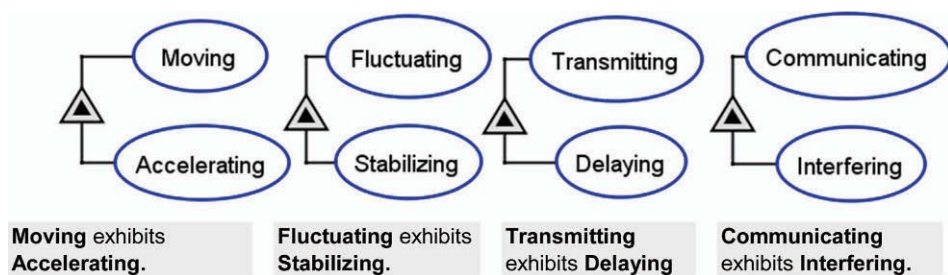
Jak ukazují tyto příklady, toto rozšíření OPM o pojmy atributů a operací OO, UML a SysML je přímým důsledkem uznání procesů jako skutečně nezávislých věcí, které nejsou nutně podřízené objektům nebo druhořadými objekty, které jsou vlastněny objekty.

18.5.4 Kombinace procesu a operace

Čtvrtá a poslední kombinace věci a vlastnosti – proces a jeho operace – je druhou kombinací, která není explicitně definována v objektově orientovaných (OO) přístupech, UML a SysML. Jedná se o nejméně rozšířenou kombinaci, která může být poněkud obtížně pochopitelná. Zde odkazujeme na proces $P_{(2)}$ – operaci – který charakterizuje proces vyšší úrovně $P_{(1)}$. Naopak říkáme, že proces $P_{(1)}$ *vykazuje* operaci $P_{(2)}$. Podle definice procesu v OPM může věc změnit pouze proces. Jinými slovy, proces je věc, která je za tuto změnu „odpovědná“. Tento proces může být operací. Operace

mění objekt, který vykazuje („vlastní“ v pojmech OO) tuto operaci. Stejně tak operace procesu mění vykazující proces – proces, který vykazuje tuto operaci.

V každodenním životě příliš nepřemýšlíme o operacích procesů. Nejlepší způsob, jak pochopit význam operace procesu, je podívat se na čas. Změna objektu v časové ose znamená, že stav objektu (nebo jeho hodnota, v případě, že objekt je atribut) zkontrolovaný v čase t se liší od jeho stavu v pozdějším čase $t + \Delta t$. Rozšíříme-li tuto myšlenku z objektů na procesy, pokud vzorkujeme proces ve dvou různých časových bodech, můžeme si všimnout změny v tomto procesu, která se projevuje jako rozdíl v hodnotě jednoho z atributů tohoto procesu, který je způsoben operací tohoto procesu.



Obr. 18.8 Příklady operací procesů

Obrázek 18.8 obsahuje čtyři částečné modely OPM, z nichž každý zobrazuje proces a jeho operaci. V modelu vlevo je **zrychlení** operací, která mění hodnotu atributu **rychlosti** procesu **pohybu**. Podobně operace **stabilizace** procesu **kolísání** mění hodnotu atributu **amplitudy kolísání**. Dále je **zpoždění** operací procesu **Přenos**, která mění jeho atribut **Trvání**. A konečně **rušení** je operací procesu **Komunikace**, která mění hodnotu atributu **Poměr signálu k šumu** procesu **Komunikace**.

Z matematického hlediska je změna objektu v čase *první derivací* určité veličiny (která je hodnotou atributu daného objektu) *vzhledem k času*. Analogicky, jelikož proces je vzorem transformace (odpovědným za transformaci objektu), operace procesu je transformací transformace, neboli *změnou změny*. Z matematického hlediska se jedná o *druhou derivaci* (derivace derivace) určité veličiny vzhledem k času. Příklady dvojic procesu a jeho operace uvedené na **obr. 18.8** skutečně obsahují pojem změny procesu a lze je matematicky kvantifikovat pomocí derivací druhého řádu. Například v modelu OPM na levé straně **obr. 18.8**, pokud označíme atribut **Rychlost** procesu **Pohyb** objektu jako funkci času $v(t)$, pak víme, že $v(t)$ je první derivace atributu **Poloha s** objektu jako funkce času: $v(t) = s'(t)$. Označíme-li $a(t)$ atribut **Zrychlení** procesu **Zrychlování**, máme $a(t) = v'(t) = s''(t)$, kde $a(t)$ je první derivace **Rychlosti** a druhá derivace **Polohy**.

18.6 Základní strukturální hierarchie

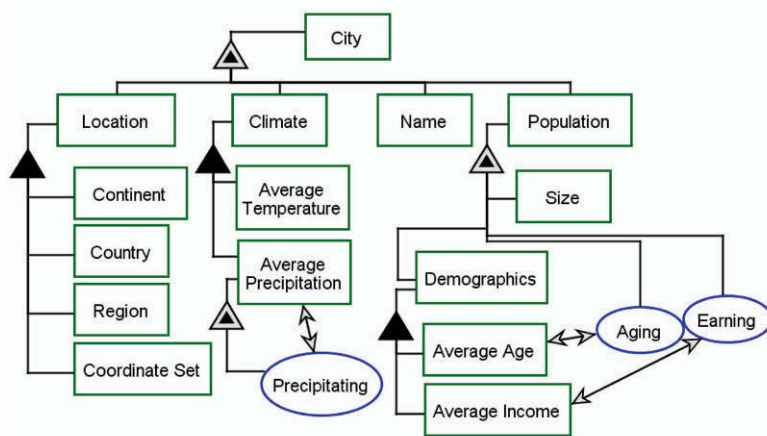
Vlastnost je relativní pojem. Věc je vlastností, pokud popisuje jinou věc. Tato vlastnost sama o sobě může mít části nebo může být dále popsána jinou vlastností nižší úrovně. Jelikož jak charakterizace výstavy, tak

agregace–účast jsou rozvětvené vztahy, lze vytvořit strukturální hierarchie těchto vztahů (stejně jako generalizace-specializace, o nichž se pojednává v několika kapitolách).

Zvažme objekt **Město**, jehož hierarchie vlastností je znázorněna na obr. 18.9. Tři důležité atributy **Města**, kromě jeho **Názvu**, jsou **Poloha**, **Počet obyvatel** a **Podnebí**. **Poloha**, **Počet obyvatel** a **Podnebí** nejsou pouze atributy **Města**, ale jsou také samostatnými objekty, takže každý z nich může mít vlastní sadu vlastností nebo částí. **Poloha** má atributy **Kontinent**, **Země**, **Region** a **Soubor souřadnic**. **Počet obyvatel** má atributy **Velikost** a **Demografie** a operace **Stárnutí** a **Příjem**. **Demografie** se skládá z **Průměrného věku** a **Průměrného příjmu**. **Stárnutí** a **Příjem** jsou dvě operace, které ovlivňují dvě části **Demografie**, a **Srážky** jsou operací **Klimatu**, která ovlivňuje **Průměrné srážky**.

18.7 Problém pojmenování atributů

Přirozené jazyky nám často poskytují konkrétní podstatné jméno pro pojmenování atributu. Například atribut, jehož dva extrémy jsou přídavná jména „krátký“ a „dlouhý“, se nazývá **Délka**. Atribut, jehož dvě extrémní přídavná jména jsou „úzký“ a „široký“, se nazývá **Šířka**, a atribut, jehož dva extrémy jsou „těžký“ a „lehký“, se nazývá **Hmotnost**. Někdy je název atributu (podstatné jméno) odvozen od stejného kořene (základního slova) jako jedna z (často krajních) hodnot (přídavné jméno) v spektru možných hodnot pro daný atribut. Příklady takových párů atribut–hodnota (podstatné jméno–přídavné jméno) jsou **Délka—dlouhý**, **Šířka—široký**, **Připravenost—připravený** a **Krása—krásný**. Z těchto párů může být kořen buď název atributu—podstatné jméno (např. **Krása**), nebo název jedné z hodnot tohoto atributu—přídavné jméno (např. **připravený**).



City exhibits Location, Name, Climate, and Population. Location of City consists of Continent, Country, Region, and Coordinate Set. Climate of City consists of Average Temperature and Average Precipitation. Average Precipitation exhibits Precipitating. Precipitating affects Average Precipitation. Population of City exhibits Size and Demographics, as well as Aging and Earning. Demographics of Population consists of Average Age and Average Income. Aging affects Average Age. Earning affects Average Income.

Obr. 18.9 Příklad strukturální hierarchie města

Názvy některých atributů jsou neutrální podstatná jména, zatímco jiné jsou převzaty z jedné z extrémních hodnot atributu a jsou k ní zaujaté. Atribut **Tvar** je například neutrální podstatné jméno. Jeho hodnoty

mohou být přídavná jména **kulatý**, **čtvercový**, **eliptický** atd. Vlastnost **Tvar** není zaujatá vůči žádné ze svých hodnot. Naopak vlastnost **Délka** je zaujatá vůči **dlouhému** extrému spektra hodnot **krátký–dlouhý**. Zvolení **atributu „krátký“** by posunulo zaujatost na druhý extrém. Proto věta jako „Tvar domu je čtvercový“ dává dokonalý smysl, zatímco „Délka tyče je dlouhá“, ačkoli je syntakticky správná, je sémanticky nevhodná. Vynecháme-li název atributu, řekneme spíše „Tyč je dlouhá“. V tomto případě je **atribut „délka“** ve větě implicitní. Mohli bychom také vynechat název atributu **Tvar** ve větě „Tvar domu je čtvercový“ a říci „Dům je čtvercový“. Takový atribut nazýváme *implicitním*. Věty s implicitními atributy se obvykle používají, když je název atributu převzat z jedné z jeho extrémních hodnot. Příklady jsou **Délka**, převzatá z dvojice **dlouhý–krátký**, **Krása**, převzatá z dvojice **krásný–ošklivý**, a **Šířka**, převzatá z dvojice **široký–úzký**. Zajímavé je, že při výběru toho, který z extrémů bude zvolen jako název atributu, se upřednostňuje ten, který je považován za lepší nebo větší. Je tedy mnohem méně přirozené pojmenovat tyto atributy **krátkost**, **ošklivost** a **úzkost**, i když se jedná o platná podstatná jména.

Používání implicitních atributových vět v přirozeném jazyce je spíše pravidlem než výjimkou. Nejčastěji se vynechává název atributu, ke kterému hodnota patří, a odkazuje se přímo na objekt, který tuto hodnotu vykazuje. Implicitní atributy jsou tak rozšířené, že v mnoha případech přirozený jazyk nemá pro samotný atribut vyhrazené podstatné jméno, zatímco přídavná jména, která jsou hodnotami nebo stavy tohoto atributu, mají široce uznávané a používané názvy.

Jako příklad uvažujme implicitní atributovou větu „Tato kniha je zajímavá.“ Přídavné jméno **zajímavá** odkazuje na atribut této knihy, jehož možnými hodnotami mohou být „zajímavá“ a „**nudná**“. Neexistuje žádné jediné podstatné jméno pro atribut, jehož hodnoty jsou **zajímavé** a **nudné**. Plausibilní názvy tohoto atributu mohou být buď „**úroveň zájmu**“ nebo „**úroveň nudy**“. Každý z nich je však zaujatý vůči jednomu z extrémů spektra nebo druhému. V ideálním případě bychom chtěli slovo, které je neutrální a není zaujaté vůči žádné z možných hodnot atributu.

V jiných případech je zřejmé, že název atributu byl vymyšlen až poté, co byla hodnota již používána. Například „**Laziness**“ (**lenost**) je název atributu, který má jako jednu ze svých hodnot „**lazy**“ (**lenivý**) a jako další „**energetic**“ (**energický**), „**industrious**“ (**pracovitý**) nebo „**hardworking**“ (**pilný**), a přípona „**ness**“ naznačuje jeho pozdější zavedení do jazyka. Je zřejmé, že pokud k **osobě** přiřadíme atribut nazvaný **Lenost**, očekáváme, že hodnotou tohoto atributu bude spíše **lenost** než **pracovitost**. Jednodušeji a přirozeněji bychom chtěli říci, že „**osoba je líná**“. Tato věta je mnohem kratší, jasnější a přímočařejší ve srovnání se dvěma větami v OPL „**osoba vykazuje lenost**“ a „**lenost osoby je líná**“. Jak bylo diskutováno v sekci

18.8 OPM má možnost implicitního atributu, kde jsou „**lenivý**“ a „**pracovitý**“ přímo modelovány jako stavy **osoby**, nikoli jako hodnoty jejího atributu „**Lenost**“, který se v tomto případě stává nadbytečným.

V obecné rovině však v OPM, kde je modelování formální, musíme často explicitně modelovat atribut, než můžeme modelovat jeho stavy nebo hodnoty, a pokud pro atribut neexistuje slovo, musíme jej vymyslet. V OPM skutečně existuje problém s nalezením vhodných názvů pro vlastnosti (atributy metamodelu) **Thing**. Název vlastnosti **Thing**, jejíž hodnoty jsou **přirozené** a **umělé**, je **Origin**. Vlastnost **Thing**, jejíž hodnoty jsou **fyzické** a **informační**, jsme také nazvali **Essence**. Jako název pro vlastnost, jejíž hodnoty jsou **trvalé** (v tomto případě je věc objektem) a **přechodné** (v tomto případě je věc procesem), bylo zvoleno **Perseverance**. Volba těchto názvů vlastností poukazuje na obtížnost najít správné slovo pro pojmenování atributu (nebo vlastnosti), jehož hodnoty jsou převládající. Například **přechodné** a **trvalé**, které jsou hodnotami vlastnosti **Perseverance**, jsou

široce používané, zatímco **vytrvalost** není v souvislosti s těmito adjektivy uznávána. **Původ** a **podstata** jsou neutrální. **Vytrvalost** je méně neutrální; American Heritage Dictionary (1996) definuje vytrvalost jako „stálou vytrvalost v dodržování postupu, přesvědčení nebo účelu; neochvějnost“. *Stálá vytrvalost* se přiklání k pojmu **Objekt**, protože její hodnota **Vytrvalost** je skutečně **trvalá**. *Postup* však má pojem procesu...

18.8 Vlastnosti prvků a odkazů

Funkce a **odkazy** mají několik vlastností (atributy na úrovni metamodelu), které jsou popsány v této části. Patří mezi ně **explicitnost**, **režim**, **dotek** a **vznik**. Některé z těchto vlastností se vztahují k **funkci** obecně, tj. jak k **atributu**, tak k **operaci**, zatímco jiné se vztahují pouze k **atributu**.

18.8.1 Explicitnost

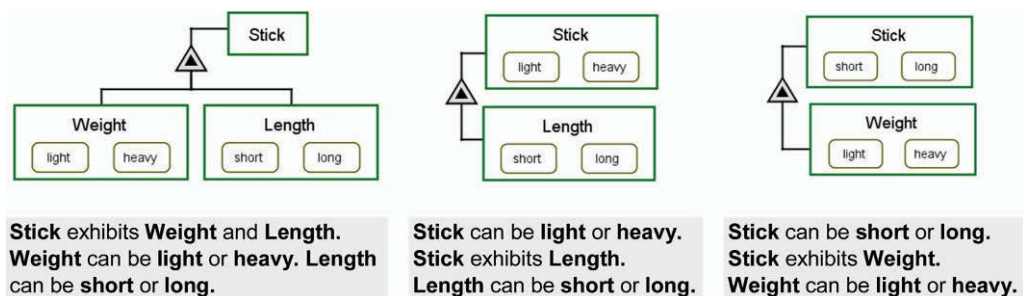
OPM vychází vstříc přirozené tendenci přirozeného jazyka přeskočit atributy a přejít přímo k jejich hodnotám, jak je popsáno v oddíle

18.7, a to tím, že poskytuje možnost implicitního modelování atributů, jak ukazuje [obr. 18.10](#).

*Atribut je **implicitní**, pokud jsou jeho hodnoty přiřazeny jako stavy přímo vystavovateli bez specifikace názvu atributu.*

*Atribut je **explicitní**, pokud se jedná o samostatný objekt, který je propojen s vystavovatelem vztahem charakterizace výstavy.*

***Explicitnost** je atributem atributu, jehož hodnoty jsou explicitní (výchozí nastavení) a implicitní.*



Obr. 18.10 Modelování explicitních a implicitních atributů

Implicitní atribut lze snadno identifikovat: Pokud má objekt stavy, které jsou umístěny přímo uvnitř jeho obdélníku, nikoli v jeho atributu, pak je atribut, jehož hodnoty jsou uvnitř objektu, implicitní. Ve výchozím nastavení je atribut explicitní – jeho hodnota **explicitnosti** je **explicitní**. Často má smysl použít **implicitní** atribut, protože to obchází problém pojmenování atributů diskutovaný v předchozí části – nutnost vymyslet název pro atribut. V předchozí části jsme viděli příklad **lenosti**. Dalším

příklad, **lampa** může být **zapnutá** nebo **vypnutá**. Bylo by obtížné definovat pro tyto stavy speciální explicitní atribut a najít pro něj vhodný název: „Zapnutost“? „Vypnutost“? „**Provozní stav**“? Žádný z těchto názvů nedává smysl.

Není možné mít více než jeden implicitní atribut pro stejnou věc, protože by to vedlo ke smíchání hodnot různých atributů ve stejné větě, aniž by byly přiřazeny správným „vlastnickým“ atributům. Například při pohledu na obr. 18.10 vidíme, že věty jako „**Tyč** může být **lehká**, **těžká**, **krátká** nebo **dlouhá**“ nedávají smysl, protože jsou v nich smíchány hodnoty **hmotnosti** a **délky tyče**. Jako implicitní atributy můžeme mít buď **hmotnost**, nebo **délku**, ale ne obojí. V modelu OPM vlevo na obr. 18.10 jsou **hmotnost** i **délka** explicitními atributy **tyče**. Uprostřed je **délka** explicitním atributem s hodnotami **dlouhá** a **krátká**, zatímco **hmotnost** je implicitní s stavy **lehká** a **těžká**. V modelu OPM vpravo je tomu naopak: **Hmotnost** je explicitní atribut s hodnotami **lehká** a **těžká**, zatímco **Délka** je implicitní.

18.8.2 Režim

Některé atributy jsou **kvalitativní**, zatímco jiné jsou **kvantitativní**. Viděli jsme příklad atributu **Tvar domu**, kde možné hodnoty mohou být **kulatý**, **čtvercový** a **obdélníkový**. Tyto hodnoty nelze kvantifikovat číselnou hodnotou. Jsou pouze kvalitativně odlišné od sebe navzájem. Proto říkáme, že **tvar** je kvalitativní atribut. Mezi další příklady kvalitativních atributů patří **nálada** (**šťastný**, **smutný**, **rozzlobený** atd.), **zdraví** (**zdravý**, **nemocný**) a **rodinný stav** (**svobodný**, **ženatý**, **rozvedený** atd.). Příklady kvantitativních atributů jsou **hmotnost** [kg] a **výška** [m]. Jak ukazují tyto příklady, kvantitativní atributy musí být následovány jednotkou měření v závorkách, jak je popsáno v kapitole 22.

22. Vzhledem k tomu, že atribut může být kvalitativní nebo kvantitativní, **kvalitativní** a **kvantitativní** jsou hodnoty vlastnosti **atributu** zvané **mód**.

*Atribut je **kvantitativní**, pokud jsou jeho hodnoty numerické nebo*

*parametrické. Atribut je **kvalitativní**, pokud jsou jeho hodnoty*

nenumerické.

*Operace je **kvantitativní**, pokud transformuje kvantitativní atribut, jinak je **kvantitativní**.*

***Režim** je vlastnost prvku, která určuje, zda je kvalitativní (výchozí nastavení) nebo kvantitativní.*

Definice pojmu „*numerický*“ zde zahrnuje i parametry – parametr je symbol, který představuje určitou numerickou hodnotu. Numerické hodnoty nebo „kódy“ můžeme přiřadit hodnotám kvalitativního atributu, například **svobodný** = 1; **ženatý** = 2. To bylo běžnou praxí v raných systémech zpracování informací a stále se často používá, zejména když je třeba data analyzovat statisticky. Sémanticky to však neznamená, že kvalitativní atribut se stává kvantitativním.

Příkladem kvantitativních operací je **měření výšky**, které vytváří hodnotu pro kvantitativní atribut **výška**. Dalším příkladem je **vážení**, které vytváří hodnotu pro kvantitativní atribut **hmotnost**. V oddíle 13.10 se popisuje, jak modelovat nastavení nebo aktualizaci hodnot pomocí procedurálních odkazů specifikovaných hodnotou.

18.8.3 Dotyk: vlastnost kvantitativního atributu

Kvantitativní atribut může být **tvrdý** nebo **měkký**, v závislosti na tom, zda jej lze vypočítat z jiných atributů. Například **datum narození osoby** je tvrdý atribut, zatímco **věk osoby** je měkký atribut. Známe-li **datum narození osoby** a aktuální hodnotu **data**, lze vypočítat **věk osoby**. Dalším příkladem je **hmotnost** jednotlivých částí **letadla**, která je tvrdým atributem, zatímco celková **hmotnost letadla** je měkkým atributem, protože ji lze vypočítat sečtením hmotností jednotlivých částí. Název vlastnosti **atributu**, jehož hodnoty jsou **tvrdé** a **měkké**, je **Touch**.

*Kvantitativní atribut je **tvrdý**, pokud jeho hodnotu nelze odvodit nebo vypočítat z jiných atributů.*

*Kvantitativní atribut je **měkký**, pokud lze jeho hodnotu odvodit nebo vypočítat z jiných atributů.*

***Dotyk** je vlastnost kvantitativního atributu, která určuje, zda je tvrdý (výchozí nastavení) nebo měkký.*

Rozhodnutí, zda by měl být měkký atribut předem vypočítán, má praktické důsledky během fáze podrobného návrhu informačního systému. Předem vypočítané hodnoty mohou být uloženy pro rychlou odezvu na úkor úložného prostoru. Alternativně mohou být měkké atributy vypočítány na požádání, což šetří místo, ale také prodlužuje dobu odezvy informačního systému. Jedná se o běžný kompromis v databázích, kde se zvažuje potřeba vysoké rychlosti odezvy oproti režii uložení.

18.8.4 Vznik

V závislosti na tom, zda je vlastnost projevována pouze objektem jako celkem, nebo pouze jednou nebo více (ale ne všemi) jeho částmi, může být **vlastnost (atribut nebo operace) inherentní** nebo **emergentní**.

*Vlastnost objektu je **inherentní**, pokud ji vykazuje alespoň jedna z částí objektu.*

*Vlastnost objektu je **emergentní**, pokud ji nevykazuje žádná z částí objektu samostatně.*

***Emergence** je vlastnost objektu, jehož hodnoty jsou inherentní (výchozí) a emergentní.*

Abychom pochopili rozdíl mezi emergentními a inherentními vlastnostmi, podívejme se na atribut **Letadlo**, jeho **vlastnost Hmotnost** a jeho operaci **Létání**. **Hmotnost letadla** je součtem jednotlivých hodnot **hmotnosti** všech částí, z nichž se **letadlo** skládá. **Létání** naopak není operací, kterou by mohla provádět kterákoli část **letadla** samostatně. Tato vlastnost spíše vzniká z jedinečného souboru částí **letadla**, které **letadlu** propůjčují schopnost provádět operaci **létání**. **Létání** je tedy **emergentní** vlastností (v tomto případě operací) **letadla**, zatímco **hmotnost** je **inherentní** vlastností (v tomto případě atributem) **letadla**.

V systémech jsou operace často emergentní, protože systémy jsou budovány s úmyslem dosáhnout určité funkce, která není lokalizována v žádné části systému ani není dosažitelná žádnou částí systému samostatně. **Létání letadla** je vynikajícím příkladem. Bar-Yam (1997) rozlišuje mezi jednoduchými a složitými systémy a tvrdí, že

že složitost může vzniknout ze souboru jednoduchých částí, které tvoří systém. Opak může být také pravdou: systém složený ze složitých částí může vykazovat jednoduché chování v širším měřítku. Například planeta Země je velmi složitý systém, ale při pohledu z perspektivy jejího pohybu kolem Slunce je relativně jednoduchá, což poukazuje na relativitu pojmu složitost.

18.8.5 Vlastnost homogenity spojení

Vlastnost, která určuje, zda odkaz spojuje věci se stejnou **vytrvalostí** – **statickou** (trvalou, definující objekt) nebo **dynamickou** (přechodnou, definující proces), se nazývá **homogenita**. Hodnoty **homogenity** jsou **homogenní**, což platí, pokud dvě věci, které odkaz spojuje, vykazují stejnou **vytrvalost** (obě jsou objekty nebo obě jsou procesy), a v opačném případě **nehomogenní** (jedna je objekt a druhá proces). Vzhledem k tomu, že většina strukturálních odkazů je mezi dvěma objekty nebo mezi dvěma procesy, je hodnota **homogenity** **homogenní** výchozí hodnotou pro strukturální odkazy. Naopak, vzhledem k tomu, že většina procedurálních odkazů je mezi objektem a procesem, je hodnota **homogenity** **nehomogenní** výchozí hodnotou pro procedurální odkazy.

*Spojení je **homogenní**, pokud spojuje dvě věci, které vykazují stejnou hodnotu vytrvalosti.*

*Odkaz je **nehomogenní**, pokud spojuje dvě věci, které vykazují opačné hodnoty vytrvalosti.*

***Homogenita** je vlastnost odkazu, jehož hodnoty jsou homogenní (výchozí nastavení pro strukturální odkazy) a nehomogenní (výchozí nastavení pro procedurální odkazy).*

Téměř všechny strukturální vazby jsou pouze **homogenní**: spojují buď dva objekty, nebo dva procesy. Jedinou výjimkou je strukturální vazba **Výstava-Charakterizace**, která může být jak **homogenní** (v případě, že spojuje objekt s atributem nebo proces s operací), tak **nehomogenní** (v případě, že spojuje objekt s operací nebo proces s atributem). Všechny ostatní strukturální vazby, a zejména zbývající tři základní strukturální vztahy, jsou **homogenní**. Analogicky jsou téměř všechny procedurální vazby **nehomogenní**, protože spojují objekt s procesem. Jediné procedurální vazby, které jsou **homogenní**, jsou vazba vyvolání diskutovaná v oddíle [10.10.3](#) a vazby výjimky přesčas a podčas diskutované v [kapitole 22](#).

18.9 Shrnutí

- **Výstava-charakterizace** je vztah mezi věcí a vlastnostmi, které ji charakterizují.
- Zkrácený název tohoto vztahu je **charakterizace** a jeho symbol je Δ .
- Charakterizace je jediný základní strukturální vztah, pro který jsou možné všechny čtyři kombinace objektu a procesu, jako vystavovatel a vlastnost.
- Vlastnost, která je objektem, se nazývá **atribut**, zatímco vlastnost, která je procesem, je **operací**.

- Atribut je **implicitní**, pokud jsou jeho hodnoty přiřazeny přímo vystavovateli bez specifikace názvu atributu.
- Atribut je **explicitní**, pokud je samostatným objektem, který je propojen s vystavovatelem vztahem výstava-charakterizace.
- **Explicitnost** je atribut atributu, jehož hodnoty jsou explicitní (výchozí) a implicitní.
- Atribut je **kvalitativní**, pokud jeho hodnoty nejsou číselné.
- Atribut je **kvantitativní**, pokud jsou jeho hodnoty číselné.
- Operace je **kvantitativní**, pokud transformuje kvantitativní atribut, jinak je **kvalitativní**.
- **Režim** je vlastnost prvku, která určuje, zda je kvalitativní (výchozí nastavení) nebo kvantitativní.
- Kvantitativní atribut je **tvrdý**, pokud jeho hodnotu nelze odvodit nebo vypočítat z jiných atributů.
- Kvantitativní atribut je **měkký**, pokud jeho hodnotu lze odvodit nebo vypočítat z jiných atributů.
- **Dotyk** je atribut kvantitativního atributu, který určuje, zda je tvrdý (výchozí nastavení) nebo měkký.
- Vlastnost objektu je **inherentní**, pokud ji vykazuje alespoň jedna z částí objektu.
- Vlastnost objektu je **emergentní**, pokud ji nevykazuje žádná z částí objektu samostatně.
- **Emergence** je vlastnost objektu, jehož hodnoty jsou inherentní (výchozí) a emergentní.
- Odkaz je **homogenní**, pokud spojuje dvě věci, které vykazují stejnou hodnotu vytrvalosti.
- Odkaz je **nehomogenní**, pokud spojuje dvě věci, které vykazují opačné hodnoty vytrvalosti.
- **Homogenita** je vlastnost odkazu, jehož hodnoty jsou homogenní (výchozí nastavení pro strukturální odkazy) a nehomogenní (výchozí nastavení pro procedurální odkazy).

18.10 Problémy

1. Pro každou ze čtyř kombinací vystavovatel-vlastnost nakreslete OPD, které není uvedeno jako příklad v této kapitole.
2. „**The quick brown fox jumps over the lazy dog**“ je anglická věta zvaná pangram – fráze, která obsahuje všechna písmena abecedy. Vytvořte model OPM této věty, ve kterém **Jumping** je operací **Fox**.
3. V modelu, který jste vytvořili v předchozí otázce, změňte každý explicitní atribut na implicitní a naopak.
4. Uveďte dva příklady inherentních vlastností a dva příklady emergentních vlastností.
5. Vytvořte model OPM struktury – částí a vlastností – pohřebního vozu z Pazyryku v Ermitáži v Petrohradě podle následujícího popisu (obrázek k dispozici na URL).

Tento velký čtyřkolový vůz je jedním z nejpozoruhodnějších nálezů z pohřebních mohyl v Pazyryku. Skládá se z řady částí spojených koženými řemínky a dřevěnými hřebíky. Korba je tvořena dvěma rámy spojenými krátkými vyřezávanými tyčemi a koženými řemínky. Rámy tvoří základ pro stříšku. Každé ze čtyř velkých kol má 34

paprsků. Nápravy nemají otočné zařízení a vzdálenost mezi zadními a předními koly je pouze 5 cm, což znamenalo, že vůz mohl být používán pouze na rovném povrchu. Mohl však být snadno rozebrán a přepravován na koních. Díky permafrostu je vůz ve výborném stavu.

Kapitola 19

Státy a hodnoty

Housenka ... slezla z houby a odplazila se do trávy, přičemž pouze poznamenala: „Na jedné straně porostete do výšky a na druhé straně se zmenšíte.“

Alenka v říši divů. Lewis Carroll, 1899

Abychom mohli výslovně hovořit o změně objektu v čase, přiřadíme mu řadu možných „legálních“ stavů. Stav je tedy situace, ve které se objekt může nacházet. Stav a hodnoty dodávají OPM expresivitu. Hodnota je stav atributu. Jako taková je specializací stavu: zatímco objekty mohou mít stavy, pouze stavy atributů, které jsou objekty popisujícími jiné objekty, se nazývají hodnoty. Stav a hodnoty umožňují modelovat změnu objektu, přičemž tento objekt si zachovává svou identitu. Termíny stavy a hodnoty používáme zcela intuitivně již od prvních kapitol této knihy. Pokud jsou objekty a procesy stavebními kameny OPM a odkazy jsou maltou, stavy lze považovat za povrchovou úpravu domu: náter, nábytek a architektonické prvky. V jakémkoli okamžiku života objektu, kdy na něj nepůsobí žádný proces, se tento objekt nachází v jednom ze svých stavů. Příčina a účinek jsou úzce spojeny s pojmy změny stavu v čase. Tato kapitola formalizuje pojmy stavů a hodnot a ukazuje, jak je lze použít ke zvýšení expresivity modelu.

19.1 Definice stavu

Abychom mohli explicitně hovořit o změně objektu, přiřadíme mu řadu vzájemně se vylučujících situací, pozic nebo hodnot, které označujeme jako *stavy*.

Stav je situace nebo pozice, ve které může objekt existovat po určitou dobu během své existence.

19.1.1 Výčet stavů

Příkladem platných stavů **planety** je **viditelná** a **neviditelná**, a věta OPL, která to specifikuje, je „**Planeta** může být **viditelná** nebo **neviditelná**“. **Planeta** může změnit svůj **neviditelný** stav na **viditelný** stav tím, že se zvedne nad obzor a když nejsou žádné mraky. Věta OPL s výčtem stavů, jako je „**Planeta** může být **viditelná** nebo **neviditelná**“, vyjmenovává všechny stavy, ve kterých se objekt může nacházet. Začíná názvem objektu, následuje vyhrazená fráze „může být“ (nebo „je“ v případě pouze jednoho stavu), za kterou následuje seznam stavů, které jsou v případě tří nebo více stavů odděleny čárkami, a končí vyhrazenou frází „nebo“ mezi posledním a předposledním stavem. Objekt nemůže být ve více než jednom stavu najednou. Sémantika věty s výčtem stavů je tedy logická exkluzivní OR, zkráceně XOR. Výchozí velká písmena názvu stavu jsou malá písmena.

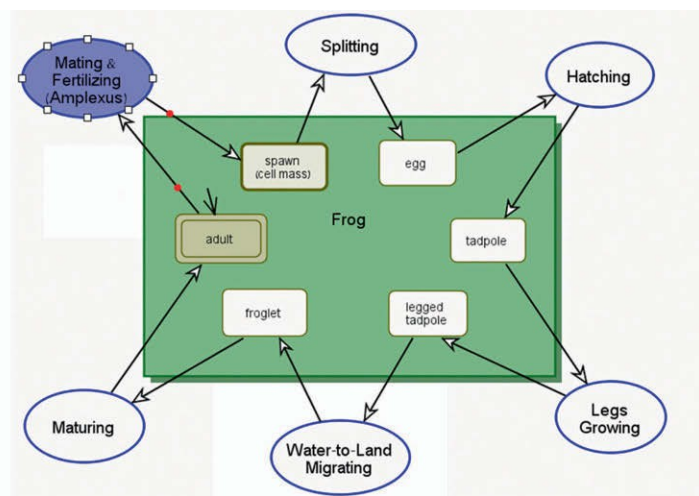
19.1.2 Počáteční, konečné a výchozí stavy

Často je výhodné nebo žádoucí specifikovat, jaký je *počáteční stav*, *konečný stav* a *výchozí stav* objektu.

Počáteční stav objektu *B* je stav, ve kterém se *B* nachází při svém vytvoření nebo při spuštění systému.

Konečný stav objektu *B* je stav, ve kterém se *B* nachází při jeho spotřebě nebo při dokončení provádění systému.

Výchozí stav objektu *B* je stav, ve kterém se *B* nachází, pokud není jeho stav specifikován.



adapted from www.tooter4kids.com/Frogs/lifecycle.jpg

Frog is physical.

Frog can be **spawn (cell mass)**, **egg**, **tadpole**, **legged tadpole**, **froglet**, or **adult**.

Frog is initially **spawn** and finally **Adult**.

State **adult** of **Frog** is default.

Mating & Fertilizing (Amplexus) changes **Frog** from **adult** to **spawn (cell mass)**.

Splitting changes **Frog** from **spawn (cell mass)** to **egg**.

Hatching changes **Frog** from **egg** to **tadpole**.

Legs Growing changes **Frog** from **tadpole** to **legged tadpole**.

Maturing changes **Frog** from **froglet** to **adult**.

Water-to-Land Migrating changes **Frog** from **legged tadpole** to **froglet**.

Obr. 19.1 Počáteční, konečný a výchozí stav znázorněný v životním cyklu žáby pomocí simulace. Počáteční stav je spawn (tučný rámeček) a dospělý jedinec je jak konečným stavem (dvojitý rámeček), tak výchozím stavem (otevřená šipka směřující k dospělému jedinci). Simulace zdůrazňuje uzavření životního cyklu.

Objekt může mít nulový nebo více počátečních stavů, nulový nebo více konečných stavů a maximálně jeden výchozí stav. Stejný stav může být libovolnou kombinací počátečního, konečného a/nebo výchozího stavu. Počáteční a konečné stavy jsou zvláště užitečné pro objekty, které vykazují vzorec životního cyklu, jako je produkt, systém nebo naše známá žába z kapitoly 13. Výchozí stav je užitečný pro určení stavu, ve kterém se objekt nachází, pokud není specifikován žádný stav. Symboly pro počáteční, konečný a výchozí stav jsou tučný stavový rámeček, dvojitý stavový rámeček a stavový rámeček označený otevřenou šipkou. Tyto jsou znázorněny v simulovaném životním cyklu žaby na obr. 19.1: Počáteční stav žaby je **spawn (buněčná hmota)**, označený tučným stavovým rámečkem. Stav **adult** je jak konečným stavem, označeným dvojitým rámečkem, tak výchozím stavem – otevřená šipka směřuje na rámeček stavu **adult**.

Koncepční simulace na obr. 19.1 ukazuje proces **Páření a oplodnění (Amplexus)** – zvýrazněná plná elipsa – působící na žabu, aby ji změnil ze stavu **dospělý** do stavu **jikry (buněčná hmota)**. Odpovídající věta OPL je:

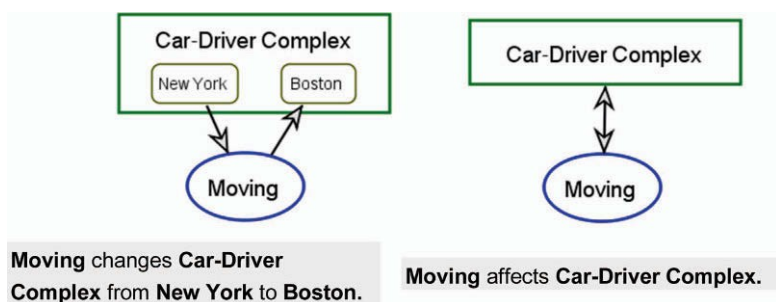
Páření a oplodnění (Amplexus) mění žabu z dospělého stavu na stav jikry (buněčná hmota).

Tento přechod mezi stavy zdůrazňuje cyklickou povahu žaby, protože konečný (a výchozí) stav žaby, **dospělý**, vede k počátečnímu stavu žaby, **jikře (buněčné hmotě)**, prostřednictvím procesu **páření a oplodnění (Amplexus)**.

19.2 Potlačení a vyjádření stavu

Odstranění symbolů stavu z objektu se nazývá *potlačení stavu*. Potlačení stavu je jednou z několika možností abstrakce. Abstrakce je prostředek ke zjednodušení OPD za cenu skrytí detailů souvisejících s věcmi v OPD. Jak lze očekávat, jak OPD, tak jejich ekvivalentní OPL věty ukazují, že potlačení stavu eliminuje informace o tom, jak přesně proces ovlivňuje objekt. Tyto informace mohou být poskytnuty v OPD nižší úrovni, kde jsou stavy procesu explicitně vyjádřeny.

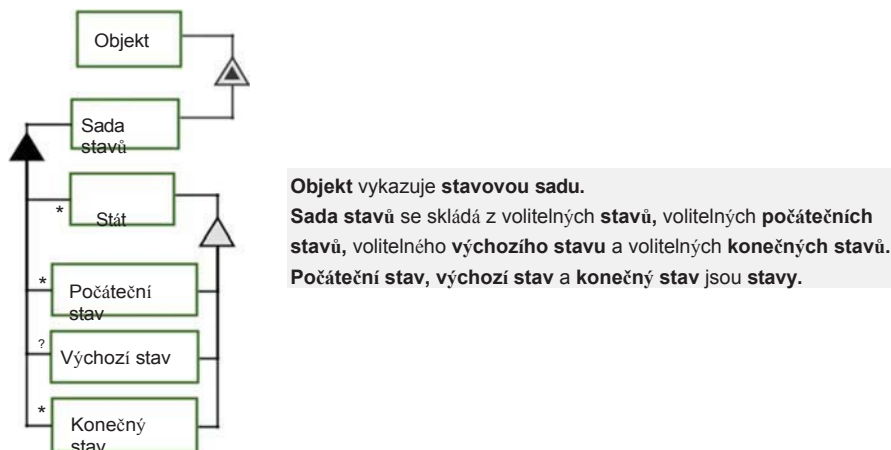
Opakem potlačení stavu je *vyjádření stavu*: zdokonalení OPD přidáním relevantních stavů. Jak ukazuje obr. 19.2, zatímco vyjádření stavu je doprovázeno rozdělením efektového spojení na jeho vstupní a výstupní složky, potlačení stavu je doprovázeno sloučením páru vstupně-výstupních spojení do jediného efektového spojení.



Obr. 19.2 Příklad potlačení stavu

19.2.1 Specializace stavů a jejich omezení účasti

Obrázek 19.3 je metamodel **objektu**, který ukazuje specializace stavu – **počáteční stav**, **konečný stav** a **výchozí stav** a jejich omezení účasti. To je také specifikováno v OPL napravo od OPD.



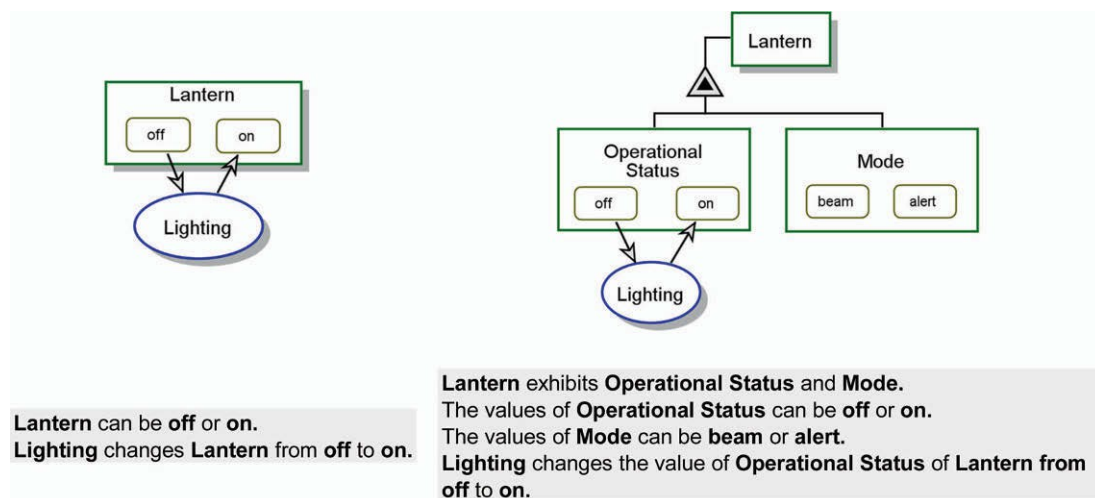
Obr. 19.3 Metamodel **objektu** zobrazující specializace **stavu**: **počáteční stav**, **konečný stav** a **výchozí stav** s jejich omezeními účasti

V metamodelu na obr. 19.3 jsou **stav** a jeho tři specializace – **počáteční stav**, **výchozí stav** a **konečný stav** – objekty. Tento metamodel specifikuje omezení účasti pro tři specializace **stavu**. OPL na pravé straně uvádí, že **State Set** může sestávat z „volitelných **počátečních** stavů“ a „volitelných **konečných** stavů“, tj. nulového, jednoho nebo více počátečních stavů a nulového, jednoho nebo více konečných stavů. Obvykle má objekt nanejvýš jeden počáteční stav, ale může mít i více než jeden. Například některý proces může vytvořit objekt v jednom počátečním stavu, zatímco jiný proces může vytvořit stejný objekt v jiném počátečním stavu. Alternativně, jak ukážeme níže, stejný proces může vytvořit objekt stochasticky v jednom ze dvou nebo více počátečních stavů. Objekt může mít také více než jeden konečný stav, ze kterého nemůže vystoupit. **Sada stavů** však může sestávat z „volitelného **výchozího** stavu“, tj. může existovat nejvýše jeden výchozí stav.

19.3 Hodnota: specializace stavu

Hodnota je stav atributu.

Jelikož hodnota je stavem atributu, jedná se o specializaci stavu. Nuance v sémantice mezi stavem a hodnotou je znázorněna na obr. 19.4, kde v OPD vlevo jsou **off** a **on** stavy objektu **Lantern**, zatímco vpravo jsou **off** a **on** hodnotami atributu **Operational Status** objektu **Lantern**.



Obr. 19.4 Příklad rozdílu mezi stavem a hodnotou. Vlevo: off a on stavy objektu Lantern. Vpravo: off a on jsou hodnoty atributu Operational Status objektu Lantern

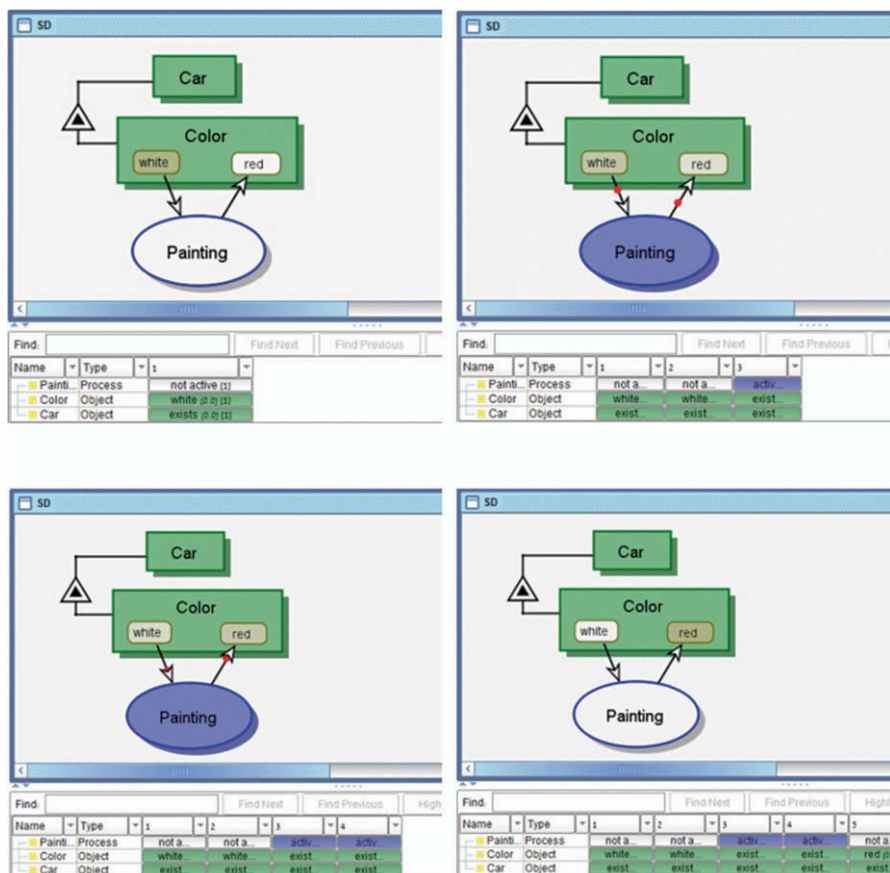
19.4 Přechod stavu: Když je proces aktivní

V daném okamžiku může být objekt maximálně v jednom ze svých stavů. Říkáme „maximálně“, protože objekt může být také v přechodu mezi dvěma stavy – vstupním stavem a výstupním stavem subjektu, na který má proces aktuálně vliv. V době, kdy probíhá proces ovlivňující objekt, objekt již opustil svůj vstupní stav, ale ještě nevstoupil do svého výstupního stavu. Jedná se o nestabilní situaci objektu, ke které dochází, když proces mění objekt ze vstupního stavu – stavu, ve kterém se objekt nacházel před zahájením procesu, do výstupního stavu – stavu, ve kterém se bude nacházet po skončení procesu. Během této doby objekt prochází *přechodem stavu*.

Přechod do jiného stavu je pro objekt nestabilním obdobím, ke kterému dochází, když na něj působí proces, který mění jeho stav.

Zvažte následující příklad lakování automobilu. Když je **bílý automobil** lakován **červenou barvou**, jeho vstupní stav (hodnota atributu **Barva** při vstupu do lakovny) je **bílá**. To je znázorněno v levém horním OPD na [obr. 19.5](#) zvýrazněním stavu **bílá** v atributu **Barva**. Výstupní stav **automobilu** (hodnota atributu **Barva** při opuštění lakovny) je **červená**. To je znázorněno v pravém dolním OPD na [obr. 19.5](#) zvýrazněním stavu **červená** atributu **Barva**. Mezi těmito dvěma stabilními stavy dochází k **lakování**. Během tohoto časového intervalu, kdy je **auto** lakováno, tj. po celou dobu lakování, které může trvat několik dní, hodnota atributu **Barva** již není zcela bílá, ale ještě není ani červená. Ve skutečnosti se **auto** během lakování nachází v *přechodném stavu* mezi dvěma stavy. Říkáme, že během procesu **lakování** je **barva auta nestabilní**. To je znázorněno v pravém horním a levém dolním OPD na [obr. 19.5](#), kde se zvýraznění **červeného** a **bílého** stavu postupně mění z **červené** a **bílé**.

Doba přechodu, tedy doba, kdy **auto** není ani zcela **červené**, ani zcela **bílé**, se rovná době trvání procesu lakování.



Obr. 19.5 Systém lakování auta v akci: Auto začíná jako bílé (vlevo nahoře) a končí jako červené (vpravo dole). Diagramy ve spodní části každého OPD jsou diagramy životnosti

Jak ukazuje příklad lakování auta, objekty a procesy v systému mají *historii*, která se hromadí, jak systém plní svou funkci. Historie objektu začíná v okamžiku, kdy je vytvořen a stává se identifikovatelnou entitou, a končí v okamžiku, kdy je spotřebován, takže již není stejnou identifikovatelnou entitou. Historie zahrnuje časový záznam o tom, kdy byl objekt vytvořen, jakým procesem, jaké změny stavu objekt prošel, zatímco si zachoval svou identitu, kdy byl objekt spotřebován a jakým procesem.

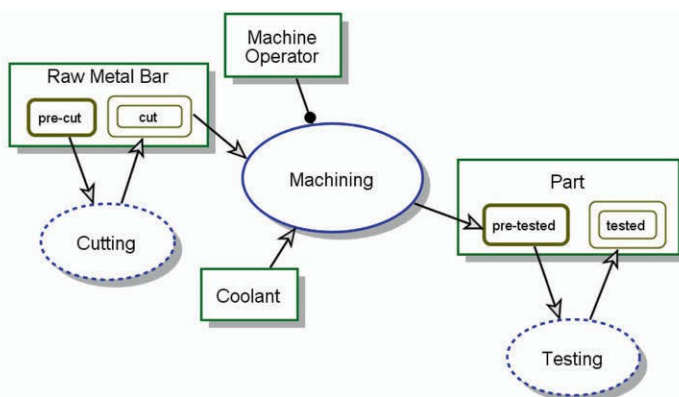
Historie má smysl pouze ve vztahu k konkrétnímu provedení systému, tj. systému na operační úrovni nebo úrovni instance, ale nikoli na koncepční úrovni nebo úrovni třídy, protože pouze když systém vykonává svou funkci, je možné sledovat a zaznamenávat, která instance procesu byla spuštěna a ukončena

kdy a jaká instance objektu byla transformována, zda byla vytvořena nebo spotřebována, nebo zda byl změněn její stav.

Historie procesu zahrnuje pro každé provedení každého procesu v systému čas, kdy byl spuštěn a ukončen. Konkrétní provedení procesu tvoří instanci procesu. Historie také zahrnuje instance transformátoru a aktivátoru v související sadě objektů. Užitečným nástrojem pro prohlížení, sledování a analýzu historie objektů a jejich stavů a procesů v systému je *diagram životnosti*, který OPCAT skutečně obsahuje.

Diagram životnosti je diagram, který pro jakýkoli okamžik během životnosti systému ukazuje, jaké objekty v systému existují, v jakém stavu se každý objekt nachází a jaké procesy jsou aktivní.

Čtyři diagramy životnosti zobrazené ve spodní části každého ze čtyř OPD na [obr. 19.5](#) zaznamenávají historii systému lakování automobilů v průběhu času. V diagramu pod OPD v levém horním rohu je zobrazeno pouze první časové období. **Lakování** není aktivní a **automobil** je **bílý**. Ve druhém diagramu jsou zobrazena první tři časová období. Ve třetím období je **lakování** aktivní a **automobil** již není **bílý**. Totéž se děje ve čtvrtém období, jak je znázorněno ve třetím diagramu. Nakonec v pátém období, znázorněném v dolním diagramu, **lakování** již není aktivní a **auto** je **červené**.



Raw Metal Bar can be **pre-cut** or **cut**.
Raw Metal Bar is initially **pre-cut** and finally **cut**.
Machining requires **Coolant**.
Machine Operator handles **Machining**.
Part can be **pre-tested** or **tested**.
Part is initially **pre- tested** and finally **tested**.

Cutting changes **Raw Metal Bar** from **pre-cut** to **cut**.
Machining consumes **Raw Metal Bar**.
Machining yields **pre-tested Part**.
Testing changes **Part** from **pre-tested** to **tested**.

Obr. 19.6 **Surová kovová tyč** a **součástka** jsou objekty, které mohou být v přechodu mezi stavy

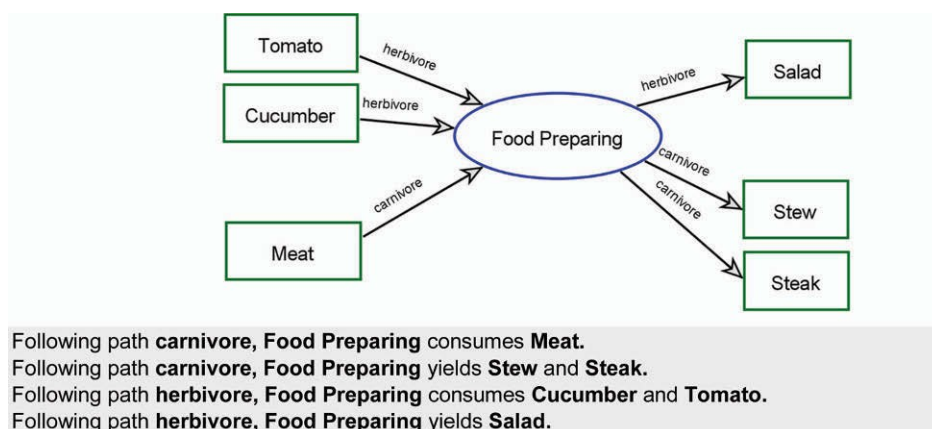
Jako další příklad lze uvést OPD na [obr. 19.6](#), kde proces řezání převádí surovou kovovou tyč z předřezaného stavu do řezaného stavu. Dokud je proces řezání aktivní, je stav surové kovové tyče v přechodu a vázán na proces řezání: proces řezání ji vyvede z předřezaného stavu, ale ještě ji nepřevédo do řezaného stavu s dokončením procesu. Během řezání je stav surové kovové tyče nestabilní, a proto neurčitý: může být

může být částečně řezaná a znovu použitelná nebo téměř celá řezaná a nepoužitelná. V obou případech není k dispozici pro **obrábění**, protože není v **řezaném** stavu. Stejně tak během **testování** není **součástka** ještě **předběžně otestovaná**, ale stále ještě není **otestovaná**.

Pokud se aktivní ovlivňující proces předčasně zastaví nebo trvá příliš dlouho, stav jakéhokoli ovlivněného subjektu zůstává neurčitý, pokud výjimka nevyřeší objekt do jednoho z jeho přípustných stavů. To lze provést pomocí odkazu na výjimku přesčasu nebo nedodržení času, který je popsán v kapitole o operační sémantice OPM.

19.5 Štítky cest a flip-flop

Když dva nebo více procedurálních odkazů vychází ze stejného procesu, není možné vědět, který odkaz sledovat, pokud odkazy nejsou označeny.



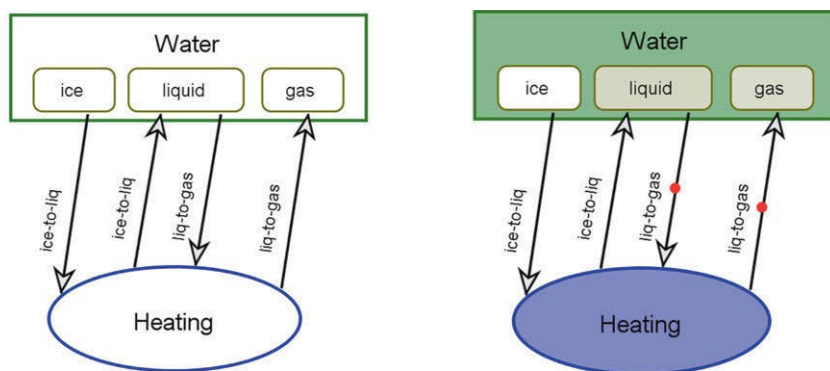
Obr. 19.7 Štítek cesty demonstrovaný na spojeních spotřeby a výsledku

Obrázek 19.7 ilustruje tento problém a použití štítků cest na spojeních spotřeby a výsledku, které tento problém řeší. Pokud existují **rajčata**, **okurky** a **maso**, výsledkem je výroba **salátu**, **guláše** a **steaku**. Nemůžeme však určit, které ingredience byly použity v jakém jídle. A co když chceme modelovat, že pro vegetariány chceme připravit pouze **salát** a pro masožravce pouze **guláš** a **steak**?

To se řeší pomocí štítků cesty **carnivore** a **herbivore**, zaznamenaných podél procedurálních odkazů, jak je znázorněno na **obr. 19.7** a vyjádřeno pomocí OPL. Štítky cesty jednoznačně určují, kterým odkazem se při opuštění procesu řídit: Odkazem, kterým se máme řídit, je ten, který má stejný štítek jako ten, kterým jsme do procesu vstoupili. Pomocí štítků cesty je možné v modelu sledovat konkrétní scénář, který zahrnuje více po sobě jdoucích procedurálních odkazů. Jak ukazuje tento příklad, popisky cest odstraňují logický požadavek AND z objektů v sadě objektů předzpracování. Zde musí existovat pouze všechny objekty v sadě objektů předzpracování, jejichž odkazy mají stejnou popisku, aby byla splněna předběžná podmínka. **Tomato** a **Cucumber** samostatně nebo **Meat** samostatně tedy splňují předběžnou podmínku pro **Food Preparing** a výsledek je určen popiskou cesty.

Štítek cesty je štítek na procedurálním odkazu, který určuje, že odkaz, který má být následován, je ten, který má stejný štítek jako ten, se kterým byl proces spuštěn.

Štítky cest odstraňují nejednoznačnost vyplývající z více odchozích procedurálních odkazů a lze je také použít pro odkazy specifikované stavem. Například na [obr. 19.8](#) jsou dva výstupní odkazy: jeden z **topení** do stavu **kapaliny vody** a druhý do stavu **plynu**. Při vstupu do tohoto procesu ze stavu **ledu** není jasné, zda by měl tok řízení směřovat do stavu **kapaliny** nebo do stavu **plynu**, pokud nepoužijeme štítky cest. Alternativou by bylo mít dva samostatné procesy, jeden s názvem „Ohřev ledu na kapalinu“ a druhý „Ohřev kapaliny na plyn“. Podobné řešení lze použít na [obr. 19.7](#). Bez označení cest musí mít každá dvojice příchozích a odchozích procedurálních odkazů svůj vlastní proces.



Water can be **ice**, **liquid**, or **gas**.

Following path **ice-to-liq**, **Heating** changes **Water** from **ice** to **liquid**.

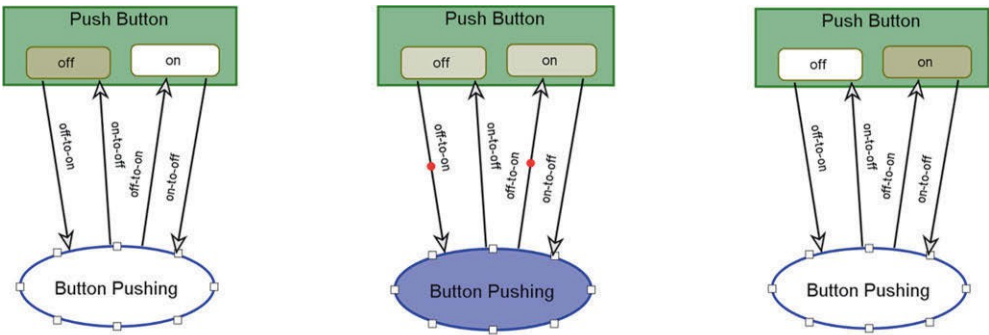
Following path **liq-to-gas**, **Heating** changes **Water** from **liquid** to **gas**.

Obr. 19.8 Označení cest demonstrováné na páru vstupních a výstupních spojů

Štítky cest poskytují *paměťový mechanismus*, který je vyžadován pro *stavové stroje*, kde další přechod do stavu závisí na stavu systému a na předchozím pohybu. Pokud předběžná podmínka procesu zahrnuje objekt nebo stav propojený prostřednictvím procedurálního propojení s označením cesty a sada objektů po zpracování má více než jednu možnost pro cílový objekt nebo stav, je vhodným cílem sady objektů po zpracování ten, který se získá po propojení se stejným označením cesty jako propojení spojující jeden nebo více objektů a/nebo stavů ze sady objektů před zpracováním.

Z pohledu metamodelu je **označení cesty** (volitelnou) vlastností **procedurálního odkazu**. Mechanismus paměti určuje, že pokud se scénář odehrál prostřednictvím cesty s určitým označením, musí pokračovat dalším krokem podle směru označeného stejným označením cesty.

[Obr. 19.9](#) představuje animovanou simulaci jednoduchého **tlačítka**, které po stisknutí **vypne zapnutou** lampu a **zapne** ji, pokud je **vypnutá**. **Tlačítko** si „pamatuje“ svůj stav, takže pokaždé, když je stisknuto, přepne se do jiného stavu. Stejný princip můžeme použít k modelování „**klopného obvodu**“, dvoustavového zařízení, které poskytuje základní paměť pro sekvenční logické operace a používá se pro digitální ukládání binárních numerických dat. Tento mechanismus modelu OPM lze také použít k dosažení logického operátoru „NOT“, jak je popsáno v oddíle [23.2](#).

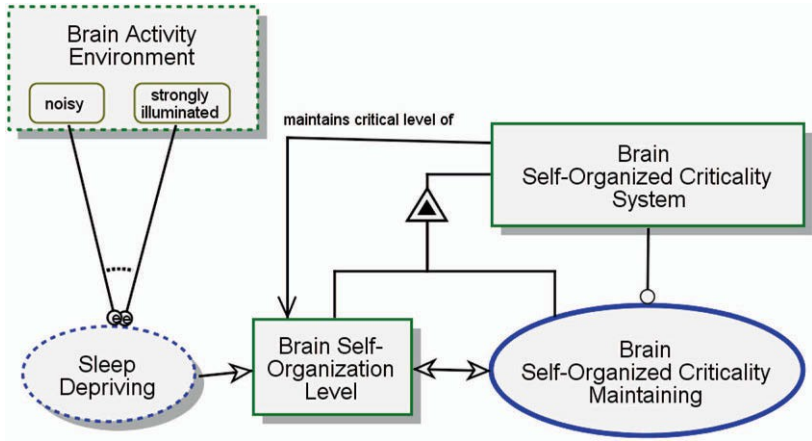


Push Button can be **off** or **on**.
Following path **off-to-on**, **Button Pushing** changes **Push Button** from **off** to **on**.
Following path **on-to-off**, **Button Pushing** changes **Push Button** from **on** to **off**.

Obr. 19.9 Jednoduché tlačítko s pamětí založené na mechanismu klopného obvodu

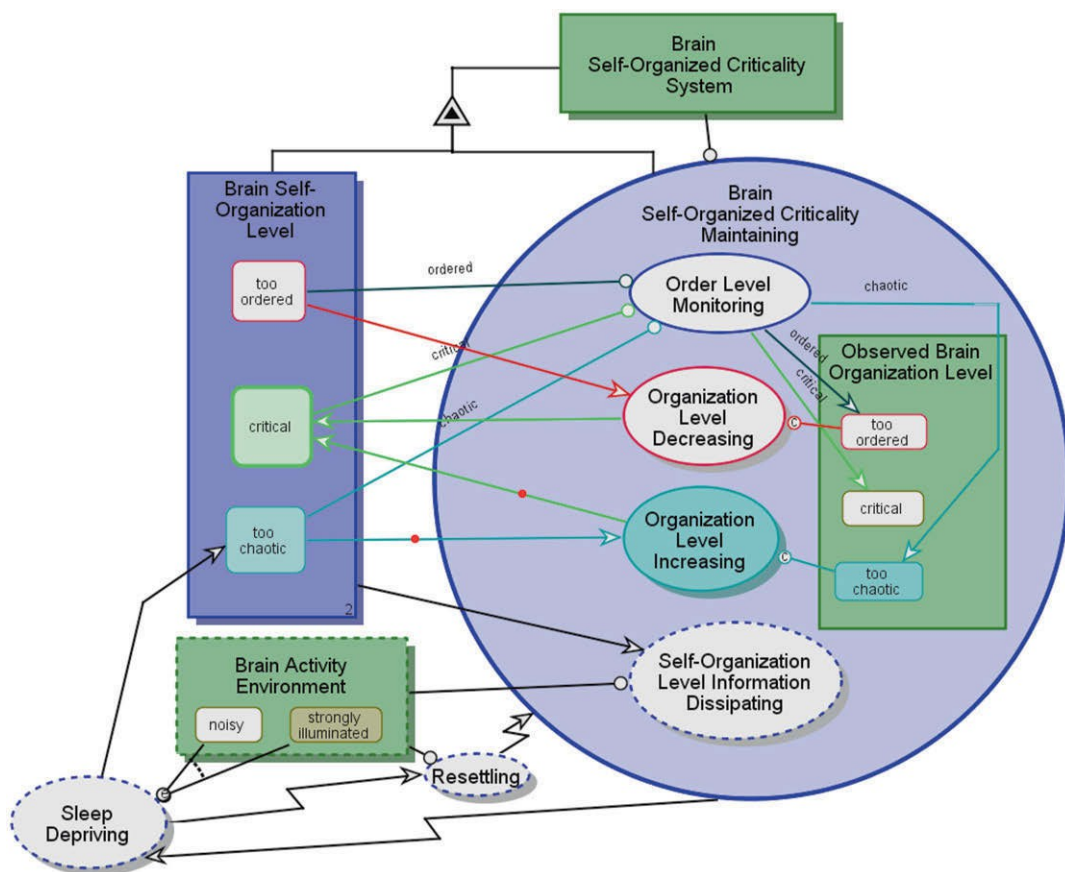
19.6 Model „samoorganizované kritičnosti“ mozku

Jak napsal Ouellette (2014) na základě Bak et al. (1987), v roce 1999 Bak prohlásil, že mozek možná funguje správně v kritickém stavu mezi přílišným řádem a úplným nepořádkem. Tato „samoorganizovaná kritičnost“ funguje na stejných základních principech jako jednoduchá hromada písku, ve které laviny různých velikostí pomáhají udržovat stabilitu celého systému. Například pokud je mozek zbaven spánku, jeho organizace se dostane mimo „optimální bod“, který se nachází na nebo blízko samoorganizované kritičnosti, a proto nemůže správně fungovat.



Obr. 19.10 Nejvyšší úroveň OPD (SD) systému samoorganizované kritičnosti mozku

Obrázek 19.10 představuje SD, nejvyšší úroveň OPD systému **samoorganizované kritičnosti mozku**. Ukazuje zpětnou vazbu: **hluché** nebo **silně osvětlené prostředí** mozkové aktivity vyvolává **nedostatek spánku**. Na **obrázku 19.11**, SD1, kde je zvětšena **samoorganizovaná kritičnost mozku**, vidíme animovanou simulaci modelu systému. Do tohoto zamrzlého okamžiku simulace došlo k tomu, že **úroveň sebeorganizace mozku**, která byla zpočátku v **kritickém** stavu, se změnila na **příliš chaotickou** a byl zahájen proces **udržování sebeorganizované kritičnosti mozku**, počínaje prvním dílčím procesem – **monitorováním úrovně řádu**. Vzhledem k tomu, že (podle **chaotické cesty**) je **úroveň sebeorganizace mozku příliš chaotická**, hodnota **pozorované úrovně organizace mozku** se změní na **příliš chaotickou**. Všimněte si, že **úroveň sebeorganizace mozku** je atributem procesu **udržování sebeorganizované kritičnosti mozku**, protože se jedná o objekt v rámci tohoto procesu, který je třeba udržovat v nominálním rozmezí hodnot nebo v určité toleranci.



Obr. 19.11 SD1 – Udržování sebeorganizované kritičnosti mozku – zvětšeno

Například **lidské tělo** musí udržovat teplotu v úzkém rozmezí 36,4–37,0 stupňů Celsia. Podobně klimatizační systém musí udržovat teplotu **vzduchu v místnosti** na 69–

72 stupňů Fahrenheita. Proces **udržování teploty** začíná dílčím procesem **monitorování teploty**, která může být **příliš nízká**, v **rozmězí** nebo **příliš vysoká**. Pokud je **naměřená teplota příliš nízká**, dojde k **ohřevu** (v **lidském těle** může být proveden **spalováním tělesného tuku**, v **klimatizačním systému zapálením pece**), čímž se změní **teplota lidského těla** nebo **vzduchu v místnosti z příliš nízké na teplotu v rozmězí**. Pokud je **naměřená teplota v rozmězí**, není třeba provádět žádné nápravné opatření, takže nedochází k žádnému dílčímu procesu, a pokud je **naměřená teplota příliš vysoká**, dochází k **ochlazování** (**pocením** nebo **kondenzací**), čímž se opět obnoví hodnota atributu **úrovně teploty v rozmězí**. Po uplynutí určité doby má **prostředí**, ať už je **horké** nebo **studené**, tendenci posunout hodnotu **teploty** z požadovaného rozsahu na **příliš vysokou**, respektive **příliš nízkou**, a dochází k novému cyklu **udržování teploty**.

Podobnost mezi udržováním teploty v biologickém systému a v umělém systému ukazuje, že složité systémy, ať už biologické nebo umělé, podléhají stejnému základnímu principu zpětné vazby, který spočívá v přijetí nápravných opatření, jsou-li nezbytná k udržení požadované hodnoty atributu. Společným prvkem obou systémů je „kyberfyziký“ kontrolní mechanismus, který zahrnuje *fyzické* i *informatické* („kybernetické“) objekty a procesy: Zatímco **teplota** je fyzikální vlastnost, proces **monitorování teploty** využívá fyzikální prostředky (**teploměr** nebo **kožní nervové buňky**) k vytvoření informatického objektu – hodnoty **pozorované teploty**. Tato hodnota je přenášena do kontrolního mechanismu – „mozku“ systému – prostřednictvím komunikačního kanálu. Může se jednat o elektronický signál přenášený drátem v klimatizačním systému nebo elektrický signál z kůže do mozku v lidském těle. Na základě hodnoty signálu pak kontrolní mechanismus určí, jaká fyzická akce (**vytápění** nebo **chlazení**) musí být provedena, čímž se uzavře zpětnovazební smyčka.

Všechny kyberfyziké systémy mají společnou tuto interakci mezi informatickými objekty, které označují stav světa, jak jej vnímá systém, a systémovými fyzikálními procesy, které působí proti nepříznivému vlivu prostředí na požadovanou hodnotu atributu, který má být udržován. Jak ukazuje model OPM, objekt **Observed Brain Organizational Level** (pozorovaná úroveň organizace mozku) **Obr. 19.11** (který je analogický s informačním objektem **Observed Temperature (pozorovaná teplota)**) je informační – nemá stín. Ačkoli jsou implementace v živých a neživých objektech odlišné, základní kyberfyziký mechanismus je velmi podobný, ne-li identický.

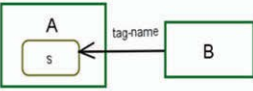
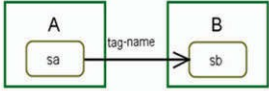
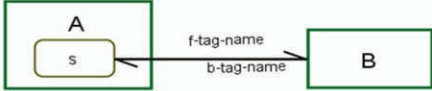
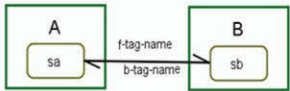
19.7 Stavově specifikované štítkované strukturální odkazy

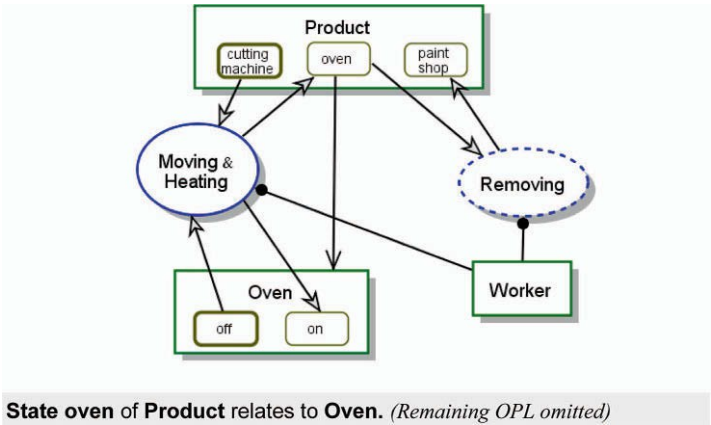
Stavově specifikované štítkované strukturální odkazy umožňují asociovat stav jednoho objektu s jiným objektem nebo se stavem jiného objektu. Odkaz asociace stavu a objektu umožňuje asociovat stav nějakého objektu s jiným objektem v modelu. Vezměme si například OPD na **obr. 19.12**, kde **Oven** je jak objektem v systému, tak stavem **Product** – možným umístěním **Product**. To je graficky vyjádřeno otevřenou šipkou – stejným symbolem, který se používá pro jednosměrné strukturální propojení s nulovým tagem, od stavu **oven** objektu **Product** k objektu **Oven**. Rozdíl mezi jednosměrným strukturálním odkazem a strukturálním odkazem specifikovaným stavem je v tom, že zatímco první z nich spojuje objekt s objektem, druhý spojuje stav s objektem nebo se stavem jiného objektu. Strukturální odkazy specifikované stavem jsou uvedeny v **tabulce 19.1**.

Jako další příklad lze uvést systém **Check-Based Paying** (platba šekem) popsany na **obr. 19.13**, kde **šek** má atribut nazvaný **Keeper** (držitel), který popisuje entitu, která drží šek v daném okamžiku během jeho zpracování. **Keeper** má tři stavy: **payer (plátce)**, **payee (příjemce)** a **bank (banka)**. Tyto tři stavy jsou spojeny s

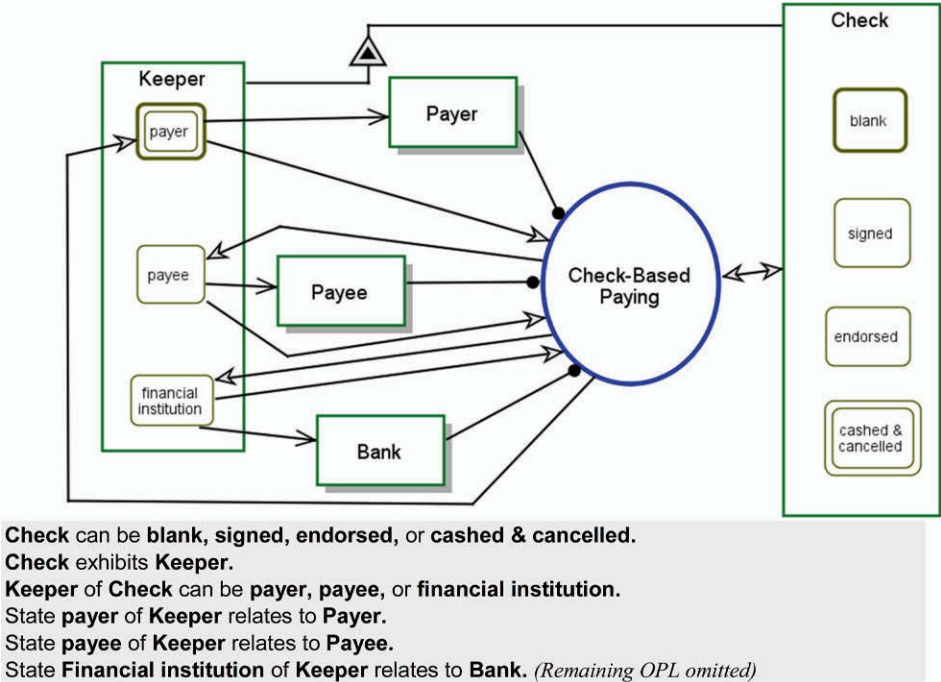
se třemi objekty **Plátce**, **Příjemce** platby a **Banka**. V OPD je to označeno třemi odpovídajícími odkazy ze stavu na přidružený objekt.

Tabulka 19.1 Shrnutí strukturálních vztahů a odkazů specifikovaných stavem

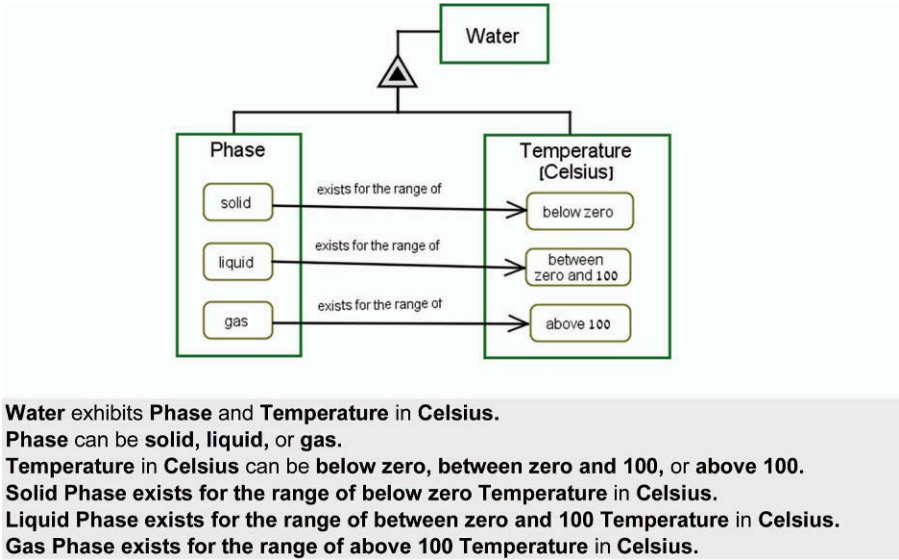
Source/ Destination Directionality	source state-specified	destination state-specified	source-and-destination state-specified
unidirectional	 S A tag-name B.	 B tag-name s A.	 Sa A tag-name sb B.
bidirectional	 S A f-tag-name B. B b-tag-name s A.	 Sa A f-tag-name sb B. Sb B b-tag-name sa A.	
reciprocal	 B and s A are recip-tag-name.	 Sa A and sb B are recip-tag-name.	



Obr. 19.12 Příklad asociace stavů s objekty prostřednictvím asociace stav-objekt: Trouba je jak objektem v systému, tak stavem produktu



Obr. 19.13 Přřazení hodnot atributů k objektům prostřednictvím strukturálního odkazu specifikovaného stavem



Obr. 19.14 Zdrojový a cílový stavově specifikovaný štítkovaný strukturální odkaz

Jako příklad odkazu specifikovaného stavem zdroje a cíle je na [obr. 19.14](#) každá ze tří hodnot **fáze vody** spojena s odpovídajícím rozsahem hodnot **teploty** prostřednictvím tří odkazů specifikovaných stavem zdroje a cíle, jejichž značka je „**existuje pro rozsah**“.

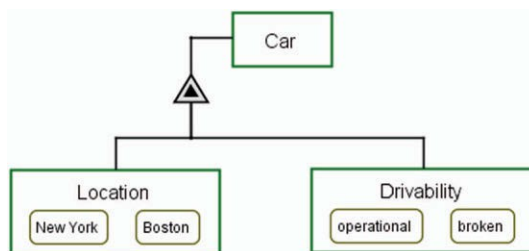
19.8 Složené stavy a stavový prostor

Příklady, které jsme dosud viděli, jsou stavy, které jsou *atomické*, tj. stavy, které nejsou kombinací jiných stavů. Zatímco většina stavů v modelech OPM je atomická, mohou existovat i *složené* stavy.

Atomický stav je stav, který není kombinací jiných stavů.

Složený stav je stav, který kombinuje alespoň dva jiné stavy.

Jako příklad lze uvést jeden atribut **automobilu** na [obr. 19.15](#), kterým je **poloha** s hodnotami **New York** a **Boston**. Dalším atributem je **pojízdnost** automobilu (jeho schopnost pohybovat se po silnici) s hodnotami **provoznuschopný** a **nepojízdný**.



Obr. 19.15 Stavový prostor Poloha-Jezdnost automobilu

Vzhledem k tomu, že **Poloha** a **Řiditelnost** jsou ortogonální (nezávislé na sobě), může být **Auto** v jednom ze čtyř *složených* stavů: **funkční v New Yorku**, **funkční v Bostonu**, **nefunkční v New Yorku** a **nefunkční v Bostonu**. 2×2 kartézský součin hodnot **Poloha** a **Řiditelnost** tvoří *stavový prostor Auto*.

Stavový prostor objektu je kartézský součin množin stavů všech atributů a částí objektu.

Přejdeme k složitějšímu příkladu. **Letiště** na [obr. 19.16](#) má čtyři atributy, každý se dvěma stavy: **povětrnostní podmínky**, **služby věže** a **radarové pokrytí**, stejně jako **znalost pilota** se třemi stavy. Kartézský součin množin stavů každého atributu vyjmenovává stavový prostor objektu. Pro náš příklad **letiště** je tento kartézský součin následující: $Weather \times Tower\ Service \times Radar\ Coverage \times Pilot\ Familiarity = \{fair, hazardous\} \times \{available, unavailable\} \times \{nonexistent, existent\} \times \{poor, fair, excellent\}$. Existují tedy $2 \times 2 \times 2 \times 3 = 24$ stavů **letiště**.

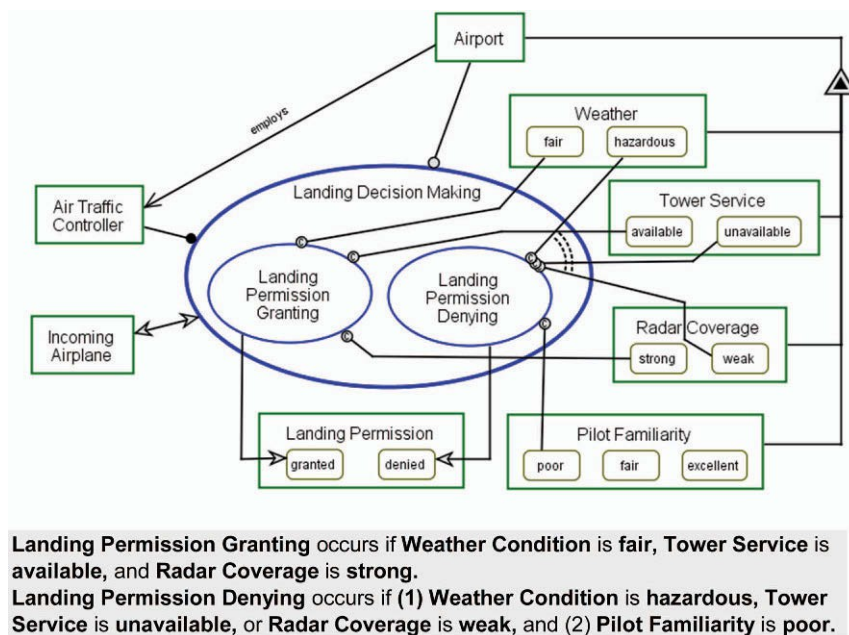
Obecně platí, že pokud má objekt n atributů (včetně implicitního, pokud existuje) a stavových částí, z nichž každá má v_i hodnot, pak velikost stavového prostoru je:

$$S = \prod_{i=1}^n v_i$$

V tomto kontextu lze každý atribut také označit jako *dimenzi*, analogicky k tomu, jak vektory mohou sloužit jako dimenze (např. tři ortogonální vektory nazvané X, Y a Z, které pokrývají trojrozměrný kartézský bodový prostor). Například jeden z 24 stavů **letišť**, získaný výčtem prvního stavu v každém ze čtyř atributů **letišť** výše, je:

(Počasí= dobré, služba věže= dostupná, radarové pokrytí= silné, znalost pilota= špatná).

Každý takový bod může být předpokladem pro nějaký proces. **Obrázek 19.16** ukazuje předpoklady pro dva alternativní dílčí procesy **rozhodování o přistání**, které provádí **letový dispečer: udělení povolení k přistání a zamítnutí povolení k přistání**. Předpoklady pro tyto dva procesy jsou vyjádřeny ve větách OPL.



Obr. 19.16 Příklady složených stavů (bodů ve stavovém prostoru) letišť požadovaných pro udělení povolení k přistání a zamítnutí povolení k přistání

Tento příklad zahrnuje použití podmíněných odkazů, které jsou popsány v kapitole o operační sémantice OPM, a logických operátorů OR a XOR (X-OR), které jsou popsány v jiné samostatné kapitole věnované těmto operacím.

19.8.1 Věta OPL s více podmínkovými klauzulemi

Druhá věta OPL na **obr. 19.16**, která je větou OPL s více podmínkami, ukazuje, jak lze v modelu OPM jednoznačně vyjádřit složité podmínky. Věta OPL s více podmínkami se skládá z $n > 1$ (dvou nebo více) klauzulí X-OR. Každá klauzule X-OR vyjadřuje podmínku OR nebo XOR a začíná číslem klauzule v závorce „(i)“, kde $i = 1, 2, \dots, n$. Číslo klauzulí umožňují OPL vytvořit libovolný počet skupin podmínek OR nebo XOR, které jsou všechny spojeny logickým AND.

19.8.2 Použití procesů k určení složených stavů

Procesy lze použít k určení složených stavů. Na [obr. 19.17 stolní lampa](#), která může mít složené stavy **zhasnutá** a **rozsvícená**, se skládá ze tří částí: **vypínače**, **napájecí zástrčky** a **žárovky**, z nichž každá má dva stavy.

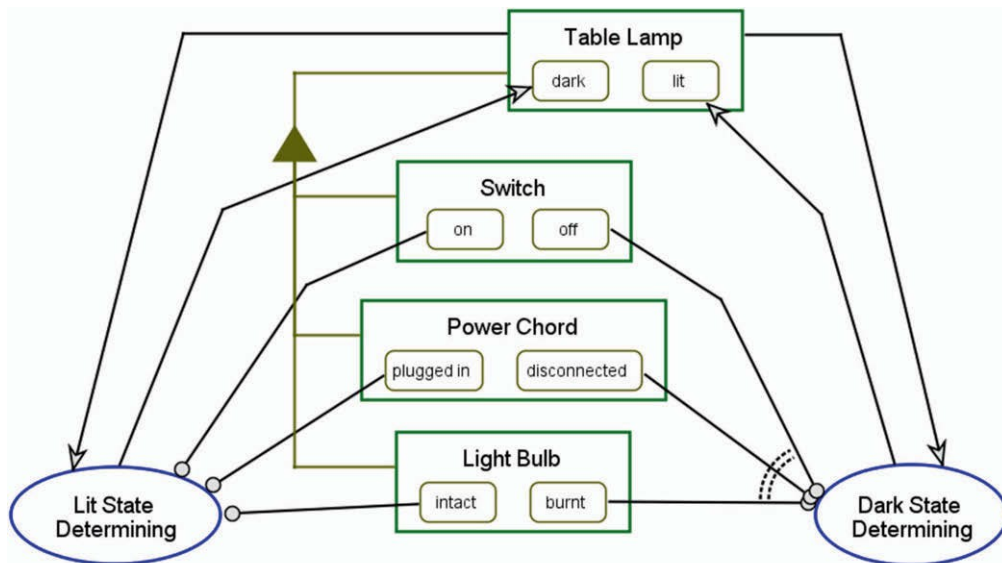


Table Lamp can be **dark** or **lit**.

Table Lamp consists of **Switch**, **Power Chord**, and **Light Bulb**.

Switch can be **off** or **on**.

Power Chord can be **plugged in** or **disconnected**.

Light Bulb can be **intact** or **burnt**.

Lit State Determining requires **on** **Switch**, **plugged in** **Power Chord**, and **intact** **Light Bulb**.

Dark State Determining changes **Table Lamp** to **dark** **Table Lamp**.

Lit State Determining changes **Table Lamp** to **lit** **Table Lamp**.

Dark State Determining requires **burnt** **Light Bulb**, **disconnected** **Power Chord**, or **off** **Switch**.

Lit State Determining yields **lit** **Table Lamp**.

Obr. 19.17 Určení složených stavů „vypnutá“ a „zapnutá“ stolní lampy pomocí procesů

Některé body ve stavovém prostoru objektu nejsou proveditelné, například: (stolní lampa= zhasnutá, vypínač = zapnutý, napájecí kabel= zapojený, žárovka= neporušená). Procesy určují, které body v prostoru stavů objektu jsou proveditelné. Pro dvě dimenze lze toto také znázornit v tabulce, případně jako dvourozměrné pole uvnitř přibližného objektu. Tabulka však nevyjadřuje důvody proveditelnosti nebo neproveditelnosti jednotlivých bodů.

19.9 Shrnutí

- **Stav** je situace nebo pozice, ve které může objekt existovat, nebo hodnota, kterou může atribut nabývat, po určitou dobu během své existence.
- **Počáteční stav** objektu B je stav, ve kterém se B nachází při svém vytvoření nebo při spuštění systému.
- **Konečný stav** objektu B je stav, ze kterého B nemůže vystoupit.
- **Výchozí stav** objektu B je stav, který se od objektu B očekává, pokud není jeho stav specifikován.
- **Hodnota** je stav atributu, proto je specializací stavu.
- Kromě toho, že se objekt nachází v určitém stavu, může být také nestabilní, když se nachází v *přechodu* mezi dvěma stavy – vstupním stavem a výstupním stavem.
- **Přechod stavu** je nestabilní časové období pro objekt, ke kterému dochází, když na něj působí proces, který mění jeho stav.
- **Diagram životnosti** je diagram, který pro jakýkoli okamžik během životnosti systému ukazuje, jaké objekty v systému existují, v jakém stavu se každý objekt nachází a jaké procesy jsou aktivní.
- **Stavově specifikovaný značkový strukturální odkaz** je značkový strukturální odkaz, který spojuje stav objektu s jiným objektem nebo se stavem jiného objektu.
- **Atomický stav** je stav, který není kombinací jiných stavů.
- **Složený stav** je stav, který kombinuje alespoň dva jiné stavy.
- **Stavový prostor** objektu je kartézský součin množin stavů všech atributů a částí objektu.

19.10 Problémy

1. Auto lze řídit, pokud má palivo, nabitou baterii a jsou k dispozici klíče od auta. Nakreslete OPD s atributy a stavy, které specifikují tyto podmínky.
2. Vyjmenujte stavový prostor automobilu z předchozí otázky: (1) jako seznam, (2) v tabulce.
3. Nakreslete OPD a napište OPL věty tří objektů, které mají počáteční a konečné stavy.
4. Nakreslete OPD a napište OPL věty tří objektů, které mají výchozí stav.
5. **Komplex automobil-řidič** vyžaduje nejen stavy automobilu uvedené v úloze 1, ale také střízlivého, bdělého řidiče s řidičským průkazem.
6. Zahrňte tyto požadavky do OPD komplexu automobil-řidič.
7. Napište odstavec OPL tohoto OPD.
8. Navrhněte alternativní způsob zobrazení OPD z (a).

9. Uspořádaná sada hodnot **velikosti** nějakého objektu může být **miniaturní, malá, střední, velká, extra velká, obrovská a kolosální**. Alternativně lze použít řadu čísel nějaké specifikované měrné jednotky (např. metry pro délku nebo kilogramy pro hmotnost), které udávají přesnější a „objektivnější“ specifikaci stejného atributu **velikosti**. Vyberte objekt se dvěma sadami atributů, jednou kvalitativní a druhou kvantitativní. Použijte textové hodnoty **velikosti** uvedené výše a přiřadte je v OPD k číselným rozsahům podle vašeho výběru.

Kapitola 20

Zobecnění a instancování

Jak se tento termín nejčastěji používá, generalizace je „všeobecné“ tvrzení, že všechny objekty určitého obecného druhu mají určitou vlastnost.

Lowe (1983)

Při diskusi o agregaci a exhibici jsme hovořili o celých skupinách objektů nebo procesů – jakékoli vědecké práci, jakémkoli zaměstnanci, jakémkoli běhu. Co však v případě, že bychom chtěli uvažovat o příkladu konkrétní práce, kterou napsal jistý John Doe? Nebo pokud bychom chtěli uvažovat o skupině zaměstnanců, konkrétně manažerech, kteří pobírají určitý plat mimo rozsah platů dostupných pro společnost? Možná bychom chtěli diskutovat o běhu v maratonu, na rozdíl od jakéhokoli jiného druhu běhu? Musíme být schopni věnovat zvláštní pozornost specializované skupině, která patří do obecnější skupiny, nebo dokonce konkrétnímu příkladu z třídy objektů. Jak jasně naznačuje název, generalizace-specializace je vztah mezi obecným a zvláštním případem věci. Klasifikace-instancování je vztah mezi třídou věcí a jedinečnou instancí z této třídy. Vzhledem k tomu, že tyto dva pojmy jsou důležité pro modelování systémů, považujeme je za dva ze čtyř základních vztahů; a protože jsou úzce propojeny, jsou v této kapitole diskutovány a vysvětlovány společně.

20.1 Generalizace-specializace: Úvod

Nejprve se podívejme na několik jednoduchých příkladů, které nám pomohou připravit půdu pro diskusi o generalizaci-specializaci, neboli „gen-spec“.¹



Man and Women are Persons.

Digital Camera and Analog Camera are Cameras.

Obr. 20.1 Příklady generalizace-specializace

Osoba v levé části OPD na [obr. 20.1](#) je obecný případ, zatímco **muž** a **žena** jsou jeho zvláštní případy. Dalšími příklady jsou „**Pes** a **kočka** jsou **domácí zvířata**“, „**Pascal**, **Java** a **C++** jsou **programovací jazyky**“.

¹ Zkratka „gen-spec“ je převzata z Coad a Yourdon (1991).

„Letadlo a auto jsou vozidla.“ „Létání a plavba jsou dopravní prostředky.“ „Kečup a hořčice jsou koření.“

Generalizace-specializace je základní strukturální vztah mezi obecnou věcí G a jednou nebo více věcmi $S_1, S_2, \dots, S_n$, které jsou specializacemi G .

Alternativním způsobem vyjádření věty OPL mohlo být „**Digitální fotoaparát a analogový fotoaparát** specializují **fotoaparáty**“. V souladu s principem zachování co nejpřirozenějšího a nejjednoduššího jazyka OPL však OPL používá jasnější a intuitivnější vyhrazené fráze „je“ (nebo „jsou“ pro množné číslo) namísto „specializuje“ nebo „specializují“ pro označení vztahu gen-spec z opačného, nebo zdola nahoru směru, od specializované věci – specializace – k generalizující věci – obecné. Je možné libovolné množství specializací. Následující příklad uvádí tři specializované objekty.

Okurka je zelenina. Rajče je zelenina.
Mrkev je zelenina.

Spojíme tři výše uvedené specializační věty do jedné:

Okurka, rajče a mrkev jsou zelenina.

Generalizace-specializace je přechodný vztah, což znamená, že pokud **A** je **B** a **B** je **C**, pak **A** je **C**. Konkrétněji, zvažte následující dvě specializační věty:

Rajče je zelenina. Zelenina je rostlina.

Jelikož generalizace-specializace je přechodná, můžeme odvodit, že:

Rajče je rostlina.

Zobecnění-specializace znamená, že zjemnitelný, obecný pojem zobecňuje dva nebo více **zjemněných pojmů**, které jsou specializacemi obecného pojmu. Vztah generalizace-specializace spojuje jednu nebo více specializací se stejnou vytrvalostí jako obecné, takže jak obecné, tak všechny jeho specializace jsou objekty (v termínech metamodelu, pokud je **vytrvalost věci trvalá**) nebo obecné a všechny jeho specializace jsou procesy (pokud je **vytrvalost věci přechodná**).

Graficky je vztah generalizace-specializace znázorněn prázdným trojúhelníkem, jehož vrchol je spojen čarou s obecným a specializace jsou spojeny čarami s opačnou základnou.

UML a SysML používají k označení generalizace-specializace bílý (prázdný) trojúhelník (stejně jako v OPM), ale v UML a SysML je vrchol trojúhelníku přímo spojen s generalizujícím objektem a základna bílého trojúhelníku není nutně vodorovná, ale spíše kolmá k čáře spojující specializaci. Navíc, podobně jako v případě agregace, protože v UML neexistuje rozvětvení, musí mít každá specializace v diagramu třídy UML a diagramu definice bloku SysML svůj vlastní symbol. Vzhledem k tomu, že UML a SysML nemají procesy v diagramu třídy, vztah agregace a specializace v UML a SysML se vztahuje pouze na objekty.

20.1.1 Specializace procesů

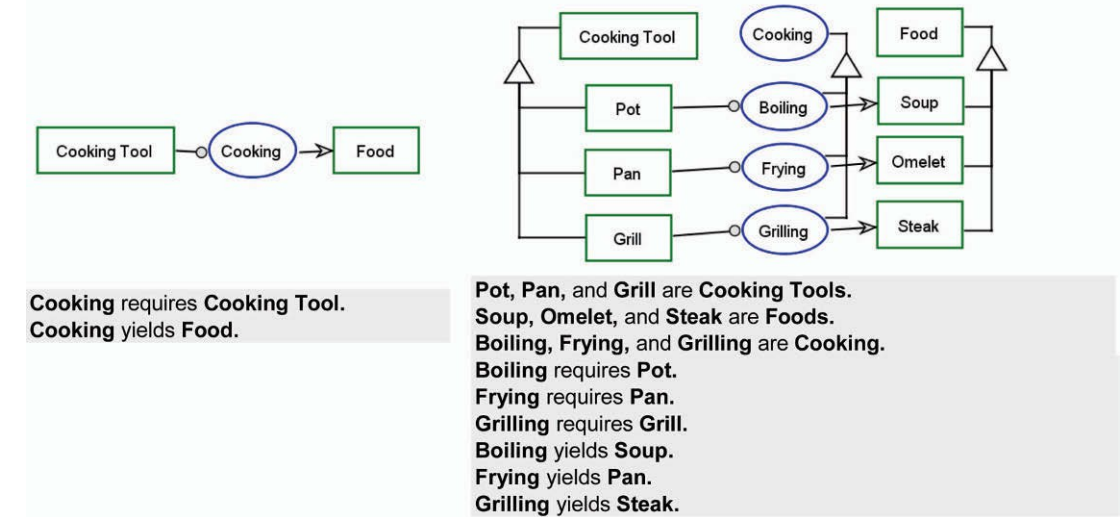
Zobecnění a specializace se netýkají pouze objektů. Stejný vztah platí i pro procesy. [Obrázek 20.2](#) ukazuje dva jednoduché příklady.



Obr. 20.2 Specializace procesů v jednotném a množném čísle

Aby byla zachována anglická gramatika, věta o specializaci procesu se mírně liší od věty o specializaci (objektu) v tom, že (1) místo vyhrazené fráze „is a“ se používá vyhrazené slovo „is“ a (2) zatímco zobecňující objekt je v množném čísle, jako v případě **Vegetables**, ve větě o specializaci více procesů je v jednotném čísle, jako v případě **Cooking**. Zvažte následující věty OPL.

Boiling is Cooking.
Frying is Cooking.
Grilling is Cooking.
The three OPL sentences above become:
Boiling, Frying, and Grilling are Cooking.



Obr. 20.3 Vlevo: Obecný vzor vaření. Vpravo: Specializace **vaření**, **vaření** a jídla a specializované odkazy mezi těmito specializacemi

Specializace objektů a procesů lze kombinovat, aby se specifikovaly specializované procedurální vazby mezi specializacemi objektů a procesů. **Obrázek 20.3** ukazuje vlevo vzor **vaření**, který používá **nástroj na vaření** jako nástroj a produkuje **jídlo**. Vpravo jsou tři specializace **nástroje na vaření**, **vaření** a **jídlo**. Každá specializace **nástroje na vaření** je nástrojem specializace **vaření**, která produkuje specializaci **jídla**.

20.1.2 Podspecifikace a nadspecifikace vazeb

K podspecifikaci odkazu by došlo, kdybychom na pravé straně OPD na [obr. 20.3](#) ponechali dva odkazy jako v OPD na levé straně a nspecifikovali šest procedurálních odkazů na pravé straně. To by znamenalo, že jakýkoli nástroj lze použít pro jakékoli vaření. K nadměrné specifikaci odkazu by došlo, kdybychom kromě šesti procedurálních odkazů v OPD na pravé straně přidali dva odkazy jako v OPD na levé straně. Obojího je třeba se vyvarovat. V případě nedostatečné specifikace ponechání jediného nástrojového propojení z **Cooking Tool (Nástroj na vaření)** do **Cooking (Vaření)** vpravo znamená, že jakýkoli **nástroj na vaření** by mohl být považován za nástroj jakéhokoli procesu **vaření** a k přípravě jakéhokoli **jídla**. Na druhé straně v případě nadměrné specifikace se dvě zobecňující propojení, ponechaná spolu se šesti specializovanými propojeními, stávají nadbytečnými. Nedostatečná a nadměrná specifikace se může vyskytnout také u strukturálních propojení.

20.2 Dědičnost

Nejvýraznější a nejbezprostřednější výhodou použití vztahu gen-spec je dědičnost, kterou vyvolává.

Dědičnost je přiřazení prvků OPM – věcí a odkazů – obecného prvku k jeho specializacím.

V OO designu znamená dědičnost, že atributy a do jisté míry také operace generalizujícího objektu jsou zděděny specializovanými objekty. V OPM je účinek dědičnosti silnější, protože kromě dědičnosti vlastností a částí zahrnuje také dědičnost strukturálních a procedurálních vazeb, stejně jako stavů. Prostřednictvím vztahu generalizace-specializace každá specializace zdědí od obecného každý z následujících čtyř druhů dědičných prvků:

- všechny **části** obecného objektu z jeho agregačního-participačního propojení,
- všechny **vlastnosti** obecného objektu z jeho vazby výstavy-charakterizace,
- všechny **označené strukturální vazby**, ke kterým se obecný objekt připojuje, a
- všechny **procedurální vazby**, ke kterým se obecně spojuje.

OPM poskytuje možnost vícenásobného dědičnosti tím, že umožňuje věci zdědit od více než jednoho obecného prvku každý z refines – čtyři dědičné prvky (části, vlastnosti, označené strukturální vazby a procedurální vazby), které existují pro tento obecný prvek.

Modelář může přepsat kteroukoli část obecného pojmu, která je ve výchozím nastavení zděděna specializací, tím, že pro každého účastníka zděděného z obecného pojmu určí specializaci tohoto účastníka s jiným názvem a jinou sadou stavů.

20.2.1 Vytvoření obecného prvku z kandidátských specializací

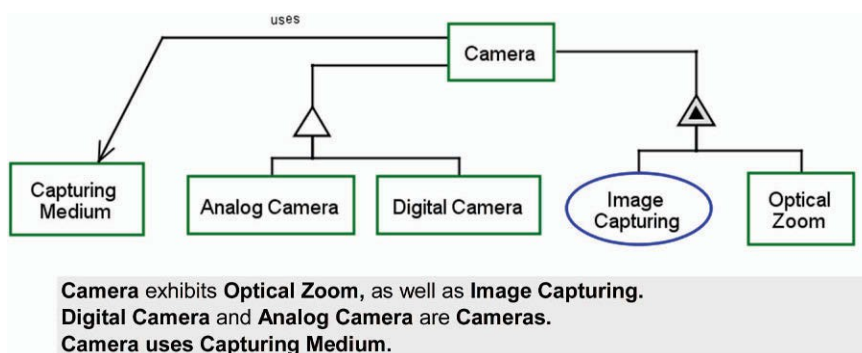
Chcete-li vytvořit obecný prvek z jedné nebo více kandidátských specializací, zděditelné prvky společné pro všechny kandidáty se „přesunou“ nahoru do obecného prvku. Manipulace se zděditelnými prvky se provádí následovně:

- Spojte všechny společné vlastnosti a společné účastníky specializací do jedné nově vytvořené obecné;

- Propojte nový obecný prvek pomocí odkazu generalizace-specializace se specializacemi;
- Odstraňte ze specializací všechny společné rysy a společné části, které specializace nyní dědí od nové obecné věci; a
- Přesuňte všechny společné označené strukturální odkazy a všechny společné procedurální odkazy, které spojují věc T s každou ze specializací, ze specializací do obecné věci, tak aby existoval jediný odkaz z T do obecné věci.

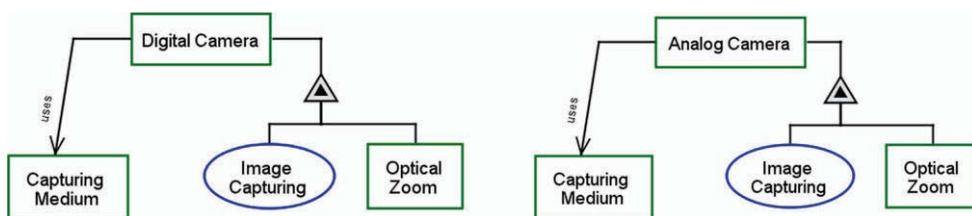
20.2.2 Dědičnost vlastností

Obecná věc dědí své vlastnosti – atributy a operace – do každé ze svých specializací. Například [obr. 20.4](#) je OPD **fotoaparátu**, který má dvě vlastnosti: atribut **Optický zoom** a operaci **Zachycení obrazu**. Tento OPD má následující odpovídající odstavec OPL, kde poslední věta OPL vyjadřuje jednosměrný označený strukturální vztah.



Obr. 20.4 Levá **kamera** a její specializace **Analogová kamera** a **Digitální kamera**

Jelikož **digitální fotoaparát** a **analogový fotoaparát** jsou specializacemi **fotoaparátu**, můžeme **fotoaparát** nahradit jeho specializacemi **digitální fotoaparát** a **analogový fotoaparát**. To bylo skutečně provedeno na [obr. 20.5](#), který demonstruje základní sémantiku dědičnosti: specializace – refinee – dědí vlastnosti (atributy a operace) od obecného – refineable.



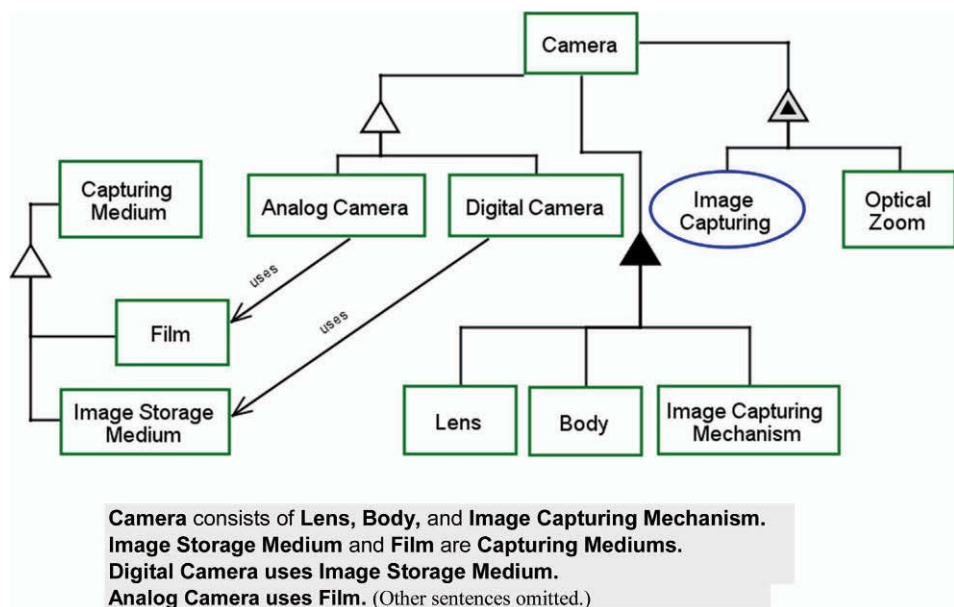
Obr. 20.5 Digitální fotoaparát a analogový fotoaparát jsou specializacemi fotoaparátu, proto každý z nich může nahradit fotoaparát z [obr. 20.4](#)

V OPM se dědí nejen vlastnosti, ale také odkazy a stavy. Dědic proto může nahradit předka. **Digitální fotoaparát** a **analogový fotoaparát** dědí nejen vlastnosti **fotoaparátu**, kterými jsou atribut **Optický zoom** a operace **Zachycení obrazu**, ale také dědí značku

strukturální vztahy **se používají** od **kamery** po **záznamové médium**. Navíc se dědí nejen strukturální vztahy, ale také procedurální vztahy. Dědic však může mít více funkcí, odkazů nebo stavů.

20.2.3 Dědičnost strukturálních vztahů

Zvažme OPD na [obr. 20.6](#), ve kterém specifikujeme části **kamery** a specializace **zachycujícího média**.



Obr. 20.6 Části, specializace a vlastnosti **kamery** jsou specifikovány spolu se specializacemi **zachycujícího média**

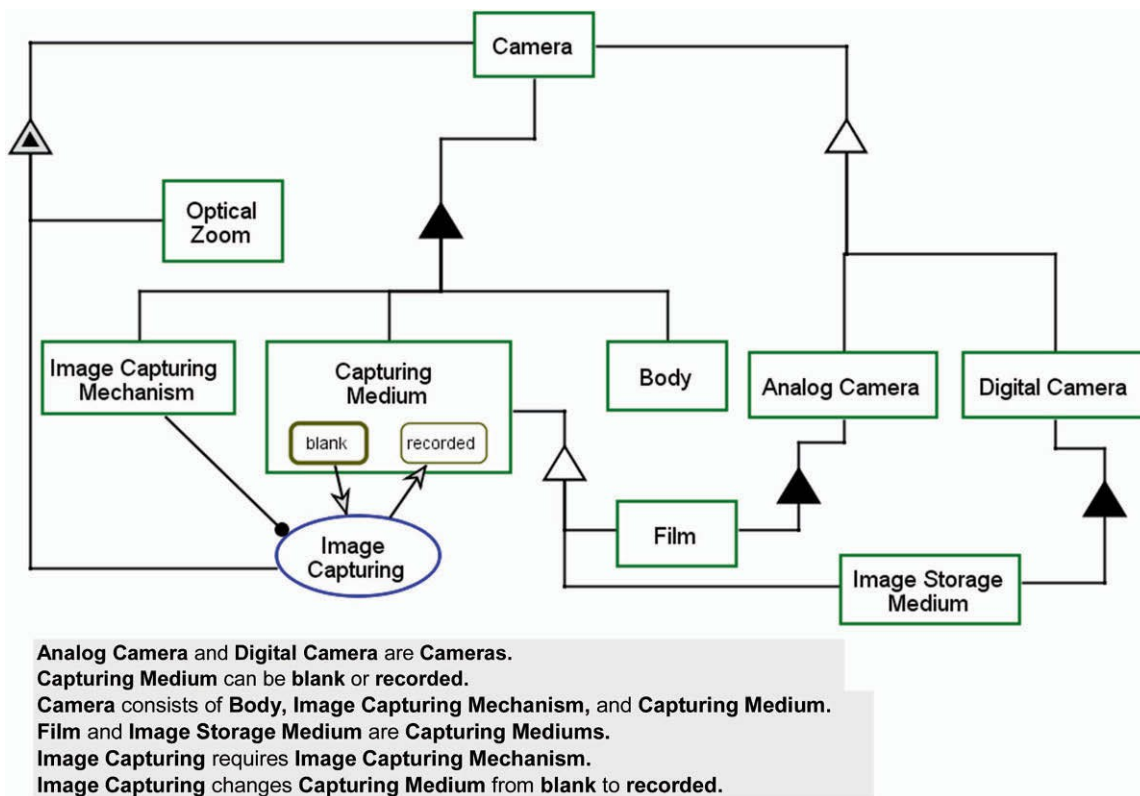
To znamená, že části, z nichž se skládá **kamera**, jsou zděděny dvěma specializacemi **kamery**:

Digitální fotoaparát se skládá z **objektivu, těla a mechanismu pro zachycení obrazu**. **Analogový fotoaparát** se skládá z **objektivu, těla a mechanismu pro zachycení obrazu**.

Dědí se nejen agregace. Dědí se také všechny označené strukturální vztahy, jako například **použití**. Jelikož označený vztah **používá** odkazy **Fotoaparát** na **Zachycující médium**, když specifikujeme specializace **Fotoaparátu** i **Zachycujícího média**, aniž bychom se zabývali strukturálním vztahem **použití**, zavádíme nedostatečnou specifikaci odkazu. Tato nedostatečná specifikace, se kterou jsme se setkali dříve, vyplývá ze skutečnosti, že strukturální vztah **použití** z **fotoaparátu** k **záznamovému médiumu** nespecifikuje, která specializace **fotoaparátu** (analogový fotoaparát nebo digitální fotoaparát) používá kterou specializaci **záznamového média** (médium pro ukládání obrazu nebo film). Abychom to vyjasnili, specifikujeme, která specializace **fotoaparátu** používá kterou specializaci **záznamového média**.

20.2.4 Dědičnost stavů a odkazů

V OPM se dědí také stavy a odkazy. Před procesem **snímání obrazu** v příkladu s **fotoaparátem** je **médium pro snímání**, které **fotoaparát používá**, **prázdné**. Po provedení procesu **snímání obrazu** se **médium pro snímání zaznamená**. **Prázdné** a **zaznamenané** jsou tedy dva stavy **média pro snímání**. OPD na [obr. 20.7](#) má dvě generalizační vazby, jednu pro **kameru** a druhou pro **médium pro zachycení**. Tyto dvě relace indukují dvě OPD na [obr. 20.8](#).



Obr. 20.7 Dědičnost stavů: **Film** a **obrazové paměťové médium** dědí stavy a vstupní a výstupní propojení s **obrazovým snímačem**.

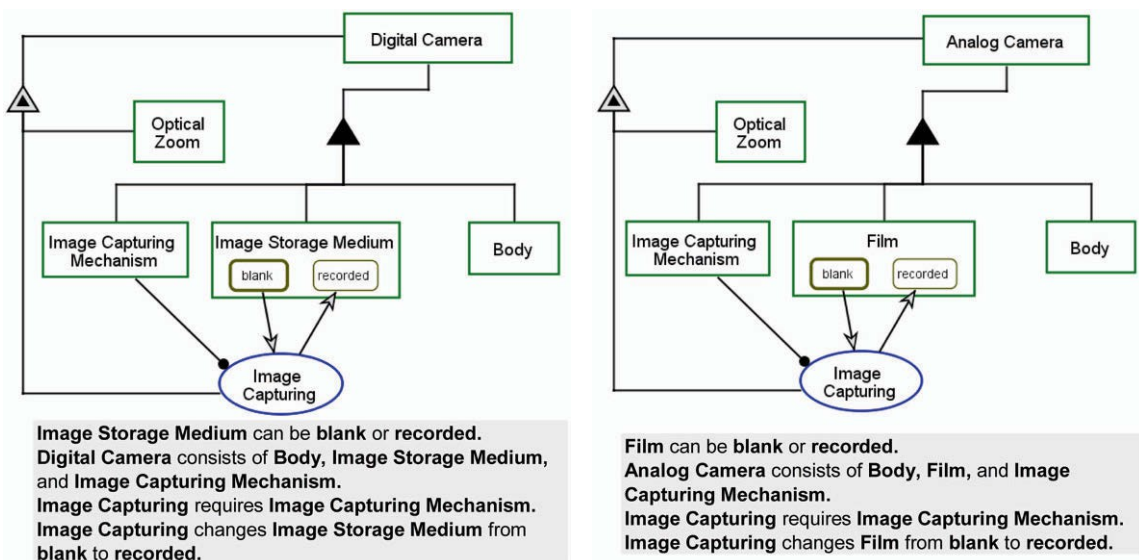
20.3 Specializace prostřednictvím rozlišovacího atributu

Poměrně často má obecný typ specializace, které se od obecného typu liší tím, že existuje určitý atribut obecného typu, jehož omezená hodnota definuje specializaci.

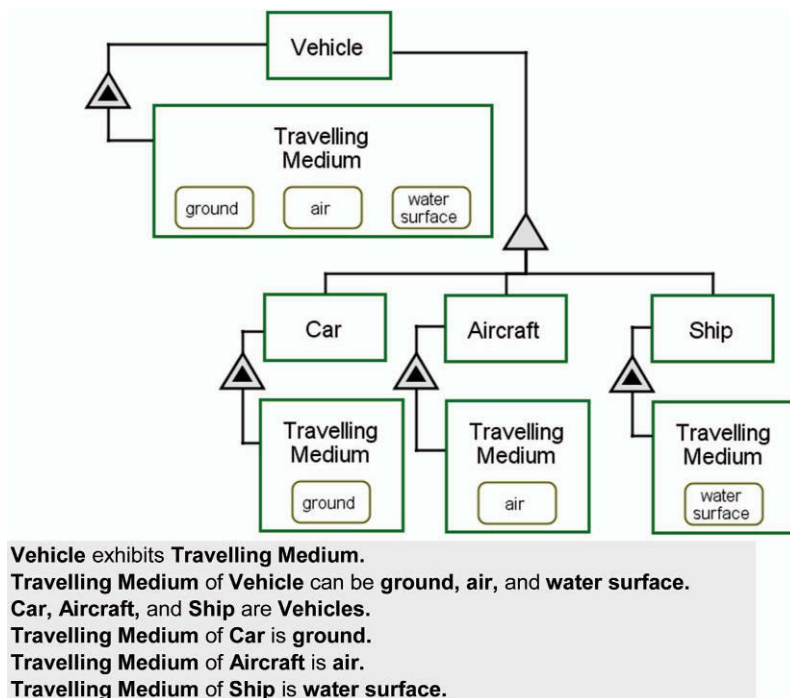
***Rozlišovací atribut** je zděděný atribut, jehož různé hodnoty definují odpovídající specializace.*

Obrázek 20.10 ukazuje OPD, ve kterém **vozidlo** vykazuje atribut **dopravní prostředek** s hodnotami **země**, **vzduch** a **vodní hladina**. **Dopravní prostředek** je rozlišující atribut **vozidla**, protože tři hodnoty **dopravního prostředku** definují tři specializace **vozidla**. Jedná se o **auto**, **letadlo** a **loď** s odpovídajícími hodnotami **dopravního prostředku** **země**, **vzduch** a **vodní hladina**.

Generál může mít více než jeden rozlišovací atribut. Maximální počet specializací s více než jedním rozlišovacím atributem je kartézský součin počtu možných hodnot pro každý rozlišovací atribut, přičemž některé kombinace hodnot atributů mohou být neplatné. Například v návaznosti na obsah **obr. 20.10** může být dalším atributem **vozidla účel** s dvěma hodnotami: **civilní** a **vojenský**. Na základě těchto dvou hodnot existují dvě specializace **vozidla**: **civilní vozidlo** a **vojenské vozidlo**. V důsledku vícenásobného dědičnosti je výsledkem dědičná mřížka, kde počet nejdetailnějších specializací by byl $3 \times 2 = 6$, a to následovně: **civilní automobil**, **civilní letadlo**, **civilní loď**, **vojenský automobil**, **vojenské letadlo** a **vojenská loď**.



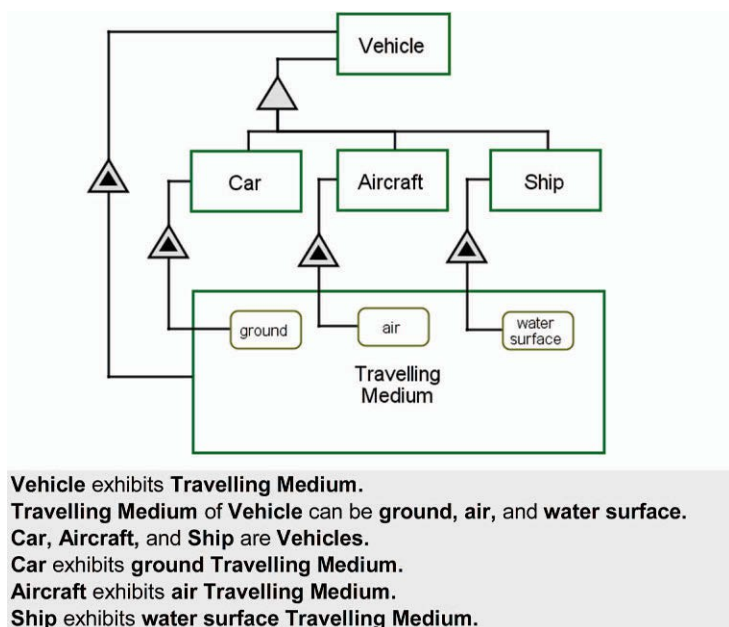
Obr. 20.8 Dědičnost stavů vyvolaná OPD na obr. 20.9. Vlevo: **Fotoaparát** je nahrazen **digitálním fotoaparátem** a médium pro zachycení obrazu médiem pro ukládání obrazu. Vpravo: **Fotoaparát** je nahrazen **analogovým fotoaparátem** a médium pro zachycení obrazu filmem.

Obr. 20.10 Rozlišovací atribut **Cestovní médium** a jeho specializace

20.4 Stavově specifikovaný charakterizační odkaz

Stavově specifikovaná charakterizační vazba je vazba mezi specializací a konkrétní hodnotou rozlišovacího atributu jejího obecného pojmu, která vyjadřuje skutečnost, že specializace může mít pro daný rozlišovací atribut pouze tuto hodnotu.

Graficky je odkaz charakterizace specifikovaný stavem trojúhelníkovým symbolem výstavy-charakterizace, jehož vrchol je spojen se specializací a jeho základna se specifickou hodnotou. Použitím odkazu charakterizace specifikovaného stavem je OPD na [obr. 20.11](#) výrazně kompaktnější než jeho ekvivalentní OPD na [obr. 20.10](#). Zde se rozlišující atribut „Dopravní prostředek vozidla“ s hodnotami „pozemní“, „vzdušný“ a „vodní“ objevuje pouze jednou, na rozdíl od čtyřkrát v [obr. 20.10](#). Model vyjadřuje „Auto“, „Letadlo“ a „Lod“ jako specializace „Vozidla“ a spojuje každou specializaci pomocí odkazu charakterizace specifikované stavem s odpovídající hodnotou „Dopravní prostředek“ „pozemní“, „vzdušný“ a „vodní“.



Obr. 20.11 Příklad charakterizačního odkazu specifikovaného stavem

20.5 Klasifikace – instancování

Instance je skutečná věc určité třídy věcí, které mají všechny stejnou sadu vlastností, stejnou strukturu a stejné chování. Například **Lassie** a **Blackie** na obr. 20.12 jsou instancemi třídy **Pes**. **Pes** je třída všech psů a **Lassie** je skutečným příkladem této třídy. Symbolem instancování je černý obrácený trojúhelník uvnitř většího bílého trojúhelníku.

V mluvené angličtině je věta „Lassie je pes“ přirozenější, ale fráze „je“ je vyhrazena pro specializační větu, takže aby se předešlo konfliktům a bylo to jasné, fráze „je instancí“ spojuje instanci s její třídou ve větě OPL, která vyjadřuje instancování. Množné číslo, používané pro více než jednu instanci, je „jsou instancemi“, jako v „**Bach**, **Beethoven** a **Brahms** jsou instancemi **skladatelů**“.

20.5.1 Třídy a instance

Věci, se kterými jsme se setkali při diskusi o generalizaci a specializaci, jsou *třídy* věcí, buď třídy objektů, nebo třídy procesů. Když jsme mluvili o objektech, měli jsme na mysli typický příklad jejich třídy objektů, vzor objektů, ze kterého lze objekty generovat.

***Třída** je šablona věci.*

***Instance** třídy je ztělesněním konkrétního identifikovatelného člena této třídy.*

Definice třídy a instance jsou obecnější než jejich protějšky v objektově orientovaném programování, protože se vztahují spíše k věcem než k objektům. V pojmech metamodelu, protože **věc** je **objekt** nebo **proces**, **třída** se specializuje na **třidu objektů** a **třidu procesů**. Stejně tak se **instance** specializuje na **instanci objektu** a **instanci procesu**: **instance objektu** je ztělesněním vzoru specifikovaného **třidou objektů** a **instance procesu** je ztělesněním vzoru specifikovaného **třidou procesů**.



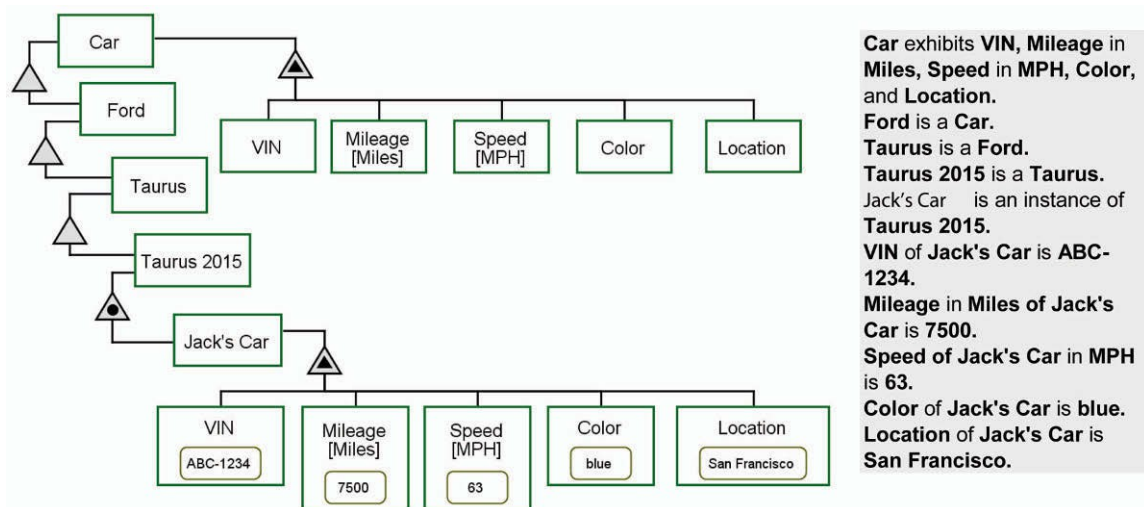
Obr. 20.12 Symbol instancování propojuje třídu (**Pes**) s jednou nebo více jejími instancemi.

Šablona, kterou třída definuje, zahrnuje vše, co je zděděno. Jak jsme viděli, v OPM to znamená, že jsou zděděny nejen vlastnosti, ale také strukturální a procedurální vztahy, a u objektových tříd jsou zděděny také stavy. Na rozdíl od specializované třídy nemůže instance vykazovat žádnou vlastnost, kterou její třída nevykazuje, ani nemůže být instance objektu ve stavu, který není stavem její třídy. Instance objektu může být v systému jednoznačně identifikována, takže v daném okamžiku je možné zjistit, zda existuje, a pokud ano, jaké jsou její stavy a hodnoty atributů.

20.5.2 Instanciace versus specializace

Generalizace-specializace je tranzitivní strukturální vztah, který vede k vytvoření hierarchického stromu. Každá úroveň v hierarchii obsahuje specializace úrovně nad ní. „Listy“ této hierarchie jsou *instance* třídy. Můžeme tedy říci, že instancování je zvláštním případem specializace, který v kontextu studovaného nebo vyvíjeného systému nelze dále specializovat. **Obrázek 20.13** ukazuje hierarchii specializace, která začíná na nejvyšší úrovni **Car** (Auto) a představuje stále více specializované třídy objektů, až se dostane k **Jack's Car** (**Jackovo auto**). Toto je první objekt, který je fyzický a jedinečný. Má **VIN** (identifikační číslo vozidla), které jej jednoznačně identifikuje, a v daném okamžiku lze specifikovat hodnoty nebo stavy všech jeho atributů, jako jsou **Color** (**Barva**), **Location** (**Umístění**), **Mileage** (**Najeto**) a **Speed** (**Rychlost**).

Instance je listem v hierarchii generalizace-specializace – není možné mít specializace instance. Dědičnost vlastností z třídy na její instance je přesně stejná jako dědičnost vlastností z nadřazené třídy na její podtřídu kdekoli v hierarchii generalizace-specializace. Jediné rozdíly jsou, že (1) instance nemůže mít další specializace, protože se nachází na spodku hierarchie, a (2) pouze instance má konkrétní hodnoty svých atributů, jak ukazuje **obr. 20.13**.



Obr. 20.13 Hierarchie specializací třídy **Car** až po instanci **Jack's Car** a její konkrétní hodnoty atributů v okamžiku pozorování

20.6 Relativita instance

Stejně jako mnoho jiných pojmů, se kterými jsme se setkali, je i pojem instance relativní vzhledem k systému diskurzu. To, co je pro určitý systém považováno za instanci třídy, může být pro jiný systém pouze podtřídou nadřazené třídy. Instance v jednom systému může být třídou, která má instance nebo která se dále rekurzivně specializuje do jemnějších tříd, které nakonec mají instance.

Abychom to demonstrovali, podívejme se na několik příkladů ze světa automobilů. Viděli jsme, že **Taurus 2015** je objektová třída všech instancí automobilů značky **Ford** modelu **Taurus** vyrobených v roce **2015**. Předpokládejme, že systém, který nás nyní zajímá, je systém pro porovnávání a hodnocení automobilů modelového roku 2015. Jednou z instancí v tomto systému je **Taurus 2015** a jedná se o instanci objektové třídy **Modelový rok 2015 Auto**. Fyzická auta se specifickým **VIN** neexistují a v tomto systému nemají žádný význam. V hierarchickém stromu gen-spec je **Taurus 2015** jedním z listů: nemá pod sebou žádné další specializace.

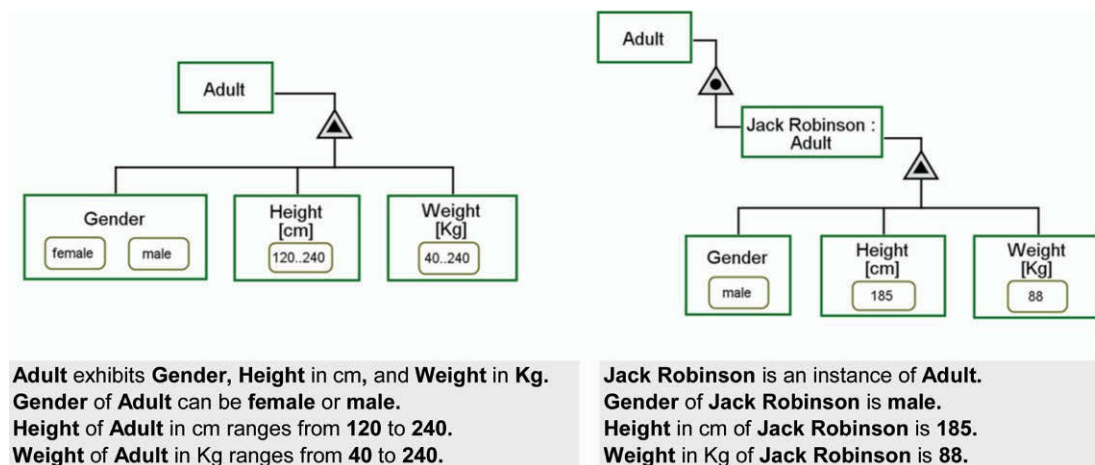
Jako další příklad uvažujme národní dálniční systém, ve kterém se architekti systému zajímají o různé typy vozidel, která používají silnice. Co je pro ně důležité, je velikost, hmotnost, průměrná rychlost a průměrná roční vzdálenost, kterou každý typ vozidla urazí. Návrháři tohoto systému se proto rozhodli rozdělit vozidla do tří typů: osobní automobily, nákladní automobily a autobusy. I když jsou tyto tři typy specializacemi vozidla, pro uvažovaný systém jsou také třemi instancemi objektové třídy vozidlo. Architekti se nezajímají o každý jednotlivý osobní automobil, autobus nebo nákladní automobil, takže počet každého typu vozidla, jeho průměrná rychlost, počet najetých kilometrů atd. jsou atributy vozidla, které jsou zděděny jeho třemi instancemi.

Zvažme nyní jiný systém **daňového úřadu** pro **motorová vozidla** v některé zemi, který pro daňové účely rozlišuje mezi **daňovými třídami motorových vozidel** takto: **užitkové vozy**, **sedany**, **sběratelská vozidla**, **sportovní užitková vozidla** a **luxusní vozy**. V tomto systému se vozy rozlišují podle jejich **tržní hodnoty** a **použití**. Systém navíc udržuje a neustále aktualizuje seznam všech **výrobci vozidel** a všech **modelů vozidel** podle **modelového roku** s uvedením, který **model vozidla** patří do které **daňové třídy**. **Daňová třída** je zde atributem **vozidla**. **Užitkové dodávky**, **sedany** atd. jsou hodnoty atributu **daňové třídy vozidla**. Instancemi třídy **vozidla** v tomto systému jsou různé **modelové roky**, protože systém se zabývá pouze stanovením daňových sazeb pro automobily podle **daňové třídy** a nezabývá se jednotlivými automobily.

Na závěr se podívejme na prodejnu automobilů. Zde má samozřejmě každý jednotlivý automobil svůj vlastní záznam, včetně VIN, značky, modelu, roku výroby, majitele atd. Jedná se o „klasický“ případ instance, podobný tomu, který je uveden na [obr. 20.13](#), kde každá instance je fyzická entita s jedinečným identifikátorem. Jak jsme však viděli, instance mohou být i informatické, jako například modely automobilů, typy vozidel nebo záznamy v souboru.

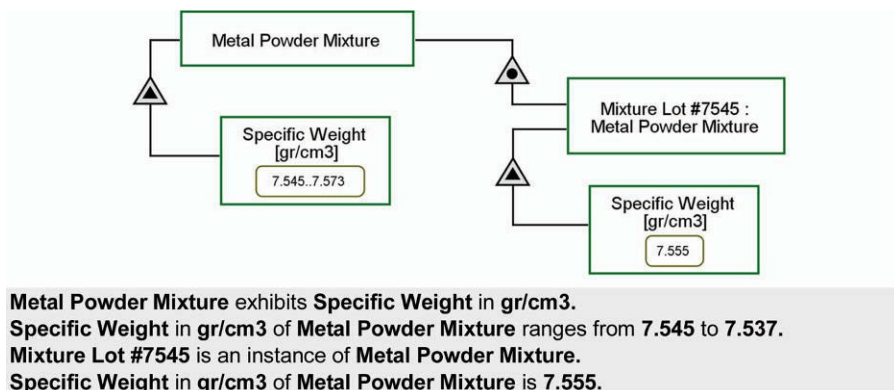
20.7 Omezení hodnot atributů

Třída může být použita k omezení možného rozsahu hodnot atributů. Na [obr. 20.14](#) je **Adult** třída se třemi atributy: **Gender** (pohlaví) s možnými hodnotami **female** (**žena**) a **male** (**muž**), **Height** (**výška**) v **cm** s možnými hodnotami **120..240** (120 až 240) a **Weight** (**váha**) v **kg** s možnými hodnotami **40..240**. **Jack Robinson** je instancí třídy **Adult** s hodnotou **Gender muž**, hodnotou **Height 185 cm** a hodnotou **Weight 88 kg**. Jak ukazuje [obr. 20.14](#), za názvem instance třídy **Adult**, v tomto případě **Jack Robinson**, může následovat symbol středníku „:“ a za ním název třídy. To je užitečné, když se v OPD objeví pouze instance, aniž by byla připojena k této třídě.



Obr. 20.14 Hodnoty atributů třídy **Person** jsou omezeny rozsahy hodnot (třída vlevo a instance vpravo)

OPD na [obr. 20.15](#) představuje třídu **Metal Powder Mixture** (směs kovového prášku) a udává, že hodnota atributu **Specific Weight** (specifická hmotnost) se může pohybovat v rozmezí **7,545 až 7,537 g/cm³**. Operační (runtime) instance třídy **Metal Powder Mixture** je **Mixture Lot #7545** (směs č. 7545) s hodnotou atributu **Specific Weight** **7,555 g/cm³**. Tato hodnota je v povoleném rozmezí.



Obr. 20.15 Omezení hodnoty atributu. Vlevo: třída a rozsah hodnot atributu. Vpravo: instance a její skutečná hodnota, která se nachází v omezeném rozsahu

Věta OPL „**Směs č. 7545** vykazuje **měrnou hmotnost v g/cm³**“ není v OPL na [obr. 20.15](#) uvedena, protože tato věta vyplývá implicitně z vyjádřeného faktu „**Směs č. 7545** je instancí **směsi kovového prášku**, a proto **směs č. 7545** zdědila tento atribut od **směsi kovového prášku**“.

20.8 Instance procesů

Instanciace OPM se nevztahuje pouze na objekty, ale také na procesy. Procesy, se kterými jsme se dosud setkali, jsou ve skutečnosti *třídy procesů*: jsou to *vzorce událostí, které zahrnují třídy objektů*.

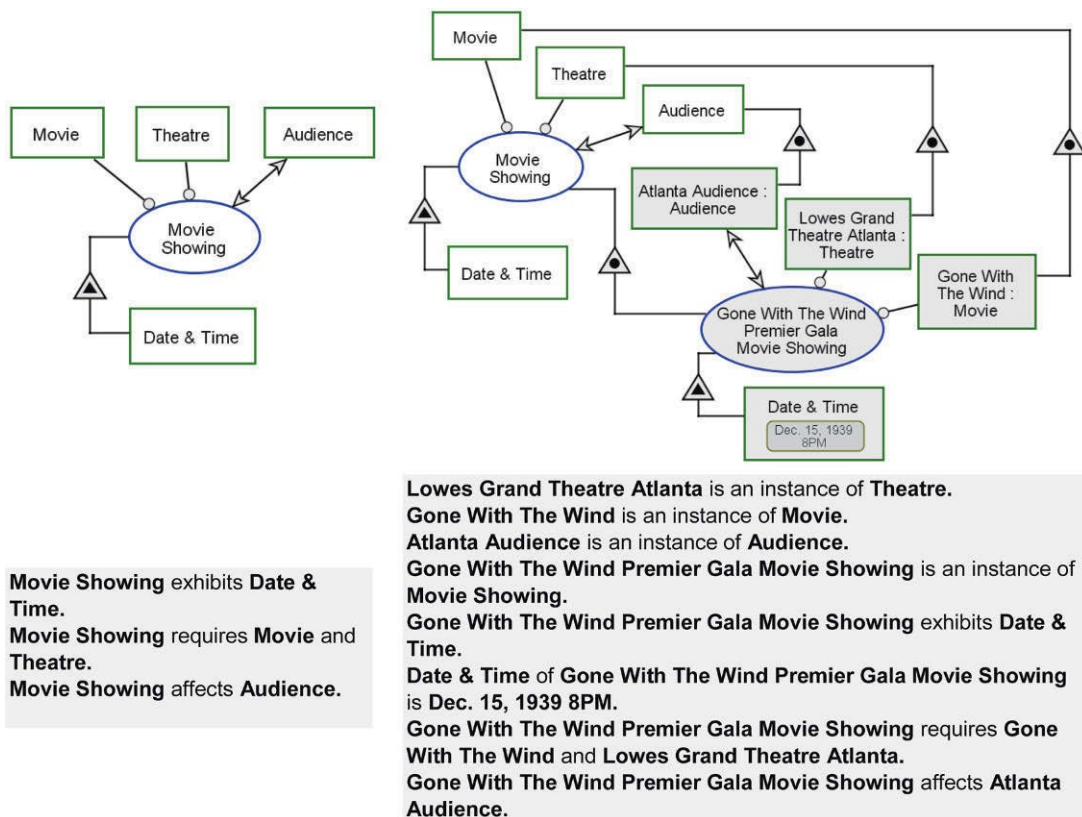
Třída procesů je vzor události (sekvence podprocesů), který zahrnuje třídy objektů, které jsou členy souboru objektů dané třídy procesů.

Výskyt procesu, který se řídí tímto vzorem a zahrnuje konkrétní instance objektů v jeho sadách objektů předprocesu a postprocesu, je instancí procesu. Instance procesu je tedy konkrétním výskytem třídy procesu, ke které tato instance patří. Každá instance procesu je proto spojena s odlišnou sadou instancí objektů předprocesu a postprocesu.

Procesní instance je konkrétní výskyt procesní třídy, ke které daná instance patří.

Síla konceptu třídy procesu spočívá v tom, že umožňuje modelování procesu jako šablony nebo protokolu pro určitou transformaci, kterou třída objektů prochází. Tato transformace nezahrnuje ani časoprostorový rámec, ani konkrétní sadu instancí objektů, s nimiž je instance procesu spojena.

související; ty lze identifikovat pouze na úrovni instance nebo na operační úrovni systému.



Obr. 20.16 Film zobrazující jako příklad třídu procesu (vlevo) a její instanci (vpravo)

Instance procesu je konkrétní výskyt třídy procesu, jehož sady objektů předprocesu a postprocesu jsou sady instancí objektů. Instance procesu má zejména časové razítko, konkrétní datum a čas, kdy proces začal nebo skončil. Obrázek 20.16 znázorňuje vlevo **promítání filmu** jako příklad třídy procesu, s **filmem** a **kinem** jako nástroji této třídy procesu, **datem a časem** jako jeho atributem a **diváky** jako ovlivněnou třídou. V OPD vpravo je **premiéra filmu Jih proti Severu** procesní instancí třídy procesu **promítání filmu**. Všechny instance jsou šedé, aby se odlišily od svých tříd. **Gone With The Wind** je instancí filmu, **Atlanta Theatre** je instancí kina, **Atlanta Audience** je instancí publika a **15. prosince 1939 20:00** (Dirks 2015) je hodnotou **Date & Time**, kdy se procesní instance odehrála. Stejná instance objektu se může účastnit dvou nebo více instancí procesu. Například stejná instance filmu, identifikovaná podle názvu **Jih proti Severu**, se může účastnit všech instancí procesu **Promítání filmu Jih proti Severu** (kromě premiérové gala), ale každé **Atlanta Audience** je jinou instancí **Audience**, protože se skládá z jiné skupiny diváků.

20.9 Shrnutí

- **Generalizace-specializace** je vztah mezi obecnou věcí a specializací této věci.
- **Klasifikace-instancování** je vztah mezi třídou věcí a jedinečnou instancí, která patří do této třídy.
- Zobecnění-specializace vede k **dědičnosti** od zobecněné věci ke specializované věci (věcem).
- Dědičnost se týká vlastností (atributů a operací), strukturálních vztahů a procedurálních vztahů. U objektů se dědí také stavy.
- Procesy OPM se specializují podobným způsobem jako objekty.
- Stavy specializovaných objektů mohou přepsat zděděné stavy.
- Třída je šablona, ze které lze generovat věci, které instanciují třídu, jako členy této třídy.
- Instance je relativní pojem. Specializace v jednom systému může být instancí v jiném systému.
- Instance procesu je konkrétní výskyt procesu v daném časovém okamžiku, jehož související sada objektů je sadou instancí objektů.

20.10 Úlohy

1. Uveďte dva příklady specializací objektů a dva příklady specializací procesů. Specifikujte je v OPD a OPL.
2. Vytvořte specializační hierarchii sportovních her, která by zahrnovala minimálně volejbal, basketbal, fotbal, americký fotbal, tenis a baseball. Použijte OPM k zobrazení společných a zděděných vlastností a vlastností specifických pro danou hru.
3. Opakujte předchozí problém pro hierarchii specializací typů atletických sportů, která by zahrnovala alespoň tři typy běhu, tři typy plavání a tři typy hodu.
4. Vzhledem k dědičnosti procedurálních odkazů, jsou efektové odkazy nadbytečné? Proč ano, nebo proč ne?
5. Nakreslete OPD vyjádřené v odstavci OPL níže.
Pilot, námořník a řidič jsou povolání.
Letadlo, loď a nákladní automobil jsou dopravní systémy. Létání, plavba a řízení jsou dopravní činnosti.
6. Doplňte OPD z předchozí otázky pomocí následujících modelových faktů: (1) Pilot, námořník a řidič se zabývají létáním, plavbou a řízením. (2) Létání, plavba a řízení vyžadují letadlo, loď a nákladní automobil.

7. Uveďte příklady dvou systémů, kde instance v prvním systému jsou specializacemi v druhém systému. Nakreslete OPD a napište OPL těchto systémů.
8. Hlavní typy svařování jsou: (1) Plynové – Používá se plamen plynu nad kovy, dokud nevznikne roztavená tavenina
Nejoblíbenějšími palivy používanými s kyslíkem jsou acetylen a vodík. (2) Obloukové svařování – dva kovy se spojují vytvořením elektrického oblouku mezi zakrytou kovovou elektrodou a základním kovem. (3) Řezání kyslíkem a obloukem – řezání kovů při svařování je odřezávání nebo odstraňování kovu plamenem nebo obloukem. Použijte OPM k popisu těchto typů svařování.
9. Uveďte tři příklady elektrických spotřebičů ve vaší domácnosti. U každého z nich popište jeho třídu objektů s alespoň třemi úrovněmi agregace-hierarchie účasti a operace, které provádí. Použijte symbol instancování k označení vašeho zařízení a uveďte atribut, který jej jednoznačně identifikuje.

Kapitola 21

Řízení složitosti: Zjemnění a abstrakce

Lidská mysl koneckonců dokáže zpracovat jen omezené množství dat najednou, než je přetížena.

C. Downton (1998)

Potřeba systémové analýzy a návrhových strategií vyplývá právě ze složitosti. Kdyby byly systémy nebo problémy natolik jednoduché, že by je člověk pochopil pouhým pohledem, nebyla by potřeba žádná metodika. Vzhledem k nutnosti řešit rozsáhlé a složité problémy musí být metodika vývoje systémů vybavena komplexním přístupem, podpořeným sadou spolehlivých a užitečných nástrojů pro kontrolu a řízení složitosti. OPM poskytuje čtyři mechanismy zjemnění a abstrakce pro řízení komplexnosti systémů: (1) rozbalení–sbalení, (2) přiblížení–oddálení, (3) vyjádření stavu–potlačení stavu a (4) vytvoření pohledu. Tyto mechanismy, definované a diskutované v této kapitole, umožňují specifikaci kontextualizovaných segmentů modelu jako samostatných, ale vzájemně propojených OPD. Společně poskytují kompletní model funkčního systému poskytujícího hodnotu. Tyto mechanismy umožňují prezentovat a prohlížet modelovaný systém a prvky, které obsahuje, v různých kontextech, které jsou vzájemně propojeny společnými objekty, procesy a vztahy. Sada jasně specifikovaných a kompatibilních propojených diagramů objektů a procesů (Object-Process Diagrams) zcela specifikuje celý systém v odpovídající míře detailu a poskytuje komplexní reprezentaci tohoto systému s odpovídajícím textovým popisem modelu v OPL. Tato kapitola se zabývá otázkami řízení složitosti a specifikuje různé mechanismy abstrahování a zpřesňování.

21.1 Potřeba řízení složitosti

Analýza je proces postupného rozšiřování znalostí a porozumění lidského analytika o architektuře systému – kombinaci struktury a chování systému, která mu umožňuje plnit jeho funkci. To je typické pro práci vědce, který se v jistém smyslu zabývá reverzním inženýrstvím přírody a systémů v ní. Analogicky je navrhování – hlavní inženýrský úkol – procesem postupného zvyšování množství detailů o navrhovaném systému. Složitost je vlastní systémům v reálném životě: Během tohoto procesu navrhování brzy množství detailů obsažených v jakémkoli reálném systému rozumné velikosti přetíží systémového analytika nebo architekta, který musí být vybaven konceptem a nástroji k řešení tohoto problému explozivního nárůstu detailů. S přirozenou složitostí systému nemůžeme moc dělat, ale pomocí jednoduchého modelovacího rámce můžeme výrazně snížit komplikovanost systému – to, jak komplikovaný se jeví člověku, který se dívá na model specifikující systém. OPM se snaží minimalizovat komplikovanost prostřednictvím jednoduchosti jazyka.

Analýza požadavků a koncepční návrh jsou prvními kroky v životním cyklu nového systému, produktu nebo projektu. Vytváření (někdy nevědomého) odporu ze strany potenciálního publika – různých zainteresovaných stran – k přijetí výsledků analýzy a návrhu, protože vypadají příliš složitě a tudíž zastrašující, může mít nepříznivý účinek a ohrozit pravděpodobnost úspěchu následujících fází vývoje produktu.

Závažnost a četnost problému explozivního nárůstu detailů vyžaduje adekvátní řešení, které uspokojí potřeby komunity zabývající se modelováním a analýzou systémů. Hlavním testem jakékoli metodiky analýzy je proto řízení složitosti – míra, do jaké poskytuje rozumné nástroje pro řízení stále rostoucí složitosti výsledků modelování koherentním, jasným a užitečným způsobem. Takové nástroje pro řízení složitosti jsou nesmírně důležité pro organizaci znalostí, které systémoví architekti a návrháři shromažďují a generují během procesu navrhování a projektování systému. Stejně důležitá je role nástrojů pro řízení složitosti při usnadňování komunikace výsledků analýzy a návrhu ostatním lidem, včetně zákazníků, příjemců, kolegů, nadřízených a vývojářů systémů v rámci vývojového cyklu – implementátorů, testerů, operátorů atd.

Při pokusu o začlenění detailů do jednoho velkého diagramu se množství nakreslených symbolů velmi zvětší a jejich vzájemné propojení se rychle stane zamotanou sítí. Protože se diagram stal tak nepřehledným, je stále méně ovladatelný a obtížně srozumitelný. Systémoví architekti se s tímto jevem explozivního nárůstu detailů setkávají každý den a každý, kdo se pokusil modelovat systém, který není hračkou a má alespoň mírnou složitost, s tímto popisem bude souhlasit a potvrdí ho. K tomuto informačnímu přetížení dochází i v případě, že jazyk (například UML a SysML) doporučuje používat více druhů diagramů pro různé aspekty systému. I když některé druhy diagramů mohou být jednodušší než jeden druh (jako v OPM), jejich kombinace za účelem získání holistického pohledu na systém je kognitivně mnohem obtížnější. Jazyk pro modelování systémů musí zahrnovat integrální mechanismy pro řízení a správu této složitosti. To znamená, že musí být schopen prezentovat a zobrazovat systém na různých úrovních detailů, které jsou vzájemně konzistentní.

21.2 Tvrzení o složitosti modelu

Základním principem řízení složitosti OPM je následující *princip hierarchie detailů OPM*.

Princip OPM detailní hierarchie

Kdykoli se OPD stane obtížně srozumitelným kvůli nadměrnému množství detailů, vytvoří se nový, odvozený OPD.

Vytvoření nového OPD se provádí pomocí jednoho z prvních dvou mechanismů řízení složitosti – přiblížení nebo rozbalení – s využitím principu OPM reprezentace modelových faktů. Tento princip stanoví, že *modelový fakt OPM se musí objevit alespoň v jednom OPD, aby mohl být reprezentován v modelu*. Na základě tohoto principu můžeme z potomka, nově vytvořeného OPD, ve kterém byla upřesněna konkrétní věc, vynechat jakýkoli modelový fakt, který se již objevil v předkovi OPD a není potřebný k vyjádření určitého bodu v novém OPD, aniž by došlo ke ztrátě tohoto faktu z modelu. Tímto způsobem lze nové OPD udržet jednoduché, protože nemusí nést veškerou „zátěž“ svých předků. To umožňuje udržovat všechny OPD dostatečně jednoduché, aby nepřekračovaly omezené kognitivní schopnosti člověka.

Rozhodnutí o tom, kdy se OPD stává příliš složitým kvůli nadměrnému množství detailů, je ponecháno na uvážení modeláře, protože jej nelze definovat pouhým stanovením maximálního počtu modelových prvků v OPD. Existují i další faktory, jako je pravidelnost, rozložení a křížení spojů, které ovlivňují srozumitelnost. Nicméně modelovací nástroj, jako je OPCAT, by měl omezit velikost plátna, na kterém je nakreslen jeden OPD. To nepřímo omezuje počet entit a vynucuje pravidelné používání přiblížení a rozbalení.

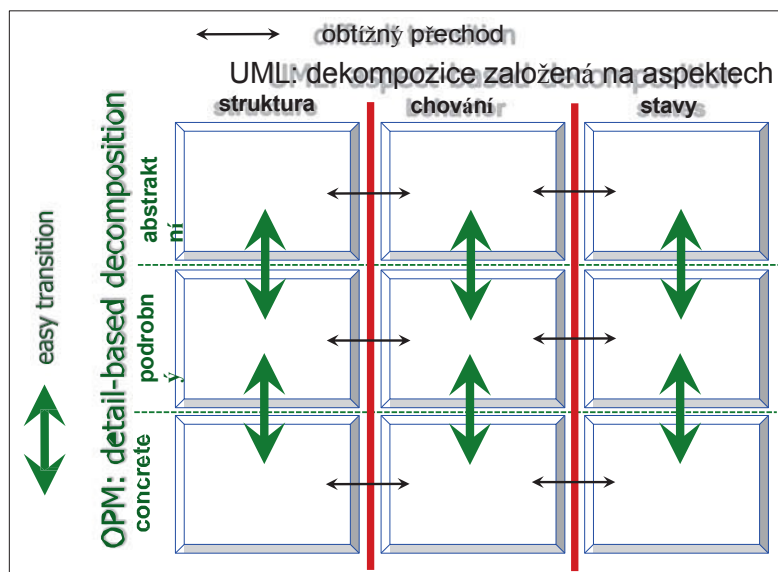
Vzhledem k tomu, že toto zjemnění a odstranění detailů lze provádět rekurzivně a libovolný početkrát, můžeme řešit vysoce komplexní systémy a přitom zachovat model lidsky přístupný a srozumitelný. Proto můžeme učinit následující tvrzení o složitosti modelu OPM:

Tvrzení o složitosti modelu OPM

Použitím mechanismů zjemnění přiblížení a rozbalení na objekty nebo procesy se stavem může OPM koncepčně modelovat systémy na jakékoli úrovni složitosti.

21.3 Rozklad založený na aspektech versus rozklad založený na úrovni detailů

UML a SysML řeší problém řízení složitosti systémů především pomocí **aspektové dekompozice** – rozdělení systémového modelu na 14 (UML) a 9 (SysML) různých typů diagramů pro modelování různých aspektů systému – struktury, dynamiky, přechodů stavů, časování atd.



Obr. 21.1 Dvě ortogonální strategie „rozděl a panuj“

OPM prosazuje integraci různých aspektů systému do jediného modelu a používá ortogonální, **detailní dekompozici**: namísto použití samostatného modelu pro každý aspekt systému řeší OPM inherentní složitost systému dekompozicí systému do hierarchie samopodobných diagramů stejného druhu – OPD – pomocí mechanismů abstrakce a zpřesňování. Tyto mechanismy umožňují prezentovat a prohlížet systém a jeho součásti na různých úrovních detailu. Celý systém je kompletně specifikován prostřednictvím své sady OPD – sady kompatibilních OPD, z nichž každý poskytuje částečný pohled na systém, který je zkoumán nebo vyvíjen, a které společně poskytují úplný obraz systému. Každý OPD je doprovázen automaticky generovaným odstavcem OPL.

Obrázek 21.1 ukazuje dvě ortogonální strategie řízení složitosti. V aspektově orientované dekompozici oddělují dvě silné, plné, svislé čáry od sebe aspekty struktury, chování a přechodu stavů. Tenké obousměrné vodorovné šipky přes tyto čáry symbolizují obtížný přechod mezi různými modely. Detailně orientovaná dekompozice je znázorněna dvěma tenkými, přerušovanými, vodorovnými čarami, které oddělují od sebe různé úrovně detailů – abstraktní, detailní a konkrétní. Silné obousměrné svislé šipky symbolizují snadný přechod mezi úrovněmi detailů. Diagram je schematický; v žádném případě neznamená, že v OPM existují pouze tři úrovně abstrakce. Ve skutečnosti není tento počet omezen. Diagram by také neměl být interpretován tak, jako by v přístupu s více typy diagramů existovaly pouze tři typy diagramů.

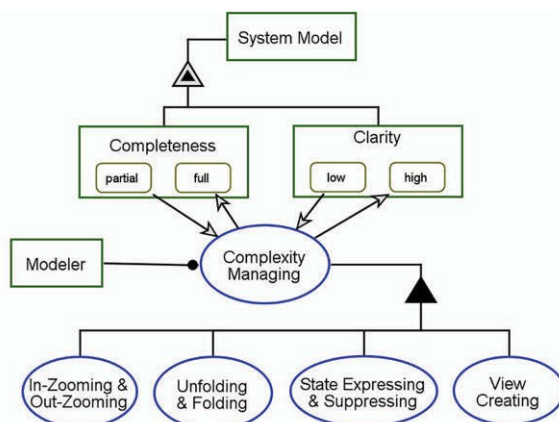
21.4 Kompromis mezi úplností a srozumitelností

Stejně jako většina klasických inženýrských problémů, i řízení složitosti vyžaduje kompromis, který musí být vyvážen mezi dvěma protichůdnými požadavky: úplností a srozumitelností. *Úplnost* znamená, že systém musí být specifikován do posledního relevantního a nezbytného detailu. *Srozumitelnost* znamená, že pro sdělení výsledků analýzy a návrhu musí být dokumentace, ať už textová nebo diagramová, čitelná a srozumitelná. Výzva složitosti vyžaduje vyvážení těchto dvou sil, které táhnou opačným směrem a je třeba je sladit: Na jedné straně úplnost vyžaduje, aby byly podrobnosti systému stanoveny v co nejširším rozsahu. Na druhé straně potřeba srozumitelnosti ukládá horní limit na úroveň složitosti každého jednotlivého diagramu a neumožňuje diagram, který je příliš přeplněný nebo zatížený.

Obrázek 21.2 je model OPM částí **řízení složitosti** a jeho vlivu na atributy **úplnosti** a **srozumitelnosti** systémového modelu. Řízení složitosti musí řešit a vyřešit tento problém kompromisu mezi úplností a srozumitelností tím, že najde správnou rovnováhu mezi těmito dvěma protichůdnými požadavky. OPM dosahuje srozumitelnosti abstrahováním a úplnosti zdokonalováním. Abstrakce, která je opakem zdokonalování, šetří místo a snižuje složitost, ale je to na úkor úplnosti. Naopak zdokonalování, které přispívá k úplnosti, je na úkor ztráty srozumitelnosti. „Nic není zadarmo“, jak je tomu obvykle u technických problémů, existuje jasný kompromis mezi úplností detailů a srozumitelností jejich prezentace. Řešení, které OPM navrhuje, spočívá v tom, že každý OPD je dostatečně jednoduchý a specifikace systému je rozdělena mezi sadu vzájemně propojených a vzájemně si vědomých OPD, které obsahují informace na různých úrovních detailu. Abstrahování a zdokonalování jsou analytické nástroje, které umožňují dosáhnout správné rovnováhy mezi srozumitelností a úplností.

21.5 Vyjádření stavu a potlačení stavu

Explicitní zobrazení stavů objektu v OPD může vést k příliš přehlněnému nebo rušivému diagramu, který je obtížné číst nebo pochopit. OPM umožňuje potlačení stavu – skrytí některých nebo všech stavů objektu, jak jsou reprezentovány v konkrétním OPD, pokud tyto stavy nejsou v kontextu daného OPD nutné. Na [obr. 21.4](#) byly potlačeny dva stavy každého z obou atributů tvořících OPD na [obr. 21.2](#), takže pár vstup-výstup se změnil na efektní odkaz ([obr. 21.3](#)).



System Model exhibits **Completeness** and **Clarity**.

Completeness of **System Model** can be **partial** or **full**.

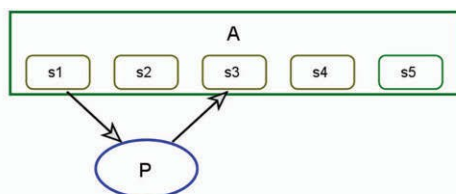
Clarity of **System Model** can be **low** or **high**.

Modeler handles **Complexity Managing**.

Complexity Managing consists of **In-Zooming & Out-Zooming**, **Unfolding & Folding**, **State Expressing & Suppressing**, and **View Creating**.

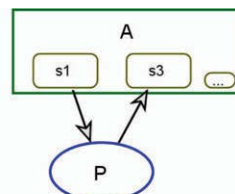
Complexity Managing changes **Clarity** from **low** to **high** and **Completeness** from **partial** to **full**.

Obr. 21.2 Součásti řízení složitosti a jeho vliv na úplnost a srozumitelnost systémového modelu atributů



A can be **s1**, **s2**, **s3**, **s4**, or **s5**.

P changes **A** from **s1** to **s3**.



A can be **s1**, **s3**, or at least one other state.

P changes **A** from **s1** to **s3**.

Obr. 21.3 Objekt se stavem, u kterého jsou vyjádřeny všechny stavy (vlevo), a potlačená verze (vpravo)

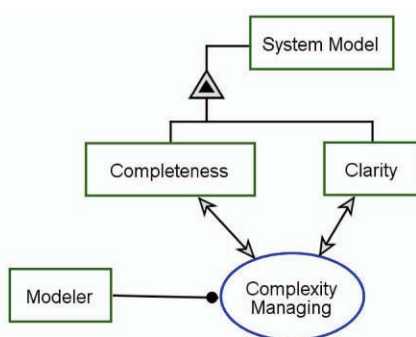
Opačná operace potlačení stavu – vyjádření stavu – odhalí jeden nebo více skrytých stavů objektu. Modelář může potlačit libovolnou podmnožinu stavů. Úplná množina stavů objektu je sjednocením

souboru stavů stejného objektu, které se objevují ve všech OPD v souboru OPD – souboru OPD celého modelu OPM.

Graficky bude anotace označující, že objekt představuje správnou podmnožinu (tj. alespoň jeden, ale ne všechny) svých stavů, malým symbolem potlačení stavu v pravém dolním rohu objektu. Tento symbol se zobrazuje jako malý stav s označením elipsou, což znamená existenci jednoho nebo více stavů, které zobrazení potlačuje. Textovou ekvivalencí symbolu potlačení stavu je vyhrazená fráze OPL „nebo alespoň jeden další stav“.

21.6 Rozbalení a sbalení

Rozbalení je mechanismus pro zpřesnění, rozpracování nebo rozložení. Rozbalení odhalí sadu věcí, které souvisejí s rozbalenou věcí – zpřesnitelnou. Výsledkem rozbalení je hierarchický strom, jehož kořenem je zpřesnitelná věc. S kořenem jsou spojeny zpřesněné věci – jedna nebo více věcí – části, specializace, vlastnosti nebo instance – které přidávají podrobnosti o zpřesnitelné věci prostřednictvím jedné nebo více ze čtyř základních strukturálních vztahů. Jakýkoli zdokonalený prvek může být zase zdokonalitelným prvkem pro další úroveň rozbalení.



System Model exhibits **Completeness** and **Clarity**.
Modeler handles **Complexity Managing**.
Complexity Managing affects **Clarity** and **Completeness**.

Obr. 21.4 OPD z obr. 21.2 po potlačení stavu dvou atributů a složení **komplexního řízení**

Skládání je opačná operace než rozbalení. Jedná se o mechanismus kolapsu a abstrakce, který lze aplikovat na hierarchii rozbaleného zdokonalitelného prvku. Skládání se aplikuje od spodní části hierarchie směrem nahoru. Každá operace skládání skryje některé nebo všechny zdokonalitelné prvky. Složením všech zdokonalitelných prvků zůstane pouze zdokonalitelný prvek – kořen stromové hierarchie.

Jelikož každé ze čtyř základních strukturálních vztahových propojení může podléhat rozbalení a složení, existují následující čtyři druhy párů rozbalení-složení.

- agregace rozvíjení — odhalování částí celku a participace skládání — skrývání částí celku,

- rozvíjení výstavy – odhalení rysů vystavovatele a skládání charakterizace – skrytí rysů vystavovatele,
- rozbalení generalizace – odhalení specializací obecného a sbalení specializace – skrytí specializací obecného a
- rozbalení klasifikace – odhalení instancí třídy a sbalení instancí – skrytí instancí třídy.

21.7 Rozbalení v diagramu a novém diagramu

Rozbalení lze provést buď v aktuálním OPD, nebo v novém OPD.

***Rozbalení v diagramu** je rozbalení, při kterém se vylepšitelné a jeho vylepšení objeví rozbalené ve stejném OPD, ve kterém bylo vylepšení původně.*

Vzhledem k tomu, že rozbalení používá jeden ze čtyř základních strukturálních odkazů, je rozbalení v diagramu graficky, syntakticky a sémanticky ekvivalentní použití odpovídajících základních strukturálních odkazů. Rozbalení v diagramu sice zvyšuje zatížení diagramu, ale ušetří nám nutnost vytvářet nový diagram. Pokud je však rafinátů mnoho nebo je aktuální OPD již obsazeno, dáme přednost rozbalení v novém diagramu.

***Rozbalení do nového diagramu** je rozbalení, při kterém se refineable a jeho refinees objeví rozbalené v novém OPD.*

Rozbalení v diagramu i v novém diagramu lze použít jak na objekty, tak na procesy. Graficky je v rozbalení v novém diagramu rozbalený prvek označen silnou konturou jak v abstraktnějším OPD, ve kterém se prvek objevuje složený, bez prvků, tak v novém, podrobnějším OPD, ve kterém se prvek objevuje rozbalený a propojený se svými prvky jedním nebo více základními strukturálními odkazy.

Modelář by měl na základě úvah o přehlednosti rozhodnout, zda použít rozvinutí v diagramu nebo v novém diagramu: Pokud je aktuální OPD již přeplněné a má tendenci být nepřehledné, mělo by být vytvořeno nové OPD, aby se zabránilo tomu, že se aktuální OPD stane neovladatelným. Pokud bylo použito rozložení v diagramu a OPD se později stalo příliš přeplněným, modelář může přejít z rozložení v diagramu na rozložení v novém diagramu, čímž zmírní složitost aktuálního OPD (za cenu dalšího OPD v sadě OPD). Modelář by tedy při rozhodování, zda použít rozbalení v diagramu nebo v novém diagramu, měl zohlednit kompromis mezi nepořádkem přidaným do stávajícího OPD a potřebou vytvořit nový OPD pro zobrazení detailů a souvisejících odkazů mezi nimi.

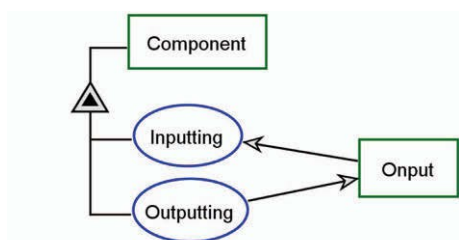
Částečné rozbalení lze znázornit pomocí symbolu neúplnosti pro agregaci, vystavení a klasifikaci. Aby modelář uspokojil konkrétní kontextovou relevanci pro OPD, může si vybrat, které refíněn se zobrazí rozbalené.

Zatímco rozbalení a sbalení lze použít jak na objekty, tak na procesy, častěji se používá pro objekty, zatímco procesy lze vylepšit pomocí přiblížení, o kterém bude řeč dále, nebo pomocí rozbalení. Rozbalení procesu je užitečné pro funkční rozklad, který je velmi důležitý v komplexních systémech. Takové systémy mají kromě základní funkce mnoho dalších pomocných funkcí, které jsou souběžné nebo

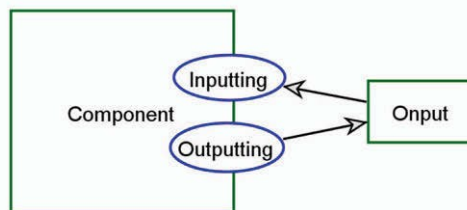
nezávislé na toku základní funkce. Obvykle existuje alespoň jedna další funkce – nastavení a správa systému, soubor mnoha služeb. Systémy orientované na služby nabízejí několik paralelních nebo souběžných služeb, které nelze považovat za pracující sériově. Systémy v reálném čase provádějí několik funkcí paralelně spíše než sériově, přičemž každá komponenta průběžně vzorkuje své vstupy z ostatních komponent a reaguje na ně.

21.8 Skládání portů

Procedurální odkaz z operace vystavovatele objektu na jiný objekt se během rozbalení operace ztratí, protože dva objekty nelze přímo propojit procedurálním odkazem. Podobně se během rozbalení operace ztratí procedurální odkaz z atributu vystavovatele procesu na jiný proces, protože dva objekty nelze přímo propojit procedurálním odkazem. Často je však žádoucí tyto odkazy zachovat (obr. 21.5).



Component exhibits **Inputting** and **Outputting**.
Inputting consumes **Onput**.
Outputting yields **Onput**.



Component exhibits **Inputting** and **Outputting** as ports.
Inputting consumes **Onput**.
Outputting yields **Onput**.

Obr. 21.5 Skládání portů. Vlevo: rozložený model. Vpravo: verze se složenými porty

Na základě Mordecai a Dori (2013) je možným řešením *skládání portů*, jak je znázorněno na obr. 21.6. Skládání portů je specializací skládání, mezistupněm mezi úplným složením a úplným rozložením, při kterém posuneme procesní refinee – operaci – na obrys objektu refineable – exhibitor. Graficky to vypadá podobně jako port diagramu aktivit SysML na složeném exhibitoru.

Skládání portů je užitečným znázorněním, pokud modelář chce použít obdélníky objektů k vyjádření fyzického uspořádání a relativních velikostí různých komponent systému. Vyhrazená fráze „jako porty“ (nebo „jako port“ v jednotném čísle) na konci věty o vystavovateli označuje skládání portů. Skládání portů lze také použít na atributy procesů.

21.9 Zvětšování a zmenšování

Přiblížení je operace zpřesnění, která se obvykle používá u procesů a která specifikuje podprocesy procesu, který je přiblížen, jakož i jejich (případně částečný) výkon nebo pořadí provedení. Jako

příkladu na [obr. 21.6](#) je proces **Check-Based Paying** z [obr. 19.13](#) přiblížen v potomku OPD vpravo, kde jsou zobrazeny jeho čtyři podprocesy, jak je vyjádřeno ve větě OPL:

Platba šekem se přiblíží na Psaní a podepisování, Doručování a přijímání, Indosování a předkládání a Proplacení a zrušení, v tomto pořadí.

Pořadí provedení těchto čtyř procesů se řídí *časovou osou OPM, která je zde zopakována*:

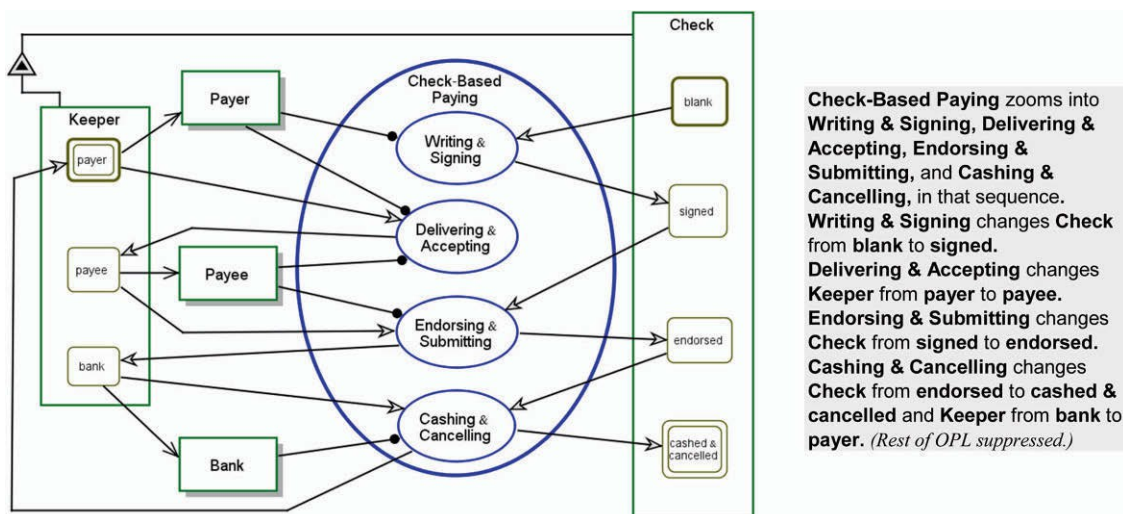
Princip časové osy OPM

Časová osa v rámci zvětšeného procesu je ve výchozím nastavení vedena od horní části zvětšené elipsy procesu k její spodní části.

Pořadí provádění je v OPL vyjádřeno vyhrazenou frází v dané sekvenci na konci zvětšené věty. Vysvětlení čtyř podprocesů v kontextu procesu **Check-Based Paying** umožňuje explicitně specifikovat, jak se stavy **Check** a **Keeper** mění v průběhu životního cyklu šeku, jak je také vyjádřeno ve větě OPL vlevo od OPD.

V rámci procesu přiblížení může existovat částečné pořadí: celkově existuje pořadí dané časovou osou, ale dva nebo více procesů může být prováděno paralelně. Jako příklad uvažujme proces P, který se zvětší na sedm podprocesů **SP1**, **SP2** ... **SP7**, takže nejprve se provede **SP1**, poté paralelně **SP2** a **SP3**, poté **SP4** a nakonec paralelně **SP5**, **SP6** a **SP7**. Věta OPL pak bude:

Pse rozdělí na SP1, paralelně SP2 a SP3, SP4 a paralelně SP5, SP6 a SP7, v tomto pořadí.



Obr. 21.6 Proces **Check-Based Paying** z [obr. 19.13](#) je rozdělen na podprocesy a ukazuje podrobnosti změn stavů, které procházejí **Check** a **Keeper**, stejně jako agenti zapojení do každého dílčího procesu.

OPM lze považovat za procesně orientovaný z hlediska upřednostnění modelování procesů (nejprve funkce systému, proces, který přináší externí hodnotu) a rekurzivního přiblížení této funkce při modelování objektů, které jsou relevantní pro každý proces na odpovídající úrovni detailu.

21.9.1 Zvětšování v diagramu a novém diagramu

Stejně jako rozbalení lze i přiblížení provést buď v aktuálním OPD, nebo v novém OPD.

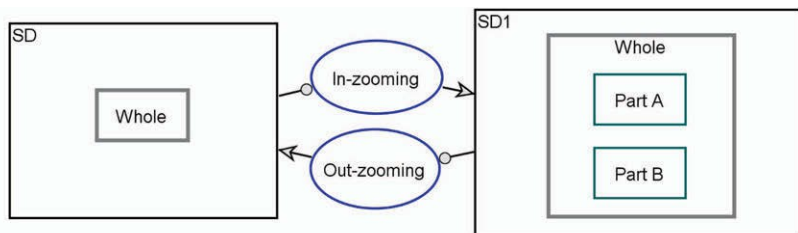
***Zvětšování v diagramu** je zvětšování, při kterém se nevytváří žádný nový OPD a zdokonalitelné prvky se zvětšují spolu se svými zdokonaleními ve stejném OPD.*

***Nové diagramové přiblížení** je přiblížení, při kterém se vylepšitelné prvky a jejich vylepšení zobrazí přiblížené v novém OPD.*

Všechny dosavadní příklady byly příklady nového diagramu. Jedná se skutečně o častější způsob přiblížení, protože přiblížení vyžaduje hodně „prostoru“ pro specifikaci vnitřních podprocesů a procesu, který je přiblížen, stejně jako pro zobrazení dalších relevantních objektů s odkazy na tyto nové podprocesy, což často vede k přeplnění aktuálního OPD. Jak však ukazuje [obr. 21.12](#), přiblížení v diagramu je také užitečné.

21.9.2 Přiblížení a oddálení objektů

Stejně jako procesní přiblížení má sémantiku agregace-účasti mezi přiblíženým procesem a jeho časově uspořádanými podprocesy, tak i objektové přiblížení má sémantiku agregace-účasti mezi přiblíženým objektem a jeho prostorově uspořádanými částmi. Jinými slovy, prostorové uspořádání podle sestupného nebo zleva doprava uspořádání částí určuje jejich pořadí. To je demonstrováno v metamodelu na [obr. 21.7](#): **Celek z SD** přiblíží **SD1** do **části A** a **části B** v tomto vertikálním pořadí.



SD is refined by in-zooming **Whole** in **SD1**.

Whole from **SD** zooms in **SD1** into **Part A** and **Part B**, in that vertical sequence.

Obr. 21.7 Metamodel přiblížení a oddálení objektů

Pokud by **části A** a **B** na [obr. 21.7](#) byly uspořádány horizontálně, věta OPL by zněla: **Celek z SD** se přiblíží v **SD1** na **část A** a **část B** v tomto horizontálním pořadí. Schopnost definovat pořadí v rámci objektů otevírá cestu k modelování tabulek a matic libovolného rozměru. Například můžeme přejmenovat **Celek** na [obr. 21.7](#) na **Tabulka** a **Část A** a **Část B** můžeme nazvat **Řádek 1** a **Řádek 2**. V další úrovni přiblížení lze každý řádek přiblížit, aby se zobrazily jeho prvky uspořádané horizontálně, např. **Řádek 1**

přibliží **Element (1,1)**, **Element (1,2)** a **Element 1,3**) v tomto horizontálním pořadí. **Element (1,2)** bude tedy druhým prvkem v prvním řádku matice. Třetí dimenzi lze dosáhnout přiblížením každého prvku, tentokrát vertikálně, a to lze provádět rekurzivně. Každá operace přiblížení, aplikovaná na všechny prvky na aktuální úrovni, přidá další dimenzi. Jelikož každý prvek může mít hodnotu, můžeme použít OPM k provádění maticových operací, jako je sčítání nebo násobení, a tabulky OPM lze použít pro relační databáze.

Čas je jednorozměrný a plyne pouze dopředu, takže k určení pořadí provádění procesů – načasování – jsme potřebovali pouze víslelou osu k určení pořadí podprocesů v přiblíženém procesu. Fyzické objekty jsou však trojrozměrné, takže pro přiblížení objektů můžeme alespoň schematicky modelovat relativní uspořádání částí objektů ve dvou rozměrech, přičemž využijeme skutečnosti, že papír nebo obrazovka počítače používané pro koncepční modelování jsou dvourozměrné. Omezením zde je, že objekty jsou obdélníkové, nikoli libovolného tvaru, ale stále můžeme získat schematické, i když hrubé, 2D uspořádání. Navíc, pokud je zvětšený objekt informatickým objektem, jako je tabulka nebo matice, jeho zvětšení může odhalit skutečné buňky tabulky nebo matice jako jednotlivé objekty.

21.10 Synchronní versus asynchronní zdokonalování procesů

Na rozdíl od rozbalení, které lze použít na každou ze čtyř základních strukturálních vazeb, má přiblížení pouze sémantiku agregace-účasti: Refineables jsou součástí přiblíženého refinee; nemohou být vlastnostmi, specializacemi ani instancemi. Kromě sémantiky celku a části však rozložení dílčích procesů v rámci přiblíženého procesu určuje jejich pořadí provedení. Naopak, když jsou procesy rozbaleny, jako jsou čtyři podprocesy **řízení složitosti** na [obr. 21.2](#), neexistuje pro ně žádný implicitní pořadí (pokud nemají pozitivní řaditelnost, která musí být označena symbolem řazení vedle černého trojúhelníku agregace). Rozbalení agregace **řízení složitosti** na [obr. 21.2](#) je správné, protože při modelování systému neexistuje předem stanovené pořadí použití čtyř operací zpřesnění. Modelář je spíše používá v libovolném pořadí podle potřeby. Jedná se o příklad asynchronního procesu. Na druhé straně je příkladem synchronního procesu **Check-Based Paying (platba na základě kontroly)** znázorněný na [obr. 21.6](#).

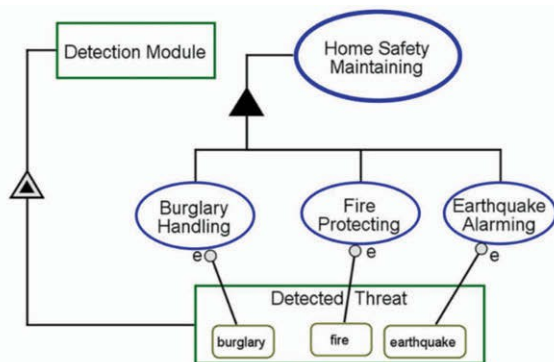
Synchronní proces je proces, jehož podprocesy mají předem dané, pevné pořadí.

Asynchronní proces je proces, jehož podprocesy nemají předem dané, pevné pořadí.

Vzhledem k rozdílu mezi agregací a přiblížením, pokud jde o procesy, je přiblížení vhodné pro modelování synchronních procesů, zatímco rozbalení agregace je vhodné pro modelování asynchronních procesů. Systém může mít kombinaci synchronních i asynchronních procesů. Navíc, pokud má proces několik synchronních podprocesů a jiné, které synchronní nejsou, může být stejný proces jak přiblížen, takže se jeho synchronní podprocesy zobrazí seřazené v přiblížené procesní elipse, tak i asynchronní – agregace rozbalená, buď ve stejném, nebo v samostatném OPD.

Jelikož základní strukturální vztah agregace-účast nepředepisuje žádné částečné pořadí výkonu procesu, musí modelování synchronního zjemnění procesu používat přiblížení, ve kterém lze pořadí definovat. Systém na [obr. 10.5](#) je synchronní: existuje pevné, dobře definované pořadí každého podprocesu v kontextu přiblížení **mytí nádobí**.

K modelování asynchronního zjemnění procesu používáme základní strukturální vazbu agregace-účast, a to buď prostřednictvím rozvinutí agregace v diagramu, nebo jako rozvinutí agregace procesu v novém diagramu. [Obrázek 21.8](#) znázorňuje část **systému domácí bezpečnosti**, který provádí funkci **udržování domácí bezpečnosti**, která zahrnuje dílčí procesy **řešení vloupání**, **požární ochrany** a **varování před zemětřesením**. Jelikož pořadí těchto tří dílčích procesů není známo, OPD používá rozvinutí agregace v diagramu s agregačně-participačním propojením z této funkce namísto přiblížené verze **udržování domácí bezpečnosti**. **Udržování domácí bezpečnosti** se přiblíží k opakujícímu se systémovému procesu **monitorování a detekce**, pro který je **detekční modul** nástrojem a **hrozba** je environmentálním procesem.



Home Safety Maintaining consists of **Burglary Handling**, **Fire Protecting**, and **Earthquake Alarming**.

Detection Module exhibits **Detection Threat**.

Detection Threat can be **burglary**, **fire**, or **earthquake**.

Burglary Detected Threat initiates **Burglary Handling**, which requires **burglary Detected Threat**.

Fire Detected Threat initiates **Fire Protecting**, which requires **fire Detected Threat**.

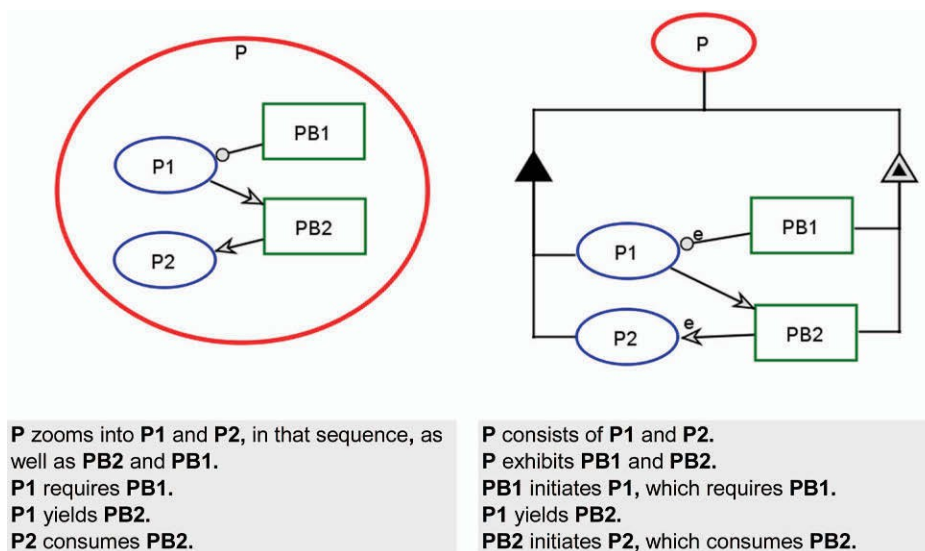
Earthquake Detected Threat initiates **Earthquake Alarming**, which requires **earthquake Detected Threat**.

Obr. 21.8 Udržování bezpečnosti domova je asynchronní systém

21.11 Ekvivalence mezi přiblížením a rozvinutím

Podrobnosti synchronního procesu lze vyjádřit jak pomocí přiblížení, tak pomocí rozbalení. [Obrázek 21.9](#) představuje proces P přiblížený v modelu OPM vlevo a jeho ekvivalentní model OPM vpravo, ve kterém je P rozbalený. Jak však vidíme na [obrázku 21.9](#), přiblížení je výhodnější, protože vyžaduje méně symbolů a výsledný odstavec OPL je kratší. Použitím přiblížení namísto rozvinutí můžeme použít odkazy na nástroje a výsledky namísto odkazů na události nástrojů a události výsledků, protože události v přiblíženém kontextu jsou implicitní.

Důležité je, že když je proces přiblížen, jeho podprocesy jsou jeho částmi, zatímco objekty odhalené v důsledku tohoto přiblížení jsou atributy procesu. Symetricky, když je objekt přiblížen, jeho vnitřní objekty jsou jeho částmi, zatímco jeho vnitřní procesy jsou jeho operacemi. Druhá skutečnost umožňuje zobrazit procesy jako operace objektu tím, že je umístíme do přiblíženého zobrazení tohoto objektu.



Obr. 21.9 Ekvivalence mezi přiblížením (vlevo) a rozvinutím (vpravo)¹

21.12 Systémová mapa a konečný OPD

Existuje přesně jeden systémový diagram, SD – OPD nejvyšší úrovně, OPD úrovně 0. Často obsahuje jeden hlavní, základní systémový proces, který je funkcí systému poskytující hodnotu. Rekursivní proces nového diagramu s iteracemi přiblížení vede k sadě OPD, které jsou uspořádány do (hierarchické) stromové struktury, přičemž SD je kořenem (úroveň detailu 0) stromu OPD, SD1, SD2 atd. jsou na úrovni detailu 1 hierarchie OPD, SD1.1, SD1.2, ... SD 2.1, SD2.2... jsou na úrovni detailu 2 hierarchie OPD atd.

Strom OPD je orientovaný stromový graf, jehož uzly jsou OPD získané rekursivním zjemňováním (zoomováním a/nebo rozbalováním) procesů v systému, počínaje funkcí – procesem v SD.

Sada OPD je sada všech uzlů ve stromu OPD.

¹ Červená kontura je automaticky přiřazena OPCAT k věci, která je jak přiblížená, tak rozvinutá.

Úroveň detailu OPD je počet uzlů ve stromu OPD, které je třeba projít od daného OPD ke kořenu SD, včetně samotného SD.

Strom OPD je strom procesů – graf, jehož uzly jsou OPD. Kořenem je SD, systémový diagram, a ostatní uzly jsou potomci OPD, označení svými štítky OPD, jako například SD1, který je na úrovni detailu 1, SD2.3, který je na úrovni detailu 2, atd. Směrové hrany stromu OPD mají štítky, přičemž každá hrana směřuje od rodičovského OPD, který obsahuje zpřesnitelný prvek, k potomkovému OPD obsahujícímu zpřesnění, které rozpracovává proces v rodičovském OPD prostřednictvím nového diagramu přiblížení pro synchronní podprocesy nebo nového diagramu agregace rozvinutí pro asynchronní podprocesy.

Vzhledem k tomu, že přiblížení má sémantiku agregace-účasti, je každé přiblížení v hierarchii také interpretováno jako agregace-účasť, aby byla zachována stromová struktura. **Obrázek 21.10** ukazuje v horní části strom OPD – hierarchii modelu OPM systému **Product Lifecycle Engineering** (Dori a Shpitalni 2005). Sada OPD modelu na **obr. 21.10** má 11 OPD rozložených do 4 úrovní detailů.

Zatímco strom OPD je prezentován jako hierarchie souborů (viz horní část **obr. 21.10**), systémová mapa, zobrazená ve spodní části **obr. 21.10**, je propracovanější prezentací stromu OPD.

Systémová mapa je propracovaný strom OPD, ve kterém každý uzel ve stromu představuje miniaturizovanou ikonu OPD, s tlustými šedými šipkami směřujícími od každého procesu v jednom OPD k jeho upřesnění (zvětšené nebo rozbalené) verzi v podřízeném OPD.

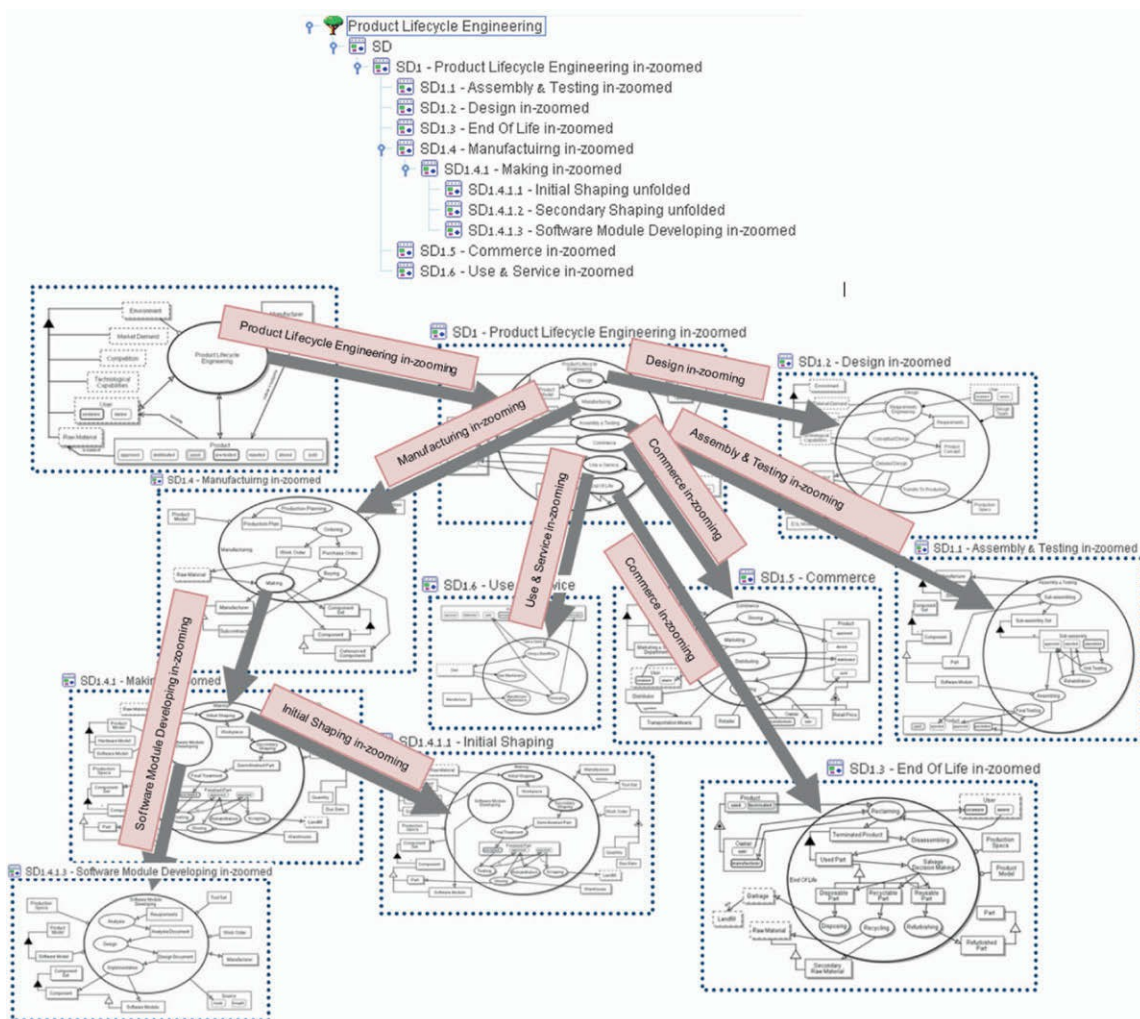
Systémová mapa explicitně zobrazuje prvky (věci a odkazy) v každém OPD (uzlu). Protože systémová mapa může být velmi rozsáhlá a nepraktická, mechanismy musí umožňovat přístup k obsahu modelu a asociacím mezi prvky. Systémová mapa pomáhá orientovat se v komplexním systému, který může zahrnovat stovky OPD na mnoha úrovních detailů. Jako příklad lze uvést **spustitelný** model OPM modelu rozpadu mRNA v **Somekh et al. (2014)**, který obsahuje stovky objektů a procesů ve více než 40 OPD na 9 úrovních detailů, s hypertextovými odkazy z prvku v modelu na článek, ze kterého byl model extrahován.

Obrázek 21.11 je snímek obrazovky simulovaného spuštění modelu **mRNA Decay OPM** (**Somekh et al. 2014**), který ukazuje, že se nachází v OPD **SD2.4.2.2.1.2.4.2 – eIF4F Dissociates Cap and Decaysome** v přiblížení, jak je také naznačeno rámečkem kolem tohoto procesu ve stromu OPD vlevo. Tento OPD demonstruje samopodobnost OPD: bez ohledu na to, na jaké úrovni detailu se OPD nachází, používá pouze stavové objekty, procesy a vztahy mezi nimi.

V současné době systém na **obr. 21.11** provádí paralelně čtyři podprocesy (tmavě modré), poté co dokončil podproces **eIF4F Dissociates Cap** nad nimi. Dissociace se projevuje v každém z těchto čtyř podprocesů spotřebováním odkazu, modelovaného jako samostatný objekt, mezi dvěma objekty, např. faktor **Xrn1** a protein **eIF4E** ve spodní části jsou disociovány procesem **eIF4E a Xrn1 Dissociation**. Pod OPD je diagram životnosti, který umožňuje kontrolu každého objektu a procesu v každém časovém bodě. Prohlížeč vlevo je otevřen na **příslušném článku**, jednom ze 43 článků, ze kterých byly převzaty modelové skutečnosti v tomto OPD, získané kliknutím na přiblížený proces.

Tento příklad demonstruje nepostradatelnost mechanismů zpřesňování, zejména přiblížení. Bez něj by nebylo možné srozumitelně zobrazit stovky věcí v modelu a tisíce odkazů mezi nimi v jediném OPD nebo v jakémkoli jiném druhu diagramu.

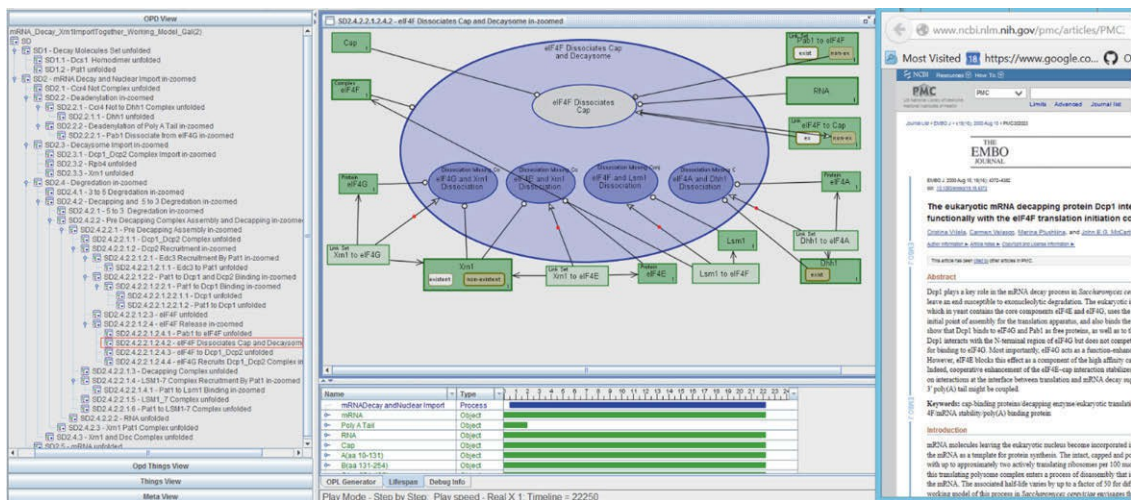
Kromě toho by sada nástrojů OPM měla poskytovat mechanismus pro vytváření pohledů, jako jsou OPD s přidruženými větami OPL, na objekty a procesy, které splňují specifická kritéria. Tyto pohledy mohou zahrnovat kritickou cestu pro minimální dobu provedení systému, seznam systémových agentů a nástrojů nebo OPD objektů a procesů zapojených do specifického druhu propojení nebo sady propojení. Například OPD lze vytvořit (1) zpřesněním (rozbalením nebo přiblížením) objektu nebo (2) shromážděním a prezentací v novém OPD věcí, které se objevují v různých OPD pro vyjádření přiřazení podfunkcí systému k objektům systémového modulu.



Obr. 21.10 Stromová hierarchie (nahore) a systémová mapa (dole) modelu OPM systému **Product Lifecycle Engineering**

Konečný OPD je jediné ploché znázornění modelu systému OPM.

Konečný OPD se získá rekurzivním zploštěním stromu OPD zdola nahoru až po vrchol stromu OPD, takže celý model je reprezentován v tomto jediném OPD. S výjimkou velmi malých systémových modelů je konečný OPD pro použití lidmi rozhodně nevhodný kvůli našim omezeným kognitivním schopnostem. Pro počítačové zpracování – správu znalostí, navigaci, dotazování atd. – je však konečný OPD velmi užitečný.



Obr. 21.11 Snímek obrazovky simulovaného provedení modelu mRNA Decay OPM (Somekh et al. 2014) zobrazující úroveň detailů 8—SD2.4.2.2.1.2.4.2—eIF4F Dissociates Cap and Decaysome v přiblížení

21.13 Strom objektů OPD a les

Na rozdíl od stromu OPD (procesu), který je výsledkem zjemnění procesu a má jediný kořen, může existovat mnoho stromů objektů OPD, alespoň jeden z každého zjemnitelného objektu, které společně tvoří les.

Strom objektů OPD je strom, jehož kořenem je objekt B a jehož uzly jsou výsledky rekurzivního zdokonalování B prostřednictvím rozbalení a přiblížení, přičemž každé přiblížení je převedeno na agregaci-účast.

Každý strom vychází z odlišného rafinovatelného objektu, který se rozbálí nebo přiblíží, aby odhalil své detaily – nemusí se nutně jednat pouze o části, jako je tomu v procesu přiblížení, ale také o funkce, specializace nebo instance. Namísto identifikace možného toku řízení provádění, jako je tomu v stromu OPD (proces),

každý strom objektů OPD zapouzdřuje informace o objektu jako hierarchickou strukturu. Vzhledem k tomu, že přiblížení má sémantiku agregace-účasti, podobně jako strom OPD, je každé přiblížení v hierarchii procesu OPD také interpretováno jako agregace-účást, aby byla zachována struktura stromu. Kompletní nebo částečné stromy objektů OPD mohou být prezentovány jako pohledy (viz oddíl 21.18). Kořen každého stromu objektů OPD lze připojit jako podřízený uzel ve stromu OPD (procesu), čímž se vytvoří mapa systému (viz oddíl 21.12).

21.14 Zmenšení

Zmenšení je opačná operace než zvětšení. Scénář, ve kterém vzniká potřeba zmenšení, nastává, když modelář zjistí, že aktuální OPD je již přeplněný, což vyžaduje skrytí obsahu zvětšeného procesu v aktuálním OPD. Zmenšení v diagramu nevytváří nový OPD, což znamená odstranění a ztrátu podprocesů a objektů uvnitř procesu, který je zmenšován. Proto, pokud modelář nerozhodne, že tyto podprocesy jsou pro daný účel příliš podrobné a je připraven je smazat, nemá oddálení v diagramu velký smysl.

Zvětšení nového diagramu rozpracovává vylepšitelné prvky ve stávajícím OPD, řekněme **SD_n**, kde **n** je aktuální úroveň detailu, vytvořením nového OPD, **SD_{n+1}**, který rozpracovává vylepšitelné prvky na další úrovni detailu přidáním podprocesů, souvisejících objektů a relevantních odkazů. Obrázek 21.12 je metamodel procesu **nového diagramu se zvětšením** a **nového diagramu se zmenšením**. Model OPM na pravé straně používá zvětšení diagramu v diagramu modelu na levé straně k rozpracování dvou procesů: **zvětšení nového diagramu** pro vytvoření kontextu nového diagramu se zvětšeným zobrazením, vyplněného podprocesy a objekty, a **zmenšení nového diagramu** pro vytvoření kontextu nového diagramu se zmenšeným zobrazením (prázdného). **Zvětšování nového diagramu** začíná **zobrazením obsahu**, po kterém následuje **zpřesnění odkazů**. **Zmenšování nového diagramu** začíná **abstrahováním odkazů**, což je opačný proces než **zpřesnění odkazů**, po kterém následuje **skrytí obsahu**, což je opačný proces než **zobrazení obsahu**.

Semi-Zoomed OPD je dočasný objekt, který je vytvořen a následně spotřebován během **New Diagram In-Zooming** i **New-Diagram Out-Zooming**. Tento dočasný objekt se objevuje pouze v kontextu **New-Diagram In-Zooming** a **New-Diagram Out-Zooming**.

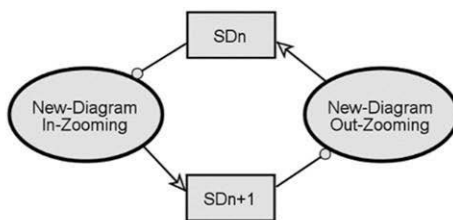
Na obr. 21.13 je metamodel na levé straně obr. 21.12 rozpracován vložením skutečných OPD do jeho objektů **SD_n**, **SD_{n+1}** a **Semi-Zoomed OPD**. V tomto konkrétním příkladu modelu OPM zahrnuje **SD_n**, zobrazený na obr. 21.13 v horní střední části, proces **P**, který je zdokonalitelný a má být zvětšen, stejně jako čtyři objekty: spotřebitel **C**, agent **A**, nástroj **D** a výsledek **B**, propojené s **P** pomocí odpovídajících různých procedurálních odkazů. Tento OPD uvnitř metaobjektu **SD_n** je nástrojem pro **New-Diagram In-Zooming** na levé straně.

Zobrazení obsahu je prvním ze dvou dílčích procesů **nového diagramu v přiblížení**. Během **zobrazení obsahu** se hranice **P** rozšiřuje, aby se vytvořil prostor pro zobrazení jeho obsahu – dílčí procesy modelu **P1**, **P2** a **P3**, stejně jako prozatímní modelový objekt **BP**. Výsledek **zobrazení obsahu** je prezentován jako obsah prozatímního objektu **Semi-Zoomed OPD**. Tento prozatímní objekt je rozpoznatelný pouze v kontextu **New-Diagram In-Zooming**. Druhý dílčí proces, **Link Refining**, prováděný modelářem, jej spotřebovává při vytváření **SD_{n+1}** prezentovaného na obr. 21.13 dole uprostřed.

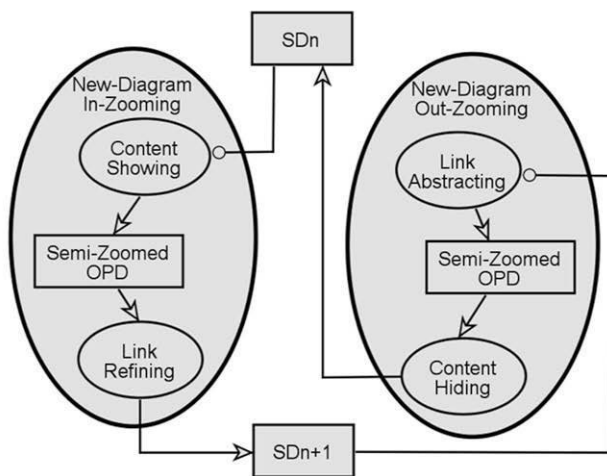
Během **zdokonalování propojení** se procedurální propojení připojená k obrysu **P** přesouvají do příslušných dílčích procesů podle rozhodnutí modeláře. Protože **P1** spotřebovává **C**, šipka propojení spotřeby

migruje z **P** do **P1**. Agent **A** zpracovává jak **P1**, tak **P2**, takže v **SDn+1** nahrazují dvě agentní spojení, jedno k **P1** a druhé k **P2**, jediné spojení v **SDn** z **A** do **P**. **P3** vyžaduje **D**, takže nástrojové spojení migruje z **P** do **P3**. Nakonec, protože **BP** je výsledkem **P1** a **P3** jej spotřebovává, jsou přidány odpovídající odkazy na výsledek a spotřebu, čímž se **BP** stává prozatímním interním objektem **P**, rozpoznatelným pouze v kontextu **P**. Stejně tak **P1**, **P2** a **P3** jsou interními procesy **P** a jako takové jsou rozpoznatelné pouze v kontextu **P**. OPD uvnitř metaobjektu **SDn+1** je nástrojem pro **New-Diagram Out-Zooming** na pravé straně. To, co se stane dál, je přesný opak toho, co jsme viděli, a to jak v pořadí podprocesů, tak v tom, co každý z nich dělá.

New-Diagram In-Zooming requires **SDn**.
New-Diagram In-Zooming yields **SDn+1**.
New-Diagram Out-Zooming requires **SDn+1**.
New-Diagram In-Zooming yields **SDn+1**.



New-Diagram In-Zooming zooms into **Content Showing** and **Link Refining**, in that sequence, as well as **Semi-Zoomed OPD**.
Content Showing requires **SDn**.
Content Showing yields **Semi-Zoomed OPD**.
Link Refining consumes **Semi-Zoomed OPD**.
Link Refining yields **SDn+1**.
New-Diagram Out-Zooming zooms into **Link Abstracting** and **Content Hiding**, in that sequence, as well as **Semi-Zoomed OPD**.
Link Abstracting requires **SDn+1**.
Link Abstracting yields **Semi-Zoomed OPD**.
Content Hiding consumes **Semi-Zoomed OPD**.
Content Hiding yields **SDn**.



Obr. 21.12 Metamodel nového diagramu in-zooming a nového diagramu out-zooming

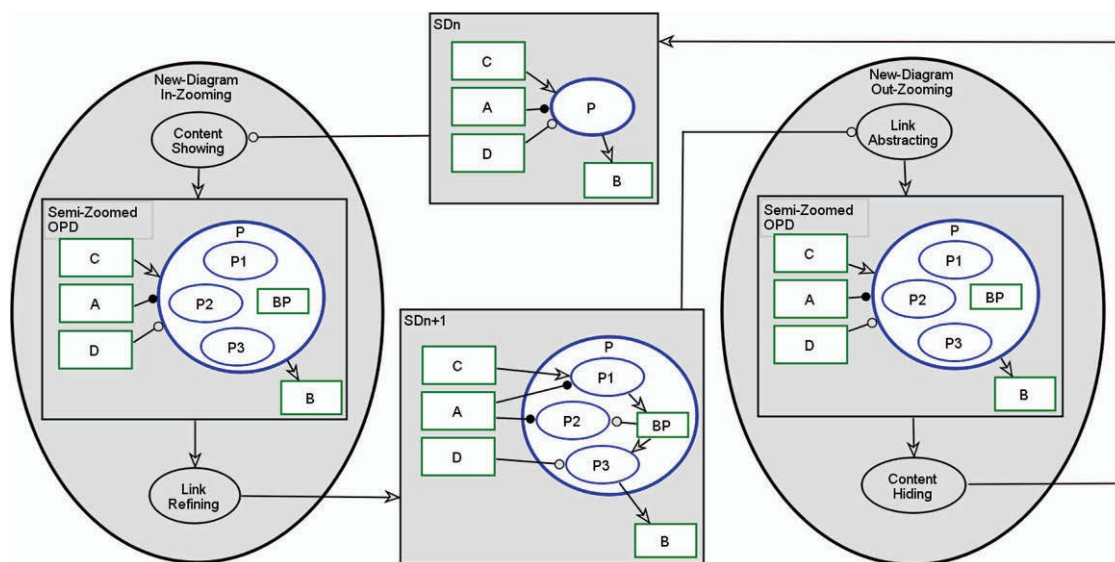
Abstrakce odkazů je prvním ze dvou dílčích procesů **nového diagramu s oddálením**. Během **abstrakce odkazů** se odkazy spojené s dílčími procesy a prozatímními objekty **P** přesouvají na (hranici, obvod elipsy) samotného **P**, což vede k přesně stejnému **částečně oddálenému OPD**, který je znázorněn uvnitř **nového diagramu s přiblížením**. Tento dočasný objekt **Semi-Zoomed OPD** je spotřebován funkcí **Content Hiding**, čímž vzniká **SDn** zobrazený na obr. 21.13 nahoře uprostřed. Hranice **P** se nyní může zmenšit, protože je prázdná a není třeba vytvářet prostor pro zobrazení jejího obsahu (modelové dílčí procesy **P1**, **P2** a **P3**

stejně jako prozatímní modelový objekt **BP**), který je nyní skrytý. Výsledek **Content Showing** je prezentován jako obsah prozatímního objektu **Semi-Zoomed OPD**.

21.15 Zjednodušení OPD

Zmenšení v diagramu – odstranění zvětšeného obsahu procesu – následované zvětšením nového diagramu může zjednodušit již namodelovaný OPD, který modelář považuje za příliš komplikovaný nebo přetížený detaily. Zmenšení diagramu snižuje kognitivní zátěž potřebnou k pochopení komplikovaného OPD na úkor přidání nového OPD do sady OPD, což je výsledek následného zvětšení nového diagramu, které vytvoří nový OPD na prozatímní úrovni detailů, jak je vysvětleno dále.

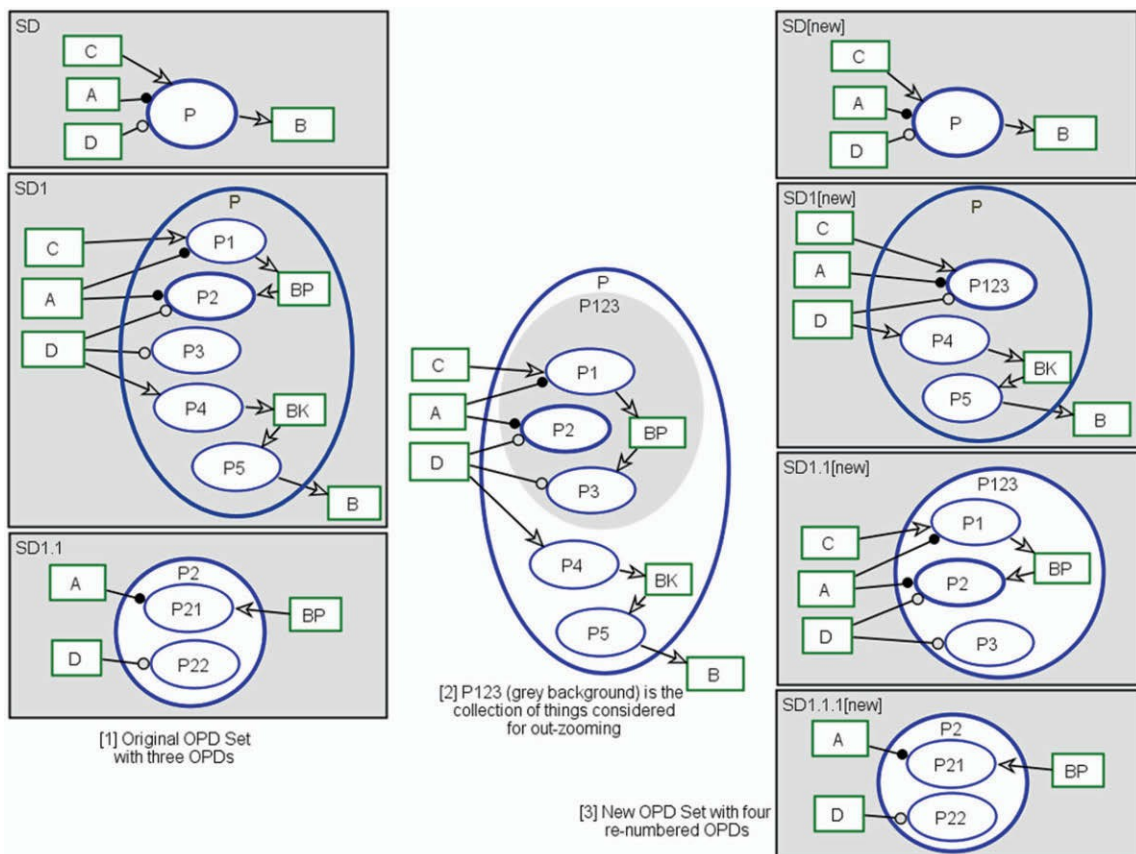
Obrázek 21.14 ukazuje zjednodušení OPD pomocí oddálení v diagramu následované oddálením nového diagramu. Vlevo je původní sada OPD se třemi OPD: **SD**, **SD1** a **SD1.1**. Modelář si uvědomuje, že **SD1** je příliš komplikovaný, a proto se rozhodne, že sada TO (věci, které mají být oddáleny), zahrnující čtyři věci v **SD1** – **P1**, **P2** a **P3** spolu s **BP** – bude nahrazena jediným novým procesem **P123** pomocí oddálení nového diagramu.



Obr. 21.13 Metamodel vlevo na obr. 21.12 rozpracovaný s příkladem skutečného modelu OPM uvnitř něj

Ve středu obr. 21.14 prochází **P123** novým diagramem s oddálením, což vede k **SD1.1[nový]** (v reálné implementaci se nové OPD neoznačují jako [nové]; tento štítek slouží pouze pro vysvětlení).

Zde je popsáno, jak se to provádí. Modelář označí prvky v sadě TO (prvky, které mají být oddáleny) a název nového prozatímního procesu, který má být vytvořen (v našem případě **P123**). Šedé pozadí označuje tyto kandidátské prvky. Budoucí proces **P123** nyní prochází oddálením nového diagramu podle dvou podprocesů popsaných výše: abstrahování odkazů a skrytí obsahu. V důsledku abstrahování odkazů se odkazy, které byly připojeny k dílčím procesům budoucího procesu **P123**, přesunuly do obrysu nově vytvořeného **P123** a v důsledku skrytí obsahu se **P123** stane prázdným, jak je znázorněno v **SD1[nový]**.



Obr. 21.14 Zjednodušení SD1 modelu OPM vlevo pomocí oddálení diagramu a následného přiblížení nového diagramu vede k novému modelu OPM vlevo, ve kterém **SD1[nový]** a **SD1.1[nový]** nahrazují **SD1**.

Aby byly zachovány odstraněné modelové skutečnosti (například skutečnosti, že **A** je agentem **P1** a **P2**), byl vytvořen nový OPD, **SD1.1[nový]**, obsahující tyto skutečnosti. Vpravo na obr. 21.14 je tedy nová sada OPD, která nyní obsahuje čtyři OPD: **SD[nový]**, **SD1[nový]**, **SD1.1[nový]** a **SD1.1.1[nový]**, přechíslované tak, aby odrážely novou hierarchii OPD. V této rozšířené hierarchii byl složitý OPD **SD1** nahrazen dvěma jednoduššími OPD – **SD1[nový]** a **SD1.1[nový]**.

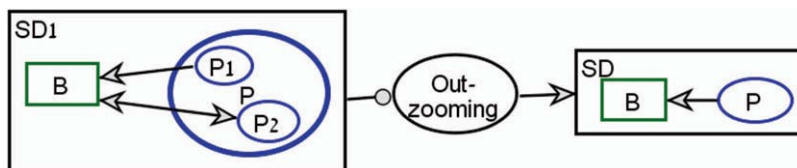
Při zkoumání **SD1**[nový] vidíme, že je skutečně méně složitý a méně přeplněný než původní **SD1**, protože má o pět prvků méně: tři odstraněné procesy, **P1**, **P2** a **P3**, jeden odstraněný objekt, **BP**, dvě odstraněné vazby a jeden přidaný proces, **P123**. Tento nový OPD je vložen do hierarchie procesů a posouvá starý **SD1.1**, který zůstává beze změny, o jednu úroveň detailu níže, z úrovně detailu 2 na úroveň detailu 3. V důsledku přidání **SD1.1**[nový] je SD1.1 přechíslován na **SD1.1.1**[nový].

21.16 Abstrakce zohledňuje prioritu procedurálních odkazů

Připomeňme, že *princip jedinečnosti procedurálních odkazů OPM* tvrdí, že na jakékoli úrovni detailu může být objekt a proces propojen maximálně jedním procedurálním odkazem, který jednoznačně určuje roli objektu ve vztahu k procesu na dané úrovni detailu.

Když modelář provádí abstrakci pomocí potlačení stavu, složení nebo oddálení, procedurální vazby mezi rafinovanými prvky a jinými prvky v OPD, které nejsou rafinované, se přesouvají do kontextu (graficky kontura nebo obvod) rafinovatelného prvku. Například při potlačení stavů na [obr. 10.4](#) se dvojice vstupně-výstupních vazeb přesouvá ze dvou stavů do **osoby** a stává se vazbou účinku. Dalším příkladem je **P123** na [obr. 21.14](#).

Tento přesun může způsobit konflikt, ve kterém dva nebo více procedurálních odkazů různých druhů propojuje objekt a proces. Podle principu jedinečnosti procedurálních odkazů OPM může objekt nebo stav objektu propojit s procesem pouze jediný, jedinečný procedurální odkaz. [Obrázek 21.15](#) demonstruje problém abstrakce procedurálních odkazů. V **SD1** je výsledný odkaz z **P1** do B významnější nebo sémanticky silnější než efektový odkaz z **P2** do B, takže když je proces P v **SD1** v **SD** oddálen, převládá výsledný odkaz.



Obr. 21.15 Abstrahování různých procedurálních odkazů vyvolává prioritu odkazů

Aby byl tento princip zachován, OPM řeší konflikt mezi kandidátskými odkazy tak, že na základě sémantické síly odkazů určí, který odkaz zůstane a který nový odkaz nahradí kandidáty v abstraktním OPD. Ztráta detailních informací je v souladu s pojmem abstrakce. Sémantická síla a priorita odkazu jsou dva pojmy, které slouží jako vodítko při určování, které odkazy zachovat a které skrýt, když je OPD oddáleno nebo složeno.

Sémantická síla procedurálního odkazu je význam informací, které odkaz nese.

Informace týkající se změny existence, ať už vytvoření nebo odstranění, jsou významnější než informace o změně existující věci. Relativní sémantická síla dvou konfliktních procedurálních odkazů určuje prioritu odkazu. Když dva nebo více procedurálních odkazů soutěží o to, aby zůstaly

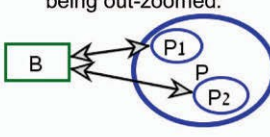
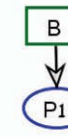
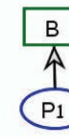
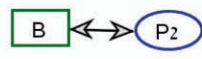
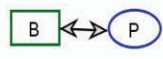
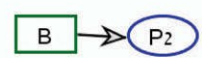
zastoupené v OPD, které je abstrahováno (zoomováno, složeno nebo potlačeno), převládá odkaz s nejvyšší sémantickou silou.

21.16.1 Priorita mezi transformujícími se odkazy

Transformační odkazy zahrnují odkazy na výsledek, účinek a spotřebu a jejich varianty s modifikátory události nebo podmínky.

Priorita odkazů je seřazený seznam procedurálních odkazů s klesající sémantickou silou.

Tabulka 21.1 Přednost odkazů mezi transformujícími odkazy

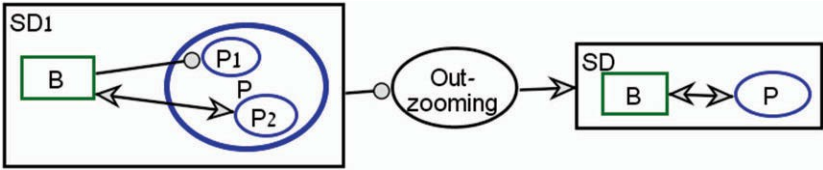
The in-zoomed process P being out-zoomed: 			
		Invalid	
		Invalid	
	Invalid		Invalid

Tabulka 21.1 ukazuje prioritu odkazů mezi transformujícími odkazy: **P** v levém horním rohu je oddáleno. Záhloví sloupců zobrazují tři možné transformační odkazy mezi **P1** a B, zatímco záhlaví řádků zobrazují tři možné odkazy mezi **P2** a B. Buňky tabulky zobrazují převládající odkaz mezi B a P po oddálení P. Buňky označené jako „Neplatné“ označují nemožnost kombinace. Například při kontrole střední buňky, pokud **P1** spotřebuje B, pak B již neexistuje, když se **P2** později pokusí jej znovu spotřebovat. Vzhledem k tomu, že vytvoření a spotřeba objektu jsou sémanticky silnější (tj. mají vyšší sémantickou sílu) než ovlivnění objektu změnou jeho stavu, mají odkazy výsledku a spotřeby přednost před odkazy účinku, jak je znázorněno v tabulce 21.1. Jelikož však odkazy výsledku a spotřeby jsou sémanticky ekvivalentní, v případě jejich konkurence bude převládajícím odkazem odkaz účinku, protože odkaz účinku umožňuje jak vytvoření, tak eliminaci jako účinky.

21.16.2 Přednost mezi transformačními a umožňujícími vazbami

Transformační vazby jsou sémanticky silnější než umožňující vazby, protože transformační vazby označují vytvoření, spotřebu nebo změnu propojeného objektu, zatímco umožňující vazby označují pouze umožnění. Transformační vazba má proto přednost před umožňující vazbou, jak je znázorněno na obr. 21.16.

V rámci povolovacích odkazů má odkaz agenta přednost před odkazem nástroje, protože v umělých systémech jsou lidé ústředním prvkem procesu, ovládají systém a musí zajistit jeho správné fungování. Kromě toho by všude, kde dochází k interakci s člověkem, mělo existovat rozhraní a tyto informace by měly být k dispozici modeláři, aby mohl navrhnout rozhraní člověk-systém podle specifikace koncepčního modelu.



Obr. 21.16 Přednost spojení mezi transformačními a umožňujícími spojeními

Shrneme-li sémantickou sílu procedurálních nekontrolních vazeb, primární prioritě vazeb je následující:

Spotřeba= Výsledek> Účinek> Agent> Nástroj
--

Zde symboly = a > odkazují na sémantickou sílu odkazů. Odkazy specifikované stavem mají vyšší prioritu než základní odkazy, které stavy nespecifikují.

21.16.3 Priorita mezi odkazy stejného druhu bez ovládání a odkazy s ovládáním

Každý druh nekontrolního odkazu má odpovídající odkaz na událost a podmínku, které jsou užitečné pro určení jemnějšího, sekundárního rozlišení priority v rámci každého druhu procedurálního odkazu. Sekundární priorita odkazu existuje v rámci každého procedurálního odkazu v primární prioritě odkazu. Odkaz na událost má vyšší sémantickou sílu než jeho odpovídající nekontrolní odkaz, zatímco odkaz na podmínku má slabší sémantickou sílu než jeho odpovídající nekontrolní odkaz. Sémantická síla odkazů na události je silnější než sémantická síla odpovídajících odkazů bez kontroly, protože každý odkaz na událost má sémantiku jak odpovídajícího odkazu bez kontroly, tak události schopné iniciovat proces. Sémantická síla podmíněných odkazů je slabší než sémantická síla odpovídajících odkazů bez kontroly, protože modifikátor podmínky oslabuje kritéria splnění předběžných podmínek pro propojovací proces.

21.16.4 Shrnutí priority procedurálních odkazů

Shrneme-li sémantickou sílu procedurálních odkazů na základě rozlišení mezi primární a sekundární prioritou, úplné pořadí priorit je následující:

1.	událost spotřeby	>	spotřeba
2.	spotřeba	=	výsledek
3.	výsledek	>	spotřeba stav
4.	podmínka spotřeby	>	účinek událost
5.	účinek události	>	účinek
6.	efekt	>	podmínka účinku
7.	podmínka účinku	>	agent událost
8.	událost agenta	>	agent
9.	agent	>	agent podmínka

10. stav agenta	>	událost nástroje
11. událost nástroje	>	nástroj stav nástroje
12. přístroj	>	

21.17 Migrace odkazu při přiblížení

Kontext (graficky vnější obvod) procesu **P** funguje jako závorky v algebře, které se používají k vyjádření distributivního zákona: Jakýkoli procedurální odkaz připojený k **P** je tedy považován za připojený ke každému z podprocesů **P**. Příklad je uveden na [obr. 8.2](#), kde **havarované vozidlo** je nástrojem všech čtyř podprocesů v rámci **automatické reakce na havárii**.

Když modelář přidává podprocesy, často se mu nepodaří ručně migrovat procedurální odkazy do konkrétních podprocesů, což vede k jejich implicitnímu připojení k nadbytečným procedurálním odkazům, které model zneplatňují. Aby se těmto situacím předešlo, měl by modelovací nástroj, jakmile modelář nakreslí první podproces **P1** uvnitř a přiblíženého procesu **P**, automaticky přesunout do **P1** všechny procedurální a kontrolní odkazy, které byly připojeny k **P** v nadřazeném OPD. Příkladem je [obr. 5.1](#), který ukazuje proces **automatické reakce na nehodu** po jeho zvětšení a po nakreslení jeho prvního podprocesu, **měření závažnosti nehody**, uvnitř něj v horní části obklopující elipsy procesu **automatické reakce na nehodu**. Odkazy, které byly připojeny k **automatické reakci na nehodu**, byly přesunuty a připojeny k **měření závažnosti nehody**.

Úkolem modeláře je zajistit, aby různé transformační odkazy, které jsou nyní připojeny k **P1**, byly vráceny zpět k **P** nebo přesunuty do následujících dílčích procesů. Podobně může být nutné přesunout aktivační odkazy do jednoho nebo více konkrétních dílčích procesů, kde je propojený aktivátor skutečně potřebný. Jako alternativa k automatickému přesunu odkazů může nástroj po vložení každého nového dílčího procesu zkontrolovat platnost odkazů a v případě potřeby upozornit modeláře.

21.18 Vytváření pohledů: čtvrtý mechanismus zdokonalování

Vytváření pohledu – čtvrtý mechanismus zdokonalení po vyjádření stavu, přiblížení a rozbalení – se dosahuje shromážděním modelových faktů z různých OPD v sadě OPD a jejich sloučením do nového OPD nazvaného Pohled za účelem demonstrace konkrétního aspektu. Příklady zahrnují (1) procesní strom – úplný nebo částečný strom procesní hierarchie systému, který je čistě procedurálním pohledem na systém, (2) objektový strom – úplný nebo částečný strom objektové hierarchie systému, který je čistě strukturálním pohledem na systém, (3) pohled na alokaci, který ukazuje, jaké objekty jsou alokovány k provádění jakých funkcí (procesů) v modelu systému, a (4) animovaný pohled motivovaný simulací, jehož cílem je usnadnit souběžnou kontrolu toho, jak určité objekty a procesy z různých OPD interagují. V modelovacím nástroji se pohledy nesmějí upravovat za účelem přidání, odstranění nebo změny jakéhokoli modelu. To by se mělo provádět v OPD mimo pohled a automaticky se to mělo promítnout do příslušných pohledů. Opakem vytváření pohledů je mazání pohledů.

21.19 Middle-Out jako de facto architektonická praxe

V ideálním případě začíná analýza a návrh shora a postupně se přesouvá dolů – od obecného k detailnímu. V reálném životě však analýza obvykle začíná na nějaké libovolné úrovni detailů a málokdy je lineární. Návrh také není lineární. Obvykle se jedná o iterativní procesy, během nichž se postupně hromadí a zdokonaluje znalost, následovaná porozuměním. Architekt systému nemůže předem znát přesnou strukturu a chování nejvyšší úrovně systému – to vyžaduje analýzu a stává se zřejmým v určitém bodě procesu analýzy. Krok za krokem analytik vytváří specifikaci systému hromaděním a zaznamenáváním faktů a pozorování o věcech v systému a vztazích mezi nimi.

Vzhledem k nelineární povaze procesů analýzy a návrhu jsou lineární, jednosměrné přístupy „zdola nahoru“ nebo „shora dolů“, které se zdají být velmi metodické, zřídka použitelné na reálné systémy. Často se spíše stává, že systém, který se buduje nebo zkoumá, je tak složitý a neprozkoumaný, že ani jeho vrchol, ani jeho spodní část nejsou od počátku s jistotou známy. Analýza a návrh reálných systémů častěji začínají na neznámém místě v hierarchii úrovní detailů systému. Analýza pokračuje „uprostřed“ kombinací technik shora dolů a zdola nahoru, aby bylo dosaženo úplného pochopení a specifikace systému na všech úrovních detailů.

Ukázalo se tedy, že i když se architekti obvykle snaží pracovat systematicky shora dolů, ve skutečnosti se častěji používá středový způsob analýzy a návrhu. Namísto boje proti tomu musí přístupy a nástroje pro modelování systémů poskytovat prostředky pro zpracování tohoto středového způsobu architektury spolu s podporou přístupů shora dolů a zdola nahoru.

21.19.1 OPM vyhovuje smíšenému přístupu

Pomocí OPM jsou nashromážděné znalosti dokumentovány a reprezentovány jako propojené modelové skutečnosti prostřednictvím sady OPD a odpovídajících odstavců OPL. Pokud se OPD, které je rozšiřováno, stane příliš přeplněným, rušným nebo nesrozumitelným, je vytvořeno nové OPD. Toto potomkové OPD opakuje jednu nebo více věcí ze svého předka OPD v rafinovanější formě. Tyto opakované věci vytvářejí spojení mezi předkem a potomkem OPD. Potomek OPD obvykle nereplikuje všechny detaily svého předka, protože některé z nich jsou abstrahovány, zatímco jiné nejsou jednoduše zahrnuty. Tento nový OPD je proto přístupný zdokonalení nových věcí, které mají být rozloženy v prostoru, který byl ušetřen tím, že nebyly zahrnuty věci z předka OPD. Jinými slovy, je v něm prostor pro vložení určitého množství dalších detailů, než se stane příliš přeplněným. V takovém případě nastává nový cyklus zdokonalování, který pokračuje, dokud není celý systém zcela specifikován. Jak jsme viděli v této kapitole, OPM vyhovuje nejen tomuto top-down přístupu, ale také bottom-up a middle-out přístupům prostřednictvím abstrahování a zjednodušování OPD spolu s přidáním prozatímní úrovně detailů.

21.19.2 Kdy by měl být vytvořen nový OPD?

Sada OPD musí být čitelná, snadno srozumitelná a pochopitelná. Následující pravidla jsou užitečná při rozhodování o tom, kdy by měla být vytvořena nová OPD, aby byly OPD co nejčitelnější a nejsrozumitelnější.

- OPD by neměl přesahovat jednu stránku nebo jednu obrazovku monitoru průměrné velikosti.
- OPD by neměl obsahovat více než 20–25 entit (objektů, procesů nebo stavů).

- Věci (objekty nebo procesy) se nesmí vzájemně překrývat. Buď jsou zcela obsaženy ve věcech vyšší úrovně, v případě přiblížení, nebo nemají žádné překrývající se oblasti. Výjimkou z tohoto pravidla je použití port folding (viz oddíl 21.8).
- Diagram by neměl obsahovat příliš mnoho odkazů.
- Odkaz by neměl překračovat oblast obsazenou věcí.
- Počet překrývajících se odkazů by měl být minimalizován.

21.20 Navigace v rámci modelu systému OPM

Jak jsme viděli, model OPM může být velmi rozsáhlý, takže navigace uvnitř modelu a orientace se stávají problémem.

21.20.1 Popisky diagramu OPM a popisky okrajů stromu

Název systému OPM je název modelu OPM, který specifikuje systém. Název OPD je název, který identifikuje každý OPD v procesu OPD. SD musí obsahovat jeden a pouze jeden systémový proces, který představuje zastřešující systémovou funkci, která poskytuje funkční hodnotu zainteresovaným stranám. Kromě toho může obsahovat jeden nebo více environmentálních procesů. SD je označení kořenového OPD v procesu OPD. Kořen stromu OPD, SD, zaujímá úroveň (vrstvu) 0 ve stromu OPD a je jediným uzlem na této úrovni. Vyšší číslované vrstvy, tj. ty, které odpovídají postupným zdokonalováním, mohou mít více než jeden OPD.

Nejen uzly ve stromu OPD jsou označeny, ale také hrany. Každá hrana (oblouk spojující dva uzly – dvě OPD) ve stromu OPD má jedinečné označení. Označení vyjadřuje vztah zpřesnění, který odpovídá implicitnímu odkazu na vyvolání nebo vztahu rozvinutí. Pokud považujeme každý OPD za objekt a celý procesní strom OPD za jeden OPD, každá hrana je jednosměrná šitkovaná strukturální vazba se štítkem, který zní: „je zpřesněn přiblížením **<Refineable Name>**; v “, nebo „je zpřesněn rozvinutím

<Refineable Name>; in “. Věta OPL pro zpřesnění OPD je věta OPL popisující vztah zpřesnění mezi zpřesnitelným prvkem přítomným v OPD vrstvy N a jeho zpřesňujícím OPD ve vrstvě $N+1$. Syntaxe věty OPL pro zpřesnění OPD se zoomováním je:

<TierN OPD label>; je upřesněn pomocí přiblížení **<Refineable Process Name>**; v **<TierN+1 OPD Label>**;

Podobně syntaxe rozbalené věty OPL o zpřesnění OPD je:

<TierN OPD label>; je upřesněn rozbalením **<Refineable Process Name>**; v **<TierN+1 OPD Label>**;

21.20.2 Specifikace OPL celého systému

Odstavec OPL je soubor vět OPL, které společně specifikují v textu to, co odpovídající OPD specifikuje graficky. Název odstavce OPL, používající název OPD, může předcházet první větě OPL každého odstavce OPL.

Specifikace modelu OPD je soubor po sobě jdoucích OPD v systému OPD stromu.

Specifikace modelu OPL je soubor po sobě jdoucích odstavců OPL odpovídajících OPD ve stromu OPD systému, ze kterého byly odstraněny duplicitní větvy OPL.

Specifikace modelu OPM je souběžné zobrazení specifikace modelu OPD a odpovídajícího odstavce OPL, který je uveden napravo od každého OPD.

Příklad specifikace modelu OPM je uveden v [tabulce 21.2](#), která obsahuje celý model OPM systému mytí nádobí na [obr. 10.5](#). Specifikace modelu OPM systému začíná úvodním názvem, jako v případě *specifikace modelu OPM systému mytí nádobí*.

Levý sloupec obsahuje OPD v sadě OPD systému OPM v pořadí podle šířky, ale modelář může toto výchozí pořadí přepsat. Odpovídající odstavce OPL jsou uvedeny v pravém sloupci tak, že každý odstavec OPL je napravo od svého OPD.

21.21 Shrnutí

- **Řízení složitosti** je nezbytné pro zvládnutí složitosti reálných systémů, ať už umělých, nebo přírodních.
- **Tvrzení o složitosti modelu OPM** je, že pomocí mechanismů zjemňování a rozbalování stavových objektů nebo procesů může OPM koncepčně modelovat systémy na jakékoli úrovni složitosti.
- Přístup OPM ke správě složitosti je založen na **rozkladu na úrovni detailů**, což je v kontrastu s přístupem UML a SysML založeným na rozkladu podle aspektů.
- **Kompromis mezi úplností a srozumitelností** je napětí mezi potřebou specifikovat systém tak, aby byly zastoupeny všechny modelové skutečnosti, a zároveň zachovat jasné a srozumitelné znázornění systému.
- Tři mechanismy zpřesňování a abstrakce jsou rozbalování–sbalování, přiblížení–oddálení a vyjádření stavu–potlačení stavu. Čtvrtým je vytvoření pohledu–smazání pohledu.
- **Vyjádření stavu** znamená zobrazení jednoho nebo více stavů objektu; potlačení stavu znamená skrytí jednoho nebo více stavů objektu.
- Každá ze čtyř základních strukturálních vztahových vazeb může procházet rozvinutím a složením, takže existují čtyři druhy párů rozvinutí–složení.
- **Rozložení v diagramu** je rozložení, při kterém se vylepšitelné a jeho vylepšení objeví rozložené ve stejném OPD, ve kterém bylo původně vylepšení.
- **Rozvinutí v novém diagramu** je rozvinutí, při kterém se rafinovatelné a jeho rafináty objevují rozvinuté v novém OPD.

- **Rozvinutí** je mechanismus pro zjemnění, propracování nebo rozložení, který odhaluje sadu zjemnitelných prvků – věcí, které souvisejí s rozvinutou věcí – zjemnitelným prvkem.
- **Synchronní proces** je proces, jehož podprocesy mají předem definované, pevné pořadí.
- **Asynchronní proces** je proces, jehož podprocesy nemají předem definované, pevné pořadí.
- **Zvětšení nového diagramu** je zvětšení, při kterém se rafinovatelné a jeho rafináty objeví zvětšené v novém OPD.
- **Zvětšování v diagramu** je zvětšování, při kterém se nevytváří žádný nový OPD a vylepšitelné prvky se zobrazují zvětšené spolu se svými vylepšeními ve stejném OPD.
- Zvětšování má sémantiku *agregace-účasti* plus pozitivní řazitelnost.
- Procesní přiblížení určuje (případně částečné) *časové pořadí* provádění jeho podprocesů.
- Zvětšování objektů určuje (případně dvourozměrné) *prostorové pořadí* jeho částí.
- **Strom OPD** je orientovaný graf s označenými uzly a hranami, jehož uzly jsou OPD získané rekurzivním přiblížením nebo rozvinutím procesů v systému, počínaje funkcí – procesem v SD.
- **Sada OPD** je sada všech uzlů v OPD stromu.
- **Úroveň detailu** OPD je počet uzlů v OPD stromu, které je třeba projít od daného OPD ke kořenu, SD, včetně samotného SD.
- **Systémová mapa** je propracovaný strom OPD, ve kterém je každý uzel ve stromu miniaturizovanou ikonou OPD, s tlustými šedými šipkami směřujícími od každého procesu v jednom OPD k jeho upřesněné (přiblížené nebo rozvinuté) verzi v podřízeném OPD.
- **Konečný OPD** je jednotné ploché znázornění modelu systému OPM.

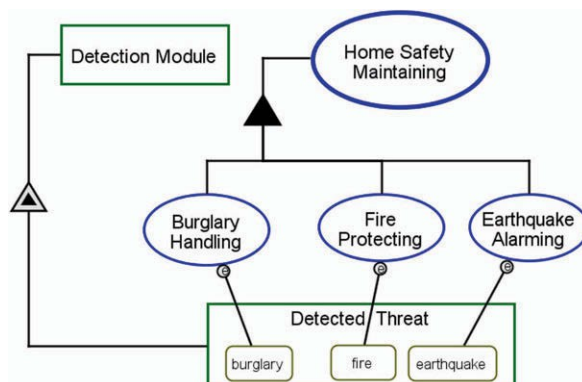
Tabulka 21.2 Specifikace modelu OPM systému mytí nádobí

Dish Washing System OPM model specification	
Graphical specification (OPD set)	Textual specification (OPL paragraph set)
SD: Dish Washing System	
	Household User handles Dish Washing. Dish Washing requires Dishwasher. Dish Washing consumes Soap. Dish Washing affects Dish Set.
SD1: Dish Washing in-zoomed	
	SD is refined by in-zooming Dish Washing in SD1. Dish Washer consists of Soap Compartment and at least one other part. Dishwasher can be empty or loaded. Dishwasher is initially empty and finally empty. Soap Compartment can be empty or loaded. Soap Compartment is initially empty and finally empty. Dish Set exhibits Cleanliness. Cleanliness of Dish Set can be dirty or clean. Cleanliness of Dish Set is initially dirty and finally clean. Household User handles Dish Washing. Dish Washing zooms into Dish Loading, Detergent Inserting, Dish Cleaning & Drying, and Dish Unloading, in that sequence. Dish Loading changes Dishwasher from empty to loaded. Detergent Inserting requires Soap. Detergent Inserting changes Soap Compartment from empty to loaded. Dish Cleaning & Drying requires Dishwasher. Dish Cleaning & Drying consumes Soap. Dish Cleaning & Drying changes Cleanliness of Dish Set from dirty to clean. Dish Unloading changes Dishwasher from loaded to empty.
End of Dish Washing System OPM model specification	

- Zoomování ven umožňuje začlenit přístup „middle-out“ do koncepčního modelování zjednodušením komplikovaného OPD a přidáním prozatímní úrovně detailů.
- **Sémantická síla** procedurálního odkazu je význam informací, které odkaz nese.
- **Priorita propojení** je seřazený seznam procedurálních propojení s klesající sémantickou silou.
- **Primární pořadí odkazů** je Spotřeba= Výsledek> Účinek> Agent> Nástroj.
- **Vytváření pohledu** spočívá ve shromažďování modelových faktů z různých OPD v sadě OPD a jejich spojování do nového OPD nazývaného *Pohled* za účelem demonstrace konkrétního aspektu.
- **Specifikace modelu OPD** je soubor po sobě jdoucích OPD v stromu OPD systému.
- **Specifikace modelu OPL** je soubor po sobě jdoucích odstavců OPL odpovídajících OPD v stromu OPD systému, ze kterého byly odstraněny duplicitní věty OPL.
- **Specifikace modelu OPM** je souběžné zobrazení specifikace modelu OPD a specifikace modelu OPL, kde vpravo od každého OPD je zobrazen odpovídající odstavec OPL.

21.22 Problémy

1. Na základě [obr. 21.1](#) vytvořte model OPM, který vysvětluje dvě specializace dekompozice, jejich význam a to, který typ používá který jazyk.
2. Zobrazte objekt se čtyřmi stavy a proces, který na něj působí.
3. Potlačte stavy, které nejsou relevantní pro model v předchozí otázce, a přidejte symbol neúplného stavu.
4. Modelujte složitý objekt se třemi úrovněmi rozvinutí, včetně agregačního rozvinutí a výstavního rozvinutí.
5. Vyberte dva podprocesy z [obr. 21.6](#). U každého z nich použijte nové diagramové přiblížení a přidejte modelové prvky podle svého uvážení.
6. Proveďte oddálení od přiblížených procesů ve dvou OPD vytvořených v předchozím problému.
7. Jaká je konečná OPD systému na [obr. 21.6](#)?
8. Je proces na [obr. 21.6](#) synchronní nebo asynchronní? Vysvětlete.
9. Je proces na [obr. 21.17](#) synchronní nebo asynchronní? Vysvětlete.



Obr. 21.17 Systém pro zajištění bezpečnosti domácnosti – částečný model

10. Nakreslete zvětšenou mapu části středozápadu USA s alespoň šesti státy, kde každý stát je objektem, při zachování přibližných prostorových vztahů mezi státy.
11. Na [obr. 21.13](#) změňte OPD uvnitř **SDn** a **SDn+1** tak, aby vznikla potřeba vyvolat procedurální prioritu odkazu.
12. Pro model v předchozím problému vytvořte částečně přiblížený OPD analogický s OPD na [obr. 21.13](#).
13. Na [obr. 21.14](#) definujte TO jako {**P3**, **P4**, **P5**, **BK**}, proved'te oddálení a zobrazte výsledek. **SD**[nové], **SD1**[nové], **SD1.1**[nové] a **SD1.1.1**[nové].

Kapitola 22

OPM Operační sémantika a řídicí odkazy

Sémantika řídicího toku představuje jednotné formální zpracování sémantiky širokého spektra pojmů řídicího toku, jak se vyskytují v sekvenčních, souběžných, logických, objektově orientovaných a funkčních programovacích jazycích.

de Bakker a de Vink (1996)

Pro řízení toku provádění systému má OPM přesnou operační sémantiku založenou na paradigmatu událost-podmínka-akce a vyjádřenou úpravou procedurálních odkazů pomocí modifikátorů řízení – symbolů událostí a podmínek. To je tématem této kapitoly.

22.1 Mechanismus řízení událost-podmínka-akce

Mechanismus aktivace procesu OPM je způsob, jakým OPM nasazuje paradigma *událost-podmínka-akce* (ECA), zmíněné v Dittrich et al. (1995) pro strukturování aktivních pravidel v architektuře řízené událostmi a aktivních databázových systémech. ECA se řídí pravidlem „*Při události, pokud je splněna podmínka, pak akce*“, tj. pokud dojde k události a v okamžiku jejího výskytu je splněna související podmínka, spustí se související akce. V terminologii OPM je *akce* procesem OPM. Takové pravidlo se tradičně skládalo ze tří částí, které jsou uvedeny níže spolu s jejich interpretacemi v OPM.

- Část *události* specifikuje objekt – *spouštěč* nebo stav či hodnotu objektu, který spouští proces.
- Část *podmínky* je logický test, který, pokud je splněn nebo vyhodnocen jako pravdivý, umožňuje provedení akce; v OPM je podmínka vyhodnocena na sadě objektů předzpracování.
- Část *akce* se skládá z aktualizací nebo vyvolání lokálních dat; v OPM to znamená aktivaci procesu, který po dokončení transformuje jeden nebo více objektů.

Paradigma ECA poskytuje základ pro operační sémantiku OPM a tok řízení provádění. V okamžiku vytvoření objektu, nebo z pohledu systému v okamžiku, kdy se objekt objeví, nebo v okamžiku, kdy objekt přejde do určitého stavu, dojde k události.

Událost je okamžik, ve kterém dojde k něčemu významnému pro provádění systému.

Objekt nebo stav objektu zapojený do události může být zdrojem procedurálního odkazu. V době běhu, tj. na úrovni instance během provádění systému, spustí výskyt této události vyhodnocení

předpokladů pro každý proces, pro který je objekt zdrojem odkazu, a událost přestává existovat. Pouze v případě, že vyhodnocení prokáže splnění předpokladu, začne se proces provádět. Události mohou nastat také na konci podprocesu uvnitř přiblíženého procesu, stejně jako prostřednictvím odkazu na vyvolání a odkazu na výjimku, které se vyskytují mezi procesy. Podle paradigmatu událost-podmínka-akce má tedy zahájení výkonu procesu („akce“) dvě předpoklady: (1) iniciační událost („událost“) a (2) splnění předpokladu („podmínka“). Události a předpoklady společně určují tok řízení provádění OPM pro výkon procesu. Tok řízení provádění je důsledkem postupných sekvencí událost-podmínka-akce, které začínají spuštěním systémové funkce externí událostí a končí, když systémová funkce buď úspěšně dokončí provádění, nebo se abnormálně ukončí.

22.2 Sady objektů předběžných podmínek, předběžného zpracování a následného zpracování

Každý proces má sadu objektů předběžného zpracování s alespoň jedním objektem, případně ve specifikovaném stavu. Sada objektů předběžného zpracování procesu určuje předpoklad, který musí být splněn před zahájením výkonu daného procesu. Sada objektů předběžného zpracování může jednoduše zahrnovat existenci jednoho nebo více objektů, případně ve specifikovaných stavech, ale může být také složitá a zahrnovat složené logické výrazy používající logické operátory AND, OR a XOR. Typickými objekty v sadě objektů předzpracování jsou transformované objekty – spotřebované a/nebo ovlivněné objekty a aktivátory. Některé z těchto objektů mohou mít další ustanovení týkající se toku řízení provádění, vyjádřené jako podmínkový odkaz, který, jak je vysvětleno níže, umožňuje přeskočení procesu, pokud není splněna jeho předběžná podmínka.

Sada objektů postprocesu určuje postpodmínku procesu, kterou splňuje dokončení procesu.

Typickými objekty v sadě objektů postprocesu jsou výsledné objekty a ovlivněné objekty.

Průnik sady objektů předprocesu a sady objektů postprocesu stejného procesu zahrnuje aktivátory a ovlivněné objekty procesu. Spotřebované objekty jsou pouze členy sady objektů předprocesu, zatímco výsledné objekty jsou pouze členy sady objektů postprocesu.

Sada zapojených objektů je sjednocením sad objektů předprocesu a postprocesu. Pokud má sada zapojených objektů pouze jeden objekt, musí se jednat o transformovaný objekt, jinak není v souladu s definicí procesu OPM jako věci, která transformuje alespoň jeden objekt. Proto v kompletním modelu OPM musí být každý proces propojen s alespoň jedním transformovaným objektem a nástroj pro modelování OPM by to měl zkontrolovat jako základní součást validace modelu.

22.3 Druhy kontrolních odkazů

Jako součást paradigmatu událost-podmínka-akce, které je základem operační sémantiky OPM, vyjadřuje odkaz události, odkaz podmínky a odkaz výjimky událost, podmínku a časovou výjimku. Tyto tři druhy odkazů jsou kontrolní odkazy OPM. Kontrolní odkazy se vyskytují buď mezi objektem a procesem, nebo mezi dvěma procesy.

Událostní a podmíněné vazby neexistují samostatně. Jsou spíše modifikovanými verzemi různých procedurálních vazeb. Každá procedurální vazba z objektu nebo stavu k procesu (tj. objekt nebo stav v předprocesním stavu objektu) má odpovídající událostní vazbu a odpovídající podmíněnou vazbu.

Modifikátor řízení je jeden ze dvou písmenných symbolů **e** a **c**, který se přidává k procedurálnímu odkazu a rozšiřuje sémantiku tohoto odkazu o sémantiku události a podmínky.

Kontrolní odkaz je procedurální odkaz s přidáním kontrolního modifikátoru.

Neexistuje žádný odkaz na výslednou událost ani odkaz na výslednou podmínku, protože se jedná o odchozí procedurální odkazy související se sadou objektů postprocesu. Po dokončení procesu se bez dalších podmínek vytvoří sada objektů postprocesu. Za předpokladu, že proces skončí úspěšně, je tedy vytvoření výsledků a změna ovlivněných objektů automatická a bezpodmínečná.

22.4 Odkazy na události

Sémantika události procesu je zahájení tohoto procesu, které spouští vyhodnocení předběžné podmínky tohoto procesu.

Odkaz na událost je procedurální odkaz s modifikátorem řízení **e**, který označuje přidání sémantiky události k cílovému procesu odkazu.

Odkaz na událost specifikuje zdrojovou událost a cílový proces – proces, který je spuštěn při výskytu události. Nastání události spustí vyhodnocení předpokladu procesu. Splnění předpokladu umožňuje pokračování výkonu (provedení) procesu, čímž se proces stává aktivním. Pokud není splněn předpoklad procesu, výkon procesu se neuskuteční. Bez ohledu na to, zda je vyhodnocení úspěšné či nikoli, událost jako bod v čase je ztracena. Pokud není splněn předpoklad procesu, zahájení procesu se neuskuteční, dokud proces neaktivuje jiná událost.

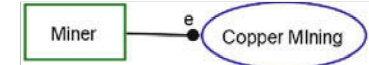
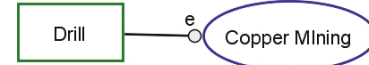
22.4.1 Zahájení ne prvního podprocesu prostřednictvím odkazu na událost

Pokud je k procesu P připojeno propojení událostí a P je přiblíženo, stejně jako všechna ostatní procesní propojení připojená k P, propojení událostí se automaticky přesune do prvního (nejvyššího) podprocesu – toho, který se provádí jako první. Modelář musí být při modelování odkazů na události připojených k jinému než prvnímu podprocesu velmi opatrný, protože se to podobá zásahu do vnitřního fungování černé skříňky! Při pokusu o spuštění podprocesu, který není prvním, nemusí být splněna jedna nebo více podmínek tohoto podprocesu, protože předchozí podprocesy byly přeskočeny. Například pokud je na [obr. 6.2](#) odkaz na událost připojen k **procesu Vytvoření zprávy** namísto k **procesu Měření závažnosti nehody**, je druhý proces přeskočen, takže **závažnost nehody** zůstane nulová, a proto bude přeskočen i proces **Vytvoření zprávy**. Navíc, protože neexistuje žádná **zpráva**, je přeskočen i **proces Zaslání pomoci**, takže **skupina cestujících ve vozidle** zůstane ve svém počátečním stavu, **kdy jsou možná zranění**, namísto toho, **aby jim byla poskytnuta pomoc**.

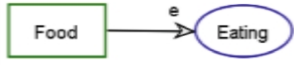
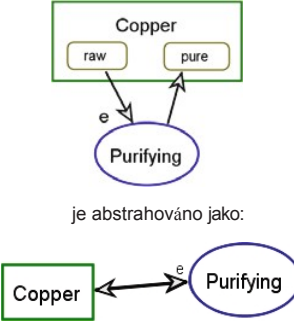
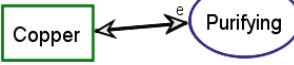
22.4.2 Povolení a transformace odkazů na události

Existují dva druhy transformujících událostních odkazů (tabulka 22.1) a dva druhy umožňujících událostních odkazů (tabulka 22.2).

Tabulka 22.1. Souhm odkazů na události umožňující

Název	Sémantika	Vzor OPD a OPL	Zdroj	Cíl
Odkaz na událost agenta	Agent – člověk – iniciuje a umožňuje proces. Agent musí být přítomen po celou dobu trvání procesu.	 Horník iniciuje a provádí těžbu mědi.	iniciující agent	zahájený proces
Odkaz na událost nástroje	Objekt iniciuje proces jako nástroj, takže se nemění, ale musí existovat po celou dobu trvání procesu.	 Vrták iniciuje těžbu mědi, která vyžaduje vrták.	iniciující nástroj	iniciovaný proces

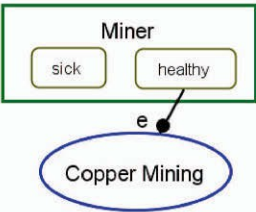
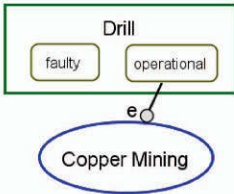
Tabulka 22.2. Shrnutí transformace odkazu na událost

Název	Sémantika	Vzor OPD a OPL	Zdroj	Cíl
Odkaz na událost spotřeby	Objekt iniciuje proces, který, pokud je proveden, spotřebuje objekt.	 Jídlo spustí proces Jídlo, který spotřebuje Jídlo.	iniciující spotřebovávající	zahájený proces, který spotřebovává iniciující spotřebu
Odkaz na událost s účinkem	Objekt iniciuje proces, který, pokud je proveden, ovlivňuje objekt. Odkaz na událost je odkaz z objektu na proces; odkaz z procesu na objekt není odkazem na událost.	 je abstrahováno jako:  Měď iniciuje čištění, který ovlivňuje měď.	iniciující ovlivněný	iniciovaný proces, který ovlivňuje iniciující ovlivněný objekt

22.4.3 Stavově specifikované aktivační a transformační událostní vazby

Tabulka 22.3 popisuje dva stavově specifikované aktivační odkazy událostí – jeden pro agenta, druhý pro nástroj. Existují čtyři druhy stavově specifikovaných transformačních odkazů událostí. Ty jsou shrnuty v tabulce 22.4.

Tabulka 22.3 Shrnutí odkazů na události umožňující specifikované stavem

Name	Semantics	Sample OPD & OPL	Source	Destination
State-specified agent event link	The human agent in the specified state both initiates the process and acts as its agent. The agent must be at the specified state throughout the process duration.	 Healthy Miner initiates and handles Copper Mining.	agent state	initiated process
State-specified instrument event link	The object at the specified state both initiates the process and is instrument for its performance. The instrument must be at the specified state throughout the process duration.	 Operational Drill initiates Copper Mining, which requires operational Drill.	instrument state	initiated process

22.4.4 Odkazy na vyvolání

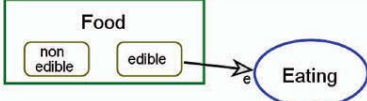
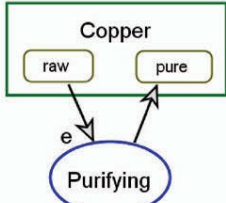
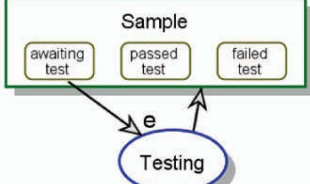
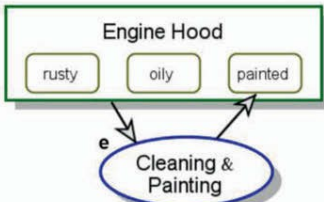
Vyvolání procesu je událost, při které jeden proces iniciuje jiný proces. Vyvolávací odkaz spojuje zdrojový proces s cílovým procesem, který iniciuje, což znamená, že když zdrojový proces úspěšně dokončí svou činnost, okamžitě iniciuje cílový proces – proces na cílovém konci vyvolávacího odkazu. V normálním nebo očekávaném toku řízení provádění zdrojový proces nespustí nový proces, pokud se předchozí proces úspěšně nedokončí. Je na modeláři, aby se postaral o modelování toho, co by se mělo stát s jakýmkoli procesem, který je přerušen, např. z důvodu časové výjimky.

Vzhledem k tomu, že proces OPM podle definice transformuje objekt, odkaz na vyvolání sémanticky implikuje vytvoření prozatímního objektu vyvolávajícím zdrojovým procesem, který následně vyvolaný cílový proces okamžitě spotřebuje. Jak je popsáno v oddíle 10.10.3 v modelu OPM, odkaz na vyvolání může nahradit přechodný, krátkodobý fyzický nebo informační objekt, který zdrojový proces vytvoří, aby inicioval cílový proces, který tento přechodný objekt okamžitě spotřebuje. Fyzický objekt **Spark** na obr.

10.11 je jedním z příkladů; dalším příkladem je ID záznamu v dotazu.

Graficky je volací odkaz znázorněn jako symbol blesku, zubatá (a případně zakřivená) čára vedoucí od volajícího zdrojového procesu k volanému cílovému procesu, která končí uzavřenou šipkou u volaného procesu. Toto je symbol běžného volacího odkazu.

Tabulka 22.4 Souhrn odkazů na transformační události specifikované stavem

Name	Semantics	Sample OPD & OPL	Source	Destination
State-specified consumption event link	The source state triggers the process, and if the precondition is satisfied, the process consumes the object.	 Edible Food initiates Eating, which consumes Food.	consumee state	consuming process
Input-output-specified effect event link pair (consisting of one state-specified event input link and one state-specified output link)	The source state triggers the process, and if the precondition is satisfied, the process changes the object from the source state to the output state.	 Raw Copper initiates Purifying, which changes Copper from raw to pure.	affectee input (source) state	affecting process
			affecting process	affectee output (destination) state
Input-specified effect event link pair (consisting of one state-specified event input link and one state-unspecified output link)	The source state triggers the process, and if the precondition is satisfied, the process changes the object from the source state to some state of the source object.	 Awaiting test Sample initiates Testing, which changes Sample from awaiting test.	affectee input (source) state	affecting process
			affecting process	affectee
Output-specified effect event link pair (consisting of one state-unspecified event input link and one state-specified output link)	The source object triggers the process, and if the precondition is satisfied, the process changes the object from some state to the destination (output) state of the object.	 Engine Hood initiates Cleaning & Painting, which changes Engine Hood to painted.	affectee	affecting process
			affecting process	affectee output (destination) state

Tabulka 22.5 Souhrn odkazů na vyvolání

Name	Semantics	Sample OPD & OPL	Source	Destination
Invocation link	As soon as the invoking process ends, it invokes the process pointed to by the invocation link.	 Product Finishing invokes Product Shipping.	Initiating process	Another initiated process
Self-invocation link	Upon completion, the process immediately invokes itself.	 Recurrent Processing invokes itself.	Initiating process	The same process

Existuje druhý druh volacího odkazu – odkaz na vlastní volání, který umožňuje modelovat volání procesu sám sebou: Po dokončení procesu se proces okamžitě volá sám. Odkaz na vlastní volání je symbolizován dvojicí volacích odkazů, které vycházejí z procesu a spojují se hlavou s ocasem, než skončí zpět v původním procesu, což označuje odkaz na vlastní volání. Odkazy na vyvolání jsou shrnuty v [tabulce 22.5](#). Pokud je mezi dvěma po sobě jdoucími vyvoláními potřeba čekací doba, lze vložit proces **čekání** se specifikovanými časovými omezeními (viz níže) jako cíl z vyvolávajícího procesu a jako cíl zpět do stejného procesu. Odkaz na vyvolání z posledního podprocesu do jeho nadřazeného procesu lze použít k vytvoření smyček.

22.5 Podmíněné odkazy

Sémantika podmínky procesu přeskočí provedení daného procesu, pokud není splněna jeho předběžná podmínka.

Podmínkový odkaz je procedurální odkaz s modifikátorem řízení **c**, který označuje přidání sémantiky podmínky k cílovému procesu odkazu.

Podmíněné propojení poskytuje mechanismus obejití, který umožňuje řízení provádění systému přeskočit nebo obejít cílový proces, pokud selže vyhodnocení splnění jeho předpokladu. Bez mechanismu obejití podmíněného propojení způsobí nesplnění předpokladu, že proces bude čekat na další událost.

Po příchodu nové události se předběžná podmínka procesu znovu vyhodnotí a pokud je splněna, proces se spustí, jinak znovu čeká na další událost. To může způsobit, že se řízení v tomto procesu zasekne na neurčito v nekonečné smyčce. Použití podmíněného propojení takovým situacím zabrání.

Jak je popsáno v oddíle 21.17, stejně jako u všech kontrolních propojení, pokud je propojení podmínky připojeno k procesu P a P je přiblíženo, podmíněné propojení se automaticky přesune do prvního podprocesu (nebo dvou či více prvních souběžných podprocesů) P. Modelář může přesunout propojení z tohoto prvního podprocesu do jiného podprocesu nebo přidat další propojení ze stejného zdroje do jednoho nebo více podprocesů jiných než prvního.

22.5.1 Přeskakování má přednost před čekáním

Sada objektů předprocesu může obsahovat jak podmíněné odkazy, tak nepodmíněné odkazy, tj. procedurální odkazy bez modifikátoru podmíněného řízení. Rozlišujícím aspektem podmíněných odkazů je jejich sémantika přeskočení – přeskočení nebo obejití procesu, pokud operační instance zdrojového objektu podmíněného odkazu neexistuje nebo není v požadovaném stavu. Bez modifikátoru podmíněné kontroly způsobuje neexistence operační instance zdrojového objektu procedurálního odkazu, že proces čeká na další událost a na existenci operačních instancí všech zdrojových objektů, případně v určeném stavu, čímž splňuje předběžnou podmínku.

Pro splnění předběžné podmínky a zahájení procesu je nutné splnit všechny podmínky spojené se všemi objekty nebo stavy v sadě objektů předzpracování propojených podmíněnými odkazy. Pokud sada objektů předzpracování obsahuje jeden nebo více objektů nebo stavů propojených nepodmíněnými odkazy a jeden nebo více objektů nebo stavů propojených podmíněnými odkazy, může dojít ke konfliktu mezi sémantikou čekání vyvolanou nepodmíněnými odkazy a sémantikou přeskočení vyvolanou podmíněnými odkazy. Pro vyřešení konfliktu je sémantika přeskočení definována jako silnější než sémantika čekání, jak je uvedeno v následujícím principu OPM o přednosti sémantiky přeskočení.

Princip OPM o prioritě sémantiky přeskočení

Sémantika přeskočení má přednost před sémantikou čekání.

I když neexistuje pouze jedna z podmínek spojených s podmínkovými odkazy propojenými s procesem, vyhodnocení splnění předběžné podmínky selže, řízení provádění přeskočí proces a dojde k události, která iniciuje další sekvenční proces (nebo další dva nebo více paralelních procesů).

Podmínky spojené s podmínkovými vazbami jsou při hodnocení předběžných podmínek *zohledněny jako první*, protože pokud nejsou splněny, proces, který má být proveden, je přeskočen, bez ohledu na výsledek hodnocení zbývajících částí jeho předběžné podmínky. Pokud je přeskočený proces v kontextu přiblížení a v tomto kontextu existuje následující proces, řízení provádění spustí tento další proces, jinak se řízení provádění vrátí zpět k procesu v kontextu přiblížení.

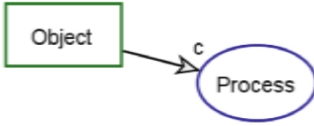
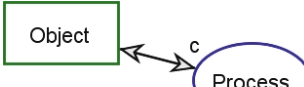
Existují dva druhy základních podmíněných odkazů: odkazy transformující podmínky a odkazy umožňující podmínky.

22.5.2 Odkazy transformující podmínky

Odkaz na spotřebu podmínky spojuje spotřebitele s procesem s přidáním modifikátoru řízení.

c. **Tabulka 22.6** shrnuje základní podmínky transformující odkazy.

Tabulka 22.6 Shrnutí podmínek transformace odkazů

Název	Sémantika	Vzor OPD a OPL	Zdroj	Cíl
Podmínka spotřeby odkazu	Pokud existuje instance objektu a jsou splněny ostatní předpoklady procesu, proces se provede a spotřebuje instanci objektu, jinak se řízení provádění přesune k zahájení dalšího procesu.	 Proces se spustí, pokud objekt existuje, v takovém případě proces spotřebuje objekt, jinak je proces přeskočen.	Podmíněný objekt	Podmíněný proces
Odkaz na účinek podmínky	Pokud existuje instance objektu a jsou splněny ostatní předpoklady procesu, pak se proces provede a ovlivní instanci objektu, jinak se řízení provádění přesune k zahájení dalšího procesu.	 Proces se spustí, pokud objekt existuje, v takovém případě proces ovlivní objekt, jinak je proces přeskočen.	Podmíněný objekt	Podmíněný proces

Pokud v době běhu (tj. během provádění systémového modelu) existuje instance spotřebitele, když událost iniciuje proces, pak přítomnost této instance spotřebitele splňuje předběžnou podmínku procesu s ohledem na tento objekt. Pokud je splněno vyhodnocení celé předběžné podmínky, která zohledňuje celou sadu objektů předběžného zpracování (jejichž součástí je spotřebitel), proces se spustí a spotřebuje danou instanci spotřebitele. Pokud však instance spotřebitele neexistuje v okamžiku, kdy událost spustí proces, pak bez ohledu na zbytek sady objektů předběžného zpracování vyhodnocení předběžné podmínky procesu selže a tok řízení provádění obejde (přeskočí) proces, aniž by jej provedl.

Podmíněný efektový odkaz, stejně jako jeho běžný protějšek bez podmíněného efektu, spojuje ovlivněný objekt s procesem, s přidáním modifikátoru řízení **c**. Pokud v době běhu existuje instance ovlivněného objektu, když událost spustí proces, pak přítomnost této instance ovlivněného objektu splňuje předběžnou podmínku procesu s ohledem na tento objekt. Stejně jako u podmíněného spotřebního propojení, pokud je splněno vyhodnocení celé předběžné podmínky, která zahrnuje celou sadu objektů předzpracování (jejichž součástí je ovlivněný objekt), proces se spustí a ovlivní danou instanci ovlivněného objektu, ale pokud není, vyhodnocení předběžné podmínky procesu selže a tok řízení provádění obejde proces bez jeho provedení.

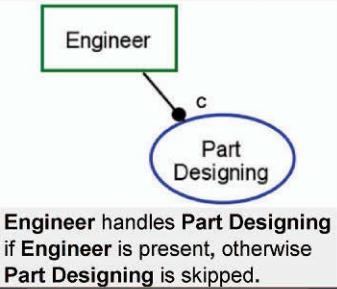
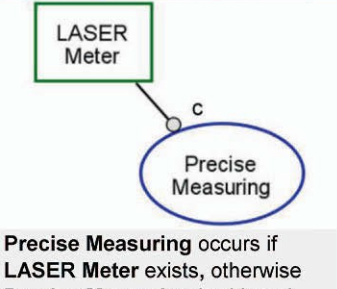
22.5.3 Podmínkové aktivační vazby

Existují dva druhy základních (nestanovených stavem) podmínek umožňujících propojení: propojení podmíněného agenta a propojení podmíněného nástroje. **Propojení podmíněného agenta** je propojení agenta s procesem s

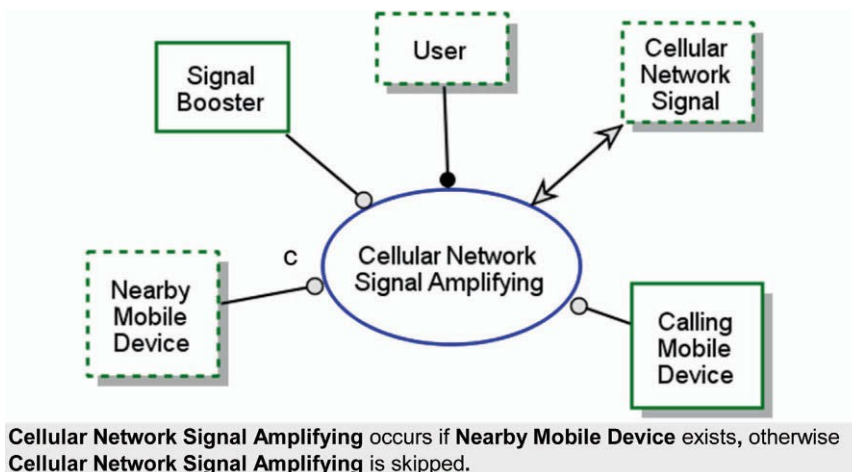
doplněním modifikátoru řízení **c**. Pokud v době běhu existuje instance agenta, když událost spustí proces, pak přítomnost této instance agenta splňuje předběžnou podmínku procesu s ohledem na tento objekt. Pokud je splněno i vyhodnocení zbývajících předpokladů, proces se spustí a daný agent se postará o jeho provedení. Pokud však v okamžiku, kdy událost spustí proces, neexistuje instance agenta, vyhodnocení předpokladu procesu selže a tok řízení provádění proces obejde nebo „přeskočí“ bez provedení procesu.

Podmíněné propojení nástroje je propojení nástroje s procesem, označené modifikátorem řízení **c**. Pokud v době běhu existuje instance nástroje, když událost spustí proces, pak přítomnost této instance nástroje splňuje předběžnou podmínku procesu s ohledem na tento objekt. Pokud vyhodnocení celé sady objektů předprocesu splňuje podmínku, proces se spustí. Pokud však v okamžiku, kdy událost spustí proces, neexistuje žádná instance nástroje, vyhodnocení podmínky procesu selže a tok řízení provádění proces obejde nebo „přeskočí“ bez provedení procesu (tabulka 22.7).

Tabulka 22.7 Souhrn podmínek umožňujících propojení

Name	Semantics	Sample OPD & OPL	Source	Destination
Agent condition link	The agent enables the process if the agent is present, otherwise the process is skipped.	 Engineer handles Part Designing if Engineer is present, otherwise Part Designing is skipped.	conditioning agent	conditioned process
Instrument condition link	The instrument enables the process if it exists, otherwise the process is skipped.	 Precise Measuring occurs if LASER Meter exists, otherwise Precise Measuring is skipped.	conditioning instrument	conditioned process

Obrázek 22.1 je OPD s podmíněným propojením nástrojů z **blízkého mobilního zařízení** do **zesilovače signálu mobilní sítě**, které se vyskytuje pouze v případě, že existuje objekt prostředí **blízké mobilní zařízení**, a v opačném případě je přeskočeno, protože nemá smysl zesilovat, pokud není v blízkosti žádné zařízení. Tabulka 22.6 shrnuje základní propojení transformující podmínky.



Obr. 22.1 Podmínkové propojení nástrojů (s částečným OPL)

22.5.4 Podmínkové transformační propojení specifikované stavem

Stejně jako jejich protějšky transformačních odkazů specifikovaných stavem události existují čtyři druhy transformačních odkazů specifikovaných stavem podmínky. Ty jsou shrnuty v [tabulce 22.8](#).

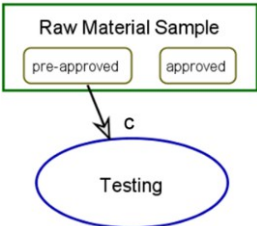
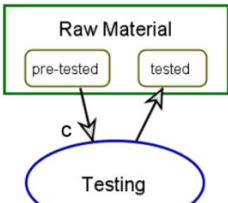
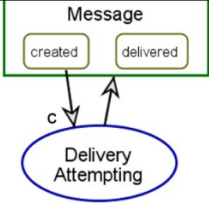
22.5.5 Podmínkové aktivující odkazy specifikované stavem

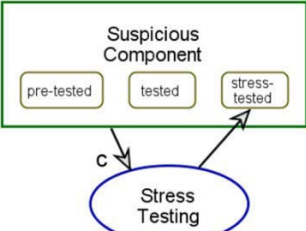
Stejně jako jejich běžné protějšky bez specifikace stavu existují dva typy aktivujících odkazů specifikovaných stavem: odkaz agenta specifikovaný stavem a odkaz nástroje specifikovaný stavem.

Podmíněné spojení agenta specifikované stavem je spojení agenta specifikované stavem, opatřené kontrolním modifikátorem **c**, ze specifikovaného stavu agenta do procesu. Pokud v době běhu existuje instance agenta nebo je přítomna ve specifikovaném stavu, když událost iniciuje proces, pak je splněna předběžná podmínka procesu s ohledem na tento objekt. Pokud vyhodnocení celé sady objektů předprocesu splňuje předpoklad, proces se spustí a daný agent musí být přítomen, aby jej zpracoval, dokud neskončí. V opačném případě vyhodnocení předpokladu procesu selže a tok řízení provádění obejde nebo „přeskočí“ provedení procesu.

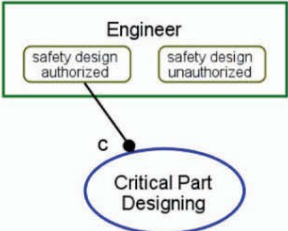
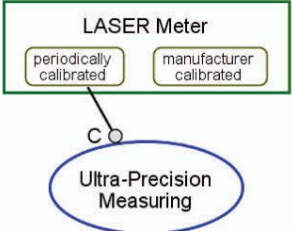
Podmíněný stavově specifikovaný nástrojový odkaz je stavově specifikovaný nástrojový odkaz, opatřený kontrolním modifikátorem **c**, z určitého stavu nástroje k procesu.

Tabulka 22.8 Souhrn odkazů transformace specifikovaných stavem podmínky

Název	Sémantika	Vzor OPD & PL	Zdroj	Cíl
Podmínka spojení spotřeby specifikovaná stavem	Proces se provede, pokud je objekt ve stavu, ze kterého odkaz pochází, jinak se proces přeskočí.	 Testování se provádí, pokud je vzorek suroviny předem schválen, v takovém případě se vzorek suroviny spotřebovává, jinak je testování přeskočeno.	podmíněný stav objektu	podmíněný proces
Podmíněný vstup-výstup-specifikovaný efekt odkaz	Proces se provede, pokud je objekt ve vstupním stavu (odkud odkaz pochází) a změní objekt z jeho vstupního stavu do jeho výstupního stavu, jinak je proces přeskočen.	 Testování se provádí, pokud je surovina předem otestována. V takovém případě se surovina změní z předem otestované na otestovanou. v opačném případě se testování přeskočí.	kondicionovaný specifikovaný vstupní stav objektu	podmíněný proces
Podmíněný vstupní efektový odkaz	Proces se provede, pokud je objekt ve vstupním stavu (ze kterého odkaz pochází), a změní objekt z jeho vstupního stavu na kterýkoli z jeho stavů, jinak je proces přeskočen.	 Doručení se pokusí, pokud je vytvořena zpráva, v takovém případě se doručení pokusí změnit zprávu z vytvořené, jinak se doručení pokusí je přeskočen.	podmíněný proces	podmíněný proces

Podmíněný výstup specifikovaný efektem odkazu	Proces se provede, pokud je objekt ve vstupním stavu (od kterého odkaz pochází) a změní objekt z jeho vstupního stavu do kteréhokoli z jeho stavů, jinak je proces přeskočen.	 Zátěžové testování se provádí, pokud existuje podezřelá součástka. V takovém případě se podezřelá součástka změní na zátěžově testovanou, jinak se zátěžové testování přeskočí.	kondicionační objekt	podmíněný proces
--	---	---	----------------------	------------------

Tabulka 22.9 Shrnutí podmíněných aktivujících odkazů specifikovaných stavem

Name	Semantics	Sample OPD & OPL	Source	Destination
State-specified agent condition link	The agent enables the process if the agent is in the specified state, otherwise the process is skipped.	 Engineer handles Critical Part Designing if Engineer is safety design authorized, otherwise Critical Part Designing is skipped.	conditioning specified state of agent	conditioned process
State-specified instrument condition link	The instrument enables the process if it is in the specified state, otherwise the process is skipped.	 Ultra-Precision Measuring occurs if LASER Meter is periodically calibrated, otherwise Precise Measuring is skipped.	conditioning specified state of instrument	conditioned process

Pokud v době běhu existuje instance nástroje a je ve specifikovaném stavu, když událost spustí proces, pak je splněna předběžná podmínka procesu s ohledem na tento objekt. Pokud vyhodnocení celé sady předběžných objektů splňuje předběžnou podmínku, proces se spustí a tento nástroj musí zůstat existující a ve stejném stavu po celou dobu trvání procesu.

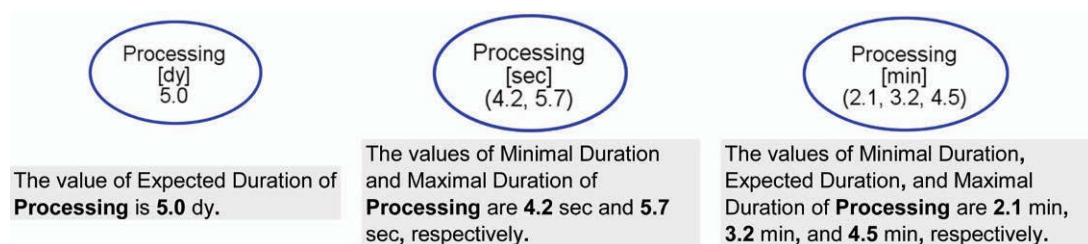
Pokud v době běhu instance nástroje neexistuje nebo existuje v jiném stavu, než je stav připojený ke zdroji propojení, pak není splněna předběžná podmínka procesu s ohledem na tento objekt, vyhodnocení předběžné podmínky procesu selže a tok řízení provádění obejde provedení procesu. [Tabulka 22.9](#) shrnuje propojení umožňující specifikované stavové podmínky.

22.6 Výjimkové odkazy

Výjimkové odkazy umožňují modelovat, co dělat v případě výjimky při provádění procesu pod minimální nebo nad maximální prahovou hodnotou.

22.6.1 Doba trvání procesu a její rozložení

Proces může mít vlastnost **Trvání** (atribut metamodelu) s hodnotou vyjádřenou v časových jednotkách, která musí být kompatibilní s normou ISO 80000-3:2006 – *Veličiny a jednotky – Část 3: Prostor a čas*, která je součástí skupiny norem ISO/IEC 80000 tvořících *Mezinárodní systém veličin*. Jednotkami času mohou být milisekundy [ms], sekundy [sec], minuty [min], hodiny [hr], dny [dy], týdny [wk], měsíce [mo] nebo roky [yr]. **Trvání** může být specializováno na **minimální trvání**, **očekávané trvání** a **maximální trvání**. **Minimální trvání** a **maximální trvání** označují minimální a maximální přípustnou dobu pro dokončení procesu. Trvání je volitelné a, jak ukazuje [obr. 22.2](#), modelář si může vybrat, zda uvede pouze očekávanou (nominální) dobu, minimální a maximální dobu, nebo všechny tři doby trvání.



Obr. 22.2 Tři způsoby, jak označit dobu trvání procesu: vlevo – pouze očekávaná (nominální) doba, uprostřed – minimální a maximální doba, vpravo – minimální, očekávaná a maximální doba trvání

Hodnota **očekávané délky trvání** procesu je statistický průměr délky trvání daného procesu. **Délka trvání** volitelně vykazuje atribut **rozložení délky trvání** s hodnotou identifikující název a parametry funkce pravděpodobnostního rozložení spojené s délkou trvání procesu nebo neanalytickým rozložením. V době běhu se hodnota **trvání** určuje samostatně pro každou instanci procesu (tj. pro každý jednotlivý výskyt procesu) vzorkováním z **distribuce trvání** procesu. Vlastnost **trvání** umožňuje definovat výjimkové odkazy. Existují dva druhy výjimkových odkazů: výjimkový odkaz pro překročení času a výjimkový odkaz pro nedosažení času.

22.6.2 Odkaz na výjimku z důvodu překročení doby trvání

Odkaz na výjimku přesčasu spojuje zdrojový proces s cílovým procesem zpracování přesčasu, aby se určilo, že pokud v době běhu výkonnostní doba instance zdrojového procesu překročí hodnotu **Maximální doba trvání**, pak událost spustí cílový proces, který je procesem zpracování přesčasu.

Proces s maximální dobou trvání je proces, pro který modelář určí maximální dobu trvání.

Proces zpracování přesčasu je proces časové výjimky, který určuje, co se má udělat v případě, že časový výkon procesu s maximální dobou trvání překročí jeho maximální povolenou dobu.

Odkaz na výjimku z prodloužení je procedurální odkaz z procesu s maximální dobou trvání na proces zpracování prodloužení, který označuje, že pokud doba trvání procesu s maximální dobou trvání překročí jeho maximální dobu trvání, spustí se proces výjimky z prodloužení.

Modifikátor řízení pro odkaz na výjimku z prodloužení je jediná šikmá krátká čára protínající odkaz v blízkosti procesu výjimky z prodloužení (viz [obr. 22.3](#) pro modifikátor řízení odkazu na výjimku z nedodržení času, který je tvořen dvojicí takových čar).

22.6.3 Odkaz na výjimku z důvodu nedodržení minimální doby trvání

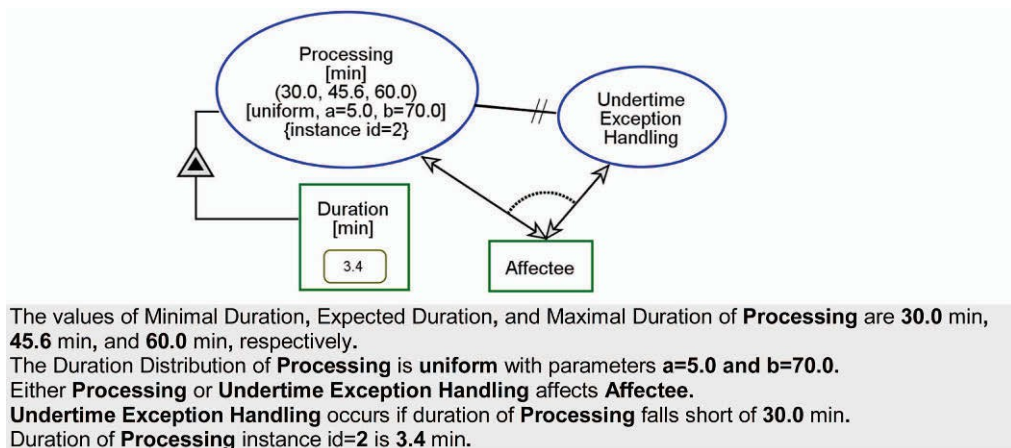
Odkaz výjimky z důvodu nedodržení minimální doby spojuje zdrojový proces s cílovým procesem zpracování nedodržení minimální doby, aby určil, že pokud je v době běhu výkonnostní doba instance zdrojového procesu nižší než jeho hodnota **minimální doby**, pak událost spustí cílový proces, který je procesem zpracování nedodržení minimální doby.

Proces s minimální dobou trvání je proces, pro který modelář určí minimální dobu trvání.

Proces zpracování podčasové výjimky je proces časové výjimky, který určuje, co dělat v případě, že časový výkon minimálně časovaného procesu nedosáhne své minimální délky.

Odkaz na výjimku z nedodržení času je procedurální odkaz z procesu s minimální dobou trvání na proces výjimky z nedodržení času, který označuje, že pokud časový výkon časovaného procesu nedosáhne minimální povolené doby, spustí se proces výjimky z nedodržení času.

Modifikátor řízení pro odkaz na výjimku z důvodu nedodržení času je dvojice paralelních šikmých krátkých čar, které protínají odkaz v blízkosti procesu výjimky z důvodu překročení času. [Obrázek 22.3](#) je příkladem **zpracování výjimky z důvodu nedodržení času**. Zde je {instance id=2} konkrétní instance (výskyt) procesu **Processing**, jehož **doba trvání** je **3,4** minuty. Jelikož tato hodnota je menší než **30,0** minut – minimální doba trvání definovaná pro třídu procesů **Processing**, dojde ke zpracování **výjimky z důvodu nedodržení času**.



Obr. 22.3 Příklad výjimky z důvodu nedodržení doby trvání

Zdrojový proces může mít odkazy na překročení a nedosažení doby, z nichž každý je připojen k jinému procesu zpracování výjimky doby v cíli. Předpokládejme, že v příkladu na [obr. 22.3](#) přidáme proces **zpracování výjimky překročení doby**, pak by další věta OPL byla:

K výjimce přesčasu dochází, pokud doba **zpracování** přesáhne 60,0 min.

Na rozdíl od většiny procedurálních odkazů, které spojují objekt a proces, ale podobně jako odkaz na vyvolání, jsou dva odkazy na časovou výjimku procedurálními odkazy, které přímo spojují dva procesy. Implicitní prozatímní objekt **Overtime Exception Message** (Zpráva o výjimce z důvodu přesčasu) nebo **Undertime Exception Message** (Zpráva o výjimce z důvodu nedodržení času) je vytvořen mechanismem provádění procesu OPM, jakmile zjistí, že proces nebyl ukončen v maximálním přiděleném čase nebo byl ukončen předčasně, respektive nedosáhl minimálního přiděleného času. Vzhledem k tomu, že operační mechanismus OPM tyto objekty vytváří a okamžitě spotřebovává, jejich zobrazení v modelu není explicitní. To je podobné jako u invokačního propojení, které potlačuje vytvoření dočasného objektu zdrojovým procesem a jeho okamžité spotřebování cílovým procesem. [Tabulka 22.10](#) shrnuje

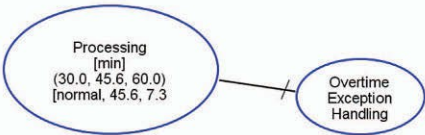
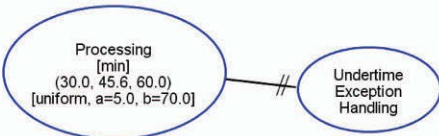
dva časové výjimkové odkazy.

Výjimky, které tyto odkazy zpracovávají, se týkají pouze času, ale lze je použít také pro modelování výjimek při provádění. Pokud je například přeskočen proces s minimální dobou trvání připojený k odkazu výjimky podčasu, což znamená, že jeho doba trvání byla 0, spustí se proces zpracování výjimky.

22.7 Míra transformace

Často vzniká potřeba modelovat spotřebu spotřebitele nebo vliv na ovlivněnou osobu nebo vytvoření výsledku nikoli jako jednorázovou událost, ale spíše jako nepřetržitý proces nebo diskrétní proces s množstvím větším než 1, transformovaný v čase. *Vlastnost* jsme definovali jako atribut prvku OPM. Například **vytrvalost** je vlastnost OPM **Thing**. Pokud je hodnota této vlastnosti **trvalá**, **Thing** je **objekt**; pokud je přechodná, je to **proces**. Jinými slovy, můžeme říci, že vlastnost je atribut na úrovni metamodelu, kde **Thing** a **Link** jsou prvky OPM. **Vytrvalost** je příkladem vlastnosti **Thing**. **Míra transformace** je vlastnost (transformujícího) **Link**.

Tabulka 22.10 Souhrn časových výjimek odkazů

Name	Semantics	Sample OPD & OPL	Source	Destination
Overtime exception link	If in runtime the process instance takes more than the maximal process class duration, the overtime exception handling process is invoked	 Engineer handles Critical Part Designing if Engineer is safety design authorized, otherwise Critical Part Designing is skipped.	process with maximal time specification	overtime exception handling process
Undertime exception link	If in runtime the process instance takes less than the minimal process class duration, the undertime exception handling process is invoked	 Ultra-Precision Measuring occurs if LASER Meter is periodically calibrated, otherwise Precise Measuring is skipped.	process with minimal time specification	undertime exception handling process

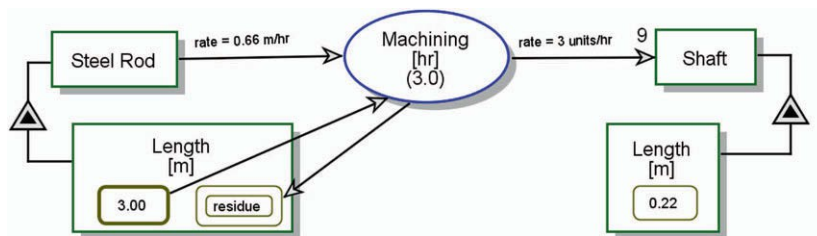
***Míra transformace** je vlastnost procedurálního odkazu spojujícího transformovaný objekt B a proces P, jehož hodnota je míra transformace B pomocí P.*

Stejně jako se transformace specializuje na spotřebu, účinek a výsledek, tak se specializuje i rychlost transformace.

***Míra spotřeby** je transformační poměr spotřebního spojení mezi spotřebitelem B a procesem P, jehož hodnota je míra spotřeby B procesem P.*

***Míra výnosu** je míra transformace výsledného spojení mezi výsledkem B a procesem P, jehož hodnota je míra tvorby B procesem P.*

***Míra účinku** je míra transformace účinkového spojení mezi ovlivňovaným B a procesem P, jehož hodnota je míra ovlivnění B procesem P.*



Steel Rod exhibits **Length**.

The value of **Length** of **Steel Rod** is initially **3.00** m and finally **residue**.

Shaft exhibits **Length**.

The value of **Length** of **Shaft** is **0.22** m.

The value of Expected Duration of **Machining** is **3.0** hr.

Machining consumes **Steel Rod** at a rate of **0.66** m/hr.

Machining yields **9 Shafts** at a rate of **3** units/hr.

Obr. 22.4 Příklad míry spotřeby a míry výnosu

Míra účinku může být vyjádřena konkrétněji jako míra změny stavu.

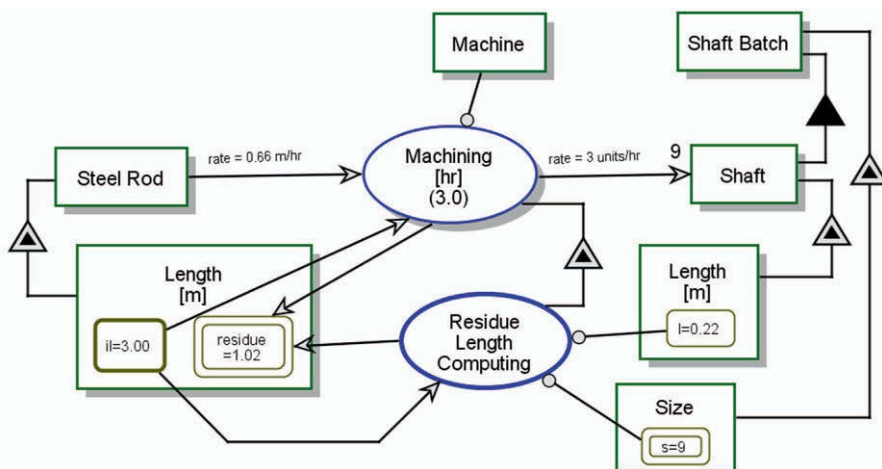
***Míra změny stavu** je míra transformace páru vstupních a výstupních vazeb, jejichž vstupní a výstupní vazby spojují vstupní stav b_i a výstupní stav b_o ovlivněného subjektu B s procesem P , jehož hodnota je míra změny stavu B procesem P z b_i na b_o .*

Obrázek 22.4 uvádí příklad míry spotřeby a míry výnosu. Modelář může vytvořit výjimku, pokud je množství výsledku nebo spotřeby menší než míra násobená očekávanou dobou trvání procesu.

22.8 Výpočty s OPM

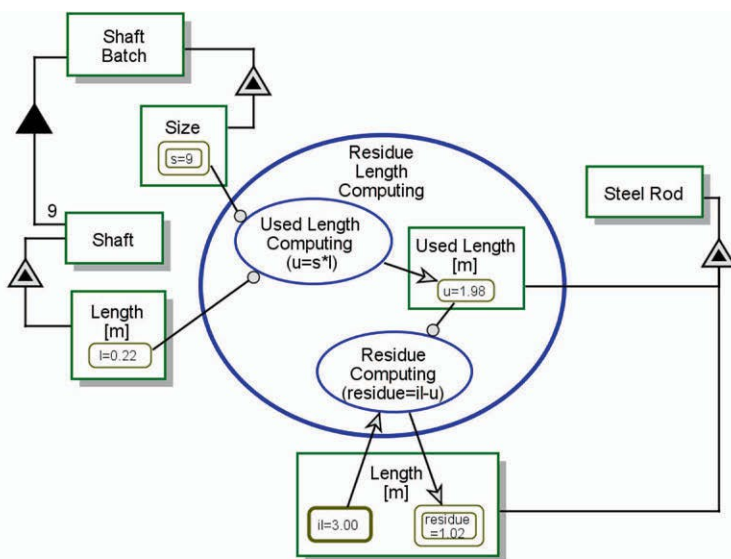
Modely OPM lze použít k provádění numerických výpočtů. Atomické procesy pro výpočty jsou čtyři základní aritmetické operace: **sčítání**, **odčítání**, **násobení** a **dělení**. Ty se používají k vymýšlení složitějších výpočtů, jako je **výpočet průměru**, **výpočet geometrického průměru** atd. Je třeba postupovat opatrně u operací, které nejsou komutativní, jako je **dělení**, kde musí být role **dělece** a **dělitele** explicitní, aby bylo možné získat správný **podíl**. Vzhledem k tomu, že matematické výrazy jsou mnohem kompaktnější a srozumitelnější, lze po dosažení dostatečně nízké úrovně výpočtu zaznamenat skutečné vzorce jako součásti názvů výpočetních procesů.

Jako průmyslový příklad předpokládejme, že pro systém na obr. 22.4 chceme vypočítat hodnotu **zbytková délka** – konečná hodnota délky **ocelové tyče** v metrech po jejím užití. To je znázorněno na obr. 22.5 procesem **Výpočet délky zbytku** a na obr. 22.6, kde je **Výpočet délky zbytku** zvětšen. Počáteční **délka ocelové tyče**, il, je **3,00 m**. **Obráběcí proces**, který trvá **3 hodiny**, spotřebovává **ocelovou tyč** rychlostí **0,66 m/hod**.



Residue Length Computing requires the value $l=0.22$ m of **Length** of **Shaft** and the value of **Size** $s=9$ of **Shaft Batch**. **Residue Length Computing** changes the value of **Length** of **Steel Rod** from $il=3.00$ m to **residue** $=1.02$.

Obr. 22.5 SD obráběcího systému s výpočtem zbytkové délky jako operací obrábění



Residue Length Computing from SD zooms in SD1 into **Used Length Computing** ($u=s \cdot l$) and **Residue Computing** ($\text{residue} = il - u$) in that sequence, as well as **Used Length** of **Steel Rod** in m.

Used Length Computing requires the value $l=0.22$ of **Length** of **Shaft** in m and the value $s=9$ of **Size** of **Shaft Set**. **Used Length Computing** yields **Used Length** of **Steel Rod** in m with value $u=1.98$.

Residue Computing changes the value of **Length** of **Steel Rod** in m from $il=3.00$ to **residue** $=1.02$.

Obr. 22.6 SD1 obráběcího systému z obr. 22.5, kde je zvětšen výpočet zbytkové délky

Proces **obrábění** generuje **hřídele** s výtěžností **3 jednotky/hodinu**, takže za **3 hodiny** získáme **9 hřídelí**, jak je uvedeno v omezení účasti u **hřídele**. Délka každé **hřídele** je **0,22 m** a **velikost dávky hřídelí** (vyřezaných během 3 hodin) je **9**. Všechna tato data jsou uvedena v modelu na [obr. 22.5](#).

Při přiblížení **výpočtu zbytkové délky** na [obr. 22.6](#) vidíme, že má dva dílčí procesy. Prvním je **výpočet použité délky** ($u=s \cdot l$) a druhým **výpočet zbytku** ($zbytky=il-u$). Názvy procesů obsahují v závorkách aritmetické výrazy, které má každý proces provést. Výraz v prvním dílčím procesu počítá u , hodnotu **použité délky tyče**, jako $u=s \cdot l$. Jako hodnotu **velikosti šachtové dávky** bere $s=9$ a jako **délku každé šachty** $l=0,22$ m. Součin $u=s \cdot l = 9 \cdot 0,22 = 1,98$ m je vstupem pro další dílčí proces, ve kterém model vypočítá **zbytek** $=il-u$, protože délka zbytku je rozdílem mezi il , hodnotou počáteční **délky tyče**, **3,00 m**, a u , hodnotou **použité délky tyče**, takže **zbytek** $=il-u = 3,00-1,98=1,02$ m. Různé hodnoty parametrů samozřejmě povedou k různým výsledkům. Tento příklad ukazuje, jak OPM umožňuje kombinovat koncepční modelování s kvantitativním modelováním, které poskytuje odůvodnění pro různé matematické kroky zahrnuté do výpočtu.

22.9 Sady a iterace

Množina je soubor instancí objektů stejné třídy.

Příklad sady je uveden na [obr. 22.7](#). **Shaft Batch** je množina devíti instancí objektů ze třídy **Shaft**, takže vytvoření **Shaft Batch** znamená devětkrát opakovat **Machining**, přičemž pokaždé se vyrobí jeden **Shaft**. Toto je krátký formální způsob modelování iterace v OPM: Kdykoli je proces připojen ke dvěma procedurálním odkazům stejného druhu, z nichž jeden je odkazem na množinu n členů a druhý na člena množiny, sémantika je iterace.

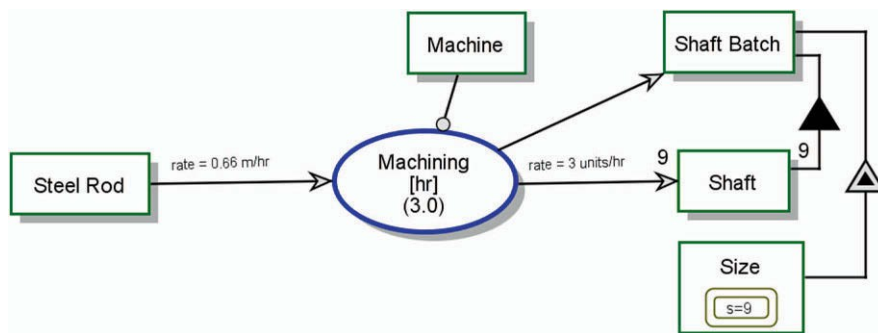
V našem příkladu jsou oba odkazy odkazy na výsledky: jeden odkaz na výsledek vede z **Obrábění** do sady **Sada hřídelí** a druhý z **Obrábění** do **Hřídelí**. Sémantika této šablony je devětkrát opakované vytvoření **Shaft**. To je jasnější, když se zaměříme na **Machining** v SD1, což vyjadřuje skutečnost, že **Cutting** a **Lathing** se provádějí postupně a opakovaně devětkrát, aby vzniklo devět **Shaftů**. Každý výskyt **Machining** je procesní instancí **Machining**, v rámci které se provádí **Cutting** a **Lathing**, aby vzniklo každé z devíti instancí **Shaft**.

Iterace může kombinovat libovolnou podmnožinu procedurálních odkazů. Iterace může být samozřejmě aplikována také na infromatické objekty, což poskytuje pohodlný a krátký způsob modelování iterací, například v algoritmech, a slouží, kromě mnoha dalších kontrolních konstrukcí (jako jsou booleovské objekty), k automatizovanému generování kódu.

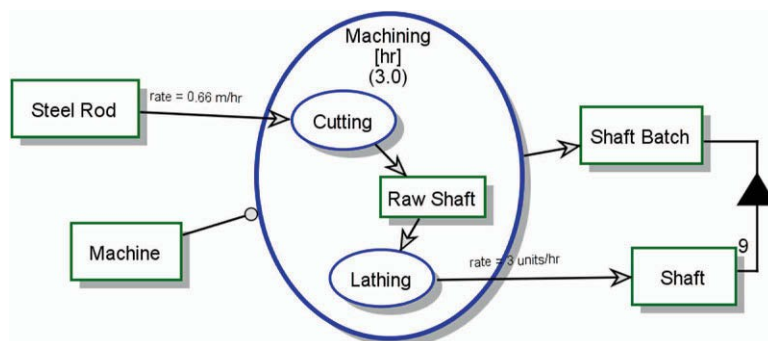
22.10 Operační sémantika v kontextech procesů se zoomem

Přiblížení procesu specifikuje přenos řízení provádění na podprocesy na další úrovni detailu. Provádění procesu s přiblíženým kontextem rekurzivně přenáší řízení provádění na nejvyšší

podprocesu (podprocesů) v kontextu nejhlubšího procesu. Řízení se vrací do přiblíženého procesu poté, co jeho poslední podproces dokončí své provedení (obr. 22.8).



Obr. 22.7 SD **obrábění**, kde **Shaft Batch** je sada 9 instancí objektů z třídy **Shaft**, takže vytvoření **Shaft Batch** znamená devětkrát opakovat **obrábění**, přičemž pokaždé vznikne jeden **Shaft**.



Obr. 22.8 SD1 **obrábění**, ve kterém je **obrábění** přiblíženo, vyjadřující skutečnost, že **řezání** a **soustružení** se provádí postupně a opakovaně 9krát, aby se získalo devět **hřídelí**

22.10.1 Implicitní vyvolávací odkaz

Implicitní volací odkaz je odkaz, který není graficky viditelný, ale je implikován vertikálním uspořádáním procesů v kontextu přiblíženého procesu.

Podobně jako jeho explicitní protějšek, implicitní volací odkaz znamená zahájení následného procesu nebo souběžné zahájení procesů. Jelikož volání je událost, je nutné splnit předpoklady pro každý podproces, aby bylo možné zahájit jeho provádění.

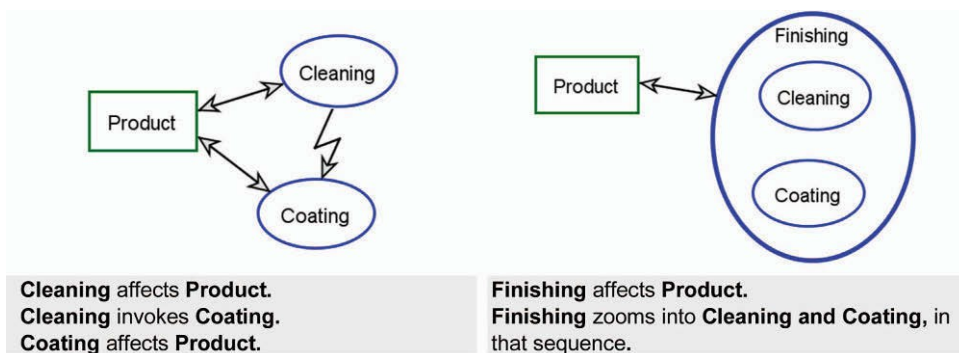
Implicitní odkaz na vyvolání může být (1) z procesu na jeho první (nebo několik) podprocesů, (2) z podprocesu na jeden nebo více podprocesů těsně pod ním podél časové osy v kontextu přibližného procesu, nebo (3) z posledního přibližného podprocesu (podprocesů) na jejich obklopující proces definující kontext.

Konkrétně (1) po příchodu do přibližného kontextu procesu se řízení okamžitě přenesse na podproces (podprocesy) s nejvyšším bodem elipsy (ovalu) v tomto přibližném kontextu procesu. Implicitní volací odkaz z přibližného procesu na jeho nejvyšší podproces přenáší řízení provádění.

(2) Podél časové osy procesu dokončení zdrojového podprocesu (nebo posledního podprocesu, který dokončil provádění v případě dvou nebo více podprocesů, které byly spuštěny současně) okamžitě iniciuje následující podprocesy pomocí implicitního volacího odkazu. (3) Po dokončení provádění podprocesu s nejvyšším bodem elipsy, který je nejnížší v tomto přibližném kontextu procesu, se řízení provádění vrací do přibližného procesu.

Pokud mají dva nebo více podprocesů nejvyšší body elipsy ve stejné výšce, implicitní volací odkaz spustí každý proces a ty se spustí paralelně po splnění jednotlivých předběžných podmínek. Proces, který se dokončí jako poslední, spustí další podproces nebo sadu paralelních podprocesů.

V OPD na levé straně [obr. 22.9](#) vyvolává **čištění nátěr**, takže **čištění** ovlivňuje nejprve **produkt** a poté **nátěr** ovlivňuje **produkt**. Volací odkaz určuje tuto posloupnost procesů. V ekvivalentním OPD na pravé straně [obr. 22.9](#) se **Dokončení** přiblíží na **Čištění** a **Povlakování**, přičemž vrchol elipsy prvního je nad vrcholem druhého, takže když se spustí **Dokončení**, řízení se okamžitě přenesse na **Čištění** a když **Čištění** skončí, implicitní volací odkaz vyvolá **Povlakování**. Oba OPD jsou sémanticky ekvivalentní, ale ten vlevo nemá **Dokončení** jako obklopující kontext, což ho činí méně výstižným z pohledu systému, přičemž používá o dva odkazy více než OPD vlevo.



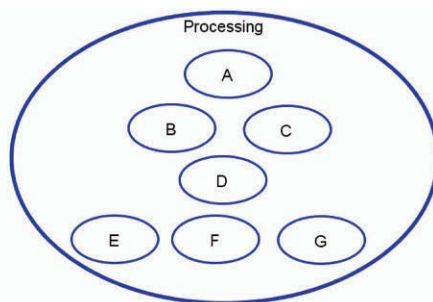
Obr. 22.9 Vyvolávací odkaz (vlevo) a implicitní vyvolávací odkaz (vpravo)

22.10.2 Implicitní sada paralelních volacích odkazů

Graficky řečeno, když jsou vrcholy elipsy dvou nebo více podprocesů v rámci přibližného procesu ve stejné výšce (s možnou přípustnou tolerancí), jsou tyto podprocesy spuštěny a začínají paralelně, přičemž každý z nich se spustí za předpokladu splnění své předběžné podmínky. V této situaci existuje sada implicitních odkazů na vyvolání ze zdroje přibližného procesu ke každému z paralelních podprocesů. Synchronizace procesů je taková, že když skončí poslední z těchto podprocesů, řídicí jednotka spustí další podprocesy. Pokud existují dva nebo více podprocesů s nižším

vrcholem elipsy— ve stejné výšce, řízení je spustí paralelně. Pokud již nejsou žádné další podprocesy k vyvolání, řízení se vrátí k přibližnému proces, který lze dále upřesnit.

Obrázek 22.10 ukazuje podprocesy **zpracování** s následujícím částečným pořadím: **A**, (**B**, **C**), **D**, (**E**, **F**, **G**). **B** a **C** se spustí po dokončení **A**. **D** se spustí po dokončení delšího procesu z **B** a **C**. **E**, **F** a **G** začínají po dokončení **D**. Řízení provádění se vrací ke **zpracování** po dokončení nejdelšího procesu z **E**, **F** a **G**.



Processing zooms into **A**, parallel **B** and **C**, **D**, and parallel **E**, **F**, **G**, in that sequence.

Obr. 22.10 Částečné pořadí podprocesů a implicitní sada paralelních volacích odkazů

Tabulka 22.11 shrnuje druhy implicitních odkazů na volání.

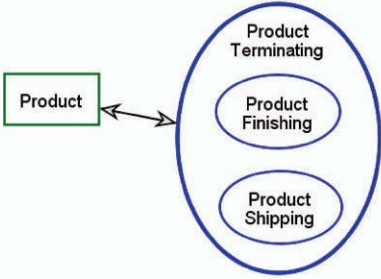
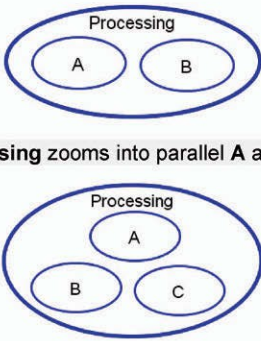
22.10.3 Rozložení odkazů v kontextu

Graficky má procedurální odkaz připojený k obrysu přibližného procesu distribuční sémantiku. Ponechání odkazu připojeného k obrysu přibližného procesu znamená, že odkaz je distribuován a připojen ke každému z dílčích procesů. Obrys přibližného procesu má sémantiku analogickou s algebraickými závorkami následujícími za symbolem násobení, které distribuují operátor násobení do výrazů uvnitř závorek.

Na **obr. 22.11** jsou OPD vlevo a vpravo ekvivalentní, ale ten vlevo je jasnější a přehlednější. Odkaz agenta z **A** do **P** znamená, že **A** zpracovává podprocesy **P1**, **P2** a **P3**. Odkaz nástroje z **B** do **P** znamená, že podprocesy **P1**, **P2** a **P3** vyžadují **B**. Analogicky v algebře předpokládejme, že spojení agenta (nebo nástroje) bylo operátorem násobení, **A** bylo násobitelem a přiblížení bylo sčítáním, takže $P = P1 + P2 + P3$ a **P** bylo násobené číslo, pak $A * P = A * (P1 + P2 + P3) = A * P1 + A * P2 + A * P3$.

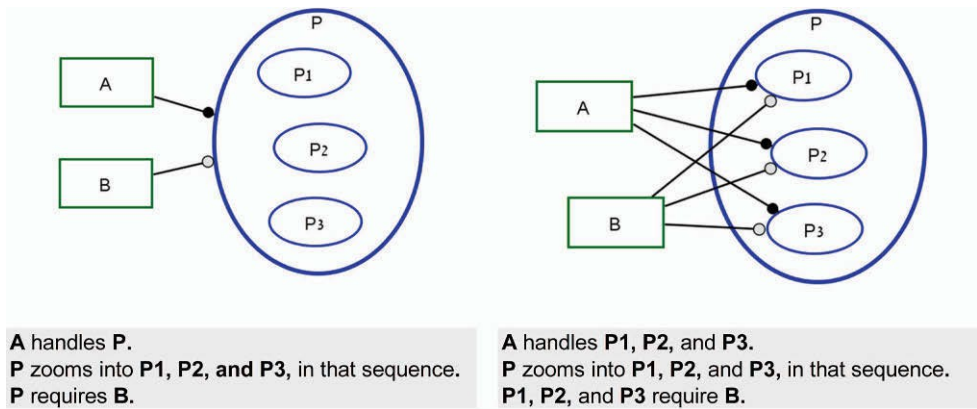
Pokud se aktivátor připojí k vnějšímu obrysu přibližného obrysu, musí se připojit alespoň k jednomu z jeho podprocesů. Spotřební a výsledkové odkazy nesmí být připojeny k vnějšímu obrysu přibližného procesu, protože to porušuje časové logické podmínky. S distribuovaným spotřebním odkazem by se pokusilo spotřebovat již spotřebovaný objekt podprocesem, který není prvním prováděným. Podobně by distribuovaný výsledkový odkaz pokusil vytvořit již existující instanci objektu. Modelář musí být opatrný, když více než jeden proces vytváří stejný objekt, tj. existuje více než jedna instance objektu, nebo dva nebo více procesů ovlivňuje nebo spotřebovává stejný objekt. Nástroje pro modelování OPM musí sledovat počet instancí objektu.

Tabulka 22.11 Shrnutí implicitních odkazů na vyvolání

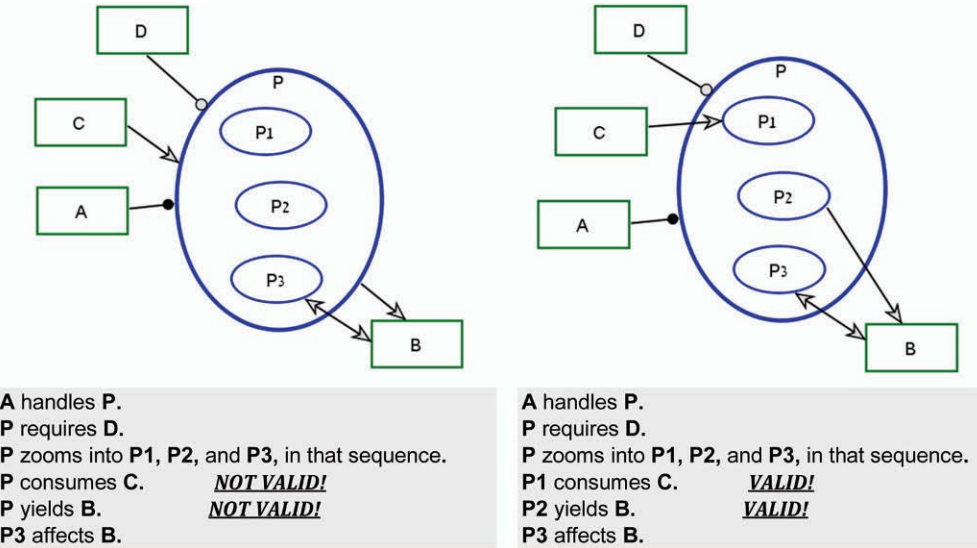
Name	Semantics	Sample OPD & OPL	Source	Destination
Implicit invocation link	Upon subprocess completion within the context of an in-zoomed process, the subprocess immediately invokes the one(s) below it.	 Product Terminating zooms into Product Finishing and Product Shipping, in that sequence.	Initiating process, whose ellipse top point is above the initiated process	Initiated process, whose ellipse top point is below the ellipse top point of the initiating process
Parallel implicit invocation link set	Top: Subprocesses A and B initiate in parallel as soon as Processing starts. Bottom: Subprocesses B and C initiate in parallel as soon as subprocess A ends.	 Processing zooms into A and parallel B and C, in that sequence.	Initiating process, whose ellipse top point is above the set of initiated processes, whose ellipse top points are at the same height (within a pre-determined tolerance).	A set of initiated processes, whose ellipse top points are at the same height (within tolerance) and below the initiating process ellipse top point

Na obr. 22.12 obsahuje OPD vlevo neplatné odkazy na spotřebu a výsledek, jak je poznamenáno v OPL. Odkaz na spotřebu vede k větě OPL „**P** spotřebovává **C**“. Důvodem je, že při použití distribuce odkazů je výsledkem tři věty OPL „**P1** spotřebovává **C**“, „**P2** spotřebovává **C**“ a „**P3** spotřebovává **C**“. Jelikož však **P1** spotřebovává **C** jako první podle časového pořadí, stejná instance **C** neexistuje, když se provádí **P2** nebo **P3**, a proto ani **P2**, ani **P3** nemohou **C** znovu spotřebovat. Podobně stejná instance **B** vede pouze k jednomu výsledku. OPD na pravé straně zobrazuje platné odkazy, protože specifikují, který z podprocesů **P** spotřebovává **C** (je to **P1**) a který vede k **B** (**P2**).

Protože připojení odkazu spotřeby nebo výsledku k přibližnému procesu je neplatné, jsou při přiblížení procesu všechny odkazy spotřeby a výsledku, které byly k němu připojeny, připojeny zpočátku nebo ve výchozím nastavení k jeho prvnímu podprocesu. Je odpovědností modeláře přesunout odkazy do následujících podprocesů podle potřeby.



Obr. 22.11 Rozložení odkazů v kontextu přiblížení. Vlevo: kratší, správná verze. Vpravo: ekvivalentní delší verze



Obr. 22.12 Omezení distribuce odkazů pro odkazy na spotřebu a výsledek

Jakmile modelář přiblíží **bod P** na [obr. 22.12](#) a vloží **bod P1** do jeho kontextu, modelovací nástroj by měl přesunout cílový konec spotřebního spojení vycházejícího z **bodu C** z **bodu P** do **bodu P1**. Podobně by se měl přesunout i zdrojový konec výsledného spojení do B z P do **P1**. Když modelář přidá **P2**, může přesunout cílový konec spotřebního spojení a/nebo zdrojový konec výsledného spojení z **P1** do **P2**, jak ukazuje [obr. 22.12](#).

22.10.4 Rozdělené páry spojů specifikované stavem

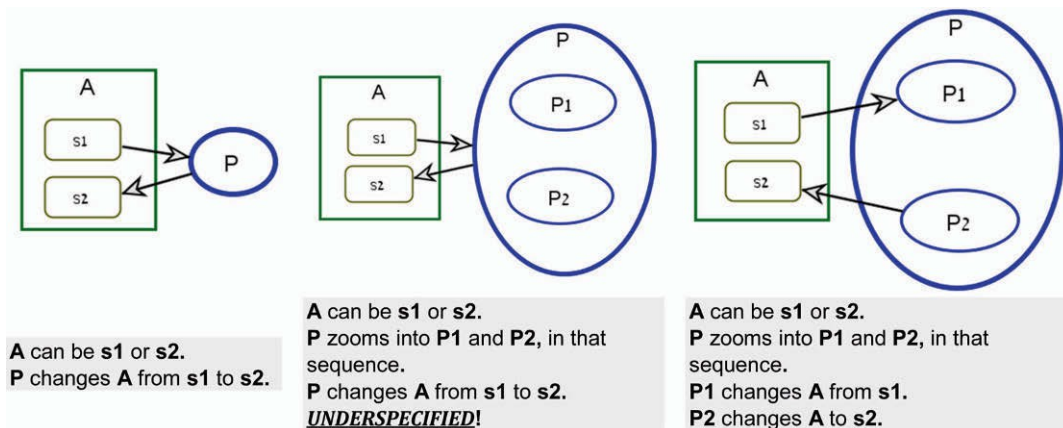
Když je proces, který mění objekt ze vstupního stavu na výstupní stav, přiblížen, OPD, ať už v diagramu nebo v novém diagramu, se stává nedostatečně specifikovaným. Aby modelář obnovil specifikaci, musí připojit jak vstupní spojení specifikované stavem, tak výstupní spojení specifikované stavem k jednomu z dílčích procesů časově proveditelným způsobem.

Rozdělená dvojice odkazů specifikovaných vstupem a výstupem procesu P je dvojice odkazů specifikovaných vstupem a výstupem, jejíž vstupní a výstupní odkazy spojují různé podprocesy P.

Rozdělené vstupní spojení je vstupní spojení rozděleného páru spojení specifikovaného vstupem a výstupem.

Rozdělený výstupní odkaz je výstupní odkaz rozděleného páru odkazů s určeným vstupem a výstupem.

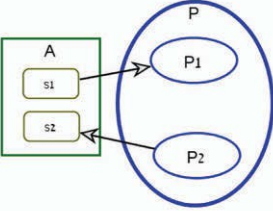
Na obr. 22.13 je OPD uprostřed nedostatečně specifikováno, protože pokud **P1** změní **A** z **s1** na **s2**, **P2** to nemůže udělat znovu, ale může to udělat opačně – změnit **A** ze **s2** zpět na **s1**, ale ani jedno z toho není explicitně specifikováno. **P1** může změnit **A** ze **s1**, tj. vzít ho ze **s1** a nechat ho v přechodu mezi **s1** a **s2**. Mezi **P1** a **P2** může být jeden nebo více dalších prozatímních podprocesů, během nichž je **A** stále v tomto přechodu. **P2** pak změní **A** na **s2**. OPD vpravo modeluje tento případ (bez prozatímních podprocesů) a vytváří rozdělené vstupní propojení z **s1 A** do **P1** a rozdělené výstupní propojení z **P2** do **s2**.



Obr. 22.13 Rozdělený stav specifikovaný transformujícím propojením vyřeší nedostatečnou specifikaci

Tabulka 22.12 shrnuje rozdělený pár efektových odkazů specifikovaných vstupem a výstupem. Neexistují žádné kontrolou modifikované verze rozděleného vstupu specifikovaného efektového propojení, protože to může způsobit zkreslení sémantiky efektového propojení. Například pokud je na obr. 22.13 **P1** přeskočeno, **A** zůstává v **s1**, takže pokud **P2** není přeskočeno, **A** nebylo vyjmuta z **s1**, takže se podle sémantiky efektového propojení nemůže změnit na **s2**.

Tabulka 22.12 Rozdělený vstup-výstup specifikovaný efektovy odkaz

Name	Semantics	Sample OPD & OPL	Source	Destination
Split input-output specified effect link pair <i>The top arrow:</i> split input link <i>The bottom arrow:</i> split output link	An early subprocess of an in-zoomed process takes an object out of its input state. A late subprocess of the same in-zoomed process changes the object to be in its output state.	 P1 changes A from s1. P2 changes A to s2.	<i>The top arrow:</i> Input state of an affected object <i>The bottom arrow:</i> Late subprocess of an in-zoomed process	<i>The top arrow:</i> Early subprocess of an in-zoomed process <i>The bottom arrow:</i> Output state of the affected object

22.11 Zapojené objekty Sada instancí Transformace

V důsledku distribuce odkazů platí pro operační instance transformovaných objektů následující omezení.

- Každá instance spotřebitele v sadě objektů předprocesu procesu přestane existovat na začátku nejpodrobnějšího podprocesu procesu, který instanci spotřebovává, takže instance není členem sady objektů postprocesu tohoto procesu.
- Každá instance ovlivněného objektu v sadě objektů předprocesu procesu, který mění tuto instanci v důsledku výkonu procesu, musí opustit svůj vstupní stav na začátku nejhlubšího (nejpodrobnějšího) podprocesu, který mění ovlivněný objekt.
- Každá instance ovlivněného objektu v sadě objektů postprocesu procesu, který mění tuto operační instanci v důsledku výkonu procesu, vstoupí do svého výstupního stavu po dokončení nejhlubšího podprocesu, který mění ovlivněný objekt.
- Každá instance výsledku v sadě objektů postprocesu procesu musí být vytvořena a začít existovat po dokončení nejpodrobnějšího podprocesu, který danou instanci výsledku vytvoří.

Stavový objekt B, u kterého má provedení procesu P za následek změnu stavu B, opouští vstupní stav na začátku nejpodrobnějšího podprocesu P, který mění B, a vstupuje do výstupního stavu na konci stejného podprocesu P nebo nějakého následného podprocesu P. Vzhledem k tomu, že provedení procesu P trvá určitou dobu, nachází se objekt B v přechodu mezi stavy, ze svého vstupního stavu do svého výstupního stavu: opustil svůj vstupní stav, ale ještě nedosáhl svého výstupního stavu.

22.12 Jazyk OCL (Object Constraint Language) v UML

Mini-jazyk OPM Parameterized Participation Constraint (PPC) popsaný v oddíle 17.3 poněkud připomíná jazyk Object Constraint Language (OCL) vyvinutý Warmerem a Kleppem (1998). OCL je „přesný textový jazyk, který poskytuje výrazy pro omezení a dotazy na objekty, které nelze vyjádřit diagramovou notací“. Aktuální verze OMG OCL (OMG OCL 2014) vysvětluje motivaci pro vývoj OCL tím, že „diagram UML, jako je například diagram tříd, obvykle není dostatečně propracovaný, aby poskytoval všechny relevantní aspekty specifikace. Mimo jiné je třeba popsat další omezení týkající se objektů v modelu. Taková omezení jsou často popsána v přirozeném jazyce. Praxe ukázala, že to vždy vede k nejednoznačnostem. ... OCL byl vyvinut, aby tuto mezeru zaplnil. Jedná se o formální jazyk, který zůstává snadno čitelný a zapisovatelný.“

Při srovnání mini-jazyka PPC OPM s OCL si všimneme, že zatímco OCL je kompletní jazyk, jehož aktuální specifikace OMG 2014 má 262 stran, mini-jazyk PPC lze specifikovat na několika stranách. Je vyjádřen v OPD a přeložen jako součást OPL a na rozdíl od OCL neumožňuje dotazování. Pokud jde o tvrzení, že OCL „zůstává snadno čitelný a zapisovatelný“, podívejme se na příklad omezení uvedený v OMG OCL (2014, s. 20):

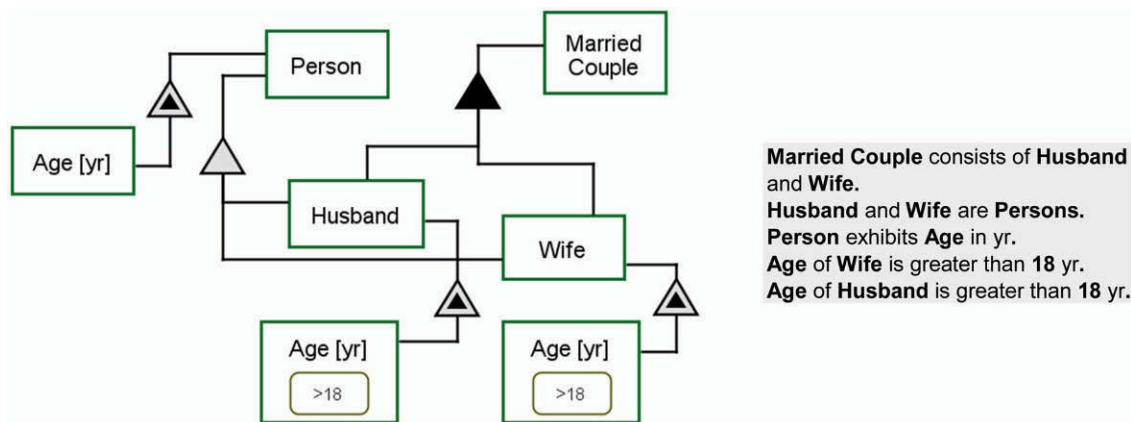
Manželé jsou plnoletí ≥ 18 . Syntaxe OCL pro toto omezení je následující.

kontext Osoba

inv: (self.wife->notEmpty()) **implies** self.wife.age ≥ 18

and (self.husband->notEmpty()) **implies** self.husband.age ≥ 18

Odpovídající model OPM je uveden na obr. 22.14. OPL tohoto modelu se zdá být o něco srozumitelnější než výše uvedená specifikace OCL.



Obr. 22.14 Model OPM omezení „Ženatí lidé jsou ve věku ≥ 18 “

22.13 Shrnutí

- **Událost** je okamžik, ve kterém dojde k něčemu významnému pro provádění systému.
- Události a předpoklady společně specifikují tok řízení provádění OPM pro výkon procesu podle paradigmatu událost-podmínka-akce.
- Paradigma **událost-podmínka-akce** stanoví, že zahájení výkonu procesu („akce“) má dvě předpoklady: iniciační událost a splnění předpokladu odvozeného z předprocesní sady objektů.
- **Modifikátor řízení** je jeden ze dvou dvoumístných symbolů **e** a **c**, který se přidává k procedurálnímu odkazu a přidává k sémantice tohoto odkazu sémantiku události a podmínky.
- **Kontrolní odkaz** je procedurální odkaz s přídatkem kontrolního modifikátoru.
- **Událostní odkaz** je procedurální odkaz s modifikátorem řízení **e**, který označuje zahájení procesu cíle odkazu a spouští vyhodnocení předběžných podmínek tohoto procesu.
- **Podmíněný odkaz** je procedurální odkaz s modifikátorem řízení **c**, který označuje, že pokud není splněna předběžná podmínka cílového procesu odkazu, je tento proces přeskočen.
- **Princip OPM o prioritě sémantiky přeskočení** stanoví, že sémantika přeskočení, vyvolaná kontrolním odkazem, má přednost před sémantikou čekání, vyvolanou nekontrolním odkazem.
- **Proces s maximální dobou trvání** je proces, pro který modelář stanoví maximální dobu trvání.
- **Proces zpracování přesčasu** je proces časové výjimky, který určuje, co dělat v případě, že časový výkon procesu s maximální dobou trvání překročí jeho maximální povolenou dobu.
- **Odkaz na výjimku z prodloužení** je procedurální odkaz z procesu s maximální dobou trvání na proces zpracování prodloužení, který označuje, že pokud doba trvání procesu s maximální dobou trvání překročí jeho maximální dobu trvání, je spuštěn proces výjimky z prodloužení.
- **Proces s minimální dobou trvání** je proces, pro který modelář stanoví minimální dobu trvání.
- **Proces zpracování nedodržení minimální doby** je proces časové výjimky, který určuje, co dělat v případě, že časový výkon procesu s minimální dobou nedosáhne své minimální doby trvání.
- **Odkaz na výjimku z nedodržení času** je procedurální odkaz z procesu s minimálním časem na proces výjimky z nedodržení času, který označuje, že pokud časový výkon časovaného procesu nedosáhne minimálního povoleného času, je spuštěn proces výjimky z nedodržení času.

22.14 Problémy

1. Proč je odkaz na událost na [obr. 3.5](#) potřebný?
2. Jaká je role podmíněného odkazu na [obr. 6.1](#)?
3. Vysvětlete, proč jsou na [obr. 7.1](#) potřeba dvě podmíněná propojení.
4. Použijte [obr. 21.15](#) jako šablonu a nahraďte v něm **B**, **P**, **P1** a **P2** smysluplnými věcmi.
5. Vysvětlete, proč je každá z pěti položek v [tabulce 21.1](#) označená jako „neplatná“ skutečně neplatná.
6. Vysvětlete, proč na [obr. 21.13 P123](#) (množina TO věcí, které se mají oddálit) nemůže obsahovat pouze **P4** a **BK**.
.
7. Co je třeba přidat k P4 a BK, aby se TO stalo platným?
8. Za předpokladu, že TO je množina, jak jste navrhli v předchozí otázce, nakreslete výsledný **SD1.1[nový]**.
9. Vytvořte model OPM nepřetržitého zavlažování vodou jako spotřebitelem pro proces zavlažování. Spotřebitel má atribut množství [litr] s hodnotou 1000 a spojení spotřeby má míru spotřeby [litr/sekunda] s hodnotou 50.
10. Vytvořte model OPM následujícího systému. **Benzín** a **motorová nafta** jsou výsledky procesu **rafinace**, který spotřebovává **ropu**. Výsledné produkty **benzín** a **motorová nafta** mají každý atribut **objem** [m3]. Výsledný odkaz **rafinace** na **benzín** má výtěžnost [m3/hod] s hodnotou **1000** a výsledný odkaz **rafinace** na **motorovou naftu** má výtěžnost [m3/hod] s hodnotou **800**. Za předpokladu, že je k dispozici dostatek **ropy**, bude-li **rafinace** aktivována a bude probíhat po dobu **10 hodin**, bude výtěžek **10 000** [m3] **benzinu** a **8 000** [m3] **ropy**.

Kapitola 23

Logické operátory a pravděpodobnosti

Logika a teorie pravděpodobnosti jsou dva hlavní nástroje formálního studia uvažování a byly úspěšně aplikovány v tak rozmanitých oblastech, jako je filozofie, umělá inteligence, kognitivní věda a matematika.

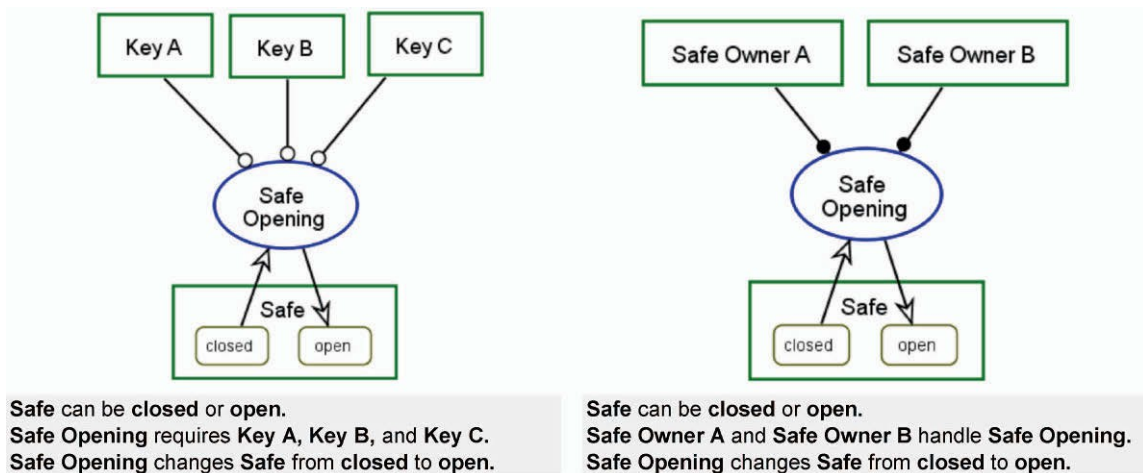
Stanfordská encyklopedie filozofie (2013)

Logické operátory, včetně AND, NOT, OR a XOR (exkluzivní OR), umožňují modelovat složité podmínky výkonu procesů. Pomocí XOR může OPM také přiřadit pravděpodobnosti k takovým výsledkům, jako je vytvoření jednoho z několika možných objektů nebo objektu v konkrétním stavu. O tom pojednáváme v této kapitole.

23.1 Logické AND procedurální odkazy

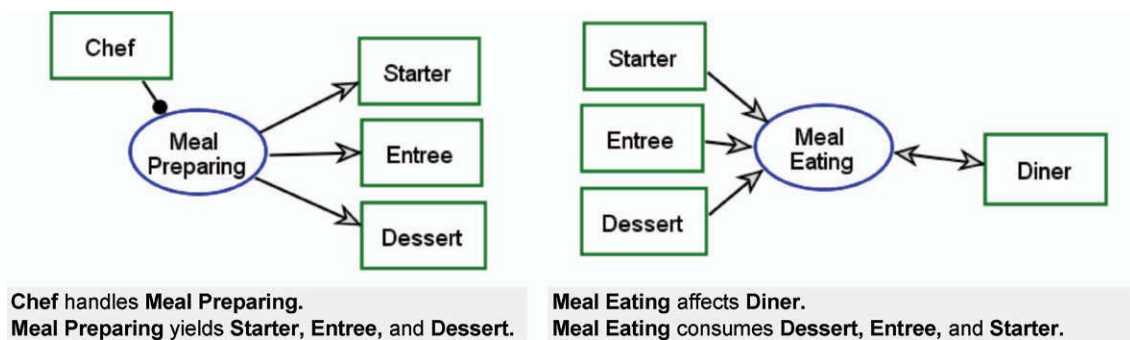
Dva nebo více procedurálních odkazů stejného druhu, které pocházejí z různých bodů podél obvodu procesní elipsy (kontext procesu) nebo do nich směřují, mají sémantiku logického operátoru AND. Graficky se odkazy se sémantikou AND na obrysu procesu nedotýkají. Tento operátor jsme dosud používali jako výchozí, aniž bychom to výslovně uváděli, protože se to jeví jako přirozené. Ve skutečnosti se v textu používá vyhrazená fráze OPL „and“ k vyjádření logického AND.

Následující tři příklady ukazují použití AND v různých procedurálních vazbách. V OPD na [obr. 23.1](#) (vpravo) vyžaduje proces **bezpečného otevření** jak **majitele trezoru A**, tak **majitele trezoru B**. Na [obr. 23.1](#) (vlevo) vyžaduje otevření **trezoru** všechny tři klíče.



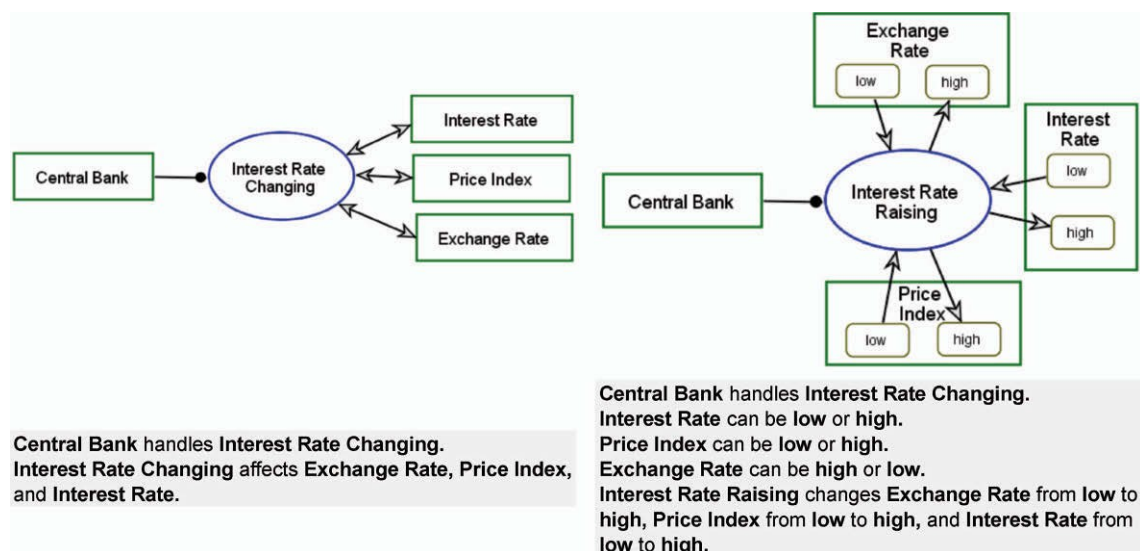
Obr. 23.1 Logické AND použité s vazbami agent a nástroj

Na [obr. 23.2](#) (vlevo) **přípravou jídla** vzniknou všechna tři jídla. Na [obr. 23.2](#) (vpravo) **konzumací jídla** spotřebuje všechna tři jídla.



Obr. 23.2 Logická funkce AND použitá s odkazy na výsledek a spotřebu

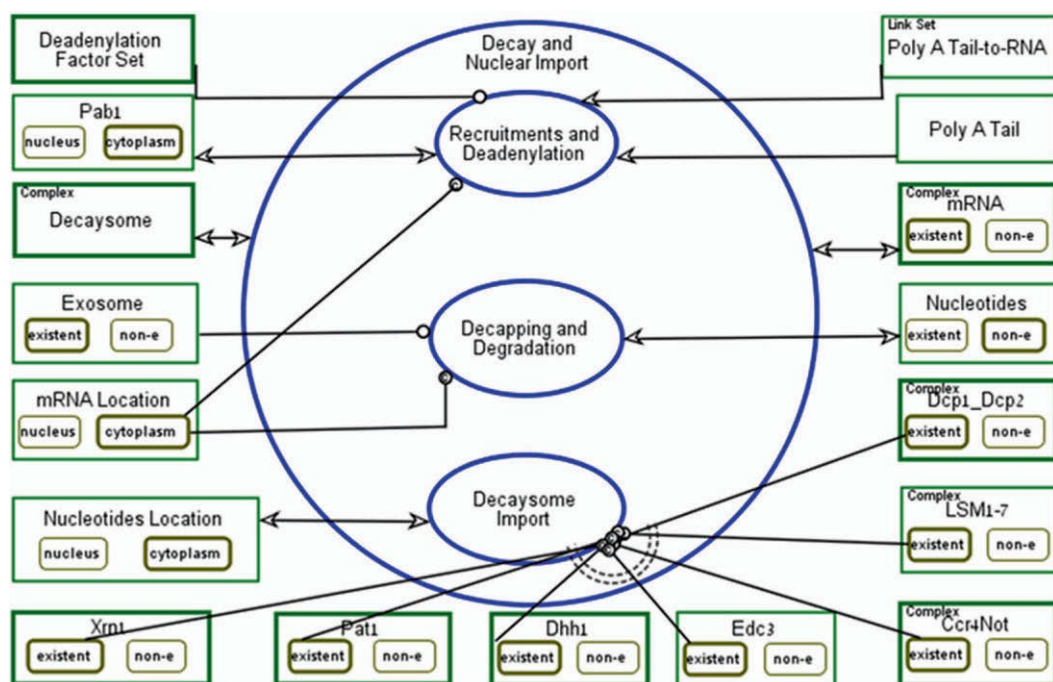
V OPD vlevo na [obr. 23.3](#) ovlivňuje **změna úrokové sazby** tři objekty: **směnný kurz**, **cenový index** a **úrokovou sazbu**. V OPD vpravo jsou všechny tři účinky **zvýšení úrokové sazby** na **směnný kurz**, **cenový index** a **úrokovou sazbu** explicitně vyjádřeny pomocí tří párů efektových vazeb s určeným vstupem a výstupem.



Obr. 23.3 Logické AND použité s efektovým propojením a s páry propojení specifikovaných jako vstup-výstup

23.2 Logické NOT

„NOT“ je unární logický operátor, který jednoduše obrací stav jakéhokoli booleanského objektu (viz oddíl 7.1): Binární vstup „ano“ (kladný, 1...) se převede na „ne“ (záporný, 0...) a naopak. Existuje několik způsobů, jak implementovat NOT v OPM. Jedním z nich je mechanismus flip-flop, popsáný v oddíle 19.5. Dalším způsobem je použití stavů jako omezení nebo podmínek pro provedení procesu. Pokud například chceme modelovat, že proces P se provede, pouze pokud látka S NENÍ přítomna, modelujeme objekt S se dvěma implicitními stavy: **existující** a **neexistující**. Stav **neexistující** propojíme s P pomocí propojení nástroje nebo propojení podmínky nástroje, takže P se může provést pouze v případě, že S je v **neexistujícím stavu**, tj. když neexistuje.



Obr. 23.4 Proces rozpadu mRNA a jaderného importu (Somekh et al. 2014) ukazující použití NOT prostřednictvím existujících a neexistujících stavů molekul

Proces rozpadu mRNA a jaderného importu je procesem zvětšeným na obr. 23.4 (Somekh et al. 2014). Tento OPD ukazuje, jak se **existující** a **neexistující** stavy molekul používají k implementaci „NOT“. Například existující stav komplexu **CCR4Not** (bez dvojitého významu), znázorněný v pravém dolním rohu, je spojen s **Decaysome Import** – třetím dílčím procesem shora, takže pouze pokud existuje **CCR4Not**, může tento dílčí proces proběhnout. V tomto případě však existuje šest dalších látek (například **Edc3**), které mohou každý z nich umožnit proces, a jsou propojeny logickým operátorem OR (diskutovaným níže), takže pouze nedostatek všech sedmi látek by zabránil výskytu **CCR4Not**. Pokud by **neexistující** (zkratka pro

non-existent) stavu **CCR4Not** byl propojen s podmínkou **Decaysome Import**, znamenalo by to (bez ohledu na ostatní propojení), že *absence* **CCR4Not** je podmínkou pro výskyt **Decaysome Import**.

23.3 Logické XOR a OR propojovací vějíře

Abychom graficky vyjádřili OR a XOR, používáme fanoušky spojení.

Fanoušek odkazů je sada f ($f \geq 2$) procedurálních odkazů stejného druhu, které pocházejí ze společného bodu nebo směřují do společného bodu na stejném objektu nebo procesu.

Konvergentní konec vějíře odkazů je konec, který je společný pro f odkazů vějíře.

Rozbíhavý konec vějíře odkazů je konec, který není společný pro f odkazů vějíře.

Konvergentní konec je připojen k jedné věci, zatímco divergentní konec je připojen k f věcem, kde f je velikost sady odkazů – počet odkazů ve sadě. Odkaz může být členem jak divergentní sady na svém zdroji, tak konvergentní sady na svém cíli.

Vzhledem k tomu, že odkazy jsou procedurální, jeden konec je připojen k objektu a druhý k procesům nebo naopak. Formálně je hodnota atributu **Vytrvalost věci** připojené ke konvergentnímu konci sady odkazů opačná než hodnota atributu **Vytrvalost f věcí** připojených k divergentnímu konci sady odkazů. Jak tedy ukazuje OPD na [obr. 23.5](#), pokud je hodnota atributu **Vytrvalost** věci připojená ke konvergentnímu konci vějíře odkazů **dynamická** (přechodná), pak je věc **procesem**. V tomto případě je hodnota atributu **Vytrvalost f věcí** připojená ke divergentnímu konci vějíře odkazů **statická** (trvalá), což znamená, že tyto f věci jsou všechny **objekty**.

23.3.1 Logický operátor XOR

Sémantika logického operátoru XOR spočívá v tom, že *přesně* jedna z f věcí připojených k divergentnímu konci vějíře odkazů je transformována, aktivována nebo se vyskytuje. Pokud je divergentní konec odkazu připojen k f objektům, pak *přesně* jeden objekt je transformován procesem na konvergentním konci vějíře odkazů nebo aktivuje tento proces. Pokud je divergentní konec odkazu připojen k f procesům, pak se vyskytuje přesně jeden proces.

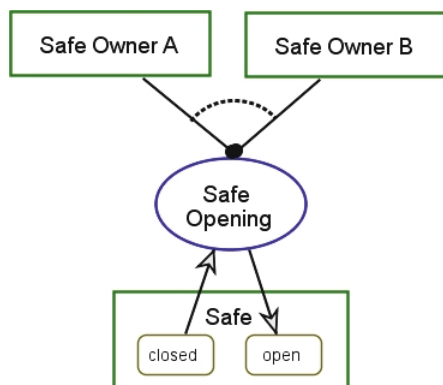
Toto použití operátoru XOR v OPM je v souladu s definicí XOR v digitálních systémech, ale může se lišit od některých interpretací binárního operátoru XOR s více vstupy, kde výstup je 1 pro lichý počet vstupů a 0 pro sudý počet vstupů. Graficky je operátor XOR znázorněn jako jediný přerušovaný oblouk přes f spojů vějíře spojů, jehož ohnisko je na konvergentním konci kontaktu (viz [obr. 23.5](#) vlevo).

Syntaxe fanoušku odkazů f věcí se sémantikou XOR se liší pro $f = 2$ a pro $f > 2$. Pro $f = 2$ se používá vyhrazený idiom (rozdělená vyhrazená fráze) „*bud’ ... nebo*“. Jelikož tento idiom je v přirozené angličtině vyhrazen pro vyjádření výběru přesně jedné ze dvou (ale ne více) položek, pro $f > 2$ se používá vyhrazená fráze „*přesně jedna z*“. =Například, protože na [obr. 23.5](#) (vlevo) sestává odkazový vějíř ze 2 odkazů agentů, $f > 2$, věta OPL zní:

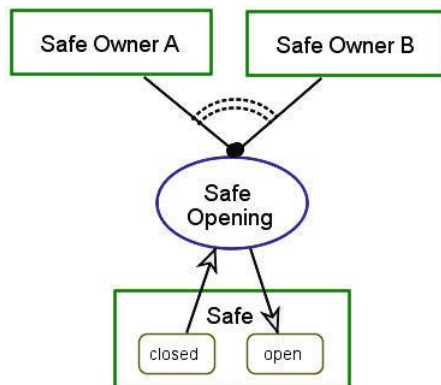
Otevírání trezoru zajišťuje buď majitel trezoru A, nebo majitel trezoru B.

Předpokládejme, že do vějíře je přidáno spojení agenta s třetím majitelem trezoru, **majitelem trezoru C**, takže $f=3$. Věta OPL pak zní:

Přesně jeden z **majitelů trezoru A**, **majitelů trezoru B** nebo **majitelů trezoru C** se stará o **otevírání trezoru**.



Trezor může být **uzavřený** nebo **otevřený**.
Otevírání trezoru provádí buď **majitel trezoru A**, nebo **majitel trezoru B**.
Otevření trezoru změní **stav trezoru** ze **zavřeného** na **otevřený**.



Trezor může být **uzavřený** nebo **otevřený**.
 Alespoň jeden z **majitelů trezoru A** a **majitelů trezoru B** manipuluje s **otevřením trezoru**.
Otevření trezoru změní **stav trezoru** z **uzavřeného** na **otevřený**.

Obr. 23.5 Příklady propojení agentů vyjadřující logické XOR (vlevo) a logické OR (vpravo)

23.3.2 Logický operátor OR

Sémantika logického operátoru OR spočívá v tom, že alespoň jedna z f věcí připojených k divergentnímu konci fanoušku odkazů je transformována, aktivována nebo se vyskytuje. Pokud je divergentní konec odkazu připojen k f objektům, pak alespoň jeden objekt je transformován procesem na konvergentním konci fanoušku odkazů nebo aktivuje tento proces. Pokud je divergentní konec odkazu připojen k f procesům, pak se alespoň jeden proces vyskytne. Toto použití operátoru OR v OPM je v souladu s binárním operátorem OR se dvěma nebo více vstupy.

Graficky je dvojitá přerušovaná křivka přes f spojů vějíře spojů, jehož ohnisko je na konvergentním konci kontaktu, označena operátorem OR (viz [obr. 23.5](#) vpravo).

Syntaxe vějíře odkazů f věcí se sémantikou OR je podobná pro $f = 2$ a $f > 2$. V obou případech se používá vyhrazená fráze „Alespoň jeden z“. Například na [obr. 23.5](#) (vpravo), kde vějíř odkazů obsahuje 2 odkazy agentů, je věta OPL:

Alespoň jeden z **majitelů trezoru A** nebo **majitelů trezoru B** se stará o **otevírání trezoru**.

Předpokládejme, že do vějíře je přidán odkaz na třetího vlastníka trezoru, **vlastníka trezoru C**, takže $f=3$. Věta OPL pak zní:

Alespoň jeden z **majitelů trezoru A**, **majitelů trezoru B** nebo **majitelů trezoru C** se stará o **otevírání trezoru**.

23.4 Rozbíhající se a sbíhající se odkazy XOR a OR

Konvergující vějíř je vějíř odkazů, jehož odkazy směřují k jeho konvergujícímu konci.

Rozbíhající se vějíř je vějíř odkazů, jehož odkazy směřují k jeho rozbíhající se části.

Tabulka 23.1 představuje shrnutí konvergujících spotřebních a výsledných odkazů XOR a OR pro $f \geq 2$ a v horním řádku ukazuje, že *konvergující spotřební odkazový vějíř* se vytvoří, když jsou zdrojové věci objekty a cílová věc je proces. V *konvergujícím vějíři výsledků* jsou zdrojové věci procesy a cílovou věcí je objekt. Naopak, jak ukazuje [tabulka 23.2](#), když je zdrojová věc objektem a cílové věci jsou procesy, dostaneme *divergující vějíř spotřebních odkazů*, zatímco když je zdrojová věc procesem a cílové věci jsou objekty, vytvoří se *divergující vějíř výsledků*.

Tabulka 23.1 Shrnutí konvergujících vějířů XOR a OR pro spotřební a výsledkové vazby

	XOR	OR
Converging consumption link fan	P consumes exactly one of A, B, or C.	P consumes at least one of A, B, or C.
Converging result link fan	Exactly one of P, Q, or R yields B.	At least one of P, Q, or R yields B.

Tabulka 23.2 Shrnutí divergujících vějířů XOR a OR pro spotřebu a výsledné odkazy

	XOR	OR
Rozbíhající se vějíř spotřebních odkazů	Přesně jeden z P, Q nebo R spotřebovává B.	Alespoň jeden z P, Q nebo R spotřebovává B.
Rozbíhající se výsledek spojení ventilátoru	Pdává přesně jeden z A, B nebo C.	Pdává alespoň jeden z A, B nebo C.

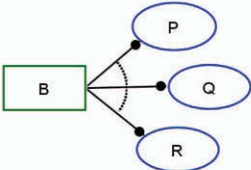
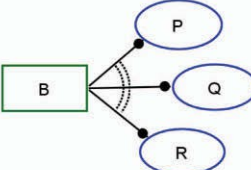
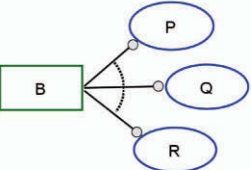
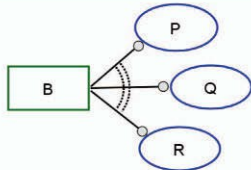
Efektové propojení je obousměrné, takže objekty propojené efektovým propojením jsou současně zdrojem i cílem, což ruší definice konvergentních a divergentních propojení. Jak ukazuje [tabulka 23.3](#), rozlišení se místo toho provádí podle počtu objektů nebo procesů, které propojení spojuje.

Tabulka 23.3 Shrnutí spojovacích efektových fanoušků XOR a OR

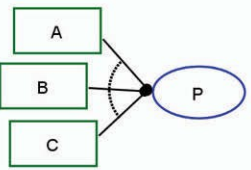
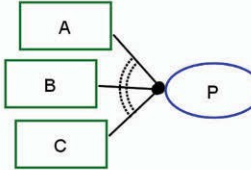
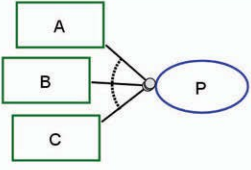
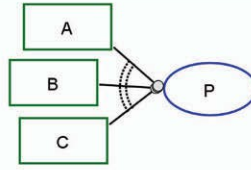
	XOR	OR
Multiple objects effect link fan	P affects exactly one of A, B, or C.	P affects at least one of A, B, or C.
Multiple processes effect link fan	Exactly one of P, Q, or R affects P.	At least one of P, Q, or R affects P.

Jelikož aktivátor je objekt, mohou být fanouškovské vazby agenta i nástroje divergentní, s více procesy jako cíli, jak ukazuje [tabulka 23.4](#), nebo konvergentní, s více aktivátory jako zdroji, jak ukazuje [tabulka 23.5](#).

Tabulka 23.4 Rozcházející se fanoušci agentů a nástrojů

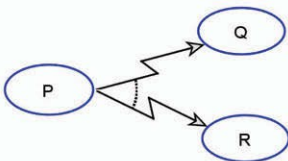
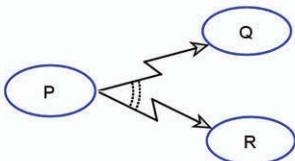
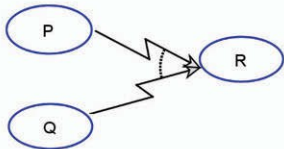
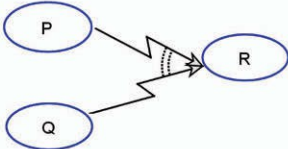
	XOR	OR
Diverging Agent link fan	 B handles exactly one of P, Q, or R.	 B handles at least one of P, Q, or R.
Diverging Instrument link fan	 Exactly one of P, Q, or R requires B.	 At least one of P, Q, or R requires B.

Tabulka 23.5 Konvergující fanoušci agentů a nástrojů

	XOR	OR
Converging Agent link fan	 Exactly one of A, B, or C handles P.	 At least one of A, B, or C handles P.
Converging Instrument link fan	 P requires exactly one of A, B, or C.	 P requires at least one of A, B, or C.

Odkazy na vyvolání mohou být také divergentní nebo konvergentní jak pro XOR, tak pro OR, jak je uvedeno v [tabulce 23.6](#), kde je specifikována sémantika sporných kombinací.

Tabulka 23.6 Fanoušci odkazů na vyvolání

	XOR	OR
Diverging invocation link fan	 P invokes either Q or R. Semantics: P invokes Q or R with probability 0.5 each. It is possible to assign specific probabilities along the links (see Section 23.7).	 P invokes at least one of Q or R.
Converging invocation link fan	 Either P or Q invokes R. Semantics: If both P and Q terminate at the same time, R is not invoked.	 At least one of P or Q invokes R. Semantics: This is the same as the OPD without the OR, because invocation is an event link, so its semantics is OR anyway (see Section 23.6).

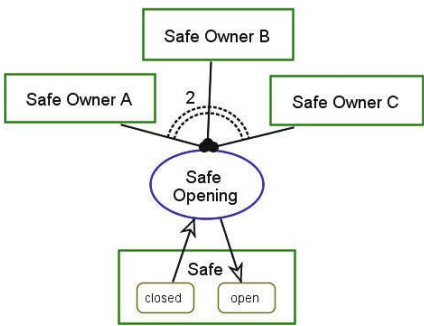
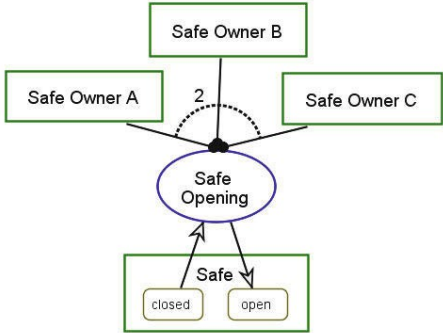
23.5 Kombinatorické XOR a kombinatorické OR

Dosud představená logika XOR a OR implikuje výběr přesně *jednoho* (pro XOR) nebo alespoň *jednoho* (pro OR). V případech, kdy velikost vějíře $f > 2$, můžeme logiku XOR a OR zobecnit na kombinatorickou logiku XOR a kombinatorickou logiku OR. Logiku rozšiřujeme z 1 na libovolný počet m odkazů (až o jeden méně než f) nahrazením „jednoho“ ve větě OPL m , kde $m \leq f$.

23.5.1 Kombinatorická XOR

Zvažte následující větu OPL, která rozšiřuje model na [obr. 23.5](#).

Přesně jeden z majitelů trezoru A, majitelů trezoru B nebo majitelů trezoru C provádí otevření trezoru.



Trezor může být uzavřený nebo otevřený. Přesně 2 z majitelů trezoru A, majitelů trezoru B nebo majitelů trezoru C se starají o otevírání trezoru. Otevření trezoru změní stav trezoru ze zavřeného na otevřený.

Trezor může být uzavřený nebo otevřený. Otevření trezoru zajišťují alespoň 2 z následujících osob: majitel trezoru A, majitel trezoru B nebo majitel trezoru C. Otevření trezoru změní trezor ze zavřeného na otevřený.

Obr. 23.5 Příklad kombinatorického XOR (vlevo) a kombinatorického OR (vpravo)

Velikost fanouškovského odkazu je zde $f=3$. Pokud chceme modelovat, že k otevření trezoru jsou potřeba přesně dva majitelé trezoru, místo „jeden“ napíšeme $m=2$, čímž efektivně zavedeme kombinatorický počet možností $3 \text{ z } 2$, $\binom{3}{2}=3$:

2

Přesně 2 z majitelů trezoru A, majitelů trezoru B nebo majitelů trezoru C se podílejí na otevření trezoru.

V OPD přidáme číslo m mimo a vedle oblouku XOR, jak ukazuje číslo 2 zaznamenané v OPD vlevo na [obr. 23.6](#).

Obecně v kombinatorickém XOR omezuje model tak, aby vybral přesně m z f odkazů, používáme vyhrazenou frázi „přesně m z“, kde $m \leq f$, a počet možností je $\binom{f}{m}$.

m

23.5.2 Kombinatorické OR

Podobně jako u kombinatorického XOR zobecňujeme logiku OR na kombinatorické OR. Činíme tak rozšířením logiky z 1 na libovolný počet m (až o jeden méně než f) odkazů nahrazením „alespoň jeden z“ ve větě OPL výrazem „alespoň m z“, kde $m < f$. Použijeme-li opět výše uvedenou větu OPL, která rozšiřuje model na [obr. 23.5](#), kde velikost vějíře odkazů je $f=3$, místo „jeden“ můžeme napsat $m=2$, čímž efektivně zavedeme součet kombinatorického počtu možností.

Otevírání trezoru zajišťují alespoň 2 z majitelů trezoru A, B nebo C.

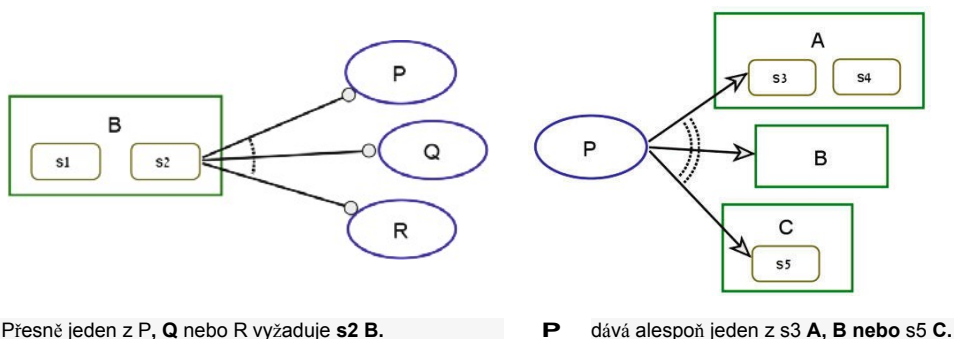
V tomto případě je počet možností $\binom{3}{2} + \binom{3}{3} = 3 + 1 = 4$. V OPD přidáme číslo m mimo a vedle oblouku OR, jak ukazuje číslo 2 zaznamenané v OPD vpravo na [obr. 23.6](#).

Obecně platí, že pro omezení modelu tak, aby vybral alespoň m z f odkazů, používáme vyhrazenou frázi „alespoň m z“, kde $m < f$ a počet možností je $\binom{f}{m} + \binom{f}{m+1} + \dots + \binom{f}{f}$.

$\binom{f}{m} + \binom{f}{m+1} + \dots + \binom{f}{f}$

23.6 Stavově specifikované XOR a OR odkazové vějíře

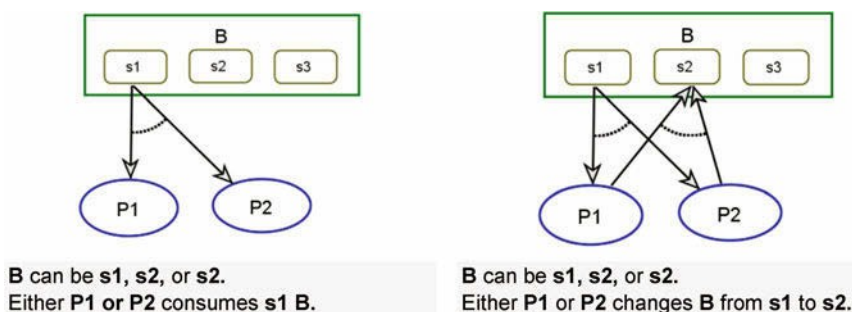
Každý z výše popsaných fanoušků odkazů má odpovídající verzi specifikovanou stavem, kde zdroj a cíl mohou být specifické stavy objektů nebo objekty bez specifikace stavu. Mohou se vyskytnout kombinace odkazů specifikovaných stavem a bez stavu jako cíle fanouška odkazů. [Obrázek 23.7](#) ukazuje vlevo vějíř odkazů nástrojů XOR specifikovaných stavem a vpravo vějíř odkazů OR se smíšeným výsledkem, kde jsou odkazy specifikovány stavem pro objekty A a C, ale ne pro B.



Obr. 23.7 Příklady odkazů XOR (vlevo) a OR (vpravo) specifikovaných stavem

Dva nebo více procesů může mít stejný stav jako jejich zdroj. Například, jak ukazuje OPD na pravé straně [obr. 23.8](#), buď P1, nebo P2 (ale ne oba) mohou spotřebovat B, když je ve stavu s1: Buď P1, nebo P2 spotřebuje s1 B. Pokud existují více než dva procesy, věta OPL se změní na: Presně jeden z P1, P2 nebo P3 spotřebovává s1 B. Podobná situace nastává při změně stavu v OPD na pravé straně [obr.](#)

[23.8](#): Buď P1, nebo P2 změní B ze s1 na s2. A pro více než dva procesy: Presně jeden z P1, P2 nebo P3 změní B ze stavu s1 na s2.



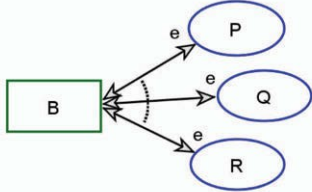
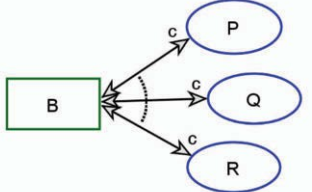
Obr. 23.8 Vlevo: P1 XOR P2 může spotřebovat B, když je ve stavu s1. Vpravo: P1 XOR P2 může změnit B ze s1 na s2

23.6.1 Ovládací modifikované propojovací vějíře

Každý z XOR odboček pro spotřebu, výsledek, efekt a povolení odboček a jejich verzí specifikovaných stavem má odpovídající odbočku s modifikovaným řízením: odbočku události a odbočku podmínky. [Tabulka](#)

23.7 představuje fanoušky propojení událostí a podmínek jako zástupce základní (nestátem specifikované) verze propojení modifikovaných fanoušků propojení.

Tabulka 23.7 Fanoušci odkazů na události a podmínky XOR

Event	Condition
	
B initiates exactly one of P, Q, or R, which affects the occurring process.	Exactly one of P, Q, or R occurs if B exists, in which case the occurring process affects B, otherwise these processes are skipped.

23.6.2 Státem specifikované kontrolní modifikované fanoušky odkazů

Každý z fanoušků odkazů upravených kontrolou, s výjimkou fanoušků odkazů upravených kontrolou, má odpovídající fanoušky odkazů upravené kontrolou specifikované stavem. Jelikož jsou tyto verze specifikované stavem složitější než jejich verze nespecifikované stavem, [tabulka 23.8](#) představuje OPD a OPL případů specifikovaných stavem a pod každým takovým případem je uvedena věta OPL pro odpovídající případ bez stavu.

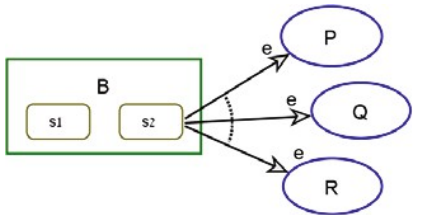
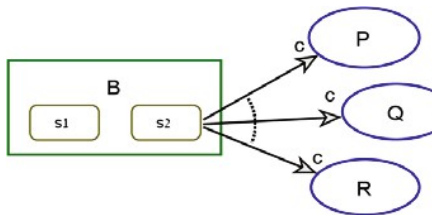
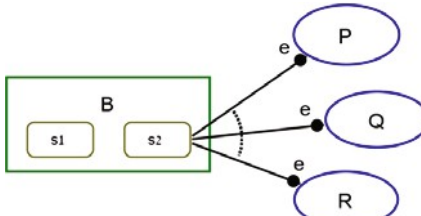
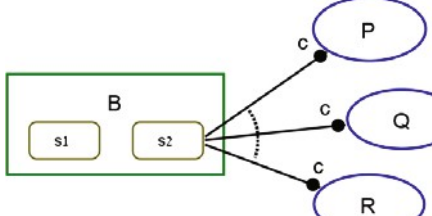
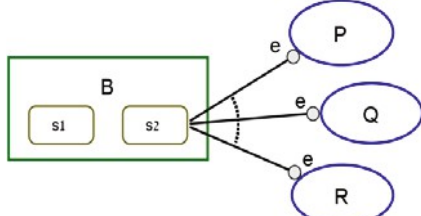
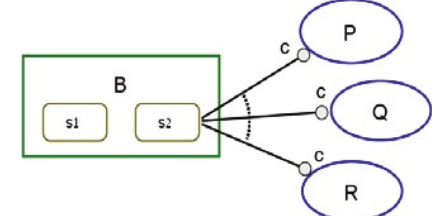
Každý XOR link fan v [tabulce 23.7](#) a v [tabulce 23.8](#) má svůj OR protějšek (označený dvojitou přerušovanou křivkou) s odpovídající větou OPL, ve které je vyhrazený výraz „alespoň“ nahrazen výrazem „přesně“.

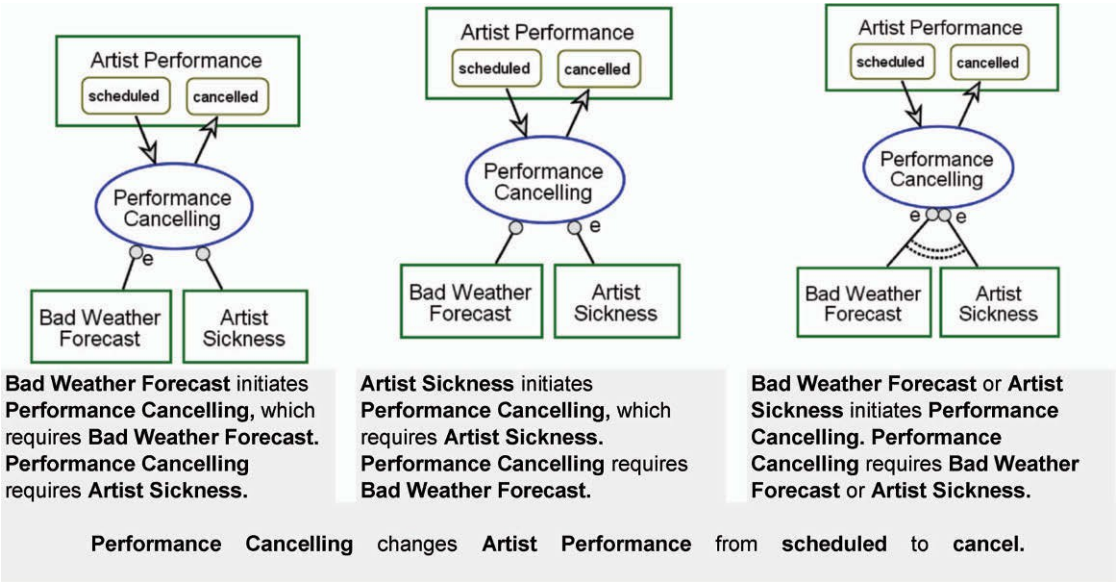
23.7 Více kontrolních odkazů má sémantiku OR

Událost spouští proces nezávisle na jakémkoli jiném odkazovém odkazu, který může být propojen se stejným procesem. Proto mají dva nebo více odkazových odkazů připojených k procesu logickou sémantiku OR. **Zrušení** na [obr. 23.9](#) může být iniciováno (spuštěno) **špatnou předpovědí počasí** nebo **nemocí umělce**. Není nutné, aby existovaly obě současně. Ve skutečnosti je pravděpodobnost, že tyto dva objekty budou vytvořeny v systému ve stejný okamžik, prakticky nulová. Proto je OPD na pravé straně [obr. 23.9](#) správné. Ten vlevo je případ, kdy událost, která iniciuje **zrušení**, je **špatná předpověď počasí**, ale v takovém případě je také nutná **nemoc umělce**. OPD uprostřed je doplňkový případ: událost, která iniciuje **zrušení**, je **nemoc umělce**, ale v takovém případě je také nutná **špatná předpověď počasí**.

Podobně, pokud je cílem procesu P se sémantikou AND více než jedna podmínka, musí být všechny podmínky splněny, aby proces P nebyl přeskočen. Předpokládejme, že podmínky jsou C1, C2 a C3. Stačí, aby jedna podmínka nebyla splněna, aby byl proces P přeskočen: C1 nebo C2 nebo C3. Zatímco sémantika AND platí z hlediska požadavků na výkon procesu, z hlediska sémantiky přeskočení se jedná o sémantiku OR. Pokud chceme modelovat, že stačí jakákoli neprázdná podmnožina podmínek, musíme použít propojovací vějíř OR, jak bylo provedeno v modelu na [obr. 23.5](#).

Tabulka 23.8 Stavby specifikované a bezstavové XOR kontrolou modifikované spojovací vějíře

Druh spojovacího vějíře	Modifikátor řízení událostí	Modifikátor řízení podmínek
Fan odkazů se specifikací stavu	 S2 B iniciuje přesně jeden z P, Q nebo R, který spotřebovává iniciovaný proces. <i>Případ bez stavu:</i> B iniciuje přesně jeden z P, Q nebo R, který spotřebovává iniciovaný proces.	 Přesně jedna z funkcí P, Q nebo R se spustí, pokud je B s2, v takovém případě spuštěný proces spotřebuje B, jinak jsou tyto procesy přeskočeny. <i>Případ bez stavu:</i> Přesně jedna z hodnot P, Q nebo R nastane, pokud existuje B, v takovém případě nastalý proces spotřebuje B, v opačném případě jsou tyto procesy přeskočeny.
Stavově specifikovaný odkaz agenta	 S2 B iniciuje a zpracovává přesně jeden z P, Q nebo R. <i>Případ bez stavu:</i> B iniciuje a zpracovává přesně jeden z P, Q nebo R.	 B zpracovává přesně jeden z P, Q nebo R, pokud je B s2, jinak jsou tyto procesy přeskočeny. <i>Případ bez stavu:</i> B zpracovává přesně jeden z P, Q nebo R, pokud B existuje, jinak jsou tyto procesy přeskočeny.
Stavově specifikovaný nástrojový odkaz fan	 S2 B iniciuje přesně jeden z P, Q nebo R, který vyžaduje s2 B. <i>Případ bez stavu:</i> S2 B spouští přesně jeden z P, Q nebo R, což vyžaduje B.	 Přesně jeden z P, Q nebo R vyžaduje, aby B bylo s2, jinak jsou tyto procesy přeskočeny. <i>Případ bez stavu:</i> Přesně jedna z hodnot P, Q nebo R vyžaduje, aby B bylo s2, jinak jsou tyto procesy přeskočeny.

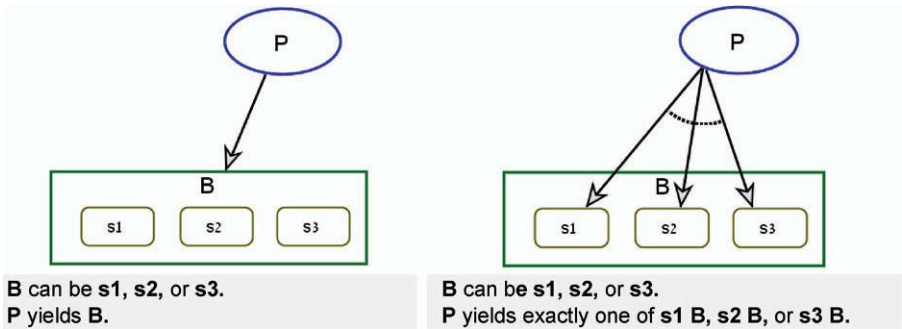


Obr. 23.9 Odkaz události má sémantiku OR (vpravo), protože je nepravděpodobné, že by se staly ve stejný okamžik.

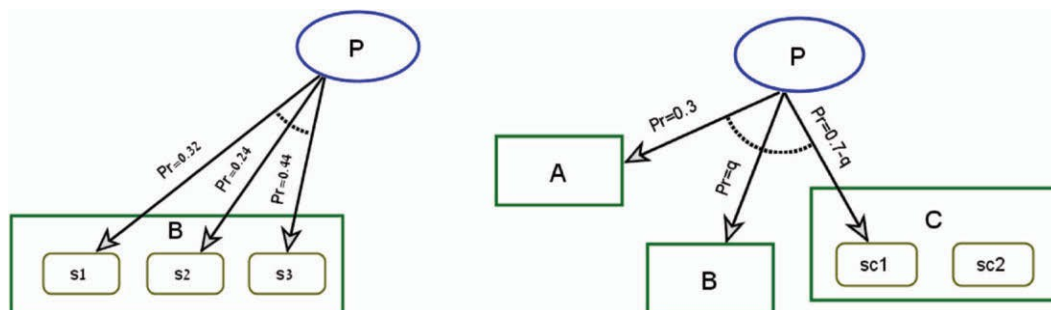
23.8 Pravděpodobnosti propojení a pravděpodobnostní propojovací vějíře

Proces **P** s výslednou vazbou, která vede k objektu **B** se stavem s n stavy, s_1 až s_n , bez specifikace konkrétního stavu, jako v OPD vlevo na [obr. 23.9](#), znamená, že pravděpodobnost generování **B** v jakémkoli konkrétním stavu bude $1/n$. V tomto případě nahrazuje jediná výsledková vazba na objekt výsledkovou vazbu fan na každý z jeho stavů, takže OPD vlevo na [obr. 23.9](#) je ekvivalentní a, protože je jednodušší než ten vpravo, je preferovanou verzí.

V levé části OPD na [obr. 23.10](#) znamená výsledný odkaz z **P** do **B**, který má 3 stavy, že **P** vytvoří **B** se stejnou pravděpodobností, $Pr = 1/3$, pro vytvoření v každém ze tří stavů.



Obr. 23.10 Ekvivalence mezi výsledným propojením a sadou výsledných propojení specifikovaných stavem XOR



P vytvoří **s1 B** s pravděpodobností **0,32**, **s2 B** s pravděpodobností **0,24** nebo **s3 B** s pravděpodobností **0,44**.

(Analogický) deterministický případ:

P dává přesně jeden z **s1 B**, **s2 B** nebo **s3 B**.

P dává **A** s pravděpodobností **0,3**, **B** s pravděpodobností **q** nebo **sc1 C** s pravděpodobností **0,7-q**.

Analogický deterministický případ:

P dává přesně jeden z **A**, **B** nebo **sc1 C**.

Obr. 23.11 Příklady pravděpodobnostního vytváření objektů specifikovaných stavem

Obecně platí, že pravděpodobnosti následování konkrétního odkazu v odkazu fan nejsou stejné.

Pravděpodobnost odkazu je volitelná hodnota atributu přiřazená procedurálnímu odkazu v rozvětveném odkazu XOR, která určuje pravděpodobnost následování tohoto konkrétního odkazu mezi možnými odkazy v rozvětveném odkazu.

Pravděpodobnostní vějíř odkazů je vějíř odkazů s hodnotou pravděpodobnosti přiřazenou každému z jeho odkazů tak, že součet hodnot pravděpodobnosti všech odkazů je přesně 1.

Graficky lze pravděpodobnostní vějíř odkazů znázornit jako pravděpodobnostní hodnotu ve tvaru $Pr = p_i$, kde p_i je číselná hodnota pravděpodobnosti odkazu

nebo parametr, takže $\sum p_i = 1$. Tento symbol $Pr = p_i$, který se objevuje podél každého z f odkazů v pravděpodobnostním odkazu, označuje pravděpodobnost, že mechanismus řízení provádění systému vybere tento konkrétní odkaz a bude následovat tuto cestu.

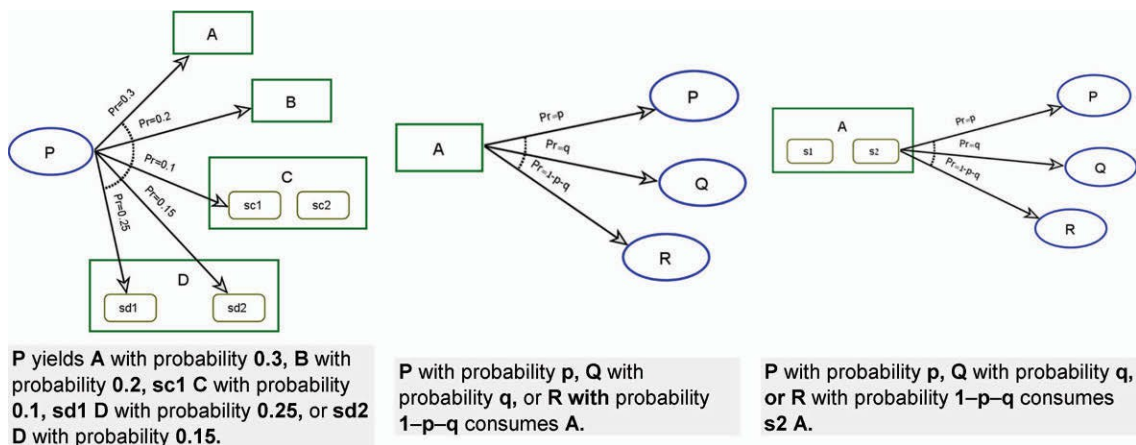
Odpovídající věta OPL je věta OPL divergujícího fanoušku odkazů XOR bez pravděpodobností odkazů, ve které je vynechána fráze „přesně jeden z...“ a místo toho je přidána fráze „...s pravděpodobností p_i “ za každým názvem zúčastněné věci s poznámkou o pravděpodobnosti $Pr = p_i$.

Obrázek 23.11 ukazuje dva příklady pravděpodobnostního vytváření objektů specifikovaných stavem a jejich deterministické analogy v OPL. V OPD vlevo může proces **P** vytvořit objekt **B** ve třech možných stavech, **s1**, **s2** nebo **s3**, s odpovídajícími pravděpodobnostmi **0,32**, **0,24** a **0,44** (celkem 1), jak je uvedeno podél každého výsledného odkazu výsledného odkazu. V OPD vpravo může **P** vytvořit jeden z objektů **A**, **B** nebo **sc1 C**, tj. **C** ve stavu **sc1**, s pravděpodobnostmi **0,3**, **q** a **0,7-q** (celkem 1).

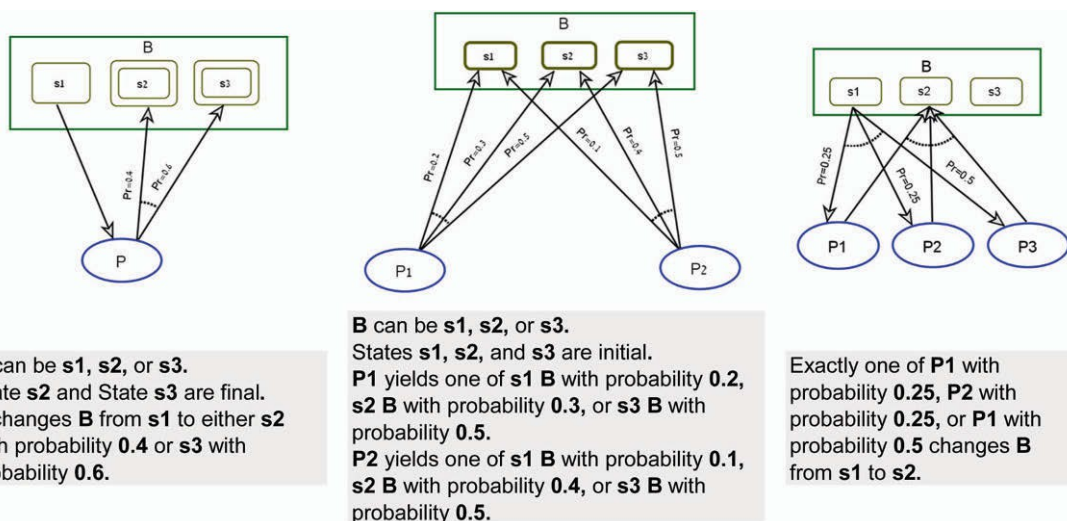
Pro proces **P** s výsledným odkazem, který vede k objektu **B** se stavy **s1** až **sn** a s počátečním stavem **si**, vytvoří **P** objekt **B** ve stavu **si** s pravděpodobností 1. Pokud má **B** $m < n$ počátečních stavů, vytvoří **P** objekt **B** v jednom z počátečních stavů s pravděpodobností $1/m$.

U pravděpodobnostního výsledného fanoušku odkazů může být kterýkoli z výsledků objektem bez specifikovaného stavu nebo s ním. U všech fanoušků odkazů zahrnujících jiné procedurální typy odkazů (včetně těch s modifikátory kontroly stavu události a

modifikátory řízení podmínek), kde cíle odkazů v odkazu jsou procesy, může být zdrojem objekt nebo specifikovaný stav objektu.



Obr. 23.12 Objekty se specifikovanými stavy a bez specifikovaných stavů jako výsledné a spotřebované prvky pravděpodobnostního propojení



Obr. 23.13 Příklady různých pravděpodobnostních změn se specifikovaným stavem: ze stavu do jednoho ze dvou konečných stavů (vlevo), pravděpodobnostní výsledek do jednoho ze tří konečných stavů (uprostřed) a pravděpodobnostní změna z jednoho stavu do druhého (vpravo)

OPD na levé straně obr. 23.12 ukazuje pravděpodobnostní výsledek spojovacího větvě, ve kterém **P** dává jeden z objektů **A** nebo **B**, nebo **C** ve stavu **sc1**, nebo **D** ve stavu **sd1** nebo **sd2**, každý se svými specifikovanými pravděpodobnostmi. OPD uprostřed obr. 23.12 ukazuje pravděpodobnostní spotřební spojovací větví, ve kterém je spotřebován **A**, s

určenými pravděpodobnostmi, jedním z procesů **P** nebo **Q** nebo **R**. OPD ve spodní části vyjadřuje totéž, s tím, že **A** musí být ve stavu **s2**.

Obrázek 23.13 představuje příklady různých pravděpodobnostních transformací specifikovaných stavem. Vlevo je změna stavu z jednoho stavu do jednoho ze dvou konečných stavů. Uprostřed je pravděpodobnostní vytvoření (výsledek) a vpravo je pravděpodobnostní změna z jednoho stavu do druhého.

23.9 Shrnutí

- **Logické operátory**, včetně AND, OR a XOR (exkluzivní OR), umožňují modelovat složité podmínky výkonu procesů.
- Dvě nebo více procedurálních vazeb stejného druhu, které pocházejí z různých bodů podél obvodu elipsy procesu (kontext procesu) nebo do nich směřují, mají sémantiku **logického** operátoru **AND**.
- **Fanoušek odkazů** je sada $f \geq 2$ procedurálních odkazů stejného druhu, které pocházejí ze společného bodu nebo směřují do společného bodu na stejném objektu nebo procesu.
- **Konvergentní konec** vějíře spojů je konec, který je společný pro f spojů vějíře.
- **Rozbíhavý konec** vějíře odkazů je konec, který není společný pro odkazy vějíře f .
- Fanoušek odkazů s jedním přerušovaným obloukem označuje **logický** operátor **XOR**.
- Fanoušek odkazů s dvojitým přerušovaným obloukem označuje **logický** operátor **OR**.
- **Konvergující vějíř** je vějíř odkazů, jehož odkazy směřují k jeho konvergujícímu konci.
- **Rozbíhající se vějíř** je vějíř odkazů, jehož odkazy směřují k jeho rozbíhající se části.
- Každý z XOR odkazových vějířů pro spotřebu, výsledek, efekt a povolení odkazů a jejich verzí specifikovaných stavem má odpovídající **kontrolou modifikovaný odkazový vějíř**: odkazový vějíř událostí a odkazový vějíř podmínek.
- **Pravděpodobnost odkazu** je volitelná hodnota atributu přiřazená procedurálnímu odkazu v rozbíhavém odkazu XOR, která určuje pravděpodobnost následování tohoto konkrétního odkazu mezi možnými odkazy v odkazu vějíře.
- **Pravděpodobnostní fan odkazů** je fan odkazů s hodnotou pravděpodobnosti přiřazenou každému z jeho odkazů tak, že součet hodnot pravděpodobnosti všech odkazů je přesně 1.

23.10 Problémy

1. Zkombinujte dva OPD na [obr. 23.1](#), abyste vyjádřili, že každý z obou majitelů trezoru musí mít všechny tři klíče, aby mohl trezor otevřít.
2. Spojte dva OPD na [obr. 23.2](#), abyste vyjádřili, že kuchař připravuje buď hlavní chod, nebo předkrm a dezert, a host jí to, co je připraveno.

3. V levém horním a pravém dolním OPD v [tabulce 23.1](#) nahraďte názvy věcí obsahem, který vytvoří smysluplné věty OPL.
4. Opakujte předchozí otázku pro [tabulku 23.2](#).
5. Udělejte totéž pro jeden OPD v každé z [tabulek 23.3](#), [23.4](#) a [23.5](#).
6. Vyberte tři libovolné OPD z posledních tří otázek a přiřaďte jim pravděpodobnosti. V případě potřeby je upravte.

Kapitola 24

Přehled normy ISO 19450

Tato kniha obsahuje komplexní popis OPM, který je kompatibilní s veřejně dostupnou specifikací (PAS) ISO 19450 s názvem „**Automatizační systémy a integrace – Metodika objektově-procesní**“ a ve francouzštině: „**Systèmes d'automatisation et intégration – Méthodologie du processus-objet**“. ISO 19450 PAS byla přijata Mezinárodní organizací pro normalizaci (ISO) v prosinci 2015 na základě práce technického výboru ISO 184/podvýboru 5 (TC184/SC5) po šestiletém úsilí, zejména Richarda Martina, Davida Shortera, Alexe Blekhmana a autora této knihy. Tato kniha byla připravena souběžně s normou ISO 19450 PAS, takže jsou obě téměř zcela v souladu. Vzhledem k tomu, že norma (formálně PAS) musí být v souladu s pravidly ISO pro tvorbu norem, má odlišnou strukturu a není tak propracovaná jako kniha. Jedná se spíše o přehledné vysvětlení OPM, které umožňuje vývojářům nástrojů použít ji spolu s touto knihou jako pevný základ pro vývoj softwarového nástroje kompatibilního s ISO 19450 na podporu koncepčního modelování založeného na OPM. Normy ISO, jako je ISO 19450 PAS, obsahují normativní části a často také jednu nebo více informativních částí. Aby byla norma dodržena, musí být normativní část přísně dodržována, zatímco informativní část není povinná. Tato kniha je nadmnožinou normy ISO 19450 PAS. Přibližně 90 % obsahu této knihy je v souladu s normou ISO 19450. Zbytek lze považovat za ekvivalent doplňku k informativní části normy – měl by být dodržován, ale norma ISO 19450 ve své současné počáteční podobě to nevyžaduje. Tato závěrečná kapitola stručně popisuje obsah normy ISO 19450 PAS, kde každá část je věnována shrnutí jedné nebo více částí normy ISO 19450.

24.1 Úvod normy ISO 19450

První odstavec úvodu dokumentu ISO 19450 (str. v) zní následovně.

Metodika objektově-procesního modelování (OPM) je kompaktní koncepční přístup, jazyk a metodika pro modelování a reprezentaci znalostí automatizačních systémů. Aplikace OPM sahá od jednoduchých sestav elementárních komponent až po komplexní, multidisciplinární, dynamické systémy. OPM je vhodná pro implementaci a podporu nástroji využívajícími informační a počítačové technologie. Tento dokument specifikuje jak jazykové, tak metodologické aspekty OPM s cílem vytvořit společný základ pro systémové architekty, designéry a vývojáře nástrojů kompatibilních s OPM pro modelování všech druhů systémů.

Úvod dále pojednává o obecnosti a použitelnosti OPM v průmyslu a podnikání jako základu pro modelové systémové inženýrství:

Notace OPM podporuje koncepční modelování systémů s formální syntaxí a sémantikou. Tato formálnost slouží jako základ pro modelové systémové inženýrství obecně, včetně architektury systémů, inženýrství, vývoje, podpory životního cyklu, komunikace a evoluce. Kromě toho doménově nezávislá povaha OPM otevírá modelování systémů celé vědecké, obchodní a průmyslové komunitě pro vývoj, výzkum a analýzu výrobních a jiných průmyslových a obchodních systémů v rámci jejich specifických aplikačních domén, čímž umožňuje společností slučovat a zajišťovat interoperabilitu různých dovedností a kompetencí do společného intuitivního, ale formálního rámce.

OPM usnadňuje společný pohled na systém ve fázi výstavby, testování, integrace a každodenní údržby a umožňuje práci v multidisciplinárním prostředí. Navíc pomocí OPM mohou společnosti zlepšit svůj celkový pohled na funkčnost systému, flexibilitu při přidělování úkolů personálu a správu výjimek a obnovení po chybách. Specifikace systému je rozšiřitelná o jakékoli potřebné detaily a zahrnuje funkční, strukturální a behaviorální aspekty systému.

Na konci úvodní části je odkaz na vypracování a tvorbu technických dokumentů obecně a mezinárodních norem zvláště:

Jednou z konkrétních aplikací OPM je vypracovávání a tvorba technických norem. OPM pomáhá načrtnout implementaci normy a identifikovat její slabiny, které je třeba omezit, čímž se výrazně zlepšuje kvalita následných návrhů. S OPM si modelový text systému zachovává vysokou míru formálnosti a konzistence, i když se rozšiřuje o další detaily.

Původní motivací pro vytvoření normy ISO pro OPM bylo její použití jako základu pro normy založené na modelech – uvažovanou novou generaci norem ISO. Ve studii Dori et al. (2010) jsme skutečně navrhli kombinovaný, na modelech založený strukturovaný grafický a textový meta-standardní přístup pro specifikaci, ověřování a validaci komplexních systémů obecně a podnikových norem ISO konkrétně. Tato metodika, vyvinutá pod záštitou studijní skupiny ISO TC 184/SC 5 OPM, je navržena tak, aby řešila současné nesrovnalosti a neúplnost technických dokumentů (Blekhman et al., 2011). Na podporu autorů technických specifikací při vytváření a úpravách modelových technických dokumentů jsme vyvinuli Model-Based Authoring of Specifications Environment (MBASE).

Aby se překonal problém, který mají lidé s čtením dlouhých textů v jazyce OPL kvůli jejich mechanickému a repetitivnímu charakteru, zahrnuje rámec MBASE jazyk *Tesperanto* (zkratka pro *Technical Esperanto*) – evoluci jazyka OPL, který je stále automaticky generován z OPD, ale je mnohem přístupnější pro čtení lidmi než OPL, i když je text dlouhý (Blekhman a Dori 2013). Tento rámec byl úspěšně aplikován při modelování komunikace v operačním sále (Blekhman et al. 2015).

Tesperanto lze považovat za textovou verzi hry *The Imitation Game*, lépe známé jako *Turingův test* – test navržený v roce 1951 Alanem Turingem, který byl navržen k vyřešení otázky strojové inteligence. Zatímco v původním Turingově testu musí lidský soudce rozhodnout, zda komunikuje s člověkem nebo počítačem, v textové verzi Turingova testu musí soudce rozhodnout, zda daný text napsal počítač nebo člověk. Je zcela zřejmé, že text v jazyce OPL, který se skládá ze syntakticky správných anglických vět, bude rychle identifikován jako napsaný počítačem, ale pro člověka bude obtížnější odhalit to, když mu bude předložen text v jazyce Tesperanto.

24.2 ISO 19450 Termíny, definice a symboly

Článek 3 normy 19450 PAS obsahuje více než 80 definic pojmů, které se v normě používají. Jsou seřazeny abecedně, přičemž slova v definicích *uvedená kurzívou* jsou sama o sobě termíny definované v tomto článku. [Obrázek 24.1](#) je ukázkou článku ISO 19450 Termíny a definice, který obsahuje některé termíny začínající písmenem p. Například **procesní vazba** je definována v horní části [obrázku 24.1](#) jako „grafické znázornění *procesního vztahu* v OPM“. Termín *procesní vztah* je *kurzívou*, protože je to také samostatný termín, který se náhodou objevuje jako další v abecedním pořadí:

3.57

procedurální vztah

spojení nebo asociace mezi *objektem* nebo *stavem objektu* a *procesem*

Podle směrnic ISO musí být definice formulovány tak, aby bylo možné nahradit kurzívou psaný termín jeho definicí a stále získat srozumitelnou a smysluplnou definici. Například když provedeme nahrazení termínu **procedurální vztah** v definici **procedurálního odkazu**, dostaneme:

3.56

procedurální spojení

grafické znázornění spojení nebo asociace mezi *objektem* nebo *stavem objektu* a *procesem* v OPM

To vysvětluje, proč žádná z definic termínů nezačíná velkým písmenem ani nekončí tečkou. Tento proces nahrazování můžeme pokračovat dvakrát, nejprve pro **proces**:

3.56

procedurální odkaz

grafické znázornění spojení nebo asociace mezi *objektem* nebo *stavem objektu* a *transformací* jednoho nebo více *objektů* v systému v OPM

Při pohledu na definici transformace zjistíme:

3.77

transformace

vytvoření (generování, konstrukce) nebo spotřeba (eliminace, destrukce) *objektu* nebo změna v *stavu objektu*

Nyní tedy dostáváme jako definici procedurálního odkazu:

3.56**procedurální vazba**

grafické znázornění spojení nebo asociace mezi *objektem* nebo *stavem* objektu a vytvořením (generováním, konstrukcí) nebo spotřebou (eliminací, zničením) *objektu* nebo změnou *stavu objektu* jednoho nebo více *objektů* v systému v *OPM*

3.56**procedural link**

graphical notation of *procedural relation* in *OPM*

3.57**procedural relation**

connection or association between an *object* or *object state* and a *process*

NOTE 1 Procedural relations specify how the system operates to attain its function, designating time-dependent or conditional initiating of processes that transform objects.

NOTE 2 An invocation or exception link signifies a transient object in the flow of execution control between two processes.

3.58**process**

transformation of one or more *objects* in the *system*

3.59**process class**

pattern for *processes* that perform the same *object transformation* pattern

Obr. 24.1 Ukázka klauzule ISO 19450 Termíny a definice

Jak vidíme, stále to funguje, i když definice se nevyhnutelně prodlužuje. To může pokračovat, dokud nejsou provedeny všechny nahrazení, a platnost se ověří tím, že se zkontroluje, zda nebyl vytvořen žádný cyklus, tj. definovaný termín se nesmí objevit nikde v definici.

Seznam definic termínů je následován klauzulí **4 – Symboly** a klauzulí **5 – Shoda**. Poté následuje klauzule 6 – **Principy a pojmy metodiky objekt-proces**, které budou popsány dále.

24.3 Principy a pojmy metodiky objekt-proces

Článek 6 je přehledem OPM. Začíná principy modelování OPM, nejprve „Modelování jako činnost sloužící určitému účelu“, který pojednává o tom, jak určit rozsah modelu:

Funkce systému a účel modelování určují rozsah a míru detailnosti modelu OPM. ... Očekávané funkce nebo přínosy zainteresovaných stran obecně a zejména příjemců určují a předepisují účel modelování. To zase určuje rozsah modelu systému.

Použití slova „musí“ je v normách povinné a běžné, jak ukazuje první řádek výše uvedené citace; znamená to povinnost, otázku shody. Dále se diskutuje o sjednocení funkce, struktury a chování:

... Kombinace struktury a chování systému umožňuje systému vykonávat funkci, která má přinést (funkční) hodnotu systému alespoň jedné zainteresované straně, která je příjemcem systému. Model OPM integruje funkční (utilitární), strukturální (statické) a behaviorální (dynamické) aspekty systému do jediného sjednoceného modelu. Tato unifikace struktury a chování, která se zaměřuje na celkovou funkci systému, poskytuje koherentní jednotný referenční rámec pro pochopení systému, který nás zajímá, a zlepšuje jeho intuitivní pochopení při dodržení formální syntaxe.

Dále se klauzule zabývá rozdílem mezi funkcí a chováním, přičemž první z nich je subjektivní, utilitární aspekt, zatímco druhý je objektivní dynamický aspekt systému. Pokud jde o stanovení hranic systému, 19450 uvádí:

Prostředí systému je soubor věcí, které se nacházejí mimo systém, ale mohou s ním interagovat a případně měnit systém a jeho prostředí. Modelář musí rozlišovat mezi těmito věcmi prostředí, které nejsou součástí systému, a systémovými věcmi, které jsou součástí systému. Modelář není schopen navrhovat, projektovat ani manipulovat se strukturou a chováním věci prostředí, i když tyto věci prostředí mohou systém ovlivňovat nebo být jím ovlivňovány.

Posledním tématem v první pododstavci článku 6 je kompromis mezi srozumitelností a úplností:

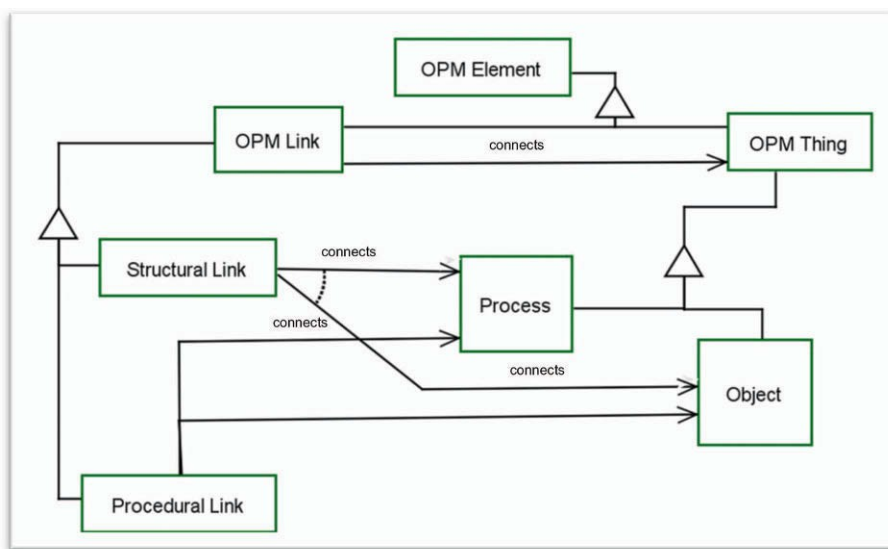
Přehlcující detaily a složitost jsou nedílnou součástí reálných systémů. Aby byly takové systémy srozumitelné, je třeba najít rovnováhu mezi dvěma protichůdnými kritérii: srozumitelností a úplností. Srozumitelnost je míra jednoznačného porozumění, které zprostředkovávají modely struktury a chování systému. Úplnost je míra specifikace všech detailů systému. Tyto dvě vlastnosti modelu jsou vzájemně v rozporu. Na jedné straně úplnost vyžaduje úplné stanovení detailů systému. Na druhé straně potřeba srozumitelnosti ukládá horní limit rozsahu detailů v rámci jednotlivého modelového diagramu, po jehož překročení se srozumitelnost zhoršuje kvůli nepřehlednosti a přetížení.

Další pododstavec v odstavci 6 – Základní pojmy OPM – nejprve představuje bimodální reprezentaci OPM – její grafickou textovou ekvivalenci:

Model OPM musí být bimodální s vyjádřením v sémanticky ekvivalentních grafických a textových reprezentacích. Každý grafický diagram modelu OPM, tj. diagram objekt-proces (OPD), musí mít ekvivalentní textový odstavec OPM složený z jedné nebo více vět v jazyce OPM používajících jazyk objekt-proces (OPL).

Prvky OPM jsou pak definovány jako věci a odkazy. Toto je první krok při definování metamodelu OPM, popsáno v normě ISO 19450, jak je znázorněno na [obr. 24.2](#).

V pokračování je vysvětlen zásadní rozdíl mezi koncepčními modely a runtime modely, přičemž je zdůrazněno, že při konstrukci modelů OPM musí modeláři rozumět rozdílu mezi koncepčním modelem, který vytvářejí, a operačním výskytem tohoto modelu, který mohou použít k posouzení chování systému. Modelář může simulovat chování systému vytvořením operačních instancí objektů a procesů a poté sledovat tok řízení provádění zakotvený ve spojeních a sémantických pravidlech OPM.



Obr. 24.2 Přehled metamodelu OPM (obrázek 1 v ISO 19450)¹

24.4 Čtyři přílohy normy ISO 19450

Hlavní dokument ISO 19450 má 100 stran. Poskytuje přehledné vysvětlení OPM, které je v souladu se specifikacemi v této knize, i když je méně podrobné a neobsahuje některé detaily, které lze považovat za „informativní“ (viz níže). Zbývajících 76 stran tohoto dokumentu obsahuje čtyři přílohy, které společně doplňují definici OPM z různých úhlů pohledu.

Příloha A představuje formální syntaxi pro OPL ve formě EBNF. Příloha B představuje konvence a vzory běžně používané v aplikacích OPM. Příloha C představuje aspekty OPM jako modely OPM. Příloha D nakonec shrnuje dynamické a simulační schopnosti OPM.

¹ Stínování, jako je na tomto obrázku, označuje OPD a výňatky převzaté z normy ISO 19450.

Každá příloha má atribut, jehož hodnoty jsou „normativní“ a „informativní“. Termín „*normativní*“ v normách ISO znamená, že se jedná o závaznou provozní část normy, kterou musí dodržovat každý, kdo prohlašuje, že je v souladu s normou.

Naopak „informativní“ znamená, že se jedná o nezávaznou část normy, kterou lze dodržovat, ale není povinné ji dodržovat, aby bylo možné prohlásit shodu s normou. V tomto smyslu lze veškerý materiál v této knize, který není zahrnut v normativních částech normy ISO 19450, shrnout do další informativní přílohy.

Na základě Bibliowicze a Doriho (2012) bylo plánováno zahrnout do normy ISO 19450 pátou (informativní) přílohu, ve které jsou OPD definovány pomocí grafické gramatiky. Nakonec byla odstraněna z důvodu technických problémů s více grafickými prvky, které bylo příliš obtížné zpracovat v novém publikačním systému ISO.

24.4.1 Příloha A: Normativní: Formální syntaxe OPL v EBNF

Formální gramatika je soubor produkčních pravidel ve tvaru $V \Rightarrow w$, která popisují, jak tvořit platné řetězce ze sady terminálů – symbolů, které tvoří abecedu jazyka. Abeceda OPL je soubor všech vyhrazených frází a interpunkčních znamének. V *kontextově nezávislé gramatice* lze každé produkční pravidlo použít bez ohledu na kontext neterminálu. Jak bylo uvedeno v oddíle 11.5, OPL je sice podmnožinou angličtiny, ale je formální. Gramatika OPL je bezkontextová. Syntaxe, znázorněná na obr. 24.3, používá notaci *Extended Backus–Naur Form (EBNF)*, notaci pro vyjádření syntaxe bezkontextových gramatických jazyků. Verze EBNF podle ISO použitá v normě ISO 19450 je specifikována v normě ISO 14977:1996.² Specifikace EBNF OPL zahrnuje asi 400 produkčních pravidel na 12 stranách. Zde je popis OPL v předmluvě k příloze.

² ISO 14977 je volně dostupná norma, kterou lze bezplatně stáhnout z adresy http://isotc.iso.org/livelink/livelink/fetch/2000/2489/Ittf_Home/PubliclyAvailableStandards.htm.

A.4.2 OPL Identifiers

(* Region: Identifiers – This region defines all identifiers used throughout the grammar *)

```

object identifier = singular object name, [" in ", measurement unit], [range clause]
                  | singular object name, " object", [" in ", measurement unit], [range clause]
                  | plural object name, [" in ", measurement unit], [range clause]
                  | plural object name, " objects", [" in ", measurement unit], [range clause] ;
process identifier = singular process name
                  | Singular process name, " process"
                  | plural process name
                  | plural process name, " processes" ;
thing identifier = object identifier
                  | process identifier ;
state identifier = non capitalized word ;
tag expression = non capitalized phrase ;

```

(* see 7.1 and 7.2 *)

(* EndRegion: Identifiers *)

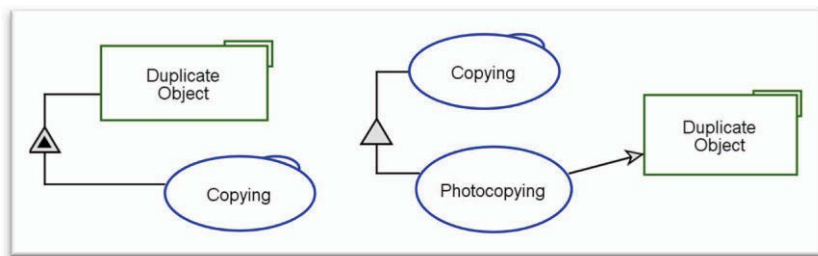
Obr. 24.3 Ukázka zápisu EBNF vyjadřujícího kontextově nezávislou gramatiku OPL

OPL je jazyk s dvojím účelem. Zprvée slouží odborníkům v dané oblasti a systémovým architektům zabývajícím se analýzou a návrhem systémů, jako jsou systémy elektronického obchodování nebo webové systémy pro plánování podnikových zdrojů. Zadruhé poskytuje pevný základ pro automatické generování navržených aplikací.

OPL je textovou obdobou grafické specifikace systému OPM, která odpovídá diagramatickému popisu v sadě OPD. OPL má být automaticky generovaným textovým popisem systému v podskupině přirozené angličtiny. Věty OPL nemají žádné zvláštnosti ani nadměrné kryptické detaily, které charakterizují programovací jazyky, a jsou srozumitelné i pro osoby bez technických nebo programátorských zkušeností.

24.4.2 Příloha B – Informativní: Pokyny pro OPM

Tato příloha popisuje několik principů OPM, které se objevují v této knize, jakož i *konvenci více kopií věci*, která je navržena tak, aby snížila nepořádek, když je třeba vytvořit spojení mezi dvěma věcmi v OPD, které jsou „geograficky“ vzdálené, tím, že umožňuje duplikaci stejné věci. Aby se usnadnilo rozpoznání opakování, může modelář nahradit symbol věci odpovídajícím duplikátem symbolu věci – malým objektem nebo procesem, který se mírně zobrazuje za opakovanou věcí, jak je znázorněno na [obr. 24.4](#).



Obr. 24.4 Duplikované symboly objektů a procesů

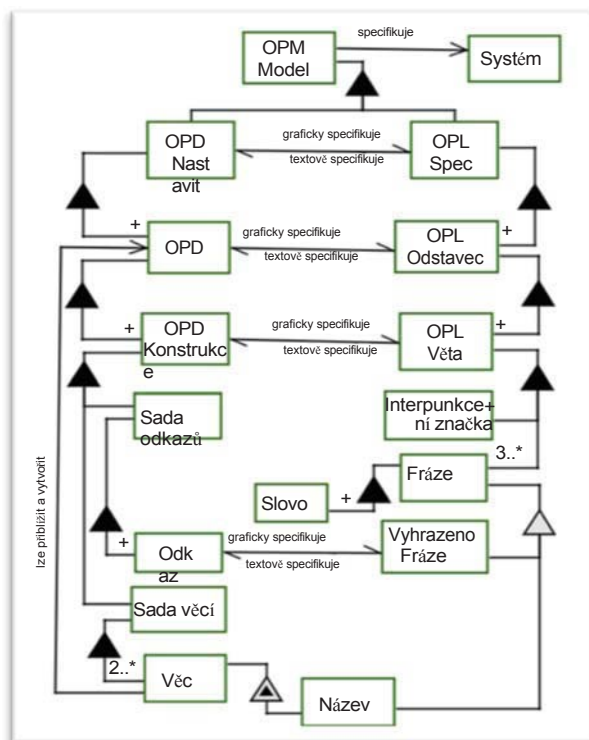
24.4.3 Příloha C – Informativní: Modelování OPM pomocí OPM

Příloha C je poměrně komplexní, i když neúplný model důležitých konceptů OPM vyjádřených v OPM. Jedná se o *reflexivní metamodel* – model OPM, který používá OPM k vlastní specifikaci (Reinhartz-Berger a Dori 2005). Klíčovým testem „dobrého“ konceptuálního modelovacího jazyka je jeho schopnost reflexivního metamodelování. Jak ukazuje příloha C, OPM to zvládá dobře. SD na [obr. 24.5](#) je podrobně popsán v příloze C s přibližně 20 OPD.

Příloha C také poskytuje metamodel **řízení výkonu procesu** – proces provádění procesu, který specifikuje všechny podrobnosti související s implementací paradigmatu událost-podmínka-akce pomocí asi 10 OPD na čtyřech úrovních detailu. [Obrázek 24.6](#) je SD1 tohoto systémového modelu. Kompletní a proveditelná specifikace tohoto systému, integrovaná do reflexivního modelu OPM, může sloužit jako spolehlivý a flexibilní zdroj pro implementaci pokročilého modelovacího nástroje OPM.

24.4.4 Příloha D – Informativní: Dynamika a simulace OPM

Příloha D popisuje animované provedení modelu OPM a způsoby, jak specifikovat a označit atribut **Trvání procesu**. Dosud představené události byly události objektu nebo stavu: nastaly, když konkrétní objekt vznikl nebo vstoupil do konkrétního stavu. Tato příloha mimo jiné specifikuje časovanou událost, která závisí na příchodu konkrétního času v systému, jak je znázorněno na [obr. 22.7](#).



Model OPM specifikuje systém.

Model OPM se skládá ze sady OPD a specifikace OPL.

Specifikace OPL se skládá z alespoň jednoho odstavce OPL.

Sada OPD se skládá z alespoň jednoho OPD.

Sada OPD graficky specifikuje specifikaci OPL.

Specifikace OPL textově specifikuje sadu OPD.

OPD se skládá z alespoň jedné konstrukce OPD.

OPL odstavec se skládá z alespoň jedné OPL věty. OPD

graficky specifikuje OPL odstavec.

OPL odstavec textově specifikuje OPD.

OPD Construct graficky specifikuje OPL Sentence. OPL

Sentence textově specifikuje OPD Construct.

OPD Construct se skládá z Thing Set a Link Set. Thing

Set se skládá ze dvou až mnoha věcí.

Sada odkazů se skládá z alespoň jednoho

odkazu. Věc má název.

OPL věta se skládá ze tří až mnoha frází a alespoň jednoho interpunkčního znaménka. Fráze se skládá z alespoň jednoho slova.

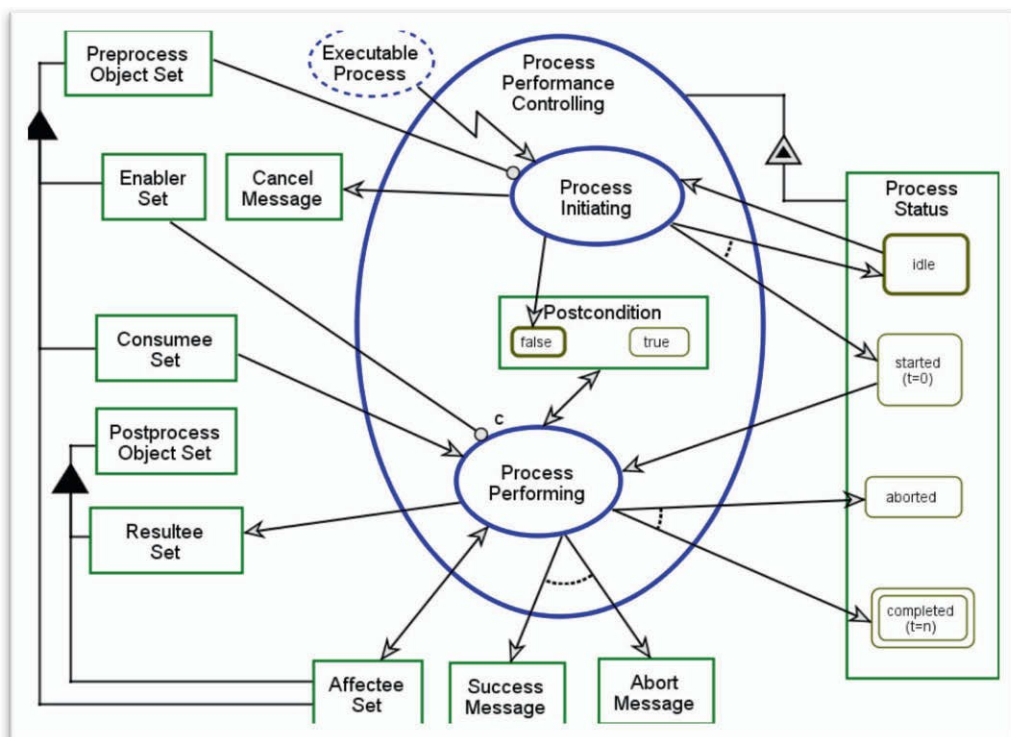
OPL Reserved Phrase a Name of Thing jsou fráze. Link

graficky specifikuje Reserved Phrase.

Rezervovaná fráze textově specifikuje odkaz.

Věc lze přiblížit, aby se vytvořil OPD

Obr. 24.5 OPD (SD) nejvyšší úrovně reflexivního metamodelu OPM (model OPM modelu OPM)



Process Performance Controlling zooms into **Process Initiating** and **Process Performing** in that sequence, as well as **Postcondition**.

Preprocess Object Set consists of **Consume Set**, **Affectee Set**, and **Enabler Set**.

Postprocess Object Set consists of **Resultee Set** and **Affectee Set**.

Executable Process is environmental.

Executable Process invokes **Process Initiating**.

Process Performance Controlling exhibits **Process Status**.

Process Status can be **idle**, **started (t=0)**, **aborted**, or **completed (t=n)**.

Process Status is initially **idle** and finally **completed (t=n)** or **aborted**.

Postcondition can be **false** or **true**.

Postcondition is initially **false**.

Process Initiating requires **Preprocess Object Set**.

Process Initiating changes **Process Status** from **idle** to one of **idle** or **started (t=0)**.

Process Initiating yields **false Postcondition** and **Cancel Message**.

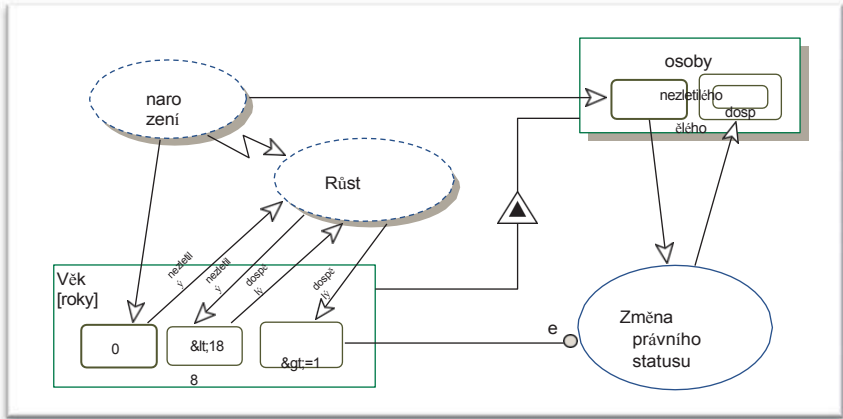
Process Performing occurs if **Enabler Set** exists, otherwise **Process Performing** is skipped.

Process Performing affects **Postcondition** and **Affectee Set**.

Process Performing changes **Process Status** from **started (t=0)** to one of **aborted** or **completed (t=n)**.

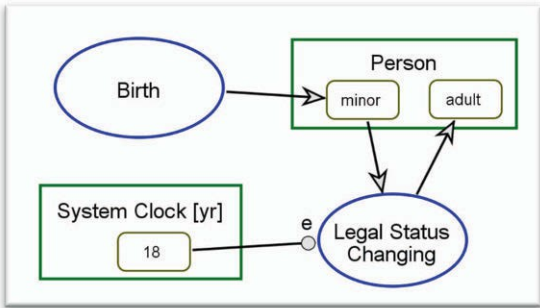
Process Performing yields **Resultee Set** and either **Success Message** or **Abortion Message**.

Obr. 24.6 Řízení výkonu procesu z SD přiblíženého v SD1



Obr. 24.7 Změna **právního statusu** mění **osobu** z **nezletilé** na **dospělou**, když **růst** mění **věk osoby** z **<18** až **≥18**. (Obrázek D.1 v normě ISO 19450)

Alternativně [obrázek 24.8](#) používá objekt **Systémové hodiny**, který může mít jakýkoli systém, buď explicitně jako v tomto příkladu, nebo implicitně, k vyvolání události, když **systémové hodiny**, které se spouštějí při **narození**, dosáhnou **18** let, vytvoří událost, která spustí **změnu právního statusu**.



Obr. 24.8 Událost System Clock spouštějící změnu právního statusu (obrázek D.2 v ISO 19450)

Odkazy

- Agile Modeling <http://agilemodeling.com/style/activityDiagram.htm#Swimlanes>, přístup 3. července 2015.
- Ashby, W.R. Concepts of Operand, Operator, Transform. George Washington University, St. Louis, 2001. http://www.gwu.edu/~asc/biographies/ashby/MATRIX/SG/sg_1.html, přístup 16. března 2015.
- Arnheim, R. *Vizuální myšlení*. University of California Press, Berkeley, CA, 1969. Baddeley, A. Pracovní paměť. *Science* 31, sv. 255, č. 5044, s. 556–559, leden 1992.
- Bak, P., Tang, C. a Wiesenfeld, K., Self-Organizing Criticality. *Physics Review Letters*, 59(4), str. 381–384, 1987.
- de Bakker, J.W. a de Vink, E.P. *Sémantika řídicího toku*. MIT Press, Cambridge, MA, 1996.
- Bar-Yam, Y. *Dynamika komplexních systémů*. Perseus Books, Reading, MA, 1997.
- Berkeley, G. *Pojednání o principech lidského poznání*. Trinity College, Dublin, 1710. Bibliowicz, A. a Dori, D. Formální validace diagramů objektů a procesů na základě grafové gramatiky. *Software and Systems Modeling*, 11(2), str. 287–302, 2012.
- Blair, C. D., Boardman, J. T. & Sauser, B. J. Komunikace strategického záměru pomocí systémových diagramů: Aplikace na výzvu umožněnou sítí. *Systémové inženýrství*, 10(4), str. 309–322, 2007.
- Blekhman, A. a Dori, D., Tesperanto – metodika a jazyk specifikace systému založený na modelech. Proc. 23. výroční mezinárodní sympozium INCOSE, Filadelfie, Pensylvánie, USA, 24.–27. června 2013.
- Blekhman, A., Dori, D. a Martin, R. Modelově založené vytváření standardů. Proc. 21. mezinárodní sympozium INCOSE, Denver, CO, USA, str. 650–659, 19.–23. června 2011.
- Blekhman, A., Wachs, J. P. a Dori, D. Modelová specifikace systému pomocí Tesperanta: čitelný text z formální grafiky. Vyjde v IEEE SMC, 2015. *Systémové inženýrství v životním cyklu produktu*
- Bock, C. Systémové inženýrství v životním cyklu produktu. *International Journal of Product Development*, 2(1–2), str. 123–140, 2005.
- Bock, C. Podpora SysML a UML 2 pro modelování aktivit, *Systems Engineering* 9(2) str. 160–186, 2006.
- Bolshchikov, S., Somekh, J., Mazor, S., Wengrowicz, N. a Dori, D. Vizualizace dynamiky Koncepční modely chování: Vivid OPM Scene Player. *Systémové inženýrství*. Vyjde v roce 2015.
- Box, G. E. P. a Draper, N. R. *Empirical Model Building and Response Surfaces*, John Wiley & Sons, New York, NY, 1987.

- Bunge, M. *Treatise on Basic Philosophy, Vol. 3, Ontology I, The Furniture of the World*. Reidel, Boston, MA, 1977.
- Bunge, M. *Treatise on Basic Philosophy, sv. 4, Ontology II, A World of Systems*. Reidel, Boston, MA, 1979.
- Chein, M. & Mugnier, M.L. Koncepční grafy: základní pojmy. *Rev. Intell. Artif.* 6(4), str. 365–406, 1992.
- Chen, P.P. Model vztahů mezi entitami – směrem k jednotnému pohledu na data. *ACM Trans. on Data Base Systems* 1(1), str. 9–36, 1976.
- Clark, J. M. & Paivio, A. Teorie duálního kódování a vzdělávání. *Educational Psychology Review*, 3(3), 149–210, 1991.
- Coad, R. a Yourdon, E. *Objektově orientovaná analýza*. Prentice-Hall, Englewood Cliffs, NJ, 1991. Cook, S. V předmluvě k Warmer a Kleppe (1999).
- Descartes, R. *Principy filozofie* (1647). Překlad Valentine Rodger Miller a Reese P. Miller. D. Reidel Publishing Company, str. 282, 1984.
- Dittrich, K.R., Gatziau, S. a Geppert, A. Manifest aktivního systému správy databází: Pravidla funkcí ADBMS. Přednáškové poznámky v počítačové vědě 985, Springer, str. 3–20, 1995.
- Dirks, T. Nejlepší filmy, 2001. <http://www.filmsite.org/gone.html>. Přístup 16. března 2015. DoDAF – DoD Architecture Framework verze 1.5, 2007.
- DoDAF – DoD Architektura Framework Verze 2.02, Změna 1, 2015. <http://dodcio.defense.gov/TodayinCIO/DoDArchitectureFramework.aspx>. Přístup 16. března 2015.
- Dori, D., *Object-Process Methodology – A Holistic Systems Paradigm*, Springer Verlag, Berlín, Heidelberg, New York, 2002 (ISBN 3-540-65471-2; předmluva Edward Crawley. Vázaná kniha, 453 stran, s CD-ROM). Verze eBook: <http://link.springer.com/book/10.1007/978-3-642-56209-9/page/1>
- Dori, D., ViSWeb – Vizuální sémantický web: Sjednocení reprezentace lidského a strojového poznání pomocí metodiky objektového zpracování. *Mezinárodní časopis o velmi velkých databázích (VLDB)*, 13, 2, str. 120–147, 2004.
- Dori, D. Slova z obrázků pro dvoukanálové zpracování: bimodální graficko-textová reprezentace komplexních systémů. *Communications of the ACM*, 51(5), str. 47–52, 2008.
- Dori, D., Martin, R. a Blekhan, A. Modelová metastandardizace: modelování podnikových standardů pomocí OPM. Mezinárodní konference IEEE o systémech, San Diego, Kalifornie, USA, 5.–8. dubna 2010.
- Dori, D., Reinhartz-Berger, I. a Sturm, A. Vývoj komplexních systémů s metodikou objekt-proces pomocí OPCAT. *LNCS 2813*, str. 570–572, 2003.

- Downton, C. In Smolan, R. a Erwit, J. *One Digital Day*. Time Book/Random House ve spolupráci s Against All Odds Production, 1998.
- Firlej, M. a Helens, D. *Získávání znalostí: Praktická příručka*. Prentice-Hall, New York, 1991. Fowler, M. *UML Distilled, druhé vydání*. Addison-Wesley, Reading, MA, 1999.
- Friedenthal, S., Moore A. a Steiner, R. *Praktický průvodce SysML (druhé vydání), The Systems Modeling Language*, Morgan Kaufmann, 2014.
- Galileo, G. The Assayer, 1623, přetištěno v (Drake, 1957, str. 274)
- Grobshtein, Y. a Dori, D. Generování pohledů SysML z modelu OPM: návrh a hodnocení. *Systems Engineering*, 14 (3), str. 327–340, 2011.
- Harel, D. Statecharts: Vizuální formalismus pro komplexní systémy. *Věda o programování počítačů* 8, str. 231–274, 1987.
- Harel, D. O vizuálních formalizmech. *Communications of the ACM* 31(5), str. 514–530, 1988.
- Helighen, F. Principia Cybernetica Web (1997). <http://pespmc1.vub.ac.be/occamraz.html> Přístup 27. května 2015.
- Holt, J. *UML pro systémové inženýrství: Sledování kol*. IEE Professional Applications of Computing, 2004.
- IATA <http://www.iata.org/about/Pages/index.aspx> Přístup 31. května 2015.
- Kant, I. *Prolegomena k jakékoli budoucí metafyzice, která bude schopna se prezentovat jako věda* (v němčině). Riga, 1783.
- Kilov, H. a Simmonds, I. D. Obchodní vzorce: Opakovaně použitelné abstraktní konstrukce pro obchodní specifikace. V Humphreys, P., Bannon, K., McCosh, A., Migliarese, P. a Pomerol, J.S. (Eds.), *Implementace systémů pro podporu manažerských rozhodnutí*. Chapman and Hall, Londýn, 1996.
- Klyne G., Carroll, J.J. a McBride, B. *RDF 1.1 Concepts and Abstract Syntax*, 2004 <http://www.w3.org/TR/rdf-concepts> Přístup 16. března 2015.
- Koffka, K. *Principy gestaltové psychologie*. New York: Harcourt-Brace, 1935.
- Kostovic, I. a Rakic, P. Vývojová historie přechodné subplate zóny ve vizuální a somatosenzorické kůře mozku makaka a lidského mozku. *J. Comp. Neurol.*, 297: 441–470, 1990. doi: 10.1002/cne.902970309
- Latimer, C. a Stevens, C. Několik poznámek k celku, částem a jejich vnímání. *Psychology* 8(13), Vnímání celku a částí (1), 1997.
- Lehman, F. (ed.) *Sémantické sítě v umělé inteligenci*. Pergamon, Oxford, Velká Británie, 1999.

- MacIntyre, J. Co je špatného na strategii objektově orientovaného designu založeném na podstatných jménech, přídavných jménech a slovesech (2010). <https://whileicompile.wordpress.com/2010/08/01/the-noun-adjective-verb-object-oriented-design-strategy-sucks/> Přístup 9. června 2015.
- Manola F. a Miller E. *RDF 1.1 Primer*, 2014. <http://www.w3.org/TR/2014/NOTE-rdfl1-primer-20140624/>. Přístup 16. března 2015.
- Mayer, R.E. Slib multimediálního učení: použití stejných metod výukového designu v různých médiích. *Učení a výuka* 13, str. 125–139, 2003.
- Mayer, R.E. a Moreno, R. Devět způsobů, jak snížit kognitivní zátěž při multimediálním učení. *Educational Psychologist* 38(1), str. 43–52, 2003.
- Mazanec, M. a Macek O. O univerzálních jazycích pro textové modelování. Proc. 12. výroční mezinárodní workshop o databázích, textech, specifikacích a objektech, Zernov, Rovensko pod Troskami, Česká republika, str. 1–12, 2012.
- Meinhardt, H. *Algoritmická krása mořských mušlí*. Springer-Verlag, Berlín, 1995.
- MIT ESD. Vize divize inženýrských systémů Massachusetts Institute of Technology. <http://esd.mit.edu/about/vmv.html> Přístup 28. května 2015.
- Mordecai, Y. a Dori, D. *I⁵ : Modelový rámec pro architekturu interoperability, propojitelnosti, rozhraní, integrace a interakce systémů systémů*,²³ mezinárodní symposium Mezinárodní rady pro systémové inženýrství INCOSE IS-2013, Filadelfie, PA, USA, červen 2013.
- Ockham, W., *Quaestiones et decisiones in quattuor libros Sententiarum*, Petri Lombardi ed., Lugd., 1495. OMG
- OCL Object Constraint Language, verze 2.4, 2014. <http://www.omg.org/spec/OCL/2.4/PDF/> Přístup 20. června 2015.
- OMG SysML System Modeling Language, verze 1.3, 2012. <http://www.omg.org/spec/SysML/1.3/PDF/> Přístup 16. března 2015.
- OMG UML Unified Modeling Language, Infrastructure, verze 2.4.1, 2011I. <http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF/> Přístup 16. března 2015.
- OMG UML Unified Modeling Language, Superstructure, verze 2.4.1, 2011S. <http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF> Přístup 16. března 2015.
- OMG UML for Systems Engineering RFP, 2003. http://syseng.omg.org/UML_for_SE_RFP.htm Přístup 16. března 2015.
- Ouellette, J. A Fundamental Theory to Model the Mind. *Quanta Magazine*, 2014. <https://www.quantamagazine.org/20140403-a-fundamental-theory-to-model-the-mind/>. Přístup 18. března 2015.
- Peleg, M. a Dori, D. Problém mnohostrannosti modelů: Experimentování s metodami specifikace v reálném čase. *IEEE Transactions on Software Engineering* 26(8), str. 742–759, 2000.

- Reinhartz-Berger, I. a Dori, D. Reflexivní metamodel metodologie objekt-proces: Stavební bloky modelování systému. V *Business Systems Analysis with Ontologies*, P. Green a M. Rosemann (Eds.), Idea Group, Hershey, PA, USA, str. 130–173, 2005.
- Rescher, N. a Oppenheim, P. Logická analýza gestaltových konceptů. *British Journal for the Philosophy of Science* 6, str. 89–106, 1955.
- Rissanen, J. Modelování pomocí nejkratšího popisu dat. *Automatica* 14 (5): 465–658, 1978.
- Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. a Lorensen, W. *Objektově orientované modelování a návrh*. Prentice-Hall, Englewood Cliffs, NJ, 1991.
- Schapiro, M. *Slova, písmo a obrázky – sémiotika vizuálního jazyka*. The New Republic, 1996. Shannon, C.E. a Weaver, J. *Matematická teorie komunikace*, University of Illinois Press, 1949.
- Smithsonian Institute <http://humanorigins.si.edu/evidence/behavior/tools> Přístup 31. května 2015. Somekh, J., Haimovich, G., Guterman, A., Dori, D. a Choder, M. Koncepční modelování rozpadu mRNA rozpadu provokuje nové hypotézy, 2014. PLoS ONE 9(9): e107085. doi:10.1371/journal.pone.0107085.
- Stanfordská encyklopedie filozofie, 2013. <http://plato.stanford.edu/entries/logic-probability/> Přístup 8. července 2015.
- TOGAF verze 9.1, The Open Group Architecture Framework, <https://www.opengroup.org/togaf/> Prezentace *Přehled správy* ; 2011. Přístup 8. června 2015.
- USPTO 7,099,809 *Modeling System*, Úřad pro patenty a ochranné známky USA, číslo 7,099,809, podáno: 15. března 2001. Uděleno: 29. srpna 2006. <http://www.google.com/patents/US20070050180>
- W3C Consortium, Resource Description Framework (RDF), 2014. <http://www.w3.org/RDF/> Přístup 16. března 2015.
- Wand, Y. a Weber, R. Ontologické hodnocení metod systémové analýzy a návrhu. V Falkenberg, E.D. a Lindgreen, P. (Eds.) *Koncepty informačních systémů: Hlubková analýza*. Elsevier Science Publishers B.V. (North Holland), IFIP, str. 145–172, 1989.
- Wand, Y. a Weber, R. O ontologické expresivitě gramatik analýzy a návrhu informačních systémů. *Journal of Information Systems* 3, str. 217–237, 1993.
- Warmer, J. a Kleppe, A. *Jazyk objektových omezení: Přesné modelování pomocí UML*. Addison-Wesley, Reading, MA, 1999.
- Weber, R. H. a Weber, R. *Internet věcí, právní perspektivy*. Springer, 2010. Wertheimer, M. a Reizler, K. Teorie gestaltu. *Sociální výzkum* 11(1), s. 78–99. 1944.
- Winograd, T. a Flores, F. *Porozumění počítačům a kognici*. Addison-Wesley, Reading, MA, 1987.

Principy OPM v kostce

Principy OPM jsou uvedeny níže v pořadí, v jakém se objevují v knize.

1. **Princip OPM „funkce jako semínko“** – Modelování systému začíná definováním, pojmenováním a popisem funkce systému, která je zároveň jeho nejvyšší úrovní procesu.
2. **Princip reprezentace faktů modelu OPM** – Fakt modelu OPM se musí objevit alespoň v jednom OPD, aby mohl být reprezentován v modelu.
3. **Princip časové osy OPM** – časová osa v přiblíženém procesu je ve výchozím nastavení vedena od horní části přiblížené elipsy procesu k její spodní části.
4. **Princip minimálního konceptuálního modelovacího jazyka OPM** – Symbolický systém – jazyk – který dokáže konceptuálně modelovat daný systém pomocí ontologie s menším počtem typů diagramů a menším počtem symbolů a vztahů mezi nimi, je lepší než větší ontologie s větším počtem typů diagramů a větším počtem symbolů a vztahů mezi nimi.
5. **Princip OPM „Důležitost věci“** – Důležitost věci T v modelu OPM přímo souvisí s nejvyšší hodnotou OPD v hierarchii OPD, kde se T objevuje.
6. **Princip transformace objektu procesem OPM** – V kompletním modelu OPM musí být každý proces propojen alespoň s jedním objektem, který proces transformuje, nebo s jedním stavem objektu, který proces transformuje.
7. **Princip jedinečnosti procedurálního propojení OPM** – Na jakékoli úrovni detailu může být objekt a proces propojeny maximálně jedním procedurálním propojením, které jednoznačně určuje roli objektu ve vztahu k procesu.
8. **Princip jednotného názvu OPM** – Název OPM musí být v jednotném čísle. Množné číslo musí být převedeno do jednotného čísla přidáním slova „sada“ pro neživé věci nebo „skupina“ pro lidi.
9. **Princip graficko-textové ekvivalence OPM** – Jakýkoli modelový fakt vyjádřený graficky v OPD je také vyjádřen textově v odpovídajícím odstavci OPL.
10. **Princip jedinečnosti názvu věci OPM** – Různé věci v modelu OPM, které nejsou vlastnostmi, musí mít různé názvy. Vlastnosti lze rozlišit přidáním vyhrazeného slova „of“ a názvu jejich vystavovatele.
11. **Princip detailní hierarchie OPM** – kdykoli se OPD stane obtížně srozumitelným kvůli nadměrnému množství detailů, vytvoří se nový, podřízený OPD.
12. **Princip OPM časové osy** – časová osa v přiblíženém procesu je ve výchozím nastavení vedena od horní části přiblížené elipsy procesu k její spodní části.
13. **Princip OPM „Přednost sémantiky přeskočení“** – Sémantika přeskočení má přednost před sémantikou čekání.

Index

Blokový stereotyp	242
zkrácené omezení účasti	201
akce	327, 155, 139
aktivita	139
diagram aktivity	139, 136
Diagram aktivit	135
aktér	167, 139, 30
přídavné jméno	221
ovlivněný objekt	167
ovlivněný subjekt	179, 169, 163
příslušnost	90, 27
Příslušnost	25
agent	180, 166, 27, 22, 20
agent podmínka odkaz	336
odkaz na událost agenta	330
odkaz na agenta	180, 20
agregované pojmenování	226
Agregace	217
komplexnost agregace	232
agregace hierarchie	230
agregace-účást	242, 223, 221, 216, 185, 183, 22
agregace-účást odkaz	27
symbol vztahu agregace-účást ▲	224
symbol agregace-účást	22
algebra	349, 207, 205, 197
algebraická rovnice	200
algebraické závorky	349, 206
předek	223, 216
animované provedení	41, 15
animovaná simulace	1, 16, 267
animace	15
anti-reciproční strukturální spojení	189
antisymetrický	188
architektura	295
architektura	91
Architektura	89
Pismo Arial	14
arity	183
umělý systém	246
ASCII znaková sada	236
asociace	194, 185, 183, 120, 111, 109
asociativní odkaz	187
spojení s časem	110
atomový stav	274, 271

atribut	254, 247, 242, 241, 40
Automatická reakce na nehodu	3
zpětná strukturální souvislost	185
název zpětné strukturální relace	216
zpětná značka	187
chování	91, 89
Chování	75
příjemce	246, 91, 87, 83, 20, 6, 4
výhoda za cenu	83
obousměrný modelově specifický nulový tag	193
obousměrný štítkový strukturální odkaz	187
bimodální graficko-textové znázornění	130
bimodální reprezentace	124
binární strukturální vztah	239, 194, 185, 183
černá skříňka	329
černý diamant	227
blok	137, 114, 32, 6
schéma definice bloku	32, 29
Diagram definice bloku	136
blokový diagram	227
plán	131
tučné písmo OPL konvence	128
Booleovský objekt	53
Booleovský pár	55
Booleovská proměnná	53
směr zdola nahoru	278
mozek	268, 151, 129, 99, 98, 16
obcházecí mechanismus	333
výzva k akci	139
kapitalizace OPM konvence	121, 111
kardinálnost	213, 202
kartézský součin	284, 274, 271, 244
příčina a následek	102
změna	179, 157
proces zabraňující změnám	117
charakterizace	217
třída	286, 114
třídní diagram	227
úroveň třídy	262
klasifikace	217
instancování klasifikace	292, 216
uzavřená trojúhelníková šipka	186
nepořádek	296
generování kódu	346
kognitivní předpoklad	129
kombinatorické NEBO	366
kombinatorický XOR	366
oddělené čárkou	25
oddělení	242

složitost	295
řízení složitosti	296, 149
složka	85
složená agregace	227
složení	221
složený stav	274, 271
komplexnost – srozumitelnost kompromis	165
programovací jazyk	127
výpočty s OPM	344
koncepční úroveň	262
koncepční model	93, 76
koncepční modelování	76
jazyk konceptního modelování	93
podmínka	327
podmínka agent odkaz	335
podmínka spotřeba odkaz	336, 335, 334
podmínka efektu odkaz	335
podmínka vstup-výstup-specifikovaný efekt odkaz	338, 337
podmínka vstup-specifikovaný efekt odkaz	338, 337
podmínka nástrojový odkaz	336
podmínka propojení	355, 333
podmínka výstupem specifikovaný efekt odkaz	339, 337
sémantika podmínky	333
podmínka stavově specifikovaný agent odkaz	337
stav specifikovaný podmínkou spotřeby odkaz	338, 337
stavový odkaz na nástroj specifikovaný stavem	338
blok vlastností omezení	155, 146
konstrukce	162, 158
spotřebitel	179, 163
spotřebovávající	109, 5
spotřeba	162, 158
spotřeba odkaz na událost	330
spotřeba odkaz	180, 164
míra spotřeby	343
hierarchie zadržování	190
vztah zadržování	143
kontextově nezávislá gramatika	132, 127
kontinuální proces	342
kontrola	53
kontrolní tok	151, 139
řídící spojení	355, 329
řídící mechanismus	268
modifikátor ovládání	355, 341, 329, 327, 53, 27
ovládací modifikátor odkazu fan	367
konvergentní konec	360
konvergující spotřeba odkaz fanoušek	362
konvergující ventilátor	362
konvergující výsledek odkaz fanoušek	362
nápravné opatření	268

kreativita	227
zákazník	296, 151, 131, 124, 87
zákaznická strana	130
kybernetický	268, 150, 75
kyberfyzický mechanismus	268
kyberfyzický systém	268
slova oddělená pomlčkou	216
člen dat	242
databázové schéma	202, 151, 131, 125
poznámka k rozhodnutí	155
rozhodovací uzel	153
výchozí kapitalizace	257
výchozí hodnota podstaty	23
výchozí nulová značka	192
výchozí omezení účasti	213, 198, 197
výchozí stav	274, 258
výchozí název systému	85
potomek	223, 216
chyba návrhu	130
požadovaná hodnota atributu	268
omezení účasti cíle	213, 201, 198
problém detailní exploze	296, 295
vývojář	124
problém s mnohostí diagramů	162
rozměr	272
diskrétní proces	342
rozlišovací atribut	284, 283
distribuční zákon	224, 207, 206, 197
distribuční zákon strukturálních vztahů	213, 205
distributivní sémantika	349
divergentní konec	360
rozbíhající se vějíř	362
DoDAF	90
odborník v dané oblasti	124
předpoklad dvoukanálového provozu	133, 129
dynamický aspekt	16
EBNF	127
paradigma ECA	327
EFFBD	139
efekt	162, 158, 13
Účinek	179, 157
efekt událost odkaz	330
efekt odkaz	180, 164, 16, 11, 7
míra účinku	343
elektrický signál	268
elektronická ekonomika	75
elektronický signál	268
prvek	80
vznik	253, 222

vznikající rys	255, 253
umožňovač	263, 169, 166, 22, 20
enablers	20
umožňující odkaz	180, 168, 162
zapouzdření	242
inženýrský systém	88
anglická gramatika	279
Vylepšené funkční blokové schéma	Viz EFFBD
prostředí	90
environmentální	150, 90
environmentální věc	27, 24
podstata	23
Podstata	27
událost	355, 347, 341, 327
modifikátor řízení události	27
odkaz na událost	355, 329, 40, 27
sémantika události	329
událost-podmínka-akce	355, 328, 327
výjimka odkaz	340
proveditelnost	15
generování spustitelného kódu	131
řízení provádění	346
charakterizace výstavy	254, 244, 239, 216, 40
charakterizace výstavy symbol	243, 40
vystavovatel	239, 40
existence	160
Očekávaná doba trvání	340
explicitní atribut	255
explicitnost	255, 251
vyjadřovací schopnost	240, 129, 123
Rozšířená Backus-Naur forma	Viz EBNF
zobrazení faktů Princip OPM	39
funkce	248, 239
zpětná vazba	130
zpětná vazba	267
konečný stav	258, 26
prvotřídní občan	15
derivace prvního řádu	24
flip-flop	359, 265
tok řízení	141
tok řízení provádění	355, 327
sémantika toku	155
rozvětveníxi, 73, 140, 155, 197, 206, 207, 208, 209, 210, 211, 212, 213, 214, 224, 225, 228, 229, 232, 237, 249, 278	
Komplexnost vidlice	210
Stupeň vidlice	209
uzel vidlice	155
větvení OPL věta	207
formulář	4

formální model	92
formální modelovací jazyk	92
formálnost	151
název předního vztahu	216
předávací strukturální vztah	185
přední značka	187
funkce	253, 246, 242, 91, 88, 83, 75, 19, 4
funkční hodnota	379, 83
Princip OPM „funkce jako semínko“	4
Rozdíl mezi funkcí a chováním	95
základní strukturální vztah	239
základní strukturální vztah	215, 185
základní strukturální vztahy	215
obecný	281, 278
generalizace	217
generalizace-specializace	292, 287, 277, 242, 216, 56
hierarchie generalizace-specializace	287
vztah generalizace-specializace odkaz	278
gerundium	247, 246, 121, 112
gerundium proces pojmenování režim	112
teorie gestalt	222
grafická modalita	149, 123
princip ekvivalence grafiky a textu	132, 125
Guillemets	31
rukojeť	213, 207
objekt rukojeti	225
rukojeť věc	213, 207
tvrdý atribut	255, 253
hardware	167
harpunovitý šíp	187
má vztah	223
hierarchie	218
hierarchický strom	217
historie	263, 262
holismus	222
homogenita	254
lidská inteligence	226
lidské rozhraní	166
identifikovatelnost	98
identifikovatelná entita	262
identita	262, 257, 179, 162, 161, 159, 158, 99, 98
IEEE STD 610.12	90
imperativní režim pojmenování procesů	113
implicitní atribut	255, 250
implicitní volací odkaz	350, 347
INCOSE	136
nekonečná smyčka	334
informatický	150, 75
informatický objekt	163, 120, 101, 100, 99, 98

informatický systém	99
informatická věc	27, 23
informační přetížení	296
kanál pro zpracování informací	151
informační systém	253
vývoj informačních systémů	202
vlastní rys	253
dědičný prvek	280
dědičnost	292, 280, 56
počáteční stav	274, 258, 25
zahájení	347
vstup-výstup	176
vstup-výstup-specifikovaný efektový pár	180, 176
vstup-výstup-specifikovaná hodnota efektového páru	178
vstup	163
vstupní spojení	259, 180, 173, 14
stav vstupu	261, 180, 173, 14, 13
pár vstupních a výstupních spojů	158, 16, 14
vstup-výstup specifikovaný efekt spojení	14
pár vstup-výstup-specifikovaný efekt událost	332
vstupem specifikovaný efektový odkaz na událost pár	332
pár efektů specifikovaných vstupem	180, 177
instance	288, 287, 286
úroveň instance	291, 262
instancování	287, 286
nástroj	180, 167, 22, 21, 20
stav nástroje odkaz	336
odkaz na událost nástroje	330
odkaz na nástroj	180
rozhraní	166, 131
prozatímní objekt	342, 331
Vnitřní blokové schéma	136, 32
Vnitřní blokové schéma	32
Mezinárodní rada pro systémové inženýrství	Viz INCOSE
Mezinárodní systém veličin	340
Internet	75
Internet věci	114, 75
křížovatka	328
intuitivnost	15
inverzní vztah	223
odkaz na vyvolání	342, 333, 331, 118
sada zapojených objektů	328, 169
proces přiblížení	183
přiblížení	42, 38, 1
přiblížení-oddálení	295
je-vztah	223
ISO 19450	xxii, 1, 127, 132, 219, 235, 375, 377, 378, 380, 381
ISO 19450 ISO 19450	375
ISO 19450 PAS	375

ISO 80000-3:2006 – Veličiny a jednotky	340
Směrnice ISO	377
ISO TC184/SC5	375
Norma ISO/IEC 152 88	83
iterace	346
zubatá čára	332
spojovací uzel	155
Kant	97
jazyk	91
list	287
křivka učení	124
levá hemisféra	129
úrovně detailů	383, 309, 298, 296, 150
lexikální analýza	127
životní cyklus	259
životní linie	141
diagram životnosti	274, 263
symbol blesku	332
odkaz	27
kardinalita odkazu	197
distribuce odkazů	349
odkaz fan	360
přesná specifikace odkazu	280
pravděpodobnost odkazu	371
nedostatečná specifikace odkazu	282, 280
živý organismus	159
logické AND	328, 264
logický operátor AND	357
logické NEBO	328
logický operátor OR	361
logický vztah	149
logický XOR	328
logický operátor XOR	360
strojově orientovaný cíl OPL	124
Manhattanovy řádky	217
mnoho-k-mnoha kardinalita	202
operace s maticemi	306
maximální přidělený čas	342
Maximální doba trvání	340
maximální časový proces	355, 341
paměťový mechanismus	265
mentální záznam	99
metamodel	260, 205, 191, 125
metamorfóza	160
meta-standard	376
metoda	246, 242
metodika	296, 295, 76
minimální přidělený čas	342
Minimální doba trvání	340

Princip minimální ontologie	94, 76
minimální univerzální ontologie	94, 78
minimálně časovaný proces	355, 341
princip minimální délky popisu (MDL)	77
modalita	123
režim	255, 252
model	92
model fakt	296, 123, 24
model faktické znázornění princip OPM	296, 209, 46, 42, 39
Prostředí pro tvorbu specifikací na základě modelu (MBASE)	37
modelový standard	376
modelové systémové inženýrství (MBSE)	76
modelování	92
chyba modelování	124
modelovací jazyk	92
modelově specifický nulový tag	193
modul	85
věta OPL s více podmínkami	272
vícenásobné dědičnost	280
násobící operátor	349
multiplicita	203, 197
přirozený jazyk	249, 206, 205, 129, 125
zpracování přirozeného jazyka	Viz NLP
NLP	226
nominální hodnota	267
symbol neúplné vidlice	211
symbol neúplnosti	232
nelidský aktivátor	168
nereciproční strukturální vazba	189
nerezervovaná fráze	132, 128
netechnický zúčastněný subjekt	130
podstatné jméno	249, 247, 110, 109, 103, 102
nový uživatel OPM	124
nulový strukturální odkaz	192
null tag	192
null tag jednosměrný strukturální odkaz	268
objekt	120, 114, 99, 98, 94, 82, 5
Jazyk omezení objektu	354
Jazyk omezení objektů	236
tok objektů	151, 139
rozvětvení objektů	214, 207
Skupina pro správu objektů	Viz OMG
Technika modelování objektů	243
jedinečnost objektu Princip OPM	121, 112
transformace objektu	194, 109
objektově orientovaný	Viz OO
objektově orientované metody	162
Objektově-procesní tvrzení	94, 80
Důsledek objekt-proces	79

objektově-procesní diagram	Viz OPD
Objektově-procesní diagram (OPD)	19, 5
jazyk objekt-proces	Viz OPL
Jazyk objekt-proces	132, 123
Metodika objektového procesu	Viz OPM
Objektově-procesní věta	94, 79
Ockhamova břitva	77
OMG	1
jednostranné omezení účasti	198
kardinalita jedna k mnoha	202
OnStar	3
ontologie	94, 76
OO	247, 246, 245, 242
OO design	280
OPCAT	263, 225, 198, 160, 123, 67, 40, 39, 38, 21, 17, 15, 12, 11, 8, 6
OPD	5
Konstrukce OPD	123
OPD prvek	123
Sada OPD	210, 132, 125, 123, 42, 39
OPD strom	38
šipka s otevřenou hlavou	186
operand	119, 20, 6
operace	254, 247, 246, 242, 241
operační instance	380
provozní úroveň	291, 262
operační sémantika	327
operátor	119
OPL	226, 124, 123, 11 Viz Objektově-procesní jazyk
odstavec OPL	132, 125
Vyhrazená fráze OPL	132, 128
OPL věta	123, 16, 14
Specifikace OPL	125
OPM	1
Metamodel OPM	125
Studijní skupina OPM	376
Tabulka OPM	306
Kritérium objednávky	214
Pravidlo objednávky	212
objednatelnost	323, 306, 237, 232, 230, 229, 214, 212, 211, 209
Objednatelnost	214, 211
objednáno	211
ortonormální	228
ortonormální polylinie	217
výstup	163
výstupní odkaz	259, 180, 173, 14
stav výstupu	261, 180, 173, 14, 13
výstupem specifikovaná dvojice efektové události a odkazu	332
výstupem specifikovaný efektový pár odkazů	177
zoomování ven	63

přepsání.....	280
odkaz na výjimku z přesčasů	355, 341, 264
proces zpracování přesčasů.....	355, 341
soubor odkazů na paralelní implicitní volání	350
parametr	199
parametr omezení sady.....	235
parametrická omezení.....	204
parametrizované omezení účasti	213, 199
parametrizované omezení účasti	235
Parametrizované omezení účasti (PPC) mini-jazyk.....	237
parametrický diagram.....	147, 146
parametrický diagram.....	135
Parametrický diagram	136
analýza.....	127
částečná agregace spotřeba	232
částečné pořadí	82
omezení účasti	260, 235, 213, 203, 202, 201, 200, 199, 198, 197
symbol omezení účasti	200
vztah „část z“	221
označení cesty	265, 264
doba představení	341
vytrvalost	278, 250, 241, 185
vytrvalost	116, 113, 102, 98, 27
fráze	186, 132, 128, 127
fyzický objekt.....	163, 120, 100, 99
fyzická věc	27, 23
fyzicko-informatické rozlišení	149
skládání portů.....	303
postpodmínka	328
postprocesní objektová sada	328, 170, 169, 105
předběžná podmínka	328, 264, 164
hodnocení splnění předpokladu	333
sada objektů předzpracování.....	328, 264, 170, 169
sada předzpracovávaných objektů.....	120, 106, 105, 104
pravděpodobnostní spotřeba propojovací ventilátor.....	372
pravděpodobnostní spojovací ventilátor	371
pravděpodobnostní výsledek propojení ventilátor	371
pravděpodobnost	149
procedurální odkaz.....	377, 162, 27, 19
Procedurální odkaz	171
jedinečnost procedurálního odkazu Princip OPM	167
omezení procedurální účasti	203
procedurální omezení účasti	203
procesní vztah.....	377
postup	242
proces	120, 101, 94, 82, 5
aktivace procesu.....	164
mechanismus aktivace procesu	327
třída procesu	290

kritéria procesu	120, 109
doba trvání procesu.....	340
rozvětvení procesu	214, 207
instance procesu.....	290
vyvolání procesu.....	331
omezení účasti v procesu	197
výkonnost procesu	329
testování procesu	194
procesy	88
Transformace objektů podle procesu Princip OPM.....	94, 82
orientované na proces	157
produkt.....	88
výrobní pravidlo	127
programový kód.....	151, 131
programovací jazyk.....	242, 151, 124, 123, 53
vlastnost.....	343, 342, 340, 195, 192, 188, 114
protokol	290
pseudokód	129
pseudoprocés	117
Veřejně dostupná specifikace	132, 127
kvalitativní atribut.....	255, 252
kvantitativní atribut.....	253
omezení rozsahu účasti	213, 199
RDF	231, 230, 229, 226, 225, 224
RDF graf.....	231
RDF trojice.....	229, 226, 225
pořadí čtení	230
zpětná vazba v reálném čase	130
reakce v reálném čase.....	130
vzájemný vztah	185
vzájemné strukturální propojení	189
vzájemný strukturální vztah	195, 188
vzájemný označený strukturální vztah.....	188
vzájemnost.....	195, 189, 188
vzájemnost značka	189
redundance	129
zdokonalitelné.....	349, 323, 322, 321, 318, 316, 312, 311, 309, 305, 302, 301, 281, 278, 239, 236, 234, 221, 218, 215
rafinovaný.....	322, 306, 302, 301, 281, 239, 218, 216, 215
rafinace	320, 42
mechanismus zdokonalení-abstrakce	295
relační databáze.....	30
diagram požadavků	143
diagram požadavků.....	136
Diagram požadavků.....	135
zjišťování požadavků	130
vyhrazená fráze OPL.....	168
vyhrazená fráze.....	257, 223, 218, 200
Vyhrazená fráze.....	14

Rámec popisu zdrojů	Viz RDF
odkaz na výsledek	180, 164
výsledek	179, 163, 59
reverzní inženýrství	75
povaha reverzního inženýrství	295
pravá hemisféra	129
role	169, 162
round-trip engineering	125
rountangle	13
rutina	242
runtime	341, 337
model běhu	380
scénář	264
věda	94, 75
derivace druhého řádu	248
odkaz na vlastní vyvolání	33
samoorganizovaná kritičnost	266
sémantická analýza vět	103
sémantika	91
sekvenční diagram	141, 139, 135
služba	246, 242, 88
nastavit	346, 229, 209
sdílená agregace	228
zkratková grafická notace	224
krátkodobý objekt	331, 162
přeskočit sémantiku	334
přeskočit sémantiku přednost principu OPM	355
SO AP 233 pracovní skupina	136
socio-technický systém	88
Sokratovská otázka	78
měkký atribut	255, 253
softwarové inženýrství	135
plný černý trojúhelník	224
omezení účasti zdroje	213, 201, 198
časoprostorový rámec	290
specializace	287, 281, 278
rozdělení na specifikované páry odkazů dovnitř a ven	352
rozdělení vstupního spojení	352
rozdělit vstupní a výstupní specifikovaný efektový pár odkazů	353
rozdělený výstupní odkaz	352
zainteresovaná strana	296, 87
zainteresované strany	6
normalizační orgán	149
stát	274, 257, 94, 82, 81, 16, 13, 12
Stát	274
míra změny stavu	344
věta o výčtu stavů	257
výraz stavu	259
stavový automat	265

diagram stavového automatu	139, 33, 29
stavový prostor	274
potlačení stavu	259
přechod stavu	261
Stavová transformace	261
vyjadřující stav – potlačující stav	295
stavový objekt	81
objekty se stavem	80
objekt bez stavu	81
proces udržující stav	223, 194, 184, 121, 116
propojení objektu a stavu	274, 270
propojení asociace stavu a objektu	269
odkaz na podmínku agenta specifikovanou stavem	343, 340, 339
odkaz na událost agenta specifikovanou stavem	331
odkaz na agenta určeného státem	171
odkaz na agenta specifikovaný stavem fan	369, 368
odkaz na charakterizaci specifikovanou státem	285
odkaz na událost spotřeby specifikovanou státem	332
odkaz na spotřebu specifikovanou státem	180, 173
odkaz na spotřebu stanovený státem ventilátor	369
Státem stanovený odběr ventilátoru	368
ventilátor s odkazem na řízení upraveným podle stavu	368
státem stanovený efektový odkaz	180, 175, 174
stavem specifikovaný odkaz na stav nástroje	340, 339
odkaz na událost nástroje specifikovanou státem	331
odkaz na nástroj specifikovaný státem	17
odkaz na nástroj určený státem fan	370
odkaz na postup stanovený státem	171
odkaz na výsledek stanovený státem	180, 174
stavem specifikovaný štitkovaný strukturální odkaz	268
odkaz na transformační událost specifikovaný stavem	331
odkaz mezi stavem a objektem	270
stereotyp	136
strukturální odkaz	195, 186, 183, 27, 19
strukturální omezení účasti	198
strukturální vztah	194, 183
strukturální vztahový pár	185
strukturální značka	186
strukturální pohled	67
struktura	97, 80, 75
Struktura	89
integrace struktury a chování	162
podproces	183
podskupina angličtiny	381, 226, 133, 132, 130, 129, 127, 123, 11
podsystém	85
dodavatel	87
plavecká dráha	141
diagram aktivit plavecké dráhy	141
synchronní proces	82

syntaxe	130, 91
SysML x, 1, 3, 4, 6, 29, 30, 31, 32, 33, 34, 35, 73, 75, 93, 114, 135, 136, 137, 138, 139, 142, 143, 144, 145, 146, 147, 148, 149, 150, 152, 153, 155, 156, 162, 167, 187, 200, 219, 227, 228, 229, 237, 242, 243, 244, 246, 247, 278, 297, 322	
SysML behaviorální diagramy	139
SysML blokový definiční diagram	278
Rámec diagramu SysML	31
Typ diagramu SysML	29
Příliš SysML	136
Specifikace SysML	137
Stereotyp SysML	31
systém	94, 83
systémový architekt	124
složitost systému	42
vývoj systému	295, 151
metodika vývoje systémů	295
Systémový diagram (SD)	19
provedení systému	262
jazyk modelování systému	296
systém diskurzu	288
systém systémů	75
systémový	150, 90
systémová věc	27, 24
rozdíl mezi systémovým a environmentálním	149
systém systémů	85
Jazyk modelování systémů	Viz SysML
značka	23
označený strukturální odkaz	198, 187, 186, 23
technická specifikace	376
šablona	290
definice pojmu	377
nahrazení termínu	377
ukončení	158
ternární strukturální vztah	183
Tesperanto	376, 131
textová modalita	149, 123
Hra na napodobování	372
věc	121, 114, 97, 27, 5
věc	121, 114
Důležitost věci Princip OPM	97, 94, 81
kombinace věcí a vlastností	245
časová výjimka	331
časový záznam	262
časová jednotka	340
časovaná událost	383
časová osa	248, 230, 194, 110
časová osa Princip OPM	304, 50, 47
hrot	213, 207
sada předmětů hroty	225, 209

sada procesů hrotu	209
sada hrotů	235
hrotová věc	214, 207
sada hrotů	214
sada vidliček	210, 209
token	127
tolerance	267
dotek	253
transformovat	119
transformace	158, 94, 7
Transformace	120, 100, 82, 7
transformační spojení	180, 163
míra transformace	343
sémantika transformace	155
transformovaný	263, 162, 6
Odkaz na transformační událost	330
transformace odkazu	186, 163, 162
přechodný objekt	331, 162, 121, 117
přechodné objekty	117
přechod	261
přechodný	239
přechodný vztah	278
přechodný strukturální vztah	287, 195, 190, 188, 184
transitivita	195, 190, 188
trojúhelníkový symbol	218
spouštěč	327
Turingův test	377
Turingův test textová verze	377
UML 322, 297, 278, 247, 246, 242, 237, 229, 228, 227, 200, 187, 167, 162, 150, 149, 143, 139, 138, 137, 136, 135, 35, 32, 6	
UML diagram tříd	278
UML pro SE RFP	136
unární strukturální vztah	183
podčasová výjimka odkaz	355, 341, 264
proces zpracování podčasí	355, 341
rozkládání–skládání	295
jednosměrná výchozí nulová značka	195, 192
jednosměrný modelově specifický nulový tag	193
Unifikovaný modelovací jazyk	Viz UML
sjednocení	328
univerzální stavební blok	97
nestabilní objekt	274
nestabilní stav	261
případ použití	167, 29
diagram případů použití	139, 29
předmět diagramu případů použití	31
diagram případů použití	169
uživatel	87
uživatелеm definovaná obousměrná nulová značka	193

uživatелем definovaný vyhrazený symbol	218
uživatелем definovaná nulová značka	195, 193
uživatелем definovaný jednosměrný nulový tag	193
hodnota	274, 40
odkaz na změnu hodnoty	180
hodnota efektu odkazu	177
odkaz na nastavení hodnoty	177
odkaz na procedurální hodnotu	177
vytvoření zobrazení	295
sémantika čekání	334
bílý diamant	228
celek-část	221, 22
výnosová míra	343