Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-5220-53005

Bc. Pavel Sluka

Špecifiká vlastností transformácií medzi vybranými konceptuálnymi modelmi

Diplomová práca

Vedúci diplomovej práce: Ing. Ján Lang, PhD.
Máj 2017

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-5220-53005

Bc. Pavel Sluka

Špecifiká vlastností transformácií medzi vybranými konceptuálnymi modelmi

Diplomová práca

Štúdijný program: Softvérové inžinierstvo
Štúdijný odbor: 9.2.5 Softvérové inžinierstvo
Miesto vypracovania: Ústav informatiky, informačných systémov a softvérového inžinierstva, FIIT STU Bratislava
Vedúci diplomovej práce: Ing. Ján Lang, PhD.
Máj 2017

Čestné prehlásenie

Čestne prehlasujem, že som na tejto správe k diplomovému projektu pracoval samostatne, využitím znalostí získaných v predchádzajúcom období, analýzou odbornej literatúry a iných zdrojov, ktoré je možné nájsť v zozname literatúry.

Poďakovanie

Týmto by som sa chcel poďakovať vedúcemu tohto diplomového projektu Ing. Jánovi Langovi, PhD., za mnohé cenné usmernenia, nápady a rady, ktoré mi poskytol, a ktorých vplyv napomohol k zvýšeniu kvality tejto práce. Rovnako ďakujem za ochotu, trpezlivosť a čas, ktorý mi bol venovaný pri pravidelných konzultáciách k projektu počas semestra.

Osobitne chcem poďakovať mojim rodičom a najbližším, bez ktorých podpory a pomoci by práca na tomto projekte bola pre mňa omnoho náročnejšia.

Anotácia

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

ŠPECIFIKÁ VLASTNOSTÍ TRANSFORMÁCIÍ MEDZI VYBRANÝMI KONCEPTUÁLNYMI MODELMI

Štúdijný program: Softvérové inžinierstvo
Autor: Bc. Pavel Sluka
Vedúci diplomovej práce: Ing. Ján Lang, PhD.
Máj 2017

Pre návrh softvéru je dôležité správne získavanie požiadaviek zo strany zákazníkov. Ich obsah je dôležité zaznamenávať vo vhodnej podobe modelov. Odborníci z domény softvérového inžinierstva na takúto vizualizáciu využívajú niektorú z odborných modelovacích techník, napríklad UML. Pre zákazníka však takáto forma môže byť problémom. Ten, z dôvodov odlišnej odbornosti, nemusí takýmto modelom bez vysvetlenia pochopiť. Pre špecialistov z domén mimo vývoja softvéru môžu byť lepšie pochopiteľné iné formy zápisu. K takýmto formám patria rôzne techniky konceptuálneho modelovania. Tie sú aj vďaka svojej jednoduchej notácii veľmi intuitívne a modely v nich dokáže vytvárať bez problémov ktokoľvek. Pre využitie v ďalších fázach vývoja však už táto technika nemusí byť vhodná. Je preto nutné ju jednoduchým spôsobom preniesť do inej notácie. Pre zachovanie počiatočnej výhody využitia konceptuálnych modelov je však nutné, aby takýto proces bol zabezpečený efektívnymi transformáciami. Práca sa preto zaoberá analýzou potenciálne využiteľných konceptuálnych techník v softvérovom inžinierstve a ich transformáciami.

Annotation

Slovak University of Technology Bratislava
Faculty of Informatics and Information Technologies

SPECIFICS OF TRANSFORMATION PROPERTIES BETWEEN SELECTED CONCEPTUAL MODELS

Degree course: Software engineering
Author: Bc. Pavel Sluka
Supervisor: Ing. Ján Lang, PhD.
May 2017

In the process of software development, it is essential to get all customers' requirements in a sufficient way. Their content has to be visualized in a suitable form of models. Software specialists use variety of such modeling techniques for it, for example UML. This representation can be considerable problem for the customer. The customer may not understand these models, because of his different knowledge. For the experts outside of software engineering domain, there may be other suitable modeling technique for requirements' gathering. These techniques include for example conceptual models. Conceptual modeling techniques are easy to use thanks to their uncomplicated notation, and almost everybody can create them with ease. However, these kinds of notations may not be suitable for next stages of software development life cycle. In this case, we have to transform these models to other modeling language and this transformation has to be efficient enough. This paper consists of analysis of potentially usable conceptual modeling techniques in software development and deals with their transformations.

Obsah

1 Úvod **1**

2 Modely a modelovanie **5**
2.1 Unified Modeling Language 6
2.2 Business Process Modeling Notation 7
2.3 Ontológie 7

3 Konceptuálne a mentálne modelovanie **9**
3.1 Myšlienkové mapy 10
3.2 Kognitívne mapy 11
3.3 Konceptuálne diagramy a konceptové mapy 11
3.4 Sémantické siete 12
3.5 Mapy námetov 13
3.6 Argumentačné mapy 14
3.7 Iné techniky 14

4 Porovnanie vizualizačnej sily modelov **17**
4.1 Porovnanie UML a myšlienkových máp 17
4.2 Porovnanie UML a kognitívnych máp 18
4.3 Porovnanie UML a konceptových máp 19

5 Transformácie medzi modelmi **21**

6 Využitie konceptuálneho modelovania **25**

7 Evaluácia schopností konceptuálneho modelovania **27**

8 Spracovanie prirodzeného jazyka **31**
8.1 Lexikálny kontext myšlienkových máp 31
8.2 Možnosti uplatnenia v zamýšľanom prístupe 32
8.3 Rozpoznávanie slovných druhov 32

9 Pohľady v myšlienkovej mape **35**
9.1 Zavedenie pohľadov 36
9.2 Využitie pohľadov v zamýšľanom prístupe 37

10 Matematické modely podobnosti mapových štruktúr **39**
10.1 Metriky podobnosti 39
10.2 Náročnosť a súvislosti metrík 42

11 Špecifikácia návrhu riešenia **43**
11.1 Zachytávanie požiadaviek 43
11.2 Transformácie modelov 44
11.3 Refaktoring modelov 44
11.4 Príspevok k obohateniu modelov z pohľadu Content Engineeringu 45
11.5 Modelom riadený vývoj 45
11.6 Obohatenie digitálnych konceptuálnych modelov 46
11.7 Prínos k zvýšeniu kvality softvéru 46

12 Navrhnutý prístup **47**
12.1 Uvažované tvary modelov 47
12.2 Definície transformačných pravidiel 49
12.3 Transformačný nástroj 51
12.4 Testovacie scenáre 53

13 Základná evaluácia prístupu **57**
13.1 Zadanie experiemntu 57
13.2 Získané modely a ich podobnosť 58
13.3 Vyhodnotenie 60

14 Rozšírený prístup **65**
14.1 Uvažované tvary modelov 65
14.2 Zasiahnuté UML diagramy 66
14.3 Transformačné pravidlá 66
14.4 Transformačný nástroj 67

15 Podrobná evaluácia prístupu **69**
15.1 Evaluačné scenáre 69
15.2 Vyhodnotenie scenárov 71
15.2.1 Porovnávanie transformovaných myšlienkových máp do UML voči očakávaným výsledkom v UML 71
15.2.2 Porovnávanie transformovaných UML modelov v tvare myšlienkových máp 76
15.2.3 Myšlienkové mapy porovnávané po transformácii do UML a späť do pôvodného tvaru 77

15.2.4 UML modely transformované do myšlienkovej mapy a späť do UML 77
15.2.5 Porovnávanie čiastkových modelov 78
15.2.6 Scenáre nejednoznačnosti myšlienkových máp 80
15.2.7 Porovnanie viacnásobného použitia transformácií tam a späť 82
15.2.8 Diskusia k rozšíreniu metamodelu myšlienkovej mapy 82
15.2.9 Možnosti aplikácie požiadaviek definovaných v myšlienkovej mape na tvorbu rôznych druhov UML diagramov 84
15.2.10 Porovnanie s existujúcimi prístupmi 87
15.3 Zhodnotenie a sumarizácia 90

16 Súvisiace práce **91**

17 Záver a ďalšia práca **93**

Literatúra **97**

A Zadanie pre experiment **101**

B Navrhnuté riešenia **103**

C Popis nástroja umInd **107**

D Návod na použitie **109**
D.1 Základné použitie na transformácie 109
D.2 Použitie na porovnávanie myšlienkových máp 110

E Popis nástroja umInd3 **111**

F Návod na použitie nástroja umind3 **113**
F.1 Potrebné nástroje pre modelovanie 113
F.2 Potrebné nástroje pre prácu s umind3 113
F.3 Formy modelov 114
F.4 Použitie nástroja 114

G Obsah priloženého CD **115**

H Dokumentácia evaluácie **117**
H.1 Zadanie experimentu 117
H.2 Výsledky po respondentoch 119

I Vyhodnotenie plánu prác **121**

J Článok z konferencie IIT.SRC **123**

K Poster z konferencie IIT.SRC **129**

Kapitola 1

Úvod

Pri vývoji akéhokoľvek softvérového produktu je mimoriadne dôležité poznať dostatočne otázky týkajúce sa problémovej domény, pre ktorú je daný softvér vyvíjaný. V momente, keď sa vývojári stretávajú s neznámou doménou, je nevyhnutné porozumieť súvislostiam, ktoré sú pre danú doménu kľúčové. Tie sú neraz netriviálne a ich pochopenie si vyžaduje mimoriadne úsilie, ktoré nie je možné získať v dostatočne krátkom časovom období. Odborníci v danej doméne získavajú prehľad o dôležitých súvislostiach v oblasti dlhšiu dobu, pričom tento prehľad je často podmienený niekoľkoročnými skúsenosťami.

Práve v prípade nedostatočného rozhľadu v problémovej oblasti sa najviac prejavuje dôležitosť zapojenia zákazníka do cyklu vývoja softvéru. Pre zachovanie dôležitých vnútorných súvislostí, ktoré sú vo vyvíjanom softvéri významné, je potrebné dostatočne pochopiť ciele, ktoré podmienili samotnú prvotnú požiadavku na vznik projektu. Je mimoriadne dôležité od zákazníka získať požiadavky v takej podobe, aby výsledok bol v súlade s jeho stanovenými očakávaniami.

V momente získavania požiadaviek je potrebné ich zachytiť v ľahko pochopiteľnej podobe. Osoba, ktorá má na starosti komunikáciu so zákazníkom, musí byť dostatočne skúsená, aby dokázala odhaľovať možné nedostatky prichádzajúce zo strany zákazníka. Dôležité sú v tejto fáze stretnutia, počas ktorých sa samotné požiadavky zaznamenávajú. Počas takéhoto stretnutia nie je často možné zaznamenávať celý priebeh a všetky požiadavky do formy uceleného textu. Z týchto dôvodov je potrebné využívať určitú grafickú reprezentáciu, pomocou ktorej je možné reprezentovať štruktúru alebo správanie vyvíjaného softvéru. Predchádzajúca veta napovedá k využitiu všeobecne používaných grafických notácií v doméne softvérového inžinierstva. Ide najmä o sadu diagramov z jazyka UML a iné zaužívané alternatívy určené na konkrétne použitie (BPMN, diagram toku údajov, ER diagram, vývojový diagram, a i.).

Na strane vývojára sa očakáva dostatočná znalosť týchto notácií. Problémom je strana zákazníka. Rozdiely v odbornosti na oboch stranách sa môžu

výrazne prejaviť práve pri otázke správneho pochopenia požiadavky zachytenej v grafickej podobe. Jej význam nemusí byť jednoducho pochopiteľný a vyžaduje si spätné vysvetlenie zaznamenaného diagramu. Až následne vie zákazník prejaviť súhlas či nesúhlas so zaznamenanou položkou. Komunikácia požiadaviek sa tak môže natiahnuť ešte viac, akoby bol zaznamenávaný celý text rozhovoru. Problém narastie, ak sa požiadavka komunikuje bez priameho osobného kontaktu. Pri nedostatočnej rozvahe zúčastnených strán sa môže do ďalšej fázy vývojového cyklu softvéru dostať zle pochopená požiadavka. Jej vplyv môže mať výrazne negatívny následok. Ten úzko závisí od modelu životného cyklu softvéru. Následok má však takmer s určitosťou dosah na čas dodania, cenu alebo kvalitu. Dosah bude o to negatívnejší, čím neskôr sa pochybenie odhalí (najväčšie problémy napr. pri vodopádovom cykle).

Problémy s odlišnou mierou skúseností s rôznymi modelovacími stratégiami je možné minimalizovať využitím iných techník. Vyberať je potrebné z takých notácií, ktoré sú zrozumiteľné väčšine. K nim patria rôznorodé stratégie vychádzajúce z konceptuálneho modelovania. Príkladmi takýchto stratégií sú všeobecné konceptuálne diagramy, konceptuálne mapy, myšlienkové mapy, kognitívne mapy a mnohé ďalšie. Tieto modelovacie prístupy sú veľmi intuitívne a počet ľudí, ktorí sa s nimi stretli, minimálne v akademickom prostredí, je vysoký. Prepájanie rôznych objektov je jednoduché z pohľadu prepájania jednoduchých entít reálneho sveta na základe logických súvislostí, ktoré sú medzi nimi.

Modely vytvárané pomocou takýchto notácií je možné využívať na veľké množstvo úloh. Pre prvotné zachytenie požiadaviek zákazníka sa môžu ukazovať ako vhodná alternatíva k technickejším modelovacím technikám. Ich priama zrozumiteľnosť pre všetky zúčastnené strany je hlavnou výhodou. Účastníci pochádzajúci z domén mimo softvérového inžinierstva dokážu v takýchto notáciách vyjadriť požiadavky aj bez výraznej pomoci zo strany softvérového inžiniera, a takto vytvárané modely dokážu sprostredkovať veľa kľúčových informácií.

Vytvárané modely môžu byť zaujímavé aj pre následnú tvorbu zápisov v softvérových modelovacích jazykoch. Ich využitie je výhodnejšie pre druhú stranu reprezentujúcu odborníka. Vďaka ich rozsiahlejšej notácii je v nich možné reprezentovať väčšie množstvo elementov, ktorých použitie je v ďalších fázach vývoja lepšie pochopiteľné. Ak uvažujeme nad využívaním konceptuálnych modelov v prvotných fázach vývoja, pričom v ďalších sa už môže prejaviť nutnosť používať iné zápisy, dostávame sa do problému s ich transformáciami. Pre transformácie musia platiť nutné kritériá, ktoré zabezpečia ich jednoznačný priebeh. Ten môže byť vďaka tejto vlastnosti zachytený sadou pravidiel pre automatizované transformácie. Problém na riešenie je však zachovanie celého vnútorného zmyslu modelu, ktorý sa prenáša do inej notácie. Tá svojou expresívnou silou môže prevyšovať pôvodnú notáciu. Je preto potrebné zaviesť také pravidlá, ktoré dokážu mapovať synonymické

elementy tak, aby boli zrozumiteľné aj v inom mapovaní než jedna k jednej.

V kapitole 2 sa z pohľadu potreby poznania všeobecných faktov o modelovaní v prostredí softvérového inžinierstva popisujú všeobecné informácie o modelovaní. Okrem toho je dôležité prejsť základné informácie o modelovacích notáciách, ktoré sú dôležité vo vývoji softvéru. Dôraz je kladený na tie modely, ktoré sú kľúčové aj z pohľadu tohto projektu.

Kapitola 3 analyzuje druhú stranu spomínaných modelovacích stratégií. Z mnohých modelovacích stratégií mentálneho a konceptuálneho modelovania sa sústreďuje na tie najpoužívanejšie, a tie, ktoré sú pre projekt prioritne dôležité. V kapitole 4 je pre potrebu poukázania modelovacích možností vo fáze analýzy, prípadne počiatočného návrhu softvéru, nevyhnutné zvýrazniť hľadanie možnej synonymie pri modelovaní v UML a vo vybraných konceptuálnych modelovacích technikách. Porovnaním je možné zvýrazniť silné a slabé stránky úvah o vzájomnom nahradení UML a týchto stratégií na modelovanie iných informácií, než na aké sú prioritne využívané.

Z pohľadu výskumu možností vzájomného obohatenia a prechodov medzi modelmi je potrebné analyzovať transformácie medzi modelmi, ktoré sú zachytené v kapitole 5. Pre transformácie je potrebné nájsť a špecifikovať výber základných vlastností, ktorých splnenie je nevyhnutné pre zachovanie cieľu zachytenom v konkrétnom diagrame.

S popísanými motiváciami z pohľadu možností výskumu súvisí kapitola 6, ktorá opisuje potenciál mentálnych konceptuálnych modelov pri použití v softvérovom inžinierstve. Niektoré možnosti boli evaluované v počiatočnom experimente v kapitole 7, ktorý potvrdzuje potenciál daných úvah.

V počiatočnej fáze nášho výskumu bol špecifikovaný možný návrh uvedený v kapitole 11, ktorý tvorí podstatnú časť neskoršie realizovaného výskumu. Vzhľadom na uvažované uplatnenie techník spracovania prirodzeného jazyka na diferenciáciu jednotlivých UML elementov je v kapitole 8 uvedená základná analýza vhodných techník a zamýšľaný princíp členenia. Pre zavádzané vrstvenie myšlienkových máp je kľúčová kapitola 9, ktorá opisuje vznik tohto prístupu. Pre evaluáciu našich predpokladov o modelovaní požiadaviek v myšlienkových mapách kýmkoľvek v jednotnej očakávanej forme, je potrebné rovnako evaluovať aj podobnosť myšlienkových máp. Na ich porovnávanie sa využívajú matematické výpočtové metódy uvedené v časti 10. Základ navrhnutého prístupu je zhrnutý v časti 12, ktorá obsahuje opis jeho oporných bodov, jeho samotnú implementáciu v prvotnom prototype a špecifikuje testovacie scenáre pre základ evaluácie. Zhrnutie evaluácie tohto prístupu na prvých príkladoch sa nachádza v časti 13, z ktorej je možné usúdiť dosiahnutie prvotne vytýčených cieľov prístupu.

Doplnenie prístupu o nové neskôr odhalené možnosti sa nachádza v kapitole 14. Obsahuje popis zmien a doplnkov oproti prvotnému návrhu a následne je takto špecifikovaný prístup evaluovaný v kapitole 15. Tá obsahuje sumarizáciu cieľov jednotlivých hodnotených scenárov a ich samotné výsledky. Súčasťou je porovnanie vizualizačných techník, rozšírenie modelova-

cej techniky myšlienkových máp a porovnanie s inými podobnými prístupmi, ktoré sa podarilo zanalyzovať a porovnať s nami navrhnutým prístupom. Kapitola zároveň zhŕňa preskúmané možnosti prístupu v rámci evaluácie.

V kapitole 17 sa nachádza zhrnutie výskumu a popis ďalších možností pokračovania vo výskume myšlienkových máp v preberaných súvislostiach. Práca sa venuje výskumu, ktorý je z určitého pohľadu novým prístupom, avšak podobnému súvisiacemu výskumu sa venovali už aj iné práce, z ktorých najvýznamnejšie sú zhrnuté v kapitole 16. V prílohách je možné nájsť súvisiace časti týkajúce sa počiatočnej evaluácie A a B a podrobnejšej priebežnej evaluácie H, rovnako ako aj dokumentáciu k dôležitým súčastiam výskumu, akými sú vytváraný nástroj C založený na vytvorenom prístupe a jeho návod na použitie D. Prílohy zároveň obsahujú aj obsah priloženého elektronického média G.

Kapitola 2

Modely a modelovanie

Modelovanie ako vytváranie modelov je vysoko využívaným prostriedkom zachytávania zjednodušenej formy reprezentácie výsledného diela. Využíva sa naprieč mnohými vednými disciplínami, pričom význam modelovania sa medzi týmito disciplínami môže výrazne líšiť. Vzhľadom na tieto existujúce odlišnosti je náročné jednoducho definovať samotný pojem model. Často využívaný význam modelu v dnešnej dobe súvisí so zobrazovaním reality do určitej zjednodušenejšej, abstraktnejšej podoby, ktorá môže predstavovať akýsi podklad alebo vzor pre modelovaný objekt.

Pre oblasť softvérového inžinierstva je modelovanie mimoriadne dôležité. V priebehu vývoja softvéru je dôležité zaznamenávať konkrétne špecifikované požiadavky v ľahko zrozumiteľnej podobe. Daná podoba, ako abstraktnejšie poňatie vyvíjaného produktu, musí svojou presnosťou a dostatočnosťou opisu napomôcť k prechodu ku konkretizácii v podobe implementácie v zdrojovom kóde. Pre modelovanie týchto skutočností je potrebné využívať také modelovacie techniky, ktoré sú schopné zachytávať všetky vlastnosti softvérového modelu z rôznych pohľadov. Tieto pohľady musia zachytávať systém v podobe jeho statických elementov, rovnako ako aj v podobe očakávaného správania. Pre statické elementy je potrebné zavádzať modely, ktoré zaručujú dostatočné zachytenie štruktúry systému. Na druhej strane, pre zachytávanie správania je nutné dokázať reprezentovať mnohé dynamické elementy, ktoré určujú chovanie sa systému v rôznych úlohách. Pre tieto reprezentácie existuje viacero modelovacích techník, ktoré sa v priebehu histórie softvérového inžinierstva menili a postupne dopĺňali o nové.

Počas počiatočnej evolúcie softvérového inžinierstva sa využívali mnohé modelovacie techniky, ktoré sa s nástupom nových technológií postupne vytláčali. Pre reprezentáciu krokov konkrétnych algoritmov bola veľmi obľúbená grafická reprezentácia pomocou vývojových diagramov. Táto technika, ktorá vznikla už v roku 1921 (v oblasti od 1947), bola využívaná už od prvopočiatkov vzniku softvérového inžinierstva, resp. informatiky ako takej. Okrem tejto, stále využívanej modelovacej techniky pre charakterizáciu po-

stupu algoritmu a jeho správania na základe rôznych udalostí, sa rozvíjali aj mnohé iné techniky. S postupne stúpajúcou potrebou modelovania uchovávaných informácií v databázach prišli prvé pokusy o modelovanie informácií v podobe entitno-relačného diagramu, ktorého vznik sa viaže na rok 1976. Najvýraznejší prelom prišiel až so zavádzaním objektovo-orientovaného prístupu.

Objektovo-orientovaný prístup, ktorý začínal postupne od osemdesiatych rokov, pričom najväčší prelom zaznamenal v deväťdesiatych rokoch, priniesol celý rad nových modelovacích technológií, ktorých prístupy boli zachytené napríklad v metodike Objektovej modelovacej techniky (OMT, Object Modeling Technique). Medzi ďalšími podobnými príkladmi je vhodné uviesť objektovú modelovaciu techniku samotného Objektovo-orientovaného softvérového inžinierstva (OOSE, Object-oriented software engineering) a Boochovu metódu. Výsledkom mnohých snáh o zjednotenie reprezentácie týchto modelovacích metód môžeme nájsť v postupne sa formujúcom jazyku UML (Unified Modeling Language). Najvýraznejší dôraz bol pritom dávaný na zjednotenie metód, za ktorými stáli traja softvéroví inžinieri známi ako The Three Amigos: Grady Booch (Boochova metóda), Ivar Jacobson (OOSE) a James Rumbaugh (OMT). Vývoj v tejto štandardizácii postupne spôsobil vytlačenie iných modelovacích metód a v súčasnosti používané modelovacie metódy sú prevažne odvodeninami z UML a jej konkrétnych súčastí.

2.1 Unified Modeling Language

Unified Modeling Language (UML) je grafický modelovací jazyk prioritne pre objektovo-orientovaný prístup. Zabezpečuje možnosti vizualizácie návrhu softvéru, pričom zachytáva navrhovaný softvér v mnohých ohľadoch jeho dynamickej aj statickej stránky. Slúži zároveň ako nápomocný nástroj, pomocou ktorého je zachytávaná dokumentácia navrhovaného softvéru. Prvá oficiálna verzia UML (UML 1.1 – predstavená 1996, prvá verzia 0.8 už v roku 1995) sa začala postupne presadzovať po štandardizácii v roku 1997.

Jazyk UML vznikol ako projekt s cieľom štandardizácie viacerých využívaných a novovznikajúcich grafických notácií, ktoré sa začali rozširovať s postupným rozvojom objektovo-orientovaného prístupu. Okrem vyššie spomínaných grafických jazykov OOSE, OMT a Boochovej metódy mali na vznik UML vplyv aj sčasti nezávislé grafické reprezentácie stavových diagramov, ktoré vznikali ešte skôr a jednotlivé modelovacie metódy si ich osvojili. Ďalším dôležitým jazykom, ktorý prispel k rozvoju UML bol aj jazyk OCL (Object Constraint Language) od IBM.

Posledná verzia UML označená ako verzia 2.5 bola vydaná ako verzia „in progress“ v októbri 2012, v roku 2015 bola už vydaná oficiálne. UML celkovo pozostáva z viacerých diagramov pre špecifické použitie v softvérovom inžinierstve. Základné členenie rozdeľuje diagramy podľa toho, či opisujú

štruktúru (statickú časť) alebo správanie (dynamickú časť).

Štruktúrne diagramy UML zobrazujú štruktúru systému ako takú a znázorňujú teda elementy systému bez ohľadu na čas. K týmto diagramom patria diagram tried, diagram objektov, diagram profilov, diagram kompozitnej štruktúry, diagram rozloženia, diagram balíkov a diagram komponentov. K diagramom modelujúcim správanie patria diagramy aktivít, diagramy prípadov použitia, stavové diagramy a skupina diagramov interakcie – sekvenčný diagram, diagram prehľadu interakcií, diagram komunikácie a diagram časovania.

Použitie jednotlivých diagramov sa odlišuje od fázy životného cyklu vývoja softvéru. Pre účely tejto práce sú najvýznamnejšie všeobecne najpoužívanejšie diagramy z fázy analýzy a návrhu. K nim môžeme zaradiť vybrané štruktúrne diagramy (najmä diagram tried, ale aj iné) a vybrané diagramy správania (najmä diagramy prípadov použitia, potom diagramy aktivít, sekvenčné diagramy a stavové diagramy). Popis jednotlivých diagramov je zaujímavý najmä z pohľadu porovnania vyjadrovacej sily notácii jednotlivých diagramov voči zamýšľaným mentálnym a konceptuálnym modelovacím technikám.

2.2 Business Process Modeling Notation

Business Process Modeling Notation vznikol ako grafická reprezentácia pre zachytávanie a špecifikáciu biznis procesov. Táto grafická notácia vznikala ako projekt Business Process Management Initiative, ktorá sa v roku 2005 zlúčila s Object Management Group (OMG), ktorá spravuje aj štandardy pre UML.

Notácia BPMN sa často porovnáva s UML, v ktorom na modelovanie biznis procesov slúži diagram aktivít. Tieto notácie využívajú veľmi podobné symboly, ktoré môžu viesť k domnienke takmer rovnakého významu. Medzi symbolmi sa však dajú nachádzať rozdiely. Rozdiely je možné vysvetliť odlišným poňatím modelovacieho prístupu. Zatiaľ čo pri UML je celkovo kladený dôraz na objektovo-orientované modelovanie, pri BPMN dáva do popredia procesy. Rozdiel je aj v kladení dôrazu na zrozumiteľnosť diagramov. Zatiaľ čo UML je jazykom, ktorý sa sústreďuje na profesionálov z oblasti vývoja softvéru, BPMN je navrhnutý tak, aby bol čo najintuitívnejší pre použitie aj medzi používateľmi z business sféry.

2.3 Ontológie

Pojem ontológie je vo všeobecnosti definovaný ako filozofická náuka o bytí, existencii, prípadne aj ako náuka o kategóriách bytia a vzťahov medzi nimi. Zároveň je chápaná aj ako náuka zaoberajúca sa generálne opisom rôznorodých objektov a vzťahov medzi nimi v reálnom svete. Pre oblasť softvéro-

vého inžinierstva, rovnako ako aj iných odvetví informatiky a informačných technológií, je dôležitý práve tento druhý význam. Definovať ich môžeme ako formálnu, explicitnú špecifikáciu zdieľanej konceptualizácie [25]. Vychádza z nej význam formálneho pomenovania typov objektov, vzťahov medzi nimi, vlastností, objektov a vlastností ich vzťahov. Ontológie v tomto význame okrem softvérového inžinierstva zohrávajú dôležitú úlohu v umelej inteligencii, systémovom inžinierstve, biomedicínskych oblastiach informatiky, v prostredí sémantického webu a mnohých ďalších.

V závislosti na využití existuje niekoľko triedení týchto jazykov. K významným jazykom ontológií patrí napríklad OWL (Web Ontology Language), OntoUML, DOGMA (Developing Ontology-Grounded Methods and Applications), Common Logic, MOF (Meta Object Facility), CASL (Common Algrebraic Specification Language) a mnohé iné.

Základnými komponentami ontológií sú indivíduá, triedy, relácie, hodnoty a dátové typy a axiómy. Indivíduá je možné chápať ako konkrétne inštancie objektov. Triedy majú význam, ktorý je možné porovnávať s definíciou triedy v objektovo-orientovanom prístupe, resp. ako typ objektu. Axiómy majú za úlohu vymedzenie vzťahu k triedam a reláciám. Pre relácie, hodnoty a dátové typy platia štandardné pravidlá použitia, podobné iným využitiam v štandardných modelovacích stratégiách.

Kapitola 3

Konceptuálne a mentálne modelovanie

Konceptuálne a mentálne modelovacie techniky tvoria dôležitú súčasť metód informačného mapovania. Informačné mapovanie sa v tomto význame môže definovať ako zapisovanie alebo zbieranie jasných, výstižných informácií sústredených okolo zvoleného cieľa. Cieľ je založený na potrebách, ktoré sú stanovené v úvode zberu informácií. V závislosti na stanovenom cieli využitia finálnych informácií je možné definovať v rámci informačného mapovania viaceré oblasti. Techniky využívané na mapovanie informácií pri tom môžu byť využité vo viac ako len jednej takejto oblasti.

Informačné mapovanie vzniklo ako výskumná práca Roberta E. Horna. Jeho záujem v oblastiach vizuálnej reprezentácie informácií prispel k rozvoju mnohých techník pre zber a zaznamenávanie informácií v určitej vizualizačnej notácií. Jeho výskumy v tejto oblasti poukázali na to, že takéto mapovanie má nezanedbateľný prínos pre zrýchlenie prístupu k informáciám, ktoré je zároveň možné jednoducho reprezentovať a pochopiť. Modelovanie tak prináša signifikantné zlepšenie chápania súvislostí informácií a výkonu práce s nimi.

Konkrétne oblasti, v ktorých sa takýto zber informácií vo veľkom využíva sú napríklad akademické oblasti. V nich je možné, prostredníctvom grafických záznamov informácií a vzťahov medzi nimi, reprezentovať sieť faktov. Najčastejšou využívanou technikou na zápis takýchto dát sú myšlienkové mapy. Okrem nej k široko využívaným modelovacím technikám patria napríklad kognitívne mapy, konceptuálne diagramy, konceptové mapy, sémantické siete a mnohé ďalšie. Vzhľadom na to, že tieto modely je možné využívať v rôznych disciplínach, prípadne interdisciplinárne, je potrebné odlišovať ich kategorizáciu v rámci mentálnych modelovacích metód a konceptuálnych modelovacích metód.

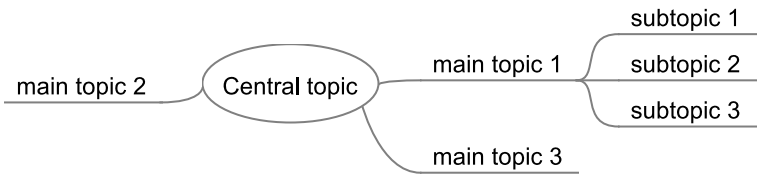
Pri mentálnom modelovaní sa modely využívajú na zachytávanie predstáv o probléme reálneho sveta, ktorého sa týkajú. Tvorca mentálneho mo-

delu pri tom zachytáva jednotlivé pojmy a vzťahy medzi nimi, pričom sa odkazuje na známe poznatky, ktoré sú pre daného tvorcu známe, má ich v pamäti a dokáže s nimi pracovať nad množinou prístupov logického myslenia. Do tejto množiny patria princípy indukcie a dedukcie, generalizácie a konkretizácie. Pri konceptuálnom modelovaní sa naopak modeluje zamýšľaný vzhľad, resp. štruktúra, ktorá vychádza len z návrhu, nie z používania faktu.

V návrhu softvéru je možné sledovať prelínanie týchto modelovacích prístupov. Vzhľadom na to, že ide o dizajn novej, často nepoznanej alebo neexistujúcej domény, pre ktorú sa softvérový produkt navrhuje. Najmä na strane zákazníka sa však zjavne prejavuje aj mentálne modelovanie vyplývajúce zo znalostí problémovej domény. Tie však musia byť prispôsobené tvorbe daného softvéru. Z toho vyplýva, že analýza a návrh softvéru sa pri konzultácii so zákazníkom sústreďuje na konceptuálne modelovanie, pričom sa však v nemalej miere opiera o známe fakty pochádzajúce skôr z mentálneho modelovania.

3.1 Myšlienkové mapy

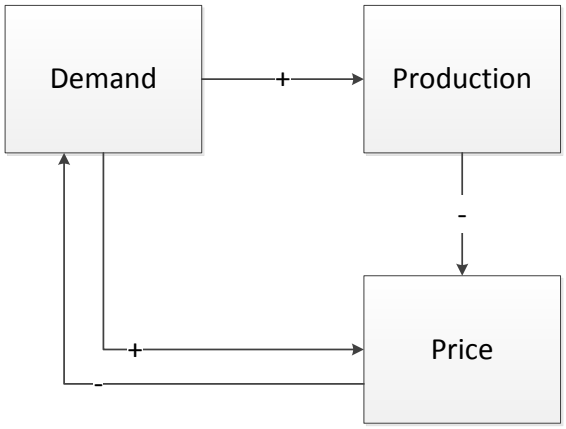
Myšlienkové mapy sú diagramom pre vizuálnu reprezentáciu informácii a vzťahov medzi nimi. Autorom myšlienkových máp je Tony Buzan [6], ktorý myšlienkové mapy formálne popísal v roku 1976. Samotná technika mapovania známych faktov do tohto typu diagramu je však omnoho staršia. Vytváranie akejkoľvek myšlienkovej mapy začína pri centrálnej téme, od ktorej sa prechádza do poduzlov. Myšlienková mapa je tak istou formou diagramu typu pavúčej siete (podobnej ako využíva spider diagram v matematike). Patrí k najrozšírenejším zápisom v mentálnom modelovaní a má široké využitie v akademickej aj firemnej sfére. Využíva sa pri tom fakt jednoduchosti tohto modelovania, na ktoré nie sú potrebné žiadne výrazné znalosti. Pri modelovaní sa využívajú len logické súvislosti medzi jednotlivými vrcholmi v stromovej štruktúre.



Obr. 3.1: Príklad myšlienkovej mapy.

3.2 Kognitívne mapy

Kognitívne mapy sú typom mentálnej reprezentácie. Myšlienku kognitívnych máp ako prvý rozvinul Edward C. Tolman. Prvýkrát boli popísané v roku 1948. Svoje využitie nachádzajú v mnohých sférach. Pre informatiku sú významné najmä jej špecializované podtypy v podobe fuzzy kognitívnych máp a kognitívnych sietí, ktoré sú používané najmä v prostredí umelej inteligencie. Pri modelovaní sa využívajú podobné pravidlá ako pri myšlienkových mapách, rozdielom sú označenia šípok a možnosti centrálnych elementov, ktorých môže byť viacero. Pre kognitívne mapy je typické aj využívanie popisov nad orientovanými hranami prípadne aj ohodnotenie hrany. Výhody ohodnocovania hrán sa využívajú najvýznamnejšie práve v prostredí fuzzy konginitívnych máp, ktoré boli predstavené Bartom Koskom.

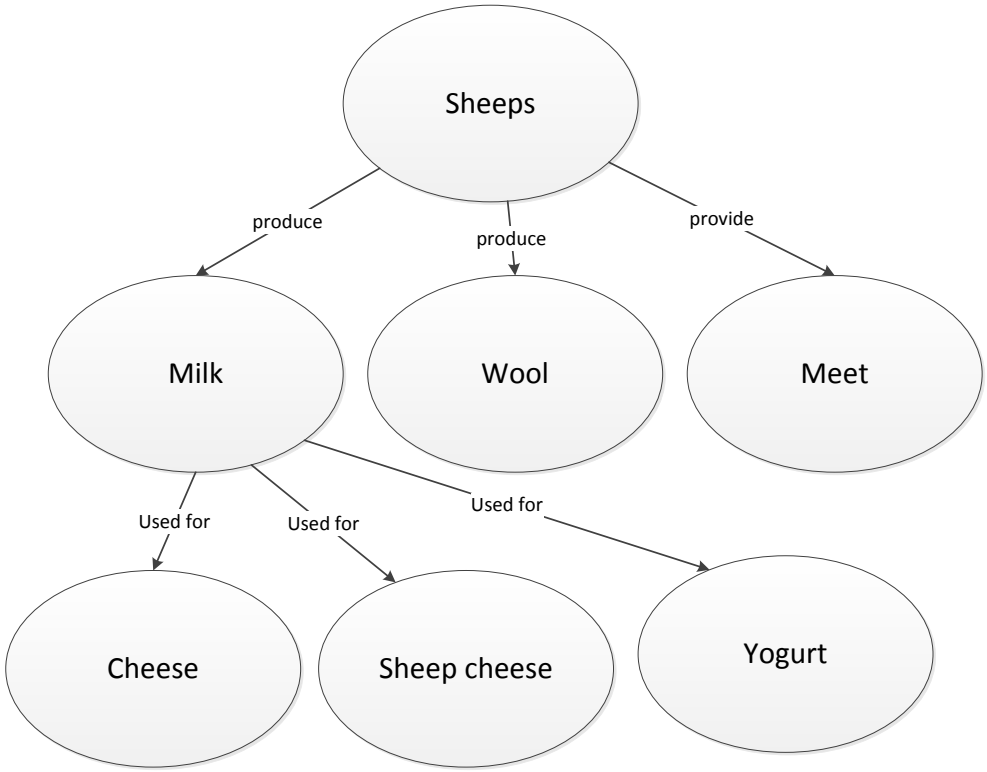


Obr. 3.2: Príklad kognitívnej mapy.

3.3 Konceptuálne diagramy a konceptové mapy

Konceptuálne diagramy a konceptové mapy sú často považované za jeden a ten istý typ diagramu. Vychádza to z definícií, ktoré o konceptových mapách poznáme od ich autora Josepha D. Novaka, ktorý ich so svojím výskumným tímom formuloval v sedemdesiatych rokoch. Medzi týmito dvomi technikami sa však dajú nachádzať rozdiely [10]. Konceptová mapa je zvyčajne charakterizovaná ako zhora nadol smerujúci diagram zachytávajúci previazané fakty. Nad zvyčajne orientovanými hranami sa nachádzajú popisy podobné tým, ktoré môžeme nájsť v ontológiách. Konceptuálny diagram tieto pravidlá obsahuje taktiež, ale nad hranami sa popis zvyčajne neuvádza. Okrem toho sa súčasťou elementu znázorňujúceho fakt často stáva aj podrobnejšie

zachytený popis vlastností danej informácie. Po stránke vzťahov sa neobmedzuje na využitie jedného top-level konceptu. Smerovanie diagramu pri konceptových mapách býva obmedzené na štandardné využitie smerom nadol, pri konceptuálnych diagramoch je rozvoj konceptu možné smerovať zväčša po štruktúre zľava doprava a zhora nadol.

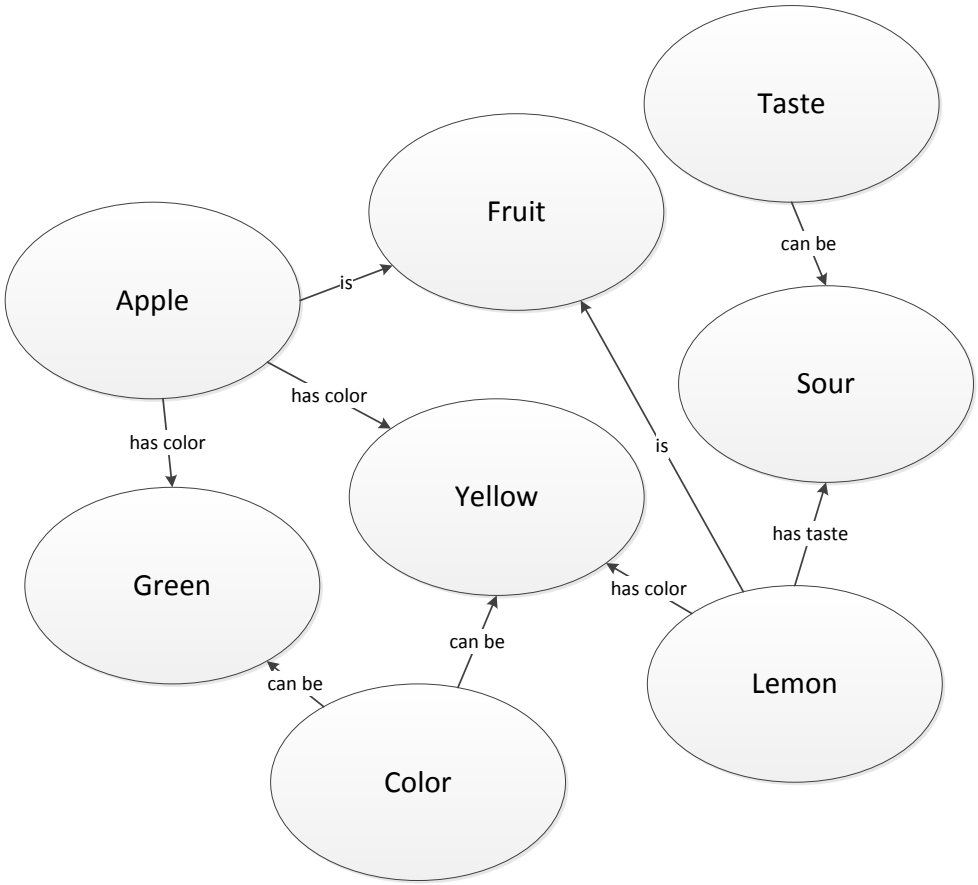


Obr. 3.3: Príklad konceptovej mapy.

3.4 Sémantické siete

Sémantické siete sa dajú považovať za predchodcu viacerých iných mentálnych a konceptuálnych modelovacích techník. Ich myšlienky pochádzajú už zo začiatku 20. storočia, pričom ich oficiálny vývoj súvisí s nástupom počítačov. V roku 1956 ich vyvinul tím okolo Richarda H. Rickensa, ktorých projekt ich označil za mediátora medzi štandardným vyjadrením prirodzeného jazyka a jeho strojového spracovania. Ako vrcholy sa používajú koncepty, ktoré sú prepájané orientovanými šípkami s popisom (rovnako podobným ontológiám). Definície tejto modelovacej techniky ju dokážu pripodobniť ku konceptovým mapám, avšak sémantické siete bývajú omnoho kompli-

kovanejšie a nemajú kvôli tomu výrazné ohraničenie v spôsobe organizácie konceptov v nich zachytených.

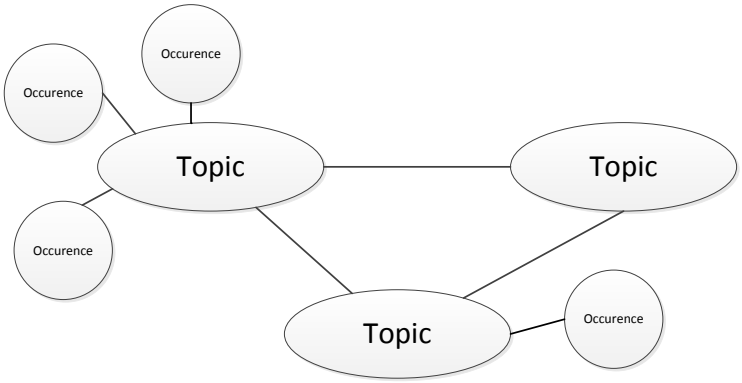


Obr. 3.4: Príklad sémantickej siete.

3.5 Mapy námetov

Mapy námetov, niekedy označované aj ako mapy tém (topic maps) sú štandardom pre reprezentáciu znalostí. Dôraz sa kladie na vyhľadateľnosť informácie na mape. Technika bola vytvorená v deväťdesiatych rokoch pre potreby indexácie výskytov informácie pri publikovaní kníh, na ktorých konci sa takýto zoznam pojmov nachádza. Počas vývoja techniky sa zvýšením abstrakcie modelov objavili nové možnosti takéhoto modelovania. Vďaka zovšeobecneniu týchto modelov pri štandardizácii získali výrazne väčšie možnosti uplatnenia. V súčasnosti spadá do ISO štandardu ISO/IEC 13250:2003. Mapy námetov využívajú vo svojej notácii tri prvky. Námety (topics), aso-

ciácie (associations) a výskyty (occurrences). Námety sú využívané ako koncepty pre reprezentáciu informácie, zatiaľ čo asociácie a výskyty sú používané na reprezentáciu vzťahov. Súvisiace námety sú prepojené medzi sebou asociáciami, zatiaľ čo výskyty reprezentujú zdroje a súvisiace fakty, ktoré sú v priamom vzťahu k informácii. Výskyty môžu byť chápané aj ako listy v stromovej štruktúre diagramu.



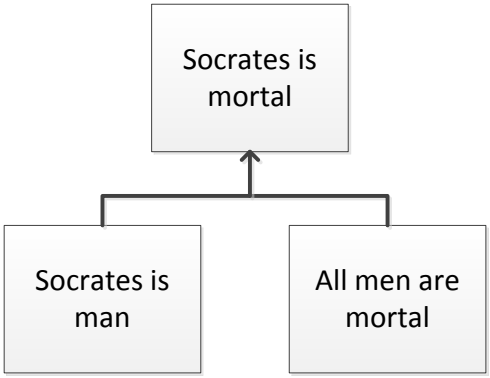
Obr. 3.5: Príklad mapy námetov.

3.6 Argumentačné mapy

Argumentačné mapy sú vizualizačnou technikou, ktorá nachádza svoje uplatnenie najmä v teórii logiky. Prvopočiatky pochádzajú od Richarda Whatelyho, ktorý sa o nich zmieňuje ako prvý v roku 1836. Argumentačná mapa ako diagram znázorňuje v sebe krabičky a šípky. V jednotlivých krabičkách sú zachytené závery a premisy konkrétneho argumentu. Z nej sa následne prechádza na ďalší prvok prepojený šípkou. Diagram je štruktúrovaný zdola nahor.

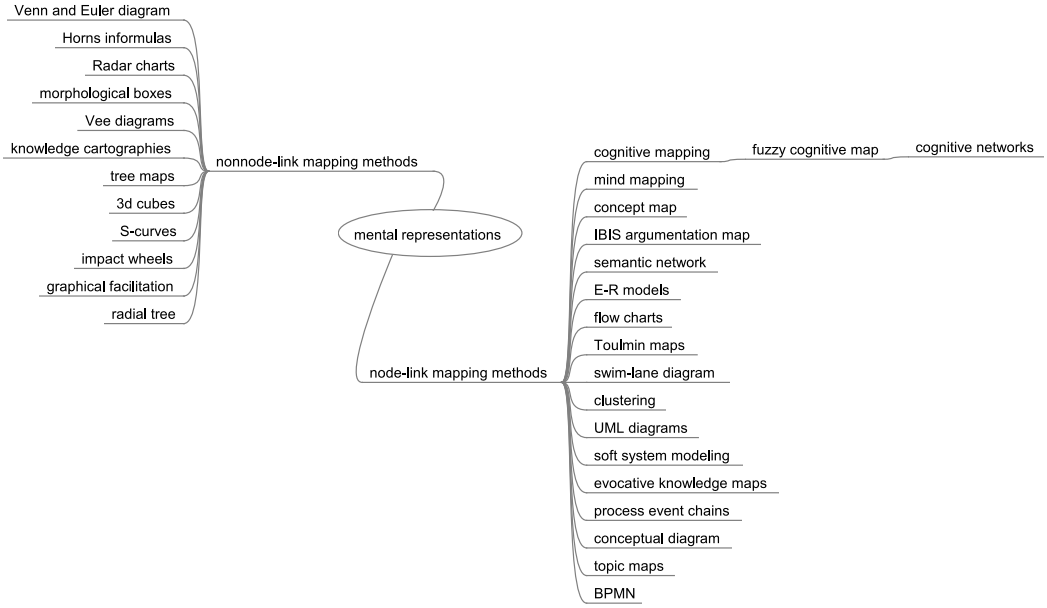
3.7 Iné techniky

Okrem vyššie spomínaných techník modelovania existuje veľké množstvo iných. Z nich najvýznamnejšie sú napríklad process event chains, evocative knowledge map, clustering diagram, flow charts, kognitívne siete, swimlane diagramy, E-R modely, Toulmin mapy, rovnako ak napríklad aj UML a BPMN. Všetky spomínané mentálne reprezentácie vychádzajú z faktu prepájania uzlov medzi elementami v diagrame. Okrem takýchto techník orientovaných na využívanie uzlov, existujú aj bezuzlové mentálne reprezentácie. K nim patria napríklad Vennové a Eulerové diagramy, Vee diagramy, stro-



Obr. 3.6: Príklad argumentačnej mapy.

mové mapy, radiálne stromy, S-curves diagramy, knowledge cartography a mnohé iné.



Obr. 3.7: Diagram rozdelenia vizualizačných techník.

Kapitola 4

Porovnanie vizualizačnej sily modelov

Z analyzovaných možností vybraných vizualizačných techník z oblasti konceptuálneho modelovania 3 boli ako najvhodnejšie pre využívanie zvolené myšlienkové mapy, kognitívne mapy a konceptové mapy. Vhodnosť ich využitia bola odôvodnená ich všeobecnou použiteľnosťou a skúsenosťami akejkoľvek skupiny ľudí. Pri výbere ktorejkoľvek osoby z takejto skupiny je vysoká pravdepodobnosť, že má daný jednotlivec skúsenosť s aspoň jednou z daných techník.

Z pohľadu analýzy a špecifikácie požiadaviek zákazníkov na výsledný produkt boli pre porovnanie vybrané dve vizualizačné metódy. Pre zjednodušenie boli v porovnávaní zvolené diagramy prípadov použitia a diagramy tried. Tieto UML diagramy a ich notácia bola porovnávaná voči notácií, ktorú poskytujú myšlienkové mapy 4.1, kognitívne mapy 4.2 a konceptové mapy 4.3. Základné porovnanie nad zvolenými diagramami UML bolo rozdelené na poznávanie rôznych elementov týchto modelovacích techník.

4.1 Porovnanie UML a myšlienkových máp

Pre myšlienkové mapy sa v ich základnej notácii objavuje len jeden typ elementu. Týmto elementom je téma alebo zjednodušene akákoľvek súvisiaca informácia alebo fakt. Tieto fakty sú navzájom prepájané začínajúc od jedného centrálneho námetu postupne do ďalších vetiev v stromovej štruktúre. Tento fakt podmieňuje, že aj medzi jednotlivými faktami existuje len jeden typ väzby – jednoduchá asociácia. V rozšírenej notácii sa k týmto základným elementom pridáva ešte väzba označovaná ako linka, ktorá prepája dva súvisiace fakty s rozdielnymi vetvami hierarchického usporiadania. Následne sa v notácii na jednotlivé elementy môžu používať rozdielne tvary, popisy, prípadne typy šípok medzi nimi. Okrem toho sa môžu využívať v základných elementoch vyjadrujúcich informáciu aj rôzne doplňujúce ikony, pripadne

môže byť obsahom takéhoto elementu obrázok alebo diagram iného typu. Pre zvýraznenie odlišnosti vetiev sa môže využívať rôznorodá farebnosť.

Keď sa budeme snažiť o porovnanie tejto notácie s možnosťami diagramu tried narazíme na viacero problémov. Samotnú triedu dokážeme reprezentovať v podobe elementu predstavujúceho informáciu (fakt). Následne, keď prejdeme k atribútom a metódam triedy, však už narážame na prvý problém. Atribúty a metódy tak musíme reprezentovať v podobe hierarchie danej v podobe: trieda – jej atribúty (metódy) – konkrétne názvy atribútov (metód). Ešte väčšiu mieru stromovej hierarchie musíme zachytiť, keď pre atribúty pridáme viditeľnosť. Dostaneme sa tak do situácie, v ktorej sa hierarchia rozšíri až na: trieda – atribúty – viditeľnosť – atribút. Pre modelovanie diagramov tried to však stále nie je všetko. Ďalšie rozšírenie hierarchie predstavuje zavedenie typov.

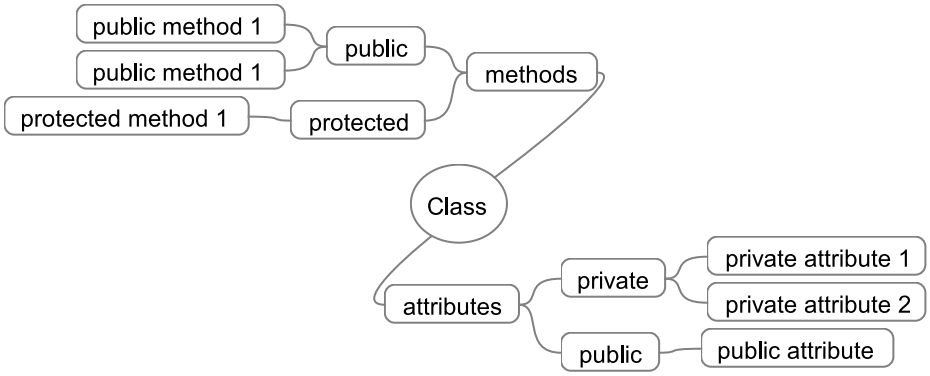
Okrem toho je však v diagramoch tried dôležité zobrazovať aj vzťahy medzi triedami. V diagramoch tried môžeme nájsť vzťahy ako je asociácia, generalizácia, realizácia, agregácia, kompozícia. V myšlienkových mapách sa však vrátane používania liniek dajú nájsť len dva typy väzieb. Tu vzniká najvýraznejší problém, ktorý je potrebné odstrániť podrobnejším návrhom. Príkladmi zamýšľaných stratégií môže byť zavedenie vrstiev a využívanie iných grafických značení štandardnej asociácie.

V prípade diagramu prípadov použitia narážame na podobné problémy. Zatiaľ čo základný artefakt v podobe samotného prípadu použitia dokážeme reprezentovať pomocou faktu, problémom sú rovnako rôzne typy väzieb. Hráčov (Actors), systém, prípadne aj kolaboráciu dokážeme rovnako vyjadriť vo forme faktu. Otázkou je však ako správne vyjadriť vzťahy (generalizácia, realizácia, závislosť, include, extend). Jedinou jasne mapovanou väzbou je opäť asociácia.

Týmto porovnaním sa ukázalo, že expresívnosť myšlienkových máp v základnom ponímaní nie je na synonymické modelovanie UML diagramov dostatočné a je potrebné ho rozširovať, čo je však jedna s výhodných vlastností myšlienkových máp.

4.2 Porovnanie UML a kognitívnych máp

Kognitívne mapy sa svojou notáciou veľmi nelíšia od myšlienkových máp. Základným rozdielom z pohľadu vizualizácie je používanie viacerých počiatočných faktov, zatiaľ čo pri myšlienkových mapách sa využíva len jeden centrálny fakt, okolo ktorého sa následne modeluje jeho okolie. Okrem toho sa v kongnitívnych mapách využívajú vo vyššej miere orientované asociácie, ktoré súvisia práve s viacerými centrálnymi prvkami. Okrem toho bývajú tieto asociácie ohodnocované, pričom ohodnocovanie býva reprezentované od najjednoduchších logických typov (Boolean, +/-) až po rôzne percentuálne ohodnotenia (prípadne reálne čísla) vplyvu previazaných faktov.



Obr. 4.1: Príklad modelovania triedy pomocou myšlienkovej mapy.

Problémy tak opäť ostávajú podobné ako pri myšlienkových mapách. Základné elementy, akými sú triedy v diagrame tried a prípady použitia v diagrame prípadov použitia je možné jednoducho reprezentovať. Problém ostáva v reprezentácii rôznorodých väzieb. Pri používaní orientovaných a ohodnotených hrán sa dajú takéto väzby lepšie využiť. Orientácia môže vyjadrovať napríklad vzťahy generalizácie vo význame kto od koho dedí a pod. Ohodnotenie zároveň môže určovať typ väzby. Všeobecne sa však pre väzby musí vytvárať opätovne nový systém prepisu, ktorý musí byť navrhovaný s prihliadnutím na všetky možné alternatívy, ktoré sa môžu pri modelovaní vyskytnúť.

4.3 Porovnanie UML a konceptových máp

V prípade konceptových máp je situácia veľmi podobná tej, ktorá platí pre myšlienkové mapy. Už z porovnávania týchto vizualizačných techník [10] je známa ich výrazná podobnosť. Významnejšou vlastnosťou, ktorá dokáže odlišovať tieto diagramy je najmä štruktúra zhora nadol, zatiaľ čo myšlienkové mapy sú koncipované od stredu. Pre konceptové mapy zároveň platí využívanie označených orientovaných asociácií, ktoré sa z istého pohľadu dajú porovnávať s ontológiami. Podobne ako pri kognitívnych mapách je tak možné uvažovať nad využívaním popisu takejto hrany pre napovedanie o type väzby. Rovnako platí, že základné elementy je jednoduché mapovať, problémom ostávajú len spomínané väzby.

Kapitola 5

Transformácie medzi modelmi

Transformácie je možné definovať ako premenu. Táto premena môže byť chápaná rozdielne vzhľadom na doménu, nad ktorou sa o transformáciách hovorí. Pre softvérové inžinierstvo a túto prácu je zaujímavý najmä význam, ktorý je transformáciám pripisovaný napríklad pri využití modelom riadeného vývoja softvéru. Pri modelom riadenom vývoji softvéru sa transformácie dajú chápať ako prostriedok napomáhajúci k prechádzaniu medzi modelmi s rôznou mierou abstrakcie. V základnom princípe modelom riadeného vývoja sú významné série transformácií medzi modelmi – model to model (M2M) transformácie a transformácie medzi abstraktnejším modelom a konkrétnejším modelom vo forme zdrojového kódu – model to text (M2T) transformácie.

Na ich využívanie v priebehu vývoja rôzneho softvéru poukázala práca [23]. Jej výsledky boli využité aj vo fáze prvotnej evaluácie zamýšľaného výskumu. Z pohľadu vývoja softvéru sa táto práca zameriavala najmä na fázy počínajúc od návrhu cez implementáciu až po testovanie. Modely boli vytvárané prostredníctvom UML, a to vďaka dostatočnej znalosti tohto modelovacieho jazyka. Pokiaľ sa však pozeráme na prvotnú fázu potenciálnej komunikácie so zákazníkom vo fáze analýzy a prvotného návrhu, je potrebné poznamenať, že skúsenosti s modelovacími technikami môžu, ako už bolo spomínané, spôsobiť nemalé problémy.

Preto sa táto práca sústreďuje práve na túto fázu, v ktorej sú predkladané možnosti využívania konceptuálneho modelovania. Modely vytvárané takýmto spôsobom však môžu byť pre ďalšie využitie problematické. Výhodné sú najmä pre netechnických ľudí, ktorí im dokážu lepšie pochopiť. Následne sa však už ako vhodnejšie javia štandardnejšie modelovacie techniky, ktoré sú dôležité pre pochopenie vytváraného softvéru a jeho častí pre odborníkov. Je preto dôležité sa zamýšľať nad možnosťami prepisov týchto modelov medzi sebou. Tieto prepisy sa pri premenách rozličných modelov

môžu pripodobňovať na transformácie využívané pri modelom riadenom vývoji.

Zatiaľ čo pri modelom riadenom vývoji sa pri transformáciách prechádza medzi modelmi s rozličnou mierou abstrakcie, v tomto prípade ide o prechádzanie medzi modelmi na veľmi úzko súvisiacich úrovniach abstrakcie. Vytváranie takýchto transformácií je významné najmä z pohľadu expresívnej sily modelov. Tie nie sú na rovnakej úrovni, pretože ich notácie neobsahujú rovnaké množstvo elementov a nemôžu tak pre všetky príklady vytvárať jasne definované synonymické dvojice. V niektorých prípadoch je problémom nájsť jeden význam, naopak v iných je možné ku konkrétnemu elementu jedného modelu priradiť viacero elementov v inom modele. Tieto fakty tvoria základný problém pri snahách o definovanie transformačných pravidiel medzi takýmito modelmi.

Potreba hľadania možností odstránenia takýchto problémov bola skúmaná už v iných prácach zaoberajúcich sa modelmi vytváranými nad jazykmi s rozdielnom expresívnosťou [19]. V takýchto prípadoch je možné hľadať zväčša len určitú uspokojivú mieru úspešných transformácií. Následne sa snažiť pre automatizované transformácie definovať nové a nové pravidlá, ktoré zabezpečia zvýšenie miery dosahovaného úspechu. V tejto práci sa však pracuje nad pomerne jednoduchými modelovacími jazykmi.

Pre analyzované ontológie rovnako často vznikajú potreby transformácií do iných jazykov, najmä do UML. Vzhľadom na to, že veľké množstvo ontológií je rovnako na nižšom expresívnom leveli ako UML, stávajú sa aj tieto transformácie problémovými [33] [32]. Pri hľadaní možností vyjadrenia elementov jedného modelu v inom je potrebné hľadať porovnanie využívaných elementov v daných modelovacích technikách. V týchto spomenutých nadväzujúcich prácach boli navrhované možnosti transformácií medzi UML a vysoko využívaným jazykom pre ontológie OWL. Nájdené rozdiely v takýchto jazykoch je potrebné pokryť alternatívnym vyjadrením elementu, čo nemusí byť vždy jednoduché.

Pre transformácie je potrebné definovať základné vlastnosti, ktoré by mala pokrývať kvalitná transformácia. Pre vytváranie transformácií je možné definovať techniky, akými by mali byť vytvárané. Niektoré zo známych best practises pre tvorbu transformácii môžeme nájsť v [21]. K nim patrí vytváranie transformácií modulárne, čiže tak, aby bolo možné transformácie skladať do kompozícií v závislosti od konkrétneho využitia. Ďalšími dôležitými vlastnosťami sú vlastnosti validácie a verifikácie – na overenie, že sa pri transformácii vytvára správna reprezentácia zdrojového modelu a na overenie správne zachytených údajov, ktoré boli prenesené z pôvodného modelu.

Takýmto vlastnostiam sa venuje aj práca [26], v ktorej sú zachytávané tieto vlastnosti aj z implementačnej stránky. Modularita by mala napríklad zabezpečiť možnosť vytvárať agregované transformácie, prípadne vytvárať nové transformácie špecializáciou všeobecnejšej transformácie tak, ako je to pri dedení. Návrh transformácie musí pri tom zaručovať možnosti znovu-

použiteľnosti transformácie. Transformácie by mali spĺňať aj mnohé ďalšie vlastnosti. Z nich práve znovupoužiteľnosť musí zabezpečiť možnosť využívať transformáciu na akýkoľvek model. Transformácia by tak mala byť vytváraná nad princípom podobným ako sú WORA (Write Once Run Anywhere)a WOCA (Write Once Compile Anywhere). Transformácia by mala byť vytváraná raz a využívaná nad akýmkoľvek modelom dlhodobo a bez nutnosti výrazných zmien.

K vlastnostiam samozrejme patrí aj pravidlo jednoznačnosti – jedna transformácia aplikovaná nad jedným modelom viackrát, musí zakaždým vrátiť rovnaký výsledok. Táto vlastnosť musí platiť rovnako aj pre spätné transformácie modelov, pri ktorých sa však nemusíme dostávať nutne pri postupe tam a späť k pôvodnému modelu. Takáto transformácia môže poslúžiť ako vhodný spôsob obohatenia modelov v miestach, ktoré sa ukážu ako vhodné na obohatenie počas transformácie. Vlastnosť opakovateľnosti však môže vyžadovať zachovávanie všetkých súčastí modelu pri transformáciách tam aj späť, čo môže okrem vylúčenia strát prípadne znamenať aj vylúčenie pridaných prvkov. Záleží na špecifikácii transformácie a jej využití. Pre transformáciu musia platiť aj vlastnosti jednoduchosti, ktorá sa musí prejaviť v tom, že jej vytvorenie nesmie byť zložitejšie ako iný spôsob premeny modelu do iného a musí sa skončiť vo vhodnom časovom úseku.

Kapitola 6

Využitie konceptuálneho modelovania

Pre konceptuálne modelovanie je v oblasti softvérového inžinierstva množstvo oblastí, v ktorých je možné hľadať ich uplatnenie. Pre špecifikáciu požiadaviek zákazníkov je možné využívať ich lepšiu zrozumiteľnosť ako modelov, ktoré nevyžadujú ďalšie odborné vzdelanie a skúsenosti s modelovaním, prípadne sú natoľko intuitívne, že k úplnému pochopeniu modelovania prostredníctvom nich je potrebné minimálne úsilie.

Vo fázach analýzy a návrhu softvéru často dochádza k potrebe analyzovať a navrhovať rovnaké, alebo výrazne podobné súčasti systému. Tie by však pri vhodnom zachytávaní bolo možné znovupoužiť. Znovupoužiteľnosť softvérových artefaktov je často využívanou stratégiou pri vývoji modulárnych častí softvéru. Najvýraznejšie sa členenie do menších častí prejavuje pri vývoji softvéru vytváraním radu softvérových produktov. Pri tejto softvérovej paradigme sa modularizované súčasti softvéru môžu jednoducho prenášať medzi jednotlivými softvérovými produktmi a ich spájaním je možné s malým pričinením vytvárať veľké množstvo rôznych kombinácií, ktoré budú vytvárané na základe požiadaviek na takéto finálne produkty.

Okrem vlastných softvérových artefaktov sa pri vývoji môžu využívať rôzne zdroje pre reprezentáciu dostupných dát v systéme. Na získavanie týchto dát je možné využiť rovnako predchádzajúce podobné produkty priamo pochádzajúce z vývoja rovnakého vývojára alebo iného vývojára, ktorý takéto dáta poskytne verejne. Zároveň je takéto informácie možné získavať aj z rôznych všeobecných verejných zdrojov, ku ktorým patria verejne dostupné elektronické encyklopédie a iné zdroje. Vo výskumnej práci [24] bola ukázaná možnosť využívania takýchto zdrojov pre modelovanie využitím konceptuálneho modelovania. Vďaka využívaniu týchto zdrojov sa podarilo dokázať prínos a efektivitu takéhoto prístupu.

Myšlienkové mapy sa za posledné obdobie stali významnou súčasťou prvotných fáz vývoja softvéru najmä v prostredí agilného vývoja. Primárna

použiteľnosť týchto modelov dáva predpoklady, že táto technika bude výhodnou možnosťou ako rýchlo, napríklad pri brainstormingu, zaznamenávať požiadavky na výsledný systém. Práce, ako je napríklad [18], sú toho jasným dôkazom. Na možnosti využívania modelovania pomocou myšlienkových máp sa zameriava v snahe ukázať pozitívny vplyv na rozvoj produktového backlogu. Práve netechnický používatelia v podobe cieľových zákazníkov, ktorí sa podieľajú na analýze majú v tejto fáze zjednodušené modelovanie. Výsledky takýchto prác dokazujú, že možnosti podobných modelovacích techník majú veľký potenciál práve v nových prístupoch ako je extrémne programovanie, lean a agilné prístupy.

Konceptuálne modelovacie techniky využívajúce myšlienkové mapy majú v prostredí scrumu potenciál vplývať aj na význam používateľských príbehov v šprintoch. Dobre mierené otázky, ako tie, ktoré boli zavedené v reporte [15] je možné využiť na premieňanie požiadaviek do uceleného grafického tvaru. Ten sa zároveň môže stať podkladom pre vytváranie UML diagramov z neho. Konverzia takýchto modelov do UML je užitočná práve pre doménového špecialistu. Pre prístup v spomenutých výskumoch sa dá hovoriť o rozdelení práce s požiadavkami na dve fázy: zbieranie požiadaviek využitím myšlienkových máp a modelovanie požiadaviek už v UML.

O zúženie rozdielov medzi špecialistami z business domény a softvérového inžinierstva sa snaží aj práca [28], resp. [7]. Tie sú zapisované kategorizovaním rôznych častí do myšlienkových máp. Môžu byť využité na modelovanie statickej časti, napríklad v podobe potrebných dát, ktoré budú ukladané v systéme, rovnako ako aj v dynamickej časti v podobe nápomocnej pre určenie používateľských príbehov v agilnom vývoji alebo úzko súvisiacimi prípadmi použitia.

Pri návrhu softvéru je dôležité odhaľovať potenciálne miesta, ktoré by bolo vhodné prefaktorovať, alebo pre existujúci systém refaktorovať. To sa často prejaví v podobe transformácie do niektorého architektonického vzoru. Hľadať také súvislosti, ktoré napomôžu k rýchlejšiemu odhaľovaniu pachov ako nevhodných súčastí modelov je ďalšou z možností, ktoré ponúka zavedenie myšlienkových máp. Tie kvôli svojej odlišnej reprezentácii dokážu zachytávať špecifické časti modelov v inej forme ako je tá, ktorá je typická pre UML. Práve takáto nová reprezentácia môže napomôcť k odhaľovaniu skrytých súčastí v modeloch. Tie nemusia byť vždy jasne viditeľné, avšak po transformácii do inej vhodnej reprezentácie môže zvýrazniť podobnosti so všeobecne používanými vzormi využívanými v návrhu softvéru.

Kapitola 7

Evaluácia schopností konceptuálneho modelovania

Na možnosti využívania konceptuálneho modelovania pre zachytávanie požiadaviek zákazníkov je možné poukazovať z rôzneho pohľadu. Pri potvrdzovaní spôsobilosti konceptuálneho modelovania sa prihliada najmä na fakt, že mnohé z týchto techník sú verejne známe, ľahko pochopiteľné a prispôsobiteľné pre konkrétne využitie. Vychádza sa z toho, že s daným modelovaním (najmä myšlienkovými mapami) sa zákazník s takmer určitosťou stretol. Nezáleží pri tom, či išlo len o využitie na akademické účely alebo aj iné využitie priamo v business sfére.

Spoločnosť Astah, ktorá sa dlhodobo venuje vývoju modelovacích nástrojov pre softvérové inžinierstvo do svojich produktov pridala možnosti využívania myšlienkových máp. Tie sú v tomto prípade použité ako vhodný spôsob rýchleho zachytávanie požiadaviek zákazníka počas stretnutia, pri ktorom sa medzi zákazníkom a zástupcom producenta softvéru ukladajú tieto požiadavky do grafickej, ľahko zrozumiteľnej podoby. Vhodnými otázkami sa dá docieliť nájdenie dôležitých častí pre ďalšie smerovanie návrhu produktu, rovnako ako aj smerovanie k hľadaniu vhodných otázok pre ďalší návrh. K dôležitým otázkam tohto typu patria otázky spadajúce do okruhu Five Ws. Tie patria k základným otázkam pri zbere informácií.

Odpovede na jednotlivé otázky z nich dokážu výrazne napovedať o celom navrhovanom produkte. Z pohľadu use casov je možné veľa získať z otázok „kto?“ a „kedy“. Otázka „kto?“ pomáha odkryť všetky typy Actorov vo výslednom systéme, zatiaľ čo otázka „kedy?“ určuje samotné potenciálne prípady použitia a ich názvy. Počas komunikácie sa môžu niektoré z nich ukázať len ako kroky iného prípadu použitia, prípadne ako špecializované typy, či možné súčasti iného prípadu použitia. Otázka „čo?“ zasa môže napovedať, čo bude dôležité pre zachytávanie v modele, alebo čo bude dôležité zachytiť v ukladaných dátach napríklad v navrhnutej databáze.

Otázka „kde?“ môže zasa napovedať o miestach, kde sa bude produkt

nachádzať, alebo ako sa bude k produktu pristupovať. To môže odhaliť potenciálne dôležité vlastnosti produktu. Ako príklad je možné uviesť webovú aplikáciu, ktorej používatelia s ňou budú pracovať kdekoľvek, kde majú prístup na internet. Odhalí sa tak požiadavka na responzívnosť navrhnutého dizajnu aplikácie. Podobne by bolo možné hľadať ďalšie vhodné využitie tejto otázky. Otázka „prečo?“ určuje hlavnú motiváciu pre vznik projektu. Odhaľujú očakávania zákazníka a to, čo sa má zmeniť k lepšiemu po zavedení produktu do praxe. Poukazuje tak na najdôležitejšie vlastnosti, ktoré finálny produkt musí spĺňať. Príkladom môže byť uľahčenie práce, skrátenie času, sktrátenie čakania, efektivita práce, jednoduchosť a pod.

V ukážke takéhoto postupu zachytávania požiadaviek [3] sa využívajú 4 tieto otázky. Astah sa pri tom snažil ukázať zachytenie spomínaných položiek, ktoré je možné ďalej využiť v ďalšom softvérovom návrhu.

Z iného pohľadu sa na využitie myšlienkových máp dá pozerať pri využívaní prístupu modelom riadeného vývoja softvéru z práce [23], v ktorej boli modelované štruktúry a prechody medzi obrazovkami mobilnej aplikácie. Použitím sady transformácií sa z platformovo nezávislého modelu v UML môže automatizovane prejsť až do finálneho modelu v podobe platformovo-špecifického modelu v zdrojovom kóde natívneho jazyka mobilnej platformy. Využívanie UML však naďalej vo vysokej miere limituje fázy analýzy a návrhu na striktnú nutnosť prítomnosti odborníka z domény vývoja softvéru. Pri použití inej modelovacej techniky ako je UML je však možné odstrániť počiatočnú nevýhodu neznalosti doménových modelovacích jazykov so strany zákazníka. Ten by mal byť v najlepšom prípade schopný vytvoriť model v takom tvare, aby bolo možné na neho uplatniť podobný typ transformácií, aký bol použitý na dané UML modely. Výsledkom by tak mohol byť vygenerovaný zdrojový kód s plnohodnotným využitím vo finálnej aplikácií. V navrhnutom prístupe sa obmedzovalo využitie len na používateľské rozhranie, celkovo sa však dá uvažovať aj nad inými dôležitými časťami aplikácie. Okrem dôležitej súčasti v podobe doménovo-špecifických jazykov sa tak dostávame k možnosti využitia konceptuálnych modelovacích techník aj ako prostriedku doménovo-špecifického modelovacieho jazyka, ktorý by mohol nájsť uplatnenie aj v prostredí modelom riadeného vývoja softvéru. Prináša tak možnosť riešenia dôležitej skutočnosti, ktorá sa v súvislosti s modelom riadeným vývojom spomína stále častejšie, a to je potreba doménovo-špecifických jazykov pre ľudí, ktorí nepatria k softvérovým odborníkom.

Pre ukážku bol vybratý podobný príklad aplikácie ako vo vyššie spomínanej práci. Plné zadanie príkladu je možné nájsť v prílohe A. Zadanie bolo smerované k vzorke ľudí s rôznym vekom, rôznym vzdelaním a ďalšími rôznymi sociálnymi aspektmi. Výsledky je možné nájsť v prílohe B. Z výsledkov vyplýva, že rôzni ľudia zachytávajú výsledný produkt rozdielne. Niektoré časti sú však zachytené vo všetkých modeloch, resp. sa v nich nachádzajú synonymické označenia. Z experimentu vychádza, že ľudia bez ohľadu na to, či sú alebo nie sú odborníci v softvérovom inžinierstve,

zachytávajú dôležité súčasti rovnako. Najpodstatnejšie rozdiely, ktoré sa v modeloch nachádzajú súvisia najmä so spôsobom zadania úlohy, ktorá dovolila prejaviť vlastnú kreativitu, prípadne zachytila obľúbené používateľské zvyky testovanej vzorky na mobilných zariadeniach (napr. vysúvacie menu zo strany).

Iné využívanie konceptuálnych modelov v prostredí informačných technológií je možné vidieť napríklad v prostredí agilného vývoja softvéru v rôznych firmách. Myšlienkové mapy nachádzajú svoje uplatnenie pri brainstormingu, odhaľovaní dôležitých požiadaviek zapracovávaných do príbehov šprintu, pri odhadoch ohodnotenia úloh a pod.

Kapitola 8

Spracovanie prirodzeného jazyka

Vychádzajúc z analýzy konceptuálnych modelovacích techník 3 poznáme, aké súčasti sa používajú pri ich tvorbe. Ide o rôzne dlhé termíny a pojmy, ktoré sú z pohľadu daného modelovania dôležité. Pokiaľ sa ich snažíme nejakým spôsobom transformovať 5, často môžeme naraziť na problém rozpoznávania ich kľúčových častí. Takéto časti môžu byť vyjadrené v úplne jednoduchom priamom porovnaní reťazca s hľadaným reťazcom (slovom), avšak môžu byť definované aj oveľa náročnejšie. V takomto prípade sa nesnažíme vyhľadávať priamo s určtosťou sa vyskytujúci pojem, ale len určitý pattern v nich. Takýto vzor môže byť reprezentovaný v podobe konkrétneho slovného druhu, či slovného člena vo vete či polovetnej konštrukcii. Preto je pre nami špecifikované transformácie zaujímavý aj výskum v oblasti spracovávania prirodzeného jazyka.

Spracovanie prirodzeného jazyka je pomerne široká oblasť informatiky a umelej inteligencie. Výskum v tejto oblasti za posledné roky výrazne pokročil. Vďaka tomuto faktu je možné zamýšľať sa nad niektorými možnými použitiami v zamýšľanom prístupe pre modelovanie a transformácie myšlienkových máp. V tejto časti sa preto zameriame len na tie fakty, ktoré sú z pohľadu myšlienkových máp zaujímavé.

8.1 Lexikálny kontext myšlienkových máp

Myšlienkové mapy majú ako centrálne pojmy uvádzané zväčša krátke pomenovania. Často ide o jednoduché pomenovania, prípadne krátke vyjadrenia podstaty. Medzi použitými slovami nájdeme najmä podstatné mená vyjadrujúce konkrétny subjekt, ktorý je opisovaný. Často pritom takýto subjekt rozširuje prívlastok, teda určité prídavné meno. Okrem nich je možné v mapách často zachytávať aj konkrétne akcie. Tie sa definujú prostredníctvom slovies, prípadne slovesných podstatných mien. V ich prípade býva pojem

dlhší a doplnený viacerými faktami, ktoré bližšie popisujú danú akciu. Z detailnosti popisu musí byť čitateľovi dostatočne jasné, o čom je daný prvok mapy. Nemal by však obsahovať zbytočne dlhý opis. Na to je vhodné použiť v myšlienkovej mape poznámku.

Ak sa v mape nachádza viacero výrazne súvisiacich pojmov, ktoré majú spoločný základ a odlišujú sa len v doplnenom opise, napríklad vo forme rôznych prívlastkov, iných predmetov a pod., mali by byť spoločné črty vložené do jedného spoločného pojmu a následne odvodzovať ich podtémy. Vychádzajúc z týchto jednoduchých pravidiel, ktorých by sa mal držať tvorca myšlienkovej mapy, môžeme vyvodiť určité závery o obsahu jednotlivých pojmov mapy. Ide o krátky a výstižný obsah, ktorý je výrazne jednoduchší než celé vety a popisy v ekvivalentnej textovej podobe. Tento fakt je možné s výhodou využiť pre jeho spracovanie.

8.2 Možnosti uplatnenia v zamýšľanom prístupe

Pre zamýšľaný prístup modelovania pomocou myšlienkových máp môže byť zaujímavé práve porovnávanie slovných druhov. Pri konverzii do UML sa musíme pridržiavať určitých pravidiel, podľa čoho budú jednotlivé pojmy mapované do tried, atribútov, metód, prípadov použitia a pod. Tu je možné nájsť zaujímavú paralelu so spomínaným obsahom mapovaným v myšlienkových mapách – krátke jednoznačné pomenovania, na základe ktorých vieme, o čo sa jedná. Rovnaké pravidlo môžeme nájsť aj v pomenovávaní spomenutých elementov v UML diagramoch. Vďaka tomu môžeme uviesť predpoklad, že v závislosti na tvare vstupného pojmu myšlienkovej mapy, môžeme určiť o aký prvok UML v mapovaní pravdepodobne pôjde. Mieru pravdepodobnosti pritom môžeme koncipovať na základe presnosti transformačných pravidiel možných vyskytujúcich sa scenárov.

8.3 Rozpoznávanie slovných druhov

Ako sme spomenuli, v myšlienkových mapách, rovnako tak aj v UML sa nachádzajú krátke a výstižné frázy vyjadrujúce podstatu daných pojmov, resp. daných elementov. Pre automatizovaný prepis z myšlienkovej mapy do UML bude dôležité určenie slovného druhu daného pojmu, resp. jeho základu. V spracovaní prirodzeného jazyka je možné odhaliť výskumy, ktoré sa zaoberali takýmto určovaním. Ak ako skúmaný jazyk použijeme angličtinu, je možné nájsť široké spektrum výskumov v problémovej oblasti.

Výskumu v oblasti rozpoznávania slovných druhov sa venuje napríklad The Stanford Natural Language Processing Group, ktorá ponúka na svojich stránkach [1] aj funkčné demo, ktoré dokáže v angličtine rozpoznávať základné slovné druhy. Tomuto výskumu sa zaoberajú v rámci podskupiny venujúcej sa POS taggovania (Parts of Speech tagging). Práve táto podob-

lasť spracovania prirodzeného jazyka je pre nás zaujímavá. Inou alternatívou pre túto úlohu môže byť aj WordNet [2]. Ten obsahuje databázu slov angličtiny a z pohľadu možností je jednoduchší, avšak úlohy, ktoré dokáže plniť ho rovnako predurčujú na experiment v navrhovanom prístupe, resp. vytváranom nástroji.

Na rozpoznávanie slovných druhov vo vete, či polovetnej konštrukcii môžeme využívať niekoľko prístupov. Základným prístupom môže byť rozpoznávanie využitím existujúceho slovníka pojmov [30]. V takomto prípade sa však dostávame k problému, ako správne takýto slovník definovať. Existujúce princípy na rozširovanie takýchto slovníkov sú založené najmä na strojovom učení. Iným princípom, ktorý nám môže efektívne umožniť dopĺňať takýto slovník je využívanie existujúcich znalostí z predchádzajúcich projektov. Takýto prístup navodzuje k využívaniu produktových línií vo fáze analýzy a návrhu. K iným prístupom patria pokročilejšie metódy rozpoznávania, založené na pravidlových systémoch a stochaickom rozpoznávaní. Pre takéto porovnávanie sa môže využívať entropia výskytu slov v jednotlivých možných typoch slovných druhov [31]. Vo väčšine takýchto prístupov je však potrebné prechádzanie priestoru možných kombinácií, pri ktorom sa využívajú pokročilé metódy najmä z oblasti strojového učenia (napr. neurónové siete) a evolučné algoritmy. Pre naše použitie sa ukazujú ako vhodné aj samotné slovníkové definície, vzhľadom na často opakujúce sa termíny v rôznych projektoch a jednoduché definície s krátkymi neplnohodnými vetami.

Okrem spomínaných výskumov sa samotnému spracovávaniu prirodzeného jazyka na poli analýzy špecifikácie softvérových požiadaviek týmto metódam venovala napríklad práca [12]. Z jej cieľov a výsledkov je zjavné, že aplikácia spracovávania prirodzeného jazyka pre analýzu požiadaviek a ich klasifikáciu má potenciál. Výskum pracuje s možnosťami transformácií týchto informácií do formálnych modelov. Výsledky tohto výskumu potvrdzujú, že zamýšľané postupy pre odhaľovanie jednotlivých súčastí v textovej forme požiadaviek je možné skutočne využiť. V prípade myšlienkových máp môžeme takéto použitie preklasifikovať na špeciálnu podmnožinu požiadaviek zachytených v prirodzenom jazyku, ktorá obsahuje navyše ešte vzťahy, a tak môže byť pre takéto spracovanie ešte vhodnejšia.

Ďalším príkladom venujúcim sa spracovaniu prirodzeného jazyka v oblasti spracovávania požiadaviek na softvérový produkt je [14]. V tomto prípade je rovnako potvrdené, že spracovávanie jazyka môže byť užitočné v ďalšej fáze a môže sa využiť na automatické transformácie do UML. V práci bol využívaný spomínaný parser zo Stanfrodskej univerzity a vytvorený prístup dokáže transformovať vety do behaviorálnych diagramov v UML (diagramy aktivít a diagramy sekvencií). Z pohľadu analýzy slovných druhov v takýchto vetách ide o náročnejší problém, ako nami študovaný problém krátkych výstižných pojmov v myšlienkových mapách.

Výskumy, ktoré sme analyzovali poukazujú na možnosti využitia spra-

covania prirodzeného jazyka. Niektoré z nich dokonca potvrdzujú, že myšlienky využívania takýchto techník v spracovávaní softvérových požiadaviek nie sú ničím novým a venovali sa im už viacerí autori. Experimentovanie nad takýmto prístupom nám otvára nové možnosti využívania rôznorodých myšlienkových máp a môže výrazne pomôcť k vyriešeniu problémov týkajúcich sa získavania požiadaviek bez prítomnosti experta pri ich konzultácii.

Kapitola 9

Pohľady v myšlienkovej mape

V špecifikovaných modelovacích technikách v častiach 2 3 sa často využíva na vizualizáciu celku viacero nezávislých, alebo nejako koncepčne prepojených nezávislých modelov, ktoré dotvárajú lepší pohľad na modelovanú realitu. V štandardných modelovacích technikách prezentovaných v časti 2 sa takéto pohľady bežne využívajú. V niektorých konceptuálnych technikách z časti 3 však ich využívanie nemusí byť vždy potrebné, prípadne sa nevyužíva, pretože jeho uplatnenie nie je správne špecifikované. Na nami prezentovaný problém modelovania požiadaviek 6 sa však takéto niečo ukazuje ako nová príležitosť pre zvýšenie obsahovej celistvosti.

Z domény softvérového inžinierstva je známe používanie rôznych architektonických pohľadov na výsledný produkt. V diagramoch reprezentujúcich jednotlivé pohľady je možné nájsť súvisiace komponenty systému. Tak je možné nájsť určité komponenty vo viacerých rôznych pohľadoch. V myšlienkovej mape, ktorá je často určená na modelovanie informácií, však niečo takéto priamo nie je definované.

Z vlastností informácií vieme, že ich význam sa môže líšiť podľa konkrétneho kontextu. Platí to pre rôzne formy slov. Príkladom môžu byť homonymá. Jedna informácia (slovo) môže byť modelovaná v rôznych myšlienkových mapách a mať v okolí celkom rozdielne súvisiace informácie. Homonymia v lingvistike je jedným z najčastejších prípadov nejednoznačnosti, a práve táto nejednoznačnosť umožňuje modelovať v okolí informácie rôzne pojmy. Namodelovanie takéhoto typu miešaných informácii však vyvoláva zmätok a preto je potrebné ich modelovať samostatne.

Neplatí to už ale pre ľudské myslenie, v ktorom sú takéto pojmy v určitej blízkosti práve z dôvodu potreby spoľahlivého odlišovania medzi nimi a následného vyjadrovania súvisiaceho kontextu. Preto môžeme uvažovať, že pri myslení v pojmoch človek vytvára siete, ktoré je v základnom tvare možné zovšeobecniť do tvaru myšlienkovej mapy. Práve tento fakt využil aj originálny autor myšlienkových máp Tony Buzan. V základnom tvare sa však nejednoznačnosť slov modeluje samostatne. Prenesenie kontextu na iný

význam pojmu je tak vnímané ako presun na zodpovedajúcu mapu, ktorá obsahuje daný pojem v danom kontexte.

Vyhľadanie takéhoto kontextu je otázkou ľudského myslenia, ktorému sa daný iný kontext darí vyhľadať v závislosti na znalostiach daného človeka. Prenášaním pohľadu môžeme vytvárať vrstvy máp, ktoré sú navzájom previazané. Postupným prenášaním kontextu jedného pojmu v homonymii sa môžeme dostať opäť k iným slovám a od nich zasa k ďalším. Takýto druh myslenia sa s výhodou dá využiť pri učení.

Jednou z vlastností myšlienkových máp je okrem iného aj jej acyklickosť. Mapa vždy začína od nejakého centrálneho pojmu a pokračuje k jej súvisiacim pojmom a od nich k ďalším. Zostáva však problém s tým, ak je nejaký pojem v súvislosti s iným v inej vetve. Určité previazanie tohto vzťahu môžeme dosiahnuť pomocou linkovania. Takýto postup však nie je úplne ekvivalentný priamemu prepojeniu. Riešenie je preto opäť vrstvenie. Vďaka vrstveniu môžeme namodelovať určitú časť mapy tak, aby spĺňala základné pravidlá pre konštrukciu mapy a zároveň obsahovala všetko čo chceme. Takéto mapovanie je rovnako dôležité a vyjadruje potenciál, ktorý je obsiahnutý v nekonečnosti myslenia v súvislostiach. Mapou pojmov v myslení môžeme postupne prechádzať všetky uchovávané pojmy. Pri častom cyklení dochádza k nadobudnutiu presvedčenia, že daný pojem si nepamätáme.

Toto je časté práve v školskom prostredí, pri ktorom na myslenie v súvislostiach máme len limitovaný čas. Nie vždy sa tak musí nutne stať, že daný pojem vôbec nemáme v pamäti, avšak nedokážeme ho v komplikovanej sieti dostatočne rýchlo nájsť. Takýto stav sa prejavuje neskorším spomenutím si na riešenie po opadnutí stresu, keď sa predchádzanie siete darí úspešnejšie.

9.1 Zavedenie pohľadov

V myšlienkovej mape sa spomínané pohľady nachádzajú, avšak ich vzájomné mapovanie nie je časté a nenachádza sa ani v najznámejších mapovacích nástrojoch na myšlienkové mapy. Dôvodom môže byť moderovanie. Takéto moderovanie môžeme považovať za vedenie medzi vrstvami konkrétnou osobou, ktorá dané mapy vytvorila a vysvetlila nám ich. Vďaka tomu je možné, že človek, ktorý určité podmnožiny daných pojmov v daných súvislostiach pozná, dokáže rozlišovať. V prípade strojového prechádzania máp sa nám to však nemusí podariť. Je to práve výsledkom nejednoznačnosti. Tá môže spôsobiť nesprávne uvažovanie v súvislostiach. Z tohto dôvodu by sa stroj musel dané súvislosti učiť pomocou nejakých metód strojového učenia. Problémom je však obrovské množstvo informácii v prírode, kvôli ktorým sa doteraz nepodarilo vytvoriť spoľahlivý model ľudského uvažovania.

Ak teda chceme vysvetliť súvislosti rovnocenných pojmov v inom kontexte pre strojové mapovanie, musíme vytvoriť priame mapovanie takýchto pojmov. Je teda potrebné určitým spôsobom vysvetliť, že dané pojmy sú na-

ozaj tým istým, len ohraničeným v iných súvislostiach. Pojem pri tom môže mať aj iný popis (napr. synonymické vyjadrenie). Z tohto dôvodu, rovnako ako aj so spomínaných homonymických nejednoznačností takéto prepojenie vytvárame pomocou iného atribútu, ktorým môže byť určitý univerzálny identifikátor.

9.2 Využitie pohľadov v zamýšľanom prístupe

Pri modelovaní požiadaviek na systém sa môžeme ocitnúť v situácii, keď chceme podčiarknuť niektoré vzťahy mapy. Príkladom môže byť jednoduchá myšlienková mapa rozdelená do špecifikovaných častí podľa kľúčových otázok. V danom tvare vieme, že sa v systéme budú nachádzať určité triedy, hráči systému a prípady použitia. Pre presnosť a použiteľnosť modelov však takéto roztriedenie nemusí stačiť. Často je potrebné definovať aj to, ktoré prípady použitia budú vyvolávané ktorými hráčmi, a ktoré triedy (resp. objekty daných tried) budú využívať. To však zjavne vytvára cykly v takejto mape. Namiesto porušenia pravidiel pre myšlienkové mapy tak vytvárame nové vrstvy, ktoré budú obsahovať práve tieto časti, ktoré sú v cyklickom vzťahu. Pre lepšiu vizualizáciu následne môžeme zobraziť aj rôzne veľké okolie z pôvodnej vrstvy, ktoré bude obsahovať pojmy z pôvodnej vrstvy.

Kapitola 10

Matematické modely podobnosti mapových štruktúr

Už z prvotnej evaluácie 7 uvažovaného prístupu sme dospeli k výsledkom, že rôzni ľudia mapujú požiadavky istým spôsobom rozdielne. Avšak aj napriek tejto rozdielnosti je možné nachádzať istú podobnosť. Ak chceme ukázať, že je takáto podobnosť v modeloch naozaj prítomná, potom musíme evaluovať ich podobnosť. Systém manuálneho porovnávania takýchto modelov sa stáva náročný, priam až nemožný s narastajúcou zložitosťou definovanej myšlienkovej mapy. Preto je potrebné zaviesť určité matematické modely, ako vypočítavať ich podobnosť. Vďaka takýmto modelom je následne možné vytvárať nástroje, ktoré porovnanie vykonajú automaticky.

10.1 Metriky podobnosti

Mapové štruktúry myšlienkových máp je možné navzájom porovnávať niekoľkými porovnávacími algoritmami. Ako základ treba zobrať predpoklad, že myšlienková mapa ako taká reprezentuje určitý graf, resp. určitú sieť. Vďaka tomuto faktu je možné ich porovnávať technikami, ktoré sú určené na siete.

Najzákladnejším výpočtovým modelom pre podobnosť myšlienkových máp môže byť absolútna podobnosť jednotlivých vyskytujúcich sa pojmov. Tú môžeme vyjadriť ako:

$$similarity = \frac{MatchN}{AllN + ExtraN}$$

Teda ako pomer pojmov, ktoré sa nachádzajú v skúmanej a očakávanej mape a počtu všetkých pojmov v očakávanej a počtu pojmov, ktoré sú v

skúmanej mape a nie sú v očakávanej. Takéto porovnanie nám dáva výsledky, ktoré nie sú závislé na samotnom tvare danej mapy. To znamená, že aj mapy, ktoré sú namodelované s nezmyselne usporiadanými pojmami, ale obsahujú ich, budú hodnotené dobre. Takéto porovnanie je teda len porovnaním slovníkov daných máp, bez nejakých konkrétnych súvislostí medzi nimi. Takáto metrika preto môže pri porovnávaní vracať výrazne veľkú chybu.

Pre vyhnutie sa tomuto problému je potrebné zaviesť do metriky také postupy, ktoré budú kvantifikovať podobnosť aj na úrovni tvarov. Z existujúcich prác [13] [16] [17], je možné usúdiť, aké možnosti máme v myšlienkových mapách.

Základnou uvádzanou metrikou v práci [13] je metrika granulárnej podobnosti, ktorá pracuje s hranami a vrcholmi. Je definovaná nasledovne:

$$GranularSimilarity = \frac{MatchE}{MissN + ExtraE + MatchE}$$

Kde $MatchE$ je počet hrán, ktoré sa nachádzajú v skúmanej aj očakávanej mape, $ExtraE$ sú tie hrany, ktoré sa nenachádzajú v očakávanej mape, ale sú prítomné v skúmanej a $MissN$ sú tie vrcholy, ktoré chýbajú v skúmanej mape (nachádzajú sa však v očakávanej). Daná metrika zlepšuje výsledky, pretože pracuje už s dvojicami vrcholov, resp. hranou medzi nimi. Pomáha si pritom aj samotnými chýbajúcimi vrcholmi. Vo svojej podstate je daná metrika odvodená od základnej. Keby sme nahradili chýbajúce vrcholy za chýbajúce hrany, dostali by sme sa k metrike:

$$EdgeSimilarity = \frac{MatchE}{AllE + ExtraE}$$

Takúto metriku môžeme považovať za zovšeobecnenie vyhľadávania na úrovni hrán. Keby sme následne chceli porovnávať dlhšiu postupnosť vrcholov a hrán, ako sú 2 vrcholy s jednou hranou, mohli by sme danú metriku zapísať ako:

$$EdgesSeqSimilarity = \frac{MatchESq}{AllESq + ExtraESq}$$

,

kde skúmame rôzne dlhé sekvencie hrán a vrcholov. Pre každú existujúcu sekvenciu sa snažíme nájsť jej odpovedajúcu sekvenciu v skúmanej mape. Takúto podobnosť môžeme následne uplatňovať nad rôzne dlhými sekvanciami o dĺžke 1 (vrchol, hrana, vrchol) až po dĺžku najdlhšej súvislej acyklickej sekvencie v očakávanej mape. Následne môžeme viacero takýchto metrík spočítať do priemernej, ktorá prideľuje váhy jednotlivým dĺžkam sekvencií. Vo všeobecnosti by takéto váhy mali počítať s tým, že podobnosť pri vyšších dĺžkach sekvencií má väčšiu dôležitosť, pretože určuje vyššiu presnosť mapy. Preto bude mať aj vyššiu váhu. Výsledná metrika pri porovnaní pomocou viacerých dĺžok sekvencií môže vyzerať takto:

$$AbsEdgesSeqSimilarity = \frac{\sum_{i=1}^{n}(weight(i) * EdgesSeqSimilarity(i))}{\sum_{i=1}^{n} weight(i)}$$

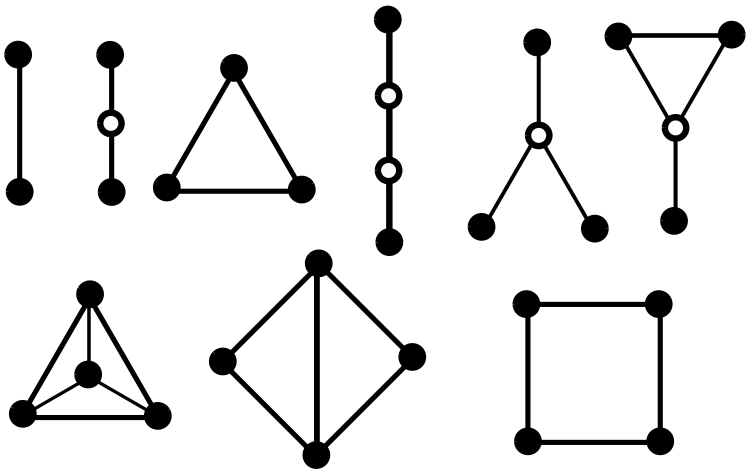
Pri takomto počítaní je dôležité zvoliť vhodnú váhu pre jednotlivé sekvencie hrán. Môžeme pri tom skúsiť využiť nejakú lineárnu závislosť, prípadne kvadratickú či exponenciálnu.

Okrem spomínanej základnej metriky pre porovnávanie myšlienkových máp bola v práci [13] prezentovaná aj myšlienka využitia Graphletov a ich RGF-Distance metriky. Graphlety sa dajú charakterizovať ako siete s malým počtom prepojených prvkov, pričom vždy ide o podmnožinu vrcholov väčšej siete. Táto definícia nasvedčuje tomu, že je skutočne aplikovateľná aj na myšlienkové mapy, ktoré sú práve sieťami a ich časti tak môžeme reprezentovať ako Graphlety. Metóda RGF-distance využíva rôzne typy 3-5 prvkových grafov v porovnávaných sieťach. Základný výpočet je definovaný ako:

$$D(G,H) = \sum_{i=1}^{29} |F_i(G) - F_i(H)|$$

kde: $F_i(G) = -log(N_i(G)/T(G))$ a $T(G) = \sum_{i=1}^{29} N_i(G)$

Pre výpočet sa teda používa porovnanie počtu zastúpenia logaritmickej ceny jednotlivých graphletov v porovnávanej a porovnávajúcej sieti. Pre ilustráciu si môžeme ukázať, aké existujú graphlety o veľkosti 3-5.



Obr. 10.1: Príklad Graphletov pre 2 - 4 vrcholy

Ako je možné vidieť, niektoré z nich sa v myšlienkových mapách vyskytovať nemôžu, čiže s nimi počítať nebudeme a tým sa rýchlosť výpočtu RGF-distance 3-5 zrýchli. Pre všetky vyššie uvedené metriky platí pravidlo

podielu, takže čím nižší výsledok dosiahneme, tým je samotná mapa podobnejšia.

Vzhľadom na existujúce výskumy porovnávania myšlienkových máp a jednoduchosť niektorých metrík sa ukazuje, že tieto uvádzané metriky sa dajú považovať za rozhodujúce pre porovnávanie myšlienkových máp. V zamýšľanom prístupe sa preto pokúsime vhodne využiť niektoré z nich.

10.2 Náročnosť a súvislosti metrík

Vo vyššie opísanej sekcii sme sa venovali opisu niektorých základných metrík pre porovnávanie myšlienkových máp. V ich kontexte je potrebné podotknúť ich vlastnosti, ktoré boli analyzované a zistené v priebehu ich implementácie.

Základná metrika je veľmi jednoduchá, keďže len extrahuje priestorové vlastnosti pojmov. Vďaka tomuto faktu je veľmi jednoduché pracovať nad takouto metrikou, avšak problémom je jej nepresnosť, práve z pohľadu straty priestorového rozloženia mapy. Strácame tak prepojenia a prichádzame tak o súvislosti. Vieme, že sa v mape určitý pojem naozaj vyskytoval, avšak nevieme v akej súvislosti. Tento problém odstraňujeme zavedením ďalších metrík podobnosti.

V prípade metrík, ktoré pracujú nad podgrafmi myšlienkovej mapy s rôznymi veľkosťami súvislých hrán, je potrebné zamyslieť sa nad ich náročnosťou. Tento problém, aj keď sa javí ako nenáročný, nad veľkými grafmi naozaj predstavuje veľký výpočtový problém známy ako Clique problem. Takýto problém rieši napríklad Bron-Kerboschov algoritmus. Podľa výskumov v tejto oblasti [9] [4], sú takéto vyhľadávania podgrafov výrazne náročné, čo sa prejavuje viac so zvyšovaním dĺžky súvislých častí grafu. Riešenia často vyžadujú náročné postupy využívajúce evolučné algoritmy a paralelené spracovanie na rýchlych výpočtových zariadeniach. V našom prípade ide síce o pomerne malé grafy, avšak aj napriek tomuto faktu je vyhľadávanie ich subgrafov náročné najmä z pohľadu štruktúrovania mapy. Pre zjednodušenie v myšlienkovej mape preto budeme pracovať len nad takými subgrafmi grafu, ktoré sú súvislými nasledovníkmi určitého pojmu mapy. V takomto prípade sa vyberajú vždy len podgrafy reprezentujúce priamu líniu prechodu medzi hlavným pojmom a jeho potomkoch na nižších úrovniach.

V analyzovaných metrikách sa nachádza aj metrika pre RGF vzdialenosť v graphletoch. Táto metrika sa však dá určitým spôsobom zovšeobecniť nad myšlienkovou mapou ako metrika, ktorá pracuje nad logaritmickým porovnávaním podobne definovaných súvislých podgrafov ako to bolo v prípade základného tvaru hľadania sekvencií. Pre RGF vzdialenosť v myšlienkovej mape navyše platí pravidlo, že pokiaľ uvažujeme definované 3-5 graphlety 10.1, tak niektoré z nich sa v základnej mape nemôžu vyskytnúť, vzhľadom na acyklickú špecifikáciu myšlienkovej mapy.

Kapitola 11

Špecifikácia návrhu riešenia

Táto kapitola sa zameriava na možnosti návrhu riešenia, ktoré sú základnými predpokladmi v ďalšej práci. Špecifikácia sa zameriava na využitie známych preskúmaných faktov konceptuálnych modelov. Z dôvodov zrozumiteľnosti modelov sú na špecifikáciu návrhu využité prioritne modely zachytené myšlienkovými, kognitívnymi a prípadne aj konceptovými mapami. Základ sa opiera o myšlienky ľahkej zrozumiteľnosti týchto modelov pre neodbornú verejnosť. Výsledné možnosti umožňujú viacero potenciálnych smerov ďalšieho riešenia projektu. Opiera sa o možnosti vyplývajúce z prvotnej analýzy. Z nej vyplývajú najmä určenia pre zachytávanie požiadaviek zákazníkov počas ich zberu vo fáze analýzy a návrhu. Dôležité myšlienky možného ďalšieho výskumu v tejto práci sú modelovanie požiadaviek, využívanie modelov v ďalších fázach vďaka transformáciám, odhaľovanie zhôd v modeloch pre hľadanie modulov softvéru na znovupoužitie a hľadanie potenciálnych miest pre zmeny v prefaktoringu modelov.

11.1 Zachytávanie požiadaviek

Zachytávanie požiadaviek sa pri komunikácii so zákazníkom už spomínalo v predchádzajúcich kapitolách. Na základe pokusov z prvotnej evaluácie na podčiarknutie zmyslu výskumu v tejto oblasti, rovnako ako aj známymi možnosťami využívania myšlienkových máp v súvisiacich projektoch, sa dokázali praktické schopnosti využívať ich v tejto oblasti. Využívanie je pri tom možné z rôzneho pohľadu, závisiaceho od viacerých zvolených triediacich faktorov daného výskumu.

Prvým faktorom je otázka riešiteľa. Pri získavaní požiadaviek je častá obojsmerná komunikácia napríklad pri osobnom stretnutí producenta produktu a zákazníka. V tomto prípade sa o modelovanie stará zúčastnená strana zhotovovateľa produktu. Na druhej strane však pre zvýšenie efektivity získavania požiadaviek je vhodné, aby aj samotný zákazník mohol vytvárať dané modely, prípadne aspoň niektoré ich časti. To by so základnými

znalosťami nemal byť problém, vzhľadom na modelovacie znalosti v myšlienkových mapách. Následne vytvorené modely môžu byť využívané v ďalších fázach, vďaka prechodu k iným modelom technicky presnejším modelom.

Základom pre vhodnú reprezentáciu je vhodne definovaný konceptuálny model. O takýto sa v podobe myšlienkovej mapy snažil poukázať projekt spoločnosti ChangeVision [3]. Ten navrhol princíp modelovania štruktúry a prípadov použitia vďaka vhodnej štruktúre rozdelenej podľa vhodných otázok. Tie boli analyzované aj v časti Evaluácia 7. Podobne zvolené otázky môžu byť vytvorené aj pre zachytávanie ďalších súčastí požiadaviek, napríklad pre reprezentáciu dynamickej zložky konkrétneho prípadu použitia.

11.2 Transformácie modelov

Zachytené modely, ktoré môže vytvárať sám zákazník, prípadne s pomocou inej osoby zo softvérovej domény nie sú vo forme, ktorá by sa následne dala s výhodou využiť v ďalšej fáze. Dostali sme sa do momentu, v ktorom sa úspešne a efektívne podarilo zaznamenať požiadavky. Tie však musia byť predané odborníkom, ktorí sú zvyknutí pracovať skôr s klasickými softvérovými modelmi. Je preto potrebné tieto modely premietnuť do takejto podoby. Manuálne pracovať na transformácií často veľkého počtu zachytených modelov môže byť výrazne náročné a môže spôsobiť nielen stratu počiatočnej výhody, ktorú sme mali pri kolaborácii so zákazníkom, ale môže dokonca spomaliť priebeh vývojového cyklu natoľko, že bude takýto spôsob menej efektívny voči štandardným modelovacím technikám. Je preto významnou myšlienkou ďalšieho výskumu aj návrh takých metód pre transformácie, ktoré umožnia pretaviť jednotlivé transformačné pravidlá do strojovo spracovateľnej podoby pre automatizované transformácie.

Vzhľadom na rozdielnosť vyjadrovacej sily využívaných modelov (myšlienková mapa a UML) je možné vytvoriť hypotézu nie celkom rovnoznačnej reprezentácie takto vytváraných modelov. Pri transformáciách môže dochádzať k viacerým alternatívnym priebehom. Najzákladnejším takýmto priebehom je transformácia, pri ktorej sa všetky artefakty vstupného modelu prejavia aj vo výstupnom (reprezentácia sa zachováva celá). Inou možnosťou je, že sa vstupná reprezentácia bude líšiť od výstupnej. Táto zmena môže byť dvojstranná. Buď sa modely rozširujú a reprezentácia sa tak obohacuje alebo je reprezentácia zúžená, pri ktorej dochádza k stratám.

11.3 Refaktoring modelov

Vytvárané modely bez ohľadu na to, či sú vytvárané od začiatku v UML alebo je na ich vytváranie najskôr využitá niektorá z konceptuálnych techník modelovania, je často potrebné následne ešte upravovať. Pre odhaľovanie bodov, ktoré sa môžu javiť ako vhodné na refaktoring, je možné napomôcť

zdvojením reprezentácie. Niektoré z potrebných zmien môžu byť jasné z modelu v UML, ale v inom modele nemusia byť jasné. Podobne to môže byť aj opačne.

Hľadanie refaktoringu sa pri tom môže diať z viacerých dôvodov. Klasické dôvody môžu vyžadovať ciele vyhľadávania klasických používaných architektonických štýlov v navrhovanom softvéri. Okrem toho je však možné odhaľovať aj možný refaktoring, ktorý napomôže k spresneniu a doplneniu reprezentácie tak, aby bola dostatočne prispôsobená na svoje finálne použitie. Pri práci na viacerých podobných softvérových produktoch sa dá očakávať, že aj keď analýza prebieha vždy od začiatku, výsledné systémy budú podobné a bude možné znovupoužívať ich moduly. Takéto odhalenia môžu zároveň napomôcť k ďalším otázkam smerujúcim k zákazníkovi, ktorý si nemusí vždy uvedomiť niektoré nedostatočne špecifikované požiadavky. Nájdením takýchto nedostatkov je možné zvýšiť kvalitu finálneho softvéru. Jej dosiahnutie sa pri tom vďaka skorému odhaleniu problémových častí môže vyhnúť potrebe zvýšiť náklady alebo čas potrebný na vývoj.

11.4 Príspevok k obohateniu modelov z pohľadu Content Engineeringu

Content engineering sa zaoberá možnosťami modelovania obsahu. Obsah modelov, ktoré sú vytvárané v doméne softvérového inžinierstva môže byť chápaný napríklad v podobe zachytených vlastností modelu. Pre diagramy tried môžu byť takýmito vlastnosťami napríklad atribúty tried, alebo vzťahy medzi triedami navzájom. Tie môžu byť veľmi vhodne modelované prostredníctvom kocneptálnych modelov. Pre získavanie obsahu sa pri tom môžu využívať rôzne zdroje. Príklad prístupu získavania obsahu pre modelovanie je napríklad práca [24], ktorá prestavila potenciálne zdroje znalostí o prepojených faktoch pochádzajúcich z rôznych zdrojov. Podobne sa môžeme zameriavať aj na dáta, ktoré môžu vznikať vo vnútri inštitúcie, ktoré môžu napomôcť k znovupužívaniu nielen faktov a prepojení medzi známymi informáciami, ale aj k znovupoužívaniu samotných ucelených softvérových modulov.

11.5 Modelom riadený vývoj

Modely vytvárané prostredníctvom konceptuálneho modelovania je možné ďalej využívať aj pri použití modelom riadeného vývoja. Využívanie konceptuálnych modelov v takomto prostredí môže prispieť k zvýšeniu produktivity [29], čo je veľmi dôležité meradlo pre softvérový vývoj. Modelom riadený vývoj v takejto forme môže byť charakterizovaný na dvoch úrovniach. Na štandardnej úrovni sa transformujú UML modely až do formy zdrojového kódu. Dôležité z tohto pohľadu je však poznamenať, ako sa dostaneme k takýmto

vstupným UML modelom. Tie môžu byť vytvárané za použitia myšlienkových máp, ktoré sú následne transformované do semištruktúrovaného UML. O jeho úpravu sa následne postará povolaný softvérový inžinier.

11.6 Obohatenie digitálnych konceptuálnych modelov

Využívanie konceptuálnych modelov napríklad pri využívaní myšlienkových máp má v sebe schované veľké množstvo potenciálnych príležitostí pre využívanie v rôznych oblastiach. Takýmto možnostiam sa venuje napríklad práca [11], v ktorej sú zachytené viaceré z nich. Z pohľadu analyzovaných nedostatkov myšlienkových máp sú zaujímavé najmä predpoklady možností viacerých nezávislých myšlienkových máp, ktoré sú skladané do vrstiev. V jednotlivých vrstvách je preto potrebné vedieť zachytávať jeden koncept, ktorý v určitej vrstve môže reprezentovať centrálny element, zatiaľ čo v inej bude len súčasťou niekoľkonásobne vnorenej hierarchie. O možnosti rozdielnych pohľadov sa snažila napríklad aj práca [20], ktorá naznačila možnosť 3D reprezentácie myšlienkových máp. Prínos takejto reprezentácie modelov môže byť rovnako významným ako je prínos 3D UML.

11.7 Prínos k zvýšeniu kvality softvéru

Všeobecným cieľom zavádzania týchto analyzovaných postupov je hlavne snaha ovplyvniť výslednú kvalitu softvéru. Jej meranie je možné vykonávať nad rôznymi charakteristikami kvalitného softvéru. Merateľnosť viacerých týchto vlastností je možné zachytávať už vo fáze návrhu [5]. Práve fázy analýzy a návrhu sa týka predkladaná možnosť využívania konceptuálnych modelov. Merateľnosť vplyvu týchto modelov na tieto vlastnosti vyplýva zo základných vlastností konkrétnych metrík. Nie všetky môžu byť odhaľované už vo fáze návrhu. Nad tými, ktoré je však možné evaluovať už v tejto fáze je potrebné dokázať možnosti a potenciál, ktorý majú konceptuálne modely. Od miery ich vplyvu na kvalitu návrhu sa odvíjajú možnosti na ich postupné rozširovanie v softvérovom inžinierstve.

Kapitola 12

Navrhnutý prístup

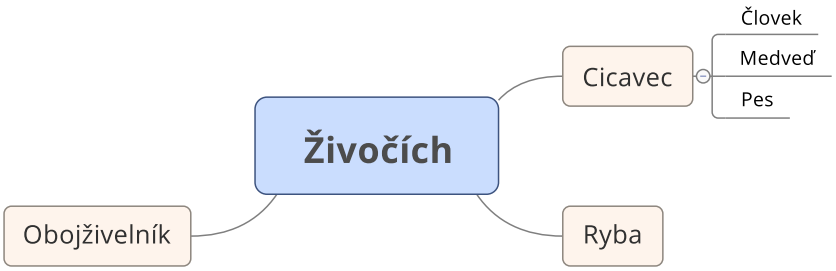
V tejto časti je špecifikovaný navrhnutý prístup a jeho jednotlivé časti. Zameriava sa na využívanie navrhnnutých možností z prechádzajúcej časti 11. Prístup sa venuje transformáciám a porovnávaniu myšlienkových máp. V prípade myšlienkových máp definuje, aké tvary myšlienkových máp sú pre transformácie vhodné, a aké sú pre nich transformačné pravidlá. Prístup je potrebné evaluovať na vhodne zvolených scenároch. Preto sa v tejto kapitole nachádza aj zhrnutie zamýšľaných scenárov pre testovanie.

12.1 Uvažované tvary modelov

V prístupe sú uvažované viaceré typy modelov. Základným modelom je myšlienková mapa, ktorá obsahuje jeden centrálny pojem a okolo neho sa postupne modelujú súvisiace pojmy. Prístup, ako takýto model transformovať je pomerne jednoduchý a vychádza z jazykovedného prístupu myšlienkových máp pri používaní jednoduchých kategorizácií.

Ako príklad môžeme uviesť jednoduchú ukážku z prírody. Ak poznáme živočíchy ako nejaký pojem, potom pojem cicavec má zjavne súvis s daným pojmom a je jeho podmnožinou. Následne pre pojem cicavec a pojmy psa, opice alebo človeka platia podobné pravidlá. Z pohľadu modelovania softvéru môžeme v takomto vyjadrení hľadať paralelu v abstraktnosti. Ak je živočích pomerne abstraktný prvok, potom je cicavec už konkrétnejšou realizáciou prvku živočích. V takomto ponímaní môžeme dané porovnanie považovať za definíciu dedenia. Preto sa základný model myšlienkovej mapy bude prevádzať do tvaru dedičnosti tried. Príklad takéhoto spracovania môžeme vidieť na obrázkoch 12.1 a 12.2.

Pokročilejší tvar myšlienkovej mapy 12.3 12.4 vychádza zo stanovených kľúčových otázok zberu požiadaviek [15]. Tie sú definované ako kto?, čo? a kedy?. Z týchto otázok dokážeme zistiť jednotlivé časti, ktoré majú pre nás rôzny súvis. V časti definovanej pod blokom Kto? nachádzame Actorov potenciálnych prípadov použitia, v časti kedy? zase samotné prípady použi-

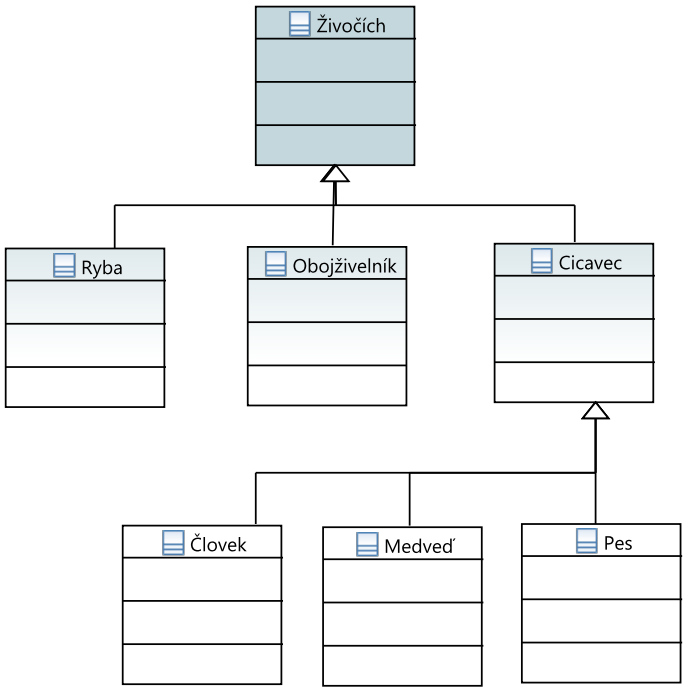


Obr. 12.1: Príklad základnej myšlienkovej mapy

tia. V časti čo? nachádzame názvy tried. Základný tvar teda bude využívať tieto pravidlá. Čo však s modelovaním ďalších podrobností. Pre myšlienkové mapy je typické, že ďalšie rozvádzanie pojmov má nejaký súvis s jeho nadradeným pojmom. Jeden z takýchto súvisov môžeme hľadať v základnom uvažovanom modely myšlienkových máp – dedičnosť 12.1. Práve takýto vzťah sa dá s výhodou aj v časti kto? alebo kedy?.

V týchto častiach definované podpojmy môžeme vyjadriť ako presnejšie typovanie hráčov systému, alebo rozširujúce prípady použitia. Pre časť čo? by sme takéto uvažovanie mohli rovnako využiť, avšak je potrebné vedieť modelovať aj podrobnosti samotných tried. Preto je potrebné nejakým spôsobom odlíšiť atribút a rozširujúcu triedu. Z tohto dôvodu sme zaviedli jednoduché lexikálne pravidlo veľkého písmena 12.4. Rovnaké pravidlo nachádzame často v rôznych odporúčaniach pre programovacie jazyky. Problémom však môže byť neznalosť takýchto štandardov odborníkom z domén mimo softvérového inžinierstva. Preto takúto možnosť radšej vypúšťame a odporúčame im v ďalšej vrstve súvisiacich pojmom uvádzať len atribúty. Súvis nejakých tried v dedičnosti zadefinujeme radšej osobitne do vlastnej myšlienkovej mapy, ktorá môže byť z určitého pohľadu vnímaná ako iná vrstva, čiže iný pohľad, na modelovaný priestor. Príklad transformácie môžeme vidieť na uml diagrame tried 12.5 a diagrame prípadov použitia 12.6. Na uvedených príkladoch je možné vidieť práve spomínanú možnosť modelovania dedenia tried a mapovania atribútov zároveň. v prípade, že neodlišujeme veľké písmená, by dané podtriedy boli mapované rovnako ako atribúty.

Okrem takéhoto tvaru však môžeme uvažovať aj nad mapou, ktorá nemá presne definované hranice, teda ani rozdelenie podľa kľúčových otázok. V takomto prípade je vhodné na transformáciu využiť niektoré pravidlá týkajúce sa spracovania prirodzeného jazyka. Z pomenovávaní základných požiadaviek a ich slovných druhov môžeme usúdiť, že sa v nich nachádzajú také slovné druhy, ktoré dokážeme presne mapovať na konkrétne elementy v UML. Príkladom môžu byť štandardne krátke, často jednoslovné pomenovania, ktorých základom je podstatné meno. V takomto prípade sa daný pojem dá jednoducho transformovať do podoby triedy v UML diagrame tried. V inom prípade, slovesá a ich viazaný popis, prípadne slovesné pod-



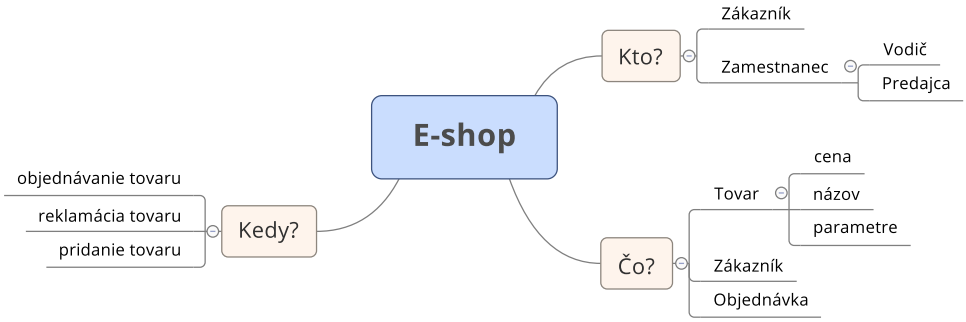
Obr. 12.2: Príklad transformácie základnej myšlienkovej mapy do UML

statné mená môžeme zase považovať za pomenovania prípadov použitia. Vďaka týmto pravidlám by sme teda boli schopný aj z bližšie nešpecifikovanej myšlienkovej mapy, bez konkrétnych ohraničení nachádzať vhodné elementy pre zodpovedajúci UML diagram. Spracovanie prirodzeného jazyka v takejto forme je však často limitované analyzovanými problémami s nejednoznačnosťou slov, pre ktoré často nedokážeme určiť ani ich správne zaradenie do slovných druhov. Príkladom je známa veta NLP v angličtine: „Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo“. V prípade myšlienkových máp však nie je vhodné a nemalo by sa modelovať v celých vetách. Na to je vhodná prikladaná poznámka, ktorá bližšie reprezentuje daný pojem. V modelovaní požiadaviek sa navyše takéto poznámky priamo očakávajú. V samotných pojmoch myšlienkovej mapy však ide o jednoduchšiu slovnú skupinu, v ktorej je odhaľovanie slov potenciálne jednoduchšie.

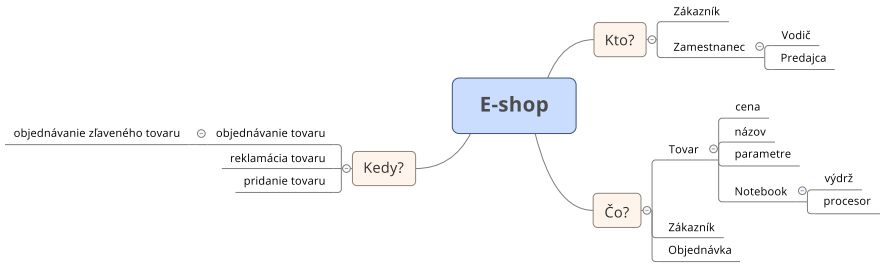
12.2 Definície transformačných pravidiel

Ako sme spomenuli v predchádzajúcej časti, zadefinovali sme jednoduché pravidlá pre transformácie. V tejto časti sumarizujeme ich krátky prehľad. Zoznam pravidiel je nasledovný:

1. Pre základný tvar myšlienkovej mapy sú jej odvádzané pojmy vždy



Obr. 12.3: Príklad pokročilejšej myšlienkovej mapy



Obr. 12.4: Príklad pokročilejšej myšlienkovej mapy 2

považované za konkretizáciu, to znamená, že v UML predstavujú základný vzťah dedičnosti 12.1 12.2.

2. Pre tvar mapy obsahujúci kľúčové otázky sa rozlišujú ich jednotlivé podčasti na rôzne modely definované v UML:

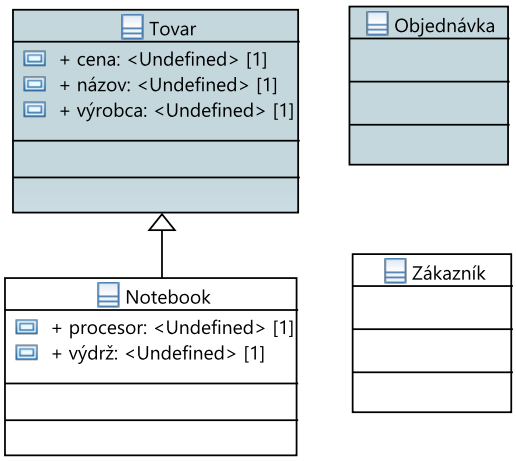
   (a) Časť kto? a časť kedy? reprezentujú UML diagram prípadov použitia 12.6

   (b) Časť čo? reprezentuje UML diagram tried 12.5

3. V diagrame s kľúčovými otázkami pre otázku kto? 12.6 definujeme pojmy na prvej úrovni ako hráčov systému, pojmy na ďalších úrovniach sú konkretizáciou predchádzajúcich, platia pre nich preto pravidlá ako v bode 1

4. V diagrame s kľúčovými otázkami pre otázku kedy? 12.6 definujeme pojmy na prvej úrovni ako prípady použitia a pojmy na ich ďalších úrovniach ako ich odvodené (extends) prípady použitia a platia pre nich pravidlá podobné ako v bode 1

5. V diagrame s kľúčovými otázkami pre otázku čo? definujeme pojmy na prvej úrovni ako triedy v systéme 12.5. Pre pojmy na ďalšej úrovni môžeme definovať voliteľne:



Obr. 12.5: Príklad transformácie pokročilejšej myšlienkovej mapy (diagram tried)

(a) Pojmy rozlíšime malými a veľkými písmenami: písané veľkým začiatočným písmenom sú konkretizácie (bod 1), pojmy malým sú atribúty

(b) Všetky pojmy na nižšej úrovni sú atribúty

6. V diagrame bez špecifikovaného tvaru sa riadime lingvistickými pravidlami rozlišovania slovných druhov pre základ definovaného pojmu:

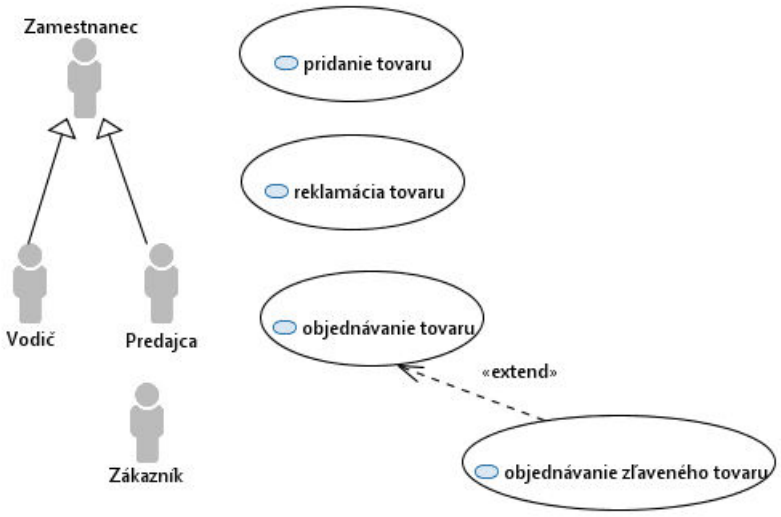
(a) Pojmy so základom s podstatným menom sú triedy alebo hráči systému. Bližšie určenie je otázka na ďalší výskum vychádzajúci z predpokladu ďalších úspešne určených súvisiacich prepojených pojmov

(b) Pojmy so základom so slovesom môžeme považovať za prípady použitia

Nejednoznačnosti v definícii pravidiel je možné obchádzať vďaka moderovaniu zberu požiadaviek. Ak sa zberu požiadaviek účastní poučená osoba, môže napríklad rozlišovať názvy s malými a veľkými písmenami. V opačnom prípade necháme definíciu dedenia na ďalšiu fázu modelovania, refaktoring, prípadne sa odkážeme na inú vrstvu myšlienkovej mapy

12.3 Transformačný nástroj

Nad takto definovanými pravidlami bol implementovaný prototyp nástroja umInd. Tento nástroj dokáže na základe definovaného vstupného tvaru modelu pretransformovať daný tvar do UML, alebo naopak konkrétne UML diagramy tried a prípadov použitia na myšlienkovú mapu. Nástroj bol vyvíjaný

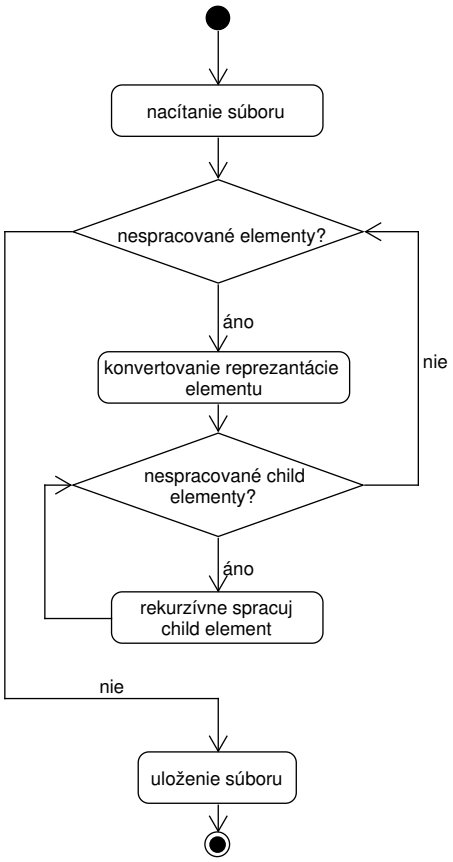


Obr. 12.6: Príklad transformácie pokročilejšej myšlienkovej mapy (diagram prípadov použitia)

pomocou programovacieho jazyka Java. Odkazuje sa na využitie najpoužívanejšieho nástroja na modelovanie myšlienkových máp Xmind a jednoduchú reprezentáciu UML vďaka rozšíreniu pre Eclipse s názvom Papyrus, ktorý sa často s výhodou využíva pri transformáciách a prácou s modelom-riadeným vývojom softvéru. Pre tieto uvažované modely platí jedna spoločná črta. Modely sú zaznamenávané pomocou XML. Vďaka nemu je možné modely jednoducho čítať a hľadať v nich súvis. Pre parsovanie daných súborov bol zvolený základný parser používajúci Document Object Model (DOM).

Pre pojmy bola v nástroji vytvorená samostatná stromová reprezentácia, ktorá je typická pre myšlienkové mapy. Vďaka nástroju dokážeme okrem vykonávania základných transformácií podľa definovaných pravidiel, určovať aj zhodu medzi jednotlivými modelovanými myšlienkovými mapami. To dosahujeme vďaka využívaniu evaluačných algoritmov pre podobnosť, ktoré boli analyzované v kapitole 10.

Implementácia nástroja je intuitívna a je rozdelená do častí podľa jednotlivých modelov a jednotlivých úloh (transformácie, počítanie podobnosti). Celková štruktúra umožňuje jednoduché doimplementovanie častí súvisiacich s novými transformačnými pravidlami (napr. uvažované spracovanie prirodzeného jazyka) alebo doplnenie podpory pre ďalšie modelové reprezentácie (napr. FreeMind myšlienkové mapy, Eneterprise Architect UML modely a pod.). Priebeh jednoduchej transformácie je vyjadrená v diagrame 12.7 a spracovávanie elementov mapy pri metrikách je uvedené v diagrame 12.8.

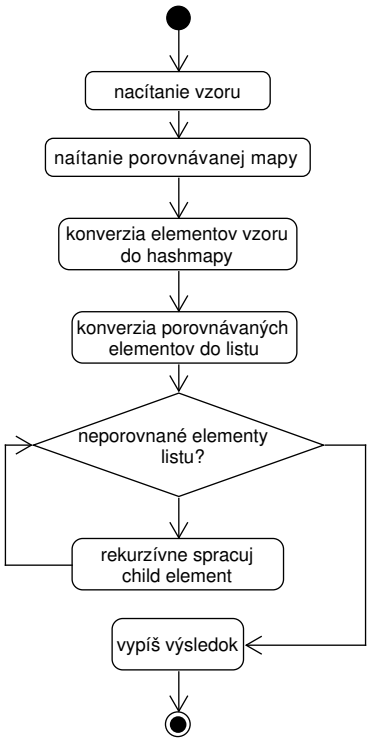


Obr. 12.7: Ukážka priebehu jednoduchej transformácie

12.4 Testovacie scenáre

Nad definovaným prístupom pre transformácie je potrebné vytvorenie vhodného setu testovacích scenárov. K takýmto scenárom môžeme zaradiť nasledovné scenáre:

- Porovnanie modelovaných máp priamo vo forme myšlienkovej mapy: prístup navrhnuje použitie myšlienkových máp, preto je potrebné ukázať, že práve takáto reprezentácia je vhodná a zhoduje sa v dostatočnej miere s očakávaniami, ktoré sú na jej výsledok kladené.
- Porovnanie máp v transformovanej forme UML: vytvárané myšlienkové mapy sú transformované do UML. Preto je potrebné otestovať, či transformácia prebehne úspešne a následné modely v tejto forme zachytávajú skutočne celý modelovaný obsah.
- Porovnanie máp po transformácii do UML a späť do myšlienkovej



Obr. 12.8: Ukážka priebehu výpočtu metriky

mapy: Transformácie fungujú nad jednoduchým zoznamom transformačných pravidiel, niektoré časti mapy tak môžu potenciálne z máp pri transformácii zmiznúť, prípadne môžu pribudnúť pre lepšie doplnenie modelu. Preto je potrebné, aby sme ukázali porovnanie aj takých máp, ktoré prešli procesom transfomácie obojsmerne a ukázať, aký na nich mala transformácia vplyv.

- Porovnanie modelov v UML modelovaných priamo v tejto forme: z pohľadu výskumu je zaujímavé porovnávať modely modelované pomocou myšlienkových máp aj pomocou UML. Tieto modely môžeme porovnávať v zmysle zistenia, do akej miery je v nich podobnosť pri ich modelovaní.

- Porovnanie UML modelov transformovaných do myšlienkovej mapy: definované transformácie umožňujú transformácie aj existujúcich modelov v UML do myšlienkovej mapy. Tú očakávame skôr na využitie pred samotným modelovaním do UML, ale môžu sa vyskytnúť situácie, kedy je forma UML už hotová a nad takýmito projektami (rozpracovanými) sa môžu ukázať vhodné zmeny po transformácii do myšlienkovej

mapy.

- Porovnanie UML modelov transformovaných do myšlienkovej mapy a späť do UML: podobne ako sme formulovali možné zmeny modelov v prípade myšlienkových máp, rovnako môžeme tieto pravidlá preniesť aj na UML modely. Po opakovanej transformácii sa môžu ukázať rôznorodé zmeny v modeloch.

- Čiastkové porovnanie nad kategóriami myšlienkovej mapy (podľa definície kto? čo? kedy?): jednotlivé časti rozšírenej myšlienkovej mapy reprezentujú rôznorodé prvky v odpovedajúcej UML reprezentácii. Tieto elementy nie sú rôzne len z pohľadu ich formy, ale aj konkrétneho diagramu, v ktorom sa v rámci UML využíva (diagram tried, diagram prípadov použitia). Preto je možné skúšať pokusy aj nad ich časťami, a tak získať prehľad o tom, v ktorej časti sa transformačné zmeny prejavia najviac.

- Tieto porovnania po normalizácii nejednoznačností: z analýzy vyplýva, že niektoré elementy môže byť problematické modelovať a ich nejednoznačnosť spôsobuje zjavné problémy pri porovnávaní. Preto je potrebné tieto nejednoznačnosti určitým spôsobom normalizovať a transformáciu aplikovať až následne.

- Tieto porovnania po vynechaní nejednoznačností: niektoré nejednoznačnosti môžu byť natoľko problematické pre porovnávanie a transformáciu, že je vhodné ich vynechať. Ich prítomnosť by bez takéhoto kroku mohla skreslovať výsledky.

Podobne je možné vytvárať ďalšie scenáre. Pre testovanie pri tom môžeme modelovať nový produkt, prípadne využívať modely existujúceho softvérového produktu, čo môže byť vhodné pri snahe nájsť vhodné miesta pre rozšírenie pomocou prevedenia UML modelu daného produktu do myšlienkovej mapy a následného dopisovania požiadaviek. S rozširovaním schopností vytvoreného prístupu sa dá očakávať rozšírenie ďalších typov testovacích scenárov.

Kapitola 13

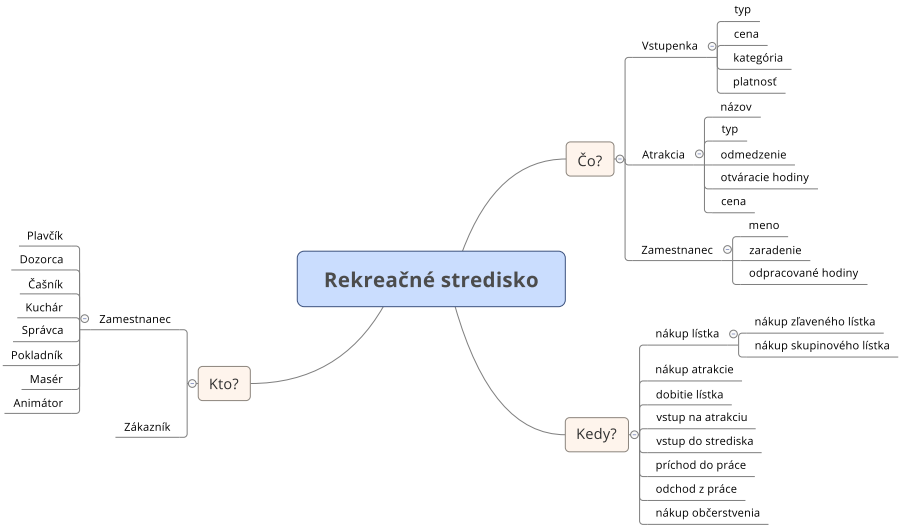
Základná evaluácia prístupu

Navrhovaný prístup bol počas analýzy výskumu niekoľkokrát evaluovaný nad radom jednoduchých experimentov. Nad týmito experimentami sa postupne podarilo špecifikovať a implementovať najvhodnejšie transformačné pravidlá pre myšlienkové mapy. Postupným prechádzaním možných reprezentácii sme sa dostali k výsledku v podobe navrhnutého prístupu 12. Pre jednotlivé tvary máp bolo potrebné vyskúšať niekoľko možných reprezentácii a evaluovať riešenie v súvislosti s pravidlami pre modelovanie správnych myšlienkových máp. Z pravidelných experimentov vzišiel ako výsledok špecifikovaný prístup. Tento prístup úspešne dokáže transformovať myšlienkové mapy do UML diagramov a rovnako niektoré tvary UML diagramov do myšlienkových máp. Tento základný prístup sme sa rozhodli evaluovať nad jednoduchým príkladom 7 zadaným malej skupine respondentov, ktorí nepatria medzi expertov z domény softvérového inžinierstva. Z tejto evaluácie sme dokázali vyvodiť závery pre ďalšie pokračovanie výskumu, ktoré napomohlo ďalšiemu smerovaniu vývoja prístupu. Vďaka jeho úspešnému vplyvu na ďalší priebeh výskumu sme sa rozhodli realizovať ďalší menší experiment, ktorý je popísaný v tejto časti.

13.1 Zadanie experiemntu

Pre evaluáciu navrhnutého prístupu bol zvolený jednoduchý príklad aplikácie. Pri evaluácii sme sa rozhodli postupovať využitím trojitého tvaru zadania. Daný trojitý tvar súvisí s predostretými faktami, ktoré majú byť v aplikácii obsiahnuté. V základnom zadaní sú zahrnuté len tie fakty, ktoré sú pre danú aplikáciu kľúčové, avšak umožňujú respondentom využíť vlastné predstavy aplikácie vo väčšej miere. V druhom prípade sa snažíme testovanej skupine ľudí predložiť väčšie množstvo parametrov, ktoré aplikácia musí obsahovať. V poslednom prípade im dávame v podstate ucelený zoznam požiadaviek, ktorý majú za úlohu preniesť do tvaru myšlienkovej mapy. Daný trojitý tvar zadania je kľúčový pre odhaľovanie zhody v myšlienkových ma-

pách. Pre experiment sme vytvorili vlastnú porovnávanú myšlienkovú mapu 13.1, ktorá poslúži ako vzor pre porovnávanie. Na vytvorené mapy sme využili vytvorený nástroj umInd, ktorý bol použitý na transformáciu do UML a na matematické vyhodnotenie podobnosti medzi namodelovanými a očakávanými modelmi.

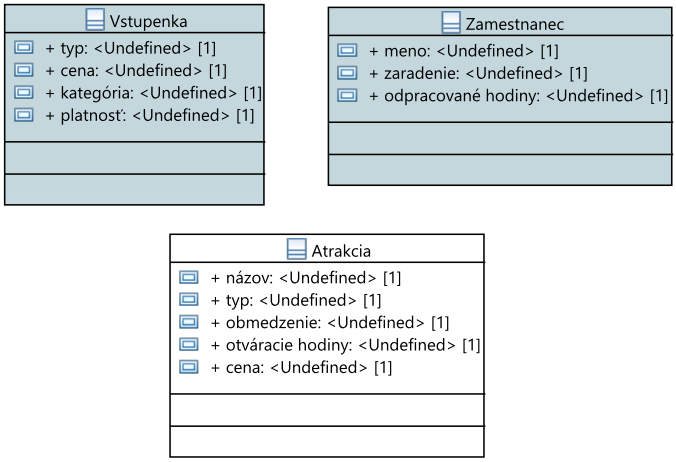


Obr. 13.1: Vzorový vzhľad myšlienkovej mapy

Ako modelovaná aplikácia bol zvolený jednoduchý príklad systému pre rekreačné stredisko. Takéto stredisko si môžeme predstaviť ako nejaký aquapark ponúkajúci rôznorodé služby pre svojich zákazníkov. Tieto služby súvisia najmä s rekreáciou, oddychom a zábavou. Patria k nim však aj iné pomocné služby ako napríklad služby gastronómie. Takto špecifikovaný príklad si môžeme reprezentovať pomocou popisu jednotlivých jeho súčastí. Na tento popis bola zvolená myšlienková mapa očakávaného modelu 13.1. Takto definované požiadavky na systém je možné vďaka v prístupe navrhnutom transformačnom nástroji previesť do UML diagramov na obrázkoch 13.2 13.3 13.4. Plné znenie experimentu je možné nájsť v prílohe H. Toto zadanie bolo určené ľuďom mimo softvérového inžinierstva, ktorí mali vytvoriť vlastné mapy.

13.2 Získané modely a ich podobnosť

Skupina ľudí, ktorá mala modelovať vlastné myšlienkové mapy nad zadaným príkladom systému pre rekreačné stredisko dosiahla rôznorodé výsledky. Niektoré z uvažovaných máp je možné nájsť v prílohách, kompletné modely je možné nájsť na priloženom elektronickom médiu. Z dôvodu viacerých modelov budeme v tejto časti uvádať len jeden príklad, ostatné je možné nájsť na priloženom elektronickom médiu G.



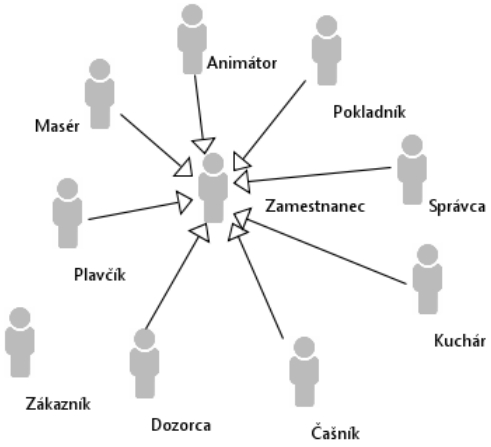
Obr. 13.2: Diagram tried získaný transformáciou vzoru

Tvar zadania	Základná metrika	Granulárna metrika	Metrika 2-4 línií
Základný	0,4011	0,3473	0,2158
Rozšírený	0,4997	0,4503	0,356
Úplný	0,9936	0,9129	0,8477

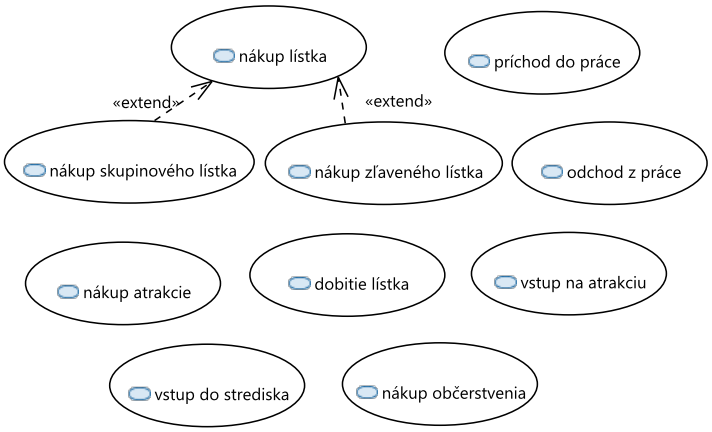
Tabuľka 13.1: Výsledky experimentu porovnávania máp, uvádzané sú priemerné pre jednotlivé tvary zadania

Ako je možné vidieť z obrázkov 13.5 13.6 13.7, modely sú podobné očakávanému modelu a ich podobnosť je postupne vyššia so zvyšovaním presnosti popisu očakávaných požiadaviek pre systém. Podobné výsledky dosiahli aj ďalší vo svojich modeloch. Aby sme však získali presnejšie výsledky podobnosti, aplikovali sme nad týmito modelmi tri metriky - základnú metriku podobnosti všetkých zachytených faktov, metriku pre porovnávanie dvojíc prvkov previazaných hranou a metriku pre porovnávanie rôzne dlhých súvislých línií hrán (od 2 do 4) 10. Výsledky sme porovnávali nad jednotlivými modelmi vo všetkých troch verziách modelu osobitne. Získané podobnosti je možné vidieť v tabuľke 13.2 a v grafe 13.8. Výsledky podľa jednotlivých respondentov je možné nájsť v prílohe H.

Okrem evaluácie podobnosti sme modely transformovali do UML. Na obrázku 13.9 je možné vidieť príklad takto transformovanej mapy, konkrétne časti týkajúcej sa diagramu tried.Pri porovnaní s diagramami očakávaného modelu vidíme zjavnú podobnosť. Pre všetky získané modely je možné takýmto spôsobom transformovať myšlienkové mapy do diagramov UML. Je to zabezpečené všeobecnou aplikovateľnosťou transformačného nástroja.



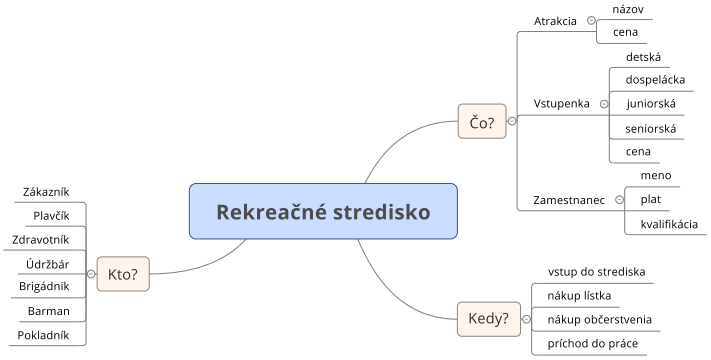
Obr. 13.3: Hráči systému získaný transformáciou vzoru



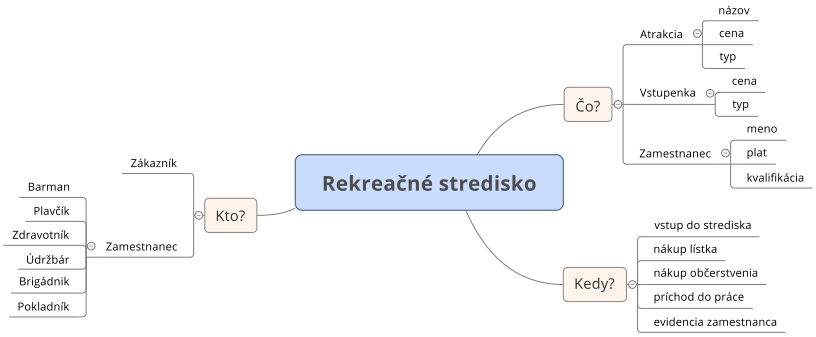
Obr. 13.4: Prípady použitia získané transformáciou vzoru

13.3 Vyhodnotenie

V experimente sme sa snažili poukázať na podobnosti modelovanej reprezentácie jednoduchého softvérového systému pre rekreačné stredisko. Pre takéto stredisko je možné modelovať požiadavky v zachytených myšlienkových mapách. Z výsledkov je známe, že testovaná vzorka ľudí dokázala pomerne úspešne reprezentovať daný systém v myšlienkovej mape. Rozdiely oproti očakávanému modelu sú jasné a vyplývajú z osobných predstáv. Avšak pri pridávaní nutných súčastí do myšlienkovej mapy, postupujú modelujúci podľa presnejšieho zadania a ich modely sú podobnejšie očakávanému. Metriky ukázali mieru podobnosti v jednotlivých prípadoch. Z pohľadu transformácií sa nám podarilo úspešne transformovať získané myšlienkové mapy do UML

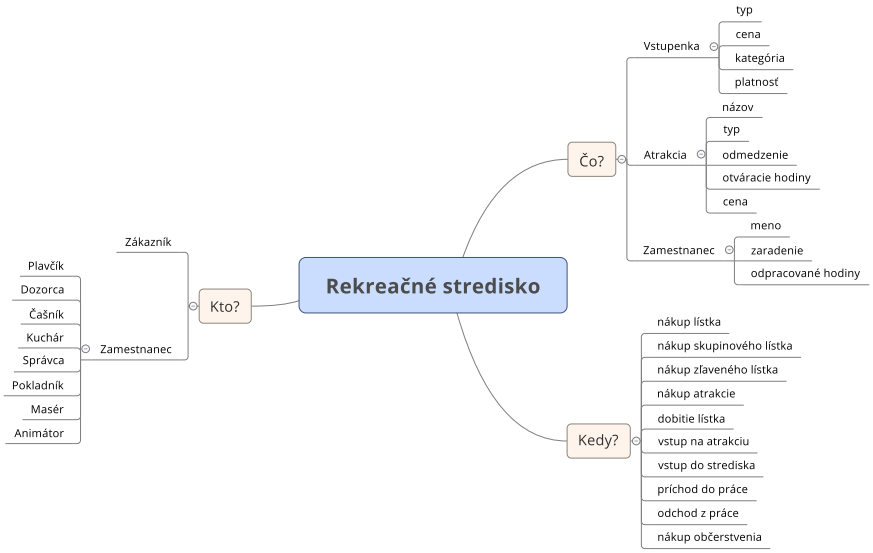


Obr. 13.5: Vzhľad myšlienkovej mapy prvej verzie

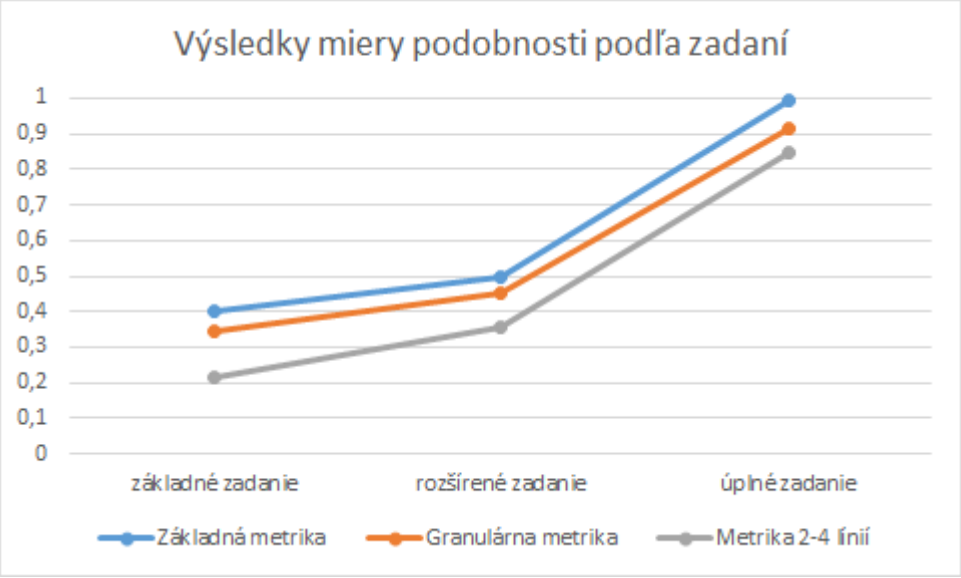


Obr. 13.6: Vzhľad myšlienkovej mapy druhej verzie

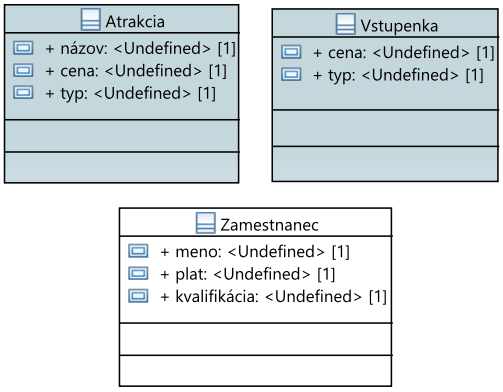
diagramov tried a diagramov prípadov použitia. Experiment opätovne potvrdzuje možnosti modelovania cez myšlienkové mapy, vďaka jednoduchosti ich notácie. Pre očakávané konkrétne systémy dostávame výrazne podobné myšlienkové mapy. Transformáciami ich následne dokážeme preniesť do formy UML. Výsledky experimentu potvrdzujú predpoklady, ktoré boli stanované pred jeho vykonaním. Výsledky však bude potrebné ešte ďalej preverovať nad väčšou skupinou ľudí a overiť tak, či v tomto experimente nemohlo dôjsť k skresleniu výsledkov.



Obr. 13.7: Vzhľad myšlienkovej mapy tretej verzie



Obr. 13.8: Graf výsledkov evaluácie podobnosti



Obr. 13.9: Vzhľad diagramu tried druhej verzie myšlienkovej mapy 13.6

Kapitola 14

Rozšírený prístup

V tejto kapitole je popísané rozšírenie špecifikovaného prístupu, ktoré dopĺňa a pozmeňuje niektoré jeho predchádzajúce časti. V časti 14.1 sa venujeme obmene uvažovaných tvarov modelov. Okrem toho rovnako zavádza nové zhrnutie transformačných pravidiel 14.3 a popisuje, ktoré UML diagramy môžu byť výstupom transformácie myšlienkovej mapy 14.2. Dopĺňa rovnako aj opis novej rozšírenej verzie transformačného nástroja 14.4.

14.1 Uvažované tvary modelov

Oproti základnému prístupu je v rozšírenom prístupe doplňaná nová vetva základnej štruktúry myšlienkovej mapy v tvare Kto?, Čo? a Kedy? (Who?, What?, When?) o nový prvok Ako? (How?). Tento nový prvok sa na základe experimentovania a postupnej analýzy možných nových kľúčových otázok pre analýzu požiadaviek ukazuje ako vhodná možnosť, ako zachytiť záležitosti týkajúce sa komponentov systému, prípadne externých komponentov pre systém. Z tohto pohľadu sa kľúčová otázka dá využiť ako základ pre generovanie diagramov komponentov, prípadne ešte skôr ako diagram nasadenia.

Ako príklad môžeme uviesť webovú aplikáciu, ktorá vyžaduje dvojaké tvary používateľského rozhrania prispôsobené pre mobilné zariadenia a stolové počítače. Takýto systém môže pracovať s konkrétnou existujúcou databázou produktov, pracovať so systémom dodávateľa alebo preberať informácie o recenziách z externého portálu. Všetky tieto komponenty tak zahrnieme do spomínanej vetvy a následne transformáciou môžeme získať tieto komponenty v diagrame nasadenia, respektíve aj ich väzby, vďaka ich vzájomnému prepojeniu. Niektoré prepojenia kvôli acyklickosti myšlienkovej mapy zachytíme v iných vrstvách.

Zvyšné časti modelu ostávajú rovnaké ako v základnom prístupe 12. Jedinou zmenou oproti základnému prístupu je zavedenie vrstvenia a niektorých nových metód práce so slovami a ich analýzou.

14.2 Zasiahnuté UML diagramy

V rámci prístupu je možné získavať diagramy, respektíve ich základné časti pre viacero rôznych diagramov. Oproti základnému prístupu 12 sa snažíme využívať možnosti, ktoré pridáva kľúčová otázka Ako? a zavedenie vrstveného modelu. Už v základnom prístupe sme sumarizovali, že prístup je schopný generovať UML diagramy tried a diagramy prípadov použitia, avšak bez viacerých typov prepojení. Vďaka pridaniu kľúčovej otázky Ako? sa pridáva aj diagram nasadenia. Vďaka vrstvám je možné vytvárať vzťahy medzi viacerými rôznymi prvkami.

Príkladom takýchto vzťahov môže byť prepojenie triedy a prípadu použitia. V tomto prípade je možné vytvárať diagram kompozitnej štruktúry ako realizáciu prípadu použitia, pričom element triedy je v tomto prípade považovaný za konkrétnu objektovú inštanciu tohto typu. Zároveň tak získame aj základ sekvenčného diagramu pre prípad použitia, konkrétne lifelines (podobne charakterizované ako inštancie triedy).

Vďaka systému rozpoznávania slovných druhov je zase možné získať niektoré ďalšie prvky. V prípade podprvkov myšlienkovej mapy pre koncept transformovaný do triedy alebo atribútu je možné získať stavy daného prvku. Vygenerovať tak môžeme diagram stavov charakterizovaný pre daný prvok, avšak bez znalosti o prechodoch medzi stavmi.

14.3 Transformačné pravidlá

Základ transformačných pravidiel vychádza z podstaty formy modelu. V prípade, že sa v myšlienkovej mape nachádza forma s kľúčovými otázkami, pracuje sa s týmito vetvami osobitne podľa vlastných pravidiel. Ak v myšlienkovej mape nie je možné nájsť tento tvar, s celým modelom sa pracuje podľa pravidiel pre časť Čo?.

Časť Kto? sa transformuje do konceptov diagramu prípadov použitia, konkrétne do hráčov systému. Podprvky sa ďalej transformujú použitím vzťahu dedičnosti (generalizácie) medzi hráčmi.

Časť Kedy? sa transformuje do konceptov diagramu prípadov použitia, konkrétne do samotných prípadov použitia. Podprvky sa transformujú použitím vzťahu extend medzi prípadmi použitia (od rodiča je odvodený konkrétnejší prípad použitia v podprvku).

Časť Ako? sa transformuje do konceptov diagramu prípadov použitia, konkrétne do komponentov, podprvky sú charakterizované ako komponenty, pričom vzťah je transformovaný do asociácie medzi takto súvisiacimi komponentami.

Časť Čo? sa transformuje do konceptov diagramu tried, konkrétne do samotných tried. Podprvky sú charakterizované ako podtriedy. Výnimkou je len zavedenie analytických možností textového spracovania, pri ktorých sa

podprvky môžu transformovať do atribútov (rozlišovanie malých a veľkých začiatočných písmen) alebo do stavov (nájdenie vhodného zodpovedajúceho slovného druhu – najmä prídavné meno).

V prípade myšlienkovej mapy v tvare s kľúčovými otázkami môžeme s výhodou využiť vrstvy. Tie sa zameriavajú na prepájanie konceptov, ktoré by inak prepojené byť nemohli. V závislosti na type prepojenia môžeme vytvárať viacero rôznych vzťahov pre existujúce diagramy, prípadne vytvárať základ pre ďalšie typy diagramov.

V prípade vrstvy s prepojením odhaleného prípadu použitia a triedy, je takáto väzba transformovaná do diagramu kompozitnej štruktúry v tvare realizácie prípadu použitia (triedy ako inštancie svojho typu). Zároveň je vytváraný základ sekvenčného diagramu.

V prípade vrstvy prepájajúcej hráčov systému a prípady použitia, je možné vytvárať medzi nimi asociácie. Vďaka čomu dokážeme charakterizovať, ktorý hráč bude pracovať s ktorým prípadom použitia.

Zaoberať sa je možné aj ďalším typom prepájania v prípadoch použitia – vzťahom Include. Keďže tento vzťah nevieme vytvoriť v základnej vrstve, pokus o jeho vytvorenie je možné realizovať v samostatnej vrstve, kde centrálny prvok vrstvy charakterizovaný ako prípad použitia bude zahŕňať prípady použitia v jeho vetvách.

V prípade transformácií opačným smerom, teda z UML do myšlienkovej mapy, sa využívajú tieto pravidlá inverzným spôsobom, pričom sa transformácie snažia o zachovanie väčšinovej časti modelu. Pre UML modely s rôznorodými typmi konceptov sa generuje najprv základná štruktúra kľúčových otázok a následne elementy podľa špecifikovaných pravidiel pre tvary modelov. Neznáme typy väzieb alebo elementov nie sú v transformácii využívané, a tieto časti sa stratia.

14.4 Transformačný nástroj

Súčasťou navrhnutého rozšíreného prístupu je aj špecifikácia nového transformačného nástroja, vďaka ktorému je možné robiť špecifikované transformácie automatizovane. Nástroj sa zameriava na transformáciu modelov myšlienkových máp vytvorených v nástroji Xmind. Ten je charakterizovaný ako jeden z najpoužívanejších nástrojov na svete v oblasti myšlienkových máp. Pre vytváranie vrstiev používame samostatné listy, v ktorých pridávame elementy s hyperlinkom na originálny element v základnej vrstve s kľúčovými otázkami.

Pre modelovanie UML diagramov bol zvolený, podobne ako v predchádzajúcej verzii nástroja, modelovací nástroj Papyrus. Ten, napriek niektorým svojim nevýhodám z hľadiska používania a grafického stvárnenia niektorých elementov, patrí k nástrojom vhodným k experimentovaniu, keďže má spoľahlivú XMI štruktúru diagramov, a diagramy vytvárané pomocou

XML je možné jednoducho zobraziť použitím systému ťahaj a pusť (drag and drop). Zároveň ide o nástroj, ktorý je možné s výhodou využiť pri ďalších transformáciách charakterizovaných v prístupoch modelom riadeného vývoja.

Samotný nástroj je navrhnutý ako ľahko rozšíriteľný o ďalšie potenciálne tvary modelov. V prípade, že by bola snaha pracovať s modelmi UML z iných nástrojov, ako napríklad Enterprise Architect, je možné navrhnúť importér podobným spôsobom ako pri Papyruse. Podobne je to aj s myšlienkovými mapami. Tam má samozrejme Xmind nespornú výhodu aj v tom, že je schopný sám transformovať myšlienkové mapy aj iných nástrojov, napríklad FreeMind, MindJet alebo NovaMind.

Nástoj je rozdelený podľa častí, ktorým sa venuje. Pre myšlienkové mapy aj pre UML sú v nástroji balíky obsahujúce ich základnú štruktúru vychádzajúcu z ich metamodelov, respektíve ich zjednodušenej podoby. Zároveň nástroj obsahuje časti venujúce sa importom a exportom modelov do vnútornej podoby modelov, a časti zameriavajúce sa na spracovanie prirodzeného jazyka použitím nástrojov Stanford CoreNLP a WordNet. Okrem toho sú v nástroji balíky týkajúce sa obsluhy a používateľského rozhrania.

Podrobný popis používateľskej a inštalačnej príručky je možné nájsť v prílohách F a E.

Kapitola 15

Podrobná evaluácia prístupu

V tejto kapitole sa nachádza zhrnutie evaluácie rozšíreného prístupu špecifikovaného v predchádzajúcej kapitole. V časti Evaluačné scenáre 15.1 charakterizujeme niekoľko vhodných sledovaných prípadov, ktoré boli vyhodnotené v časti vyhodnotenie scenárov 15.2. Záverečná sumarizácia a zrhnutie tejto etapy sa nachádza v časti zhodnotenie 15.3

15.1 Evaluačné scenáre

Nad vytvoreným prístupom boli špecifikované viaceré testovacie scenáre. Zahŕňajú všetky potrebné súčasti, ktoré boli v prístupe navrhnuté, a ktoré je potrebné overiť voči očakávaným výsledkom a iným podobným prístupom, ktoré boli analyzované. Dané testovacie scenáre sú nasledovné:

Porovnávanie požiadaviek modelovaných v tvare myšlienkovej mapy voči očakávaným výsledkom v tejto forme. Ak očakávame, že je modelovacia technika myšlienkových máp dostatočne jednoduchá na to, aby jej pochopili ľubovoľní doménoví experti, musí byť výsledok takéhoto porovnania dostatočne smerodajný. Jednoduchosť modelovania požiadaviek v tejto forme je základom celého rozšíreného prístupu využívajúceho túto techniku. Tento evaluačný scenár sme sa snažili naplno vyhodnotiť už v prechádzajúcom vyhodnotení v kapitole 13.

Porovnávanie transformovaných myšlienkových máp do UML voči očakávaným výsledkom v UML. Po úspešnom modelovaní požiadaviek v tvare myšlienkovej mapy musí byť jasné, že sa dané požiadavky dajú preniesť aj do podoby, ktorá je výhodnejšia pre softvérového inžiniera. Vďaka takémuto modelovaniu sa tak zrýchli prenos základných modelovaných požiadaviek od zákazníkov k tvorcom softvéru. Výsledné modely sa využijú v ďalších fázach vývoja alebo môžu byť základom pre ďalšie transformácie v modelom riadených prístupoch.

Porovnávanie transformovaných UML modelov v tvare myšlienkových máp. V niektorých prípadoch sa môže ukázať ako vhodné

prechádzať medzi modelmi aj opačným smerom. Takýmto prípadom môže byť doplnenie špecifikácie už existujúceho produktu, prípadne spätné prekonzultovanie niektorých pozmenených častí modelu.

Myšlienkové mapy porovnávané po transformácii do UML a späť do pôvodného tvaru. Takýto testovací prípad dokáže napovedať, do akej miery sú transformačné pravidlá špecifikované dostatočne a nakoľko sa v akých prvkoch požiadaviek objavujú problémy, ktoré môžu spôsobiť stratu alebo pridanie ďalších požiadaviek.

UML modely transformované do myšlienkovej mapy a späť do UML sú podobne ako predchádzajúci bod týkajúci sa myšlienkových máp dôležité najmä z dôvodu ohrození vyplývajúcich zo samotnej transformácie a jej dostatočnej špecifikácie.

Porovnávanie čiastkových modelov. Pri tvorbe požiadaviek v špecifikovaných tvaroch sa požiadavky rozdeľujú do kategórií na základe príslušnosti k určitému typu kľúčovej otázky. Takéto porovnanie môže pomôcť k bližšiemu určeniu problémových požiadaviek, pri ktorých dochádza k častejším problémom v transformácii. Naopak môže napovedať, nakoľko sú niektoré iné požiadavky spoľahlivo transformovateľné.

V myšlienkových mapách sa môžu objaviť nejednoznačnosti, ktoré ovplyvnia finálne výsledky transformácie. Takéto výsledky môžu vyplývať z nevhodnej formulácie danej požiadavky, prípadne z dôvodu nedostatočnej úspešnosti odhaľovania uvažovaných slovných druhov a foriem použitím súčasných metód spracovania prirodzeného jazyka. Takéto formy požiadaviek je v takom prípade vhodné neuvažovať a porovnať model bez ich zohľadňovania.

Porovnanie nejednoznačných modelov po odstránení nejednoznačností je takisto veľmi vhodnou alternatívou, ktorá súvisí s predchádzajúcim bodom. Podobne ako v ňom sme odstraňovali problém jeho zanedbaním, v tomto porovnaní je možné daný model porovnať až po úprave nejednoznačných častí na časti s rovnakým významom, ktoré však nevytvárajú tieto zmieňované problémy.

Porovnanie viacnásobného použitia transformácii tam a späť. Podobne ako pri transformáciách medzi myšlienkovou mapou do UML a späť môžu nastať problémy, rovnaké problémy sa viac alebo menej môžu prejaviť aj pri opakovaní tohto postupu viackrát za sebou. Najhorší scenár takejto úvahy by mohol smerovať až k postupnému zániku modelu. Všeobecne by však transformačné pravidlá mali zaistiť dostatočnú bezpečnosť, aby po prvotných stratách alebo nárastoch modelov nedochádzalo k ich ďalším zmenám.

Diskusia k rozšíreniu metamodelu myšlienkovej mapy. Zavedením pohľadov, a tým vrstveného systému myšlienkových máp, upravujeme jej základný metamodel o možnosti duplikovaného využitia rovnakých pojmov v rôznych vrstvách modelu a získavame tak možnosť prepájať ľubovoľné prvky myšlienkovej mapy medzi sebou pri zachovaní štandardnej acyklickej

stromovej hierarchie vizualizovanej myšlienkovej mapy. Takúto zmenu môžeme teda reprezentovať ako zmenu vzťahov medzi vrstvou mapy, jej prvkami (vrcholmi mapy) a vzťahmi medzi nimi (hranami).

Možnosti aplikácie takto definovaných požiadaviek na rôzne druhy UML diagramov. V rámci špecifikácie UML 2.5 existuje celkovo až 12 štruktúrnych a 11 behaviorálnych typov alebo podtypov diagramov. Nad nimi je možné sledovať rôznu mieru ich využitia v rámci navrhnutého prístupu zachytávania a transformácii požiadaviek. Pri delení diagramu na kľúčové otázky v nich priamo môžeme vyhľadávať časti, ktoré priamo odkazujú na použitie konkrétneho diagramu. Niektoré diagramy takýto priamy vzťah so špecifikovaným prístupom nemajú, avšak je možné hľadať aspoň ich čiastkové uplatnenie. Iné typy diagramov v navrhnutom prístupe nemajú uplatnenie a ich využitie súvisí s inou formuláciou požiadaviek, prevažne v textovej podobe.

Porovnanie s existujúcimi prístupmi. V analýze boli zrhnuté niektoré dôležité práce týkajúce sa modelovania pomocou myšlienkových máp. Z hľadiska zachytávania požiadaviek sme odhalili dva hlavné výskumy, ktoré sú po obsahovej stránke najbližšie k nášmu prístupu. Pre získanie kontextu výhod a nevýhod týchto prístupov preto poslúži porovnanie.

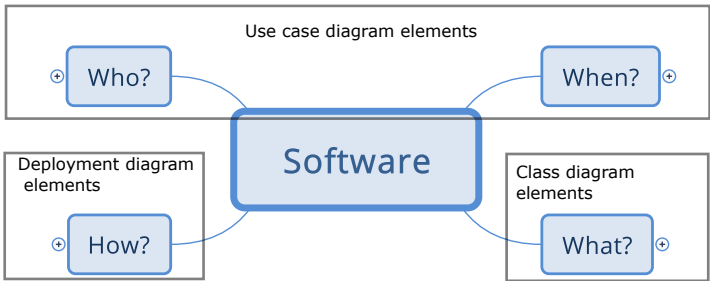
15.2 Vyhodnotenie scenárov

V tejto časti sa nachádza sumarizácia evaluácie podľa definovaných evaluačných scenárov v predchádzajúcej časti 15.1.

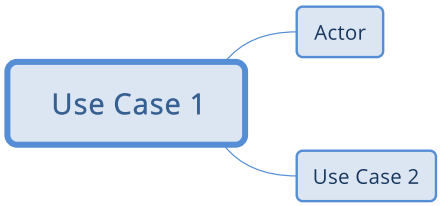
15.2.1 Porovnávanie transformovaných myšlienkových máp do UML voči očakávaným výsledkom v UML

V prípade, že je naším cieľom modelovať požiadavky v podobe myšlienkovej mapy a následne ich transformovať do UML, môžeme na základe navrhnutého prístupu rátať so získaním viacerých prvkov a druhov diagramov. Vďaka základnej štruktúre dokážeme získať základné elementy 3 druhov UML diagramov, čo je možné vidieť na obrázku 15.1. Konkrétne ide o triedy a atribúty diagramu tried a dedičnosť tried navzájom. V prípade častí vo vetvách myšlienkovej mapy Kto? a Kedy? dokážeme získať hráčov systému a prípady použitia. V prípade hráčov dokážeme získať aj ich ďalšiu generalizáciu a v prípade prípadov použitia rovnako aj ich vzájomné vzťahy extend. Na základe časti Ako? získame základné komponenty pre diagram nasadenia a asociácie previazaných prvkov.

Využitím textovej analýzy dokážeme získať rovnako aj stavy entít (tried alebo ich atribútov). Získame tak základ diagramu stavov. Vďaka využívaniu vrstiev je možné získať aj ďalšie vzťahy pre asociované triedy, asociácie medzi hráčmi a prípadmi použitia, include vzťahu pre prípady použitia a základ sekvenčného diagramu alebo diagramu kompozitnej štruktúry. Tieto vzťahy



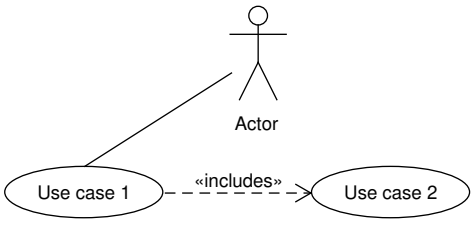
Obr. 15.1: Ukážka základného princípu transformácie



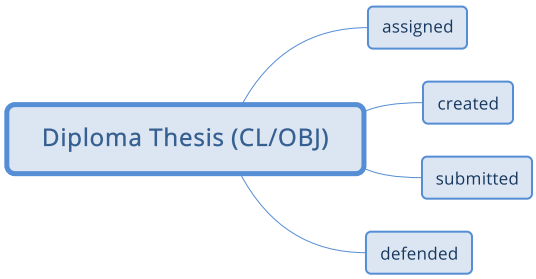
Obr. 15.2: Ukážka mapovania asociácie hráčov a include prípadu použitia

je možné vysledovať na modeloch na obrázkoch 15.2, 15.3, 15.4, 15.5, 15.6, 15.6 a 15.7.

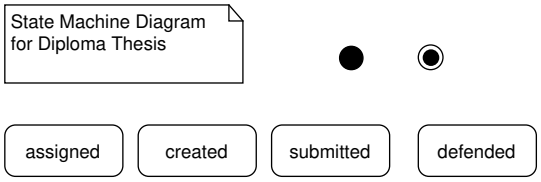
Tento smer mapovania tak dokáže zachytiť požiadavky transformovateľné do viacerých typov diagramov, respektíve ich častí. Ak by sme však požiadavky chceli zachytiť štandardnou formou a následne vytvoriť diagramy, získali by sme aj ďalšie časti. Príkladom môžu byť iné typy asociácii v diagrame tried, agregácia a kompozícia tried. V prípade diagramu stavov by bolo možné modelovať aj prechody medzi jednotlivými stavmi. Pre sekvenčný diagram sú základom hlavne vzájomné vzťahy medzi lifelinami, ktoré z myšlienkovej mapy nedokážeme získať. Podobne by sme mohli špecifikovať aj ďalšie rozdiely vyplývajúce z rozdielov vo vizualizačnej sile, ktorú sme schopní získať z navrhnutej štruktúry myšlienkovej mapy a štandardnej



Obr. 15.3: Transformovaná myšlienková mapa 15.2



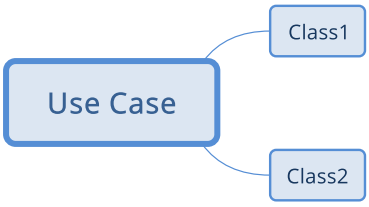
Obr. 15.4: Ukážka mapovania stavov na typ



Obr. 15.5: Transformovaná myšlienková mapa 15.4

množiny prvkov UML modelov.

Ak prejdeme množinou základných najdôležitejších prvkov a vzťahov v UML diagramoch, dostaneme sa k priamemu zhodnoteniu možností. V prípade diagramu tried k takýmto prvkom a vzťahom patria triedy, atribúty, asociácie a generalizácie, ktoré dokážeme v prístupe modelovať. K iným základným prvkom patria však aj metódy, agregácie, kompozície, realizácia, závislosť alebo kolaborácia. Zhrnutie takýchto základných elementov je v tabuľke 15.1. Takýmto spôsobom môžeme sledovať aj prvky pre iné typy zapojených diagramov, s ktorými sa počíta pri modelovaní myšlienkových máp. K nim patrí diagram prípadov použitia zhrnutý v tabuľke 15.2, diagram nasadenia v tabuľke 15.3, stavový diagram 15.4, sekvenčný diagram v tabuľke 15.5 a diagram kompozitných štruktúr pre realizáciu prípadu použitia v tabuľke 15.6.



Obr. 15.6: Ukážka mapovania tried a prípadu použitia

Diagram tried	
Zachytené základné prvky a vzťahy	Chýbajúce základné prvky a vzťahy
trieda atribút asociácia generalizácia	metóda agregácia kompozícia závislosť realizácia kolaborácia

Tabuľka 15.1: Tabuľka zhrnutia prvkov diagramu tried voči prístupu

Diagram prípadov použitia	
Zachytené základné prvky a vzťahy	Chýbajúce základné prvky a vzťahy
prípad použitia hráč systému asociácia include extend	extension point generalizácia

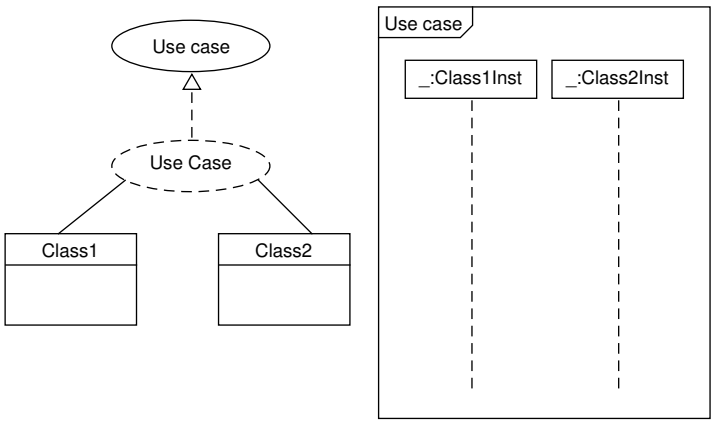
Tabuľka 15.2: Tabuľka zhrnutia prvkov diagramu prípadov použitia voči prístupu

Diagram nasadenia	
Zachytené základné prvky a vzťahy	Chýbajúce základné prvky a vzťahy
komponent asociácia	typy komponentov

Tabuľka 15.3: Tabuľka zhrnutia prvkov diagramu nasadenia voči prístupu

Diagram stavov	
Zachytené základné prvky a vzťahy	Chýbajúce základné prvky a vzťahy
stav štart/koniec	vykonávanie v stave prechod vetvenie

Tabuľka 15.4: Tabuľka zhrnutia prvkov diagramu stavov voči prístupu



Obr. 15.7: Transformovaná myšlienková mapa 15.6

Sekvenčný diagram	
Zachytené základné prvky a vzťahy	Chýbajúce základné prvky a vzťahy
lifeline	posielanie správ

Tabuľka 15.5: Tabuľka zhrnutia prvkov diagramu sekvencií voči prístupu

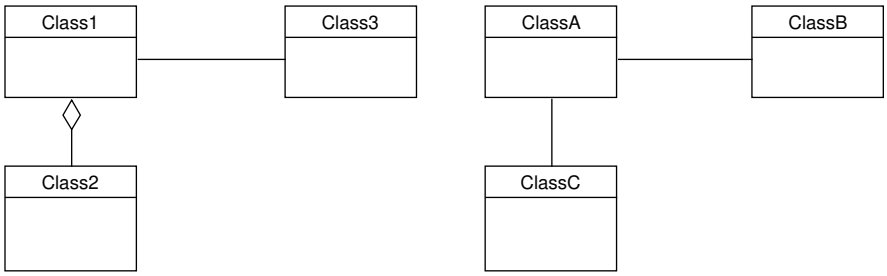
Na základe tohto porovnania dokážeme ohodnotiť vhodnosť využívania myšlienkovej mapy na modelovanie konkrétneho druhu UML diagramov. Z tohto porovnania sa tak ukazujú ako vhodnejšie diagramy prípadov použitia, diagram nasadenia, diagram kompozitnej štruktúry a diagram tried pre modelovací prístup, než snaha o zachytávanie správania v sekvenčných diagramoch alebo diagramoch stavov.

Diagram kompozitných štruktúr ako realizácia prípadu použitia	
Zachytené základné prvky a vzťahy	Chýbajúce základné prvky a vzťahy
kolaborácia trieda (objekt) závislosť	na tento účel nič konkrétne

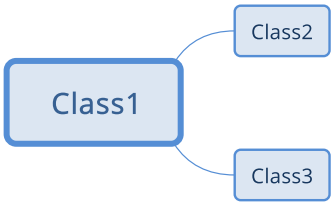
Tabuľka 15.6: Tabuľka zhrnutia prvkov diagramu kompozitných štruktúr voči prístupu

15.2.2 Porovnávanie transformovaných UML modelov v tvare myšlienkových máp

V prípade UML modelov postupujeme v transformácii opačne ako v prípade vyššie. Pracujeme nad väčšou množinou prvkov, ktorú transformujeme do špecifikovanej štruktúry diagramu. Dokážeme tak transformovať viacero elementov. Avšak po transformácii do určitej miery strácame niektoré jeho časti. Niektoré neznáme elementy nedokážeme využiť v myšlienkovej mape vôbec. Iné, najmä vzťahy medzi známymi elementami, sa môžu zmeniť na iný typ. Napríklad agregácie a kompozície v diagrame tried. Tie sa ďalej dostanú v rovnakom tvare ako štandardná asociácia. Príklad tohto modelovacieho problému v transformáciách je možné vidieť na obrázku 15.8, kde sú dve samostatné prípady prepojenia tried. Ich transformáciou do korešpondujúcich myšlienkových máp získame modely zobrazené na obrázkoch 15.9 a 15.10.

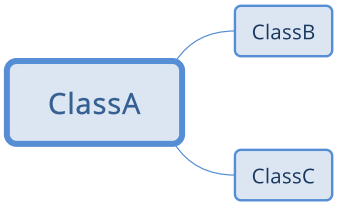


Obr. 15.8: Ukážka problému v diagrame tried



Obr. 15.9: Transformovaný model pre triedy 1, 2, 3 z 15.8

Získané modely sú tak zjednodušené, čo môže spôsobiť stratu dôležitých elementov. V prípade skreslených elementov je dané skreslenie, z pohľadu zrozumiteľnosti modelu pre odborníka mimo softvérové inžinierstvo, žiadané. Ten nedokáže často rozlišovať detaily a zameriava sa skôr na všeobecné vzťahy ako je súvis (obsahuje, je vo vzťahu s) a konkretizácia (je rodič, je potomok typu). Diferenciácia na vzťahy ako sú agregácia a kompozícia sú už úplne mimo základného uvažovania o entitách v doménovej



Obr. 15.10: Transformovaný model pre triedy A, B, C z 15.8

oblasti. Ide už o technické detaily, ktoré nemusia byť zrozumiteľné a sú skôr záležitosťou návrhu.

Transformované UML modely v tvare myšlienkových máp sa tak môžu výrazne líšiť. V prípade práce len so známymi prvkami a vzťahmi získame úplnú reprezentáciu modelu v tvare myšlienkovej mapy. So zvyšovaním neznámych prvkov sa však úspešnosť zahrnutia celého modelu stráca.

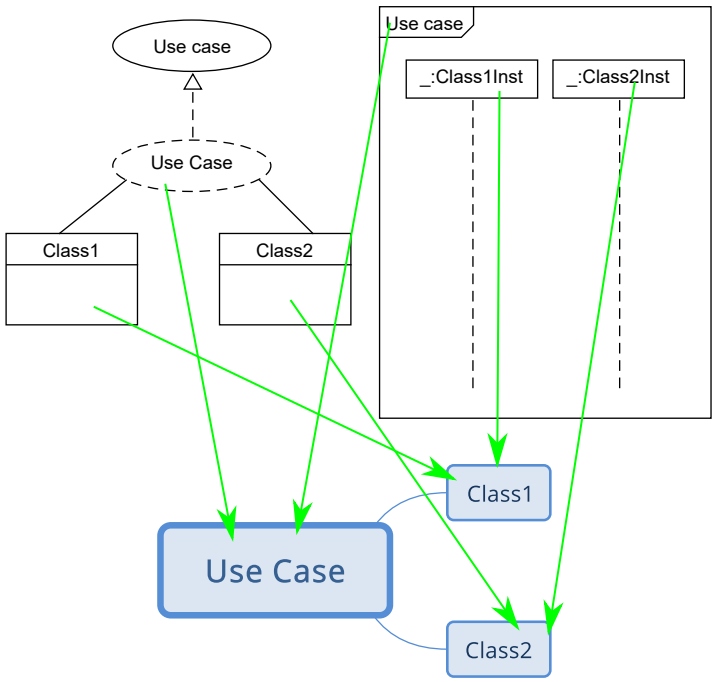
15.2.3 Myšlienkové mapy porovnávané po transformácii do UML a späť do pôvodného tvaru

Navrhnuté transformačné pravidlá sú vypracované tak, aby vygenerovaný UML model obsahoval všetky zachytené prvky v podobách, ktoré dokážeme zachytiť. V prípade spätnej transformácie tak transformujeme len známe elementy UML, s ktorými sme už predtým dokázali pracovať. Preto pri spätnej transformácii do myšlienkovej mapy nedochádza k ďalšiemu skresleniu. Dôjsť môže len k zmene poradia transformovaných vrstiev, čo však nie je z pohľadu modelovania kľúčové.

K problému by došlo iba v prípade, že výsledok prvej transformácie by sme ďalej upravovali. Bez toho je model dostatočne zrozumiteľný pre nástroj, pretože pracuje na základe pravidiel pre každý typ transformovaného elementu. Transformácie tak fungujú na princípe: 1 a viac prvkov UML sa transformuje do 1 prvku myšlienkovej mapy. Viac prvkov sa transformuje do jedného v prípade, že reprezentujú rovnakú záležitosť, len z rôznych pohľadov. K takýmto patrí napríklad stavový diagram ako taký, lifeline pre sekvenčný diagram konkrétneho prípadu použitia, prípadne náležitosti týkajúce sa diagramu kompozitných štruktúr. Takýto prípad môžeme vidieť na obrázku 15.11. Vzťah transformovaných modelov je tak možné vizualizovať ako na obrázku 15.12.

15.2.4 UML modely transformované do myšlienkovej mapy a späť do UML

Táto transformácia je opakom k predchádzajúcej transformácii. A to nielen z pohľadu samotného princípu. Pri transformácii sa mení UML model, ktorý



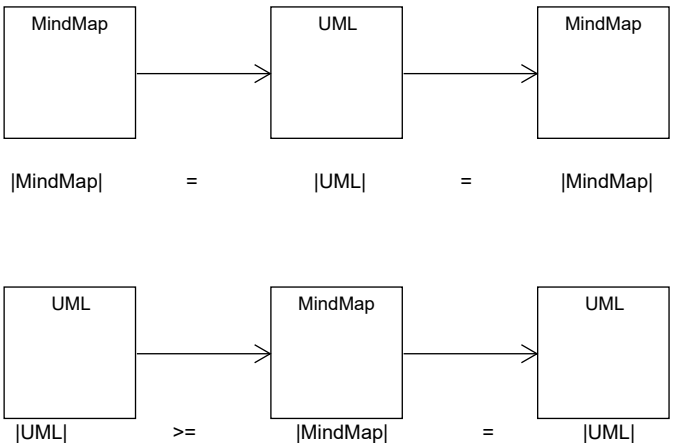
Obr. 15.11: Transformácia viacerých UML elementov do jedného prvku v myšlienkovej mape

má viacero rôznych druhov elementov do jednoduchšej podoby. Pri tom sa pracuje so stratovosťou, ktorú sme spomínali v časti 15.2.2. V tomto prípade takto vytváraná myšlienková mapa skreslí niektoré vzťahy a nezachová niektoré neznáme prvky UML. Pri spätnej transformácii tak dochádza ku generovaniu diagramu, ktorý už nekorešponduje s pôvodným. Niektoré elementy zmiznú a niektoré vzťahy zmenia typ. Transformácia tak pôvodný model mení na zjednodušený, v najlepšom prípade sa zachová v rovnakej podobe. To však musí znamenať, že vstupný model nebude obsahovať žiadne neznáme prvky ani nejednoznačné vzťahy. Príklad obmeny vzťahu je možné vidieť na obrázku 15.13. Vzťah medzi entitami a vzťahmi v pôvodnom diagrame, myšlienkovej mape a transformovanom modeli je vidieť v spodnej časti obrázku 15.12.

15.2.5 Porovnávanie čiastkových modelov

Ak sa na myšlienkovú mapu pozeráme z pohľadu jej jednotlivých častí, môžeme odhaliť niektoré vlastnosti konkrétnych modelovaných UML prvkov a diagramov. To súvisí s evaluáciou možných elementov UML zachytených pomocou myšlienkových máp.

V prípade časti Kto? a Kedy? sa elementy transformujú do diagramu prípadov použitia. Pre tento typ diagramu sa z pohľadu základných vzťa-



Obr. 15.12: Vzťah transformovaných modelov tam a späť pre myšlienkovú mapu a UML

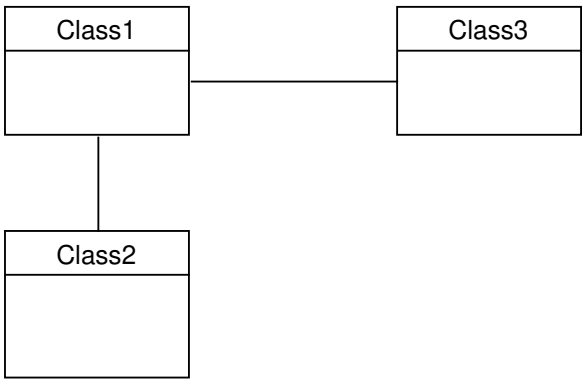
hov dokážeme dostať k väčšine uvažovaných prvkov. To je opakom k časti Čo?, v ktorej zaznamenávame elementy diagramu tried. Pre ten sa nám nedarí zachytiť viacero druhov prvkov a vzťahov v očakávanej miere. V časti Ako? sa nám diagram podarí zachytiť dostatočne, nezachytávame síce typy komponentov, ale to z pohľadu doménového odborníka nie je kľúčové.

Na základe tohto pozorovania môžeme usúdiť, že myšlienková mapa sa dá s výhodou využiť najmä na plnú definíciu komponentov systému a charakterizáciu prípadov použitia systému. V prípade diagramu prípadov použitia je možné pozorovať podobnosť základnej modelovacej techniky. V oboch prípadoch sa najčastejšie ako základné elementy používajú elipsy. V myšlienkovej mape však kvôli jej acyklickosti musíme zachytiť niektoré vzťahy v iných vrstvách. Z pohľadu zrozumiteľnosti sa tak myšlienková mapa ukazuje ako vhodná alternatíva pre zakresľovanie prípadov použitia a hráčov systému k zaužívanému diagramu prípadov použitia.

V prípade diagramu tried nedokážeme zachytiť všetky vzťahy a konkrétne metódy tried. Táto záležitosť však rovnako nie je natoľko zaujímavá pre doménového experta, ktorý sa v architektonicko-implementačných detailoch informačných systémov nevyzná.

Najvhodnejšie časti na zachytávanie pomocou myšlienkových máp sú teda:

- Diagram nasadenia (prípadne základný diagram komponentov)
- Diagram prípadov použitia
- Diagram kompozitnej štruktúry pre realizáciu prípadu použitia
- Diagram tried pre doménového experta



Obr. 15.13: Ukážka po transformácii späť diagramu 15.8

Nevhodné pre zachytávanie čisto pomocou myšlienkových máp:

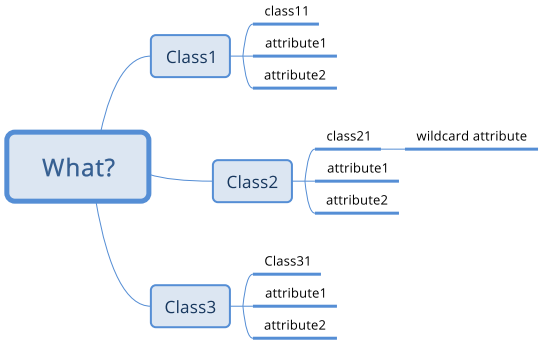
- Diagram sekvencií
- Diagram stavov

Tieto vyžadujú pre plnú definíciu ešte iný tvar zachytených požiadaviek – najmä textovú podobu. To však následne môže prístup dostávať do pozície vytvárania duplicít v zachytávaní a modelovaní.

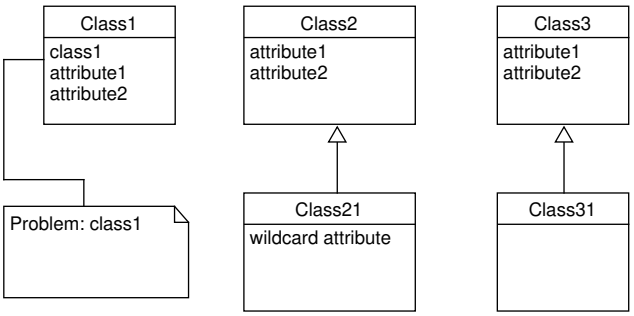
15.2.6 Scenáre nejednoznačnosti myšlienkových máp

V prípade modelovania požiadaviek v myšlienkových mapách sa môžu objaviť aj prípady, kedy niektoré elementy nedokážeme jednoznačne identifikovať a transformovať ich tak do správnych typov. Táto nejednoznačnosť sa prejavuje v častiach, v ktorých sa pracuje nad textovou analýzou reťazcov v zachytených pojmoch mapy. Príkladom môžu byť zle odhalené atribúty a podtriedy. Takýto prípad je zobrazený na obrázku 15.14 a transformovanom modeli 15.15. Tento rozdiel sme postavili na myšlienke vlastných mien tried začínajúcich veľkým písmenom. V prípade, že nedokážeme takéto rozlíšenie charakterizovať, môžeme vnímať každý pojem bez ďalších podpojmov ako atribút. To však spôsobuje problém v prípade tried, ktoré budú v systéme vystupovať, avšak nedefinovali sme im nateraz konkrétne atribúty. Transformáciou tak môžeme stratiť niektoré podtriedy a získať nový prebytočný atribút, alebo naopak získať ďalšiu triedu navyše. Pre odstránenie tohto problému je potrebné využívať vždy pravidlo kapitalizácie názvov tried a atribúty zachytávať len malými písmenami. Inou možnosťou je zapísanie nového zástupného atribútu pre triedu bez odhalených atribútov. Tento atribút môže poslúžiť ako podklad pre uvažovanie o potrebe pridania nového

prvku, avšak na druhej strane môže pôsobiť aj zmätočne a zahlcujúco, najmä ak je takýchto tried bez atribútov viacero.



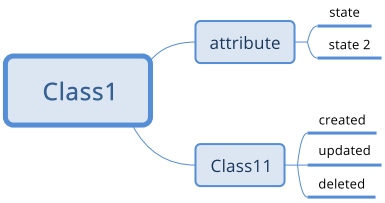
Obr. 15.14: Problém s nejednoznačnosťou tried a atribútov v myšlienkovej mape



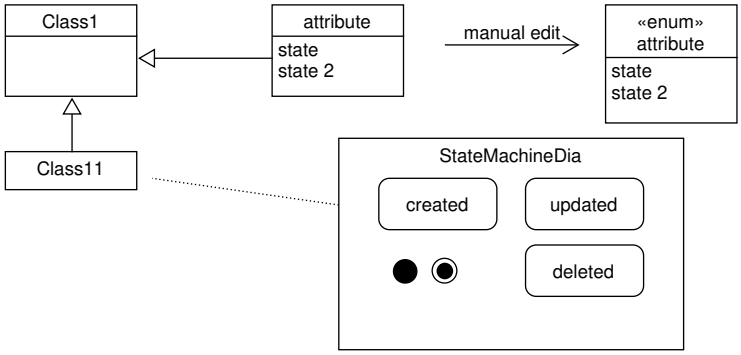
Obr. 15.15: Výsledok transformácie nejednoznačnosti atribútov a tried

Iným prípadom nejednoznačnosti je prípad, v ktorom sa objavia nesprávne odhalené typy slovných druhov. Takýto príklad je možné vidieť na obrázku 15.16 a transformovanom modeli na obrázku 15.17. To sa objavuje v prípade snahy odhalenia stavov pre opis stavového priestoru entity. Stavy vo všeobecnosti očakávame v tvare prídavného mena, prípadne tvaru, ktorý je úzko súvisiaci s prídavným menom. Ak sa stane, že stav budeme charakterizovať iným spôsobom, nepodarí sa nám stav odhaliť. Naopak stavy sa prejavia ako podtriedy alebo atribúty triedy. V prípade atribútu s vymenovanými hodnotami svojho stavu sa taktiež môže stať, že sa stane triedou a stavy jeho atribútmi. To môže byť využiteľné, ak by sme plánovali neskôr triedu charakterizovať ako enumeračný typ. Všeobecne sa však dá očakávať, že takýto prípad negatívne ovplyvní výsledný model. Preto je potrebné nahradiť názvy stavov iným synonymickým opisom, ktorého základom je

prídavné meno.



Obr. 15.16: Modelovanie stavov pre triedy v myšlienkovej mape a problémy



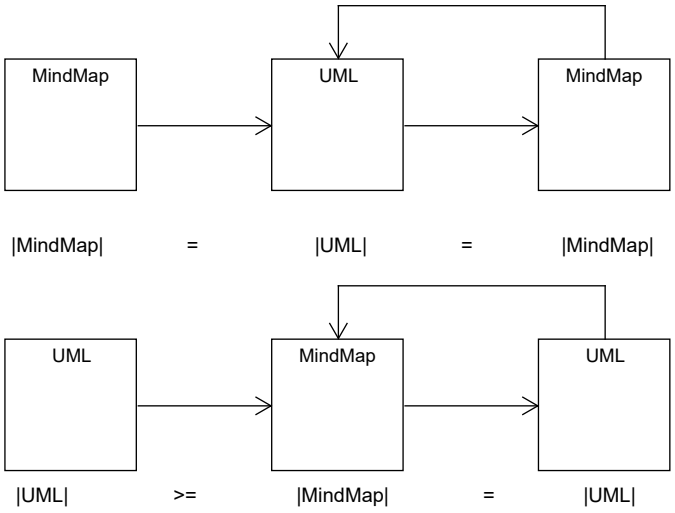
Obr. 15.17: Výsledok transformácie nejednoznačnosti stavov

15.2.7 Porovnanie viacnásobného použitia transformácií tam a späť

V prípade viacnásobných transformácií sme sa snažili v návrhu pravidiel o to, aby sme prípadné skreslenia už ďalej nerozširovali. Skreslenie získané v prvej fáze transformácie sa tak pri opakovanom postupe už len opakuje a nedochádza k jeho zvýrazňovaniu. V prípade transformácií so začiatočným modelom v myšlienkovej mape sa tak ani vplyvom viacerých transformácií neobjavia vynechané prvky a vzťahy. V prípade vstupného modelu v UML sa po prvotnej transformácii, pri ktorej môže dôjsť ku skresleniu, ďalej neprejaví ešte negatívnejšie ako prechádzajúca. Výsledný proces mapovania prvkov a vzťahov v UML modeloch a myšlienkových mapách je tak možné vidieť na obrázku 15.18.

15.2.8 Diskusia k rozšíreniu metamodelu myšlienkovej mapy

Počas výskumu možností modelovania požiadaviek v myšlienkových mapách sme sa zamerali aj na otázku rozšírenia modelovacej techniky. Jednoduché,

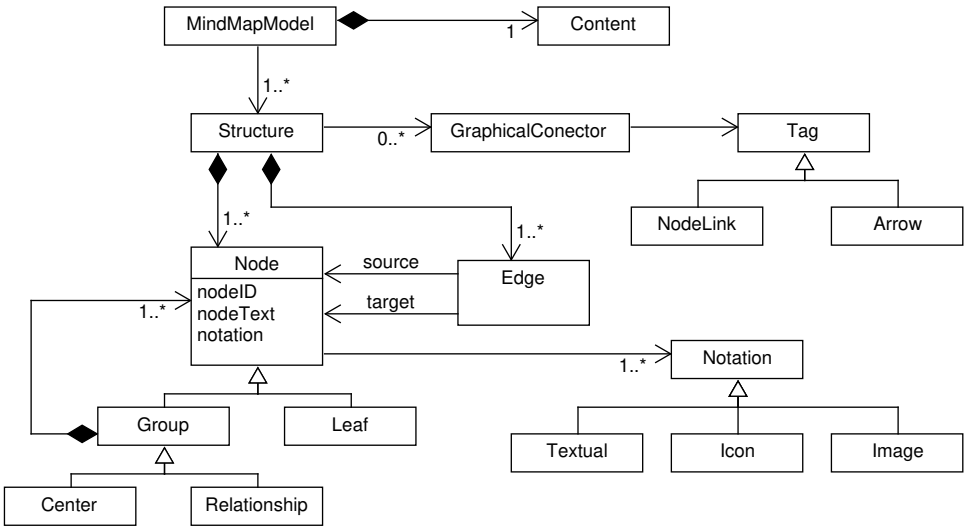


Obr. 15.18: Vzťah transformovaných modelov tam a späť viackrát

avšak veľmi vhodné rozšírenie sme našli v možnosti zavedenia vrstiev. Toto rozšírenie pritom vychádza z jednoduchej myšlienky myslenia v obrazoch [6], ohraničenia aktuálnej domény, nad ktorou momentálne pracujeme a prechodov do iných domén. V prípade, že sa človek sústreďuje pri myslení na jeden konkrétny fakt, dokáže vytvoriť sieť naviazaných pojmov, ktoré s ním súvisia, alebo ho nejakým spôsobom charakterizujú. V prípade, že sa v uvažovaní dostane na nejaký pojem, ktorý sa dostane do centra jeho záujmu, zvyšné prvky už nebudú natoľko zaujímavé, takže ich v myslení potlačí do úzadia. Naopak, akonáhle sa zmení cieľ uvažovania, začnú sa objavovať iné vzťahy, ktoré predtým boli skryté, pretože nemali priamy súvis s predchádzajúcim centrom záujmu. Takto môžeme charakterizovať jednoduché premýšľanie nad konkrétnymi pojmami z pohľadu vizuálneho myslenia. Práve to je základom modelovacej techniky myšlienkových máp. Využíva sa pri tom fakt, že veľká časť populácie využíva vizuálne myslenie ako jeden zo základných spôsobov uvažovania.

Ak sa pozrieme na tento spôsob prechodov medzi pojmami v centre záujmu z pohľadu požiadaviek, dostávame sa k nami zvolenej ceste v rámci špecifikovaného prístupu. V ňom sme navrhli základnú štruktúru myšlienkovej mapy a na previazanie pojmov, ktoré by narúšali acyklickosť sme sa rozhodli zmeniť centrum záujmu – hlavný pojem, nad ktorým modelujeme okolie. Táto vlastnosť funguje aj v bežnom hľadaní súvislostí v grafe pojmov v pamäti. Na jeden pojem je naviazaných viacero pojmov. Keď sa však pozrieme na tieto pojmy, môže sa stať, že aj medzi nimi je priame prepojenie, avšak nie v rámci uvažovaného centrálneho pojmu – ústrednej témy. Ak sa chceme pozrieť na ich vzájomný vzťah, musíme tak zmeniť práve ústrednú

tému.



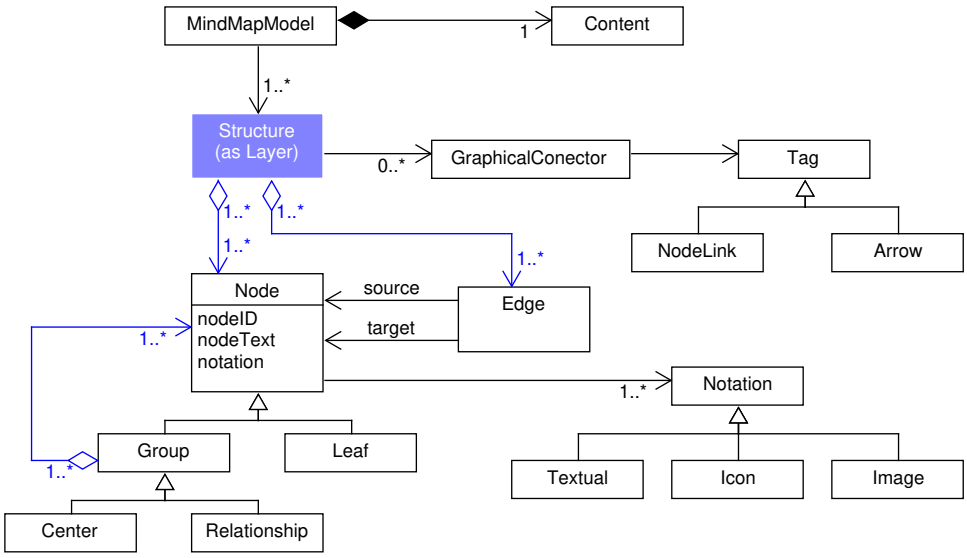
Obr. 15.19: Bežný príklad metamodelu myšlienkovej mapy [27]

Pri modelovaní myšlienkovej mapy postupujeme vždy podľa jej základných pravidiel. Tie vychádzajú z jej metamodelu. Na obrázku 15.20 vidíme príklad metamodelu myšlienkovej mapy. V metamodeli je vhodné zamerať sa na hárok. Každý hárok je vlastná myšlienková mapa, ktorá obsahuje vlastné prvky. Ak by sme chceli vytvárať myšlienkovú mapu celého grafu pojmov v pamäti, museli by sme vedieť vytvárať viacnásobné prepojenia, ktoré nie sú kvôli acyklickosti jednej mapy dovolené. Ak sa však pozrieme na hárok ako na vrstvu, ktorá sa zameriava vždy na inú ústrednú tému, môžeme znovupoužiť niektoré pojmy aj v iných vrstvách. Preto bude mať metamodel takto špecifikovanej štruktúry myšlienkových máp tvar ako na obr. 15.20. V tomto tvare je možné priraďovať jeden pojem viacerým vrstvám, a tak vytvárať aj nové vzťahy, ktoré by inak neboli v pôvodnom modeli možné.

Vďaka tomuto vytvárame nový tvar myšlienkovej mapy, ktorá má vrstvy, a pritom stále spĺňa základné pravidlá. Cyklickosť pojmov je totižto zachytená mimo konkrétneho hárku (vrstvy), ale v priestore medzi vrstvami. Príklad takejto kompletnej grafovej štruktúry je na obrázku 15.21.

15.2.9 Možnosti aplikácie požiadaviek definovaných v myšlienkovej mape na tvorbu rôznych druhov UML diagramov

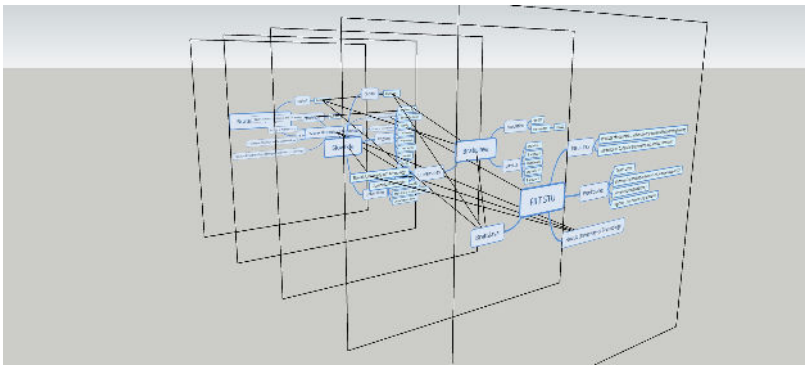
V rámci špecifikácie prístupu a jeho postupného rozširovania sa nám podarilo odhaliť viacero možných uplatnení zachytených prvkov vo viacerých



Obr. 15.20: Upravený metamodel myšlienkovej mapy

diagramoch. Z tohto hľadiska sa v rámci hierarchie UML diagramov dajú vysledovať viaceré vlastnosti, ktoré predurčujú využitie daného typu diagramu a možnosť zachytenia jeho jednotlivých elementov.

Základný prístup sa opieral o využitie troch kľúčových otázok. Z nich sme boli schopní získať diagram tried a prípadov použitia. Získali sme väčšiu časť najpoužívanejších konceptov týchto diagramov aj základné vzťahy. Z tohto pohľadu môžeme povedať, že tieto dva typy diagramov sa nám podarilo do prístupu zakomponovať v úplnej miere. V rozšírení prístupu sme vďaka odhaleniu kľúčovej otázky Ako? dokázali reprezentovať základné komponenty systému, ktoré je možné reprezentovať pomocou diagramu na-



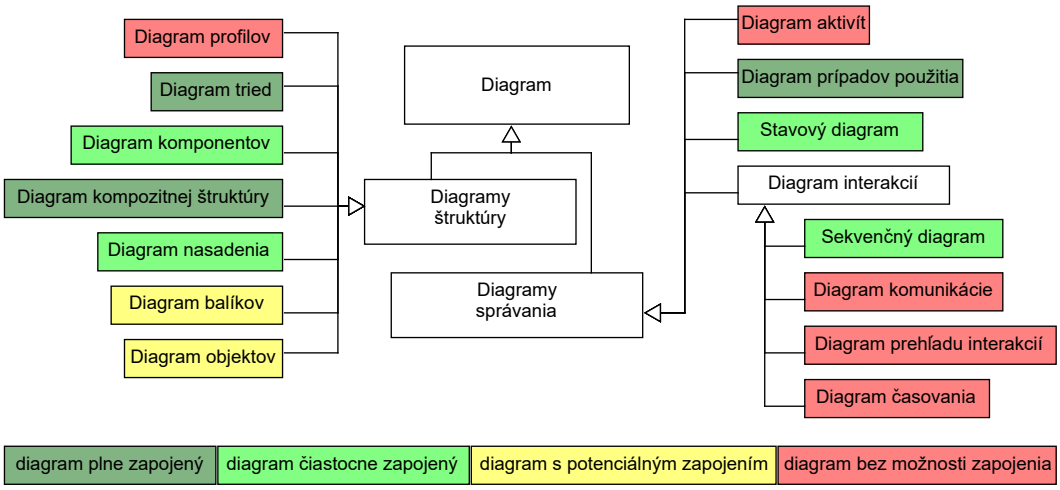
Obr. 15.21: Ukážka 3D vizualizácie myšlienkovej mapy

sadenia, prípadne pomocou diagramu komponentov. Z tohto vyplýva, že sa nám podarilo do prístupu čiastočne zapojiť aj tieto typy diagramov.

Okrem toho sme vďaka vrstveniu a analýze slov dokázali získať aj ďalšie prvky iných typov diagramov. Základným príkladom je diagram kompozitnej štruktúry, ktorý sme tak v plnej miere zapojili na modelovanie realizácie prípadu použitia. Z ďalších zapojených diagramov je možné spomenúť ešte stavový diagram a sekvenčný diagram, ktoré sme však dokázali využiť len v malej miere – len na zachytenie základných konceptov bez ich vzájomných prepojení.

Na základe tohto prehľadu je možné vidieť, že k najzapojenejším diagramom patria štruktúrne diagramy. Chýba len diagram profilov, ktorý sme na základe jeho analýzy vylúčili, diagram objektov a diagram balíkov, ktorých použitie by bolo možné ešte ďalej skúmať, avšak toto použitie nie je pre nás dostatočne zaujímavé.

Na strane diagramov správania sa nachádza dobre využitý len diagram prípadov použitia a čiastočné využitie môžeme nájsť v sekvenčnom diagrame a diagrame stavov. Ostatné diagramy najmä z hľadiska potreby rôznorodých prepojení konceptov nie sú vhodné.



Obr. 15.22: UML diagramy z pohľadu ich zapracovania do prístupu

Ak sa teda pozrieme na prístup a jeho možnosti generovania rôznych druhov diagramov, dostaneme sa k zapojeniu diagramov podľa obrázka 15.22. Z neho je možné dobre vidieť, že myšlienkové mapy sa ukazujú ako vhodné skôr na modelovanie štruktúry systému, než na jeho správanie. Pre zachytenie charakterizácie správania je potrebné využiť pri získavaní požiadaviek iný prístup. Myšlienkové mapy môžu byť len veľmi limitovane nápomocné.

15.2.10 Porovnanie s existujúcimi prístupmi

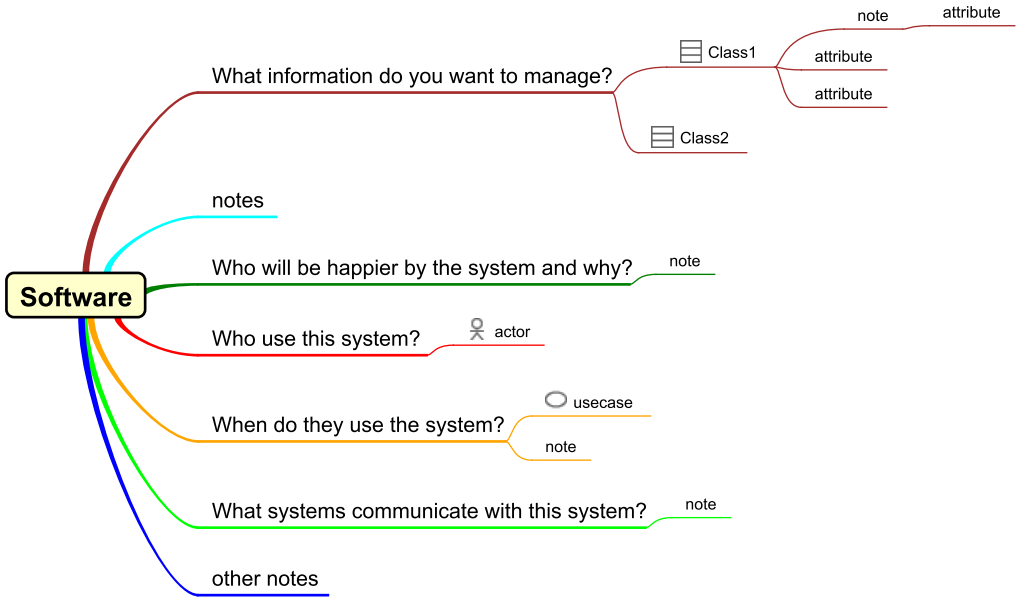
V tejto časti sa nachádza vyhodnotenie porovnania špecifikovaného prístupu s inými prístupmi na podobnom princípe, pochádzajúcimi z analyzovaných výskumov.

Porovnanie s prístupom ChangeVision – Astah

Prístup špecifikovaný spoločnosťou ChangeVision [15] je zahrnutý do ich modelovacieho nástroja Astah. Ten dokáže modelovať UML aj myšlienkové mapy. Prístup je založený na jednej myšlienkovej mape, ktorá pozostáva z viacerých vetiev. Takúto špecifikáciu obsahuje myšlienková mapa na obrázku 15.23. Centrálnym konceptom je samotný softvér, respektíve jeho požiadavky získané počas interview. Myšlienková mapa má prehľadné delenie na vetvy s podobne koncipovanými kľúčovými otázkami, ako má náš prístup. Okrem týchto vetiev obsahuje aj vetvy, ktoré sa pri modelovaní UML priamo nevyužijú, avšak môžu byť zaujímavé ako poznámky o kľúčových vlastnostiach systému, alebo informáciách o ďalšom stretnutí. Samotné elementy majú názvy, ktoré sú na podobnom princípe ako v špecifikovanom prístupe. Rozdiel však je v notácii jednotlivých prvkov.

Prístup ChangeVision vychádza pri transformáciách z doplnenia základného tvaru myšlienkovej mapy o notačné značky UML konceptov. Hráči systému tak majú vo svojej vetve ikonu, aká sa používa v UML. Podobne je to aj pri triedach a prípadoch použitia. Transformácia takéhoto diagramu do UML nie je automatizovaná. Voľný tvar myšlienkovej mapy nie je na to prispôsobený. V prípade, že je záujem preniesť takto zachytené požiadavky do UML, je potrebné elementy manuálne skopírovať do UML diagramu. V ňom sa elementy vďaka notačnému symbolu elementu automaticky zmenia na zodpovedajúci UML koncept. Neplatí to však už pre atribúty, ktoré takýto symbol nemajú. Vzťahy, ktoré sú charakterizované v myšlienkovej mape nemajú žiadny vplyv na UML model a musia byť vytvorené nanovo. Vzhľadom na to, že takúto informáciu myšlienková mapa nemá, je takto špecifikovaný model pri transformácii nepoužiteľný bez ďalšieho dodaného materiálu, pravdepodobne textu zo stretnutia.

Prístup je tak obmedzený len na zachytenie jednoduchých želaní zákazníka z pohľadu požiadaviek na finálny produkt. Nie je možné ho použiť na generovanie UML diagramov ani na prechod od UML do myšlienkových máp tak, aby diagramu zákazník lepšie porozumel. Počet zachytávaných typov elementov je limitovaný na použitie typov ako v našom prístupe. Nástroj obsahuje notačné symboly aj pre iné typy UML konceptov, avšak z hľadiska návrhu základnej štruktúry myšlienkovej mapy nemajú svoje určené miesto a ich transformácia do UML je rovnako ťažkopádna.



Obr. 15.23: Základ modelu v prístupe ChangeVision

Porovnanie s prístupom MindDomain

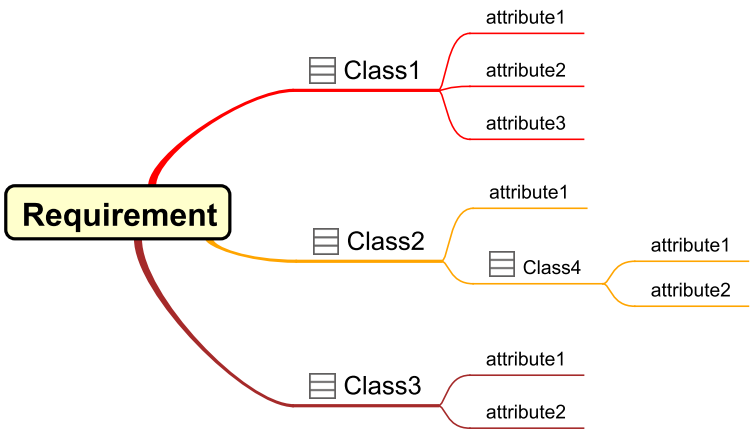
Prístup navrhnutý v prácach [29, 28] a najmä práci [7] je navrhnutý z hľadiska použiteľnosti v transformáciách do UML. Zameriava sa však len na transformáciu týmto smerom, nie aj na opačnú. Jednoduchosť navrhnutých transformácií však predznamenáva, že návrh opačnej transformácie by nemal byť zložitý. Prístup sa opiera o myšlienkovú mapu v tvare, v ktorom je centrálnym elementom konkrétna jednoduchá požiadavka a jej okolie. Príklad základnej štruktúry takéhoto modelu je na obrázku 15.24. Vo vetvách centrálneho elementu sa vždy nachádzajú súvisiace triedy. Pre tieto triedy sú následne v ich podprvkoch zachytené ich atribúty, prípadne asociované triedy.

Pre rozlišovanie tried a atribútov sa používajú rovnaké notačné ikony ako v prístupe od ChangeVision. Tieto značky sú potrebné najmä na diferenciáciu atribútu a asociovanej triedy. Vzhľadom na to, že asociované triedy by mali mať vždy aj charakterizáciu týchto elementov, pôsobí takéto členenie značkami zbytočne. Prístup v tomto tvare by pri tomto uvažovaní o triedach a atribútoch mohol pracovať podobne ako v špecifikácii nášho prístupu, kde v prípade rozlišovania atribútov a tried, sú všetky koncové elementy bez potomkov transformované do atribútov. V našom prípade sa však snažíme využívať koncepciu zovšeobecnenia a konkretizácie, takže triedy v podprvkoch dedia od svojho rodiča. V tomto prípade sú naopak vytvárané asociácie.

Prístup sa zameriava len na využitie tejto formy myšlienkovej mapy a

jej transformácie. Transformačné pravidlá si pomáhajú notačnými ikonami a výsledkom je diagram tried, v ktorom je transformovaný aj centrálny združujúci element do triedy. Tá následne agreguje svojich potomkov.

Tento prístup tak pracuje len s jediným typom UML diagramu – diagramom tried, pričom transformuje len triedy, atribúty a asociácie medzi nimi. Tie sú následne agregované požiadavkou. Prístup sa ďalej nezameriava na ďalšie využitie tried pre diagramy sekvencie alebo diagramy kompozitných štruktúr.



Obr. 15.24: Základ modelu v prístupe MindDomain

Zhrnutie porovnania podobných prístupov

V tejto tabuľke je možné nájsť prehľad porovnania podobných prístupov k modelovaniu a transformácii požiadaviek v myšlienkových mapách.

Prístup	Počet zachytávaných elementov	Počet spracovávaných vzťahov	Automatické generovanie	Transformácie v oboch smeroch	Počet zasiahnutých UML diagramov
ChangeVision	2	0	nie	nie	2(-?)
MindDomain	2	1	áno	nie	1
navrhnutý prístup	8	7	áno	áno	3(+3)

15.3 Zhodnotenie a sumarizácia

Na základe uskutočnenej evaluácie sa nám podarilo odhaliť silné a slabé stránky prístupu. Ten dokáže pracovať v rámci transformácií s viacerými druhmi prvkov a vzťahov UML. Tie je schopný transformovať obojsmerne, prípadne ich skresľuje do vhodnej formy. Z pohľadu existujúcej notácie UML sa zameriava na ich hlavné časti, ktoré sú v jednotlivých diagramoch najviac využívané. Tieto notačné symboly pritom patria do kategórií, ktoré sú pre doménového experta zaujímavejšie a dokáže s nimi na určitej úrovni pracovať. Rôzne typy transformácií dokážu vstupný model spracovať v plnej miere, alebo vynechajú či skreslia niektoré jeho časti.

Z porovnania existujúcich výskumov a ich prístupov na podobné zachytávanie požiadaviek sme zistili, že navrhnutý prístup má svoje podobné vlastnosti s týmito prístupmi, avšak má aj viacero ďalších zaujímavých vlastností. Najmä z pohľadu kompletnosti zachytávania požiadaviek je navrhnutý prístup výrazným krokom vpred. Dokáže pracovať s výrazne väčšou sadou diagramov a väčšou množinou notačných symbolov UML.

Dôležitou súčasťou, ktorá pomáha ku zvýšeniu kvality výsledkov, je zavedenie vrstveného modelu myšlienkovej mapy. Práve vďaka nej sme dokázali výrazne zvýšiť schopnosti myšlienkových máp bez potreby výrazne upravovať jej notáciu a základné vlastnosti. Vychádzame pri tom z jednoduchých princípov uvažovania v obrazoch, ktoré boli aj základom pre vytvorenie samotnej konceptuálnej modelovacej techniky.

Kapitola 16

Súvisiace práce

V priebehu analýzy možností tohto výskumu bolo odhalených viacero potenciálnych prístupov k využívaniu konceptuálnych modelov. Pre jednotlivé zamýšľané oblasti využitia týchto modelov boli odhalené rozdielne práce, ktorých výsledky sú pre túto prácu zaujímavé. Časť z takýchto prác sa nevenuje priamo využívaniu konceptuálnych modelov, ale ich súvisiacich problémových domén.

Základ poznávania konceptuálnych modelov, ich vplyvu a ich výhod ako modelovacích jazykov, je kvalitne zachytený v práci [22]. Práca potvrdzuje pôvodné stanovené hypotézy o možnostiach uplatnenia konceptuálnych modelov. Na konceptuálne modely je nazerané cez rôzne využívané pohľady. Za zmienku stojí aj práca [8], ktorá je postavená na snahe prepájania konceptuálnych modelov s ontológiami. Výhody takýchto úvah sú v práci poukázané na príklade kognitívnych máp. Zlúčenie princípov konceptuálneho modelovania a ontológií môže prispieť k rozšíreniu vyjadrovacej sily zlúčením výhod, ktoré tieto metódy poskytujú.

Okrem týchto prác je dôležité neopomenúť práce, ktoré boli v určitej forme spomenuté v ústrednej časti tejto práce. Viaceré z týchto prác sa zaoberajú možnosťami využitia konceptuálnych modelov v prostredí Requirements engineeringu, teda zberu požiadaviek. K takýmto prácam patrí napríklad práca [3], prípadne [29], ktoré sa venujú analýze možností mapovanie požiadaviek od zákazníkov vo vhodnej forme. Druhá spomínaná práca sa zároveň snaží o priblíženie možností využitia konceptuálnych modelov v prostredí modelom riadeného vývoja softvéru. Pre analyzované nedostatky v schopnostiach niektorých vybraných konceptuálnych techník, ktoré vyplývajú z ich vyjadrovacej sily, existujú viaceré výskumy [11] [20]. Ich výsledky poukazujú na možnosti rozširovania reprezentácií a možnosti využívať základné modelovacie techniky iným spôsobom, než je ich klasické využitie.

Ako dôležité pre ďalšiu prácu, boli určené aj možnosti transformácií modelov medzi modelmi s rôznou expresívnosťou. Analýzou boli potvrdené rozdiely medzi klasickými modelovacími technikami na modelovanie softvéru a

konceptuálnym modelovaním. Problémy rozdielnej expresívnosti nie sú novým problémom. Vyskytuje sa aj v prácach [19], [33] a [32]. Nad modelmi s rôznou vyjadrovacou silou špecifikovali možnosti reprezentácie konkrétnych elementov, ktoré nenachádzajú v jazyku s menšou expresívnou silou priameho synonymického zástupcu. Pre analýzu vhodných transformácií je nutné poznať aj všeobecné pravidlá a požiadavky na návrh transformácií a ich pravidiel, aké boli zachytené napríklad v [21] a [26].

Pri využívaní konceptuálnych modelov najmä za použitia myšlienkových máp je možné odhaľovať viacero ich výhod. Tie sa často prejavujú v moderných princípoch vývoja, akými sú najmä agilné techniky. Ich výhodami sa zaoberajú napríklad práce [18] a [15]. Všebecne sa pre zužovanie rozdielov medzi modelovacími schopnosťami odborníkov zo softvérového inžinierstva a inej domény dajú poukázať výhody využívania konceptuálnych modelov, tak ako to je spracované v práci [28] a jej nadväzujúcom výskume [7].

Pre tvorbu konceptuálnych modelov je potrebné vychádzať zo známych informácií. K nim je možné dostať sa aj iným spôsobom než len od inej osoby. Dnešné webové zdroje, akými sú mnohé encyklopédie, sa stávajú vhodným príkladom zdroja, z ktorého je možné čerpať veľa zaujímavých faktov, ktoré sú dôležité pre modelovanie previazaných faktov. Takýmto myšlienkam sa venuje napríklad práca [24]. Takéto informácie by okrem spomínaných myšlienok mohli využívať aj zdroje z vnútra inštitúcie, ktoré môžu napomôcť k tvorbe softvéru využívajúceho rôzne veľké moduly existujúcich starších projektov, podobne ako je to pri prístupe využívajúcom rad softvérových výrobkov.

Pre prvotnú evaluáciu, ktorá bola uskutočnená, je významný aj bakalársky projekt [23], ktorého cieľom bolo navrhovanie štruktúry obrazoviek používateľských rozhraní pre mobilné zariadenia rôznych platforiem. Využívaný bol pri tom modelom riadený vývoj, ktorý má aj v tejto práci možné využitie. Dôležitou časťou z pohľadu vytvorenej evaluácie je však vytváranie pôvodného platformovo nezávislého modelu štruktúry prechodov medzi obrazovkami, ktoré sa v časti 7 podarilo uskutočniť aj nad skupinou ľudí, ktorá nepatrí k expertom v softvérovom inžinierstve.

Kapitola 17

Záver a ďalšia práca

V prostredí vývoja softvéru je modelovanie mimoriadne dôležitá činnosť. Prostredníctvom nej sa už v prvotných fázach analýzy a návrhu zachytávajú potrebné požiadavky, ktoré prichádzajú zo strany zákazníka. Medzi zákazníkom a tvorcom softvéru sú však rozdiely vyplývajúce z rozdielnej doménovej špecializácie. Tie môžu signifikantne ovplyvniť prvotné fázy vývojového cyklu softvéru. Vzhľadom na to, že experti z rôznych domén, ktoré sú mimo softvérového inžinierstva, nemajú dostatočné skúsenosti s využívanými softvérovými modelovacími metódami, dochádza v tejto fáze k problémom pochopenia navrhovaných modelov. Tie sú zväčša modelované pomocou UML, prípadne iných modelovacích jazykov, ktoré sa využívajú pri návrhu softvéru. Tieto jazyky však nemusia byť dostatočne zrozumiteľné pre zákaznícku sféru. Z toho dôvodu je potrebné pokúšať sa nájsť vhodný kompromis pri zachytávaní požiadaviek.

V práci boli analyzované štandardné modelovacie techniky, ktoré sa využívajú vo veľkej miere v softvérovom inžinierstve. K nim patrí okrem spomínaného jazyka UML, napríklad jazyk BPMN. Okrem takýchto štandardizovaných notácií existuje množstvo jazykov, ktoré sú založené na princípe ontológií. Tie sú rovnako dôležité a používajú sa často vo viacerých samostatných odboroch informatiky a informačných technológií.

Dôležitou časťou analýzy je hľadanie iných možných vizualizačných jazykov, ktoré môžu nájsť využitie v doméne softvérového inžinierstva. Z veľkého množstva jazykov, ktoré sa využívajú pre konceptuálne a mentálne modelovanie boli analýzou zvýraznené ich najvýznamnejšie typy. Okrem jedných z najpoužívanejších modelov v podobe myšlienkových máp, existuje aj mnoho iných. K nim patria napríklad kognitívne mapy, konceptuálne diagramy a konceptové mapy a mnohé ďalšie. Práve tieto spomínané sa však vďaka znalostiam a skúsenostiam širokej verejnosti s nimi, dajú označiť za najvhodnejších kandidátov pre využívanie na účely zberu požiadaviek a následný počiatočný návrh softvérového produktu. Z dôvodu výberu týchto jazykov a z pohľadu potrieb modelovania rozličných modelov, boli tieto modelova-

cie techniky porovnávané voči modelovaciemu jazyku UML. Pre základné využitie boli pre porovnávanie zvolené základné diagramy – diagramy prípadov použitia a diagramy tried. Pri ich porovnávaní sa ukázali slabé a silné stránky hľadania rovnoznačného vyjadrenia jednoduchých modelov.

Vzhľadom na to, že sa takto vytvárané modely nakoniec pre použitie zachytávajú opäť, v pre softvérové inžinierstvo presnejšej forme, ktorá je práve v UML, je potrebné poukázať na možnosti transformácií týchto modelov. Snahou transformácií je prioritne ich jednoznačnosť a jednoduchá pochopiteľnosť. Pri dostatočnej definícii pravidiel platiacich pre takéto transformácie nie je následne problém vytvárať strojovo vykonateľné automatizované transformácie nad takýmito modelmi. Pritom takéto transformácie môžu mať výrazne vyššie možnosti, ako len zjednodušenie práce inžiniera pri prepise z jednej notácie do inej.

Pre poukázanie možností využívania konceptuálneho modelovania boli analyzované mnohé práce, ktorých výsledky poukazujú na vhodnosť ich využitia v zamýšľanej doméne. Okrem toho boli počas analýzy odhalené aj mnohé ďalšie možnosti využitia takto vytváraných modelov. Pre potvrdenie prvotných hypotéz výskumu boli v prvotnej fáze preukázané ich možnosti na experimente. Výsledky daného experimentu dokázali, že ľudia s rôznym vzdelaním a profesiou dokážu efektívne vytvárať zamýšľané diagramy štruktúr softvéru, resp. jeho častí.

Počas experimentu sa odhalili aj záležitosti týkajúce sa podobnosti takto vytváraných myšlienkových máp. Z tohto dôvodu boli analyzované a navrhnuté matematické modely pre ich porovnávanie, ktoré sú dôležité pre evaluáciu modelovania softvérových produktov pomocou myšlienkových máp. Okrem toho sa už v prvotnom experimente ukázali problémy týkajúce sa rôznych tvarov slov a potreba ich identifikácie. Pre niektoré takéto riešenia postačuje manuálna úprava do spoločnej synonymickej podoby, avšak v niektorých prípadoch to nie je dostačujúce. Navyše bez identifikácie niektorých kľúčových pojmov a ich triedenia do konkrétnych mapovaných UML elementov, nie je možné využívať myšlienkové mapy bez preddefinovanej štruktúry. Preto sme sa v analýze venovali aj možnostiam pre rozpoznávanie slovných druhov.

Pre ďalšiu prácu bolo kľúčovým analyzovať možnosti prvotného návrhu. Tie sa zameriavajú na využívanie týchto modelov v rôznych rovinách, pričom sa opierajú o existujúce možnosti. Nad špecifikovanými postupmi, ktoré sme navrhli na základe dlhodobých pokusov a pozorovaní, bol vytvorený prístup pre transfomácie a evaluácie podobností, ktorého výsledky sa podarilo overiť jednoduchým experimentom. Navrhnutý prístup v danej podobe, v akej je prezentovaný, je plne funkčný a využíva základné myšlienky prístupu.

V záverečných fázach výskumu sa podarilo výrazne vylepšiť prístup o nové možnosti najmä vďaka zavedeniu vrstveného modelu myšlienkovej mapy. Vďaka tomuto tvaru modelu sa prístup rozšíril o ďalšie diagramy UML, preto boli analyzované všetky. Na základe hľadania možností uplatnenia rôznych

diagramov v prístupe sa odhalilo, že myšlienkové mapy sú vhodné najmä na charakterizáciu štruktúry systému. Behaviorálnu časť modelov je vhodnejšie riešiť inými prístupmi. Prístup sa rozšíril aj využitím analýzy obsahu pojmov v myšlienkových mapách. Na základe toho je rovnako možné nájsť nové typy elementov. Evaluácia ukázala silné a slabé stránky prístupu v konfrontácii s UML. Zároveň ukazuje, že prístup je v porovnaní s podobnými existujúcimi prístupmi naozaj použiteľný a má viacero nesporných výhod a zlepšení oproti nim. V rámci oblasti myšlienkového mapovania na zachytávanie požiadaviek v softvéri sa tak dostal výskum na dostatočne zaujímavú úroveň, ktorú by bolo možné ďalej zlepšovať v detailoch.

Pre ďalšiu prácu sa ukazuje ako zaujímavé pracovať ďalej nad súvislosťami myšlienkových máp v agilnom modelovaní, ktoré sa dostáva v tejto vývojovej paradigme stále viac do popredia. Z pohľadu využívania spracovania prirodzeného jazyka by mohlo byť zaujímavé ďalej rozšírene pracovať nad požiadavkami a ich formou. Pre rozlišovanie niektorých elementov v myšlienkových mapách sa zároveň ukazuje ako potenciál uplatnenie báz znalostí, ktoré by mohli automaticky identifikovať atribúty pre nazvanú triedu a ponúkať tak odporúčania pri modelovaní. Podobne to môže byť aj pri iných prvkoch v požiadavkách.

Literatúra

[1] Stanford log-linear part-of-speech tagger. `http://nlp.stanford.edu/software/tagger.shtml`. dátum citovania: 01.05.2017.

[2] Wordnet - a lexical database for english. `https://wordnet.princeton.edu/`. dátum citovania: 01.15.2017.

[3] Zen and the art of user requirements. `http://astah.net/resources/documents/zen-and-the-art-of-user-requirements.pdf`. Available online (last accessed in May 2016).

[4] D. E. Baz, M. Hifi, L. Wu, and X. Shi. A parallel ant colony optimization for the maximum-weight clique problem. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 796–800, May 2016.

[5] J. K. Blundell, M. L. Hines, and J. Stach. The measurement of software design quality. *Annals of Software Engineering*, 4(1):235–255, 1997.

[6] T. Buzan and B. Buzan. *The Mind Map Book*. Mind set. BBC Active, 2006.

[7] A. Ceballos, F. Wanderley, E. Souza, and G. Cysneiros. *MindDomain: An Interoperability Tool to Generate Domain Models Through Mind Maps*, pages 469–479. Springer International Publishing, Cham, 2016.

[8] L. Chauvin, D. Genest, and S. Loiseau. Ontological cognitive map. In *2008 20th IEEE International Conference on Tools with Artificial Intelligence*, volume 2, pages 225–232, Nov 2008.

[9] P. P. Das and M. H. A. Khan. Solving maximum clique problem using a novel quantum-inspired evolutionary algorithm. In *2015 International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)*, pages 1–6, May 2015.

[10] M. J. Eppler. A comparison between concept maps, mind maps, conceptual diagrams, and visual metaphors as complementary tools for knowledge construction and sharing. *Information Visualization*, 5(3):202–210, 2006.

[11] H. Faste and H. Lin. The untapped promise of digital mind maps. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 1017–1026, 2012.

[12] A. Fatwanto. Natural language requirements specification analysis using part-of-speech tagging. In *Second International Conference on Future Generation Communication Technologies (FGCT 2013)*, pages 98–102, Nov 2013.

[13] A. Franklin, T. Li, P. Jamieson, J. Semlak, and W. Vanderbush. Evaluating metrics for automatic mind map assessment in various classes. In *2015 IEEE Frontiers in Education Conference (FIE)*, pages 1–8, Oct 2015.

[14] S. Gulia and T. Choudhury. An efficient automated design to generate uml diagram from natural language specifications. In *2016 6th International Conference - Cloud System and Big Data Engineering (Confluence)*, pages 641–648, Jan 2016.

[15] K. Hiranabe. Agile modeling with mind map and uml. `http://www.change-vision.com/en/UMLandMindMap.pdf`. Available online (last accessed in May 2016).

[16] P. Jamieson. Using modern graph analysis techniques on mind maps to help quantify learning. In *2012 Frontiers in Education Conference Proceedings*, pages 1–6, Oct 2012.

[17] P. Jamieson. More graph comparison techniques on mind maps to provide students with feedback. In *2013 IEEE Frontiers in Education Conference (FIE)*, pages 992–998, Oct 2013.

[18] I. Mahmud and V. Veneziano. Mind-mapping: An effective technique to facilitate requirements engineering in agile software development. In *Computer and Information Technology (ICCIT), 2011 14th International Conference on*, pages 157–162, Dec 2011.

[19] S. Nähring, C. Maus, R. Ewald, and A. M. Uhrmacher. Automated transformation between modeling languages with different expressiveness: Challenges and results from a use case with sbml and ml-rules. In *Proceedings of the Winter Simulation Conference*, WSC '12, pages 371:1–371:2. Winter Simulation Conference, 2012.

[20] R. V. Patil, N. Singhal, S. Shaikh, B. Shidore, and A. Chandgude. Integrating various views in mind mapping tool. In *Advance Computing Conference (IACC), 2014 IEEE International*, pages 90–94, Feb 2014.

[21] S. K. Rahimi. Specification of uml model transformations. In *2010 Third International Conference on Software Testing, Verification and Validation*, pages 323–326, April 2010.

[22] S. Robinson, G. Arbez, L. G. Birta, A. Tolk, and G. Wagner. Conceptual modeling: Definition, purpose and benefits. In *2015 Winter Simulation Conference (WSC)*, pages 2812–2826, Dec 2015.

[23] P. Sluka. Model-driven development of mobile applications. Bachelor thesis, STU FIIT, Bratislava, Slovensko, 2015.

[24] J. Trinkunas and O. Vasilecas. Ontology transformation: From requirements to conceptual model. *Computer Science and Information Technologies*, 751:52–64, 2009.

[25] V. O. Trinkunas J. A graph-oriented model for ontology transformation into conceptual data model. *Information Technology and Control, Kaunas, Technologija*, 36:126–131, 2007.

[26] R. Táborský and V. Vranić. Feature model driven generation of software artifacts. In *Computer Science and Information Systems (FedCSIS), 2015 Federated Conference on*, pages 1007–1018, Sept 2015.

[27] F. Wanderley, D. S. da Silveira, J. Araújo, and A. Moreira. Transforming creative requirements into conceptual models. In *IEEE 7th International Conference on Research Challenges in Information Science (RCIS)*, pages 1–10, May 2013.

[28] F. Wanderley and D. S. da Silveria. A framework to diminish the gap between the business specialist and the software designer. In *Quality of Information and Communications Technology (QUATIC), 2012 Eighth International Conference on the*, pages 199–204, Sept 2012.

[29] F. Wanderley, D. Silveira, J. Araujo, A. Moreira, and E. Guerra. *Computational Science and Its Applications – ICCSA 2014: 14th International Conference, Guimarães, Portugal, June 30 – July 3, 2014, Proceedings, Part V*, chapter Experimental Evaluation of Conceptual Modelling through Mind Maps and Model Driven Engineering, pages 200–214. Springer International Publishing, Cham, 2014.

[30] Z. Ye, Z. Jia, J. Huang, and H. Yin. Part-of-speech tagging based on dictionary and statistical machine learning. In *2016 35th Chinese Control Conference (CCC)*, pages 6993–6998, July 2016.

[31] C. Yi. An english pos tagging approach based on maximum entropy. In *2015 International Conference on Intelligent Transportation, Big Data and Smart City*, pages 81–84, Dec 2015.

[32] J. Zedlitz, J. Jörke, and N. Luttenberger. *Knowledge Technology: Third Knowledge Technology Week, KTW 2011, Kajang, Malaysia, July 18-22, 2011. Revised Selected Papers*, chapter From UML to OWL 2, pages 154–163. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[33] J. Zedlitz and N. Luttenberger. Transforming between uml conceptual models and owl 2 ontologies. *Terra Cognita 2012 Workshop*, 6:15, 2012.

Dodatok A

Zadanie pre experiment

Turistický sprievodca

Mobilná aplikácia turistický sprievodca je určená pre pomoc všetkým turistom, ktorí prídu na Slovensko. Aplikácia sa zameriava na ponuku rôznych aktivít, ktoré sú k dispozícii v letnej alebo zimnej turistickej sezóne, alebo sú k dispozícii počas celého roka. Používateľovi sa po spustení aplikácie zobrazuje na pár sekúnd niekoľko informačných obrazoviek (2-3), ktoré obsahujú: prvá je obrazovka loga aplikácie a ďalšie 1-2 sú obrazovky, ktoré obsahujú reklamu na nejakú akciu, ktorá sa koná v blízkom období. Po prejdení cez tieto obrazovky sa používateľovi zobrazuje hlavné menu aplikácie. Vzhľadom na to, že aplikácia umožňuje ukladať obľúbené miesta/akcie do zoznamu obľúbených, v aplikácii existuje možnosť registrovať sa na externý portál, s ktorým aplikácia spolupracuje. Aplikácia je použitá na doplnenie služieb klasickej webovej stránky, na ktorej sú možné takmer všetky funkcie, ktoré sú aj v aplikácií. Používateľ sa v prípade existujúceho konta, alebo po registrácii, môže priamo v menu prihlásiť. Po prihlásení sa mu menu mierne upraví tak, že sa v ňom objaví položka „obľúbené“. Okrem tlačidiel súvisiacich s registráciou a prihlásením a obľúbenými miestami, sú v menu prítomné tlačidlá pre zmenu nastavení aplikácie ako takej a pre zmenu nastavení používateľského konta prihláseného používateľa.

Pre používateľa je dôležité, aby sa k hľadaným údajom dostal rýchlo a jednoducho. Na hlavnej stránke je preto položka slúžiaca na vyhľadávanie. Je možné tam zadať napríklad obec, typ atrakcie, a iné, prípadne to kombinovať. Podľa zadaných parametrov sa aplikácia pokúsi používateľovi zobraziť výsledky, ktoré by ho mohli zaujímať, pričom ich zoraďuje podľa relevantnosti, zhody vo vyhľadávaní a hodnotení iných používateľov.

Okrem možnosti využitia priameho vyhľadávania sa v menu nachádzajú tlačidlá na konkrétne typy atrakcií (to sú napr. kúpaliská, hory, lyžiarske strediská, historické pamiatky, múzeá, ...). Po prejdení na stránku jedného typu atrakcie sa zobrazí stránka zoznamu atrakcií. Tie je možné filtrovať podľa niekoľkých parametrov (napr. miesta). Po kliknutí na konkrétnu at-

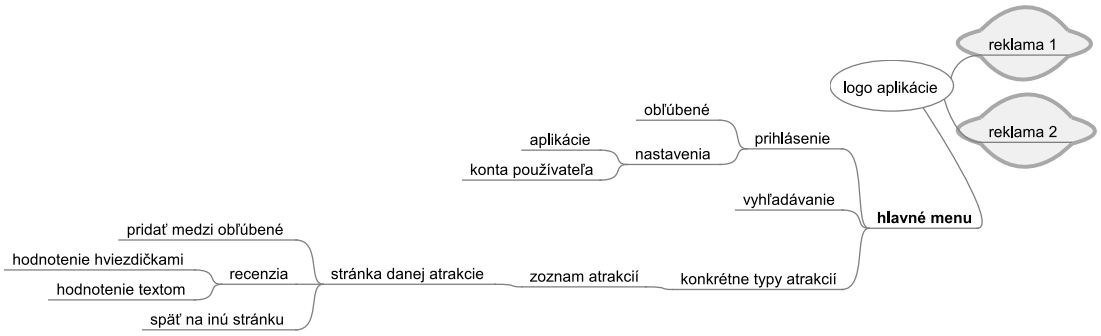
rakciu sa zobrazí stránka atrakcie. Tá sa líši v závislosti na type atrakcie. Všeobecne sa na stránke atrakcie nachádza názov, obrázok, popis, mapa, prípadne galéria. Konkrétne typy sa líšia. Turistické ciele v horách majú k dispozícii opis navigácie k hľadanému napr. vrchu a výsek z turistickej mapy. Historické ciele majú na stránke zobrazenú stručnú históriu. Múzeá zobrazujú informácie o zameraní múzea. Podobne by sme mohli opísať aj iné typy atrakcií. Pre tie, ktoré sú platené je na stránke aj základný cenník. Niektoré môžu mať informácie o ďalších údajoch, kontakty, ...

Konkrétnu atrakciu môže používateľ označiť ako obľúbenú a pridať si ju tak do svojho listu. Zároveň môže na jednotlivé atrakcie vytvárať recenzie – ohodnotením hviezdičkami a krátkym komentárom. Zo stránky konkrétnej atrakcie sa môže vrátiť na stránku kategórie typu atrakcií, atrakcií v konkrétnom meste, alebo na zoznam svojich obľúbených, či zoznam tých, na ktoré vytváral recenzie. V nastaveniach aplikácie a používateľského profilu sa nachádzajú bežné nastavenia, aké poznáte aj z iných aplikácií, pre zadanie nie sú až také dôležité, môžete sa však nad nimi zamyslieť.

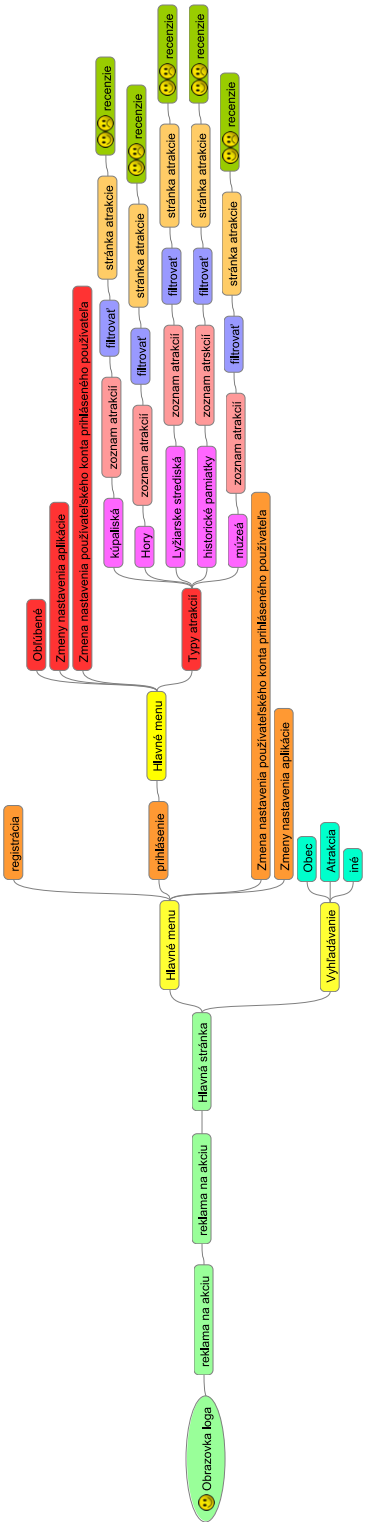
Pokúste sa namodelovať hierarchiu aplikácie a jej obrazoviek pomocou myšlienkovej mapy. Zamýšľajte sa pri tom najmä nad logickou hierarchiou, rozdielmi a vhodnými asociáciami medzi stránkami. Prípadné nejednoznačnosti v správnom hierarchickom usporiadaní, môžete zakresliť na viackrát. Mapu môžete zakresliť ručne, v grafických nástrojoch, alebo v niektorom nástroji na myšlienkové mapy (najlepšie pre jednoduchosť FreeMind alebo Xmind alebo niečo online).

Dodatok B

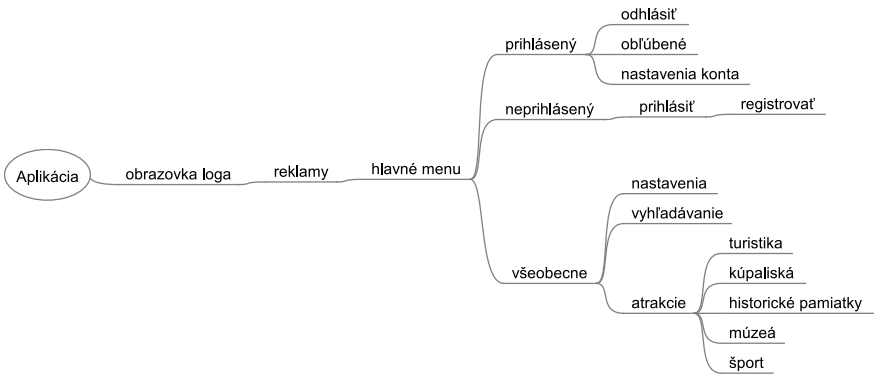
Navrhnuté riešenia



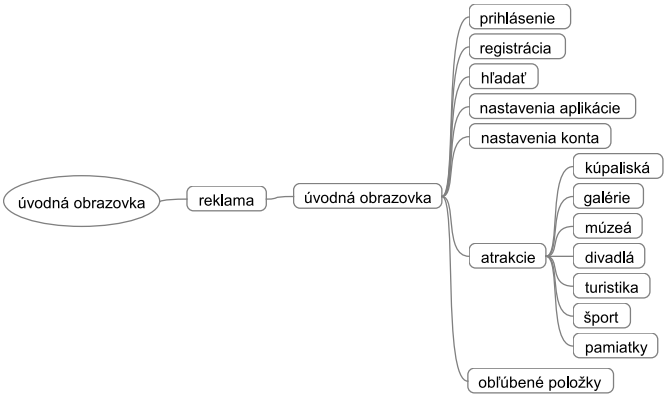
Obr. B.1: Riešenie 1



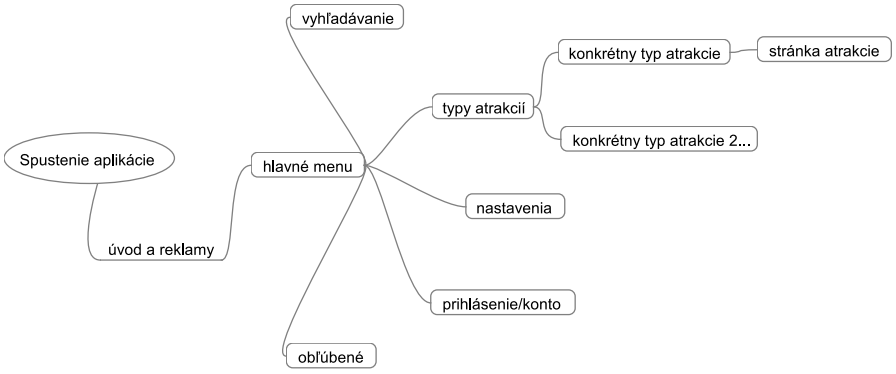
Obr. B.2: Riešenie 2



Obr. B.3: Riešenie 3



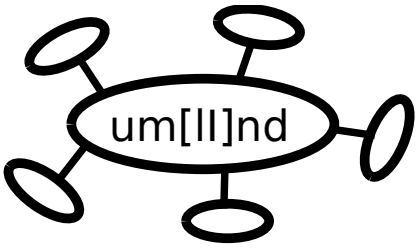
Obr. B.4: Riešenie 4



Obr. B.5: Riešenie 5

Dodatok C

Popis nástroja umInd



Nástroj umInd je nástroj na transformácie a porovnávanie myšlienkových máp. V aktuálnej verzii podporuje myšlienkové mapy vytvorené v nástroji Xmind a UML modely vytvorené v nástroji Papyrus. Umožňuje vykonávať transformácie medzi týmito modelmi na základe definovaných pravidiel špecifikovaného prístupu 12. Modely, pre správne fungovanie, môžu byť vytvorené vo formátoch, ktoré opisuje vytvorený prístup. Nástroj je implementovaný v jazyku Java.

Nástroj pracuje so vstupnými modelmi vo formátoch XML (XMI), ktoré načítava do vnútornej reprezentácie vlastných tried. Pre prácu so vstupnými modelmi a výstupnými modelmi XML pracuje nad možnosťami, ktoré ponúka DOM parser, ktorý je štandardnou súčasťou knižníc v Jave. Tento parser bol pre nástroj zvolený ako jednoduchá alternatíva na dosahovanie vytýčených cieľov pre jednoduché a všeobecne použiteľné použitie.

Pre transformácie definuje viaceré možnosti, ako pracovať s jednotlivými prvkami, a ako ich transformovať. V závislosti na pravidlách vychádzajúcich zo stanoveného prístupu dokáže odhaľovať konkrétne prvky pre stanovenie konkrétnych UML elementov. Modely dokáže vytvárať na základe týchto pravidiel.

Nástroj obsahuje aj systém porovnávania myšlienkových máp, ktorý využíva viaceré navrhnuté metriky. Systém v takom prípade akceptuje dva modely (porovnávaný a porovnávajúci), nad ktorými prebieha porovnanie. V tomto prípade sa pri porovnávaní modely rovnako prevádzajú na tvar

reprezentovaný vnútornými triedami nástroja a nad týmito triedami sa aplikujú porovnávacie algoritmy pre myšlienkové mapy.

Nástroj je rozdelený do balíčkov podľa špecifického použitia. Obsahuje balíčky týkajúce sa importov a exportov modelov pre jednotlivé nástroje, balíček definujúci vnútorný model tried, balíček pre výpočty porovnávania. Implementácia je navhnrutá na rýchle prispôsobenie a možné rozšírenia, ktoré sa očakávajú v súvislosti s pokračujúcim výskumom.

Implementácia je pripravená na ďalšie rozširovanie. V budúcnosti sa uvažuje nad rozšírením podpory modelovacích nástrojov o Enterprise Architect a FreeMind.

Pri práci so súbormi XML využíva štandardné xml elementy a atribúty jednotlivých nástrojov. Tie vkladá ako automatickú súčasť exportovaných súborov. V štruktúre sú dôležité napríklad generátory univerzálnych identifikátorov jednotlivých prvkov a mapovanie ich vzťahov. V prípade modelov pre Xmind sa využíva štruktúra súborov definovaná týmto nástrojom v súbore content.xml, kde sú jednotlivé elementy mapované v rekurzívnej štruktúre myšlienkovej mapy. Pre Papyrus sa využíva štruktúra UML elementov v rôznej forme. Elementy akými sú vzťahy, alebo atribúty tried sú mapované ako child elementy im prislúchajúceho elementu. V prípade samotných elementov ako napríklad triedy, prípady použitia a hráči systému, sa využíva samostatná definícia, pričom prípadný vzájomný vzťah je uvedený pomocou previazania cez univerzálny identifikátor prislúchajúceho elementu (napríklad v generalizácii je uvedené id triedy, od ktorej je dedené).

Väčšina funkcií je z dôvodu rekurzívnych definícií implementovaná ako rekurzívne spracovávanie jednotlivých elementov. Podobne je to aj pri výpočtoch podobnosti myšlienkových máp, pričom v tejto fáze sa rekurzia využíva na preprocesing a samotný výpočet sa aplikuje nad získanými množinami porovnávaných pojmov.

Pre použitie nástroja je potrebné vytvárať modely v nástroji Papyrus poskytovanom ako rozšírenie pre modelovaciu verziu Eclipse. Z týchto modelov je potrebné vybrať súbor uml, ktorý obsahuje definovanú štruktúru modelu v xml. V prípade Xmind je potrebné vytvárať štandardnú myšlienkovú mapu (základnú alebo professional) a zo súboru .xmind vybrať len časť content.xml, ktorá obsahuje základnú štruktúru modelu. Tú získame extrahovaním súboru použitím bežného komprimačného nástroja (.xmind je interná špecifikácia .zip súboru spoločnosti Xmind).

Dodatok D

Návod na použitie

Pre použitie a prácu s navrhnutým prístupom je potrebné mať k dispozícii tieto nástroje: - Eclipse (na preklad zdrojového kódu nástroja) - Eclipse modeling tools (na zobrazovanie transformovaných modelov) - Papyrus extension (na zobrazovanie transformovaných modelov) - Xmind (na vytváranie myšlienkových máp)

D.1 Základné použitie na transformácie

Popis použitia:

1. V nástroji Xmind vytvoríme myšlienkovú mapu štandardne pomocou vytvoriť nový model.

2. Vytvoríme jednotlivé pojmy reprezentujúce zamýšľaný softvérový produkt.

3. Uložíme vytvorenú myšlienkovú mapu vo formáte .xmind

4. Prejdeme do Eclipse a v hlavnej triede nástroja Maintransformation nastavíme jednu z predpripravených možností.

5. V tomto prípade je vhodné zvoliť transformáciu z myšlienkovej mapy do UML, nastavíme pri tom cestu pre vstupný a výstupný súbor XML (z .xmind a z .uml).

6. Pre model z Xmind nástroja je potrebné extrahovať súbor content.xml z archívu .xmind - je možné použiť bežný nástroj (napr. 7zip).

7. Vykonáme transformáciu spustením programu.

8. Získaný model vložíme do nástroja Papyrus presunutím vygenerovaného súboru do model explorera.

9. V modeli potiahneme všetky existujúce časti modelu do zobrazenia na paneli modelu.

10. Upravíme rozloženie jednotlivých komponentov uml podľa zváženia.

Podobne môžeme v nástroji zvoliť opačný postup tak, že vytvoríme model v nástroji Papyrus a jeho model následne využijeme ako vstupný do nástroja. Zvolíme pri tom ponúkanú možnosť, ktorá sa týka transformácie vstupného modelu z Papyrusu. Pri transformáciách môžeme využiť možnosti transformovania aj z konkrétneho nástroja späť do toho istého nástroja. Výsledný model ostáva nezmenený. Takáto transformácia však nie je výhodná, keďže sa pri nej nijako nevyužíva navrhnutý prístup.

Nástroj obsahuje niekoľko možností pre transformácie, ktoré sú vypísané v MainTransformation. Ide o transformácie využívajúce rôznu mieru podrobností - základná je určená na transformáciu do tvaru dedičnosti UML diagramu tried. Pokročilá exportuje do UML elementov diagramu tried a diagramu prípadov použitia.

D.2 Použitie na porovnávanie myšlienkových máp

Pre počítanie podobnosti myšlienkových máp vo formáte Xmind postupujeme rovnako ako v predchádzajúcom prípade.

1. Najskôr vytvoríme nami uvažované modely pre porovnanie v nástroji Xmind.

2. Následne v hlavnom programe MainMatcher.java vyberieme jednu z ponúkaných možností podľa zváženia, akú metriku na porovnávanie použijeme.

3. Spustíme vykonávanie programu.

4. Program v konzole vypíše výsledok evaluácie podobnosti, pričom vypíše aj počty jednotlivých rozdielov a názvy rozdielnych pojmov v mapách.

V hlavnej triede MainMatcher sa nachádza niekoľko možných metrík, ktoré je možné zamieňať odkomentovaním ich častí a zakomentovaním inej metriky.

Dodatok E

Popis nástroja umInd3

Nástroj umind3 je odvodený od pôvodného nástroja umind. Dopĺňa ho však o nové súčasti poskytované v rozšírenom prístupe na modelovanie a transformácie myšlienkových máp. Nástroj podporuje priame používanie xmind súborov modelovacej aplikácie Xmind. Tie vďaka svojej zipovej forme dokáže extrahovať a využívať ich súčasti z xml súborov. Nástroj v tejto verzii je realizovaný ako prototyp implementačnej stránky navrhnutého prístupu. Obsahuje aj používateľské rozhranie, testovanie transformácií je však možné robiť aj pomocou priložených testovacích tried.

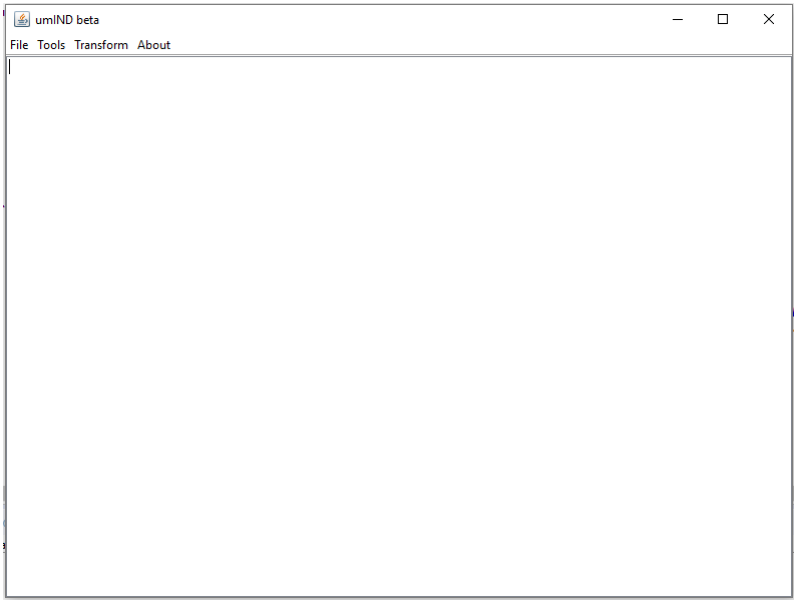
Používateľské rozhranie E.1 je riešené minimalisticky. Rozdeľuje sa na menu a telo aplikácie. V tele sa nachádza textové pole, do ktorého sú logované výstupy načítavania súborov a transformácií. V menu sa nachádza viacero položiek prvej úrovne. Patria medzi ne časti File, Tools, Transform a About. About časť obsahuje len krátky popis verzie programu.

V časti menu File sa nachádzajú možnosti na otvorenie (Open) a uloženie (Save) súborov a ukončenie programu (Close). Možnosť uložiť súbor je k dispozícií iba v prípade, že pracujeme už s nejakou formou súboru. V prípade potreby načítania súboru do programu je potrebné vybrať položku otvoriť (Open) a vybrať vhodný súbor. V prípade načítavania Papyrus modelu je potrebné vybrať jeho .uml časť, v prípade myšlienkových máp v Xminde priamo .xmind súbor. Po otvorení súboru sa model načíta do korešpondujúcej vnútornej formy myšlienkovej mapy, resp. UML.

V časti Tools sa nachádzajú možnosti, ktoré môžeme využiť pri transformácii myšlienkovej mapy do UML. Týkajú sa využívania špecifikovaných pravidiel pre kapitalizáciu tried, neprázdnosť koncových tried a využívanie nástrojov spracovania prirodzeného jazyka pre identifikáciu potenciálnych stavov. Podľa potreby môžeme pred transformovaním myšlienkovej mapy tieto možnosti vybrať.

V časti Transform sa nachádzajú samotné možnosti transformovania. Tie sú dve – transformácia myšlienkovej mapy do UML a transformácia UML do myšlienkovej mapy. Funkcie sú k dispozícii striedavo podľa toho, s akou

formou modelu sme naposledy pracovali.



Obr. E.1: Používateľské rozhranie nástroja umind3

Dodatok F

Návod na použitie nástroja umind3

Pre úspešnú prácu s nástrojom umind3 sa predpokladá inštalácia niekoľkých nástrojov a vývojových prostredí. V tomto návode sa venujeme konkrétnym častiam týchto potrebných nástrojov a ďalším krokom potrebným k úspešnej práci s implementovaným prototypom postaveným nad špecifikovaným prístupom.

Základ postupu sa zhoduje s návodom v prílohe D.

F.1 Potrebné nástroje pre modelovanie

V prípade potreby modelovania myšlienkových máp je nutné mať nainštalovaný nástoj na ich modelovanie Xmind. Ten je možné stiahnuť z oficiálnych stránok tohto najpoužívanejšieho nástroja na modelovanie myšlienkových máp. Stránky Xmind nájdeme na www.xmind.net. V prípade potreby modelovania UML modelov v nástroji Papyrus je prvým krokom inštalácia vývojového prostredia Eclipse, konkrétne vo verzii pre modelovanie. Túto verziu môžeme stiahnuť na stránkach www.eclipse.org v časti download. Po spustení stiahnutého inštalátora vyberieme položku eclipse modeling. Po úspešnej inštalácii Eclipse Modeling ho spustíme a prejdeme do časti Help ->Install Modeling Components a vyberieme nástroj Papyrus. Po jeho inštalácii môžeme s modelovacím nástrojom pracovať vytvorením nového projektu pre Papyrus a následne jeho modelov.

F.2 Potrebné nástroje pre prácu s umind3

Pre používanie nástroja umind3 je potrebné mať nainštalovanú javu (to isté aj pre Papyrus) a vhodnú verziu vývojového prostredia Eclipse. Následne môžeme importovať existujúci projekt do workspacu výberom File ->Import a vybratím možnosti Existing Project to Workspace. Tu zvolíme

zložku s projektom a počkáme na dokončenie importu. Následne môžeme spúšťať súčasti nástroja – testovacie triedy alebo časť obsahujúcu používateľské rozhranie. Okrem využitia samostatného spustenia nástroja môžeme využiť aj priložený jar súbor nástroja, ktorý spustíme štandardne.

F.3 Formy modelov

Pre vytvorenie modelov v nástrojoch je potrebné držať sa požiadaviek pre model, ktoré sú zahrnuté v časti špecifikácie navrhnutého prístupu. Pre testovanie alebo inšpiráciu modelov je možné využiť aj priložené modely v Xmind a Papyrus formátoch, ktoré sú súčasťou dodávaných súborov na disku. Pre nástroj Papyrus je potrebné dodržovať základné konvencie jazyka UML a využívať prevažne zasiahnuté typy prvkov a vzťahov jazyka UML z prístupu. Pri vytváraní myšlienkovej mapy vytvárame formu podľa špecifikácie – základ je daný na prvej vrstve obsahujúcej kľúčové otázky a následne existujúce prvky môžeme ďalej prepájať navzájom v ďalších vrstvách. Pre určenie toho, že ide o už existujúci element využívame možnosť vložiť hypertext linku na prvok zo základnej vrstvy. Pri pomenovávaní sa rovnako držíme pôvodného názvu.

F.4 Použitie nástroja

Pri práci s nástrojom vychádzame z jeho popisu, podľa jednotlivých častí používateľského rozhrania a testovacích tried. Pre otvorenie súboru s modelom (Xmind/Papyrus) zvolíme otvoriť. Následne môžeme meniť nastavenie transformácie v časti Tools a používať transformácie v časti Transformations. Model po určitom transformačnom cykle môžeme uložiť do súboru, ktorý si zvolíme a pomenujeme podľa vlastných predstáv. Pre opätovné použitie vytvoreného súboru UML modelu papyrus je však nutné pomenovať súbor s koncovkou .uml. V prípade myšlienkovej mapy môže byť užitočné súbor pomenovať content.xml. Tak sa volajú súbory s obsahovou stránkou myšlienkovej mapy, ktoré sú vo vnútri .xmind archívu. Ak chceme využiť vytvorený xml súbor v xmind je potrebné využiť niektorý z nástrojov na extrahovanie a tvorbu archívov, napríklad voľne dostupný 7zip. Následne v ňom otvoriť nejaký existujúci xmind archív, vymazať z neho všetky časti a vložiť do neho súbor content.xml. Následne po uložení by mal byť model k dispozícii po otvorení v nástroji Xmind.

Dodatok G

Obsah priloženého CD

Priložené elektronické médium obsahuje všetky súvisiace súbory. Ide najmä o navrhnutý nástroj využívajúci navrhnutý prístup, ďalej dokumentáciu práce a výsledky evaluácie. Okrem toho obsahuje aj priložené súbory k prvotnej evaluácii zámeru. Všetky tieto časti sú rozdelené do priečinkov.

V priečinku /doc sa nachádza dokumentácia samotnej práce. V priečinku /tool sa nachádza transformačný nástroj umInd postavený nad navrhnutým prístupom. Obsahuje nástroj na transformácie medzi myšlienkovými mapami a UML a prostriedky výpočtu podobnosti myšlienkových máp. Implementácia je v jazyku Java. Popri základnej verzii nástroja použitej v prvotnej evaluácii obsahuje v priečinku /umInd3 aj hotový prototyp pracujúci nad rozšíreným prístupom. V priečinku /eval sa nachádzajú súbory súvisiace s evaluáciou podobnosti myšlienkových máp a výsledky transformácií týchto máp do UML. Zároveň obsahuje získané grafy, zobrazujúce výsledky porovnávania s očakávaným výsledkom myšlienkovej mapy. Okrem toho obsahuje aj samotné zadanie pre evaluáciu. V priečinku /testeval sa nachádzajú súbory prvotnej evaluácie projektu, ktorá bola vykonaná s cieľom presvedčenia sa o správnom smerovaní projektu. Pre testovanie nástroja umind3 je vhodné použiť ukážkové modely myšlienkových máp a UML, ktoré sa nachádzajú v časti /umInd3models.

Štruktúra elektronického média je teda nasledovná:

```
/doc
/tool
   /umInd
   /umInd3
/eval
   /assignment
   /graphs
   /models
   /transformatedmodels
/testeval
```

```
   /assignment
   /models
/umInd3models
```

Dodatok H

Dokumentácia evaluácie

V tejto časti sa nachádza zhrnutie evaluácie z kapitoly 13, konkrétne zadanie samotného experimentu a výsledky podľa respondnetov.

H.1 Zadanie experimentu

Uvažujte systém pre nasledovne špecifikované rekreačné stredisko:

Rekreačné stredisko (predstavte si aquapark) ponúka zákazníkom rôzne atrakcie a služby. Pre ich zabezpečenie je potrebné na príslušné časy naplánovať prítomnosť zamestnancov obsluhy s potrebnou kvalifikáciou. Zároveň je potrebné umožniť zákazníkom aby si mohli zakúpiť (online alebo osobne) vstupenky, či už na jednotlivé atrakcie, prípadne konkrétne termíny alebo vstupenky, ktoré umožnia vstup na viaceré atrakcie, prípadne aj opakovane. Platnosť vstupenky môže byť obmedzená počtom vstupov, časovo alebo súčtom ceny využitých atrakcií a služieb. Možné je aj prečerpanie a následné doplatenie ceny pri pokladni. Atrakcie sú kapacitne aj časovo obmedzené. Je tiež potrebné riešiť obmedzenie a sledovanie fyzického prístupu zamestnancov a zákazníkov do jednotlivých častí strediska. Systém by mal slúžiť aj na sledovanie dochádzky zamestnancov.

Modelujte daný systém použitím myšlienkových máp v troch verziách. Využite pritom základnú štruktúru mapy s centrálnym pojmom Rekreačné stredisko a troma vetvami Kto?, Čo? a Kedy?.

Vo vetve Kto? vypíšte, aké typy osôb budú interagovať so systémom v nejakej podobe.

Vo vetve Čo? namodelujte jednotlivé elementy, ktoré budú musieť byť v systéme uchovávané v nejakej forme databázy. Pre jednotlivé pojmy v tejto oblasti následne vypíšte Vami uvažované konkrétne údaje potrebné pre uchovanie.

Vo vetve Kedy? vypíšte prípady, kedy bude systém v rekreačnom stredisku naozaj využívaný. Pre špecifikáciu jednotlivých pojmov v mape využite ako inšpiráciu základnú uvádzanú definíciu.

Modelujte 3 verzie mapy:

1. V prvej verzii uvažujte nad možnými modelovanými prvkami samostatne, pokúste sa predstaviť si vlastné stredisko a jeho jednotlivé súčasti.

2. V druhej verzii sa zamerajte na dôležité predostreté pojmy: zamestnanec, zákazník, vstup do strediska, vstup na atrakciu, zakúpenie vstupenky, evidencia zamestnanca, vstupenky, atrakcie, ich typ a pod.

3. V tretej verzii sa zamerajte aj na konkrétne prvky jednotlivých elementov tak ako sú vypísané tu:

   - Vstupenka (typ, cena, kategória a platnosť)
   - Atrakcia (názov, typ, obmedzenie, otváracie hodiny, cena)
   - Zamestnanec (meno, zaradenie, dochádzka)
   - Zákazník
   - Zamestnanec a jeho typy (Plavčík, Dozorca, Čašník, Kuchár, Správca, Pokladník, Masér, Animátor)
   - Možné použitia na: nákup lístka (aj zľaveného a skupinového), nákup atrakcie, dobitie lístka, vstup na atrakciu, príchod do práce, odchod z práce, nákup občerstvenia

V tretej verzii vynechajte všetky svoje predstavy o stredisku, modelujte len predpísané pojmy.

Vo všetkých verziách pracujte nad stanovenou základnou štruktúrou mapy. Následne v jednotlivých verziách môžete pridávať a odstraňovať vami uvažované definované pojmy. Pojmy špecifikujte čo najjednoduchšie tak, aby bolo zrozumiteľné, čo predstavujú.

Príklad jednoduchej štruktúry pre internetovú žiacku knižku:
Internetová žiacka knižka:

- Kto?
  - Žiak
  - Učiteľ
- Čo?
  - Známka
    - percentil
    - váha
    - hodnotenie
  - Predmet

Tvar zadania	Základná metrika	Granulárna metrika	Metrika 2-4 línií
Základný	0,3673	0,3191	0,1974
Rozšírený	0,4565	0,4444	0,3424
Úplný	1,000	0,9474	0,9029

Tabuľka H.1: Respondent 1

        * názov
    - Používateľ
        * typ
        * práva
        * meno
        * heslo

- Kedy?
    - zápis známok
    - prezretie známok

H.2 Výsledky po respondentoch

V tejto časti sa nachádzajú výsledky jednotlivých ľudí, ktorí sa zúčastnili experimentu. V jednotlivých tabuľkách sú uvedené podobnosti voči základnej vzorovej myšlienkovej mape. Z výsledkov je možné vidieť, že so zvyšovaním presnosti definovaných požiadaviek dochádza k približovaniu sa k očakávanej myšlienkovej mape. Vhodnosť použitých metrík sa v jednotlivých prípadoch môže líšiť vzhľadom na jednoduchosť definovaného experimentu. Z pozorovaní jednotlivých vytvorených myšlienkových máp totiž je možné usúdiť, že v niektorých prípadoch môže byť skutočná podobnosť určená až manuálne a výsledok môže byť vyšší. Jednotlivé vytvorené mapy vstupujúce do experimentu boli normalizované na rozdiely v nejednoznačnosti (najmä synonymickej) pojmov.

Tvar zadania	Základná metrika	Granulárna metrika	Metrika 2-4 línií
Základný	0,3864	0,3902	0,2651
Rozšírený	0,4545	0,4419	0,3440
Úplný	1,000	0,9474	0,9029

Tabuľka H.2: Respondent 2

Tvar zadania	Základná metrika	Granulárna metrika	Metrika 2-4 línií
Základný	0,2927	0,2750	0,1012
Rozšírený	0,4600	0,3922	0,2957
Úplný	1,0000	1,0000	1,0000

Tabuľka H.3: Respondent 3

Tvar zadania	Základná metrika	Granulárna metrika	Metrika 2-4 línií
Základný	0,5581	0,4048	0,2995
Rozšírený	0,6279	0,5227	0,4419
Úplný	0,9744	0,7568	0,5851

Tabuľka H.4: Respondent 4

Dodatok I

Vyhodnotenie plánu prác

V tejto prílohe sa nachádza zhrnutie vyhodnotenia plánu prác, ktoré boli definované v predchádzajúcom období práce na diplomovom projekte. Obsahuje zoznam cieľov a ich splnenie, rovnako ako aj súpis nových odhalených možností, ktoré boli spracované nad rámec pôvodného plánu.

Harmonogram práce sa zameriaval na riešenie evaluácie a experimentovania nad navrhnutým prístupom. V rámci toho sa podarilo doplniť do prístupu zapracovanie vrstvenia myšlienkových máp, vďaka čomu sa nám otvorili viaceré nové možnosti, ktoré sme pôvodne v prístupe nemali zahrnuté. Vďaka vrstveniu sa podarilo špecifikovať transformácie aj do iných typov UML diagramov a ich lepšie preskúmanie.

V prípade experimentov nad odhalovaním kľúčových slov prirodzeného jazyka boli vykonané experimenty nad obsahom požiadaviek z pohľadu slovných druhov a väzieb medzi slovami. Na základe toho boli odhalené viaceré vlastnosti jednotlivých elementov v požiadavkách a UML. Na základe experimentov sme identifikovali ako najvhodnejšie pracovať s kapitalizáciou (v podobe vlastných názvov) a s identifikáciou prídavných mien ako základom pre stavy entít v stavovom diagrame.

Exerimentami nad možnosťami UML sme preskúmali všetky predtým nepreskúmané diagramy a ich možnosť zahrnutia do prístupu. Odhalili sme tak vlastnosti, ktoré majú jednotlivé elementy diagramov a koncepty myšlienkovej mapy, ktoré sa do týchto elementov transformujú.

Na základe experimentovania sme doplnili v prístupe nové pravidlá a transformácie, ktoré umožňujú využitie skúmaných prvkov a diagramov. Evaluovali sme nad softvérovým projektom a jeho modelom v podobe UML a hľadali možnosti, ako ich využiť v myšlienkových mapách. Na základe experimentov sa ukázalo, že je vhodnejšie pracovať s požiadavkami v menšom rozsahu. V prípade väčších projektov skôr rozdeliť kompletné požiadavky na viacero podčastí. Tieto časti sa pritom môžu charakterizovať ako časti spracované pri jednotlivých stretnutiach s doménovým expertom alebo ako individuálne komponenty systému. V ďalšej fáze sme odhalili ešte ďalšie po-

tenciálne časti pre ďalší výskum a prebiehala finalizácia práce a jej častí.

Tu je možné vidieť zhrnutie plánovaného harmonogramu a splnenie jeho častí:

Týždeň	Popis	Splnenie
1. - 2.	Doplnenie vrstvenia do transformačných pravidiel a experimentovanie nad nimi	áno
3. - 4.	Experimentovanie nad identifikáciou kľúčových slov prirodzeného jazyka	áno
5. - 6.	Porovnávanie UML a pokročilejšie experimenty	áno
7. - 9.	Doplnenie transformačných pravidiel o novo zistené pravidlá, podrobnejšia evaluácia nad náročnejšími softvérovými projektmi	áno
10. - 12.	Zhrnutie výsledkov, dokončenie chýbajúcich častí práce a prípadné nutné úpravy nástroja	áno

Z hľadiska predpripraveného plánu sa podarilo realizovať všetky úlohy. Nie všetky však boli spracované v rovnakej miere. Plán bol navrhnutý tak, aby sa bolo možné ďalej venovať aj predtým neuvažovaným častiam. Práve vďaka voľnejšej forme harmonogramu sa podarilo odhaliť aj ďalšie predtým neuvažované zálažitosti. Niektoré z nich boli realizované, iné sú predpokladom pre vhodnosť ďalšieho výskumu v budúcnosti. Celkovo sa plán dá charakterizovať ako úspešne splnený, a to aj napriek tomu, že jeho priebeh plnenia bol voľnejší než bol pôvodný plán.

Dodatok J

Článok z konferencie IIT.SRC

Requirements modeling and transformations using mind mapping

Pavel SLUKA*

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova 2, 842 16 Bratislava, Slovakia
pasluka@gmail.com

Abstract. In any software development process, it is important to get customers' requirements for the product. These requirements have to be marked down in suitable form, so they can be passed to another development phase without any problems. Requirements have to be modeled in such a way, that both sides of requirements gathering recognize these models. In this case, they have to be represented in a known notation for both participated sides. This article shows that mind maps can be potential solution. This approach is based on using them for requirements modeling and shows, how to transform these models to UML.

1 Introduction

When developing any kind of software product, it is important to understand problem domain. Software developers are experts in domain of software development, but they may not understand everything in professional domain of other fields. These fields have their own experts and becoming an expert can be long-term business, often demanding years of professional experience. Because of these facts, domain experts have considerable effect for development cycle and have to participate in it. Main responsibility of domain expert is formulation of requirements in a way that software expert understands them. They have to be marked down in suitable form. Suitability of modeling technique can be viewed in terms of accuracy, integrity, clarity and faultlessness. Violation of these terms can have strongly negative impact on software development and its phases.

In software development, there are used software modeling techniques, mostly UML. However, these modeling notations are often hard to understand and can result in bad understandability by domain expert. This problem can be eliminated using such modeling techniques, that both software engineers and domain experts are familiar with. Such techniques consist of lot of different conceptual modeling notations. Some of the most used examples are conceptual diagrams, cognitive maps and mind maps. After analysis of these modeling techniques, mind maps look like the best choice. It comes directly from definition created by mind mapping author Tony Buzan [2]. From this definition, mind maps are characterized as powerful technique, in spite of its simplicity. It is thanks to its mapping is similar to human brain functionality when dealing with real world facts. Other advantages over other techniques are its broad usage, ease of use, learnability and probability of experience with this technique. Also some other related researches shows its potential in software engineering.

Requirements modeled using mind maps have advantage especially from customer's view. On the other hand, software experts usually use standard models (such as UML models) and their use can be more suitable. When requirements are modeled using mind maps, they have to be transformed to UML. Transitions between these modeling techniques have to have clear transformation rules. When there are defined clear transformation rules, models can be transformed automatically. Definition of such rules can be difficult, because of different expressive power of mind maps and UML. Because of that, rules have to define expected model forms and use only relevant subset of more expressive notation's set.

2 Related work

Working with mind maps in software development can have significant effect to improve its process. Nowadays, mind maps can be found mostly in agile environment. They can be used not only for brainstorming,

* Master study programme in field: Software Engineering
Supervisor: Dr. Ján Lang, Institute of Informatics, Information Systems and Software Engineering, Faculty of Informatics and Information Technologies STU in Bratislava

IIT.SRC 2017, Bratislava, April 27, 2017, pp. 1–8.

but research [4] shows, that their use can improve quality of product backlog. As we have mentioned, requirements modeling using mind maps is suitable alternative mainly because of business specialists – domain experts [8]. Modeling in this way have potential and these models can be in some way transformed to UML. Modeling requirements using mind maps can also significantly improve time spend in requirements gathering [9].

Since we are studying modeling techniques with different expressiveness, we have to look for good solution when dealing with their differences. Transformations between models with different expressive power can be found in different approaches [5, 6]. Regardless of used model techniques, basic properties of these transformations are almost always the same. Models have to have clearly defined transformation rules and we have to deal with absence of some corresponding transitions.

In this paper, we propose multiple different model forms. Very useful form for requirements gathering was proposed also in [1, 3], where authors find key questions. Answers on these questions forms basics of requirements information.

3 Mind maps solution

Mind map solution for requirements modeling is based on usage of mind maps in one or more specified model forms and their follow-up transformation to UML. Because of transformation between modeling techniques with different expressiveness, there are used only some of the selected items of UML, while focusing on their most important parts. Method deals with their transformations and defines clear transformation rules, capable to transform from mind map to UML and also in reverse.

3.1 Proposed model forms

This approach deals with multiple different model forms. Each form is formally defined by its characterization and specific modeling usage. Some models are used to capture only potential UML classes, and the other also model potential UML use cases. These models are specified by their mind map representation and have corresponding transformation rules.

The most basic model works on fundamental conceptual processes of abstraction and instantiation. Central topic consists of most abstract concept, while its subtopics are instantiated subsets. Next level of instantiated topics has also their parent abstract concept and so on. In software engineering, we can compare it to inheritance (for example class inheritance). Example of such model form is in figure 1.

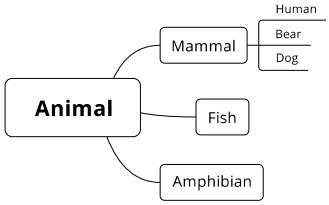


Figure 1. Basic mind map

When dealing with topics in this way, we need to create some opportunity to describe some attributes specific for each topic. In this case, we can adopt some best practices from software engineering – distinction of lower case and upper case. Similarly, like classes attributes begin with lowercase letter, we can also model these attributes of topics and then names of topics itself will begin with upper case as shown in figure 2.

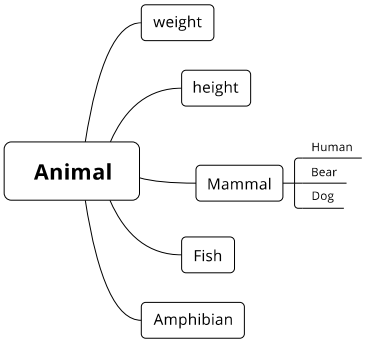


Figure 2. Basic mind map with lower case attributes

These basic model forms are not dealing with other requirements, like what will developing software be used for or who will use it. There comes definition of fundamental questions, which can be used in requirements gathering [1]. These questions are Who? (who will use it), What? (what will be stored, what kind of information will it provide) and When? (when will the software be used, in what use cases). These three questions clearly links to corresponding UML diagrams (class and use case diagrams) and their items. Central topic in this case is software product itself and consists of three branches with questions as in figure 3.

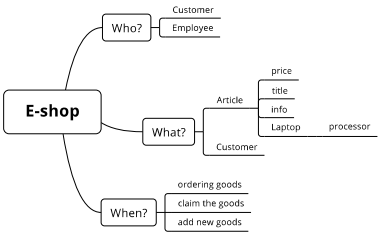


Figure 3. Who? When? What? map

3.2 Transformation rules

Formalized model forms have to have clear transformation rules, thanks to which, we will be able to transform them to UML or transform UML models to mind maps.

Main rules are based on mentioned formalization of abstraction and instantiation and its related topic of inheritance. All subtopics are transformed as subelements of parent topic. In class diagram formalization subtopic is subclass of parent topic class. Example of such transformation of mind map from figure 1 can be found in figure 4.

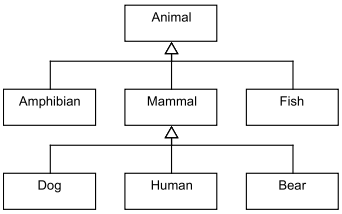


Figure 4. Transformed class diagram of basic model

When we extend this with letter case rules, subtopics starting with upper case are subelements (subclasses), while subtopics starting with lower case are attributes of parent topic (class). Transformed class diagram with attributes is shown in figure 5.

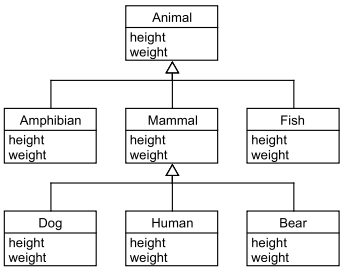


Figure 5. Transformed class diagram with attributes

In three questions model form, we transform each part of mind map individually 6. Who? and When? parts are transformed to UML use case diagram components. Topics in who? part are actors of system and their instantiated subtypes, while topics in when? part are use cases and their instantiated subtypes (extended use cases). Topics in What? are classes and their attributes.

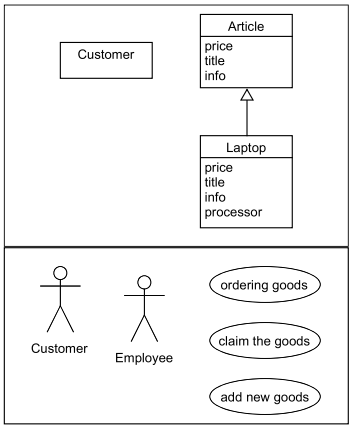


Figure 6. Transformation result of who? when? what? map

3.3 Mind map viewpoints

Model forms in approach can be too simple to model all relations between each components of mind map.

Mind map itself however does not support multiple links, which represent relation between different branches. Mind maps are generally acyclic graphs (trees). However, in some situations it can be appropriate to create such links. Because of that, we propose mind map viewpoints, as a solution for this issue.

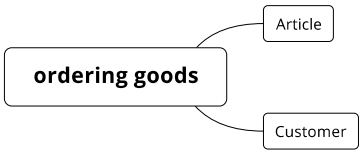


Figure 7. New layer for one use case

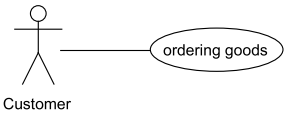


Figure 8. Result of transformation

Viewpoints in mind maps can be perceived similar to software architecture views and viewpoints defined by Rozanski and Woods [7]. Similarly, as their viewpoints represent software product from specific point of view, we can create more mind maps, representing same requirements from different point of view. Exemplarily, we will create basic mind maps as tree structures, but we will extend them by using multiple layers of independent topics, representing same topic as in other layer. With this solution, we are able for example join actors with their corresponding use cases by creating new layer describing this relationship as shown in figures 7 and 8.

4 Conclusion

In this paper, there was proposed new approach for dealing with requirements and their gathering using mind maps. This modeling techniques can be really useful, thanks to its simplicity and can be more appropriate because of customer's lack of experience with advanced modeling techniques such as UML. Mind maps in this approach are not used to replace UML, but to support initial phases of software development cycle. Because of that, it is important to formalize transformation between these modeling techniques. Simplicity of mind maps can cause problems when trying to model all relations between components. To eliminate this issue, we proposed viewpoints for mind maps. Thanks to defined models and transformation rules created for them, approach successfully shows its potential and ability.

Further work will focus on dealing with advanced UML class diagram and UML use case diagram components, which are not so important in phase of requirements gathering, but can bring new possibilities when transforming UML models to mind maps.

References

[1] Zen and the Art of User Requirements. `http://astah.net/resources/documents/zen-and-the-art-of-user-requirements.pdf`, Available online (last accessed in 2017).

[2] Buzan, T., Buzan, B.: *The Mind Map Book*. Mind set. BBC Active, 2006.

[3] Hiranabe, K.: Agile Modeling with Mind Map and UML. `http://www.change-vision.com/en/UMLandMindMap.pdf`, Available online (last accessed in 2017).

[4] Mahmud, I., Veneziano, V.: Mind-mapping: An effective technique to facilitate requirements engineering in agile software development. In: *Computer and Information Technology (ICCIT), 2011 14th International Conference on*, 2011, pp. 157–162.

[5] Nähring, S., Maus, C., Ewald, R., Uhrmacher, A.M.: Automated Transformation Between Modeling Languages with Different Expressiveness: Challenges and Results from a Use Case with SBML and ML-Rules. In: *Proceedings of the Winter Simulation Conference*. WSC '12, Winter Simulation Conference, 2012, pp. 371:1–371:2.

[6] Rahimi, S.K.: Specification of UML Model Transformations. In: *2010 Third International Conference on Software Testing, Verification and Validation*, 2010, pp. 323–326.

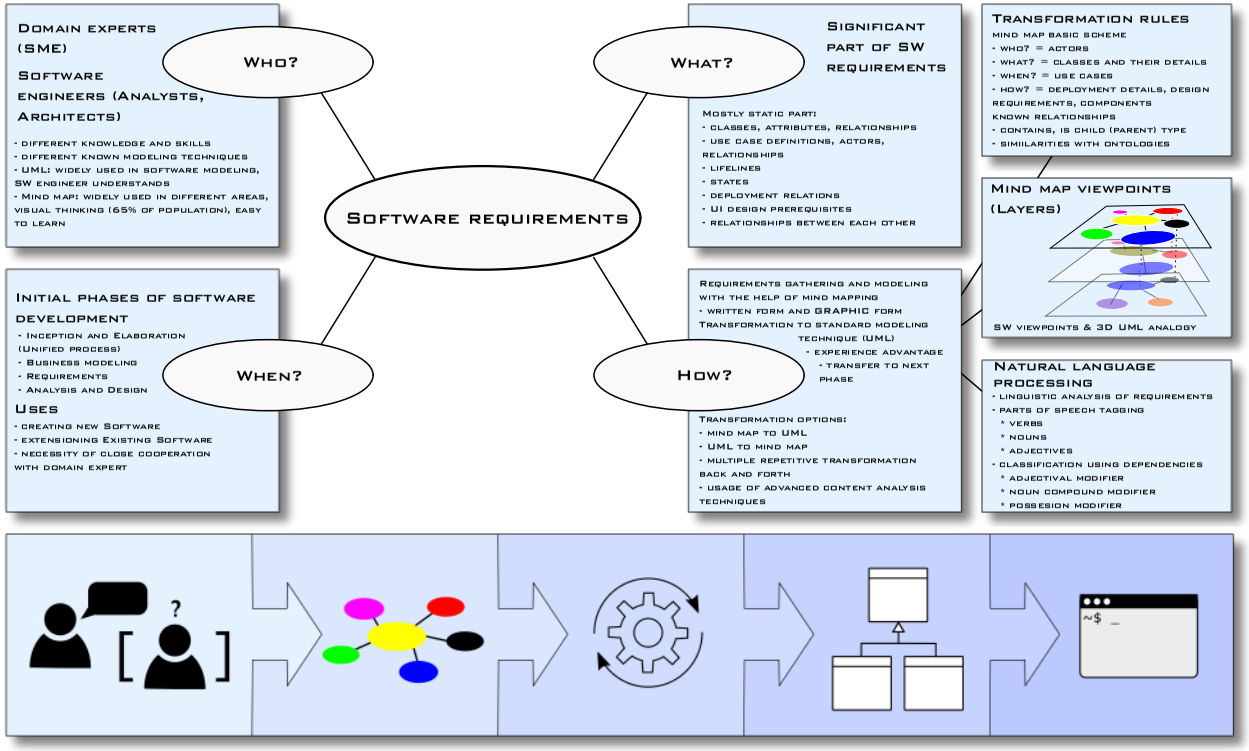
[7] Rozanski, N., Woods, E.: *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley Professional, 2005.

[8] Wanderley, F., da Silveria, D.S.: A Framework to Diminish the Gap between the Business Specialist and the Software Designer. In: *Quality of Information and Communications Technology (QUATIC), 2012 Eighth International Conference on the*, 2012, pp. 199–204.

[9] Wanderley, F., Silveira, D., Araujo, J., Moreira, A., Guerra, E.: Chap. Experimental Evaluation of Conceptual Modelling through Mind Maps and Model Driven Engineering. In: *Computational Science and Its Applications – ICCSA 2014: 14th International Conference, Guimarães, Portugal, June 30 – July 3, 2014, Proceedings, Part V*. Springer International Publishing, Cham, 2014, pp. 200–214.

Dodatok K

Poster z konferencie IIT.SRC

Requirements modeling and transformations using mind mapping

Pavel Sluka, Supervisor: Ján Lang