Objektově orientovaná analýza v systémovém inženýrství

Object-oriented analysis in system engineering

Bc. Pavel Pospíšil

Diplomová práce
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2009/2010

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Pavel POSPÍŠIL**
Osobní číslo:	**A07340**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Objektově orientovaná analýza v systémovém inženýrství**

Zásady pro vypracování:

1. **Možnosti a použití jazyka UML.**
2. **Vymezení systémového inženýrství.**
3. **Rozšíření jazyka UML pro návrh systémů.**
4. **Případová studie využití UML pro návrh systémů.**
5. **Aplikace vybraných metod na reálný projekt.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **ARLOW, Jim, NEUSTADT, Ila. UML 2 a unifikovaný proces vývoje aplikací : Objektově orientovaná analýza a návrh prakticky. 2007. vyd. Praha : Computer Pres a.s., 2007. 560 s. ISBN 978-80-251-1503-9.**
2. **PITMAN, Neil, PILONE, Dan. UML 2.0 in a Nutshell. 2005th edition. Sebastopol : OReilly Media, 2005. 234 s. ISBN 0-596-00795-7.**
3. **JALLOUL, Ghinwa. UML by Example. 1st edition. [s.l.] : Cambridge University Press, 2004. 249 s. ISBN 0521810515.**
4. **UML Resource Page [online]. [cit. 2009-03-29]. Dostupný z WWW: http://www.uml.org/.**
5. **OMG SYSML [online]. 2009 [cit. 2009-12-28]. Dostupný z WWW: http://www.omgsysml.org/.**
6. **FRIEDENTHAL, Sanford. Practical Guide to SYSML. [s.l.] : [s.n.], 2008. 576 s. ISBN 0123743796.**
7. **WEILKIENS, Tim . Systems Engineering with SysML/UML: Modeling, Analysis, Design. [s.l.] : [s.n.], 2007. 307 s. ISBN 978-0-12-374274-2.**

Vedoucí diplomové práce: **Ing. Radek Šilhavý, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce: **19. února 2010**

Termín odevzdání diplomové práce: **8. června 2010**

Ve Zlíně dne 19. února 2010

prof. Ing. Vladimír Vašek, CSc.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.
1

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Daná diplomová práce se zabývá rozšířením jazyka UML, známým jako SysML. Zpracováním materiálů vznikly podklady základů softwarového inženýrství a jazyka SysML, které mohou sloužit pro seznámení s možnostmi využití tohoto jazyka pro výuku nebo pro jeho praktické využití.

Obsahem práce je základní seznámení se softwarovým inženýrstvím, výčet používaných diagramů v jazyce SysML a ukázka použití diagramů na praktickém příkladě.

Klíčová slova:

UML, SysML, softwarové inženýrství.

ABSTRACT

This thesis deals with extension of UML known like SysML. These documents contains basics of software engineering and language SysML and should be used as manual texts for students or for practical using.

The thesis consists of the following parts: basics of software engineering, SysML diagram specifications and practical using of these diagrams.

Keywords:

UML, SysML, Software engineering.

PODĚKOVÁNÍ, MOTTO

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Radkovi Šilhavému za poskytování rad, připomínky při návrhu a realizaci této diplomové práce.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně …….

Podpis diplomanta

OBSAH

ÚVOD

Cílem této diplomové práce je obeznámení se základy softwarového inženýrství a jazyka SysML, který je rozšířením jazyka UML.

Tvorba a ukázky jazyka jsou prezentovány v programu Enterprise Architect. Vlastníkem toho produktu je australská firma Sparx Systems.

Enterprise Architect je vizuální modelovací platforma s možností:

- Komplexní UML analýzy s použitím návrhových nástrojů.
- Obsáhlých modelovacích možností pro obchod, software a systémy.
- Plné propojitelnosti od požadavků k rozmístění.
- Programování v desítkách jazyků, atd.

Práce je rozdělena do několika částí:

- Co je to UML?
- Systémové inženýrství
- SysML – systémový modelovací jazyk
- Aplikace na reálný projekt

Co je to UML je krátké vysvětlení jazyka UML. Druhá část systémové inženýrství je obecný a základní pohled na vnímání pojmu systémové inženýrství, jeho původ, příklady systémů, náhled a možnost využití jako profese. Třetí část SysML je seznámení s jazykem doplněné o popis používaných diagramů v jazyce SysML s jejich krátkým popiskem. Poslední a neobsáhlejší část je praktická část. Tato část obsahuje náhledy na dva reálné řešené příklady popisující možnosti využití SysML jazyka v praxi. První příklad návrh části designu v autě řeší modelování pouze jednoho systému a detailně popisuje postup tvorby a využití nástrojů pro jeho tvorbu. Druhý příklad automaticky řízený stěrač je komplexnější a řeší i možnosti vzniku kombinovaných diagramů.

Možnou alternativou k Enterprise Architect je například Magic Draw nebo Artisan Studio.

I. TEORETICKÁ ČÁST

1 CO JE TO UML?

The Unified Modelling Language (UML) je grafický jazyk pro vizualizaci, specifikaci, konstrukci a dokumentaci částí programového systému. UML nabízí standardní způsob, jak psát systémové projekty včetně koncepčních částí, což jsou obchodní procesy a systémové funkce nebo konkrétní části, což jsou specifikace programovacího jazyka, databázová schémata a znovupoužitelné programové komponenty [1].

1.1 Historie jazyka

Vývoj jazyka UML začal v roce 1994, kdy Grady Booch a Jim Rumbaugh začli pracovat na sjednocení Boochovi a OMT (Object Modeling Technique) metody. V roce 1995 se připojil Ivar Jacobson a vznikla OOSE (Object-Oriented Software Engineering) metoda.

Všichni tři autoři byli motivováni vytvořit spojující modelovací jazyk ze tří důvodů:

- všechny tří metody se nezávisle vyvíjely a nebyl důvod tak dále pokračovat kvůli rozdílům, které by způsobily komplikace pro konečné uživatele
- společná sémantika a notace by přinesla stabilitu pro vývoj softwaru
- očekávali, že spojení umožní řešení dřívějších problémů

Snaha Booche, Rumbaugha a Jacobsena vyústila v říjnu v roce 1996 ve vydání verze UML 0.9 a dokumentů verze 0.91. V roce 1995 se rozhodli Ivar Jacobson a Richard Soley pro standartizaci metod pod hlavičkou OMG (Object Management Group).

Během roku 1996 dochází k rozvoji UML jako strategického prvku pro mnoho společností. OMG nabízí záštitu pro tyto organizace a vydává první definici jazyka UML 1.0. Mezi tyto organizace patří například HP, IBM, Microsoft, Oracle a další.

V lednu roku 1997 vzniká verze UML 1.1 jako vylepšení UML 1.0, společně s novými prvky nově připojených společností do této komunity [2].

V dalších letech vznikaly další verze:

- 1998 - UML 1.2
- 1999 - UML 1.3
- 2001 - UML 1.4

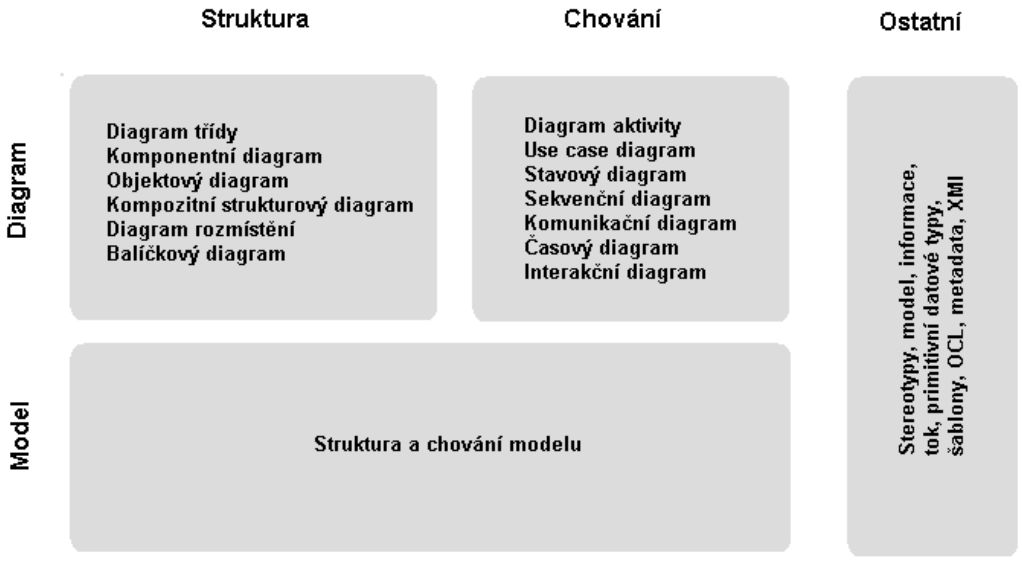
Poslední vydanou verzí je UML 2.0 [3].

1.2 Struktura a návrhy

UML je velmi rozšířený jazyk a může na uživatele na první pohled působit nepřekonatelně. Po zvládnutí znalostí základních struktur, už takovým problémem není.

Existují dvě úrovně základní části UML jazyka, struktura a element chování. Struktury jsou například třídy nebo komponenty. Ty se používají k popisu systémových struktur. Funkce popisují chování elementů. Mezi ně patří activity, stavové přístroje a interakce. Položka "Ostatní" obsahuje elementy, které patří ke strukturám i chování.

Dalšími prvky jsou model a diagram. Model obsahuje kompletní popis systému. Naproti tomu diagram obsahuje pouze vizualizaci modelu s ohledem na určité aspekty. Struktura UML je zobrazena na následujícím obrázku [4].



Obr. č. 1 - Celková struktura UML

2 SYSTÉMOVÉ INŽENÝRSTVÍ

Pro systémové inženýrství existuje mnoho definic. Jedna z nejvíce používaných je tato:

Funkcí systémového inženýrství je způsob řešení inženýrství komplexních systémů. Způsobem je zde myšleno to, jak vést, spravovat, řídit systémy podle svých zkušeností. Tato charakterizace určuje, jak vybírat správný směr pro dané řešení. Inženýrství je pak aplikace těchto principů do praxe (návrh, tvorba, nástroje a systémy). Systém je sada spojujících se částí pracujících k nějakému společnému cíli [4].

2.1 Původ systémového inženýrství

Původ systémového inženýrství není spojen s žádným konkrétním časovým obdobím. Principy na určitém ševelu byly použity již při stavbě pyramid a možná i dříve.

Uznání systémového inženýrství jako významného odvětví je nejčastěji spojováno s 2. světovou válkou. Nejvíce se systémové inženýrství projevilo v druhé polovině 20. století s prudkým vývojem technologií a jejich aplikací v armádě a obchodě.

Během 2. světové války došlo k rozvoji technologií. Vývoj vysoce výkonných letadel, radarů, raket a zejména atomová bomba vyžadovaly revoluční přístup k aplikacím, materiálům a informacím. Tyto systémy byly komplexní a vyžadovaly kombinované technické disciplíny a jejich vývoj představoval posun v inženýrství. Nedostatek času při vývoji technologií způsobil nový přístup k řešení plánování, technické koordinace a inženýrského managementu.

Během studené války došlo opět k rozvoji technologií, kontrolních systémů a materiálů. Nicméně za největší rozvoj se považuje vývoj polovodičové elektroniky. Dalším prvkem vývoje byl osobní počítač, který výrazně zvýšil složitost systémů a je důležitou součástí systémového inženýrství.

Vztahy moderního systémového inženýrství si lze představit pomocí tří základních faktorů:

- Progresivní technologie, která poskytuje příležitost pro zvýšení výkonnosti systému, ale také představuje vývojové rizika spojené se správou systémového inženýrství.
- Konkurence, jehož různé formy vyžadují specifické systémové řešení s využitím kompromisů mezi různými přístupy.

- Specializace, která vyžaduje rozdělení systému na bloky korespondující k specifickému typu produktu a jejich navržení specialisty pro přesnou správu jejich rozhraní a interakcí.

2.1.1 Progresivní technologie – rizika

Enormní růst technologií v druhé polovině dvacátého století byl největším faktorem ve vývoji systémového inženýrství a esenciálním prvkem v inženýrství komplexních systémů. Progresivní technologie nebyla jen možností jak rozšířit možnosti stávajících systémů, ale také dopomohla ke vzniku nových systémů. Změny v technologiích od základů změnily pohled na inženýrství, produkci a operace.

Moderní technologie měla výrazný vliv na přístup k inženýrství. Obvykle inženýrství aplikovalo pravidla vedoucí k praktickému konci. Inovace, ale způsobila vznik nových materiálů, ovladačů a procesů, jejichž vlastnosti už nebyly jednoznačné a snadno pochopitelné. Takto vzniklé aplikace představovaly riziko spojené s nečekanými vlastnostmi systému a projevovaly se sníženým výkonem, nákladnými opravami a zpomaleným výkonem.

Nicméně odmítnutí nové technologie může také nést svá rizika. Rizikem je tvorba méně kvalitního systému. Takový systém se může stát předčasně zastaralý. Pro úspěch je tedy nutné si pečlivě zvolit technologická rizika a překonat je kvalitním návrhem, systémovým inženýrstvím a programovou správou.

Přístup systémového inženýrství k dřívějším aplikacím ztělesňuje praxe v rizikovém managementu. Rizikový management je proces analyzování rizik, jeho vývoj, testování a omyly.

Řešení rizik je jedním z největších problémů systémového inženýrství vyžadující rozsáhlé znalosti celého systému a jeho kritických částí. Především to znamená, že systémové inženýrství je centrem rozhodování, jak dosáhnout nejlepší rovnováhy mezi riziky, a který systém nejlépe využije výhod nových technologií.

2.1.2 Konkurence – kompromisy

Konkurenční tlaky na vývoj systémových procesů se objevil na několika úrovních. V případě obranných systémů se objevují se zvyšujícím se počtem nepřátel a jejich

sníženou efektivní obranou vůči nim. Takové tlaky pak vedly k zvýšení efektivity těchto systémů.

Dalším zdrojem konkurenčního prostředí je soupeření firem o vývoj nových systémů. Každý zájemce se snaží vytvořit nejlevnější a nejefektivnější produkt na trhu.

Všechny formy konkurence se snaží vynaložit úsilí na vytvoření tlaku na vývoj systémových procesů tak pro nejlepší výkon, cenově dostupný systém za nejkratší možný čas. Proces výběru nejlepšího přístupu vyžaduje výběr z mnoha alternativních řešení, praxi a odhad při řešení. V tomto případě se lze pak odkázat na konkurenční analýzy a základy systémového inženýrství.

2.1.3 Specializace – rozhraní

Komplexní systémy vykonávající různý počet funkcí musí být konfigurovány takovým způsobem, aby každá významná funkce pracovala v oddělených komponentech, kde by se mohla vyvíjet a být testována jako individuální entita. Takovéto začlenění dává výhodu organizaci pro členění různých typů produktů a zároveň lze pak jednotlivé komponenty vyrábět ve vysoké kvalitě za nejnižší ceny.

Rozsáhlost a rozlišnost inženýrských znalostí, které stále rostou, ji donutily rozdělit vzdělání a praxi do různých specializací. Tato specializace se pak stala základní podmínkou pro vývoj systémových procesů a vznik rozdělených komplexních systémů.

Výhodou těchto rozdělených komplexních systémů jsou individuální stavební bloky, které pak spolu pracují jako jeden efektivní operační systém. Integrace znamená, že jednotlivé stavební bloky do sebe dobře zapadaly a zároveň také do vnějšího prostředí, s kterým byly propojeny. Spojení nebylo pouze fyzické ale i funkční a jeho podoba ovlivňovala i další elementy. Fyzické propojení byla fyzická hranice nazvaná jako rozhraní a funkční hranice byla pojmenována jako interakce.

Přímým důsledkem rozdělení systémů do stavebních bloků je koncepce modularity. Modularita je míra společné nezávislosti jednotlivých komponent systému. Cílem systémového inženýrství je pak dosáhnout co největšího stupně modularity. To dosáhneme tak, že rozhraní a interakce budou co nejjednodušší pro efektivní výrobu, systémovou integraci, testování, operační údržbu, spolehlivost a snadnou aktualizaci. Proces rozdělení

systému do modulárních stavebních bloků se nazývá funkční alokace a je další částí systémového inženýrství.

2.2 Příklady systémů vyžadujících systémové inženýrství

Všeobecnou definicí systému je sada souvisejících komponent, pracujících společně jako celek snažící se dosáhnout společného cíle. Většina domácích spotřebičů využívá mnoho užitečných operací systematickým způsobem, zahrnuje to jednu nebo dvě inženýrské disciplíny a jejich design je založen na osvědčené technologii. Nesplňují však kritérium komplexního systému.

Vývoj moderních systému je silně řízen technologickými změnami a to se stalo další charakteristickým požadavkem pro systémové inženýrství. Charakteristiky systému a jejich vývoj, testování a požadavky aplikací vyžadují praxi v systémovém inženýrství a je to tedy:

- Inženýrský produkt splňující specifické potřeby.
- Obsahuje rozdílné komponenty, které mají spolu složité vztahy a jsou multidisciplinární a relativně komplexní.
- Používá pokročilou technologii v mnoha směrech, které jsou centrem výkonu primárních funkcí a proto mají vývojová rizika a často relativně vysokou cenu.

Inženýrský nebo komplexní systém musí tedy splňovat všechny tři tyto parametry.

2.2.1 Ukázky komplexních inženýrských systémů

Systém	Vstupy	Procesy	Výstupy
Meteorologický satelit	Obrázky	Přenos dat	Kódované obrazy
Letecký rezervační systém	Cestovní požadavky	Správa dat	Rezervační lístky
Lokační systém nákl. vozidel	Požadavky pro směrování nákladu	Komunikace Pohyb po mapě	Cest. informace Náklad

Tabulka č. 1 – Příklady inženýrských komplexních systémů

Ukázka je zobrazena v tabulce se svými hlavními vstupy, procesy a výstupy. Systém obsahuje rozmanité prvky a některé z nich jsou mohou být komplexní a mít dokonce i vlastní pravidla.

2.3 Náhled na systémové inženýrství

Náhledem systémového inženýrství je hledání cíle systému jako celku a jeho úspěšné řešení. To znamená závislost individuálních cílů a atributů na celém systému.

2.3.1 Úspěšný systém

Největším zájmem systémového inženýrství je úspěšnost systému. To zahrnuje požadavky, cíle vývoje a dlouhý užitečný operační život. Náhled na systémové inženýrství zahrnuje všechny tyto cíle. Systém se snaží pochopit problémy uživatele a ekologické podmínky během jeho průběhu. Cílem je zřízení technického přístupu, který usnadňuje údržbu operačního systému a poslouží k možnosti pozdějšího upgradu systému. Snaží se odhadnout vývojové problémy a řeší je před jejich vznikem.

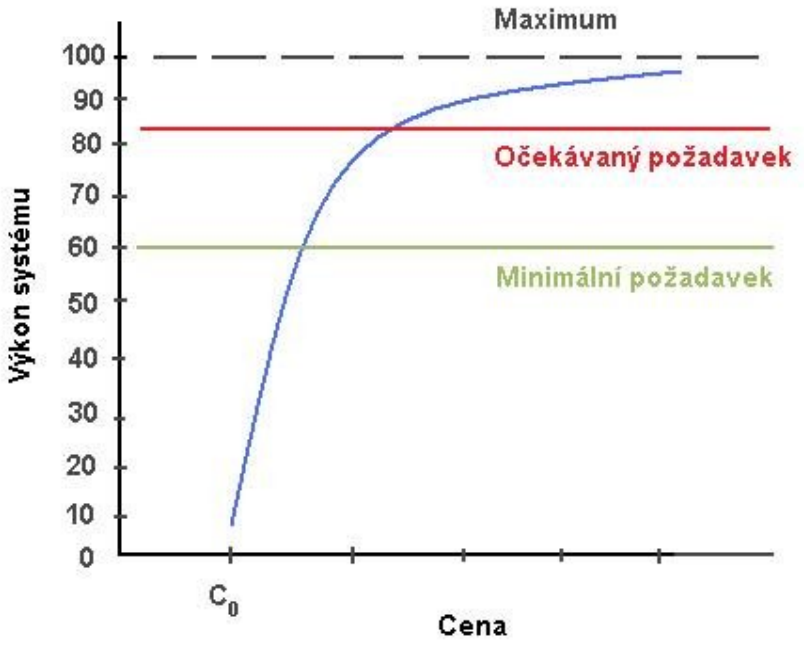
Vývoj úspěšného systému vyžaduje užití konzistentního inženýrského přístupu. To zahrnuje praxi v metodickém a disciplinovaném řízení s rozšířeným plánováním, analýzou, hodnocením a dokumentací. Další často přehlíženou vlastností je inovace. Pro úspěšné dokončení nového komplexního systému je nutné použít nejnovějších technologických postupů. Ty představují nová rizika, které vyžadují pak značné úsilí při řešení nových návrhů. Výběr nejvíce nadějných technologických přístupů, přináší svá rizika, plánování kritických pokusů a rozhoduje o potenciálních nouzích, které jsou základní povinností systémového inženýrství.

2.3.2 Nejlepší systém

V charakterizaci inženýrských systémů existují dva náhledy. První nejčastěji zmiňované je, že nejlepší je nepřítel toho dobrého, a že systémové inženýrství je uměním toho dobrého. Naopak systémové inženýrství nedosahuje ve snaze toho nejlepšího možného systému a neposkytuje ani nejlepší výkon. Oblíbená maxima pro výkon systému, které se používají je nejlepší a dost dobrý. Náhled inženýrství na výkon systému je pak jeden z nejvíce kritických vlastností stejně jako přístupnost, časová dostupnost k uživateli a údržba

systému. Systémový inženýr se snaží o nejlepší rovnováhu kritických vlastností systému z hlediska úspěchu vývoje programu a ceny systému pro uživatele.

Nezávislost výkonu a ceny může být pak vysvětlena v termínech zákona o zkrácené návratnosti. Podle přesného technického přístupu k dosažení daného výkonu v systému vznikají typické varianty úrovně výkonu v závislosti výkonu na ceně systému. Varianty sou zobrazeny na obrázku.



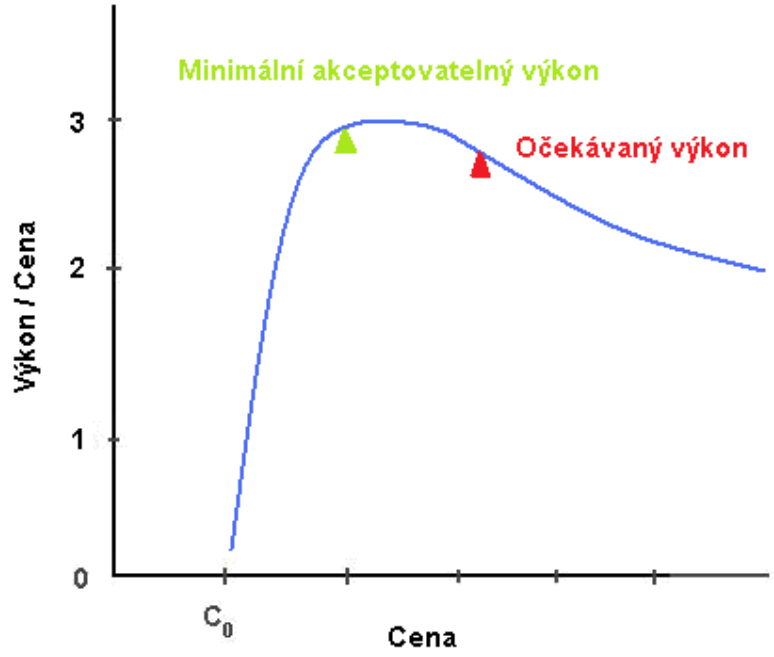
Obr. č. 2 – Výkon vs. cena

Horizontální část grafu reprezentuje teoretické limity ve výkonu závislé na daném technickém přístupu. Více důmyslnější systém může produkovat vyšší limity za odpovídající vyšší cenu. Barevné vodorovné limity odpovídají minimální hodnotě očekávané hodnotě výkonu systému.

Křivka začínající v C_o reprezentuje cenu již dosaženého výkonu systému. Zakřivení je zpočátku prudké a asymptoticky se přibližuje k teoretickému limitu. Toto snížené zakřivení je míra zvýšeného výkonu za vyšší ceny.

Během vývoje nových technologií došlo k novému technickému přístupu, zejména u automobilů, kde se zvyšující cena stala krajním limitem pro jejich vývoj. Vyváženost se

stala mezí krajních limitů výkonu. Přístup určující tuto vyváženost je na následujícím obrázku.



Obr. č. 3 – Výkon / Cena vs. Cena

Obrázek zobrazuje závislost výkonu dělené ceny na ceně. Tento podíl je ekvivalentem konceptu cenové efektivity. Křivce se po dosaženém maximu snižuje efektivita. To poukazuje na to, že výkon nejlepšího systému má blíže k tomu přístupu, kde je bod očekávaného výkonu za minimálním akceptovaným výkonem.

2.3.3 Vyvážený systém

Podle definice vzhledem k systémového návrhu je vyváženost harmonické a vyhovující uspořádání nebo podíl jednotlivých části či elementů v návrhu či kompozici. Základní funkcí systémového inženýrství je přinášet vyváženost mezi různými komponentami systému. Ty pak slouží k optimalizaci vlastností jednotlivých komponent.

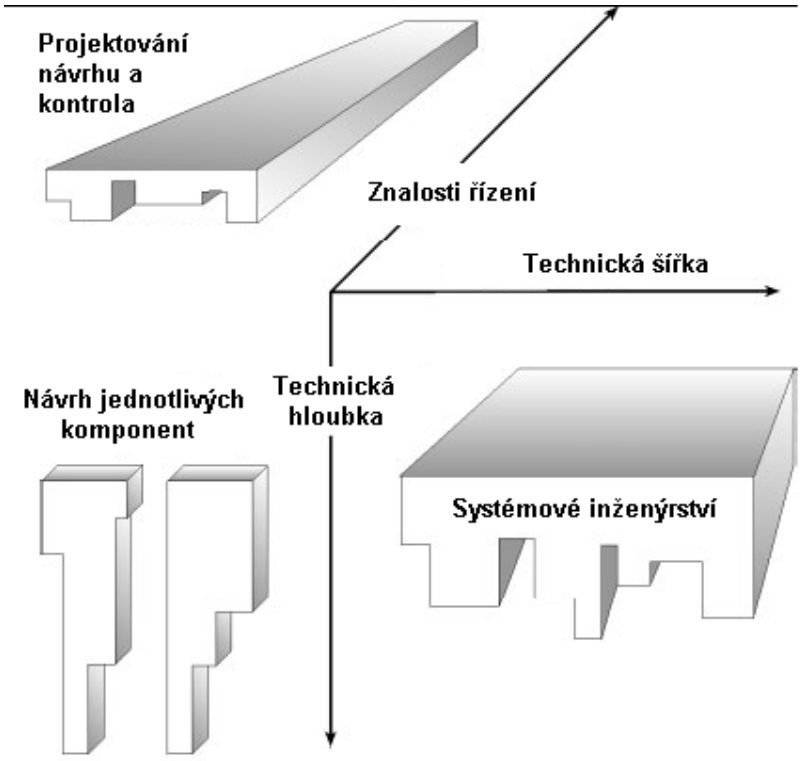
Při návrhu se očekává, znalost systému, kde každá možná kombinace prvků přináší specifické vlastnosti při jejich vývoji a je nutné se zaměřit pozornost na sledování výskytu možných chyb při technické expertíze tohoto systému. Dále musí inženýr sledovat celkový

výkon systému, rizika při vývoji, cenu a systémovou životaschopnost a tak se snaží o optimální chování celého systému.

2.3.4 Vyvážený náhled

Systémový náhled také znamená zaměření na vyváženost, kde žádný systémový atribut nesmí být zvýhodňován před jiným. Všechny kritické vlastnosti systému jsou nezávislé a charakteristická vyváženost musí být řízena rozhodnutím podle daného návrhu. Charakteristiky jsou sice nesourodé, ale musejí být řešeny podle funkčnosti systému.

Systémy tedy vyžadují různé znalosti pro specifické odvětví. Povaha těchto rozdílu je zobrazena na obrázku č. 4. Dimenze reprezentují technickou hloubku, technickou šířku a řízení. Designér má tedy hluboké znalosti v jedné z mnoha oblastí technologie a projektový manažer pak potřebuje částečné technické znalosti v mnoha odvětvích a musí mít schopnost řídit lidi. Naopak systémový inženýr má obsáhlé znalosti ve všech třech dimenzích a převyšuje ostatní spolupracovníky ve svých znalostech.



Obr. č. 4 – Dimenze návrhu, systémového inženýrství, projektování návrhu a kontroly.

2.4 Systémové inženýrství jako profese

Navzdory zvýšeného rozšíření komplexních systémů do moderního společenství a již významné roli systémového inženýrství ve vývoji systémů, se systémové inženýrství jako profese nestalo dosud široce prozkoumanou oblastí. Základní uznání získalo ve firmách specializujících se na vývoj obsáhlých systémů.

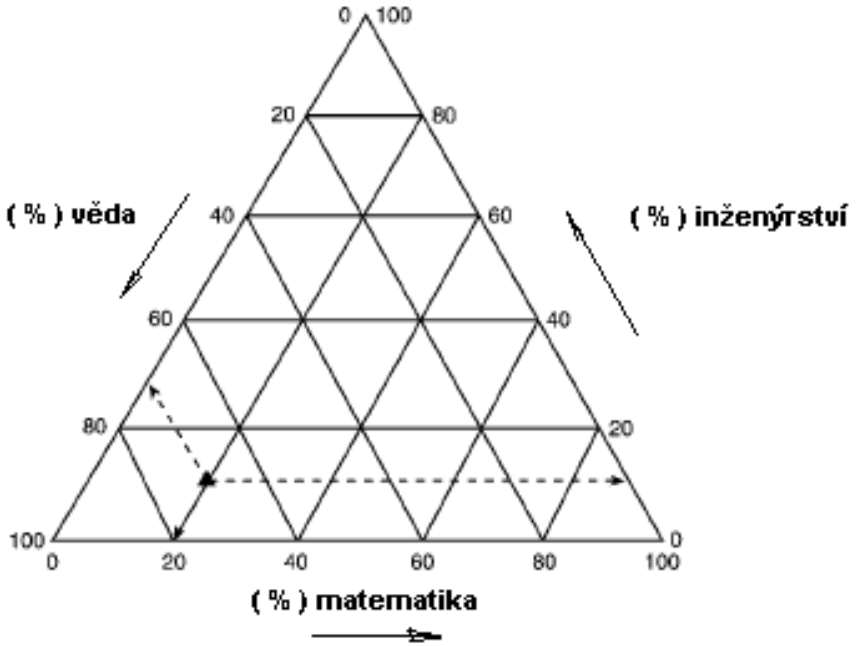
Snad největším důvodem zpomalení rozmachu tohoto oboru jako zaměstnání je, že neodpovídá tradiční akademické inženýrské disciplíně. Inženýrské disciplíny by měly být postaveny na vztazích založených na fyzikálních zákonech, měřitelných vlastnostech materiálů, energie a informací. Systémové inženýrství na druhou stranu zabývá problémy, které nejsou dosud vysvětleny, užívá proměnné bez známých porovnání, a kde rovnováha mezi konfliktními objekty zahrnuje i nesouměrné atributy. Absence kvantitativních znalostí systémů byla důvodem, proč se systémové inženýrství nestalo unikátní vědeckou disciplínou.

Navzdory těmto překážkám byly potřeba systémy v průmyslu a byly založeny programy nabízející tituly v oboru systémového inženýrství.

2.4.1 Orientace v technické profesi

Obecně je orientace v technické profesi řízeno modelem podle tří ortogonálních vektorů, kde každý reprezentuje rozsah individuální orientace ve vědě, matematice nebo inženýrství.

Reprezentaci si lze pak představit jako model obsahující všechny tři komponenty a je zobrazena na následujícím obrázku.



Obr. č. 5 – Technický orientační fázový diagram

Tato specializace na jednu z těchto profesí pak způsobuje nedostatečné znalosti v jiných oblastech. Řešením je pak aktivní spolupráce specialistů v různých oborech. Systémoví inženýři se tedy vyvíjeli, podle grafu tak, že se snížila specializace pro jednotlivé obory, ale zvýšila specializace ve více oborech. Tím se pak systémoví inženýři stali technickými leadery vývoji systémů a programů.

2.4.2 Výzva systémového inženýrství

Blokujícím faktorem pro profesionalizaci systémového inženýrství je výchylka od klasických zavedených disciplín k více rozmanité, komplikované a problematické profesionální praxi. Vyžaduje to zejména časovou investici a úsilí o výrazné rozšíření inženýrské základny a také učení komunikačních a manažerských schopností.

Cesta této profese je složitá, ne často dobře placená a málo atraktivní. Využití klasického inženýrského vzdělání je limitováno a přináší málo informací pro tuto profesi. Pro systémového inženýra je úspěch měřen reálnou nepřítomností programových problémů než efektními úspěchy.

2.4.3 Atributy a motivace inženýrů

Pro identifikaci vhodných kandidátů pro inženýrskou kariéru je vhodné určit charakteristiky specifický pro tento obor. Očekává se zejména dobré znalosti v oblasti matematiky a vědy.

Systémový inženýr pracuje v multidisciplinárním prostředí a chápe podstatu příbuzných oborů. Nadání pro vědu a inženýrství pomáhá v práci, protože snadněji umožňuje jedinci chápat podstatu nových oborů. Je schopen kreativně řešit problémy a jeho řešení je pro něj stává největší výzvou. Následující vlastnosti jsou běžné pro úspěšné systémové inženýry:

- Radost z učení nových věcí a řešení problémů.
- Radost z výzvy.
- Skepse u neosvědčených výroků.
- Otevřená mysl pro nové nápady.
- Dobrá průprava ve vědě a inženýrství.
- Technické úspěchy ve specializované oblasti.
- Znalosti v některých inženýrských oblastech.
- Rychlé přijímání nových nápadů a informací.
- Dobré mezilidské a komunikační vlastnosti.

3 SYSML – THE SYSTEMS MODELLING LANGUAGE

SysML je obecný modelovací jazyk pro aplikace systémového inženýrství. Je to dialekt jazyka UML, průmyslového standardu pro modelování softwarově zaměřených systémů [8].

3.1 Historie

SysML je velice mladý jazyk a jeho historie je velice krátká. Nicméně vznikl jazyk, který je schopen dokonale porozumět objektům a strukturám.

UML byl využíván v mnoha projektech systémového inženýrství, ale chybělo mu standardizované rozšíření. Pro rozšíření bylo zapotřebí pokrýt speciální potřeby, včetně příkladů, jak modelovat různé formy požadavků podporující spojité funkce a distribuční struktury. V září 2001 společně OMG (Object Management Domain) a INCOSE založili SE DSIG (Systéme Engineering Domain Special Interest Group) pro vývoj standardizovaného rozšíření UML, který by sloužil jako modelovací jazyk pro specifikaci, návrh a potvrzení komplexních systémů. Podle toho standardizační proces firmy OMG vydal RFI (Request for Proposal), které bylo publikováno v únoru roku 2002 a obsahovalo informace o požadavcích k UML rozšíření. V březnu 2003 byl založen UML pro systémové inženýrství, kde byl popsán přibližný systém a základní návrh specifikovaných požadavků. Pracovní skupina jménem SysML Partners vznikla v květnu roku 2003. Tak vzniklo rozšíření SysML. Skupina byla složena ze zástupců průmyslu, vládních autorit, výrobců modelových nástrojů a dalších spolupracovníků. První jazykové rozšíření proběhlo v říjnu roku 2003. V roce 2005 se skupina SysML Partners kvůli sporům rozdělila na dvě skupiny. V listopadu roku 2005, pak obě skupiny navrhly kompletní systém se specifikacemi k OMG. Tyto zmatky pak vedly k tomu, že vznikly dvě verze a způsobily zmatek mezi uživateli. Na začátku roku 2006 se opět verze sjednotily. Počáteční neshody a rozdělení skupin v konečné fázi způsobily, že výsledná specifikace byla více kreativnější a kvalitnější.

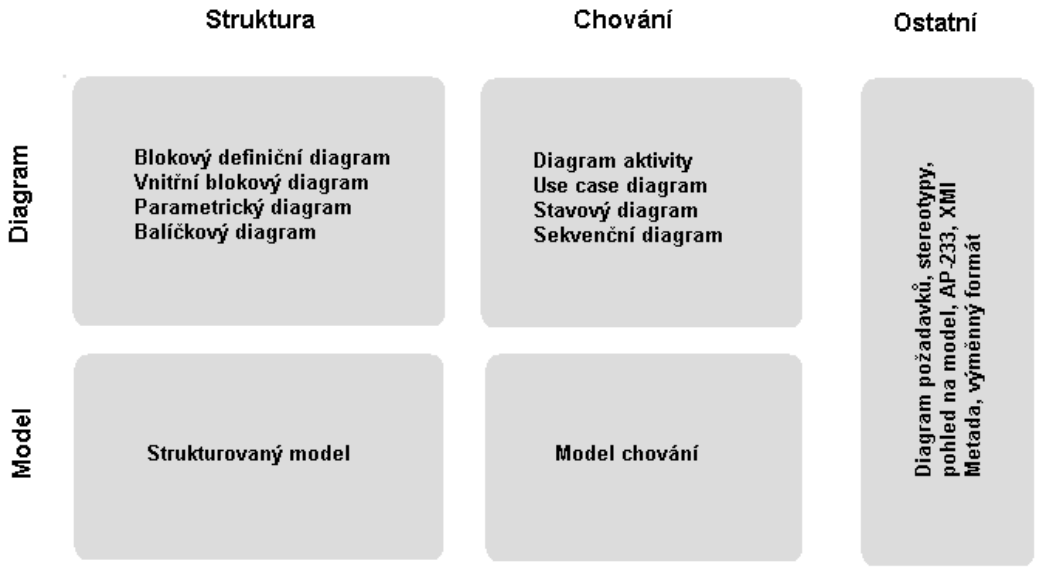
Na konci dubna roku 2006 u příležitosti srazu OMG byl SysML přijat jako standard. Nyní byla specifikace zastavena a připravena k použití bez obav z jejich změn. Finální standardizační proces zabral téměř rok a ukončil specifikaci. OMG publikovala SysML verzí 1.0 jako oficiální standard v listopadu roku 2007. Vývoj se nezastavil, verze 1.0

obsahovala chyby a některé prvky stále chyběly. Jak se UML vyvíjelo od verze 0.9 do 2.1.1., došlo k mnoha změnám reflektující zkušenosti získané praxí. Další verze nadále vznikají [5].

3.2 Struktura a návrhy

SysML a UML jsou podobné a jejich struktury a koncepty jsou také velmi podobné. SysML je rozšířením UML, ale vynechává UML prvky. Rozšíření představuje do SysML omezení stereotypů a několik nových diagramů. Stereotypy mohou být definovány v UML programech. Pouze nové diagramy vyžadují speciální podporu řešící pouze úroveň diagramu. SysML modelovací nástroje mají lepší podporu a nabízí vhodné funkce.

Absence některých UML prvků nemají v praxi výraznou roli. Při návrhu si je pak nutné dát pozor na to, že model není ani UML nebo SysML, ale je to kombinovaná nestandardní forma. Rozšíření struktury SysML o diagram požadavků je zobrazeno na následujícím obrázku.



Obr. č. 6 – Struktura SysML

3.3 SySML diagramy

Výčet diagramů je zobrazen na následující tabulce:

Název diagramu	Zkratka
Diagram aktivity	act
Diagram požadavků	req
Případy užití	uc
Blokový definiční diagram	bdd
Vnitřní blokový diagram	ibd
Balíčkový diagram	pkg
Diagram parametrů	par
Sekvenční diagram	sd
Stavový diagram	stm

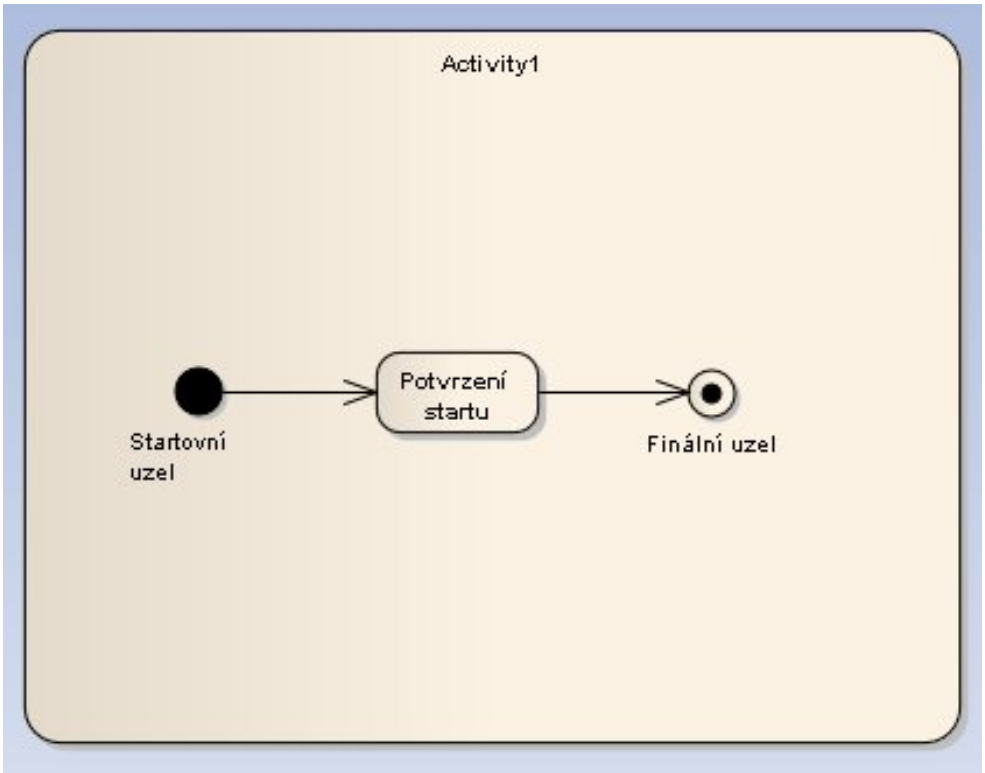
Tabulka č. 2 – SysML diagramy

Pro znázornění diagramů jsem zvolil 30-denní zkušební verzi programu Enterprise Architect od firmy Spart Systéme volně dostupnou na jejich stránkách: http://www.sparxsystems.com/. Pro zobrazení SysML diagramů se program rozšířil o nadstavbu dostupnou na stejných stránkách.

3.3.1 Diagram aktivity

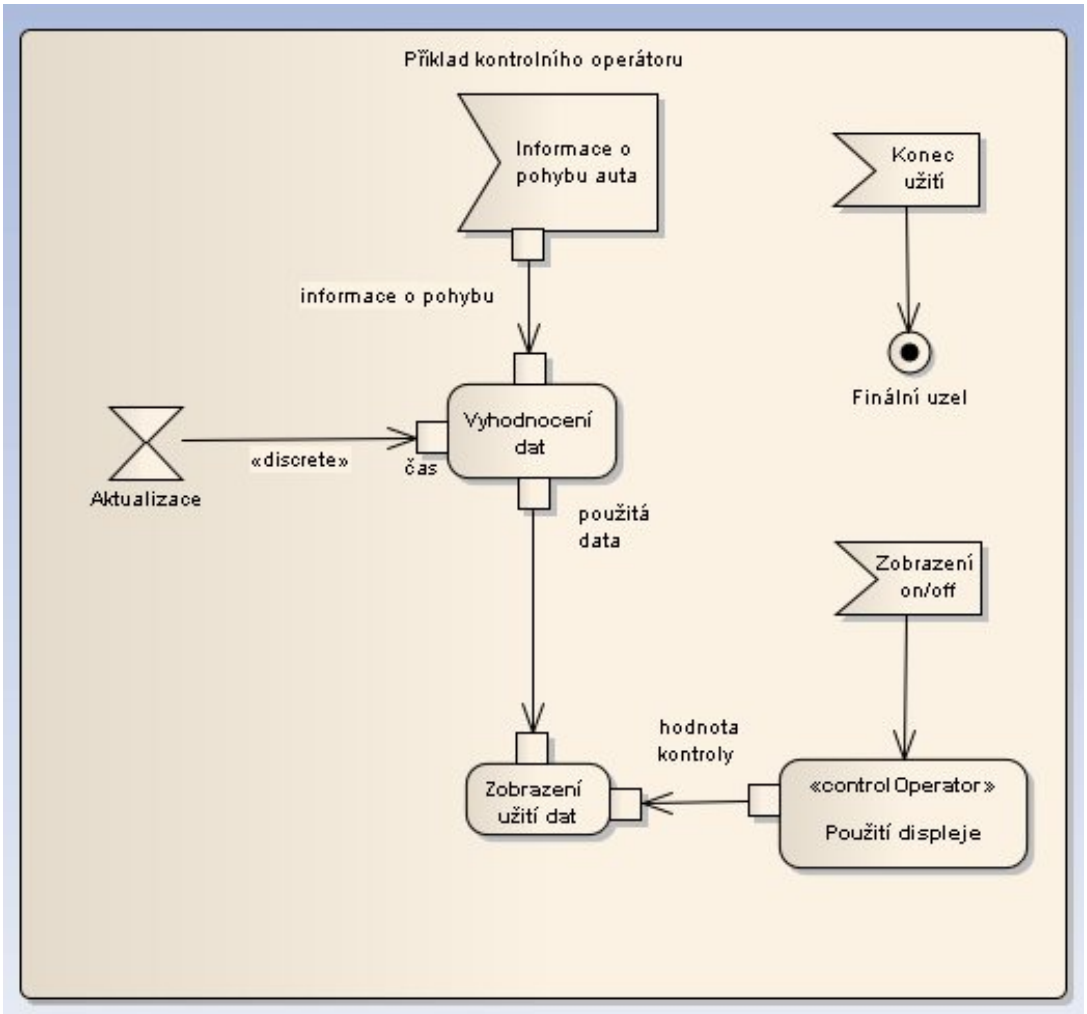
Diagram aktivity je využíván při popisu toků a jejich detailních algoritmů a operací. Tento diagram reprezentuje model popisující sekvenci elementárních akcí. Tato sekvence může být rozdělena do několika toků, které mohou být synchronizovány a rozdělovány podle různých podmínek.

Základní diagram se skládá ze startovního uzlu, akčního prvku a finálního uzlu. Všechny tyto části jsou uvnitř prvku aktivity. Ukázka je na obrázku č. 7.



Obr. č. 7 – Základní diagram aktivity

Komplexnější příklad obsahuje více prvků a je zobrazen na obrázku č. 8. Příklad obsahuje různé akční členy a jejich vazby. Konec využití je ukončen finálním uzlem.



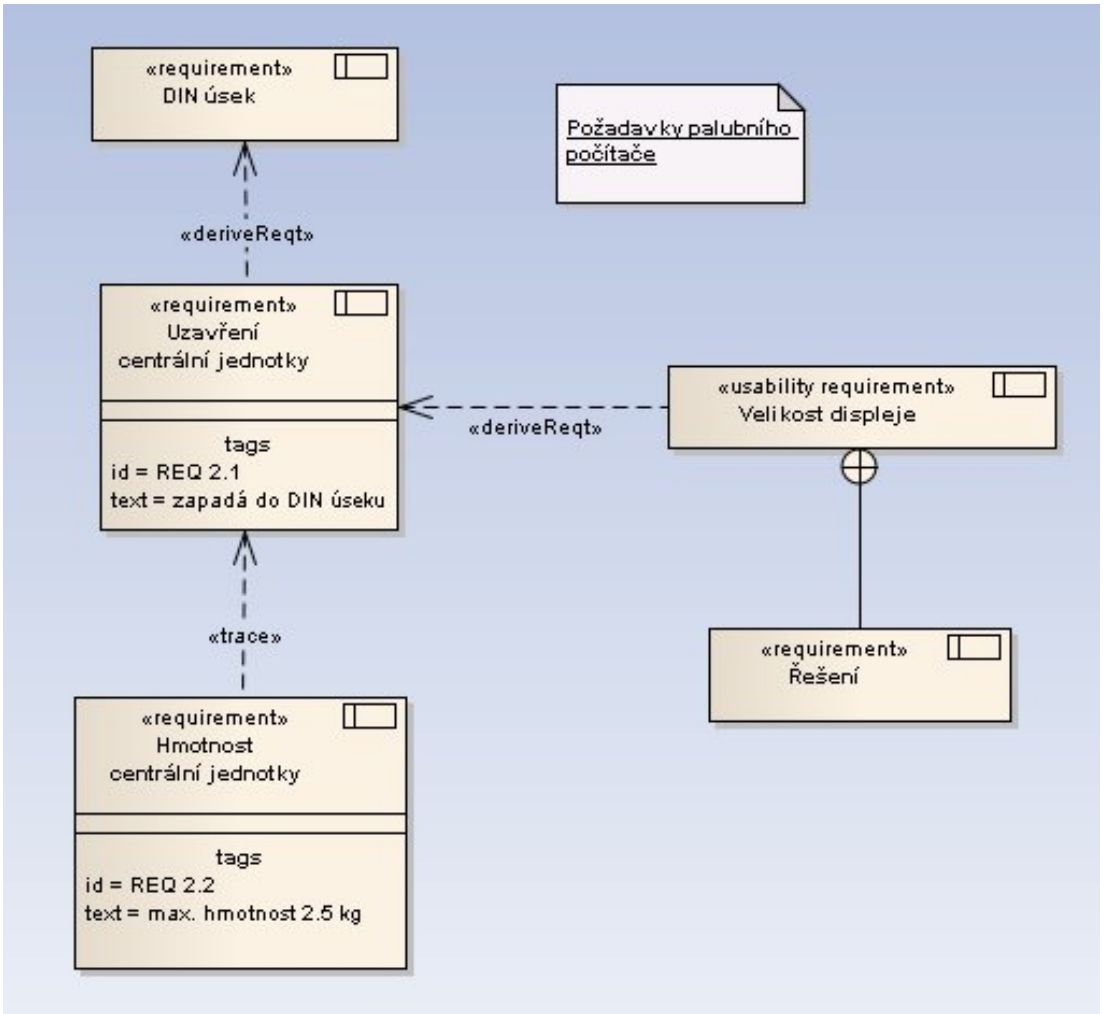
Obr. č. 8 – Komplexní příklad s využitím prvků aktivity

3.3.2 Diagram požadavků

Požadavek popisuje jeden nebo více vlastností nebo chování systému. Jedná se o stereotyp «requirement» (požadavek). Požadavky popisují vztah mezi vedoucím a ostatními, kteří vytvořili návrh systému a jeho implementaci. Požadavek specifikuje toky a podmínky po seznámení se systémem.

Požadavek je stereotypní třída. To znamená, že požadavek má sémantiku třídy rozšířenou o specifikaci a vlastnosti stereotypu. Dvě základní vlastnosti stereotypu jsou unikátní identifikátor ID a popisující text. Tyto vlastnosti sou použity pro modelováni požadavků.

Příklad použití diagramů požadavků je zobrazeno na obrázku č. 7.



Obr. č. 9 – Příklad diagramu požadavků

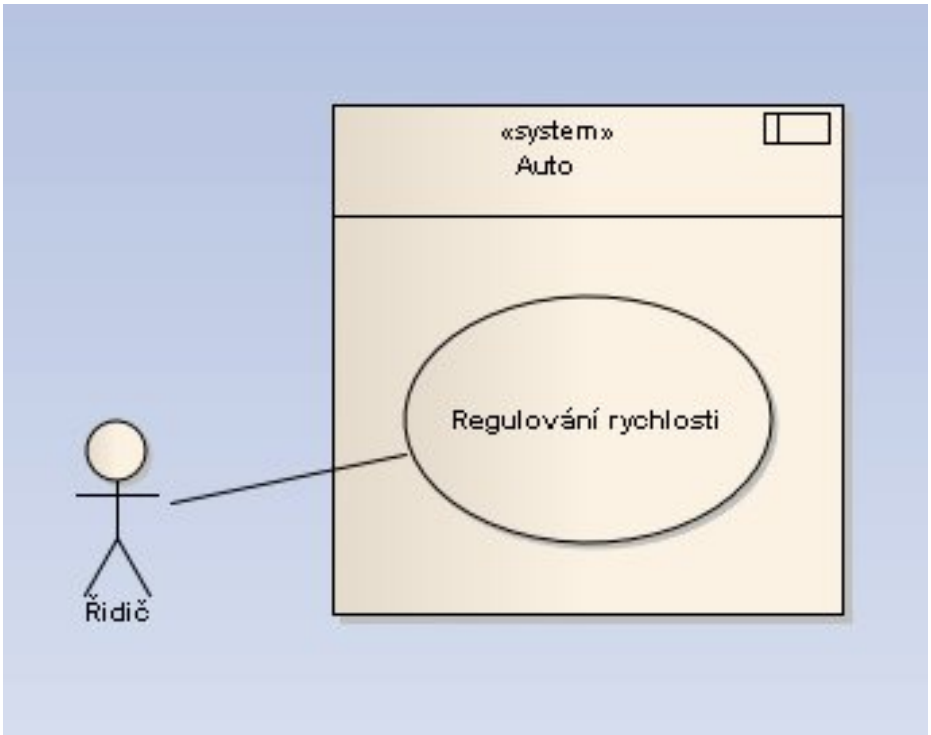
Obrázek popisuje požadavky palubního počítače. Stereotyp «deriveReqt» je odvozující požadavek a popisuje požadavek odvozený z jiného požadavku. Vztah obsažení (containment) na obrázku jako kulička s plus uvnitř popisuje, že požadavek je obsažen v jiném požadavku. Vztah «trace» (cesta) se používá k požadavkům na cestu. Některé požadavky jsou doplněny o hodnoty jako například identifikace (id) a popis požadavku (text).

3.3.3 Případy užití

Model případu užití zaznamenává požadavky systému. Pojmem případ užití je míněna komunikace mezi uživateli a dalšími partnery a nařizuje systému co dělat.

Případ užití zobrazuje interakci mezi systémem a prvky externího systému. Tyto externí entity jsou označovány jako aktér (Actor). Aktér reprezentuje roli uživatele, externího hardwaru nebo dalších systémů.

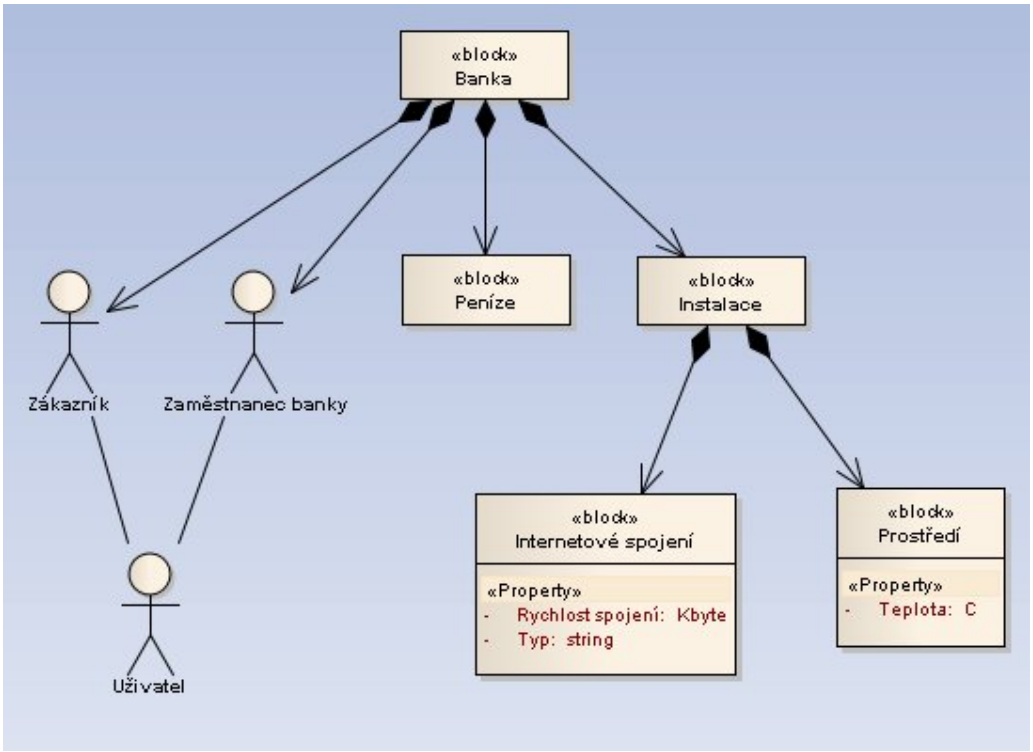
Příkladem použití je například regulování rychlosti auta řidičem.



Obr. č. 10 – Diagram příkladu užití

3.3.4 Blokový definiční diagram

Blokový definiční diagram je založen na UML diagramu s restrikcemi a rozšířením definovaným podle SysML. Diagram popisuje vztahy mezi různými bloky, což vede k systémového hierarchii nebo klasifikaci. Příklad vypadá například takto [9]:



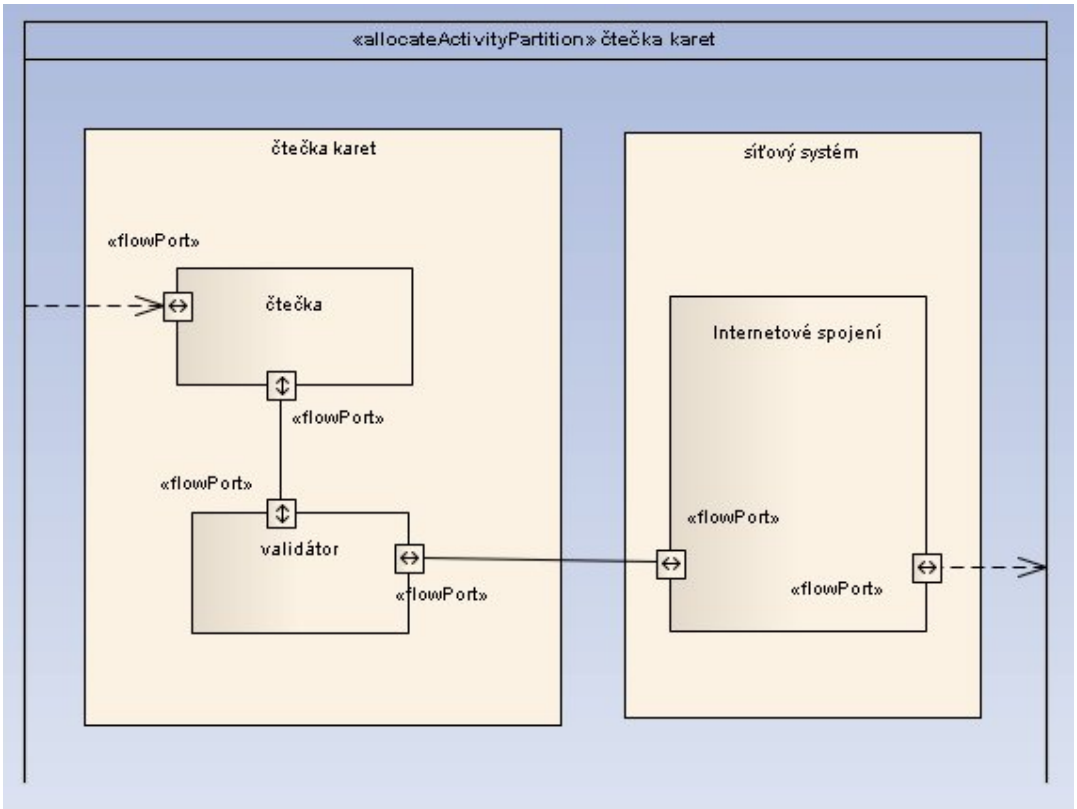
Obr. č. 11 – Ukázka blokového definičního diagramu

Daný diagram obsahuje aktéry (Zákazník, Zaměstnanec banky, Uživatel) a bloky (Banka).

Blok Internetové spojení obsahuje atributy Rychlost spojení a Typ. Jednotlivé prvky jsou propojeny pomocí asociace.

3.3.5 Vnitřní blokový diagram

Vnitřní blokový diagram popisuje vnitřní strukturu bloku propojených pomocí portů. Příklad použití je demonstrován na čtečce karet [9]:

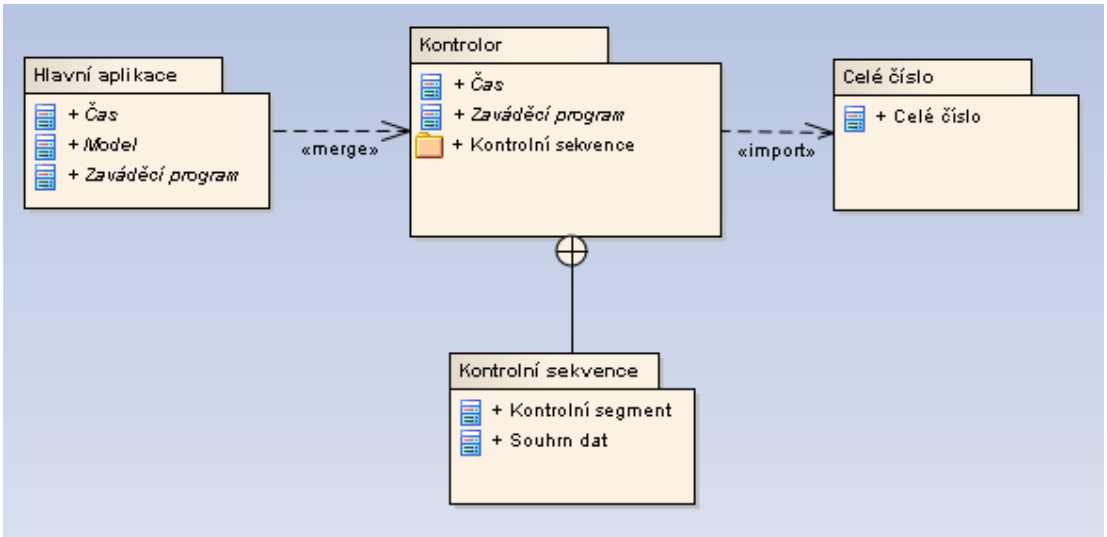


Obr. č. 12 – Příklad vnitřního blokového diagramu

Celý příklad je umístěn do úseku (Allocate partition). Jednotlivé časti jsou pak rozděleny do pomocí hranic (Boundary). Jednotlivé bloky (Part) jsou propojeny skrz port (Port Flow) pomocí asociačního spojení.

3.3.6 Balíčkový diagram

Balíčkové diagramy vykreslují organizaci modelů do balíčků a jejich vzájemnou závislost včetně importu a rozšíření balíčků. Také umožňují vizualizaci odpovídajících jmen funkcí. Příklad je zobrazen na následujícím obrázku.



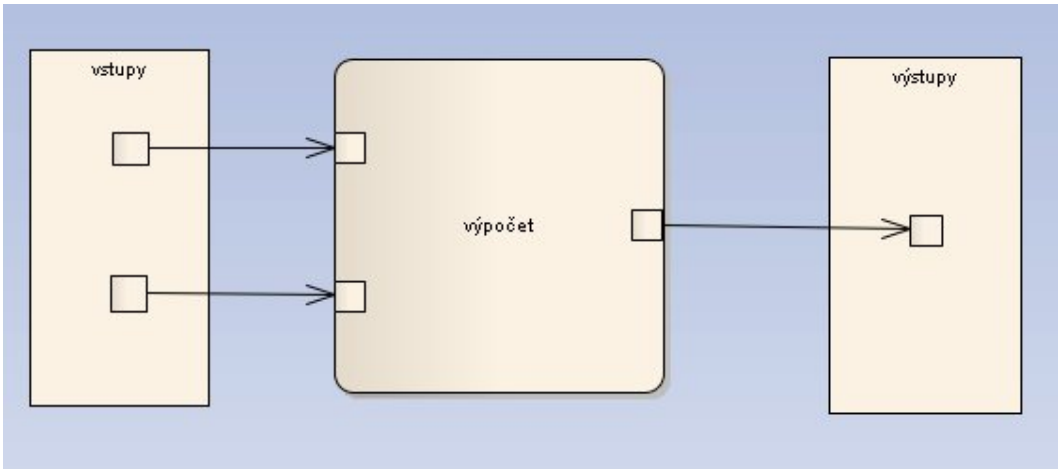
Obr. č. 13 – Příklad balíčkového diagramu

Vnořený spojovací člen mezí a kontrolní sekvencí a kontrolorem ukazuje co balíček kontrolní sekvence obsahuje. Obsahy balíčků se zobrazí kliknutím. Spojovací člen «import» naznačuje, že elementy uvnitř cílového balíčku jsou třídy integer. Spojovací člen «merge» naznačuje, že prvky balíčku kontrolor jsou importovány do balíčku hlavní aplikace.

3.3.7 Diagram parametrů

SysML umožňuje definovat parametrické vztahy mezi vlastnostmi bloků. Lze modelovat takové vztahy, které začleňují výkon nebo spolehlivost modelů v modelovém systému.

Diagramy parametrů se definují pomocí omezovacích bloků (constraint block), které popisují omezení v systémové struktuře včetně parametrů. Stereotyp «constraint» je specializací SysML stereotypu «block». Příklad je na obrázku:



Obr. č. 14 – Parametrický diagram

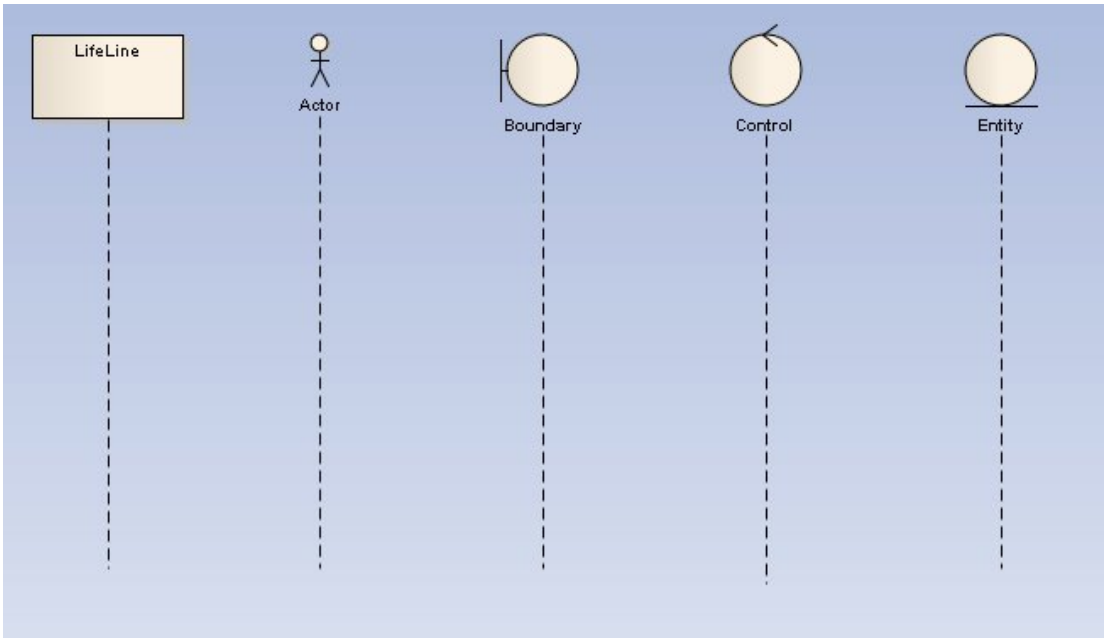
Na příkladu se dá například řešit výpočet dvou vstupních prvků.

3.3.8 Sekvenční diagram

Sekvenční diagram je forma interakčního diagramu ukazující objekty jako čáry života směřující směrem dolů s jejich interakcemi během časových okamžiků zobrazenými šipkami mezi objekty. Sekvenční diagramy jsou vhodné pro zobrazení komunikace mezi objekty, kde zprávy spouštějí tuto komunikaci. Sekvenční diagram nezobrazuje komplexní procedurální logiku.

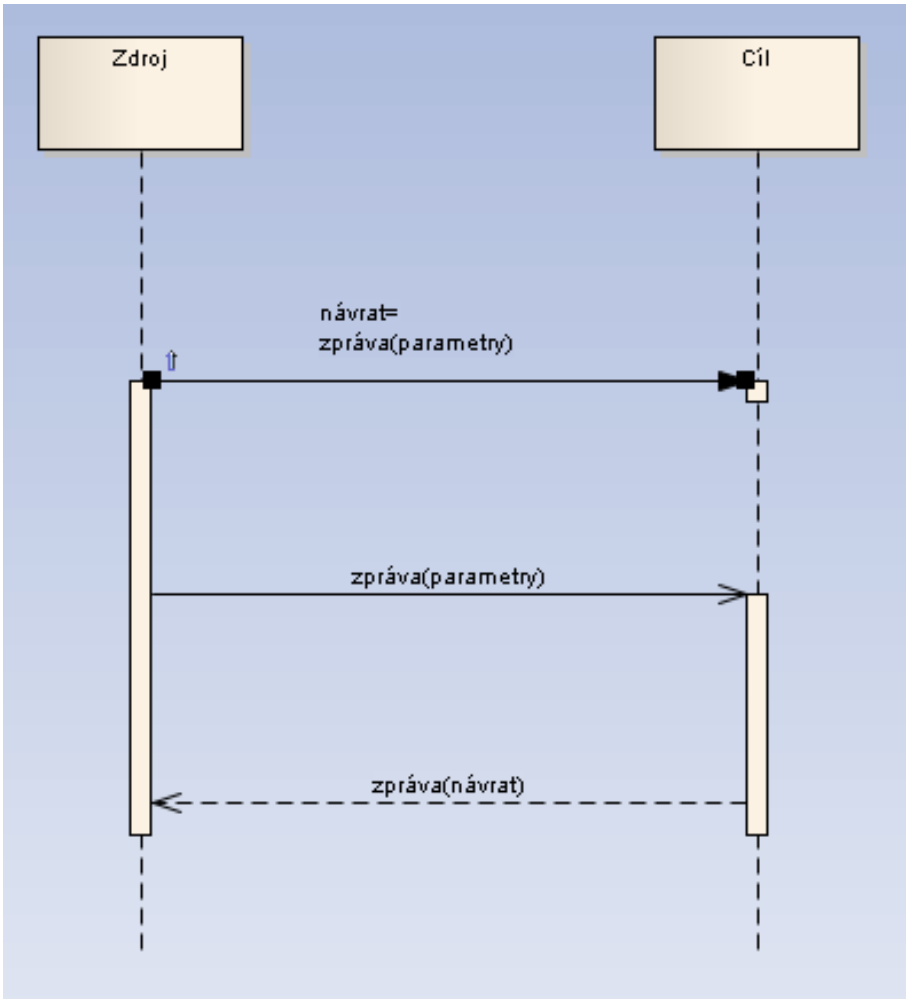
Čáry života reprezentují individuální účastníky v sekvenčním diagramu. Čára života je připojena většinou k obdélníku, což je jméno objektu. Pokud je jméno SELF, pak to naznačuje, že čára života reprezentuje klasifikátor vlastnící daný sekvenční diagram.

Příklady různých objektů a jejich čar života jsou na následujícím obrázku.



Obr. č. 15 – Příklady čar života (LifeLines)

Mezi čarami života se komunikuje pomocí zpráv (Message). Zprávy mohou být kompletní, ztracené nebo nalezené, dále také synchronizované a nesynchronizované a nakonec volání nebo signál. Příklad celého sekvenčního diagramu je zobrazen na dalším obrázku.

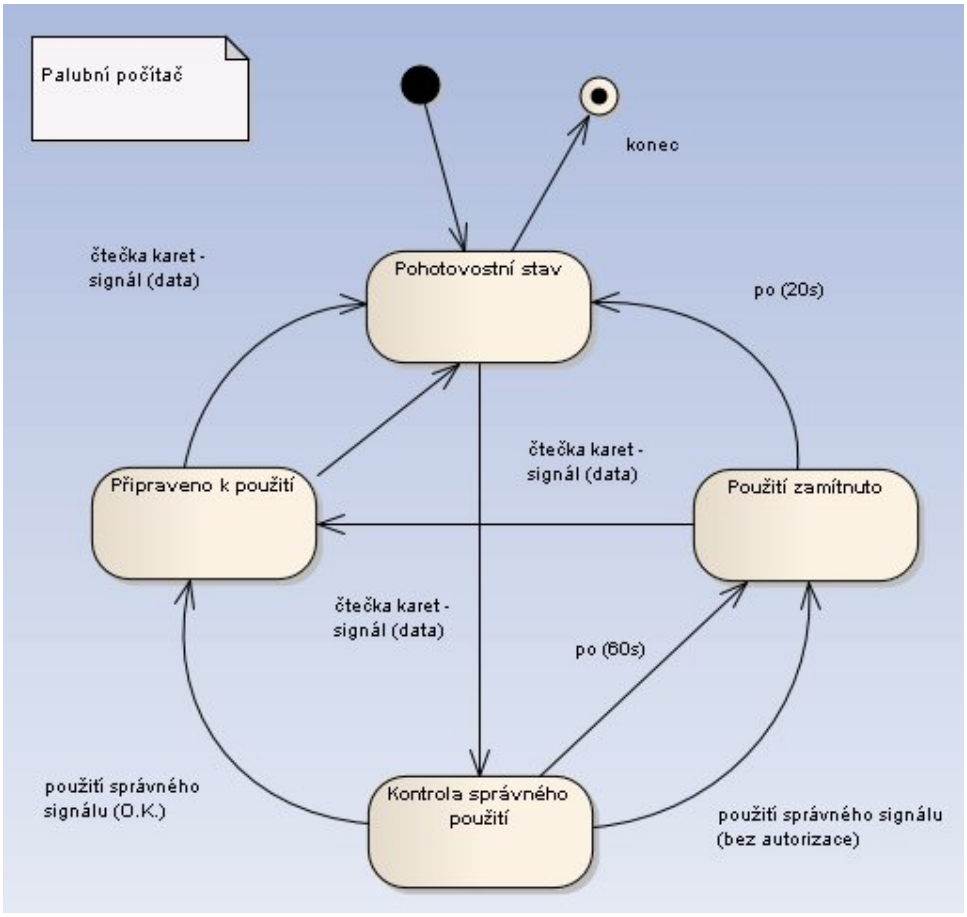


Obr. č. 16 – Sekvenční diagram

První zpráva je synchronní a je ukončena třetí zprávou, což je implicitní návrat. Druhá zpráva je asynchronní.

3.3.9 Stavový diagram

SysML verze stavového diagramu je stejná jako v UML. Stavové diagramy jsou nezávislé na příslušném odvětví systému, takže je lze modelovat bez softwarových specifických elementů. Každý systém má stavy, které lze popsat stavovým modelem. Příklad stavového diagramu je na obrázku.



Obr. č. 17 – Příklad stavového diagramu u palubního počítače

II. PRAKTICKÁ ČÁST

4 APLIKACE VYBRANÝCH METOD NA REÁLNÝ PROJEKT

Pro ukázku praktického využití jsem použil program Enterprise Architekt s rozšířením SysML. Pro vizualizaci jsem použil demonstrační obrázky s popisem průběhu tvorby projektu.

4.1 Praktický příklad – návrh části designu v autě

Podklady k příkladu jsem čerpal z internetových stránek tvůrců programu Enterprise Architect: http://www.sparxsystems.com/. Pro příklad jsem si vybral návrh části designu v autě pro zábavu (Rádio, CD, MP3, DVD zařízení) [6].

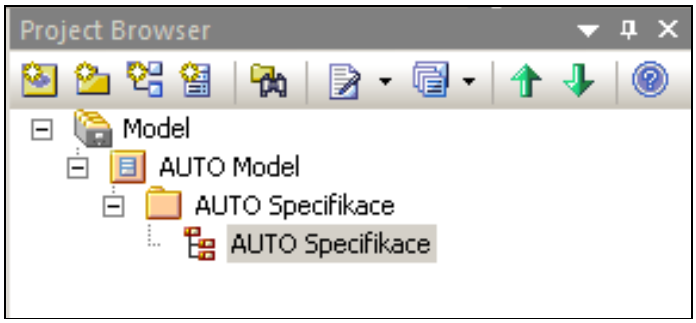
4.1.1 Tvorba projektu

Před tvorbou projektu jsem si nainstaloval program Enterprise Architect s nadstavbu SysML. Nyní se vytvořil nový projekt (create new project) a nazval ho design auta a zvolil technologii SysML.

4.1.2 Správa projektového prohlížeče

V projektovém prohlížeči jsem si nadefinoval balíčky (AUTO Model, AUTO Specifikace), které budou obsahovat specifikaci auta pomocí diagramu požadavků (Auto Specifikace).

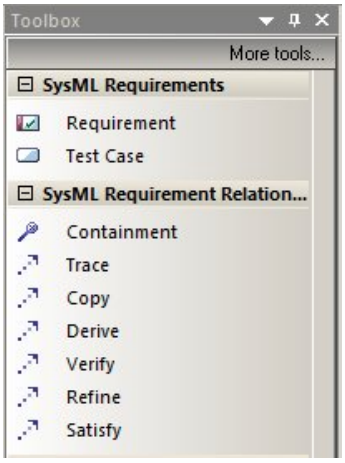
Pro názornost to vypadá takto:



Obr. č. 18 – Náhled na Project Browser

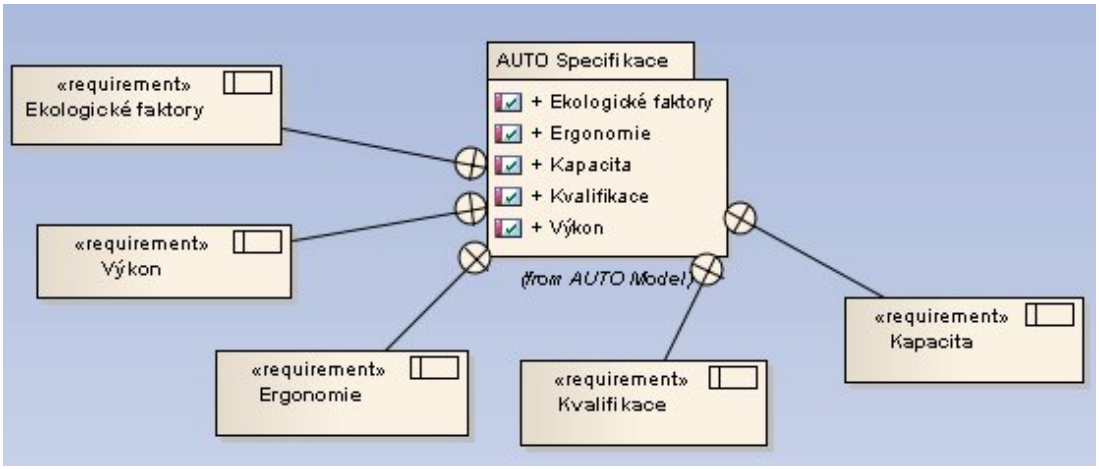
4.1.3 Diagram požadavků

Kliknutím na diagram AUTO Specifikace v projektovém prohlížeči se nám v Toolboxu zobrazí prvky diagramu požadavků:



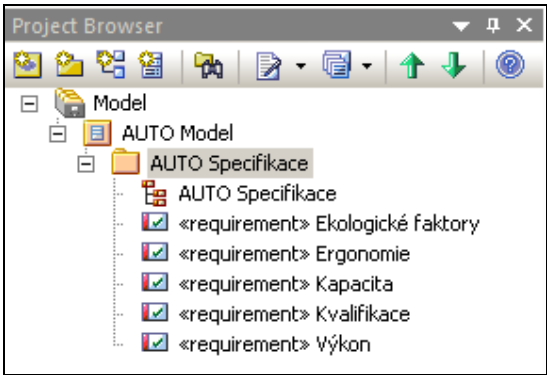
Obr. č. 19 – Toolbox

Náš přednastavený příklad specifikace vypadá takto:



Obr. č. 20 – Diagram požadavků (původní nastavení)

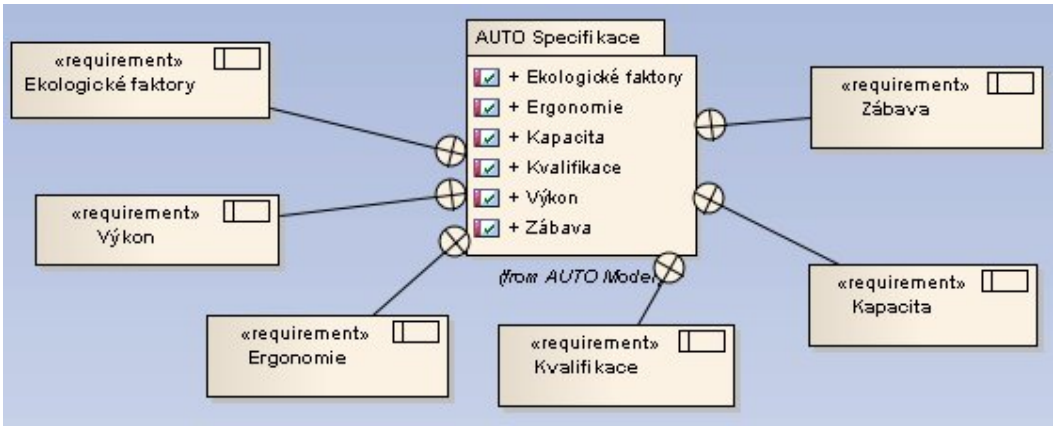
Prvky v projektovém prohlížeči takto:



Obr. č. 21 – Náhled v prohlížeči

Na obrázku lze vidět použití balíčkového diagramu, kde je pak zpětně vidět na diagramu použití vnořeného spojení (containment), kde jsou jednotlivé požadavky (Ergonomie, atd.) obsahem balíčku (AUTO Specifikace).

Do diagramu nyní vložíme nový požadavek (requirement), který bude obsahovat zábavné prvky v autě (Zábava - prvek se pak zobrazí i v projektovém prohlížeči). Diagram pak vypadá takto:



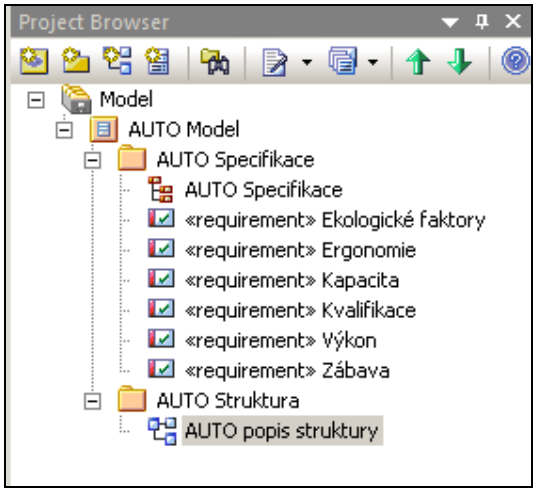
Obr. č. 22 – Požadavek Zábava

Nyní lze nadefinovat bloky a blokové kompozice, které definují strukturu auta.

4.1.4 Blokový definiční diagram

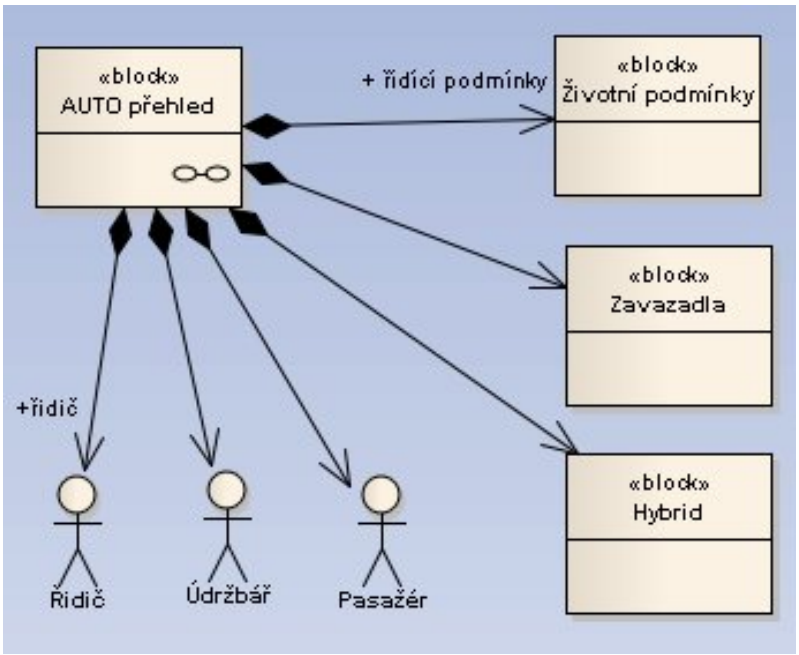
V projektovém prohlížeči si nadefinujeme balíček AUTO Struktura a do něj přidáme blokový definiční diagram AUTO popis struktury s již některými předdefinovanými bloky.

Náhled v projektovém prohlížeči vypadá pak takto:



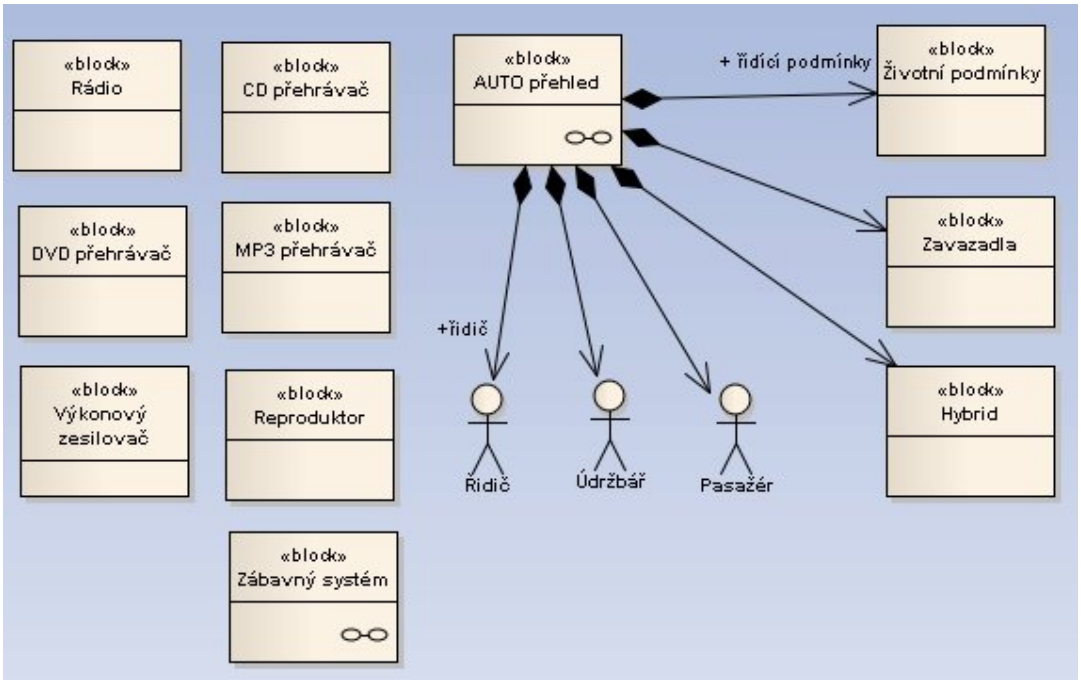
Obr. č. 23 – projektový náhled

A předdefinovaný blokový definiční diagram vypadá takto:



Obr. č. 24 – Blokový definiční diagram

Nyní si nadefinujeme bloky pro zábavu. Mezi tyto bloky patří: Rádio, CD přehrávač, DVD přehrávač, MP3 přehrávač, Výkonový zesilovač, Reproduktor a Zábavný systém. Diagram pak vypadá takto:



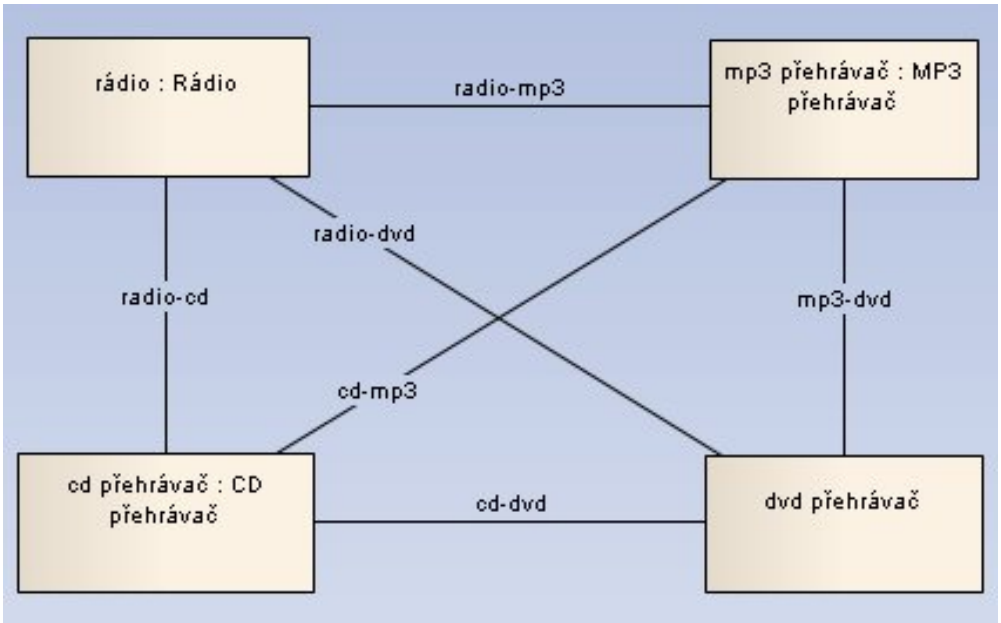
Obr. č. 25 – Blokový definiční diagram včetně zábavných bloků

Blok Zábavný systém se pak nastaví jako kompozitní (pravé tlačítko na blok, advanced, make it composite), což umožňuje definovat uvnitř toho bloku definovat vnitřní blokový diagram. Po kliknutí na tento diagram se dostaneme dovnitř.

4.1.5 Vnitřní blokový diagram

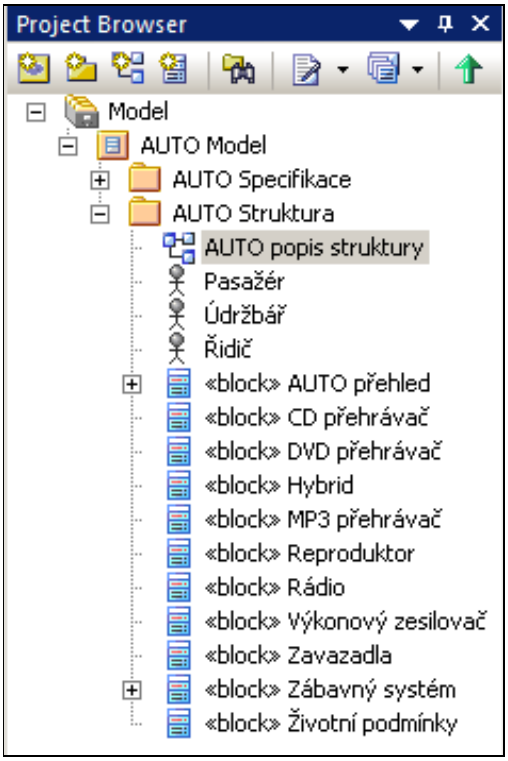
Nyní si tedy nadefinujeme vnitřní bloky zábavných prvků. Po kliknutí se v toolboxu automaticky načetly bloky náležící tomuto diagramu.

Vložíme do diagramu prvek Part, nazveme ho a nastavíme typ objektu (pravé tlačítko u myši, Advanced, Set Property Type) jako blok. Prvky jsou rádio, cd přehrávač, mp3 přehrávač a dvd přehrávač. Po propojení bloků výsledný diagram vypadá takto:



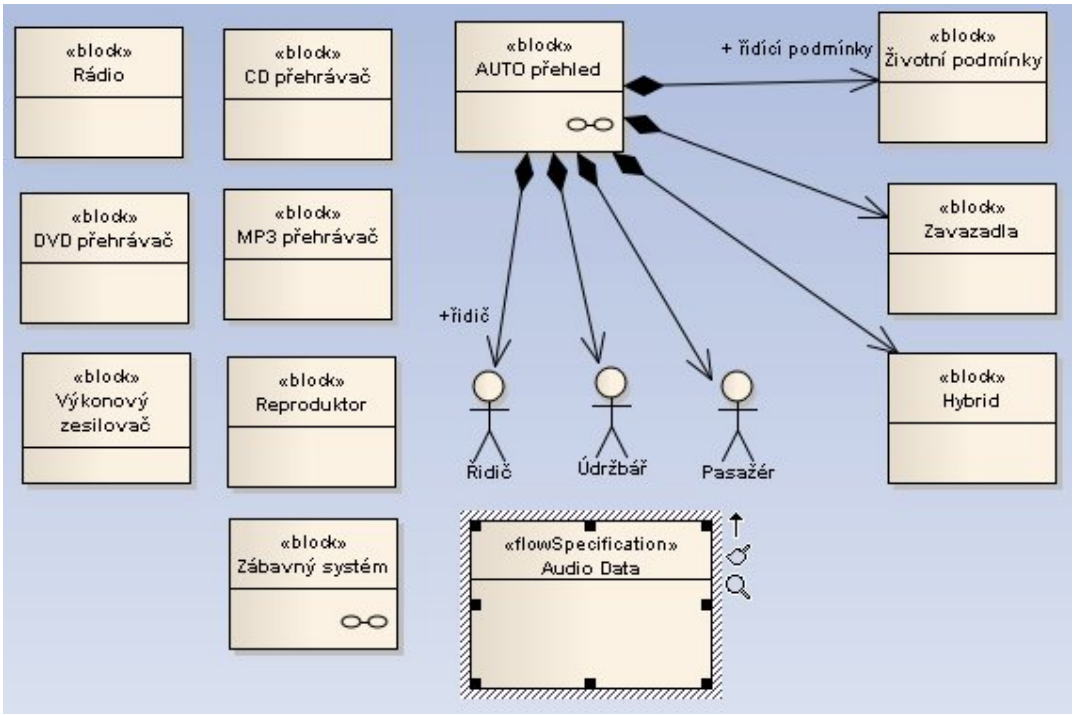
Obr. č. 26 – Vnitřní blokový diagram

Nyní lze nadefinovat audio signály, které jsou zprostředkovávány jako toky (flows). Nejdříve je však nutné dodefinovat bloky do modelu pro návrh audio systému. Vrátíme se tedy v projekt prohlížeči a načteme diagram AUTO popis struktury:



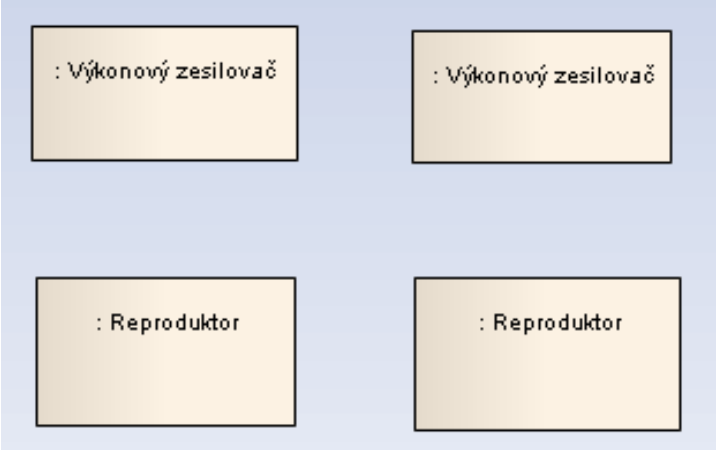
Obr. č. 27 – náhled prohlížeče

Do diagramu přidáme prvek Flow Specification (specifikace toku dat) pojmenovaný Audio data, diagram po doplnění vypadá takto:



Obr. č. 28 – Specifikace toku dat – Audio Data

Nyní máme bloky a specifikace toku dat nadefinovanou, poté vytvoříme specifické vlastnosti pro náš vnitřní systém. Vrátíme se tedy zpět do diagramu Zábavný systém. Do diagramu přidáme 4 bloky přetažením z projektového prohlížeče, kde se objeví tabulky s výběrem typu elementu, zvolíme as Property of Element (stejně jako vlastnost prvku). Přidáme 2x zesilovač a 2x reproduktor. Diagram pak vypadá takto:

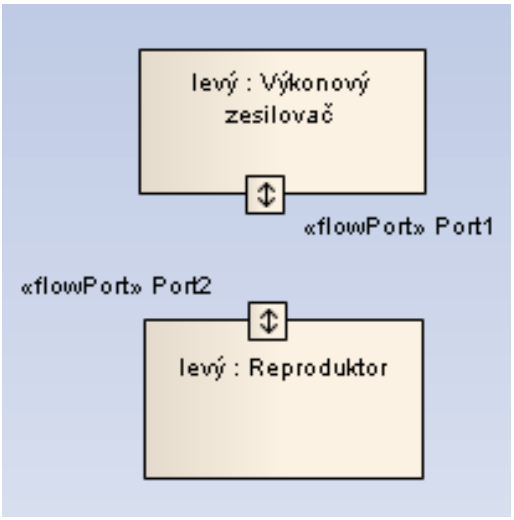


Obr. č. 29 – Přesunuté bloky z prohlížeče

Kliknutím na každý blok a vyplněním name (jména) označíme umístění zařízení (levý, pravý):

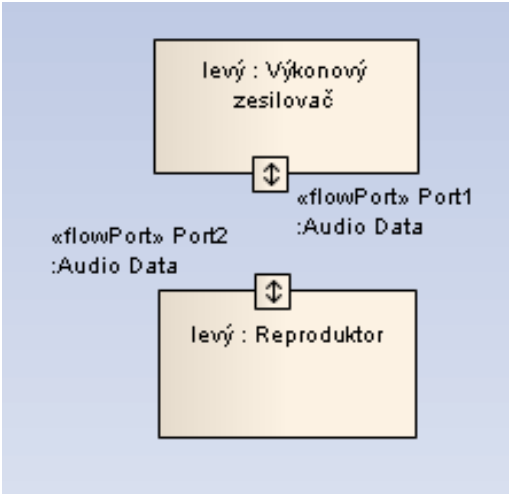
Obr. č. 30 – Označení stran bloků

Poté přidáme porty toku dat k zesilovači a reproduktoru a specifikujeme charakteristiky audio dat procházející mezi těmito prvky. Přidáme tedy k levému Výkonovému zesilovači a levému reproduktoru Porty – Port (flow) a nazveme je Port1 a Port2:



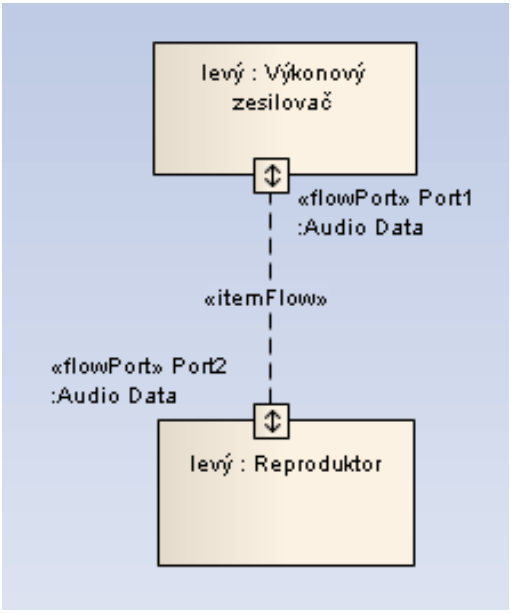
Obr. č. 31 – Vložení portů

Pak si nastavíme typ portu (Advanced, Set Property Type) a vybereme specifikaci «flowSpecification»Audio Data:



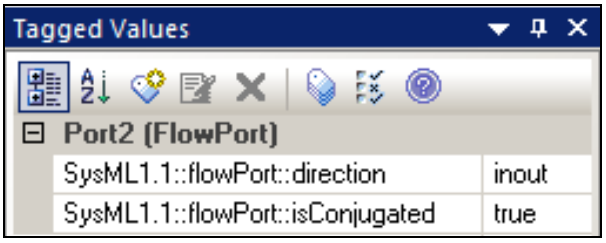
Obr. č. 32 – Nastavení portů

Potom oba porty propojíme přetažením portů k sobě a vybereme ItemFlow:



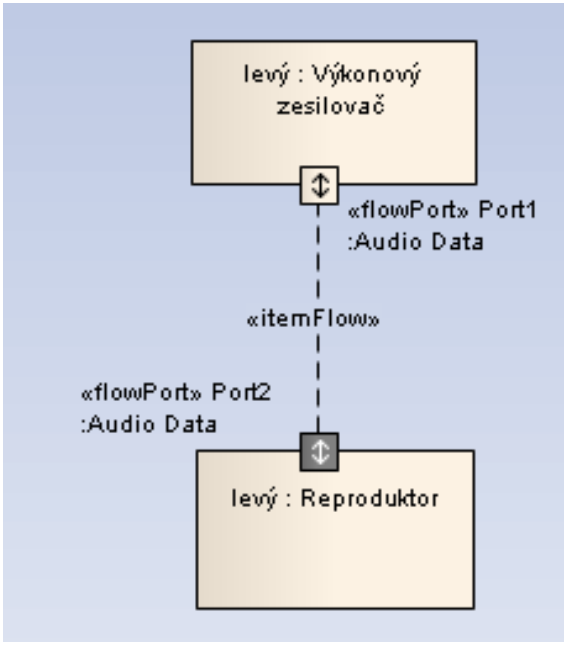
Obr. č. 33 – Propojení portů

Protože jsou prvky párové musí se u druhého portu nastavit, že je opačný. Klikneme pravým tlačítkem u myši na spodní port, vpravo nám vyskočí tabulka Tagged Values, kde nastavíme u flowPort:isConjugated místo false dame true:



Obr. č. 34 – Změna portu

Tím se změní barva portu. Výsledek pak vypadá po změně takto:

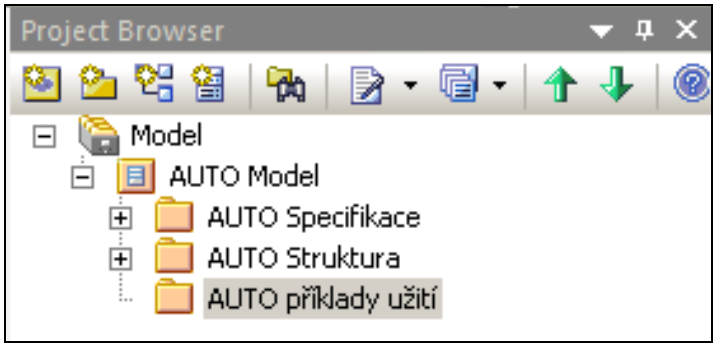


Obr. č. 35 – Výsledný pohled na propojení

Nyní je ukončen návrh struktury Zábavného systému, přesuneme se k definici charakteristiky chování.

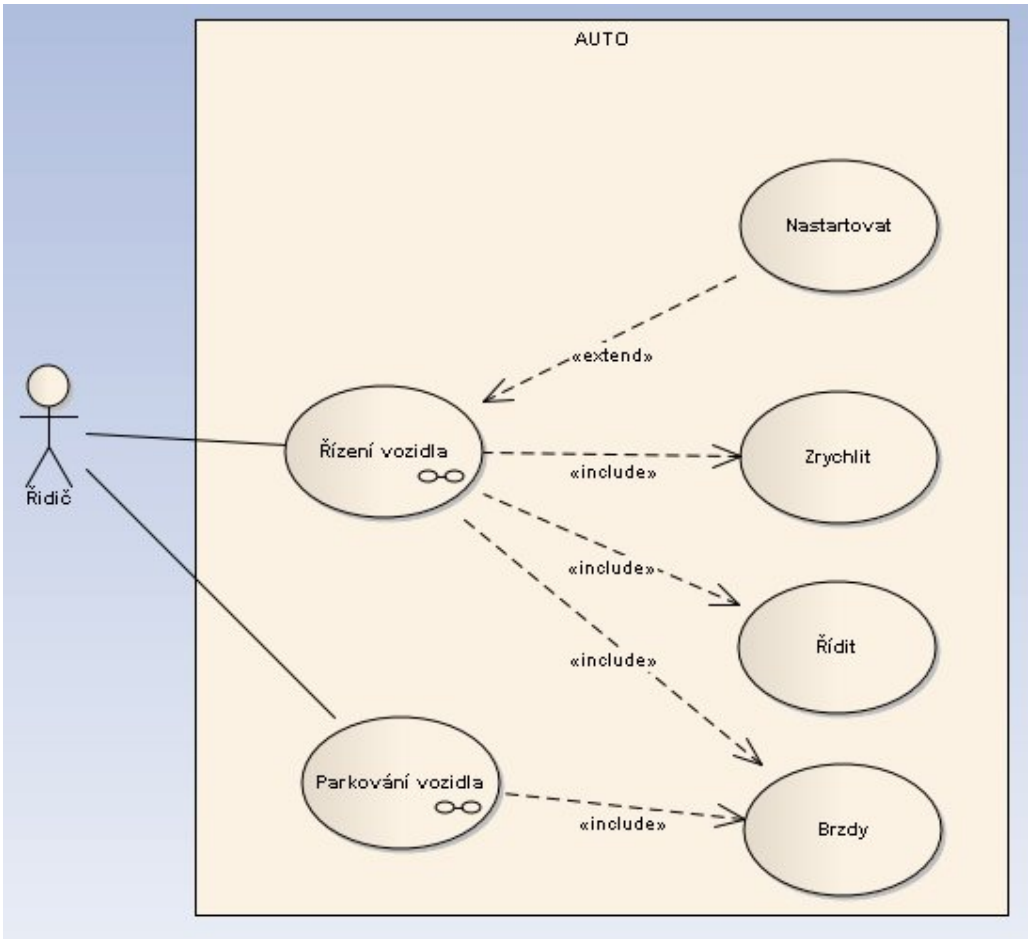
4.1.6 Případy užití

Pro příklady užití si nadefinujeme nový balíček AUTO příklady užití v projekt prohlížeči:



Obr. č. 36 – Náhled do projektového prohlížeče

Nyní si opět vložíme do balíčku pro lepší názornost již existující diagram s prvky týkajících se našeho automobilu a dále budeme řešit náš Zábavný systém:

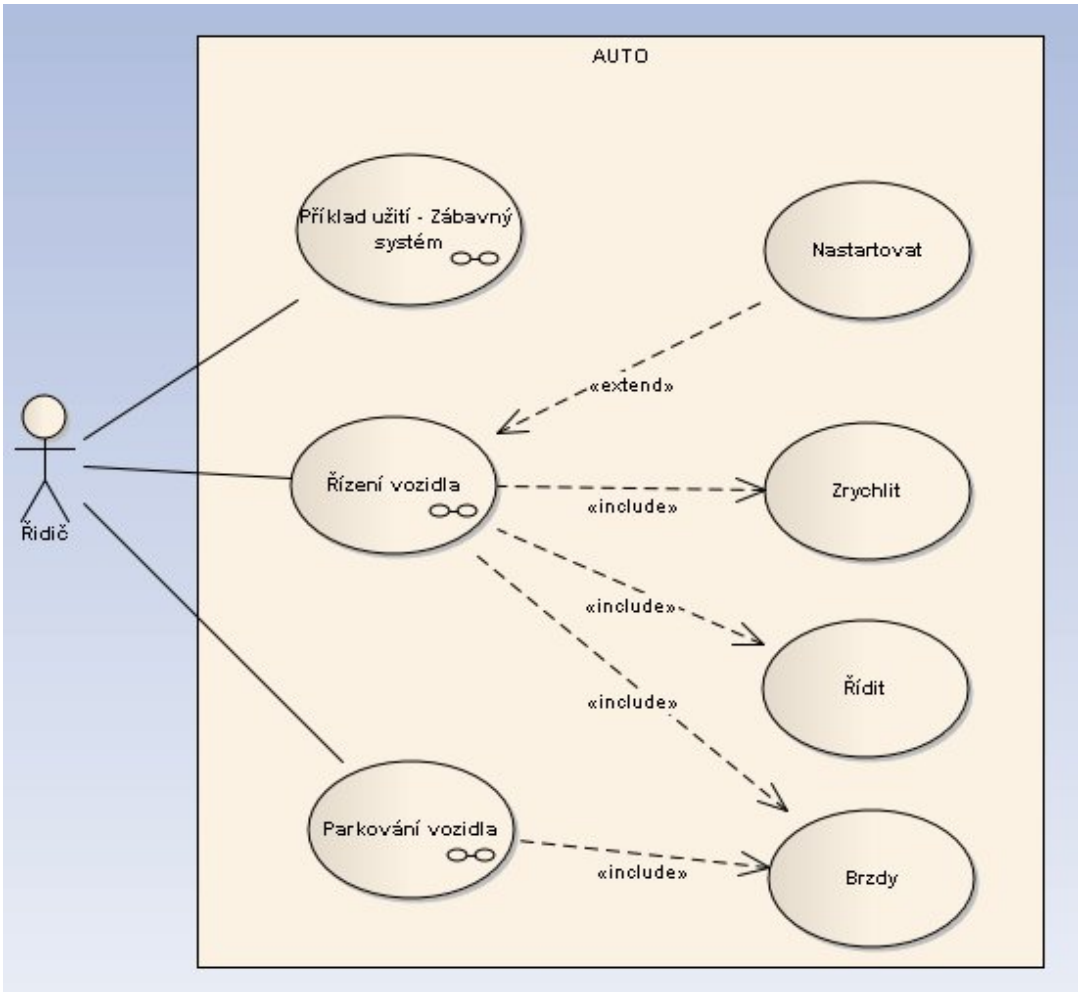


Obr. č. 37 - Předdefinovaný diagram příkladu užití

Diagram obsahuje hranici (Boundary), kde se jsou umisťovány příklady užití. Příklady užití (tvar elipsy) jsou mají mezi sebou vztahy naznačené propojením. Nakonec poslední prvek

je aktér (Actor), který má taktéž vztahy s příklady užití uvnitř hranice. Vztah «include» naznačuje, že označený příklad užití má stejné chování jako druhý prvek. Vztah «extend» naznačuje rozšíření chování jinému prvku. Poslední použité spojení je asociace (association, plná čára), a to je všeobecný vztah pro spojení. Příklady užití Řízení vozidla a Parkování vozidla označíme jako kompozitní (prvek je složený – uvnitř obsahuje další prvky; pravé tlačítko u myši, Advanced, Make it composite).

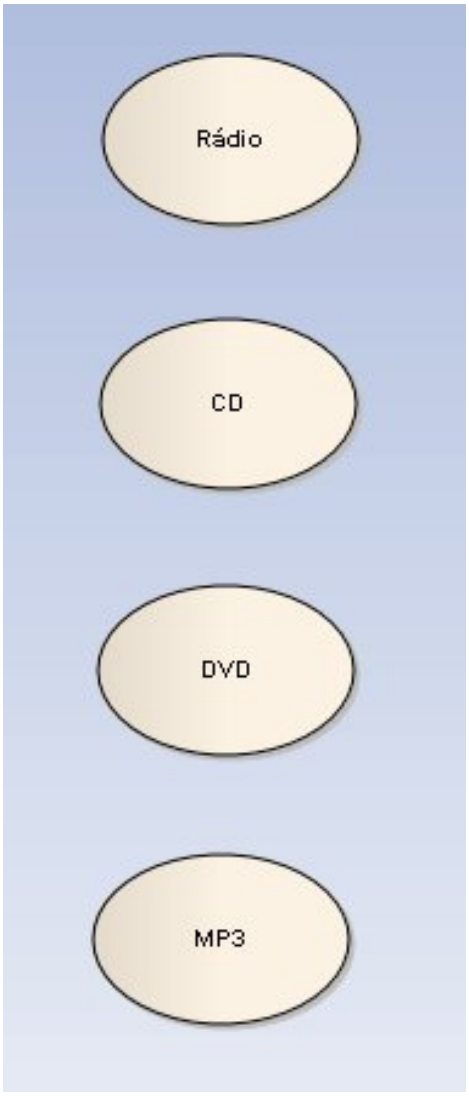
Nyní do tohoto existujícího diagramu přidáme příklad užití Zábavného systému a opět ho nastavíme jako kompozitní:



Obr. č. 38 – Příklad užití pro Zábavný systém

Nyní si nastavíme příklad užití pro náš Zábavný systém, dovnitř diagram se dostaneme kliknutím na příklad užití.

Nyní si nadefinujeme jednotlivé příklady užití pro Zábavný systém – Rádio, CD, DVD a MP3:

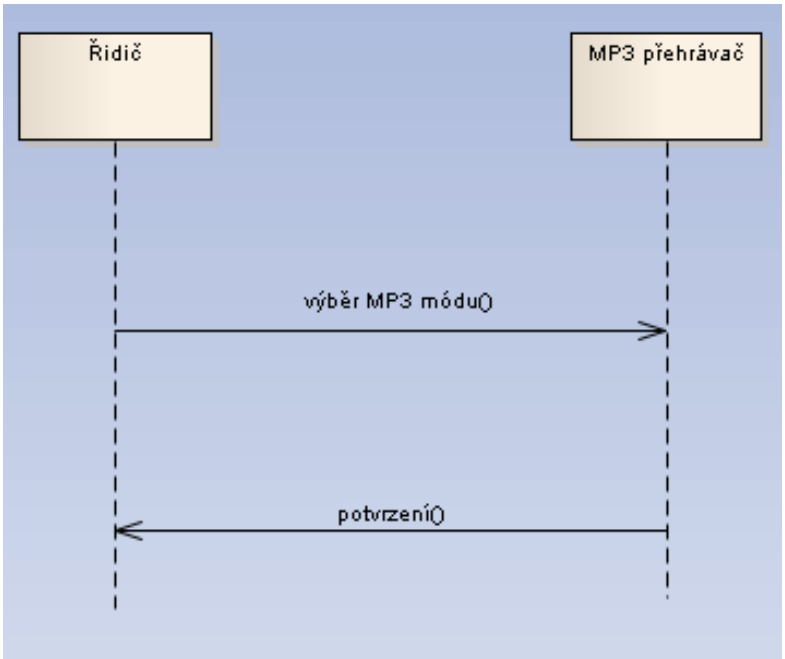


Obr. č. 39 – Příklady užití

Nyní si nadefinujeme blokové interakce pro příklad užití MP3. Opět nastavíme prvek jako kompozitní a podíváme se klikem dovnitř.

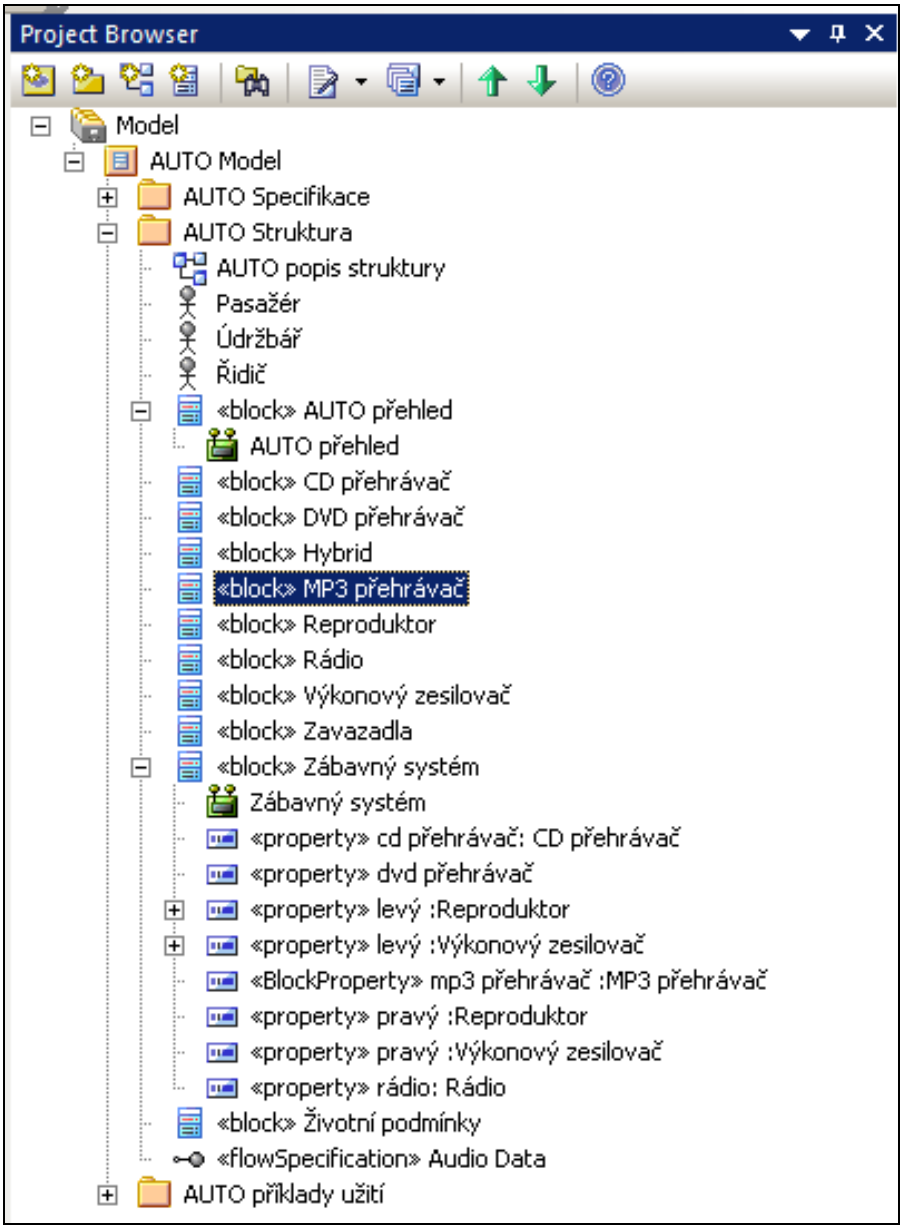
4.1.7 Interakční diagram

Nadefinujeme si dvě sekvence Řidiče a MP3 přehrávač a spojíme zprávami (Message). Řidič si vybere mód přehrávání a přehrávač pošle zprávu zpět:



Obr. č. 40 – Interakční diagram

Nyní si nadefinujeme vlastnosti chování MP3 přehrávače. Nalezneme si blok MP3 přehrávače v projektovém prohlížeči a v balíčku AUTO struktura:

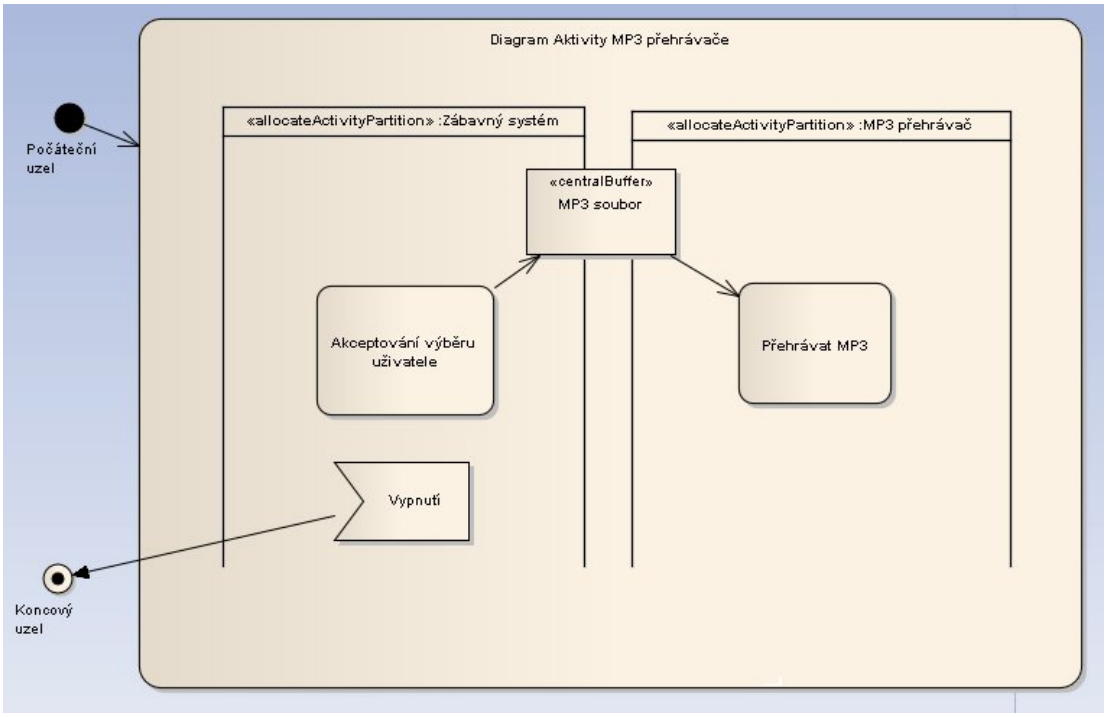


Obr. č. 41 – Výběr bloku z projektového prohlížeče

Poté přidám blok do diagramu aktivity (pravým klikem vyberuAdd, Add diagram, SysML a vyberu Activity).

4.1.8 Diagram aktivity

Tím jsem si vybral, že budu vytvářet diagram aktivity pro blokový prvek MP3. Diagram pak vypadá takto:



Obr. č. 42 – Diagram aktivity pro MP3 přehrávač

Alokační přehrazení (allocate Aktivity Partition) umožňuje, aby se mohly prvky diagramu propojit. V diagramu jsou dvě aktivity, obě propojeny zároveň k Zábavnému systému i k MP3 přehrávači. Diagram je ohraničen prvkem Aktivity (Diagram aktivity MP3 přehrávače). Samotná aktivita začíná počáteční uzlem a končí propojením akčního prvku (action) Vypnutí s koncovým uzlem. Pro propojení obou přehrazení je zvolen prvek Central Buffer Node (centrální uzel), který je objektovým uzlem schopným spravovat toky z různých zdrojů a k různým cílům.

4.2 Další praktický příklad – Automaticky řízený stěrač

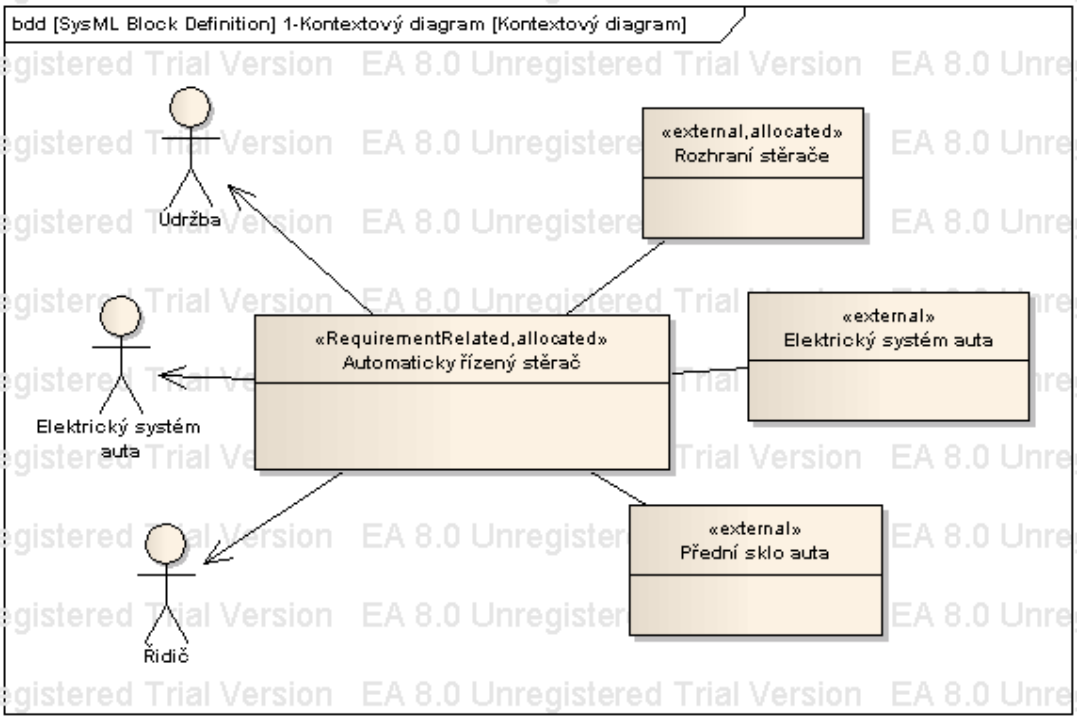
Automaticky řízený stěrač neboli Rain sensing wiper (RSW) je optické zařízení automaticky reagující na stav čistoty předního okna v automobilu. Pokud například prší, tak zařízení zapne stěrače a začnou se samy stírat okna. Zařízení se skládá ze 3 částí:

- Programu, který ovládá chování stěrače.
- Elektronické kontrolní jednotky, která spoušťí program
- Senzoru umístěného na vnitřní straně předního skla auta, který snímá kapky dopadající na přední sklo.

Tento příklad jsem si vybral na stránkách ibm.com, jeho autorem je Laurent Balmelli, PhD. Příklad vytvářel v oficiálním IBM softwaru [7].

4.2.1 Kontextový diagram

Kontextový diagram je pouze informačním diagramem a naznačuje hranice, co patří a nepatří do daného systému. Kontextový diagram v SysML neexistuje, proto se využívá blokového definičního diagramu. Pro náš vybraný systém vypadá kontextový diagram takto:



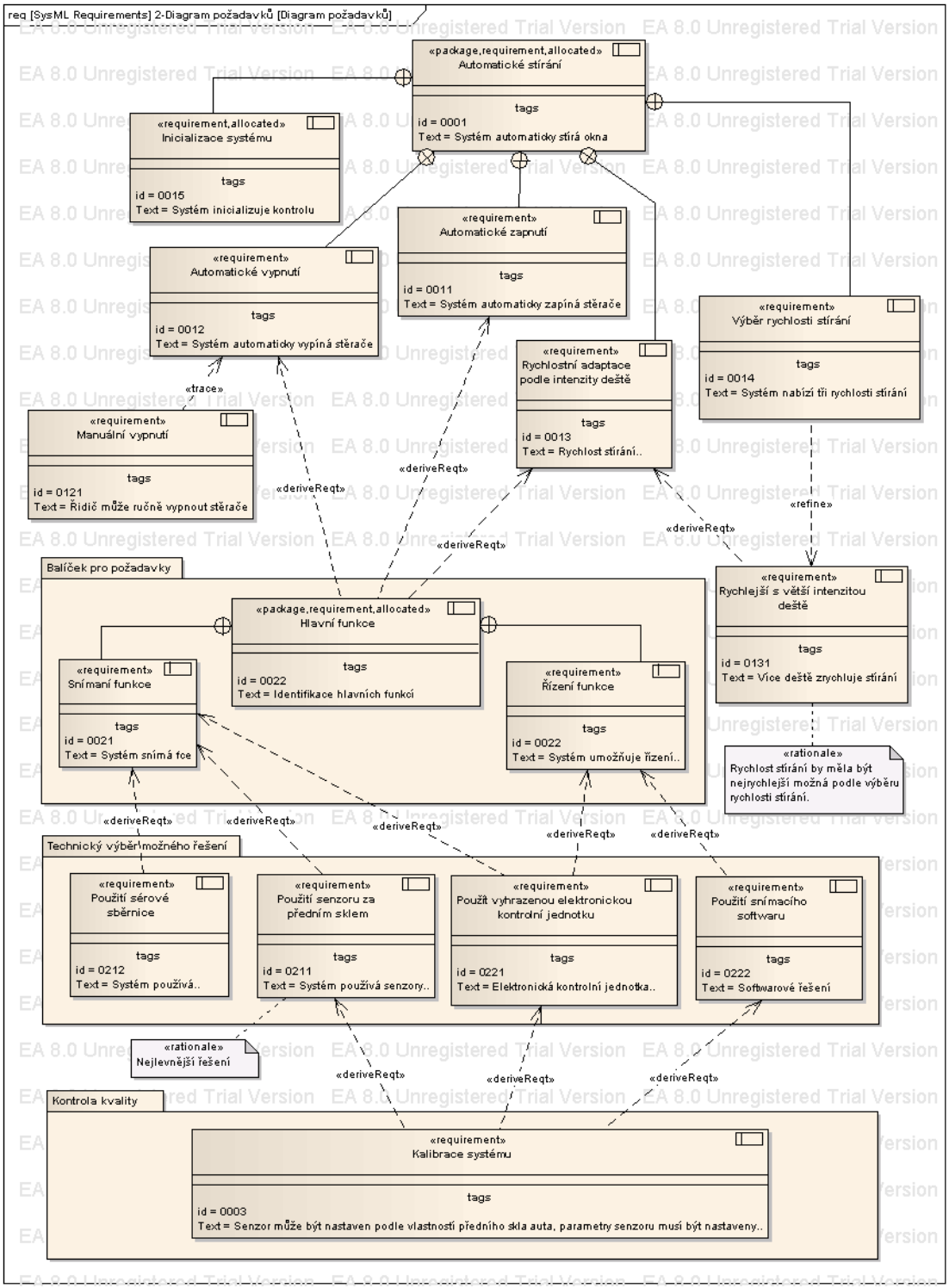
Obr. č. 43 – Kontextový diagram

Kontextový diagram umožňuje celkový náhled na systém. Diagram ukazuje tři aktéry v systému: údržbu (pro možnosti oprav), elektrický systém auta (aktivace systému v autě) a řidiče. Kromě toho jsou zde tři systémy: rozhraní stěrače, elektrický systém auta a přední sklo auta. Typ external v tomto případě naznačuje kvalifikaci jako externí komponent a elektrický systém auta poskytuje také elektrický pohon pro automaticky řízené stěrače.

Enterprise Architect neumožňuje spojení některých typů objektů, proto pro náš příklad je vynechán typ block.

4.2.2 Diagram požadavků

Další diagram dává náhled na požadavky systému – diagram požadavků:

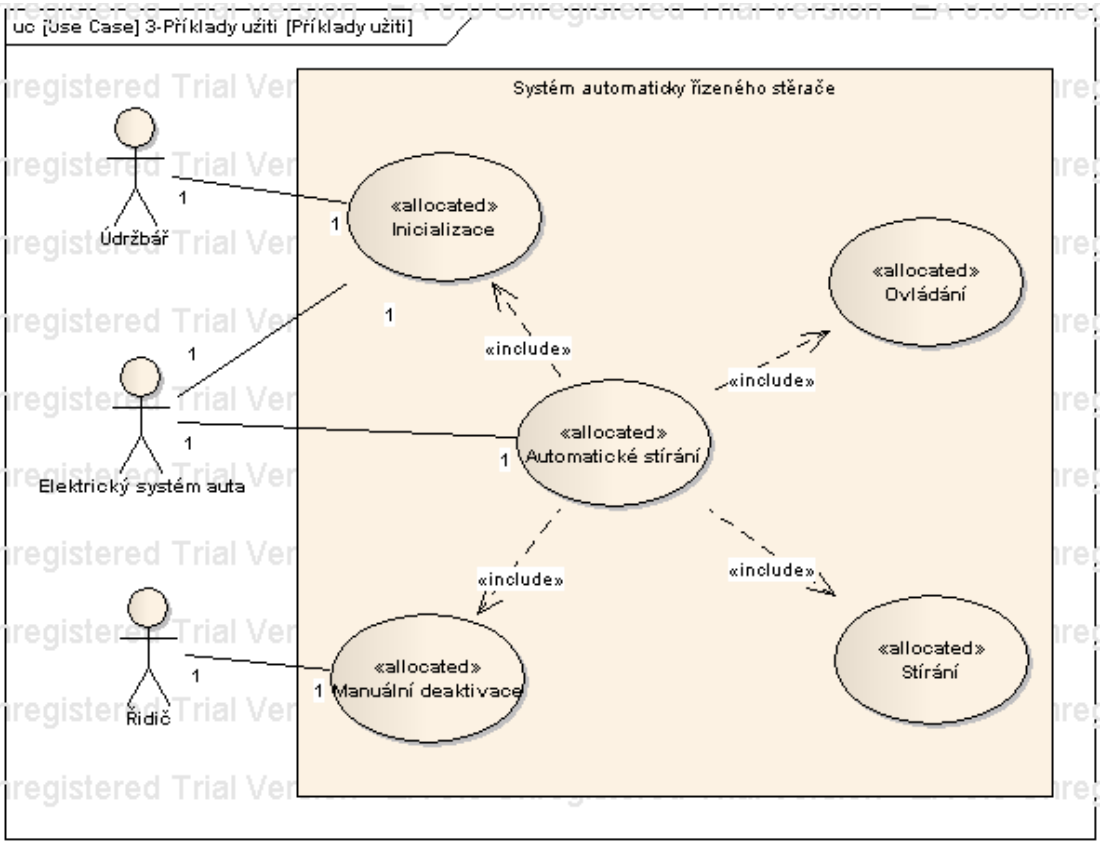


Obr. č. 44 – Diagram požadavků pro automaticky řízený stěrač

Hlavním požadavkem je Automatické stírání a ostatní požadavky jsou vložené pomocí spojení containment a stávají se pod-požadavkem. Během analýzy se ostatní nové požadavky vznikají derivací. Spojení se nazývá deriveReqt. Dalším prvkem v diagramu je Rationale, což je komentovaný popis nějaké situace v diagramu. V našem případě třeba výběr technického řešení. Diagram dále obsahuje balíčky, které mohou obsahovat další požadavky. V našem případě například technický výběr možného řešení, kde jsou nastíněny možnosti řešení pro náš systém nebo kontrola kvality, která určuje stav a kontroluje systém. Dalším použitým vztahem je Refine, který upřesňuje určitým způsobem předchozí požadavek. Trace vztah naznačuje pak sledování podobnosti požadavku, pro náš případ automatické a manuální vypnutí.

4.2.3 Diagram případu užití

Následující diagram zobrazuje interakci externích aktérů s některými případy užití (elipsy) patřícím k našemu systému:



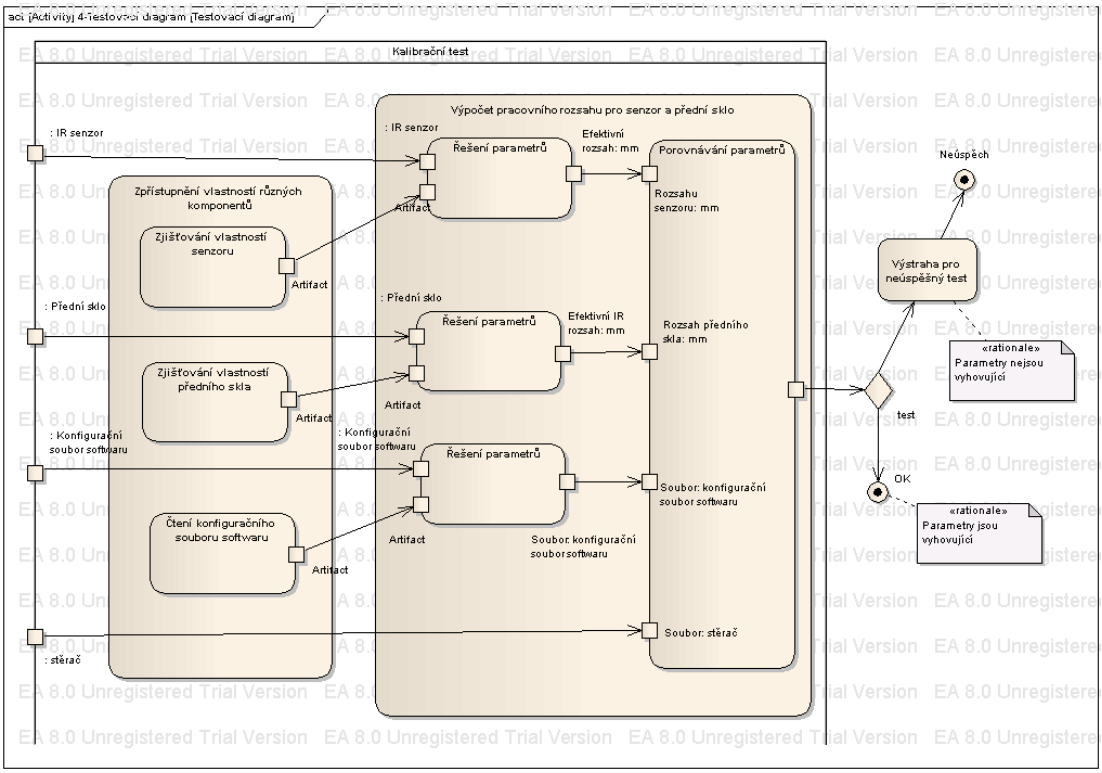
Obr. č. 45 – Příklady užití

Tento případ užití reprezentuje tři aktéry a jejich spojení. Centrálním případem užití je Automatické stírání a z něj vycházejí ostatní případy užití. Hierarchie vztahů je modelována pomocí vztahu include, což je obsažení.

Všechny příklady užití jsou umístěny do hranice (boundary) a jejich spojení s aktéry je zajištěno pomocí vztahu asociace. Pro náš případ lze pro asociaci nastavit i multiplicitu (počet počátečních a koncových prvků ve spojení). Ve vlastnostech spojení nastavíme ve zdroji (source role) a cíli (target role) naši požadovanou hodnotu. My si nastavíme multiplicitu pro zdroj i cíl hodnotu 1. Pro naše případy užití si nastavíme stereotyp allocated.

4.2.4 Testovací případy

Kromě možnosti příkladů užití má SysML i možnost testování případů, které lze spojovat s příslušnými požadavky a příklady užití. Testovací případy mohou být operací nebo modelem chování. Uvedený testovací případ ověřuje požadavek kalibrace systému:



Obr. č. 46 – Testovací případ požadavku kalibrace systému

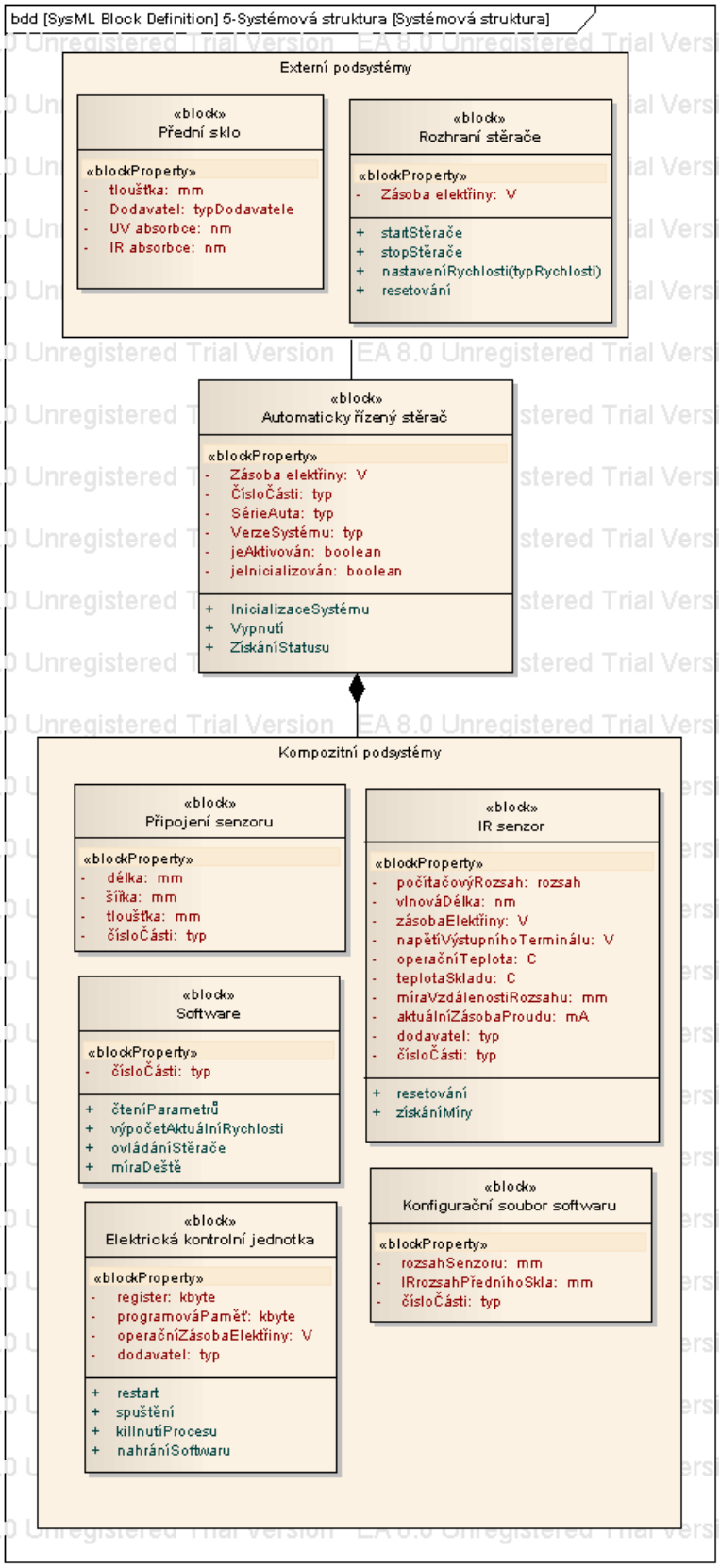
Postup při testování:

- Nejdříve se získají charakteristiky různých komponent (senzor, přední sklo, konfigurační soubor).
- Poté se získané charakteristiky použijí při výpočtu operačních rozsahů senzoru a předního skla pro jejich kompatibilitu. Jestliže senzor a přední sklo vyhovují testu, pak je test úspěšný, v opačném případě je spuštěna výstraha. Akční prvky v diagramu aktivity slouží pro zobrazení každého kroku v testování.

Celý test je v diagramu aktivity ohraničen sekcí (Partition). Uvnitř sekce jsou prvky Aktivity (Zpřístupnění vlastností různých komponentů), kde jsou umístěny jednotlivé akční členy (Action) s objektovými uzly (Object node). Jako rozhodovací prvek je Decision a pro ukončení jsou zde koncové uzly (Final).

4.2.5 Systémová struktura

Pro definici struktury systému automaticky řízeného stěrače se použije opět blokový definiční diagram. Struktura pro tento systém vypadá takto:



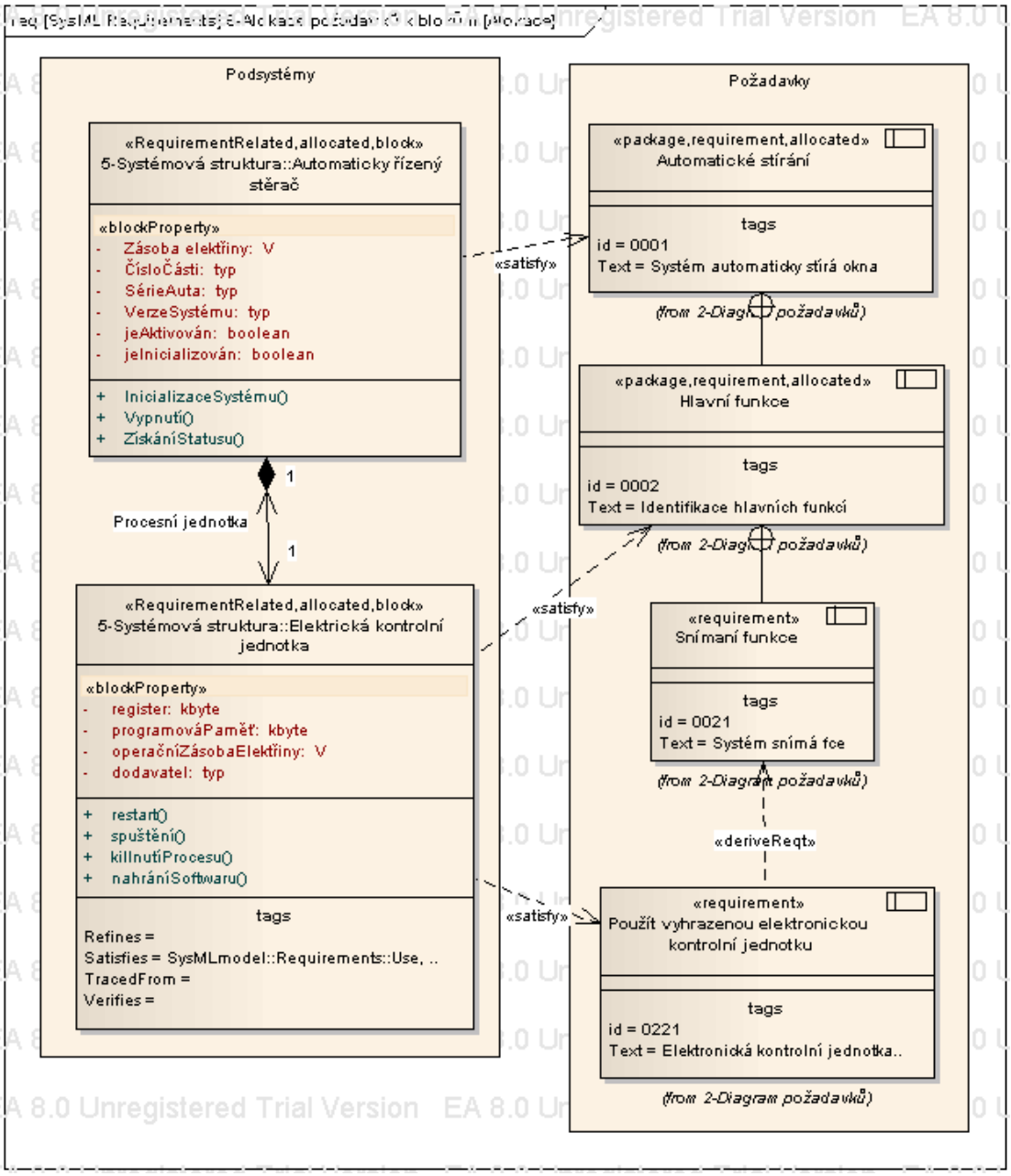
Obr. č. 47 – Blokový definiční diagram

Diagram zobrazuje asociační vztah mezi externími subsystémy blokem Automaticky řízený stěrač a mezi blokem a kompozitními subsystémy. Hlavními komponentami jsou: rozhraní stěrače, elektrická kontrolní jednotka, senzor a prvek přední sklo. Rozhraní i přední sklo mohou existovat i mimo systém, v SysML se nazývají referenčními vlastnostmi.

Jednotlivé bloky v diagramu mají své atributy (červeně vyznačené) a operace (zeleně vyznačené). Pro jejich zobrazení v EA je nutné nastavit jejich viditelnost (Feature Visibility, zaškrtnout – Show Attributes, Show Operations, Atribute Visibility All).

4.2.6 Alokace požadavků k blokům

Jeden z nejdůležitějších důsledků tvorby požadavků jako elementu je, že umožňuje specifikovat, které komponenty v systému splňují danou sadu požadavků. Říká se tomu alokační proces. Příklad alokace požadavků je na následujícím obrázku:



Obr. č. 48 - Příklad alokace požadavků

Diagram vzniká přenesením již existujících elementů do nového diagramu. Na levé straně jsou některé bloky a na pravé jsou požadavky. Dalším způsobem jak reprezentovat alokaci je použití úseku Requirement related. Tento úsek zobrazuje stav sady derivovaných vlastností příslušných k požadavkům.

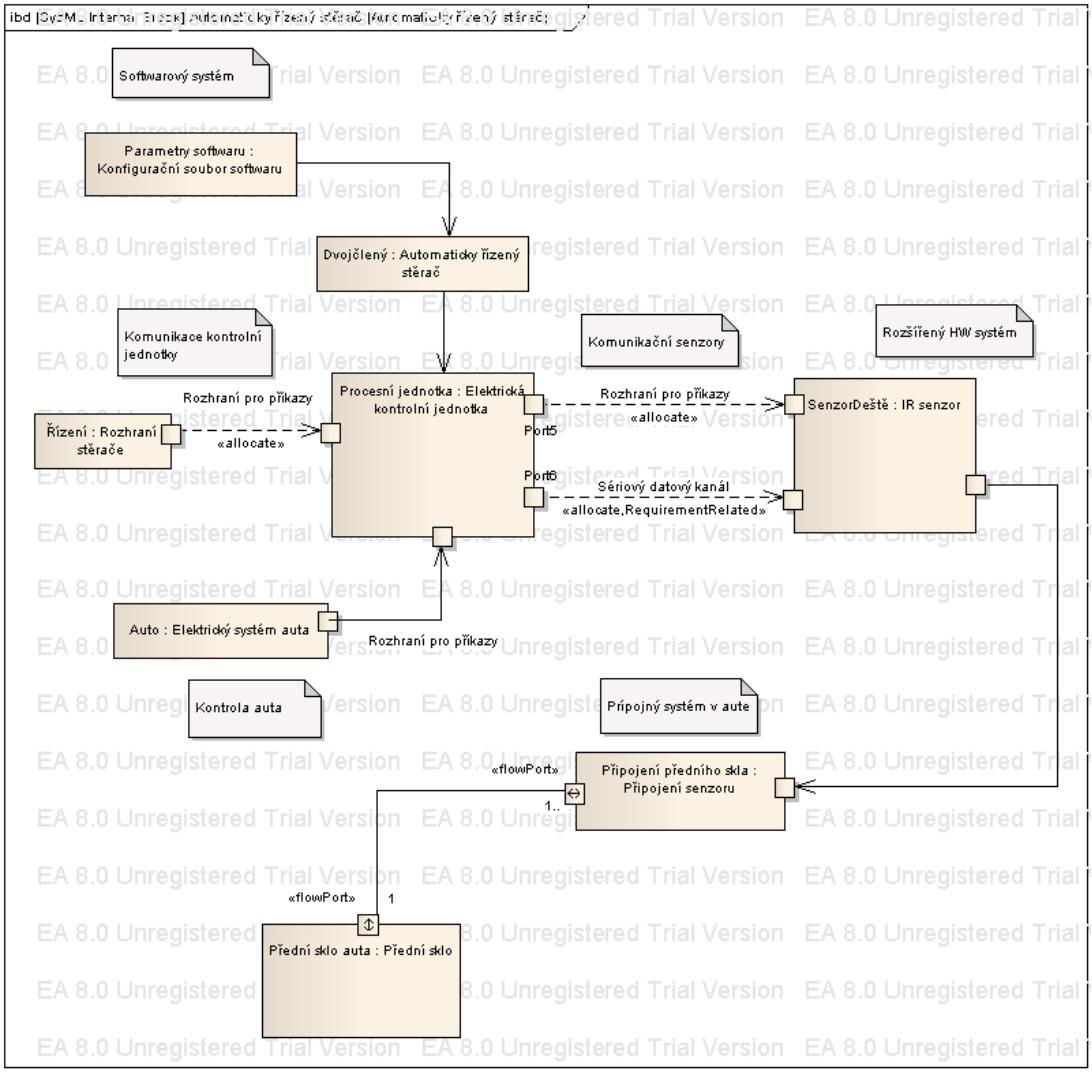
4.2.7 Vnitřní blokový diagram

Nejdůležitějším aspektem vnitřního blokového diagramu je možnost zdokonalovat definici vztahů použitých bloků skrz jejich porty. Porty jsou části používané pro spojení a jsou rozdělené podle typu rozhraní.

V SysML existují dva typy portů:

- Standardní – řeší požadavky a realizaci služeb s jinými bloky (v EA Port).
- Flow – dovolují blokům vyměňovat toky informací nebo materiálu (v EA Port Flow).

Vnitřní blokový diagram je zobrazen na následujícím obrázku (Abychom mohli tvořit vnitřní blokový diagram je nutné daný prvek nastavit jako kompozitní):



Obr. č. 49 – Vnitřní blokový diagram

Jednotlivé bloky v diagramy mají různé porty podle druhu propojení. Pro poslední propojení jsou použity flow porty. Jednotlivé části jsou doplněny o poznámky.

Celý příklad je mnohem obsáhlejší a pro potřeby diplomové práce jsem použil jen část. V programu Enterprise Architect lze namodelovat různé kombinace prvků a tak mohou vznikat různé variace diagramů.

ZÁVĚR

Zpracováním dostupného materiálu vznikl popis softwarového inženýrství, popis jazyka SysML a praktické příklady užití tohoto jazyka. Tento materiál lze použít pro výukové i praktické účely.

Diplomová práce obsahuje krátké seznámení s jazykem UML. Další část softwarové inženýrství mapuje základní pohled na tuto vědeckou disciplínu. SysML část popisuje historii jazyka a používané diagramy s jejich krátkým popisem. Poslední část popisuje praktické použití jazyka SysML na dvou příkladech. Příklady popisují použitelnost jednotlivých diagramů v různých situacích.

Pro tvorbu ukázkových diagramů a příkladů byl použit program Enterprise Architect, který umožňuje modelování různých reálných strukturálních a behavoriálních situací. Pro zobrazení SysML diagramů byl nainstalován doplněk ze stránek firmy Sparx Systems.

ZÁVĚR V ANGLIČTINĚ

Using available materials was created description of software engineering, SysML description and practical examples of that language. That material is also useful for education and practical purposes.

This thesis contains short brief of UML language. Following part software engineering map over basic short view of that science discipline. SysML part of that thesis explains language history and used diagrams with basic description. Last part contains practical use of SysML language on two examples. Examples describe usability of particular diagrams in different situations.

For creation of ilustrated diagrams and examples was used program Enterprise Architect which allows user to model various real structural and behavioural situations. For correct displaying was neccessary install software from Sparx Systems.

SEZNAM POUŽITÉ LITERATURY

[1] UML Tutorial [online]. [cit. 2010-02-22]. Dostupný z WWW: http://www.sparxsystems.com/uml-tutorial.html.

[2] History of UML [online]. [cit. 2010-02-22]. Dostupný z WWW: http://atlas.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/history_of_uml.htm

[3] UML: co je UML, historie UML [online]. [cit. 2010-02-22]. Dostupný z WWW: http://mpavus.wz.cz/uml/uml-uvod-1.php

[4] KOSSIAKOFF, A.; SWEET, W., N., Systems engineering principles and practice. John Wiley & Sons, Inc., 2003, s. 463.

[5] WEILKIENS, T., Systems Engineering with SysML/UML. Morgan Kaufmann OMG Press, 2006, s. 299.

[6] Enterprise Architect – Product Demonstrations. [online]. [cit. 2010-02-22]. Dostupný z WWW: http://www.sparxsystems.com/resources/demos/sysml/sysml_interactive.htm

[7] An overview of the System Modelling Language for product and system development. [online]. [cit. 2006-08-15]. Dostupný z WWW: http://www.ibm.com/developerworks/rational/library/aug06/balmelli/.

[8] SysML – Open Source Specification Project. [online]. [cit. 2010-02-22]. Dostupný z WWW: http://www.sysml.org/.

[9] Altova UModel 2010 [online]. [cit. 2010-02-22]. Dostupný z WWW: http://manual.altova.com/.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

UML	Unified Modelling Language, spojovací modelovací jazyk
OMT	Object Modeling Technique
OOSE	Objected-Oriented Software Engineering
OMG	Object Management Group
HP	Hewlett Packard
SysML	Systems Modeling Language
EA	Enterprise Architect

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: NÁZEV PŘÍLOHY