Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A IFORMAČNÝCH TECHNOLÓGIÍ
Študijný program: Softvérové inžinierstvo

Bc. Pavol Michalco

PRÍPADY POUŽITIA A TÉMY V PRÍSTUPE THEME/DOC

Diplomová práca

Vedúci diplomového projektu: Ing. Valentino Vranić PhD.

máj,2009

Anotácia

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program: SOFTVÉROVÉ INŽINIERSTVO

Autor: Bc. Pavol Michalco
Diplomová práca: Prípady použitia a témy v prístupe Theme/Doc
Vedúci diplomovej práce: Ing. Valentino Vranić PhD.
máj, 2009

Táto diplomová práca sa zaoberá aspektovo-orientovaným vývojom softvéru vo fáze analýzy. Sú v nej predstavené dva prístupy aspektovo-orientovanej analýzy – aspektovo-orientovaný vývoj pomocou prípadov použitia a Theme/Doc prístup. Jadrom práce je návrh transformácie Theme/Doc modelu do modelu prípadov použitia a naopak. Na základe týchto transformácií sú potom tieto prístupy bližšie analyzované, porovnávajú sa ich výhody a nevýhody a hľadajú sa analógie s cieľom ukázať, že základné a aspektové témy v Theme/Doc prístupe zodpovedajú základným a rozširujúcim prípadom použitia a naopak.

Annotation

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES
Degree Course: SOFTWARE ENGINEERING

Author: Bc. Pavol Michalco
Thesis: Use cases and themes in Theme/Doc approach
Supervisor: Ing. Valentino Vranić PhD.
2009, May

This diploma project is concerned with the aspect-oriented software development in the analysis phase. Two approaches to aspect-oriented analysis are briefly introduced – aspect-oriented software development with use cases and Theme/Doc approach. The core of diploma project is devoted to propose a process of transformation from Theme/Doc model into use case model and vice versa. On the basis of these transformations, these approaches are analyzed, their advantages and disadvantages are compared and analogies are finding to demonstrate that base and aspect themes in the Theme/Doc correspond to base and extension use cases and vice versa.

Zadanie projektu

Kópia licenčnej zmluvy

Čestné prehlásenie:
Čestne prehlasuje, že som diplomovú prácu na tému Prípady použitia a témy v prístupe Theme/Doc vypracoval samostatne a všetku použitú literatúru uvádzam v zozname.

V Bratislave dňa 13. mája 2009

.....

Poďakovanie
Ďakujem vedúcemu mojej diplomovej práce, Ing. Valentinovi Vranićovi PhD., za cenné rady a podnety, ktoré mi poskytol počas tvorby tejto práce.

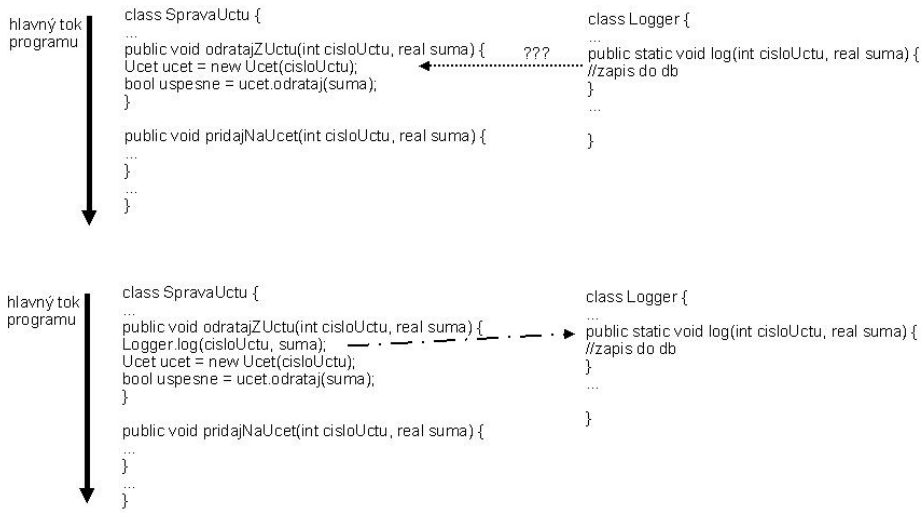
Obsah

1 Úvod

1.1 Pretínajúce záležitosti

Prax vo vývoji softvéru ukázala, že dôležité je nielen vytvoriť softvér, ktorý je funkčný, ale musí byť tiež ľahko udržiavateľný. Základným východiskom pre ľahšiu údržbu je modularizácia, t.j. rozdelenie programu na moduly, ktoré vnútorne súvisia. Rozdelenie programu na moduly pomáha aj pri riešení zložitého problému, keď problém rozložíme na jednotlivé záležitosti, ktorými sa môžeme zaoberať samostatne. Charakteristické pre modul je, že rieši jednu záležitosť, je textovo lokálny (t.j. text programu patriaci do jedného modulu sa nachádza na jednom mieste), má definované rozhranie, ktoré opisuje ako interaguje so zvyškom systému, pričom je možné robiť zmeny v kóde modulu bez zmien rozhrania. Dnes najznámejším príkladom takého modulu je trieda v objektovo orientovanom programovaní. Rozdelenie programu na takéto moduly je pomerne náročné. Dôvodom sú tzv. pretínajúce záležitosti, kedy nie je možné jednotlivé záležitosti oddeliť do samostatných modulov. Vtedy dochádza k javom známym ako zapletenie (tangled) kódu a roztrúsenie (scattered) kódu. Zapletený kód je taký, keď v rámci jedného modulu sa nachádza kód rôznych záležitosti. Roztrúsený kód je zase taký, keď kód jednej záležitosti alebo jej časti sa nachádza v rôznych moduloch. Sú to dva pohľady na ten istý problém.

Typickým príkladom pretínajúcich záležitosti je logovanie alebo autorizácia. Ak chceme, aby boli logované vstupy všetkých metód vo všetkých triedach (t.j. moduloch), tak musíme ťažkopádne do každej metódy na jej začiatok vložiť kód, ktorý bude obsluhovať logovanie, aj keď to nie je primárna záležitosť daného modulu. V prípade, že by sme sa rozhodli, že zrušíme logovanie, tak by sme museli naopak tieto kódy ručne odstraňovať. Problémom je, že hoci by bol vytvorený modul na logovanie, aj tak nie je možné z tohto modulu presmerovať tok programu, ktorý prebieha v inom module (obr. 1.1). Preto presmerovanie toku do modulu logovania je potrebné zabezpečiť v samotnom module, v ktorom tok prebieha, volaním metódy modulu logovania, čo v konečnom dôsledku vedie k spomínanému prepleteniu a roztrúseniu kódu.



Obr. 1.1: Tok vykonávania programu a nemožnosť jeho presmerovania do iného modulu bez vzniku prepleteného, resp. roztrúseného kódu.

1.2 Aspektovo-orientované programovanie

Riešením na implementačnej úrovni bol vznik nového prístupu k programovaniu, tzv. aspektovo orientované programovanie (Aspect-Oriented Programming, ďalej len AOP). Do programovacích jazykov sa zaviedli nové konštrukcie, ktoré umožňovali presmerovať tok z iného modulu do svojho modulu a vykonať potrebný kód. Vznikli konštrukcie ako aspekt (aspect), videnie (advice), bodový prierez (pointcut), bod spájania (join point) a iné, čo umožnilo špecifikovať pretínanie na vyššej úrovni abstrakcie. Pomocou týchto konštrukcií je možné v každom module definovať, na ktorých miestach sa má prerušiť tok v programe (bodové prierezy), aby sa mohol vykonať iný požadovaný kód (videnie) a rozhodnúť, či po vykonaní tohto kódu pokračovať v mieste prerušenie toku programu (t.j. modifikovať základný tok programu). Samotná modifikácia môže zahŕňať aj pridanie nových prvkov a závislostí do iných modulov (tzv. medzitypová deklarácia – inter-type declaration). Tieto nové konštrukcie tak umožnili mudularizáciu aj pretínajúcich záležitostí. Základná konštrukcia, ktorá tvorí akýsi modul pretínajúcej záležitosti sa nazýva aspekt (podľa pretínajúca záležitosť = aspekt). V ňom je možné vytvárať spomínané konštrukcie videnia, bodových prierezov, atď.

Ak máme byť presnejší, tak aj záležitosti, ktoré nie sú pretínajúce môžu byť označené za aspekty. Samotné slovo aspekt znamená pohľad na vec alebo stránka veci. Keďže každá záležitosť (hoci nie je pretínajúca) predstavuje nejakú stránku nášho problému, tak každú záležitosť je možné označiť za aspekt nášho problému. Tento prístup, keď program vzniká spájaním rozličných aspektov (moduly sú aspekty) sa nazýva symetricky aspektovo-orientovaný prístup. Príkladom je programovací jazyk Hyper/J. Oveľa častejšie sa môžeme stretnúť s označením aspekt len pre pretínajúce záležitosti. Nepretínajúcich záležitosti sú oddelené základnou dekompozíciou

a pretínajúce záležitosti sú oddelené pomocou konštrukcií aspektov. Tento prístup sa nazýva asymetricky aspektovo-orientovaný prístup. Príkladom je programovací jazyk Aspect/J, ktorý je aspektovým rozšírením objektovo-orientovaného jazyka Java, kde nepretínajúce záležistosti sú zachytené triedami a pretínajúce inými konštrukciami, ktoré sa nazývajú aspekty (konštrukcia class vs. konštrukcia aspect) a ktoré môžu obsahovať spomínané konštrukcie potrebné na modifikáciu základného toku programu a medzitypovú deklaráciu. Najznámejší aspektovo-orientovaný jazyk je práve Aspect/J, preto aj v tejto práci sa budeme odvolávať práve naňho.

AOP nie je možné považovať za nový revolučný prístup. Je to vlastne len prirodzený vývoj v dôsledku toho, že programátorom sa napríklad nechcelo vkladať kód do každého miesta, ktoré chceli logovať. Vznikali rôzne nástroje, ktoré im to uľahčovali. Takýto jednoúčelový nástroj by sme si mohli aj sami naprogramovať. Stačilo vytvoriť program, ktorý by na vstupe očakával regulárny výraz identifikujúci metódu, ktorej výstup chceme logovať a cestu k zdrojovému kódu programu. Po spustení tohto programu by sa do zdrojového kódu na definované miesta zapísali potrebné inštrukcie tak, aby bolo zabezpečené logovanie výstupov požadovaných metód. Po tejto činnosti by sme potom len program skompilovali a spustili. Typickým príkladom je debugger, ktorý sa používa vo vývojových prostrediach. Spôsob práce aspektovo-orientovaných jazykov je podobný (tzv. vtkanie - weaving), no hlavnou myšlienkou AOP je zovšeobecniť tieto a podobné príklady do úrovne zmien zákl. toku programu a medzitypovej deklarácie prostredníctvom bežných konštrukcií jazyka. Takto je možné všetky tieto problémy abstrahovať na problém pretínajúcich záležitostí.

1.3 Apektovo-orientovaný vývoj softvéru

Pretínajúce záležitosti sú jasne viditeľné až vo fáze implementácie. No bolo by vhodné identifikovať tieto záležitosti skôr než sa začne programovať, aby sa s nimi rátalo už pri návrhu vhodnej štruktúry programu. Preto je dôležité zaoberať sa odhaľovaním pretínajúcich záležitosti už v začiatočných fázach vývoja softvéru (tzv. early aspects – skoré alebo včasné aspekty). Odhalenie pretínajúcich a nepretínajúcich záležitosti vo fáze analýzy znamená, že v prípade implementácie aspektovo-orientovaným jazykom je možné vytvoriť štruktúru programu zodpovedajúcu odhaleným záležitostiam, samozrejme po ich spresnení a určení vzájomných vzťahov vo fáze návrhu. Získame tým softvér, ktorého štruktúra bude nepriamo zodpovedať kritériám, ktoré na neho kladú všetci zainteresovaní účastníci (tzv. stakeholders). Takáto štruktúra je výhodná najmä vo fáze údržby softvéru, kedy môže účastník definovať novú alebo zmeniť nejakú pôvodnú požiadavku. Identifikuje sa záležitosť, ktorej sa požiadavka týka a následne sa v programe upraví daná štruktúra (aspekt alebo trieda v prípade asym. AO jazyku), prípadne sa doplní nová záležitosť. Program, štruktúrovaný podľa záležitostí a vlastnosti AOP, zmenší námahu pri implementovaní takýchto zmien. Tento spôsob vývoja softvéru sa nazýva aspektovo-orientovaný vývoj softvéru (Aspect-Oriented Software Development, ďalej len AOSD).

1.4 Prehľad

Nasledujúce časti tejto práce sa budú zaoberať opisom dvoch najznámejších prístupov AOSD a ich vzájomným porovnávaním vo fáze analýzy. Najprv to budú prípady použitia a ich použitie v aspektovo-orientovaným vývoji. Potom to bude prístup Theme/Doc. Po ich bližšom predstavení sa práca bude venovať analýze vzťahu medzi prípadmi použitia a témami v Theme/Doc. S cieľom odhaliť ďalšie bližšie podobnosti a rozdiely medzi týmito prístupmi, bude navrhnutá transformácia modelu prípadov použitia na model v notácii Theme/Doc a naopak.

2 AOSD pomocou prípadov použitia (AOSD/UC)

S myšlienkou AOSD pomocou prípadov použitia ako prvý prišiel Ivar Jacobson (Jacobson, 2003). Počiatočným impulzom bolo zistenie, že väzba rozšírenia medzi jednotlivými prípadmi použitia, ktorá doteraz nebola podporovaná v žiadnom z analytických a návrhových modeloch a ani žiadnym implementačným prostredím (označované ako hlavný problém modularity prípadov použitia), mohla byť pomocou AOP ľahko definovaná programovacím jazykom. AOP označuje ako *„the missing link“* (prekl.: chýbajúce spojenie), ktoré doteraz chýbalo, aby bolo možné oddelené prípady použitia preniesť cez všetky modely až k implementácii a tam ich aj oddelene implementovať (do samostatných modulov). Dokonca zaviedol ekvivalenciu medzi rozšíreniami v UML a aspektami v AOP (*„Extensions in UML ≈ Aspects in AOP“*). Reakciou bol článok (Pawlak, et al., 2004), kde autori polemizujú nad touto ekvivalenciou a prezentujú vlastné riešenie. Argumentujú to tým, že prípady použitia a ich prenos až na implementačnú úroveň, tak ako to popísal Ivar Jacobson, nedokážu pokryť všetky črty, ktoré AOP momentálne ponúka. Kritizujú, že Jacobson popísal iba prípady before a after napojenia na body spájania (vykonanie videnia pred a po bode spájania, chýba napr. around) a neboli spomenuté ani možnosti spájania viacerých aspektov nad jedným bodom spájania, tak ako to je v AOP (v AOP riešené pomocou určovania poradia vykonávania týchto aspektov). V roku 2004 Ivar Jacobson vydáva knihu, v ktorej detailne opisuje AOSD pomocou prípadov použitia a obhajuje svoj prístup (Jacobson, et al., 2004).

2.1 Prípady použitia a aspekty

Diagram prípadov použitia je často používaný spôsob špecifikácie funkcionálnych požiadaviek. Zachytáva v požiadavkách identifikované záležitosti všetkých zainteresovaných účastníkov a vzťahy medzi nimi, čím modeluje správanie systému. Diagram prípadov použitia má slúžiť na komunikáciu medzi všetkými účastníkmi, preto sa na ich opis používa prirodzený jazyk. Samotný prípad použitia obsahuje scenár, ktorý hovorí, čo musí systém robiť, aby pokryl danú záležitosť. Väzby v diagrame prípadov použitia znázorňujú závislosti medzi jednotlivými prípadmi použitia a účastníkmi. Tieto väzby môžu byť bližšie konkretizované ich typovaním. Existujú väzby závislosti typu include (zahŕňa) a extend (rozširuje). Okrem nich existuje aj väzba generalization (generalizácia) na znázornenie abstrakcie. Vzťah medzi účastníkom a prípadom použitia sa znázorňuje obyčajnou asociáciou. Je zvykom tieto väzby zovšeobecňovať len na väzbu typu závislosť, pretože nie všetci účastníci dokážu správne pochopiť význam týchto typov väzieb a ich správne použitie si vyžaduje skúsenosti. Nesprávnym použitím týchto väzieb sa môžeme dostať do situácie, ktorá sa nazýva funkcionálna dekompozícia, kedy pomocou väzieb include a extend v diagrame prípadov použitia rozdeľujeme jednu záležitosť na funkcie a tie

zase na podfunkcie atď., ktoré nemajú pre žiadneho účastníka (rôzneho od vývojára) žiadnu jasne viditeľnú hodnotu. Takto vzniká rozsiahli diagram, ktorý obsahuje veľké množstvo pre účastníkov neinteresantných prípadov použitia (Ővergaard, et al., 2004). Objavujú sa aj prípady, kedy sú použité vlastné typy väzieb (pomocou tzv. stereotypov – konvencií definovaných medzi znakmi << a >>), ktoré lepšie konkretizujú danú závislosť, no nie sú tak náročné na použitie ako spomínané typy, napr. invokes alebo precedes (Rosenberg, et al., 2007). No v prípade prístupu AOSD pomocou prípadov použitia je hlavná idea založená práve na základných väzbách include, extend a generalization, preto je dôležité správne porozumieť ich významu a vedieť ich správne použiť.
Na margo prepojenia väzieb include a extend s funkcionálnou dekompozíciou Ivar Jacobson tvrdí, že štruktúrovanie prípadov použitia pomocou týchto väzieb nie je príčinou funkcionálnej dekompozície, ale príčinou je hlavne neskúsenosť ľudí, ktorí ich používajú a preto nesúhlasí s ich zatracovaním. V knihe (Jacobson, et al., 2004) to analogicky prirovnáva ku skúsenostiam s riadením automobilu. Ak osoba nemá žiadne skúsenosti s riadením automobilu, tak pravdepodobne do niečoho nabúra nezávisle od toho, či má malé alebo veľké auto, ani či má manuálnu alebo automatickú prevodovku.

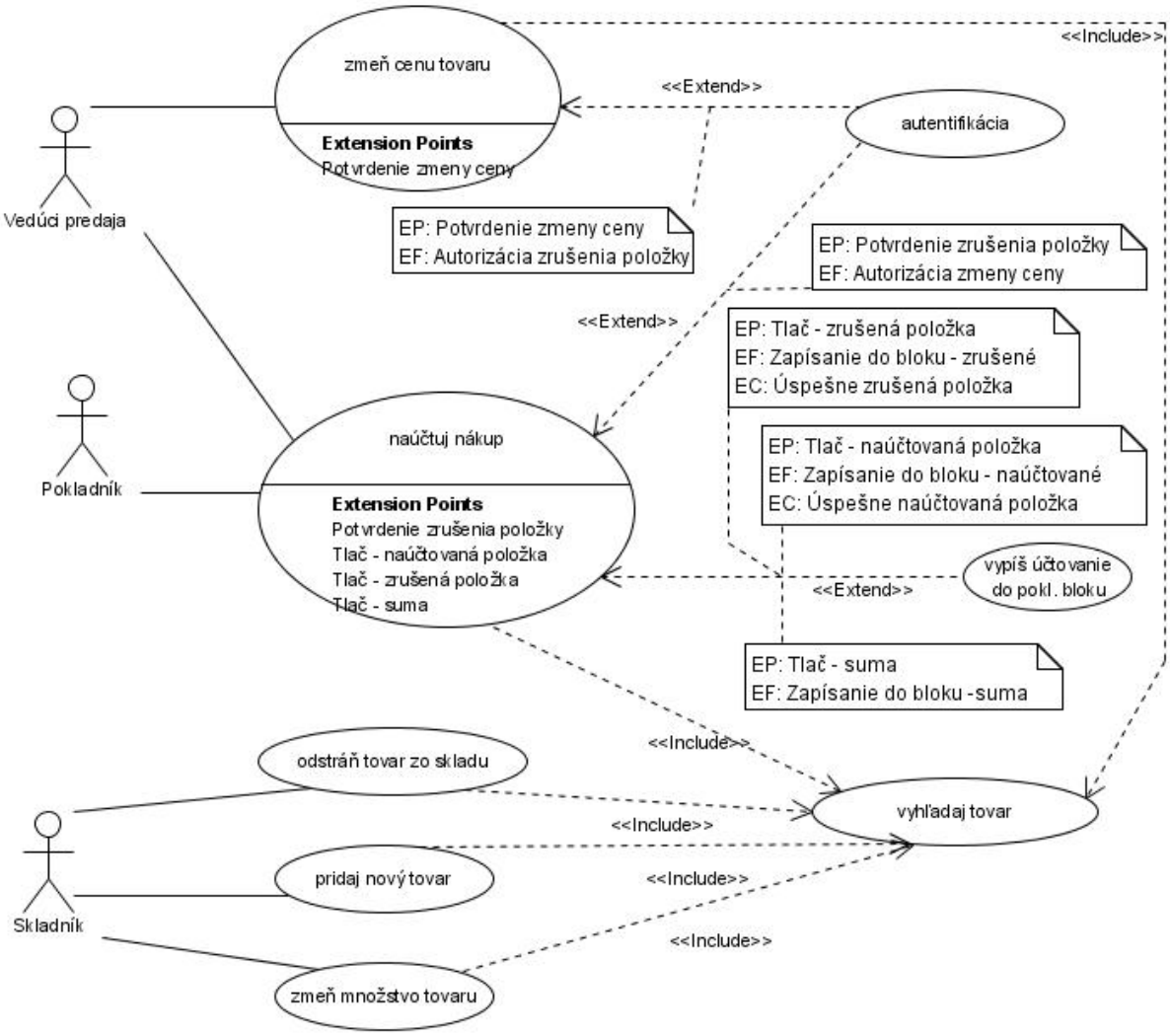
Pre lepšie pochopenie prípadov použitia, väzieb medzi nimi a súvislosti s AOP si uvedieme jeden príklad, ktorý si rozoberieme.

Zákazník, chce systém na správu tovaru a jeho predaja v obchode. Chce, aby účtovanie nákupu pri pokladniach priamo odrážalo stav tovaru, ktorý bude evidovaný týmto systémom a naopak. Pokladne sú počítače typu PC a tvoria počítačovú sieť s ostanými počítačmi v obchode. Databázu tovaru v systéme bude spravovať skladník, ale ocenenie tohto tovaru bude výlučne záležitosť vedúceho predaja. Boli spísané tieto požiadavky zákazníka (pre zjednodušenie nebude uvedený celý zoznam požiadaviek, len tie najzákladnejšie):

1. Systém bude evidovať stav tovaru v centrálnej databáze dynamicky podľa práve predaných položiek.
2. Skladník bude môcť zo systému odstraňovať tovar.
3. Skladník bude môcť do systému pridávať nový tovar.
4. Skladník bude môcť meniť v systéme množstvo položiek tovaru.
5. Pokladník bude môcť položku nákupu naúčtovať ručným zadaním čiarového kódu alebo pomocou čítačky čiarových kódov.
6. Naúčtovať sa môžu iba položky, ktoré sú evidované v databáze.
7. Počas účtovania nákupu bude možné odstrániť položku z naúčtovaných položiek.
8. Odstránenie položky z naúčtovanej položky musí potvrdiť vedúci predaja svojou autentifikáciou.
9. Naúčtované položky sa budú priebežne zaznamenávať na pokladničný blok. Na pokladničnom bloku sa budú nachádzať naúčtované položky, zrušené položky a celková suma nákupu.

10. Upraviť cenu tovaru v centrálnej databáze môže iba vedúci predaja, ktorý svoju identitu potvrdí autentifikáciou.
11. Čas od zadania čiarového kódu po zobrazenie ceny identifikovanej položky bude trvať maximálne pol sekundy.

Tieto požiadavky sú modelované prostredníctvom diagramu prípadov použitia[1] (obr. 2.1).



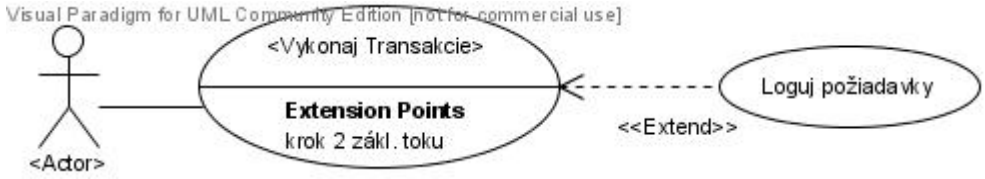
Obr. 2.1: Diagram prípadov použitia pre náš systém.

[1] Napriek tomu, že v slovenskej literatúre je zaužívané písať názvy prípadov použitia ako slovesné podstatné mená, tak v tejto práci budú názvy prípadov použitia v aktívnom slovesnom tvare, tak ako sa používajú v anglickej literatúre.

Je dôležité si všimnúť, že diagram neobsahuje požiadavky, tak ako sú zapísané vyššie. Tieto požiadavky sú zachytené niekde v scenároch jednotlivých prípadov použitia, ktoré určujú správanie systému, keď interaguje s používateľom. Tieto scenáre zahrňujú požiadavky z pohľadu používateľa a cieľom akcií v scenároch je priniesť konkrétnemu používateľovi nejakú užitočnú hodnotu. Čiže prípad použitia tieto požiadavky modeluje a modularizuje ich do záležitostí používateľov a účastníkov, ktorých sa týkajú a diagram prípadov použitia všetky tieto záležitosti organizuje. Väčšina ľudí si myslí, že prípady použitia zachytávajú iba funkcionálne požiadavky. Ani na obrázku (obr. 2.1) nie je nikde zaznačená požiadavka č. 11. Ale Ivar Jacobson predstavil spôsob, ako niektoré takéto požiadavky zachytiť v prípadoch použitia. Nefunkcionálne požiadavky, ktorých úlohou je zvyšovať kvalitu softvéru, sa môžu zakresliť ako rozšírenie k prípadu použitia vykonania transakcie (obr. 2.2). Transakcia znamená pár požiadavka účastníka – odpoveď systému. Základný scenár sa skladá zo štyroch krokov (Jacobson, et al., 2004):

1. Systém požiada inštanciu účastníka o identifikovanie požadovanej inštancie entity.
2. Inštancia účastníka vloží hodnoty a odošle požiadavku.
3. Systém získa inštanciu entity z úložiska a vráti jej hodnoty.
4. Ukončenie prípadu použitia.

Tento prípad použitia je na nižšej úrovní ako bežné prípady použitia a predstavuje každú transakciu vo všetkých scenároch bežných prípadov použitia. Tento prípad použitia môžu prípady použitia, ako je napr. logovanie, rozšíriť. Napríklad pri logovaní sa definuje rozšírenie, že pri kroku 2 základného toku sa ma požiadavka uložiť do logovacieho súboru. Účastník a prípad použitia vykonania transakcie majú parametrizované meno, to znamená, že skutočné meno získajú až pri vykonávaní z prostredia. Takto bude každá odoslaná požiadavka v bežných prípadoch použitia logovaná. Takéto prípady použitia sa nazývajú infraštruktúrne prípady použitia.



Obr. 2.2: Infraštruktúrny prípad použitia.

Teraz upriamime pozornosť na väzby medzi prípadmi použitia. Väzby priamo súvisia s tokmi v prípadoch použitia. Tok predstavuje vykonávanie scenára inštancie prípadu použitia. Prípad použitia môže mať rôzny počet tokov a môžu byť rôzneho typu. UML definuje štyri typy tokov: základný tok (basic flow), alternatívny tok (alternate flow), podtok (subflow) a rozširujúci tok (extension flow). Aby sme toky navzájom rozlíšili, tak každý tok má svoje meno (vyznačené kurzívou). Zoberme si napríklad scenáre

prípadov použitia *Naúčtuj nákupu*, *Vyhľadaj tovaru* a *Vypíš účtovanie do pokladničného bloku*:

Prípad použitia Naúčtuj nákup:
Stručný popis:
Pokladník postupne naúčtuje všetky položky nákupu.
Základné toky:
Z1. Naúčtovanie nákupu
Prípad použitia začne, ak chce pokladník naúčtovať nákup.

1. Pokladník spustí účtovanie nového nákupu.
2. Systém na obrazovke zobrazí prázdnu tabuľku predaných položiek nákupu a celkovú sumu nákupu nastavenú na 0.
3. Pokladník priloží čiarový kód tovaru k čítačke čiarových kódov alebo ručne zadá kód tovaru.
4. Spustí sa Vyhľadanie tovaru.
5. Systém pripočíta cenu identifikovanej položky k celkovej cene nákupu a do tabuľky pridá meno tovaru, cenu tovaru a celkovú sumu nákupu.
6. Skladník pokračuje krokom číslo 3 alebo zrušením naúčtovanej položky, alebo oznámi ukončenie nákupu.
7. Systém zaeviduje nákup a ukončí tok.

Alternatívne toky:
A1. Zrušenie naúčtovanej položky
Ak pokladník v šiestom kroku toku Z1 oznámi zrušenie naúčtovanej položky nákupu.

1. Pokladník priloží čiarový kód tovaru k čítačke čiarových kódov alebo ručne zadá kód naúčtovanej položky tovaru, ktorý chce zrušiť.
2. Spustí sa Vyhľadanie tovaru.
3. Skontroluje tabuľku predaných položiek, či sa tam identifikovaný tovar nachádza.
4. Systém odpočíta cenu identifikovanej položky z celkovej ceny nákupu a vypíše meno tovaru, cenu tovaru a celkovú cenu nákupu.
5. Pokladník pokračuje krokom číslo 6 toku Z1.

A2. Položka na zrušenie neexistuje.
Ak systém v kroku 3 toku A1 alebo kroku zistí, že požadovaná položka na zrušenie sa medzi naúčtovanými položkami nákupu nenachádza, tak zobrazí hlásenie „Položka nebola naúčtovaná“ a pokračuje krokom 6 toku Z1.

1. Systém vypíše na obrazovku hlásenie „Položka nebola vyúčtovaná“.
2. Pokračuje krokom 6 toku Z1.

A3. Tovar neexistuje.
Ak systém v kroku 4 toku Z1 alebo v kroku 2 toku A1 nenájde tovar v databáze, tak vypíše „Tovar neexistuje“ a pokračuje krokom 6 toku Z1.

1. Systém vypíše na obrazovku hlásenie „Tovar neexistuje“.

2. Pokračuje krokom 6 toku Z1.

Body rozšírenia:
EP1. Potvrdenie zrušenia položky
Bod rozšírenia Autentifikované zrušenie položky nastane v kroku 4 toku A1.

EP2. Tlač – naúčtovaná položka
Bod rozšírenia Tlač – naúčtovaná položka nastane v kroku 5 toku Z1.

EP3. Tlač – zrušená položka
Bod rozšírenia Tlač – zrušená položka nastane v kroku 4 toku A1.

EP4. Tlač – suma
Bod rozšírenia Tlač – suma nastane v kroku 7 toku Z1.

Prípad použitia Vyhľadaj tovar:
Stručný popis:
Tento prípad použitia popisuje priebeh vyhľadávania tovaru v databáze.
Podtoky:
P1. Vyhľadanie tovaru
Systém identifikuje tovar a poskytne detaily o tovare.

1. Systém v databáze vyhľadá tovar podľa čiarového kódu položky a zistí detaily tovaru.

Prípad použitia Vypíš účtovanie do pokl. bloku:
Stručný popis: Systém zapíše na pokladničný blok informácie o každej naúčtovanej alebo zrušenej položke nákupu a tiež celkovú sumu naúčtovaného nákupu.
Rozširujúce toky:
EF1. Zapísanie do bloku - zrušené
Rozširujúci tok sa odohrá v bode rozšírenia Tlač – zrušená položka v prípade použitia Naúčtovanie nákupu, keď sa úspešne zruší naúčtovaná položka nákupu.

1. Systém vytlačí na ďalší riadok bloku meno tovaru, cenu tovaru so znamienkom mínus pred cenou a DPH.
2. Systém pokračuje v existujúcom toku.

EF2. Zapísanie do bloku - naúčtované
Rozširujúci tok sa odohrá v bode rozšírenia Tlač – naúčtovaná položka v prípade použitia Naúčtovanie nákupu, keď sa úspešne naúčtuje položka nákupu.

1. Systém vytlačí na ďalší riadok bloku meno tovaru, cenu tovaru a DPH.
2. Systém pokračuje v existujúcom toku.

EF3. Zapísanie do bloku - suma
Rozširujúci tok sa odohrá v bode rozšírenia Tlač – suma v prípade použitia Naúčtovanie nákupu, keď sa ukončí nákup.

1. Systém vytlačí na ďalší riadok bloku celkovú sumu.

2. Systém pokračuje v existujúcom toku.

Základný tok *Z1. Naúčtovanie nákupu* predstavuje typický scenár prípadu použitia. Môže byť definovaných viacero základných scenárov, ak je možných viacero spôsobov vytvorenia inštancie prípadu použitia. Ak prípad použitia nemá základný tok, tak to znamená, že sa nemôže vytvoriť inštancia tohto prípadu použitia (používateľ nemôže aktivovať daný prípad použitia). Prípady použitia bez základného toku sa nazývajú „*utility use cases*“. Takýto prípad použitia je napríklad aj príp. použitia *Vyhľadaj tovar*. Ten obsahuje iba podtok *P1.Vyhľadanie tovaru*. Je to dôsledok toho, že je to tzv. inclusion prípad použitia, t.j. prípad použitia, ku ktorému smeruje väzba include, a ktorý používateľ explicitne neaktivuje. Väzba include znamená, že prípad použitia, od ktorého väzba smeruje, obsahuje tok prípadu použitia, ku ktorému väzba smeruje (t.j. inclusion prípad použitia). Teda prípad použitia typu inclusion definuje podtok prípadu použitia, od ktorého smeruje šípka. Podtok je vyňatá časť scenára, ktorý chceme z nejakého dôvodu osamostatniť. Najčastejším dôvodom je možnosť jednoduchšieho znovupoužitia tejto časti scenára, či už v alternatívnych scenároch rovnakého prípadu použitia, alebo v iných prípadoch použitia. Pri použití aj v iných prípadoch použitia je potrebné vytvoriť samostatný inclusion prípad použitia, v ktorom sa bude tento podtok nachádzať (v našom prípade je použitý v ďalších dvoch prípadoch použitia, preto bol vyňatý do samostatného prípadu použitia). Je dôležité si uvedomiť, že hlavný tok odvolávajúci sa na podtok na nejakom mieste svojho scenára (v našom prípade je dané miesto podčiarknuté), je od tohto podtoku závislý. Preto ho väčšinou nie je možné odstrániť bez zásahu do scenára hlavného toku (analogicky – nie je možné odstrániť väzbu include bez zásahu do scenára prípadu použitia, od ktorého väzba vychádza). Hlavný tok potom nemusí mať k dispozícii všetky potrebné informácie, aby mohol pokračovať. Taktiež je dôležité, že pri prechádzaní do podtoku sa nevytvára nová inštancia inclusion prípadu použitia, ale jeho scenár sa akoby skopíruje do scenára hlavného toku v mieste volania podtoku a vykonáva sa v inštancii hlavného toku. To znamená, že hlavný scenár môže narábať so všetkými hodnotami, ktoré vygeneroval podtok.

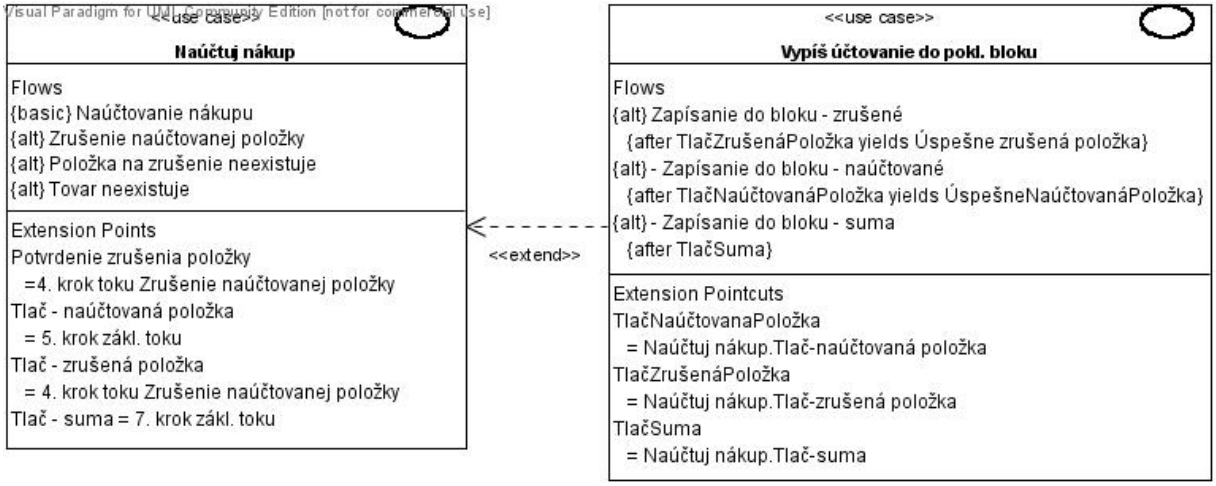
Podtok môže byť na mieste jeho volania označený aj ako *abstract*. Prípad použitia, ktorý takýto abstraktný podtok zahŕňa, musí byť generealizáciou nejakých iných prípadov použitia (čo je v diagrame zobrazené väzbou generalizácie) a tie musia tento podtok dodefinovať sami.

Alternatívne toky určujú, čo sa ma stať v situáciách, ktoré nie sú popísané v základnom scenári. Väčšinou sa používajú na zachytenie výnimiek.

Rozširujúce toky sa nachádzajú v tzv. extension prípadoch použitia. Sú to prípady použitia, ktoré rozširujú existujúce prípady použitia (nazývané rozšírené (extended) alebo základné (base) prípady použitia). Z hľadiska AOSD hrajú kľúčovú úlohu, pretože sú analógiou k videniam v AOP. Extension prípady použitia sa používajú na znázornenie nejakej prídavnej služby alebo doplnenie ďalších možností k existujúcim prípadom použitia. Hlavným rozdielom oproti inclusion prípadom použitia je ten, že existujúci prípad použitia nie je závislý od extension prípadu použitia, ktorý ho

rozširuje a preto o ňom nemusí ani vedieť, že existuje. Je to znázornené aj smerom šípky, ktorá smeruje od extension prípadu použitia k existujúcemu prípadu použitia, čiže extension prípad použitia je závislý na existujúcom prípade použitia. Preto odstránením extension prípadu použitia nezabránime vykonaniu scenára v existujúcom prípade použitia. Definovanie miest, v ktorých sa môže spustiť rozširujúci tok sa robí v existujúcich prípadoch použitia pomocou tzv. bodov rozšírenia (extension points). Každý bod rozšírenia je pomenovaný a neformálnym popisom je naznačené, na ktorom mieste v scenári sa môže spustiť rozširujúci tok. Tieto body rozšírenia sú analógiou k tzv. bodom spájania (join points) v AOP. Pri každom rozširujúcom toku je potom odkaz na meno bodu rozšírenia, kde sa má rozširujúci tok spustiť. Tieto odkazy sa nazývajú bodové prierezy rozšírenia (extension pointcuts) a sú analógiou k bodovým prierezom v AOP. Tieto odkazy sú v našom textovom opise popísané neformálne, keďže prípady použitia oficiálne nepodporujú tieto prvky. Bodové prierezy rozšírenia môžu odkazovať aj na konkrétny krok v scenári, ale nie je to vhodné, pretože tvorca rozširujúceho prípadu použitia nemá tento scenár pod kontrolou. V prípade zmeny scenára by sa rozširujúci tok už nemusel správať podľa predstav jeho tvorcu. Na odkazovanie sú vhodné práve mená bodov rozšírenia, ktoré tvoria akési nemenné rozhranie, za ktorým sa skrýva skutočné miesto rozšírenia definované tvorcom scenára rozširovaného prípadu použitia. Z obrázka (obr. 2.1) si je možné všimnúť, že UML diagram prípadov použitia podporuje iba vizualizáciu bodov rozšírenia a prostredníctvom poznámky k väzbe extend aj názov rozširujúceho toku (na obrázku pod skratkou EF), a podmienku jeho aktivácie (na obrázku pod skratkou EC). V prípade väčšieho počtu väzieb extend sa diagram stáva neprehľadným.

Ivar Jacobson preto odporúča namiesto bežnej notácie prípadov použitia (názov prípadu použitia vo vnútri oválu) používať notáciu, aká sa používa pri triedach, keďže prípady použitia ako aj triedy sú vlastne špeciálne typy klasifikátora. Klasifikátor v UML je každá vec, z ktorej sa dá vytvoriť inštancia. Od neho sa potom môžu odvodiť konkrétnejšie typy akým je aj prípad použitia. Okrem toho zaviedol do notácie prípadov použitia aj nové „formálnejšie“ prvky, aby uľahčil zobrazenie do AOP. Prirodzený jazyk používaný v prípadoch použitia je príliš neformálny na to, aby sme štruktúru diagramu prípadov použitia jednoducho zobrazili do implementačnej úrovne AOP, ktorá je formálna. Preto sa môže zdať, že AOSD pomocou prípadov použitia nepokrýva AOP. Takýmito prvkami sú napríklad presné označenie typu toku v zložených zátvorkách (*{alt}* pre alternatívny tok a rozširujúci tok, *{sub}* pre podtok, *{basic}* pre základný tok, *{abstract}* pre abstraktný tok.), priradenie bodov rozšírenia k bodovým prierezom rozšírenia pomocou bodkovej konvencie podľa mena prípadu použitia a mena bodu rozšírenia, relatívny čas kedy, vzhľadom k miestu rozšírenia, dochádza k rozšíreniu (*after*, *before*, *around*) (obr. 2.3). Na označenie typu toku sa v UML používajú tagy. UML síce umožňuje používanie vlastných tagov, no I. Jacobson tvrdí, že by tieto tagy mali byť aj priamo v špecifikácii UML, keďže odrážajú štandardnú prax pri písaní prípadov použitia. Samotné zobrazenie tokov a bodových prierezov rozšírenia v prípade použitia v spojení s UML už ale nespomína, no dá sa predpokladať, že je tam využitá možnosť vytvorenia si vlastných streotypov.

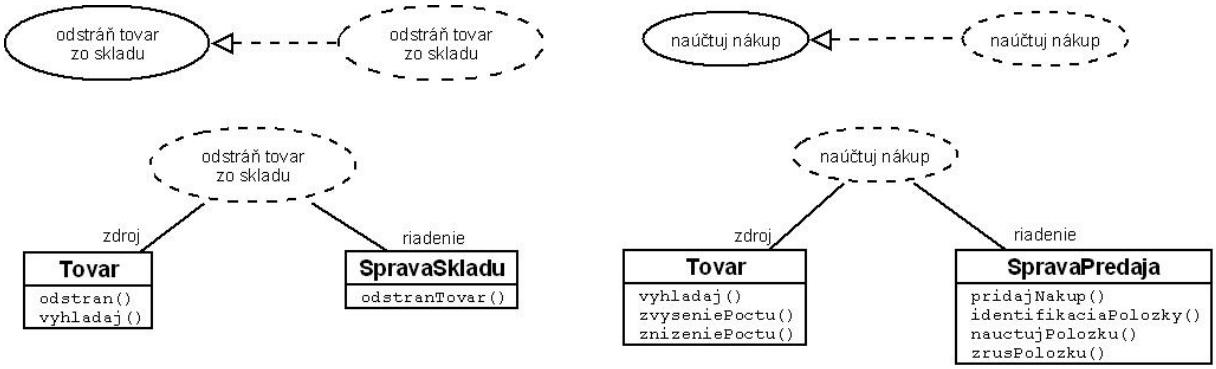


Obr. 2.3: Prípady použitia v štvorcovom tvare s poliami ako ich definuje I. Jacobson.

Zatiaľ boli pomocou prípadov použitia oddelené jednotlivé záležitosti, ktoré sa vyskytovali v požiadavkách. To ale nie je všetko, pretože je potrebné zachovať záležitosti oddelené aj počas návrhu a implementácie. Zachovávanie oddelených záležitostí sa robí pomocou novej modulárnej jednotky *use-case slice*. Use-case slice obsahuje špecifiká modelu v jednom balíku. Use-case slice analytického alebo návrhového modelu obsahuje triedy, rozšírenia k existujúcim triedam zorganizované v aspektoch a zodpovedajúce kolaborácie, ktoré realizujú prípad použitia.

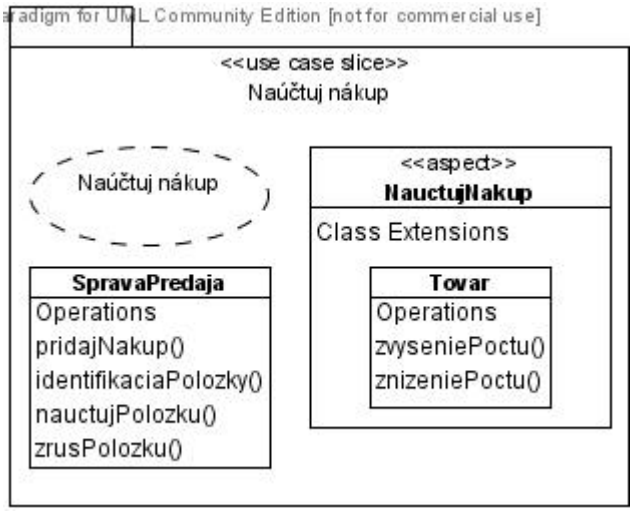
2.2 Peer prípady použitia

Oddelenie záležitosti v analytickom alebo návrhovom modeli sa môže porušiť v prípade realizácie tzv. peer prípadov použitia alebo extension prípadov použitia. Peer prípady použitia sú také, medzi ktorými nie je väzba. Znamená to, že sú od seba navzájom nezávisle a mohli by sa vykonávať paralelne, ale počas ich realizácie zistíme, že majú dopad na nejakú spoločnú triedu. Realizácia prípadov použitia sa robí pomocou diagramu kolaborácie, ktorý môže byť vyjadrený sekvenčným diagramom, diagramom komunikácie alebo diagramom tried. V prípade peer use case ide najmä o štruktúru tried, preto bude najvhodnejšie použiť diagram tried. Ten bude obsahovať všetky triedy a ich atribúty a operácie, ktoré sa podieľajú na realizácii daného prípadu použitia, tzv. rozšírenia triedy. Základné triedy, ktoré sa budú v systéme vyskytovať, zistíme napríklad z domény problému alebo vytvorením sekvenčného diagramu prípadov použitia. Ako príklad si vezmime prípady použitia *Odstráň tovar zo skladu* a *Naúčtuj nákup*. Tieto prípady použitia nemajú medzi sebou žiadnu väzbu a dokonca sú záležitosťami rôznych účastníkov. No pri realizácii sa ukáže, že používajú rovnakú triedu *Tovar* (obr. 2.4).



Obr. 2.4: Realizácia prípadov použitia Odstráň tovar zo skladu a Naúčtuj nákup.

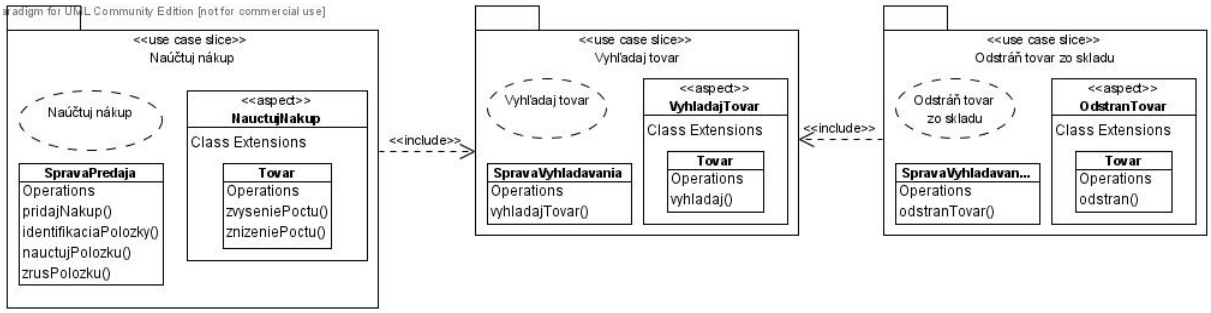
To znamená, že tieto záležitosti sú prepletené. Ale je možné ich rozpliesť, keďže vieme o existencii AOP a medzitypovej deklarácii. Vytvorí sa use-case slice *Naúčtuj nákup*, ktorý vďaka AOP bude obsahovať iba špecifiká, ktoré tento prípad použitia realizujú (obr. 2.5). To isté platí aj pre *Odstráň tovar zo skladu*.



Obr. 2.5: Use-case slice Naúčtuj nákup.

Ale je tu ešte problém s metódou *vyhladaj()* triedy *Tovar*, ktorá sa nachádza v use-case slice *Naúčtuj nákup* ako aj v *Odstráň tovar zo skladu*, t.j. je zdieľaná. Tento jav sa nazýva prekrývanie (overlap). Nie je možné ju ponechať v tomto stave, pretože v AO jazyku nám to nebude fungovať a taktiež nie je možné ju nechať iba v jednom use-case slice, pretože by medzi nimi vznikla závislosť a potom by to sa už nejednalo o peer prípady použitia. Rieši sa to tromi spôsobmi. Buď sa vyjme do nového use-case slice, resp. príp. použitia a vytvorí sa závislosť include, alebo sa vyjme a definuje v nadriadenom prípade použitia, alebo sa vyjme do tzv. non-use-case slice. Non-use-case

slice obsahuje iba základ triedy, ktorú ostatné use-case slice podľa vlastnej potreby rozširujú, čo je zaznačené väzbou extend medzi týmito use-case slice a non-use-case slice. Non-use-case sa nazýva preto, lebo neobsahuje žiadne aspekty a jeho pôvod ani nie je v prípade použitia. Je len akýsi reprezentátor domény a poskytuje ostatným use-case slice prístup na jednom mieste k triedam, ktoré sú typické pre doménu problému a sú často využívané viacerými use-case slice. V našom prípade by non-use-case slice obsahoval triedu *Tovar* s jednou metódou *vyhladaj()* a use-case slice *Naúčtuj nákup* a *Odstráň tovar* zo skladu by sa na neho pripojili väzbou extend. V prípade nadriadeného prípadu použitia vznikne use-case slice, ktorý bude obsahovať triedu, ktorá bude implementovať metódu *vyhladaj()*. Ostatné use-case slice budú od tohto use-case slice dediť. Pre náš prípad sa zdá byť najlepšia voľba väzba include. V diagrame prípadov použitia je totiž prípad použitia, ktorý priamo s vyhľadávaním súvisí a zahrňujú ho takmer všetky ostatné prípady použitia. Pre nás je najdôležitejšie aby ho zahrňovali prípady použitia, ktoré na svoju realizáciu vyžadujú metódu *hladaj()* triedy *Tovar*. Use-case slice prípadu použitia *Vyhľadaj tovar* bude mať rozšírenie triedy *Tovar* o metódu *hladaj()* vo vnútri aspektu. Ostatné use-case slice, ktorých kolaborácie používajú túto metódu triedy *Tovar* budú väzbou include spojené s use-case slice *Vyhľadaj tovar* (obr. 2.6). Väzba include medzi use-case slice znamená, že okrem aspektov má use-case slice, ktorý zahrňuje, k dispozícii všetky prvky, ktoré sa nachádzajú v zahrňovanom use-case slice, t.j. kolaborácie, triedy a rozšírenia tried.



Obr. 2.6: Väzba include medzi use-case slice.

Už si je možné aj predstaviť, ako to bude vyzerať na úrovni AOP (tab. 1).

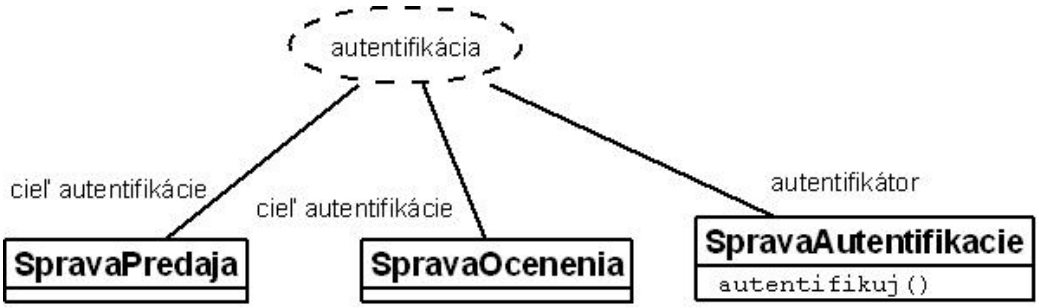
Tab. 1: Mapovanie UML konštrukcií na AOP konštrukcie (Jacobson, et al., 2004).

UML konštrukcia	**AOP konštrukcia**
Use-case slice	Package
Kolaborácia	-
Aspect	Aspect
Class extension	Zoskupenie medzitypových deklarácií pre každú triedu

2.3 Extension prípady použitia

Pri realizácii peer prípadov použitia sme sa museli vysporiadať len s kompletnými operáciami a zaujímala nás iba štruktúra triedy, teraz nás bude zaujímať aj vykonávanie samotnej operácie. V prípade extension prípadov použitia je potrebné pridať správanie do existujúcich operácií na miesta definované bodovými prierezmi. Je nutné určiť dve veci, pokiaľ chceme identifikovať miesto vykonávania rozšírenia operácie. Je to štrukturálny kontext a kontext správania. Štrukturálny kontext opisuje, kde v návrhovanej štruktúre bude rozšírenie operácie. To znamená, v rámci ktorého balíka, triedy a operácie bude rozšírenie vykonávané. Kontext správania opisuje bod v toku vykonávania, kde rozšírená operácia rozšíri existujúcu operáciu. Tieto body sú v prípadoch použitia definované ako body rozšírenia a v prípade AOP sú to body spájania.

V našom príklade boli identifikovali triedy, v ktorých je záležitosť autentifikácie roztrúsená (obr. 2.7).

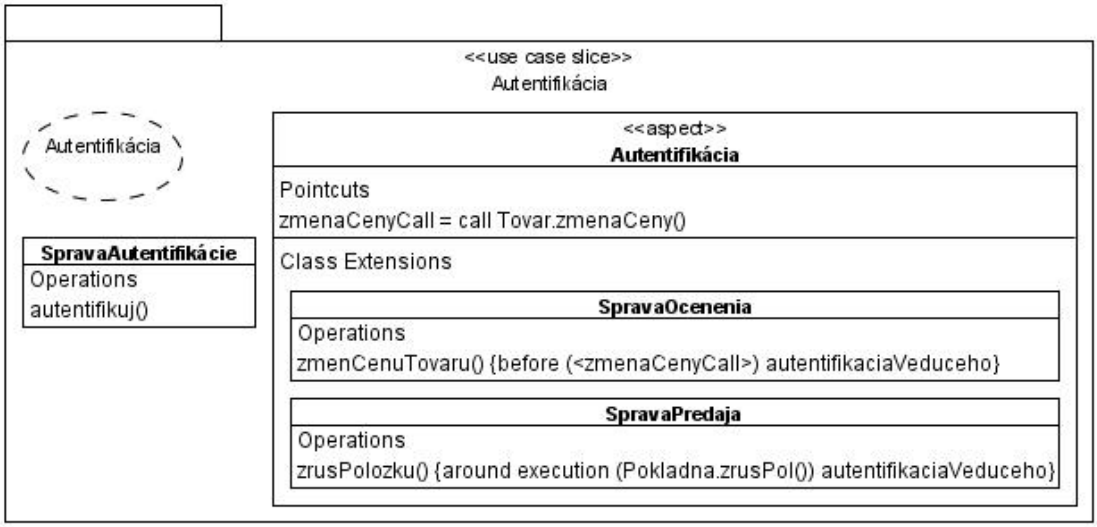


Obr. 2.7: Realizácia prípadu použitia autentifikácia.

Štruktúra use case slice teraz bude navyše obsahovať ďalšie prvky, ktoré sú nevyhnutné na popísanie rozsahu záležitosti autentifikácie, pretože táto záležitosť pracuje s jemnejšou granuláciou štruktúry triedy ako záležitosti pri peer prípadoch použitia (je tu navyše kontext správania) (obr. 2.8). Definícia operácií v rozšíreniach tried sa skladá z názvu operácie, v rámci ktorej bude sa bude vykonávať rozšírenie operácie, bod rozšírenia (v AOP bodový prierez), kľúčové slovo, ktoré hovorí, kde relatívne k bodu rozšírenia, sa bude rozšírenie vykonávať a nakoniec text indikujúci čo rozšírenie operácie robí. Na obrázku (obr. 2.8) je to v rámci operácie *zrusPolozku()* triedy *SpravaPredaja*, v okolí bodu rozšírenia *execution Pokladna.zrusPol()* a bude sa vykonávať autentifikácia vedúceho predaja. Neodporúča sa definovať bod rozšírenia priamo v definícii operácie rozšírenia triedy, ale odkazovať sa na neho prostredníctvom parametra. V našom prípade je takto spravený odkaz v operácii rozšírenia triedy *SpravaOcenenia*. Parametrizovaný odkaz znamená, že jeho hodnota nie je konštantná, ale získava ju počas behu z prostredia. Hodnotu môže napríklad získavať z časti *Poincuts*, v ktorej sú definované mená bodových prierezov, ku ktorým je priradený skutočný bod rozšírenia. V prípade zmeny názvu metódy *Tovar.zmenaCena()*, tak nie

je potrebné kontrolovať každé rozšírenie triedy, ale stačí zmeniť meno v časti *Poincuts*. Taktiež je to výhodné z hľadiska znovupoužiteľnosti bodového prierezu.

Je možné parametrizovať aj iné hodnoty, napr. meno triedy rozšírenia, názov operácie, v ktorej sa bude rozšírenie vykonávať, atď. Pri definovaní hodnoty bodového prierezu je možné použiť aj regulárne výrazy. Aby nedošlo k nejednoznačnosti medzi skutočným názvom a parametrom, tak parameter sa vkladá medzi zátvorky ‹›.



Obr. 2.8: Use-case slice Autentifikácia vo fáze návrhu.

Aspekt *Autentifikácia* by v progr. jazyku Aspect/J vyzeral približne takto:

```
1.  package foo.Autentifikacia;
2.  import app.veduci.SpravaOcenenia ;
3.  import app.pokladnik.SpravaPredaja ;
4.  import domain.tovar.Pokladna ;
5.  import domain.tovar.Tovar ;
6.
7.  public aspect Autentifikacia {
8.  pointcut zmenaCenyCall() :
9.     withincode(void SpravaOcenenia.zmenCenuTovaru())
10.             &&  call(void Tovar.zmenaCeny()) ;
11.
12. before () : zmenaCenyCall() {
13. // kod autentifikacie veduceho
14. }
15.
16.       void       around()       :       withincode(void
SpravaPredaja.zrusPolozku())
17.    && execution(void Pokladna.zrusPol()) {
18.  // kod autentifikacie veduceho
```

```
19. }
20. }
```

Rozšírenia je tiež možné generalizovať. Je to užitočné v prípade, ak by bolo rovnaké správanie pre autentifikáciu vedúceho ako aj pre autentifikáciu zmeny ceny, tak aj pre zrušenie položky na pokladni, ale mali by rôzne body rozšírenia. Správanie by bolo vo všeobecnom aspekte, v ktorom by ale rozdielne body rozšírenia boli definované ako abstraktné. Konkrétne aspekty autentifikácie ceny a autentifikácie zrušenia položky by zdedili správanie, ale body rozšírenia by si definovali sami. Na úrovni AOP by sa to riešilo štruktúrami abstraktných aspektov a ich dedením inými aspektmi.

Okrem use case slice sa v AOSD pomocou prípadov použitia spomína aj tzv. use-case modul. Use case modul obsahuje konkrétne slice z rôznych fáz vývoja softvéru, keďže v rôznych fázach ma slice rôznu podobu, hoci predstavuje tú istú záležitosť. Konkrétne sa jedná o use case specification slice (slice vo fáze špecifikácie), analysis slice (fáza analýzy), design slice (fáza návrhu), implementation slice (fáza implementácie), test implementation slice (fáza testovania implementácie) a test design slice (fáza testovania návrhu). Use case modul, tak zahŕňa všetky potrebné informácie pre všetky fázy vývoja softvéru jednej záležitosti.

3 Prístup Theme

Prístup Theme predstavuje komplexný prístup k AOSD. Slúži na identifikovanie aspektov v požiadavkách a ich modelovanie vo fáze analýzy a návrhu. Myšlienka pochádza od ľudí Siobhán Clark a Elisa Baniassad. Sú autorkami knihy o prístupe Theme, v ktorej detailne popisujú Theme/Doc a Theme/UML (Clarke, et al., 2005). Theme/Doc je prístup na identifikovanie aspektov v požiadavkách a ich modelovanie vo fáze analýzy a Theme/UML popisuje, ako tieto získané aspekty modelovať vo fáze návrhu pomocou bežnej notácie UML bez ohľadu na programovací jazyk. Prístup Theme je založený na tzv. témach, ktoré označujú záležitosti. Témy môžu byť prepojené dvomi spôsobmi (alebo nemusia byť vôbec prepojené). Prvý spôsob sa nazýva tzv. zdieľanie konceptu. Každá téma opisuje triedy, ktoré požaduje zo svojej perspektívy, bez ohľadu na to, že aj iné témy môžu mať triedy, ktoré opisujú rovnaký koncept. Tieto triedy väčšinou súvisia s doménou problémy, ktorý systém rieši. Ak nejaká téma potrebuje na svoju realizáciu nejakú triedu, tak je zahrnutá do témy, ale obsahuje iba časti, ktoré sú pre ňu relevantné. Všetky časti sa objavia spolu v jednej triede až vo fáze kompozície.

Druhý spôsob sa nazýva pretínanie. Jedna téma pretína druhú ak jej priebeh je aktivovaný priebehom druhej témy. Pretínajúce témy sa nazývajú aspektové (*aspect* themes) a nepretínajúce, nad ktorými aspekty operujú, sa nazývajú základné témy (base themes).

Už tu je vidieť analógiu medzi prípadmi použitia a témami. Táto práca sa zaujíma najmä Theme/Doc prístupom, pretože súvisí s analýzou a preto ho je možné porovnávať s prípadmi použitia.

3.1 Theme/Doc

Úlohou Theme/Doc (skratka pre Themes in documentation) je identifikovať tzv. základné a aspektové témy a vzťahy medzi nimi. Theme/Doc nie je nástroj, ktorý za nás identifikuje témy a ich vzťahy. On len odporúča postupy a pomáha lepšie vizualizovať vzťahy medzi požiadavkami a témami. Theme/Doc pozná tri typy vizualizácie:

- pohľad tém a vzťahov
- pohľad pretínajúcich tém
- individuálny pohľad

Vymenované pohľady sú v takom poradí, v akom nasledujú v Theme/Doc. Pri identifikovaní tém a vzťahov sa pracuje s rôznymi operáciami nad požiadavkami a témami ako je pridanie, odstránenie, rozdelenie, zoskupenie tém, pridanie, rozdelenie, pripojenie k téme, asociovanie a odročenie požiadaviek (postpone). Niektoré operácie sú špecifické iba pre určité procesy a pohľady. V analýze prostredníctvom Theme/Doc prebiehajú tieto procesy:

- Rozhodnutie o témach

- Určenie zodpovedností tém
- Plánovanie návrhu

Tieto procesy síce prebiehajú za sebou tak, ako sú uvedené, ale nie sú striktne oddelené. To znamená, že je možné sa vrátiť do predchádzajúceho procesu. Tieto procesy sa môžu cyklicky opakovať, až kým nie sme s výsledkom spokojní.

3.2 Rozhodnutie o témach

Rozhodnutie o témach súvisí s identifikovaním tém v požiadavkách. V tejto fáze sa pracuje s pohľadom tém a vzťahov. Tento pohľad zobrazuje prepojenie tém s požiadavkami, ktoré sa danej témy týkajú. V tomto pohľade sú témy zobrazené ako uzly tvaru kosoštvorca a požiadavky ako uzly tvaru obdĺžnika so zaoblenými rohmi. Hrany predstavujú prepojenie medzi požiadavkami a témami a celý pohľad tak tvorí graf. Hrany môžu byť iba medzi požiadavkou a témou. Ak je požiadavka prepojená s viacerými témami, tak sa tieto požiadavky nazývajú zdieľané (shared). Ak požiadavka nemá žiadne spojenie s témou, tak sa nazýva osirotená (orphaned).

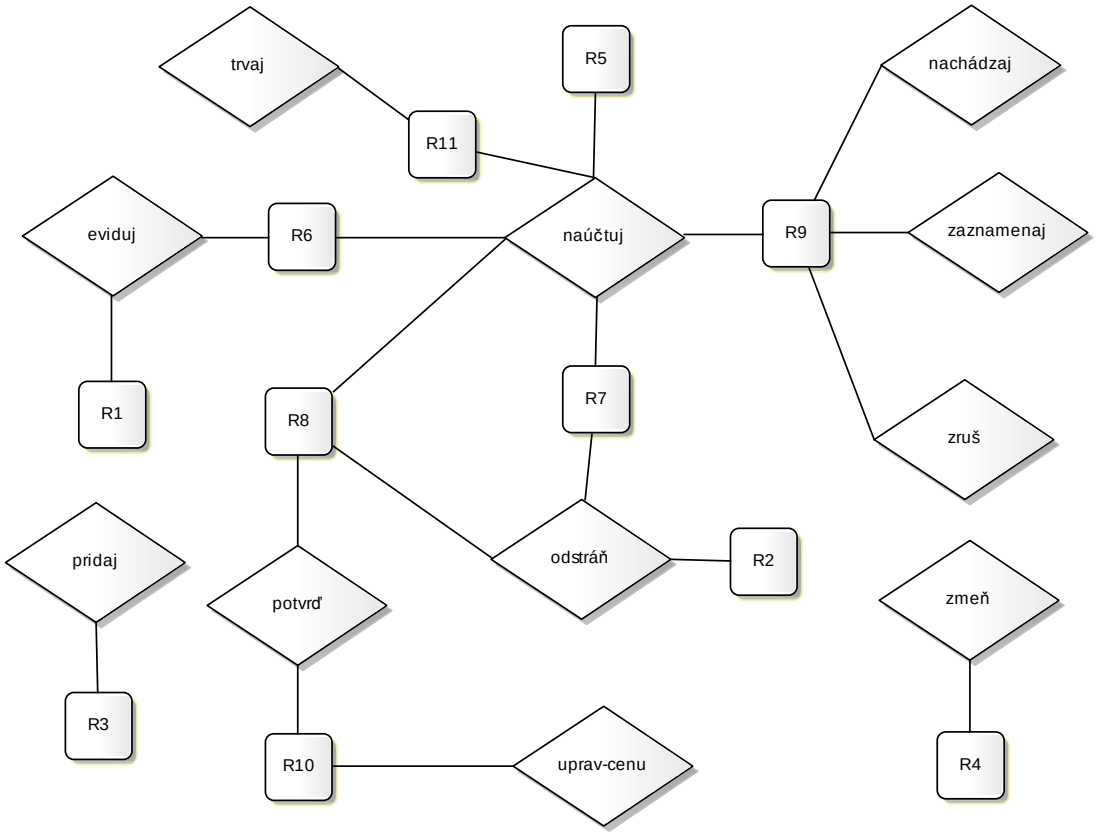
V prvej fáze rozhodnutia o témach sa určujú inicializačné témy. Inicializačné témy môžu byť získané napríklad z prípadov použitia, kde každý prípad použitia predstavuje jednu inicializačnú tému. Pokiaľ je k dispozícii iba zoznam požiadaviek, tak je možné postupovať jednou z troch možností (Clarke, et al., 2005).

Prvá možnosť je identifikovanie len kľúčových funkcionalít v požiadavkách, ktoré sa označia ako potenciálne témy. V príklade, ktorý bol vyššie rozoberaný, by to boli napr. funkcionality *eviduj stav tovaru*, *odstráň tovar*, *pridaj tovar*, *zmeň množstvo tovaru*, *naúčtuj položky*, *odstráň položky*, *vypíš na pokl. blok*. V takomto prípade sa potom budú tieto témy, v prípade, že budú príliš všeobecné alebo nie súdržné, rozdeľovať.

Druhá možnosť je presne opačný prístup. V takomto prípade sa za potenciálne témy označia všetky slovesá, ktoré sa nachádzajú v požiadavkách. Na to je možné použiť aj nejaký program, ktorý dokáže rozoznať slovesá v texte. Takto sa získa veľké množstvo potenciálnych tém, preto bude potrebné zoskupiť tieto slovesá do väčších tém.

Tretia možnosť je kombináciou predchádzajúcich dvoch možností. Prechádza sa zoznamom požiadaviek a vyčlení sa všetko, čo sa javí ako záležitosť, správanie alebo črta. Predstavme si, že v našom príklade boli identifikované tieto potenciálne témy: *eviduj*, *odstráň*, *pridaj*, *zmeň*, *naúčtuj*, *zruš*, *zaznamenaj*, *uprav*, *trvaj*, *nachádzaj*, *potvrď*.

Na obrázku (obr. 3.1), je zobrazený pohľad tém a vzťahov pre tento príklad.



Obr. 3.1: Prvotný pohľad tém a vzťahov.

S týmito potenciálnymi témami je možné ďalej pracovať. Najprv sa rozdelia témy, ktoré sú príliš všeobecné. Takouto témou je napríklad téma *odstráň*. V zozname požiadaviek sa odstránenie používa na dve rozdielne veci – na odstránenie naúčtovanej položky a na odstránenie tovaru z databázy tovaru. Preto sa táto téma rozdelí na dve konkrétnejšie témy. Zanikne tak téma *odstráň* a vzniknú témy *odstráň-položku* a *odstráň-tovar*. Taktiež téma *eviduj* je príliš všeobecná. V zozname požiadaviek má dva významy. V 1. požiadavke znamená, že zakúpený tovar sa zaeviduje do databázy a v 6. požiadavke znamená, že položka, ktorá sa má naúčtovať sa v databáze tovaru nachádza. Preto sa táto téma rozdelí na *identifikuj-položku* a na *zaeviduj-nákup*. Ďalšie rozdeľovanie nie je potrebné.

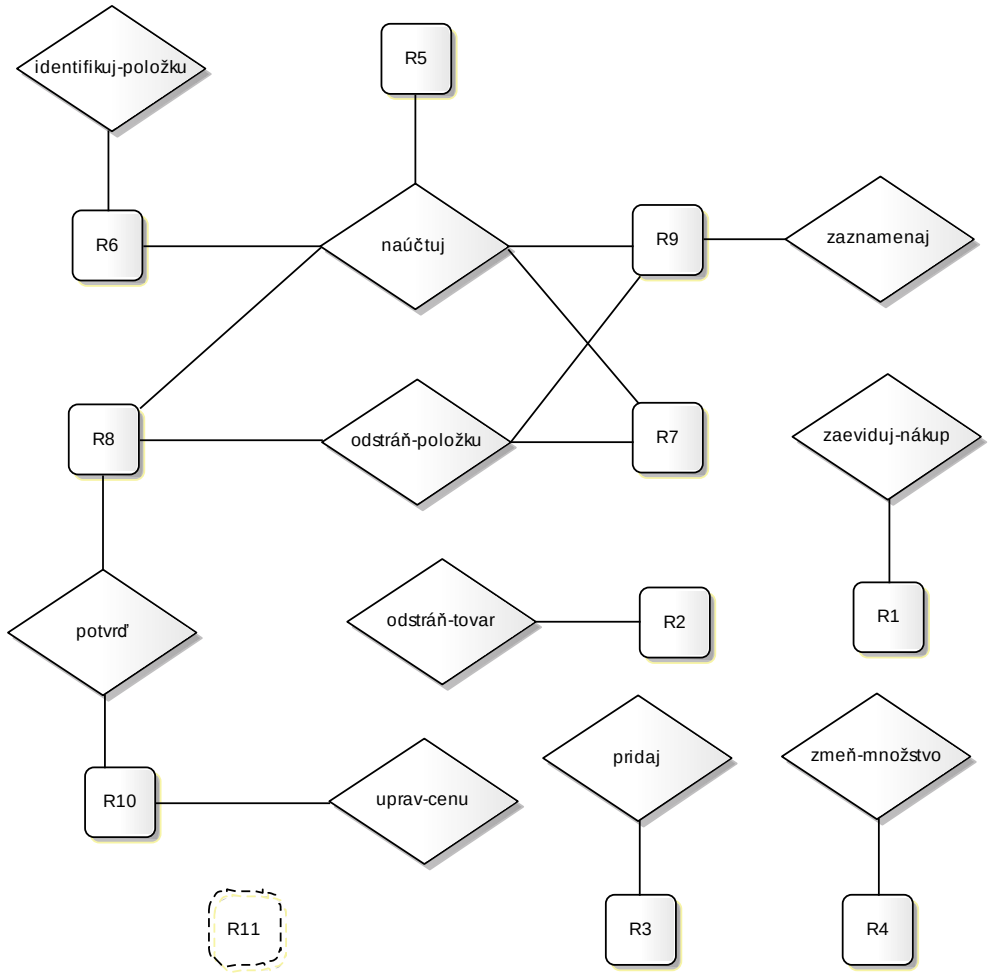
Teraz by sme mali prejsť ku zoskupovaniu. Zoskupovanie znamená zobrazenie viacerých tém ako jednu tému za účelom spriehľadnenia grafu. Tieto témy budú vystupovať ako jedna téma v pohľade tém a vzťahov a v pohľade pretínajúcich tém. Pri individuálnom pohľade bude téma opäť rozdelená na jednotlivé témy, ktoré zahrňovala. Náš graf nie je veľmi rozsiahli preto tento krok preskočíme.

V ďalšom kroku je potrebné zjednotiť témy, ktoré navonok vyzerajú odlišne, ale v skutočnosti majú rovnaký účel. Takéto témy sa najčastejšie objavujú kvôli synonymám, ktoré sa nachádzajú v zozname požiadaviek. Preto je nevyhnutné porovnať význam týchto slov, resp. tém. V našom grafe je to napríklad téma *zruš*, ktorá ma ten istý význam ako téma *odstráň-položku*. Tie sa zjednotia do jednej témy *odstráň-položku*. Taktiež témy *zaznamenaj* a *nachádzaj* môžu byť zjednotené do jednej témy s názvom *zaznamenaj*, keďže téma *nachádzaj* len presnejšie špecifikuje čo má téma *zaznamenaj* na starosti. V diagrame sa tiež nachádzajú synonymá *uprav* a *zmeň*, ale ich skutočný význam má v danom kontexte iný zmysel. Preto ostanú nezjednotené a pre istotu sa zmenia ich názvy tak, aby nespôsobovali pochybnosti, *uprav-cenu* a *zmeň-množstvo*.

Poslednou akciou je vymazanie tém, ktoré sú irelevantné alebo nie príliš dobre charakterizujú systém. V našom prípade sa vymaže téma *trvanie*.

Doteraz sme sa zaoberali témami. Teraz by mali byť prehodnotené požiadavky. V prvom rade určiť požiadavky, ktorými sa teraz nechceme zaoberať. Teraz sa napríklad nechceme zaoberať požiadavkou *R11*, preto bude označená prerušovanou čiarou a rozhodnutie o tejto požiadavke bude odročené do neskorších fáz vývoja softvéru. Ďalej môžu byť pridané ďalšie požiadavky, buď pridaním novej alebo rozdelením pôvodnej, alebo je možné osirotenú požiadavku priradiť k nejakej téme. Po pridaní požiadavky je nutné prehodnotiť jej dopad na ostatné témy. Rozdelenie požiadaviek môže tiež pomôcť rozpliesť témy.

Na obrázku (obr. 3.2) je zobrazený diagram po vykonaných akciách. Po tejto fáze nastáva fáza určenia zodpovednosti tém.



Obr. 3.2: Pohľad tém a vzťahov po úpravách.

3.3 Určenie zodpovednosti tém

V tejto fáze sa snažíme identifikovať pretínajúce témy. Aspekt v požiadavkách je definovaný ako funkcionalita, ktorej opis v rámci požiadavky je prepletený s opisom inej funkcionality. Preložením do reči tém je aspekt téma, ktorej opis v rámci požiadavky je prepletený s inou témou. Požiadavky, ktoré sú spojené s viac než jednou témou (zdieľané požiadavky) sú miesta, kde preplietanie nastáva (Clarke, et al., 2005). Preto je dôležité, ak je to možné, rozdeliť tieto zdieľané požiadavky. Existujú štyri hlavné pravidlá, ktoré je možné použiť na identifikovanie pretínajúcich tém. Ak nastane situácia, že platia všetky štyri, tak bola nájdená pretínajúca téma (Clarke, et al., 2005):

1. rozdelenie požiadavky nie je možné – je nemožné rozdeliť požiadavku tak, aby sa izolovali témy a odstránilo zdieľanie.
2. dominancia znamená asociáciu – téma, ktorá prevažuje pokrytím požiadavky nad ostatnými, je vybraná za kandidáta aspektu a asociuje s danou požiadavkou.
3. základná téma aktivuje aspekt – aspekty sú aktivované v rámci správania základnej témy.
4. aktivujúca a zároveň dominantná téma je externe aktivovaná vo viacerých situáciách (napr. situácia opísaná v inej požiadavke) alebo je aspekt dostatočne prepletený.

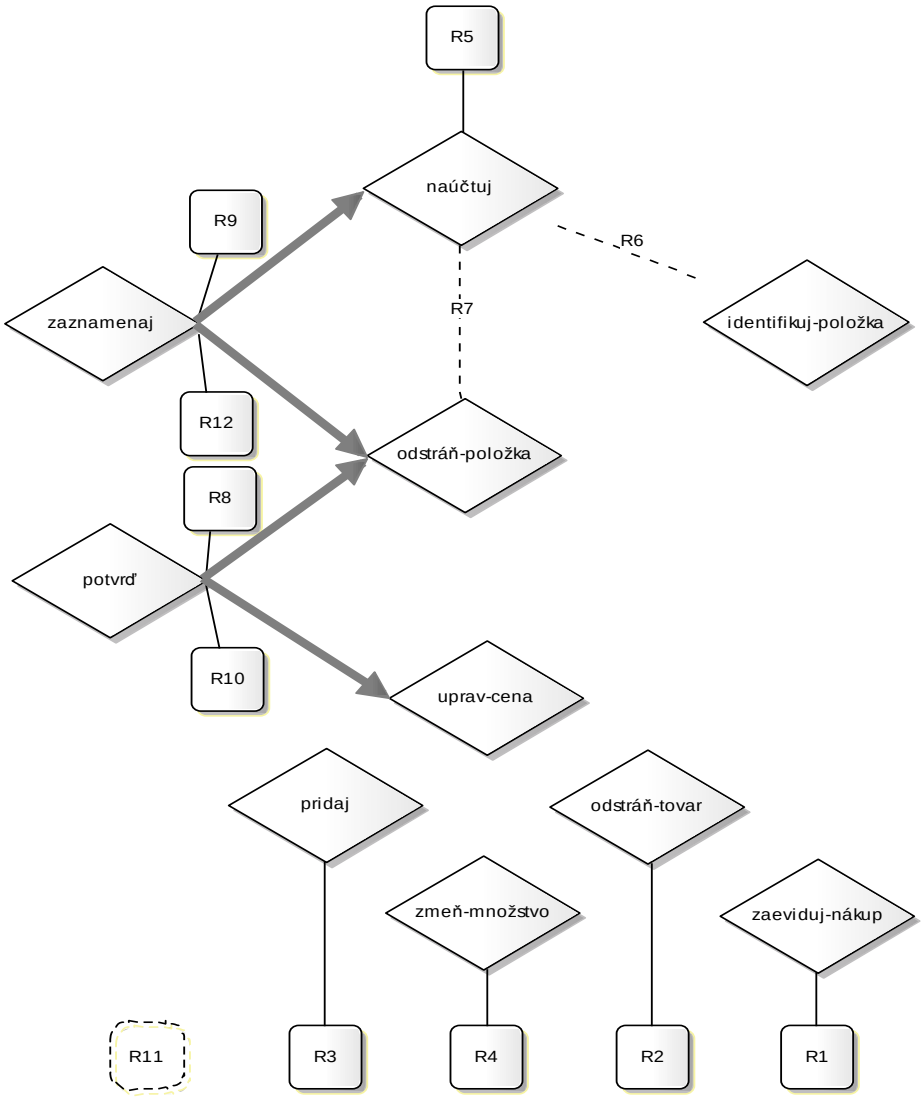
Začneme prvým bodom. V našom príklade je 5 zdieľaných požiadaviek, ktoré je potrebné sa pokúsiť prepísať tak, aby neboli zdieľané. Požiadavka *R8* môže byť upravená tak, že fráza „z naúčtovanej položky “ sa odstráni. Tento fakt, že je možné odstrániť len naúčtovanú položku, je už zaznamenaný v požiadavke *R7*. Odstráni sa tak spojenie s témou *naúčtuj*. Taktiež požiadavka *R9* sa rozdelí na dve časti, na zaznamenávanie naúčtovaných položiek s celkovou sumou (*R9*) a na naúčtovanie zrušených položiek (*R12*). Rozdelí sa tak trojnásobné zdieľanie požiadavky. To by bolo pre prvý bod všetko.

Druhý a nasledujúci bod veľmi pripomínajú prípady použitia, pri určovaní väzby extend. Jedným zo spôsobov, ako zistiť dominantnú tému (ak existuje), je skontrolovať požiadavku a pýtať sa, ktorá z tém by mala mať informácie o jej správaní (Clarke, et al., 2005). Často sa ako príklad uvádza logovanie, kde pri logovaní metódy nechceme, aby metóda niečo vedela o logovaní (resp. aby si uvedomovala logovanie). Cieľom je, aby sa metóda správala normálne, bez ohľadu pripojeného priebehu správania (logovania). Je to práve logovanie, ktoré musí poznať správanie metódy opísané danou požiadavkou. Pomôcť nám tiež môže pohľad na ostatné požiadavky, ktoré témy zdieľajú a podľa nich sa pokúsiť zistiť, ktorá téma potrebuje vedieť, kedy sa má spustiť (t.j. potrebuje poznať správanie, je závislá na požiadavke). Tento návod sa podobá návodu ako určiť, kde má byť väzba extend medzi prípadmi použitia. Alebo minimálne návodu na určovanie smeru väzby závislostí, čo je všeobecnejšia väzba zahrňujúca aj väzbu extend. V každom prípade, ďalšie dva body konkretizujú túto závislosť na väzbu extend. V našom príklade boli identifikované dominantné témy *potvrď* a *zaznamenaj*.

Tretí bod hovorí, že priebeh aspektu je aktivovaný nejakým priebehom v základnej téme. Aspektová orientácia umožňuje spustenie priebehu aspektu implicitne v mieste spúšťača. Ak nemáme aspect/base vzťah, potom musí byť priebeh aspektu spustený explicitne zo základnej témy (Clarke, et al., 2005). Zjednodušene, ak je k dispozícii AOP, tak to funguje, ak nie je, tak musí byť v základnej téme „ručne“ spustený aspekt. Ak to pre tému platí, tak splňuje tento bod. V našom prípade obe dominantné témy tento bod spĺňajú.

Štvrtý bod poukazuje na to, že existuje veľa prípadov, kedy jeden priebeh aktivuje druhý priebeh. Prakticky by tak bolo možné spraviť volanie každej metódy. Preto je potrebné si rozmyslieť, či sa aktivovanie priebehu pomocou aspektu oplatí. V prípade, že sa aktivuje priebeh len na jednom mieste v celom systéme, tak je zbytočné vyvolávať tento priebeh pomocou aspektov. V našom príklade témy *potvrď* a *zaznamenaj* vyhovujú aj tomuto bodu, čiže môžu byť označené za aspekty.

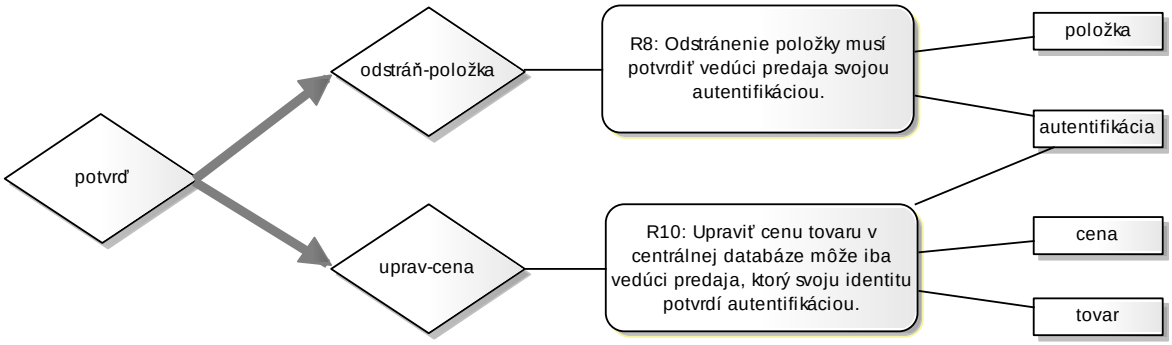
Na zobrazenie identifikovaných pretínajúcich tém slúži pohľad pretínajúcich tém (obr. 3.3). V ňom sú pomocou sivej šípky znázornené aspect/based vzťahy a asociáciou s aspektom je znázornená požiadavka, ktorej sa daný vzťah týka. Prerušovanou čiarou sú znázornené vzťahy, pri ktorých nebolo rozhodnuté, ktorá téma je základná a ktorá aspektová. Rozhodnutie sa tak presúva do neskorších fáz vývoja softvéru, kedy je k dispozícii viac informácii o ich správaní. Pri prerušovanej čiare sa nachádza aj požiadavka, ktorej sa daná väzba týka.



Obr. 3.3: Pohľad pretínajúcich tém.

3.4 Plánovanie návrhu

Na záver analýzy sa používa individuálny pohľad. Tento pohľad slúži ako príprava na návrh. Každý pohľad sa týka výlučne iba jednej témy. Pre túto tému sú zobrazené požiadavky, ktorých sa týka a kľúčové entity témy. V prípade aspektovej témy individuálny pohľad zahŕňa aj individuálne pohľady základných tém, ktoré aspektová téma ovplyvňuje (obr. 3.4). Témy, ktoré boli zoskupené v pohľade tém a vzťahov a v pohľade pretínajúcich tém, sú v tomto pohľade nezoskupené, no sú spojené väzbou agregácie s témou, pod ktorou boli zoskupené. Požiadavky odročených vzťahov ako aj ich väzby sú zobrazené prerušovanou čiarou. Nakoniec sa všetky uzly v individuálnom pohľade každej témy, ktoré sa nachádzajú vo viacerých individuálnych pohľadoch, označia nejakou inou farbou, čo má indikovať pravdepodobné zdieľanie konceptu.



Obr. 3.4: Individuálny pohľad témy potvrdiť.

S takto definovanými témami a entitami sa prechádza do fázy návrhu pomocou Theme/UML prístupu, čo je už nad rámec tejto práce.

4 Transformácia Theme/Doc modelu do modelu prípadov použitia

Na odhalenie podobností a rozdielov medzi Theme/Doc prístupom a prípadmi použitia v aspektovo-orientovanej analýze je najlepšie sa pokúsiť o ich vzájomnú transformáciu. Táto kapitola sa bude zaoberať transformáciou Theme/Doc pohľadov do prípadov použitia.

Proces transformácie bude pozostávať z nasledujúcich krokov:

1. Prevod tém na prípady použitia.
2. Určenie väzieb include, extend a generalization z pohľadov v Theme/Doc.
3. Doplnenie bodov rozšírenia a bodových prierezov rozšírenia do transformovaného modelu prípadov použitia.
4. Doplnenie tokov do transformovaného modelu prípadov použitia.
5. Úprava granulácie transformovaného modelu prípadov použitia.

Model prípadov použitia disponuje väčším počtom prvkov než model v notácii Theme/Doc, preto sa dá očakávať, že nie všetky kroky tejto transformácie sa budú dať automatizovať. Väčšina krokov v procese transformácie súvisí práve s hľadaním prvkov, ktoré sa v Theme/Doc modeli nenachádzajú, no sú nevyhnutné pre vytvorenie plnohodnotného modelu prípadov použitia tak, ako ho definuje Ivar Jacobson.
Ak sa tento prevod podarí, tak s využitím poznatkov z tohto prevodu by opačná transformácia mala byť jednoduchšia, kedže model v notácii Theme/Doc obsahuje menej prvkov než model prípadov použitia.

4.1 Analógia medzi prípadmi použitia a témami v Theme/Doc

Pri bežnom vzhliadnutí diagramov prípadov použitia a pohľadov v Theme/Doc toho istého systému, je možné spozorovať niektoré podobnosti medzi témami a prípadmi použitia a medzi niektorými väzbami.
Objasnenie týchto podobností si vyžaduje lepšie sa pozrieť na to, ako sa vlastne identifikujú témy a prípady použitia, t.j. čo vlastne znamenajú.

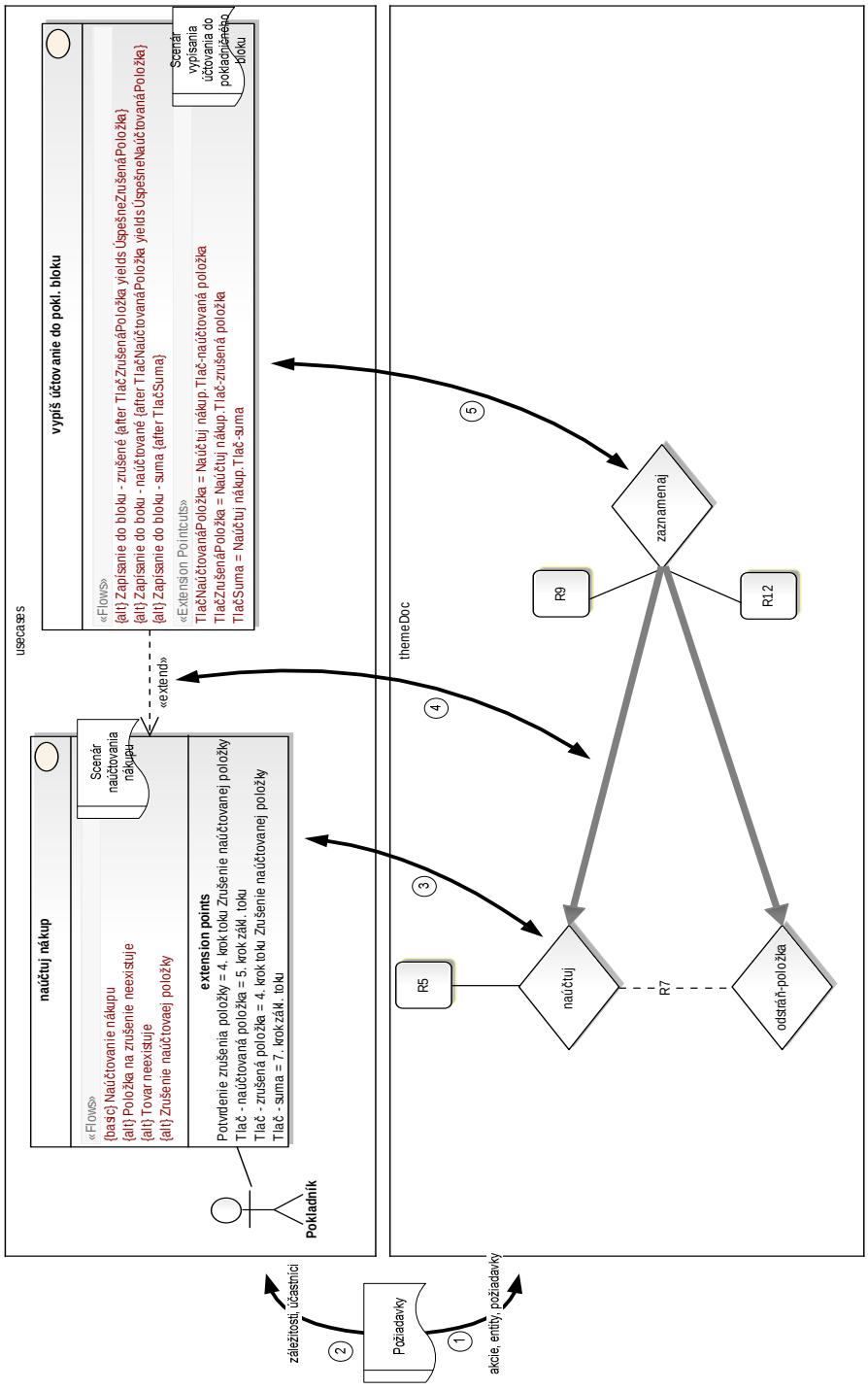
V prípade identifikácie prípadov použitia sa vychádza zo špecifikovaných požiadaviek alebo priamo z komunikácie so zainteresovanými osobami a hľadajú sa záležitosti, ktoré sú pre niektorého z účastníkov dôležité, najčastejšie ide o nejakú kľúčovú funkcionalitu (obr. 4.1, šípka č. 2).

Pri identifikácií tém sa môže postupovať tromi spôsobmi (Clarke, et al., 2005):

1. Identifikáciou akcií (slovies) v požiadavkách a postupným rozdeľovaním, spájaním, zoskupovaním, odstraňovaním akcií a požiadaviek sa získajú kľúčové témy (obr. 4.1, šípka č. 1).
2. Priamou identifikáciou kľúčových funkcionalít v požiadavkách, ktoré budú prehlásené za témy.

3. Z modelu prípadov použitia.

Z porovnania spôsobu identifikácie prípadov použitia a identifikácií tém je vidieť, že cieľom oboch prístupov je zachytiť kľúčové funkcionality, ktoré sú potom v modeli zobrazené ako prípady použitia, resp. témy. Z toho vyplýva, že prípad použitia predstavuje vlastne to isté ako téma, t.j. nejakú kľúčovú funkcionalitu. Dokonca aj samotný postup identifikácie tém dovoľuje vychádzať z diagramu prípadov použitia. V prípade, že sú prípady použitia správne identifikované, tak k ďalším úpravám by už nemalo dochádzať a môžu byť priamo prehlásené za témy v pohľade pretínajúcich tém, samozrejme, ak nám vyhovuje rovnaká granulácia v oboch prístupoch. Takto vzniká analógia medzi prípadmi použitia v diagrame prípadov použitia a témami v Theme/Doc, konkrétne v pohľade pretínajúcich tém (obr. 4.1, šípky č. 3 a č. 5) (Neskôr sa ukáže, že to platí iba pre základné a rozširujúce prípady použitia).



Obr. 4.1: Analógia medzi témami a prípadmi použitia.

Ak by sa takto poprepájali všetky témy a prípady použitia, tak by bola viditeľná istá podobnosť medzi aspect/based väzbami v pohľade pretínajúcich tém a väzbami extend v diagrame prípadov použitia (obr. 4.1, šípka č. 4). Tak ako sú spojené rozširujúce prípady použitia s existujúcimi prípadmi použitia väzbou extend, tak sú aj témy súvisiace s týmito prípadmi použitia spojené aspect/based väzbou. Je to dôsledok toho, že princíp určovania aspect/based väzieb je rovnaký ako princíp určovania extend väzieb. Postup ako rozhodnúť, ktorá téma je základná a ktorá pretínajúca, určujú štyri podmienky spomínané v predchádzajúcich kapitolách (kap. 3.3). Dôležité je, že sa tam pracuje s pravidlami, ktoré sú typické pre vzťahy medzi rozširujúcimi a rozširovanými prípadmi použitia (tab. 2).

Tab. 2: Použitie rovnakých pravidiel pre rozhodnutie o aspect/based vzťahoch v Theme/Doc a pre rozhodnutie o závislosti extend medzi prípadmi použitia.

Rozhodnutie o aspect/based vzťahu medzi témami	**Rozhodnutie o extend závislosti medzi prípadmi použitia**
Dominantná téma je tá, ktorá musí mať informácie o správaní požiadavky, aby vedela, kedy sa má aktivovať.	Rozširujúci prípad použitia musí poznať prípad použitia, ktorý rozširuje, aby vedel , kedy sa má spustiť, t.j. je na ňom závislý (šípka závislosti smeruje od rozširujúceho k rozširovanému prípadu použitia).
Pretínajúca téma je aktivovaná v rámci základnej tém.	Rozširujúci prípad použitia je aktivovaný v rámci správania rozširovaného prípadu použitia.
Základná téma sa správa normálne bez ohľadu na správanie pretínajúcej témy, ktorá ju pretína.	Rozširovaný prípad použitia je nezávislý na prípade použitia, ktorý ho rozširuje.

Z tabuľky (tab. 2) je vidieť, že pravidlá pri rozhodovaní o základnej a pretínajúcej téme sú rovnaké ako pravidlá rozhodovania o rozširovanom a rozširujúcom prípade použitia. Spolu s faktom, že identifikácia tém a identifikácia prípadov použitia je to isté, vyplýva analógia, ktorá hovorí, že základné a pretínajúce témy zodpovedajú rozšíreným a rozširujúcim prípadom použitia a naopak.

4.2 Určenie include väzieb

V predošlých kapitolách boli naznačené niektoré viac menej jasné analógie, no na transformáciu pohľadov na prípady použitia to nestačí. Neboli tam spomenuté najmä ďalšie väzby, ktoré sa vyskytujú v prípadoch použitia. Ide o väzby generalizácie a include.

Väzba include je praktická pomôcka, kedy je podtok nejakého toku, z dôvodu jeho znovupoužitia aj inými prípadmi použitia, vyčlenený do vlastného prípadu použitia. Rozhodnutie o väzbe include prebieha v čase vytvárania scenára, keď sa zistí, že je potrebné postupnosťou krokov opísať správanie nejakej minoritnej funkcionality vzhľadom na daný prípad použitia. Vtedy sa tieto kroky vyčlenia do vlastného podtoku. Ak sa k tomu pridajú aj ostatné prípady použitia, ktoré používajú rovnakú

postupnosť krokov, tak je vhodné tento podtok vyčleniť do samostatného prípadu použitia a prepojiť ho s ostatnými prípadmi použitia pomocou väzby include.

Keďže Theme/Doc nepracuje so scenármi, tak takýmto praktickým spôsobom sa k väzbe include nie je možné dostať. Preto ani samotný Theme/Doc väzbu include nepozná. Je ale pravdepodobné, že ak sa pri identifikácii tém postupuje zdola nahor (vypísaním slovies), tak v pohľade tém a vzťahov bude vystupovať akcia, ktorá bude reprezentovať daný podtok. No následným zjednocovaním a zoskupovaním tém tieto akcie splynú s nejakou témou. V literatúre pre prístup Theme sa pri opisovaní postupu ako zoskupovať témy nepriamo spomína aj postup, ktorý sa podobá určovaniu väzby include. Operácia zoskupenia je užitočná pre zoskupenie úzko súvisiaceho správania, ktoré by sme pri návrhu mohli brať ako jednu tému. Zoskupením taktiež môžeme vyjadriť rozhodnutie degradovať správanie vyjadrené vlastnou témou na správanie vyjadrené metódou (Clarke, et al., 2005).

V prípade hľadania väzby include je zaujímavá najmä veta o zoskupení na základe degradácie správania, ktorá hovorí, že ak máme nejakú tému, ktorej správanie by sme mohli opísať skôr metódou ako témou, tak je vhodné ju zoskupiť so súvisiacou témou, čím sa stane časťou (metódou) jej celkového správania. V súvislosti s týmto sa v literatúre nič viac nepíše, no dá sa predpokladať, že súvisiaca téma je tá, ktorej sa táto časť správania skutočne týka, t.j. na vyjadrenie svojho celkového správania potrebuje aj túto časť. Analogicky k prípadom použitia je možné si túto časť predstaviť ako podtok prípadu použitia, na ktorý sa odvoláva hlavný tok daného prípadu použitia. V literatúre sa taktiež nič nepíše o tom, že čo sa stane, ak takáto téma má viac súvisiacich tém. Nevie sa, či sa všetky tieto témy zoskupia do jednej, alebo sa takáto téma najprv „znásobí“ tak, aby pre každú súvisiacu tému existovala takáto jedna podtéma, s ktorou sa zoskupí. Je ťažké si predstaviť, že by sa zoskupili všetky témy do jednej témy, ak jediné čo ich spája je použitie rovnakej podtémy. Prijateľnejšia je teda druhá možnosť. Dôležité je ale to, že ak by bolo viac súvisiacich tém s jednou podtémou, tak by to v prípade prípadov použitia znamenalo, že máme podtok, ktorý je znovupoužiteľný aj pre iné prípady použitia, čiže by bol tento podtok vyčlenený ako samostatný prípad použitia, ktorý by bol so súvisiacimi prípadmi použitia prepojený väzbou include.

Nájsť všetky súvisiace témy na takéto zoskupenie nie je jednoduché. Pri zoskupovaní sa väčšinou porovnávajú všetky témy navzájom a v spojení s požiadavkami sa zisťuje, či tieto témy sú vhodné na zoskupenie alebo nie. V prípade podtokov by sme očakávali, že tieto podtoky budú nejakým spôsobom spomenuté v požiadavkách, s ktorými je téma prepojená. Sú určité pravidlá, ktoré hovoria, že čo musí požiadavka obsahovať, aby bola správne zaznamená (kto, čo, kedy, ...), no nie vždy sa na to dá spoliehať. Ak by to tak bolo, tak by bola veľká pravdepodobnosť, že súvisiace témy k našej podtéme budú niektoré z tém, s ktorými je prostredníctvom požiadaviek prepojená v pohľade tém a vzťahov. Napríklad pri zoskupení témy *identifikuj-položku* by sa zistilo, že okrem témy *naúčtuj*, s ktorou je prepojená, môže ako podtéma vystupovať aj v rámci tém *pridaj*, *zmeň-množstvo*, *odstráň-tovar*, *uprav-cenu*, s ktorými nie je nijako prepojená.

Ak by sme sa pozreli na požiadavky, s ktorými sú tieto štyri témy prepojené, tak by sme zistili, že na rozdiel od témy *naúčtuj* nedisponujú dané témy dostatočne

presnou formuláciou požiadavky, preto v pohľade tém a vzťahov v Theme/Doc nie sú prepojené s témou *identifikuj-položku*, aj keď je jasné, že skôr ako sa bude odstraňovať, pridávať, meniť množstvo tovaru alebo jeho cena, tak sa musí najprv tento tovar identifikovať v databáze. Preto je vhodné tieto požiadavky spísať lepším spôsobom (text zátvorkách označuje témy, ktorých sa daná požiadavka týka).

Pôvodné požiadavky (aj so zmenami, ktoré boli vykonané v kap. 3.3):

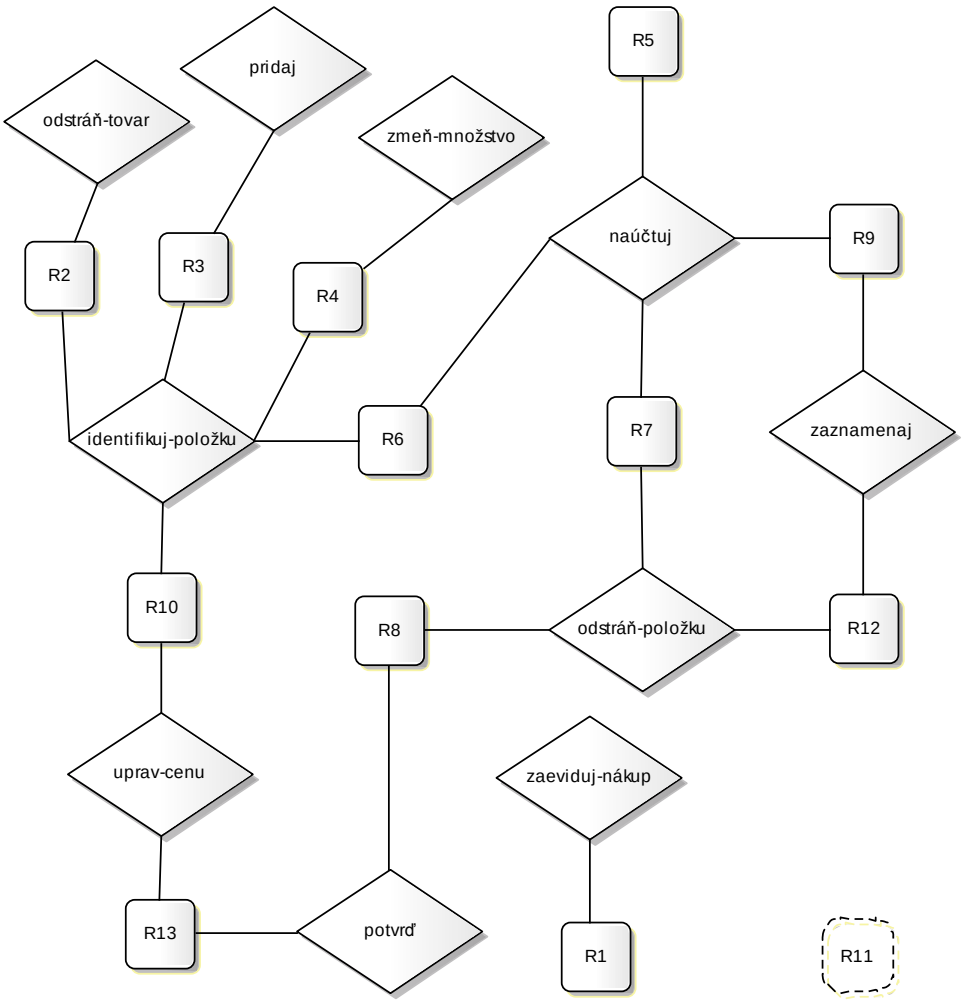
1. Systém bude evidovať stav tovaru v centrálnej databáze dynamicky podľa práve predaných položiek.
2. Skladník bude môcť zo systému odstraňovať tovar. (odstráň-tovar)
3. Skladník bude môcť do systému pridávať nový tovar. (pridaj)
4. Skladník bude môcť meniť v systéme množstvo položiek tovaru. (zmeň-množstvo)
5. Pokladník bude môcť položku nákupu naúčtovať ručným zadaním čiarového kódu alebo pomocou čítačky čiarových kódov.
6. Naúčtovať sa môžu iba položky, ktoré sú evidované v databáze. (naúčtuj, identifikuj-položky)
7. Počas účtovania nákupu bude možné odstrániť položku z naúčtovaných položiek.
8. Odstránenie položky musí potvrdiť vedúci predaja svojou autentifikáciou.
9. Naúčtované položky sa budú priebežne zaznamenávať na pokladničný blok. Na pokladničnom bloku sa budú nachádzať naúčtované položky a celková suma nákupu.
10. Upraviť cenu tovaru v centrálnej databáze môže iba vedúci predaja, ktorý svoju identitu potvrdí autentifikáciou. (uprav-cenu, potvrď)
11. Čas od zadania čiarového kódu po zobrazenie ceny identifikovanej položky bude trvať maximálne pol sekundy.
12. Zrušené položky sa budú priebežne zaznamenávať na pokladničný blok.

Lepšie spísané požiadavky:

1. ...
2. Skladník bude môcť zo systému odstraňovať tovar, ktorý je evidovaný v databáze. (odstráň-tovar, identifikuj-položky)
3. Skladník bude môcť do systému pridávať nový tovar, ktorý nie je evidovaný v databáze. (pridaj, identifikuj-položky)
4. Skladník bude môcť meniť v systéme množstvo položiek tovaru, ktorý je evidovaný v databáze. (zmeň-množstvo, identifikuj-položky)
5. ...
6. Naúčtovať sa môžu iba položky, ktoré sú evidované v databáze. (naúčtuj, identifikuj-položky)
7. ...
8. ...
9. ...

10. Vedúci predaja bude môcť upraviť cenu tovaru, ktorý je evidovaný v centrálnej databáze. (uprav-cenu, identifikuj-položky)
11. ...
12. ...
13. Vedúci predaja pri úprave ceny tovaru musí svoju identitu potvrdiť autentifikáciou. (uprav-cenu, potvrď)

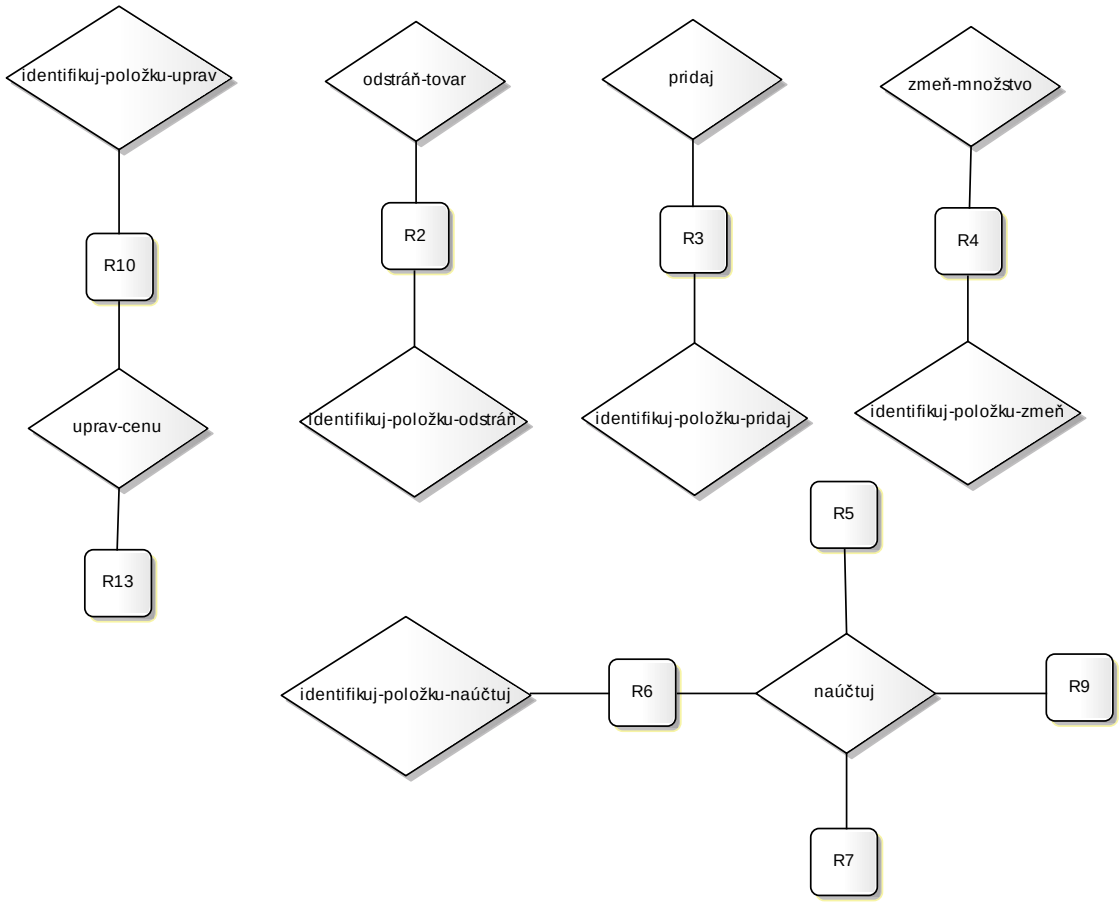
Po tejto zmene by už mali byť jasne viditeľné prepojenia medzi témou *identifikuj-tovar* a súvisiacimi témami v súvislosti so zoskupovaním tém podľa degradácie správania témy (obr. 4.2).



Obr. 4.2: Pohľad tém a vzťahov po zlepšenom spísaní požiadaviek.

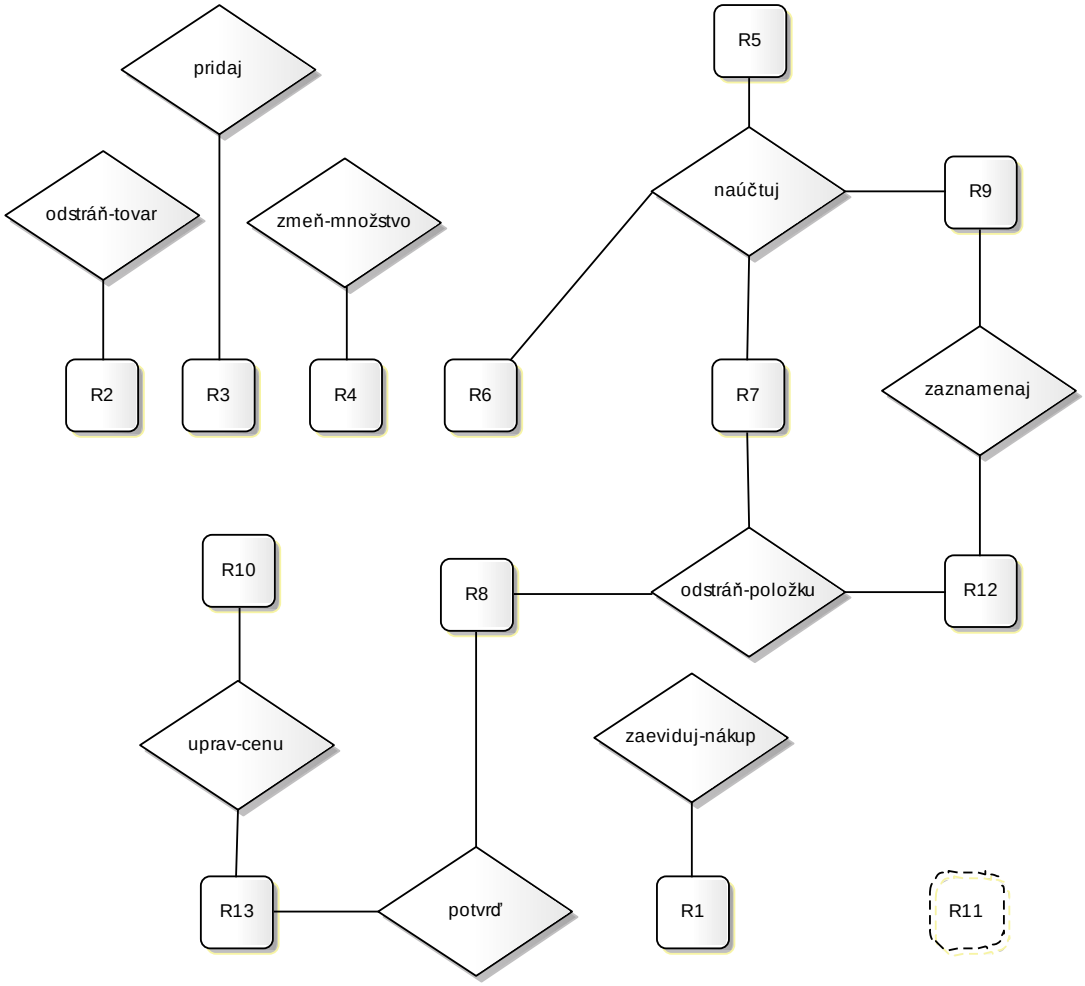
Keďže sa nie je možné spoliehať na to, že znovupoužitie degradovanej témy na podtému súvisí iba s témami, ktoré sú navzájom prepojené prostredníctvom požiadavky, tak je lepšie pri takomto zoskupovaní skontrolovať všetky témy navzájom.

Teraz by mala nasledovať degradácia tému *identifikuj-položku* na podtému a zoskupenie s nejakou inou témou. Problémom ale je, že ak by mala byť podtéma znovupoužitaľná inými témami, tak by mala vystupovať ako podtéma vo viacerých témach. V Theme/Doc neexistujú prostriedky ako takéto znovupoužitie vyjadriť. Nie je možné, aby jedná podtéma s rovnakým názvom bola zoskupená pod viacerými témami súčasne. Preto sa bude musieť téma *identifikuj-položku* „umelo“ rozdeliť podľa toho, s akou témou má spolupracovať (obr. 4.3). Znamená to, že požiadavky, v ktorej sa spomína téma *identifikuj-položku* (obr. 4.3, požiadavky *R2*, *R3*, *R4*), budú disponovať samostatnou témou opisujúcej identifikovanie položky.



Obr. 4.3: Rozdelenie témy identifikovať-položku.

Následne sa tieto nové témy môžu zoskupiť s prislúchajúcimi témami pod zámienkou ich degradácie (obr. 4.4).

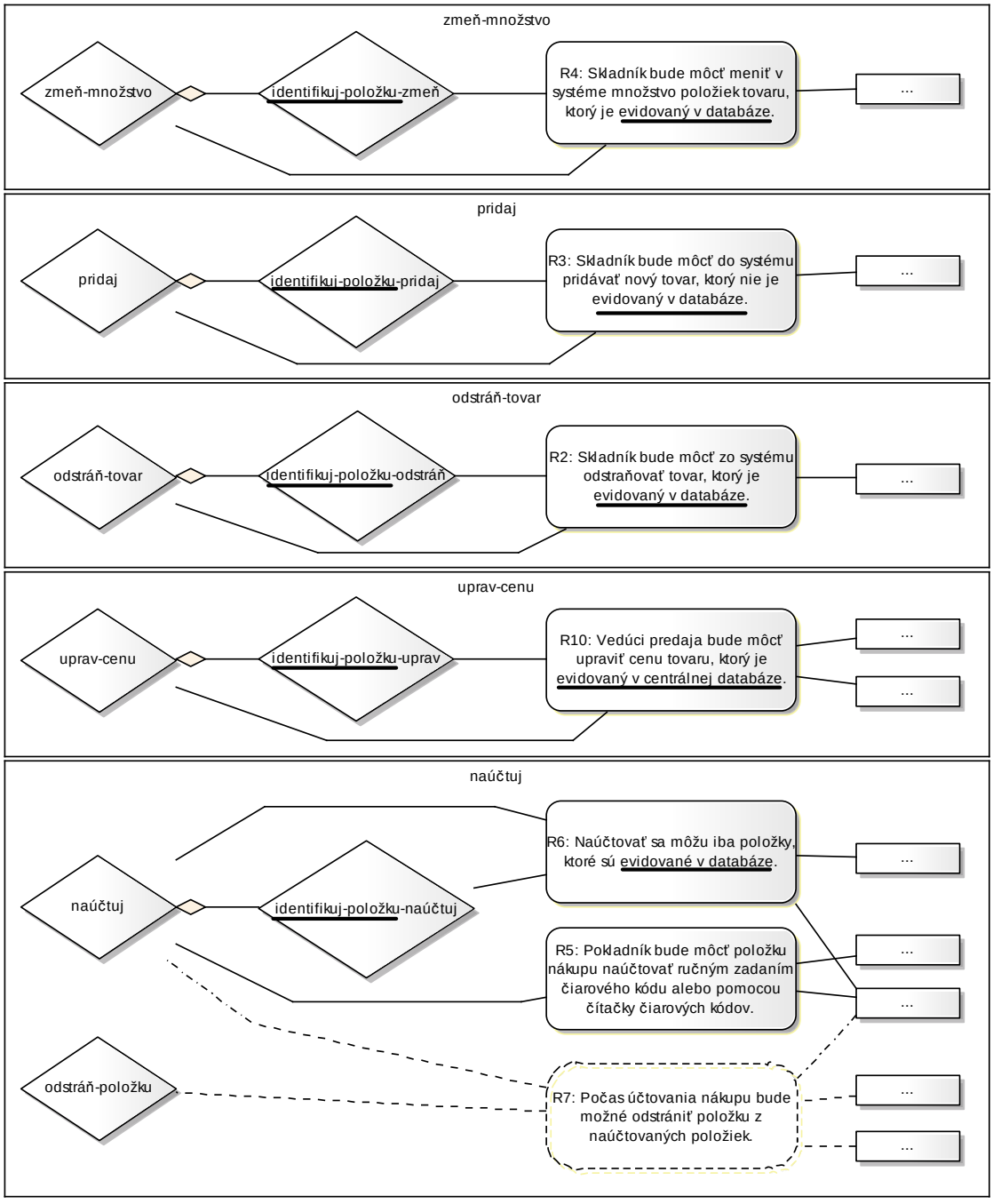


Obr. 4.4: Témy po zoskupení.

Takto by mal vyzerať finálny stav pohľadu tém a vzťahov. Môžeme si všimnúť, že tento pohľad nezobrazuje žiadne informácie o tom, že sa niečo zoskupovalo. Podobne aj pohľad pretínajúcich tém o tom nič nehovorí. Zoskupené témy sú zobrazené až v individuálnom pohľade témy prostredníctvom väzby agregácie, ktorá spája zoskupené témy a pred návrhom je ešte možné sa rozhodnúť, či témy ostanú zoskupené, alebo sa znovu rozdelia. Pokiaľ nie sú k dispozícii nejaké záznamy o tom, ako vznikal finálny pohľad tém a vzťahov, tak jediné miesto, kde je možné identifikovať, že pri vytváraní pohľadu tém a vzťahov došlo k zoskupeniam, je

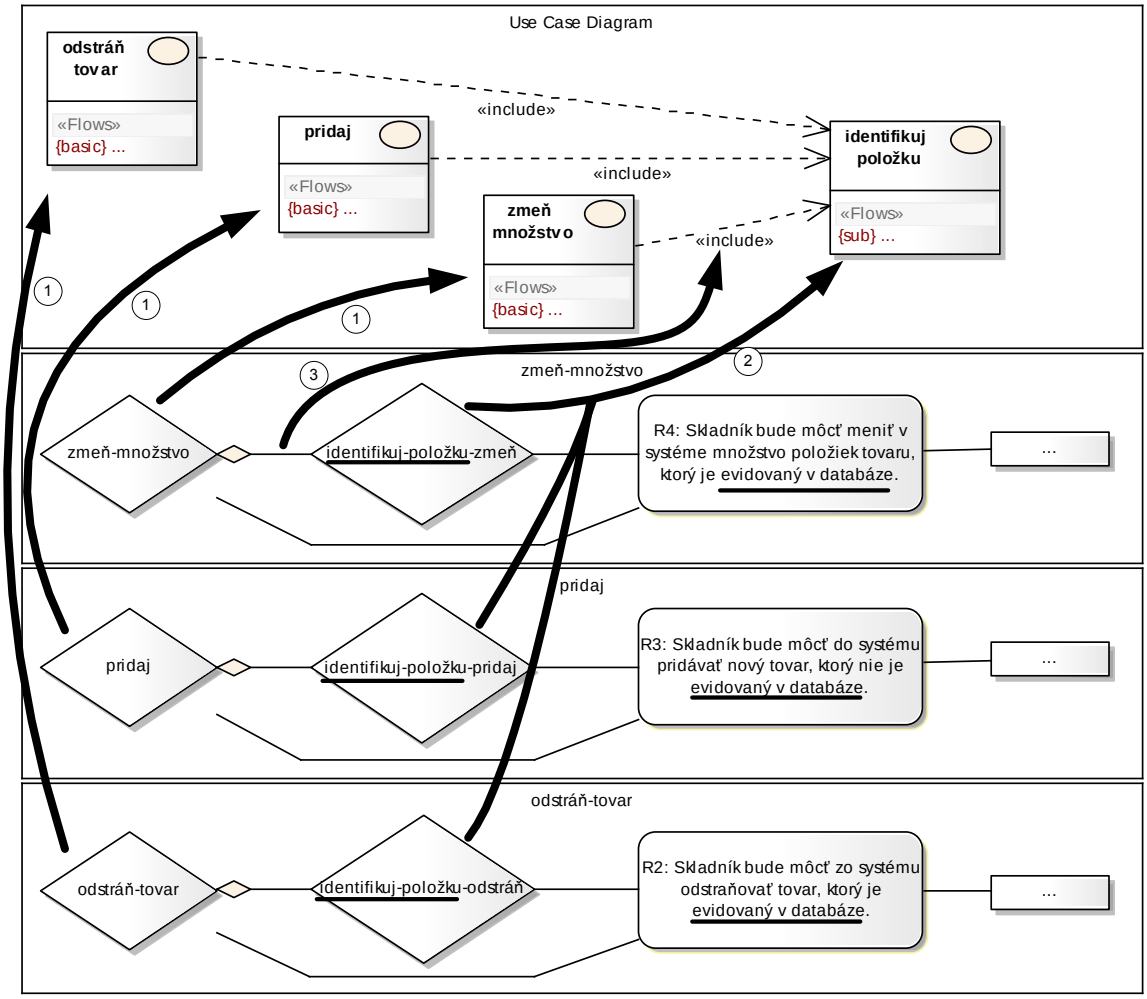
individuálny pohľad tém. Preto väzbu include je potrebné identifikovať práve podľa individuálneho pohľadu tém.

Pre tento príklad sú najdôležitejšie individuálne pohľady tém *naúčtuj*, *uprav-cenu*, *odstráň-tovar*, *pridaj* a *zmeň-množstvo* (obr. 4.5). Entity na obrázku sú zobrazené len symbolicky, pretože nie momentálne dôležité. Dôležité je sledovať, čo v skutočnosti daná podtéma robí. Keďže Theme/Doc nedisponuje aparátom, ktorý by vyjadril znovupoužiteľnosť podtémy, tak podtéma bola najprv „znásobená“ a výsledné podtémy boli odlíšené zmenou ich názvov. Preto sa podľa názvu, ktorý už nie je rovnaký, nedá zistiť, že tieto podtémy špecifikujú to isté správanie.



Obr. 4.5: Individuálny pohľad tém zmeniť-množstvo, pridať, odstrániť-tovar, upraviť-cenu a naúčtovať so zvýrazneným správaním jednotlivých podtém.

No v individuálnom pohľade tém je možné rovnaké správanie podtém, resp. tém, objaviť na základe pripojených úplných požiadavkách. Na obrázku (obr. 4.5) sú individuálne pohľady tém, ktorých sa zoskupenie týkalo. V požiadavkách je vyznačená pôsobnosť každej podtémy. Ak si vyznačené miesta navzájom porovnajú, tak sa zistí, že dané podtémy, aj keď majú rôzne názvy, opisujú rovnaké správanie, preto by teoreticky postačila iba jedna takáto podtéma, na ktorú sa budú ostatné témy odvolávať. V individuálnom pohľade, ako už bolo spomenuté, takéto zakreslenie nie je možné, no nebráni to tomu, aby sa pri transformácii Theme/Doc prístupu na prípady použitia na to neprihliadalo. Ak sa počas transformácie pri analýze pohľadu individuálnych odhalia takéto vlastnosti podtém, tak takéto podtémy môžu byť zakreslené ako jeden samostatný inclusion prípad použitia (obr. 4.6, šípka č. 2) a témy, pod ktorými tieto podtémy boli zoskupené, budú vystupovať ako prípady použitia (obr. 4.6, šípky č. 1), ktoré sa na tento prípad použitia budú odvolávať prostredníctvom väzby include (obr. 4.6, šípka č. 3).



Obr. 4.6: Analógia medzi zoskupovaním tém a väzbou include medzi prípadmi použitia.

Čo sa týka zoskupovania podľa degradácie a väzby include, tak sa často stretneme so zoskupovením, kde podtémy síce vystupujú ako degradované témy, ale nie sú znovu použité v žiadnych iných témach. Takáto podtéma sa potom pri konvertovaní pohľadov do diagramov prípadov použitia nevyčlení do samostatného inclusion prípadu použitia, ale v prípade použitia, ktorý je zobrazením hlavnej témy danej podtémy, sa vytvorí nový podtok, ktorý bude predstavovať túto podtému.

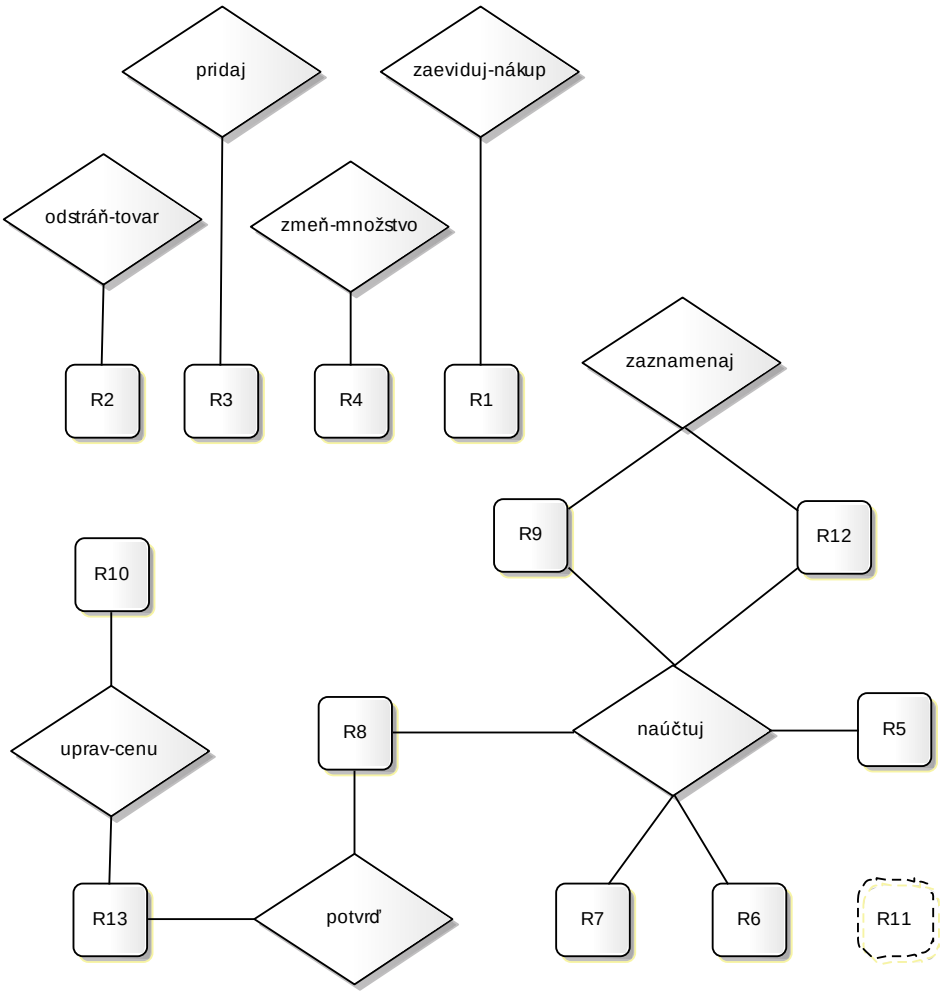
4.3 Určenie väzieb generalizácie

V prípade hľadania väzby include je zaujímavá najmä veta o zoskupení na základe degradácie správania, ktorá hovorí, že ak máme nejakú tému, ktorej správanie by sme

mohli opísať skôr metódou ako témou, tak je vhodné ju zoskupiť so súvisiacou témou, čím sa stane časťou (metódou) jej celkového správania. Operácia zoskupenia je užitočná pre zoskupenie úzko súvisiaceho správania, ktoré by sme pri návrhu mohli brať ako jednu tému.

Väzbu generalizácie, je možné v Theme/Doc identifikovať taktiež v zoskupených témach. Konkrétne sa jedná o vetu, ktorá hovorí o zoskupovaní na základe úzko súvisiaceho správania (*Operácia zoskupenia je užitočná pre zoskupenie úzko súvisiaceho správania, ktoré by sme pri návrhu mohli brať ako jednu tému* (Clarke, et al., 2005)). Pod úzko súvisiacim správaním sa chápe to, že niektoré témy majú príbuzné správanie alebo koncept. Z bližšieho opisu takéhoto zoskupovania v oficiálnej literatúre (Clarke, et al., 2005) vidno, že takéto zoskupovanie ma rovnaké vlastnosti ako generalizácia v diagrame prípadov použitia.

V našom príklade je to prípad tém *naúčtuj* a *odstráň-položku*, ktoré úzko súvisia, keďže obe témy predstavujú naúčtovanie položky len s tým rozdielom, že pri odstraňovaní naúčtovanej položky sa z výslednej ceny odpočíta suma (čo potvrdí vedúci predaja, ale to nie je teraz podstatné), zatiaľ čo pri naúčtovaní položky sa pripočíta. Tieto témy môžu zdieľať podstatné časti štruktúry a správania, preto budú zoskupené pod jednu tému *naúčtuj*, pričom kvôli rozlíšeniu tejto základnej témy, ktorú budú spomínané dve témy konkretizovať, sa pôvodný názov konkrétnej témy *naúčtuj* zmení na *naúčtuj-položku* (obr. 4.7).

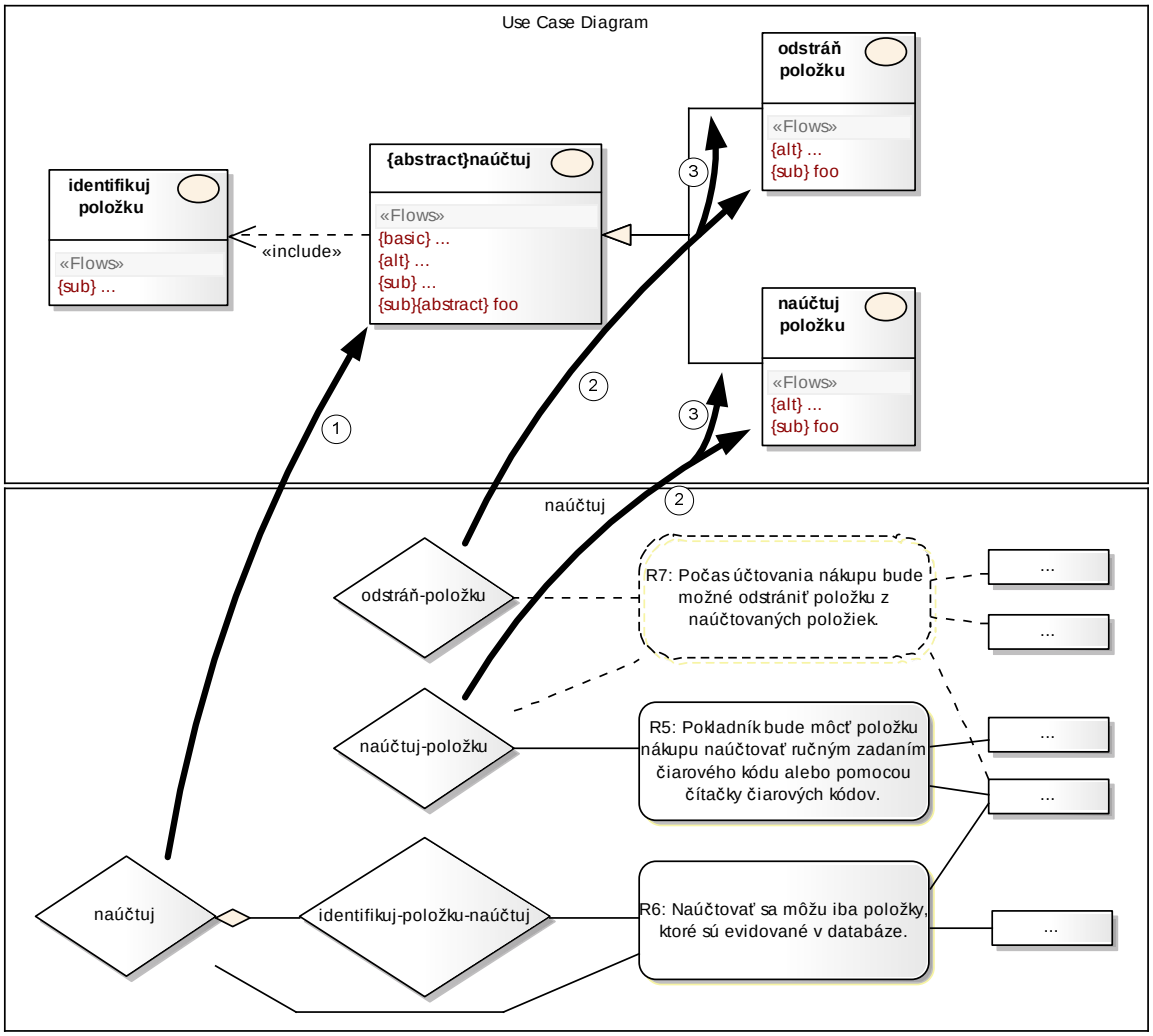


Obr. 4.7: Pohľad tém a vzťahov po zoskupení tém naúčtovať-položku a odstrániť-položku pod tému naúčtovať.

Podobne ako v prípade predošlého zoskupovania, tak aj teraz informáciu o tom, že došlo k zoskupeniu, sa je možné z diagramov dozvedieť až pri individuálnom pohľade témy *naúčtuj*. Preto pri konvertovaní Theme/Doc modelu na prípady použitia bude väzba generalizácie identifikovaná práve pomocou individuálneho pohľadu tém. Zoskupovanie v individuálnom pohľade tém je znázornené pomocou väzby agregácie, no v prípade zoskupovania podľa úzkej súvislosti sa v oficiálnej literatúre (Clarke, et al., 2005) objavujú prípady, kde takéto témy nie sú prepojené väzbou agregácie, presnejšie nie sú vôbec prepojené. Preto aj individuálny pohľad témy *naúčtuj* je takto zobrazený (obr. 4.8). Prepojenie pomocou väzby agregácie by nebolo najšťastnejšie

riešenie, pretože toto zoskupenie je robené na základe podobnosti a nie na základe degradácie témy na podtému, ktorú hlavná téma obsahuje.

Keby aj napriek tomu boli takto zoskupené témy v individuálnom pohľade prepojené väzbami agregácie, tak to, či zoskupenie bolo robené na základe úzkej súvislosti alebo na základe degradácie témy na podtému, sa dá zistiť jedine buď z názvov tém alebo je potrebné sa pozrieť aj na samotné požiadavky, s ktorými tieto témy súvisia.



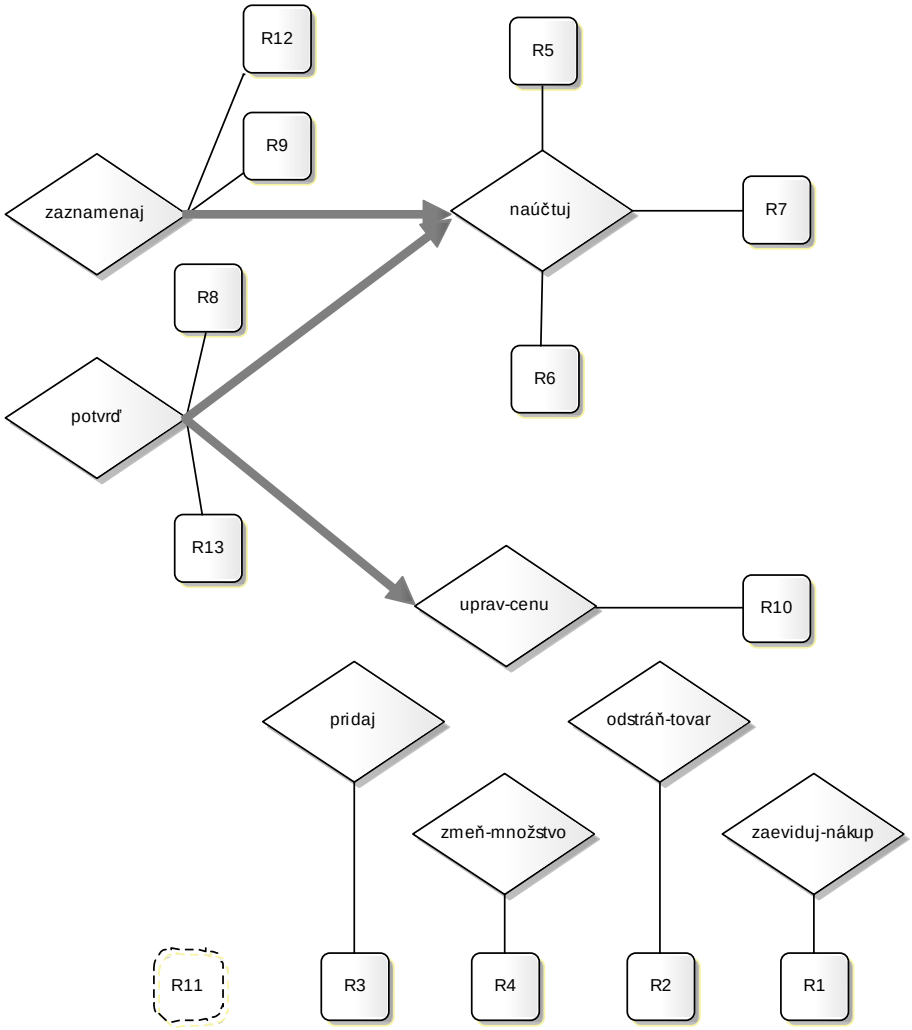
Obr. 4.8: Analógia medzi zoskupovaním tém a väzbou generalizácie medzi prípadmi použitia.

V našom príklade sa individuálny pohľad témy *naúčtuj* zobrazí do diagramu prípadov použitia tak, že hlavná téma *naúčtuj*, pod ktorou vystupujú témy *odstráň-položku* a *naúčtuj-položku*, sa zobrazí do abstraktného prípadu použitia *naúčtuj* (obr. 4.8, šípka č. 1) a témy *odstráň-položku* a *naúčtuj-položku* sa zobrazia do prípadov použitia *odstráň položku* a *naúčtuj položku* (obr. 4.8, šípky č. 2), pričom sú prostredníctvom väzieb generalizácie prepojené s prípadom použitia *naúčtuj*(obr. 4.8, šípky č. 3).

Pre tento konkrétny príklad je vhodné tému *naúčtuj* transformovať na abstraktný prípad použitia, no nie je to pravidlom. Všeobecnejší prípad použitia, ktorý vznikne generalizáciou konkrétnejších prípadov použitia, nemusí obsahovať žiadny abstraktný tok, teda nemusí byť abstraktný. V prístupe Theme/Doc sa nikde neuvádza, či sa jedná o abstraktnú tému, ktorá zoskupuje podtémy, alebo nie. Preto pri jej transformácii na prípad použitia sa rozhodnutie o tom, či sa vytvorí abstraktný prípad použitia alebo nie, necháva na skúsenostiach analytika, ktorý sa rozhodne na základe vlastností záležitostí, ktoré tieto zoskupené téme zachytávajú.

4.4 Určenie väzieb extend

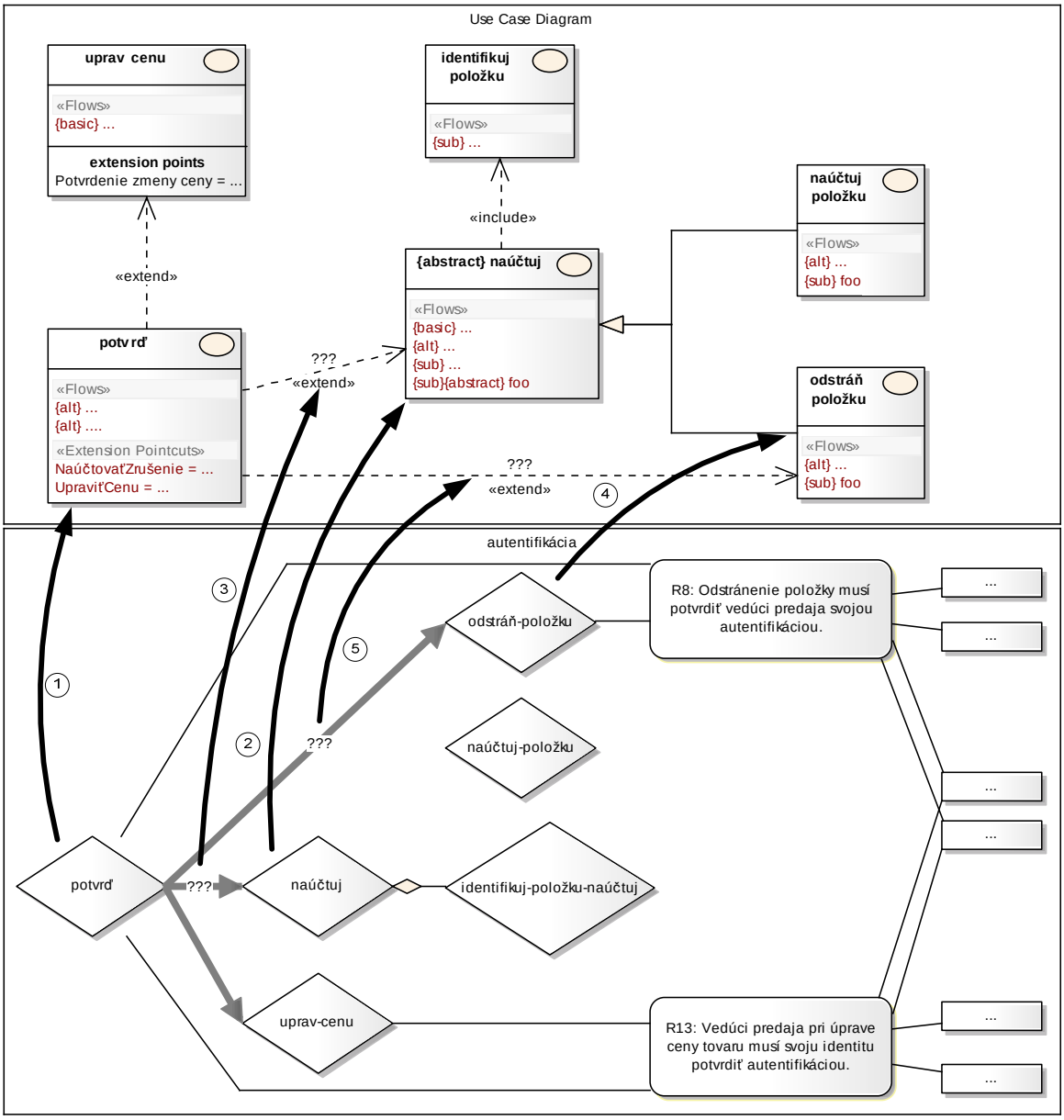
Pre určenie väzieb extend platia pravidlá, ktoré už boli spomínané v súvislosti s analógiou základných a pretínajúcich tém a rozširovaných a rozširujúcich prípadoch použitia. No po zoskupeniach, ktoré boli vykonané na našom príklade, nastávajú určité problémy. Na obrázku je zobrazené pohľad pretínajúcich tém po doterajších zmenách (obr. 4.9).



Obr. 4.9: Pohľad pretínajúcich tém po zoskupovaní.

Ako je možné vidieť, tak po zoskupení do jednej témy *naúčtuj*, už aspect/based väzba z témy *potvrď* smeruje k zjednotenej téme *naúčtuj* a nie k téme *odstráň-položku*. Problém je, ako to zakresliť do individuálneho pohľadu témy *potvrď* (obr. 4.10). Podľa pohľadu pretínajúcich tém by aspect/based väzba v pohľade individuálnych tém mala viesť od témy *potvrď* k téme *naúčtuj*. To by ale znamenalo, že bod rozšírenia pri potvrdení sa bude nachádzať v prípade použitia *naúčtuj*, čo by zase znamenalo, že tento bod rozšírenia by dedili oba prípady použitia, *naúčtuj položku* aj *odstráň položku* (obr. 4.10, šípka č. 3). Ale z požiadaviek systému je zrejmé, že potvrdiť sa má iba odstraňovanie položky, preto by mal byť bod rozšírenia špecifikovaný iba pre prípad použitia *odstráň položku*. To znamená, že aspect/based väzba by mala v

individuálnom pohľade témy *potvrď* viesť od témy *potvrď* k téme *odstráň-položku* a následne väzba extend v diagrame prípadov použitia by mala viesť od prípadu použitia *potvrď* k prípadu použitia *odstráň položku* (obr. 4.10, šípka č. 5). Ak by bola väzba extend robená podľa pohľadu pretínajúcich tém, ktorý bol vytváraný podľa pravidiel Theme/Doc (aspect/based väzba smeruje len k téme *naúčtuj*), tak by tento fakt nebol viditeľný. Preto je vhodné situáciu skontrolovať aj v individuálnom pohľade témy, či pôvodcom aspect/based väzby nie je náhodou nejaká zoskupená téma, čo samozrejme je možné zistiť z požiadaviek pripojených k témam v tomto pohľade (v našom prípade je to požiadavka *R8*).



Obr. 4.10: Určenie väzby extend pri zoskupených témach.

4.5 Odročené väzby v Theme/Doc

V modeli Theme/Doc sa môžu vyskytnúť aj odročené väzby medzi témami. Sú to väzby, ktoré nebolo možné bližšie špecifikovať a preto sa o nich rozhodne až v návrhu,

keď bude k dispozícii lepšia predstava o ich správaní. Väčšinou sa jedná o aspect/based väzby, pri ktorých nebolo možné sa rozhodnúť o ich smerovaní.

V modeli prípadov použitia je zvykom bližšie nešpecifikované závislosti označovať ako obyčajnú asociáciu, no pri aspektovo-orientovanom vývoji softvéru pomocou prípadov použitia je zakázané používať takéto nešpecifikované závislosti. Už vo fáze analýzy je potrebné sa rozhodnúť, či sa jedná o väzbu extend, include alebo generalizáciu.

V momente transformácie je teda možné odročené väzby v Theme/Doc dočasne previesť na obyčajnú asociáciu, no pri finalizácii modelu prípadov použitia treba presne špecifikovať, či sa jedná o väzbu icnlude, extend alebo generalizáciu.

4.6 Body rozšírenia a bodové prierezy rozšírenia

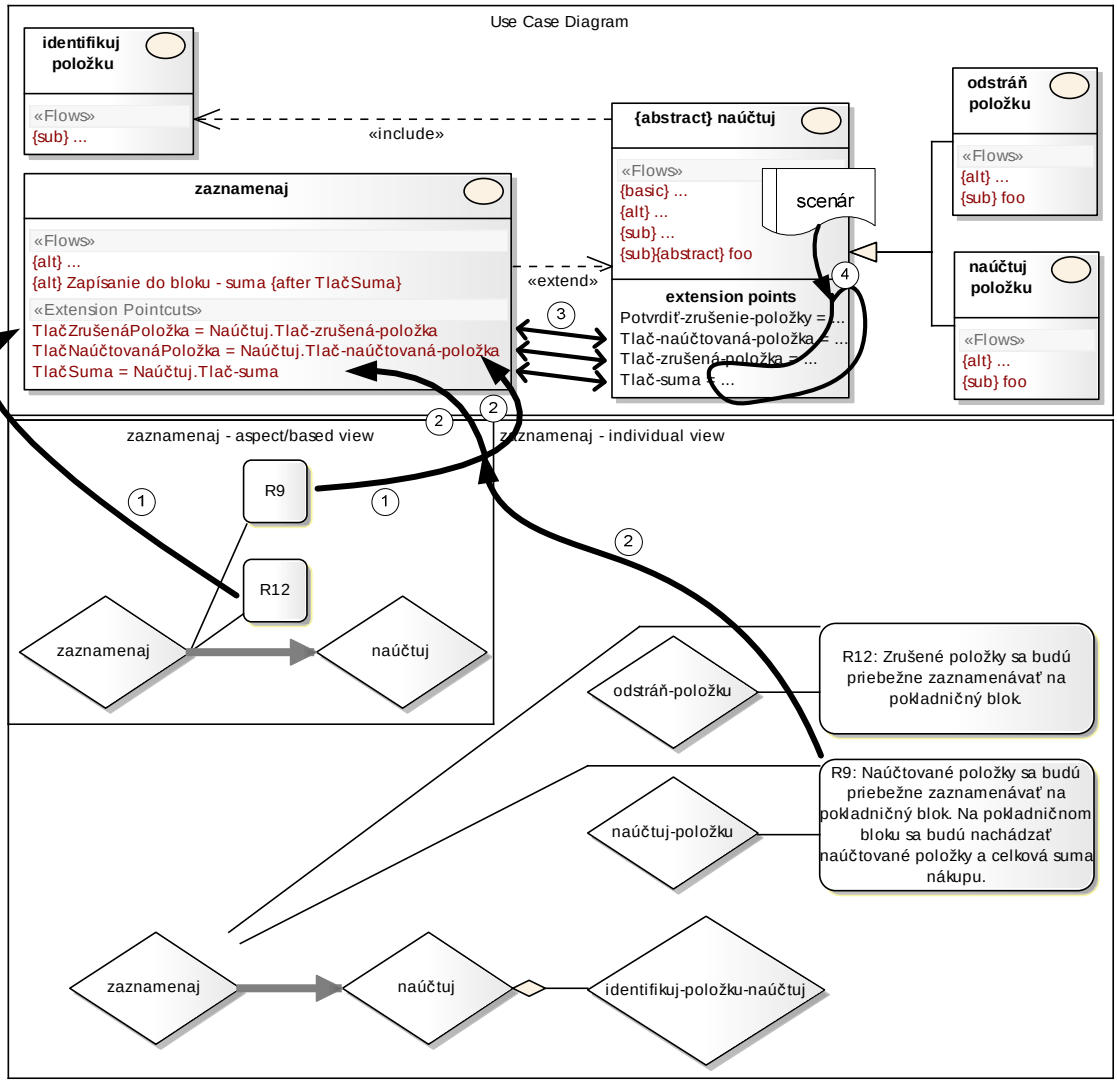
Kvôli neprítomnosti scenára v Theme/Doc, nie je možné presné určenie bodov rozšírenia a bodových prierezov rozšírenia. No vzhľadom na existenciu aspect/based väzieb v pohľade pretínajúcich tém a v individuálnom pohľade tém je možné sa pokúsiť zistiť aspoň ich počet a priradiť im názov.

V pohľade pretínajúcich tém je možné vidieť, že pri každej aspect/based väzbe sú na strane aspektovej témy zobrazené identifikátory požiadaviek, ktoré sú zodpovedné za existenciu tejto väzby (obr. 4.9, požiadavky *R9* a *R12* pri téme *zaznamenaj* a *R8*, *R13* pri téme *potvrď*). Takže je možné sa domnievať, že minimálny počet bodových prierezov rozšírenia v prípade použitia bude rovný počtu týchto identifikátorov požiadaviek pripojených k téme, ktorá je obrazom daného prípadu použitia (obr. 4.11, šípky č. 1). Pre zistenie presného počtu si už treba pomôcť textom požiadavky, ktorá sa nachádza v individuálnom pohľade témy. Napríklad aspect/based väzba medzi témou *zaznamenaj* a *naúčtuj* je výsledkom požiadaviek *R9* („*Naúčtované položky sa budú priebežne zaznamenávať na pokladničný blok. Na pokladničnom bloku sa budú nachádzať naúčtované položky a celková suma nákupu.*“) a *R12* („*Zrušené položky sa budú priebežne zaznamenávať na pokladničný blok.*“).

Analýzou požiadavky *R9* sa zistí, že aspect téma *zaznamenaj* pretína požiadavku v dvoch rôznych miestach, pri účtovaní položky a naúčtovaní celkovej sumy, preto by mali byť aj dva bodové prierezy pre túto tému. Spolu s požiadavkou *R12* to znamená, že prípad použitia, ktorý je obrazom témy *zaznamenaj*, bude obsahovať tri bodové prierezy rozšírenia (obr. 4.11, šípky č. 2). Tu ale opäť treba dávať pozor na to, že hoci v pohľade pretínajúcich tém sa aspect/based väzba spája so zoskupenou témou *naúčtuj*, tak v individuálnom pohľade môže byť táto väzba spojená s podtémou tejto zoskupenej témy (napr. *odstráň-položku*) tak, ako je to na obrázku (obr. 4.10) a preto musí byť bodový prierez rozšírenia vložený vždy do takého prípadu použitia, ktorý je obrazom témy podľa individuálneho pohľadu témy. Na určenie názvov týchto bodov spájania je možné sa riadiť pripojenými požiadavkami, pričom ich presné znenie záleží len na subjektívnych pocitoch analytika.

Čo sa týka bodov rozšírenia, tak najjednoduchší spôsob je automaticky pre každý jeden bodový prierez rozšírenia vytvoriť jeden bod rozšírenia v prípade použitia, ktorý je na druhej strane väzby extend (rozšírovaný prípad použitia) a v bodovom priereze rozšírenia sa odkazovať na tento bod rozšírenia (obr. 4.11, šípky č. 3). Prípadné

zjednotenie týchto bodov rozšírenia v dôsledku znovupoužitia je už v pôsobnosti samotného prístupu prípadov použitia, keďže body rozšírenia sa odkazujú na konkrétne miesta v scenári (ktorý sa v prístupe Theme/Doc nevyskytuje) a preto tieto odkazy je možné určiť až po vytvorení scenára prípadu použitia (obr. 4.11, šípka č. 4).

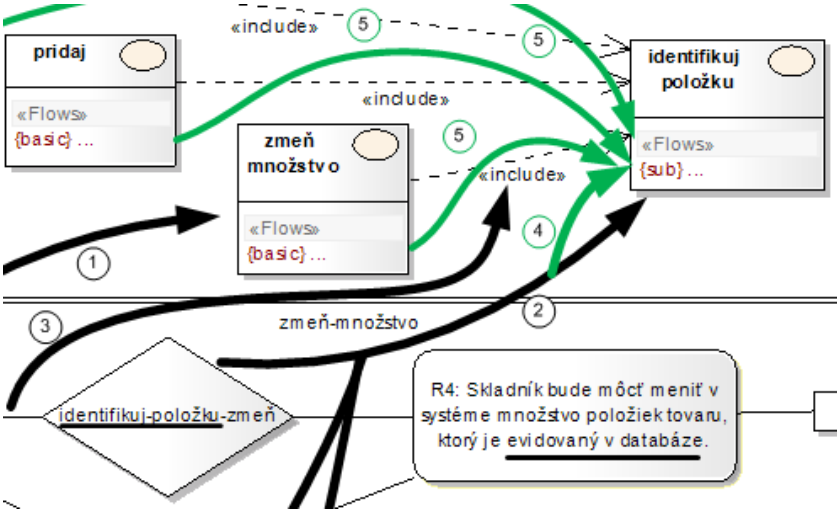


Obr. 4.11: Určenie extension pointcuts a bodov rozšírenia prípadov použitia z pohľadov v Theme/Doc.

4.7 Toky

Toky, ktoré sú zobrazené v prípade použitia, sú priamym zobrazením tokov v scenári daného prípadu použitia. Bez neprítomnosti scenára v Theme/Doc je preto veľmi problematické hľadať toky v Theme/Doc pohľadoch.

Isté je, že podtémy, ktoré boli zoskupené na základe ich degradácie, budú mať svoje vlastné podtoky v prípade použitia, ktorý je zobrazením témy, pod ktorou tieto podtémy v Theme/Doc vystupujú. V prípade, že tieto podtémy boli z dôvodu znovupoužitia vyčlenené do samostatného inclusion prípadu použitia, tak pre tieto podtémy bude existovať iba jeden spoločný podtok vo vyčlenenom prípade použitia (obr. 4.12, šípka č. 4), na ktorý sa budú odvolávať ostatné prípady použitia, ktoré sú zobrazením tém, pod ktorými boli tieto podtémy zoskupené (obr. 4.12, šípky č. 5).

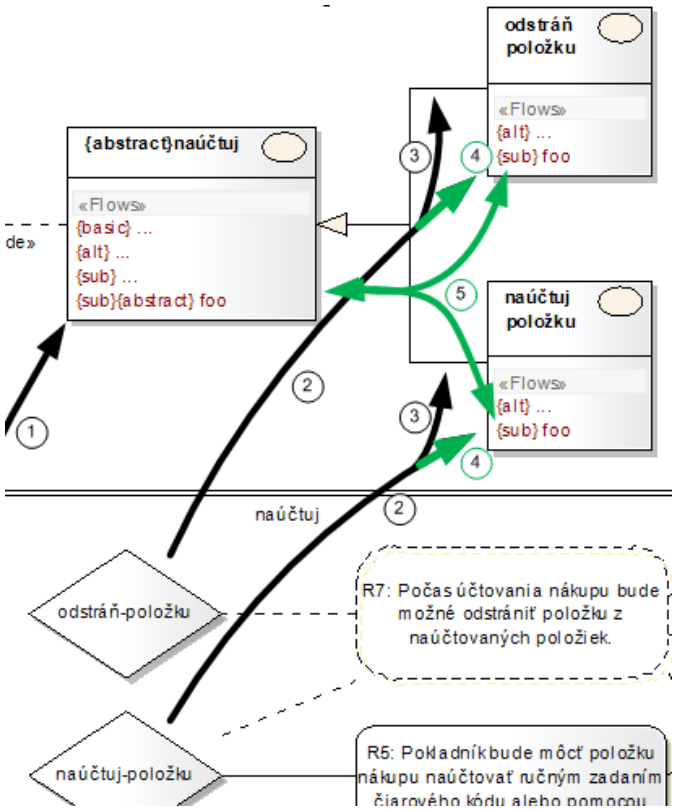


Obr. 4.12: Vytvorenie podtoku vo vyčlenenom prípade použitia, ktorý vznikol prekonvertovaním zoskupených znovupoužiteľných tém do diagramu prípadov použitia. Výsek z obrázka obr. 4.6.

Je tiež isté, že pri zoskupení podľa úzkej súvislosti, kde výsledkom transformácie je väzba generalizácie, bude vo výslednom abstraktnom prípade použitia minimálne jeden abstraktný tok, ktorý bude vo výsledných konkrétnych prípadoch použitia konkretizovaný. Každý prípad použitia, ktorý je spojený väzbou generalizácie s abstraktným prípadom použitia, musí všetky abstraktné toky konkretizovať, čiže existuje medzi nimi závislosť a preto by v Theme/Doc mali byť všetky témy, ktoré v Theme/Doc reprezentujú tieto prípady použitia, v jednom zoskupení. Keďže musia byť tieto toky konkretizované, tak v prípade použitia, ktorý ho konkretizuje, bude minimálne jeden tok výsledkom tohto konkretizovania (obr. 4.13, šípky č. 4) a bude mať rovnaký názov ako abstraktný tok, ktorý tento tok generalizuje (obr. 4.13, šípky č. 5).

Nie je ale vylúčené, že v abstraktnom prípade použitia bude viac ako jeden abstraktný tok. Taktiež nie je isté, či transformáciou tém, zoskupených na základe

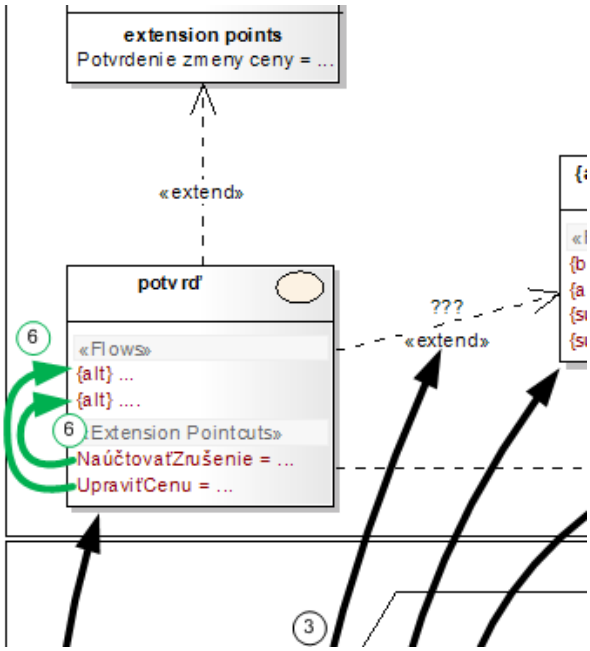
úzkej súvislosti, vôbec vznikne abstraktný prípad použitia. V takomto prípade nebude žiadny abstraktný tok a ani toky, ktoré by ho konkretizovali. No určite bude vo všeobecnom prípade použitia minimálne jeden prvok (tok, bod rozšírenia, bodový prierez rozšírenia), ktorý budú konkrétne prípady použitia zdieľať a v konkrétnych prípadoch použitia minimálne jeden prvok, ktorý ich bude odlišovať, ináč by celá generalizácia nemala zmysel.



Obr. 4.13: Vytvorenie abstraktného toku v abstraktnom prípade použitia a jeho konkrétnych tokov v prípadoch použitia pri generalizácii, ktoré vznikli zobrazením tém do diagramu prípadov použitia, ktoré boli zoskupené podľa spoločnej závislosti. Výsek z obrázka obr. 4.8.

Teoretický by sa dali zakresliť niektoré toky aj do rozširujúceho prípadu použitia, ktorý vznikol ako výsledok analógie medzi aspect/based väzbou a väzbou extend. Konkrétne sa jedná o toky, ktoré by vznikli jednoznačným priradením ku každému bodovému prierezu rozšírenia. Pre každý bodový prierez rozšírenia vytvoril vlastný alternatívny tok v danom prípade použitia (obr. 4.14). Teoreticky by to fungovalo, no nebrala by sa do úvahy možnosť znovupoužitia bodových prierezov rozšírenia viacerými tokmi. Jedná sa len o automatické vytvorenie tokov bez nejakého zamýšľania sa nad ich významom.

To isté nie je možné spraviť pri bodoch rozšírenia v rozširovaných prípadoch použitia. V rozširovaných prípadoch použitia sú toky nezávislé od bodov rozšírenia, prakticky o nich ani nevedia, preto sa s nimi nemôže pri vytváraní tokov rátať. Od bodov rozšírenia viac závisí rozširujúci prípad použitia ako samotný rozširovaný prípad použitia, keďže vystupujú ako rozhranie, ktoré musí rozširujúci prípad použitia poznať.



Obr. 4.14: Vytvorenie alternatívnych tokov z bodových prierezov rozšírenia. Výsek z obrázka obr. 4.10. Bližšie informácie ako určiť bodové prierezy rozšírenia sa nachádzajú v kapitole 4.6.

Určenie ostatých tokov a priradenie významu automaticky vytvoreným tokom už musí byť ďalej koordinované s vytváraním scenára pre ten ktorý prípad použitia.

4.8 Granulácia

Granulácia prípadov použitia a tém je subjektívna záležitosť analytika. V prípade tém môže byť granulácia ovplyvnená automatickým spracovaním požiadaviek. Keďže v tejto práci sa žiadny program nepoužíval, ale spoliehalo sa len na úsudok autora (hoci bolo z časti simulované automatické vyhľadávanie tém), je preto pochopiteľné, že po transformácii Theme/Doc pohľadov do diagramu prípadov použitia vyšiel diagram prípadov použitia vo veľkej miere podobný z úvodu tejto práce (obr. 2.1). Najmä preto, že oba modely robil jeden autor. No aj v prípade, že by tieto modely robili rozdielni ľudia, tak je veľká pravdepodobnosť, že témy budú jemnejšej granulácie ako prípady použitia.

Ak sa porovná transformovaný model s modelom z úvodu práce, tak je vidieť, že transformovaný diagram prípadov použitia obsahuje abstraktný prípad použitia *naúčtuj*, ktorý bližšie konkretizujú prípady použitia *odstráň položku* a *naúčtuj položku*. V pôvodnom modely je namiesto týchto prípadov použitia iba jeden s názvom *naúčtuj nákup*. Je tu jasný rozdiel v granulácii. V pôvodnom modely sa ako prípad použitia malo na mysli naúčtovanie celého nákupu, zatiaľ čo v druhom prípade postup zdola pri hľadaní tém spôsobil, že sa skončilo na úrovni naúčtovania jednej položky.
Je ťažké posúdiť, ktorý model je lepší. Jemnejšia granulácia môže zjednodušiť prácu vo fáze návrhu, ale na druhú stranu sa diagram môže stať nečitateľným pre zainteresované osoby, ktoré s takýmito diagramami nepracujú každý deň, čo je zase proti poslaniu diagramov prípadov použitia. V tejto práci je ale dôležitejší postup, ako vytvoriť hrubšiu granuláciu diagramu, ktorý nám vznikol transformáciou Theme/Doc pohľadov.
Keď sa rozhodne o zmene na hrubšiu granuláciu modelu prípadov použitia, tak sa musí pristúpiť nielen k zoskupovaniu jednotlivých prípadov použitia, ale aj k zmene ich významu, čo v diagrame znamená zmeniť ich názov. Myslí sa tým zmena významu čo do rozšírenia pôsobnosti záležitosti a nie zmena podstaty záležitosti. Spájaním súvisiacich záležitosti by sa tak mala získať záležitosť so širším významovým záberom. Môže sa tak stať, že záležitosti, ktoré boli pri jemnejšej granulácii oddelené, sa pri viacnásobných zoskupovaniach do prípadov použitia hrubšej granulácie môžu ocitnúť pod jedným prípadom použitia, t.j. pod jednou záležitosťou.

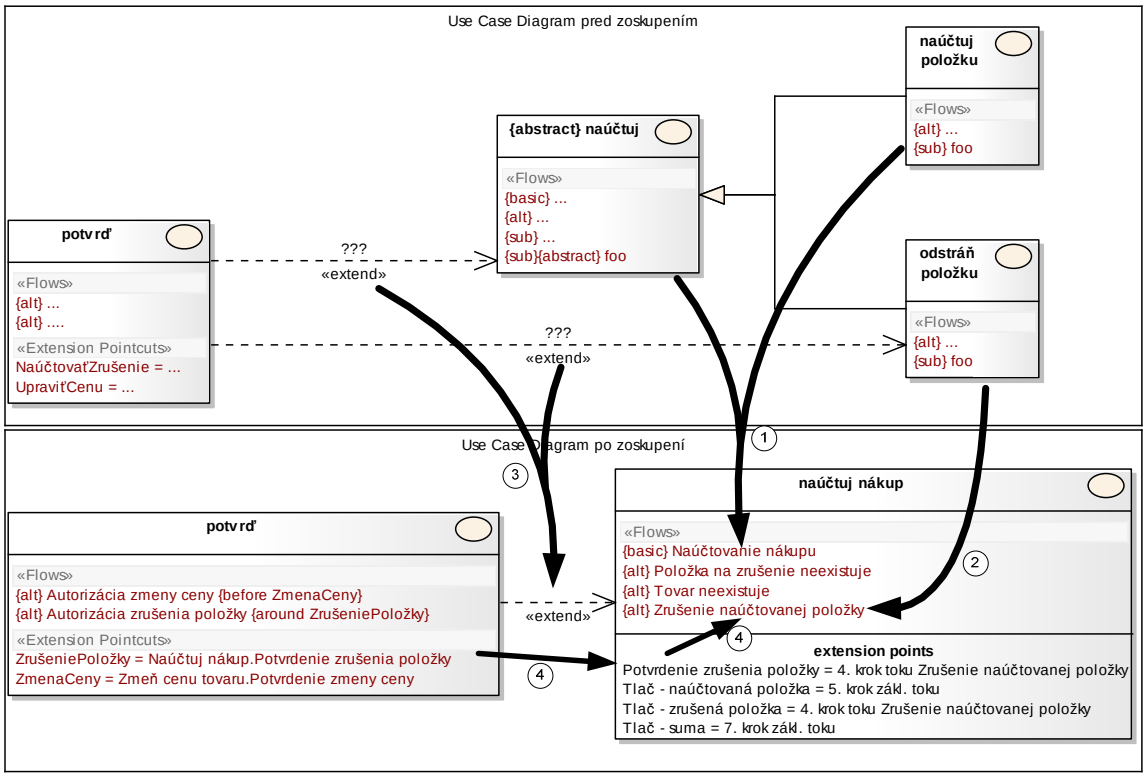
Zoskupovať prípady použitia, ktoré sú prepojené väzbou include nie je vhodné, keďže zahrňovaný prípad použitia musí byť k dispozícii aj pre iné prípady použitia. V prípade, že by to tak nebolo, tak v zoskupenom prípade použitia bude tento pôvodne zahrňovaný prípad použitia vystupovať ako nový podtok, ktorý sa pri zoskupovaní vytvorí. Keďže zahrňované prípady použitia sú vyčlenené do samotného prípadu použitia z dôvodu jeho znovupoužitia inými prípadmi použitia, tak k takémuto zoskupeniu bude dochádzať málokedy. O to viac, keď prípad použitia, ktorý je zahrňovaný len jedným prípadom použitia, by sa teoreticky ani nemal v diagrame prípadov použitia vyskytnúť.

Podobne sa môžu zoskupiť aj prípady použitia, ktoré sú spojené väzbou generalizácie. Je to aj prípad nášho transformovaného modelu prípadov použitia. Vytvorí sa prípad použitia, ktorý bude z hľadiska granulácie na vyššej úrovni a do neho sa vložia navzájom súvisiace prípady použitia *naúčtuj*, *odstráň položku* a *naúčtuj položku*. Nový prípad použitia sa bude volať *naúčtuj nákup*. Po tomto vložení väzba generalizácie zanikne. Môže ale naznačiť, že prípady použitia *odstráň položku* a *naúčtuj položku* majú podobné správanie a preto je možné pre oba prípady použitia vytvoriť len jeden spoločný tok. Inak je lepšie pre každý takýto prípad použitia vytvoriť samostatný tok, resp. podtok.

To či to bude podtok alebo alternatívny tok záleží od povahy týchto prípadov použitia, ako aj od toho, do akej miery sa čo do úrovne líšia od nového prípadu použitia, do ktorého sú tieto prípady použitia vkladané. Ak sa úrovňovo veľmi nelíšia, tak je možné tieto prípady použitia vložiť ako nové alternatívne toky. Ináč je vhodnejšie ich vložiť ako podtoky.

Je potrebné taktiež dbať aj na povahu týchto prípadov použitia. Pri naúčtovaní jednej položky mali tieto prípady použitia veľa spoločného, rozdiel bol len v tom, že sa pri jednom pripočíta cena k výslednej sume tovaru a pri druhom sa odčíta po úspešnej autentifikácii, preto vlastne boli zoskupené v Theme/Doc modeli. No pri vytvorení nového prípadu použitia *naúčtuj nákup* sa ich úloha zmenila. Prípad použitia *naúčtuj položku* teraz môže vystupovať ako neoddeliteľná súčasť základného scenára (obr. 4.15, šípka č. 1), alebo bude vystupovať ako podtok, ktorý sa bude volať z hlavného scenára. Ale prípad použitia *odstráň položku* môže teraz vystupovať ako alternatívny tok, keďže predstavuje výnimku, ktorá môže nastať v hlavnom toku. A výnimky sa v scenári modelujú prostredníctvom alternatívnych tokov (obr. 4.15, šípka č. 2).

V kapitole 4.4 bol problém s určením väzby extend podľa individuálneho pohľadu zoskupenej témy. Po takejto zmene granulácie sa však tento problém automaticky vyriešil a väzba extend sa teraz nachádza medzi prípadmi použitia *potvrď* a *naúčtuj nákup* (obr. 4.15, šípka č. 3). Keďže prípad použitia *odstráň položku* je vložený do vlastného alternatívneho toku, tak pre danú extend väzbu sa v prípade použitia *naúčtovať nákup* vytvorí bod rozšírenia, ktorý sa bude odvolávať na presné miesto v tomto alternatívnom toku (obr. 4.15, šípky č. 4).



Obr. 4.15: Vytvorenie hrubšej granulácie diagramu prípadu použitia.

Pri vkladaní rozširujúcich prípadov použitia do prípadu použitia hrubšej granulácie si treba dávať pozor na to, že tento rozširujúci prípad použitia môže rozširovať aj iné prípady použitia. Vtedy by sa aj prípad hrubšej granulácie mal stať rozširujúcim prípadom použitia pre tieto prípady použitia a bývalý rozširujúci prípad použitia by mal vystupovať pod vlastným alternatívnym tokom.

Takýmto spôsobom je možné spájať aj prípady použitia, medzi ktorými nie je žiadny vzťah. Treba si ale dobre rozmyslieť, či má význam spájať takéto oddelené záležitosti, medzi ktorými nie je žiadny súvis, pretože pri návrhu sa môže zistiť, že tieto záležitosti sú natoľko oddelené, že ich udržiavanie ako jednu záležitosť sťažuje návrh systému.

4.9 Zhrnutie postupu transformácie

V tejto časti budú zhrnuté kroky transformácie Theme/Doc pohľadov do modelu prípadov použitia:

1. Z pohľadu pretínajúcich tém sa prevedú všetky aspektové témy na rozširujúce prípady použitia a základné témy na rozširované prípady použitia a vytvorí sa medzi nimi väzba typu extend tým istým smerom, ako sú vytvorené aspect/based väzby v pohľade pretínajúcich tém. Osamostatnené základné témy sa prevedú na základné prípady použitia.
2. Prejdú sa individuálne pohľady každej jednej témy a hľadajú sa podtémy, ktoré boli zoskupené podľa pravidla degradácie. Všetky takéto podtémy sa potom navzájom porovnajú, aby sa zistilo, či nezodpovedajú za rovnaké správanie. Ich správanie je možné vyčítať nielen z ich názvu, ale aj z požiadavky, ktorá je k nim pripojená. Všetky takéto podtémy s rovnakým správaním sa prevedú na jeden inclusion prípad použitia s minimálne jedným podtokom a s názvom, ktorý najlepšie charakterizuje jeho správanie. Tento prípad použitia sa potom prepojí väzbou include s každým prípadom použitia, ktorý je zobrazením témy, ktorá takúto podtému s rovnakým správaním zoskupovala. Pre všetky ostatné podtémy zoskupené podľa pravidla degradácie sa vytvorí v prípade použitia, ktorý je obrazom témy, ktorá danú podtému zoskupovala, podtok s názvom, ktorý najlepšie charakterizuje správanie danej podtémy (alebo samotný názov podtémy).
3. Prejdú sa individuálne pohľady každej jednej témy a hľadajú sa podtémy, ktoré boli zoskupené podľa pravidla úzkej súvislosti. Pre takéto podtémy sa vytvorí prípad použitia, ktorý je väzbou generalizácie prepojený s prípadom použitia, ktorý je obrazom témy, ktorá danú podtému zoskupovala. Podľa charakteru záležitosti sa môže prípad použitia, ktorý danú tému zoskupoval, označiť ako abstraktný. Vtedy sa v ňom vytvorí abstraktný tok a v prípadoch použitia, ktoré sú obrazom daných podtém, sa vytvorí tok, ktorý bude vytvorený abstraktný tok konkretizovať.
4. Prejdú sa individuálne pohľady tém a pri témach, ktoré zoskupujú nejaké podtémy, sa skontroluje, či náhodou nejaký aspect/based vzťah nemá byť v

skutočnosti spojený s podtémou podtémy, ktorú zoskupuje. Ak áno, tak je danú väzbu potrebné zmeniť aj v modeli prípadov použitia.

5. Všetky odročené väzby v Theme/Doc pohľadoch sa prevedú na bližšie nešpecifikované závislosti medzi príslušnými prípadmi použitia.
6. Z pohľadu pretínajúcich tém sa pre každú aspektovú tému zistia požiadavky, ktoré súvisia s jej aspect/based vzťahmi a podľa nich sa v každom rozširujúcom prípade použitia, ktorý je jej obrazom, vytvoria príslušné bodové prierezy rozšírenia. Pre každý bodový prierez rozšírenia rozširujúceho prípadu použitia sa vytvorí v prípade použitia, ktorý je rozširovaný, prislúchajúci bod rozšírenia. Zároveň sa pre každý bodový prierez rozšírenia vytvoria v danom rozširujúcom prípade použitia prislúchajúce alternatívne toky.
7. Podľa potreby sa vytvoria nové prípady použitia s hrubšou granuláciou, do ktorých sa vložia prípady použitia, ktoré dokáže záležitosť modelovaná novým prípadom použitia pokryť. Vložené prípady použitia budú v novom prípade použitia reprezentované ako toky.
8. Z požiadaviek a sledovaním prekonvertovaného modelu sa vytvorí pre každý prípad použitia scenár, podľa ktorého sa doplnia toky, odkazy bodov rozšírenia, konkretizujú sa bližšie nešpecifikované závislosti a rozhodne sa o konečných názvoch prípadov použitia.

5 Transformácia modelu prípadov použitia do Theme/Doc modelu

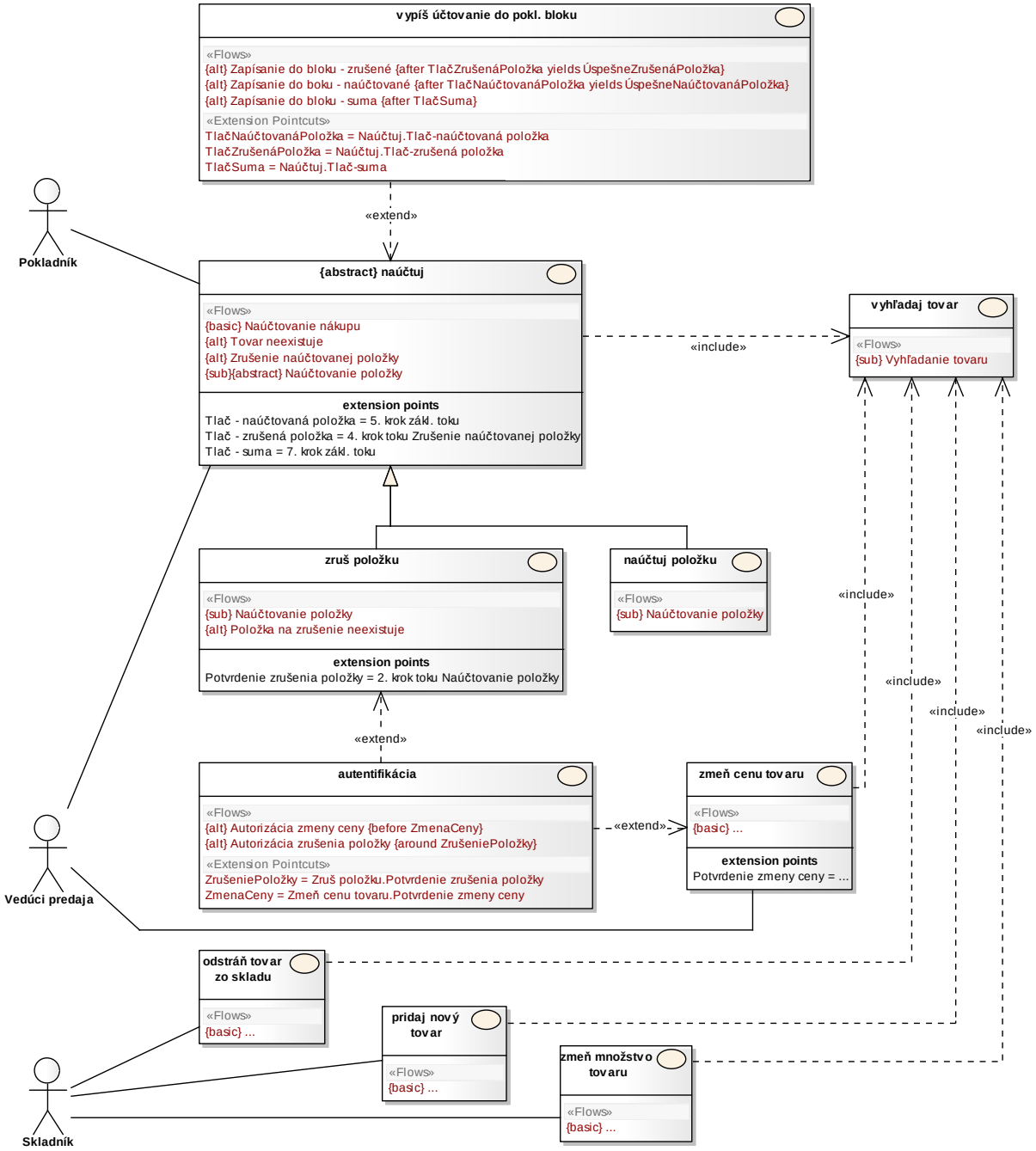
Nasledujúca kapitola sa bude venovať transformácii modelu prípadov použitia do modelu v notácii Theme/Doc.

Cieľom je navrhnúť taký postup transformácie, aby výsledný model v notácii Theme/Doc bol plnohodnotný a čo najpresnejšie odzrkadľoval záležitosti daného modelu prípadov použitia. To znamená, že snahou bude popísať prvky modelu prípadov použitia (prípady použitia, väzby, extension points, extension pointcuts, atď.) prvkami notácie Theme/Doc (témy, aspect/based vzťahy, požiadavky, atď.). Z doterajších poznatkov o vzťahoch medzi prípadmi použitia a témami a opačnom prevode sa dajú v tomto procese očakávať nasledujúce kroky:

1. Prevod prípadov použitia na témy.
2. Transformácia vzťahov medzi prípadmi použitia na vzťahy medzi témami
3. Pridanie požiadaviek do modelu
4. Pridanie entít do modelu
5. Úprava granulácie transformovaného modelu v notácii Theme/Doc.

Výstupom prvých štyroch krokov by mali byť všetky prvky, ktoré notácia Theme/Doc podporuje. Prvky prípadov použitia ako sú extension points, extension pointcuts, toky, účastníci a ostatné nemajú jednoznačný obraz v žiadnom z prvkov modelu v notácií Theme/Doc, no určite budú mať vplyv na rozhodovanie v spomínaných krokoch transformácie. Posledný krok nie je nevyhnutný na vytvorenie plnohodnotného modelu v notácii Theme/Doc.

Pri transformácii sa bude vychádzať z modelu prípadov použitia z úvodu dokumentu, ktorý ale bude mierne upravený, tak aby sa tam nachádzala aj väzba generalizácie (obr. 5.1).



Obr. 5.1: Model prípadov použitia systému.

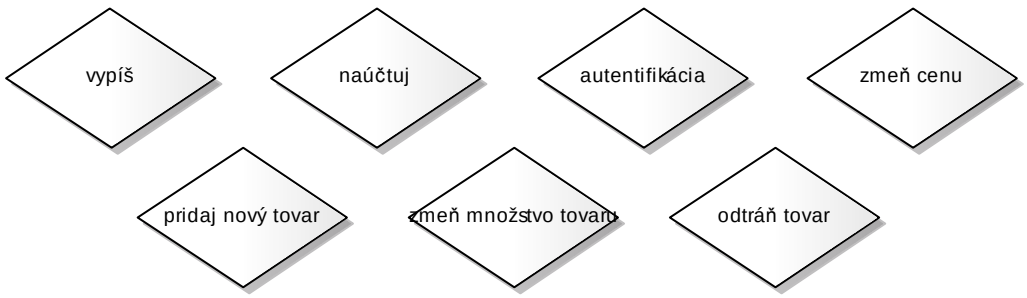
5.1 Prevod prípadov použitia na témy

5.1.1 Pohľad pretínajúcich tém

Je potrebné pripomenúť, že diagram prípadov použitia a samotné prípady použitia sú v stave, kedy je už rozhodnuté o ich zodpovednosti a vzťahoch, to znamená, že sa vie, ktoré sú základné, rozšírené, zahrňované, generalizované. Preto pri ich transformácii by mal vzniknúť pohľad tém, v ktorom už bude o vzťahoch medzi témami a ich zodpovednosti rozhodnuté. Takže proces rozhodovania o témach sa preskočí a začne sa s vytváraním pohľadu pretínajúcich tém.

Samozrejme, je možné vytvoriť najprv pohľad tém a vzťahov a nezaoberať sa už ďalej prípadmi použitia, pričom na vytvorenie pohľadu pretínajúcich tém by sa použil daný pohľad tém a vzťahov a pravidlá a postupy definované v Theme/Doc. No cieľom je vytvoriť taký pohľad pretínajúcich tém, aby čo najlepšie odzrkadľoval náš referenčný model prípadov použitia. Pri vychádzaní z pohľadu tém a vzťahov a Theme/Doc postupov by výsledný pohľad pretínajúcich tém nemusel presne zodpovedať nášmu modelu prípadov použitia. Preto je lepšie, ak sa hneď začne s pohľadom pretínajúcich tém, ktorý bude vychádzať z modelu prípadov použitia.

Z časti venovanej vzťahom medzi prípadmi použitia a témami je známe, že základné a pretínajúce témy zodpovedajú základným a rozširujúcim prípadom použitia a naopak. Preto prvým krokom bude prehlásenie všetkých základných a rozširujúcich prípadov použitia za témy v pohľade pretínajúcich tém (obr. 5.2).



Obr. 5.2: Témy, ktoré budú použité v pohľade pretínajúcich tém, získané transformáciou základných a rozširujúcich prípadov použitia.

5.1.2 Individuálny pohľad tém

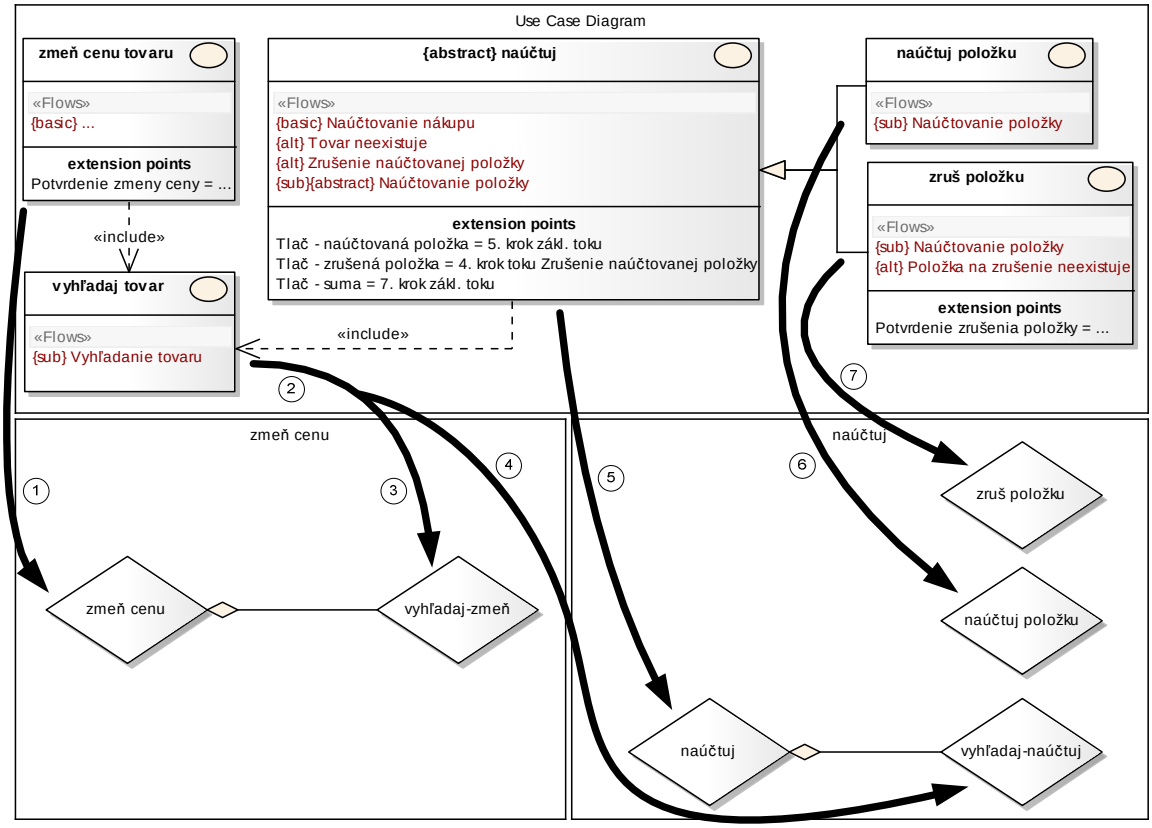
V pohľade pretínajúcich tém sú podtémy zoskupené pod svojimi hlavnými témami a nie je ich vidieť. Sú ale k dispozícii pri individuálnom pohľade tém, preto je vhodné, ak sa paralelne s vytváraním pohľadu pretínajúcich tém vytvára aj individuálny pohľad.

Pri transformácii platia podobné pravidlá ako pri opačnom procese. Generalizácia sa zobrazí ako zoskupenie podľa úzko súvisiaceho správania a podtoky ako zoskupenie podľa degradácie správania vyjadrené témou na správanie vyjadrené podtémou.

Keďže jedna podtéma nemôže byť zoskupená pod viacerými témami, tak v prípade, že je podtok vytiahnutý do samostatného inclusion prípadu použitia, je potrebné prislúchajúcu podtému rozdeliť na viac podtém podľa toho, koľko prípadov použitia tento inclusion prípad použitia zahŕňa. V našom príklade sa preto prípad použitia *vyhľadaj tovar* prevedie nie na jednu podtému, ale pre každý prípad použitia, ktorý zahŕňa tento inclusion prípad použitia, sa vytvorí vlastná podtéma s názvom *naúčtuj*, ktorý je navyše doplnený o nejaký jednoznačný identifikátor tak, aby boli tieto podtémy jednoznačne rozlíšiteľné. Na obrázku (obr. 5.3) je znázornený prevod prípadu použitia *vyhľadaj tovar*, ktorý je zahrňovaný okrem iných aj prípadmi použitia *zmeň cenu tovaru* a *naúčtuj*. V individuálnom pohľade tém, ktoré vznikli prevodom týchto dvoch prípadov použitia, sa inclusion prípad použitia *vyhľadaj* (obr. 5.3, šípka č. 2) transformoval na podtémy s názvami *vyhľadaj-zmeň* (obr. 5.3, šípka č. 3) pre tému *zmeň-cenu* (obr. 5.3, šípka č. 1) a *vyhľadaj-naúčtuj* (obr. 5.3, šípka č. 4) pre tému *naúčtuj* (obr. 5.3, šípka č. 5).

Generalizácia prípadov použitia *naúčtuj položku* a *zruš položku* je v našom príklade transformovaná na podtémy s rovnakým názvom (obr. 5.3, šípky č. 6 a 7), ktoré sú zoskupené na základe blízkej súvislosti pod témou *naúčtuj* (obr. 5.3, šípka č. 5), ktorá zodpovedá rovnomennému abstraktnému prípadu použitia. Táto generalizácia je špecifická tým, že hoci obsahuje podtoky, tak boli vytvorené podtémy na základe blízkej súvislosti a nie na základe degradácie témy, resp. ďalej sa tieto podtémy nerozbíjali na menšie podtémy. Tieto podtoky neboli vyčlenené do ďalších podtém, pretože nám zabezpečujú špecifickosť prípadov použitia *naúčtuj položku* a *zruš položku*. Pri ich vyčlenení do vlastných podtém by prípady použitia, resp. podtémy *naúčtuj položku* a *zruš položku* stratili to čo ich odlišovalo a mohlo by sa stať, že by predstavovali rovnaké správanie, t. j. samotná generalizácia by stratila význam. Takže podtoky, ktoré konkretizujú nejaký abstraktný podtok sa ďalej netransformujú na menšie podtémy. V prípade, že by tieto podtoky neplnili úlohu špecializácie abstraktného podtoku, tak by sa dalo uvažovať nad ich prevodom do vlastných podtém.

V prípade, že by podtok zahŕňal nejaký ďalší podtok (resp. inclusion prípad použitia iný inclusion prípad použitia), tak nie je vhodné tieto menšie podtoky prevádzať na vlastné podtémy. Takéto niekoľko úrovňové rozdiely medzi témami pôsobia nekonzistentne a je dobré porozmýšľať nad zmenou granulácie niektorých tém, aby sa celý model dostal rovnakú úroveň. V diagrame prípadov použitia by sa takéto prípady ani nemali objavovať, keďže to zaváňa funkcionálnou dekompozíciou. V prípade generalizácie a zoskupovania tém podľa blízkej súvislosti to nie je až taký problém, keďže tam sa základný prípad použitia, resp. téma nerozkladá na menšie časti, len sa konkretizuje.



Obr. 5.3: Prevod generalizácie a inclusion prípadu použitia na podtémy v individuálnom pohľade tém zmeň cenu a naúčtuj.

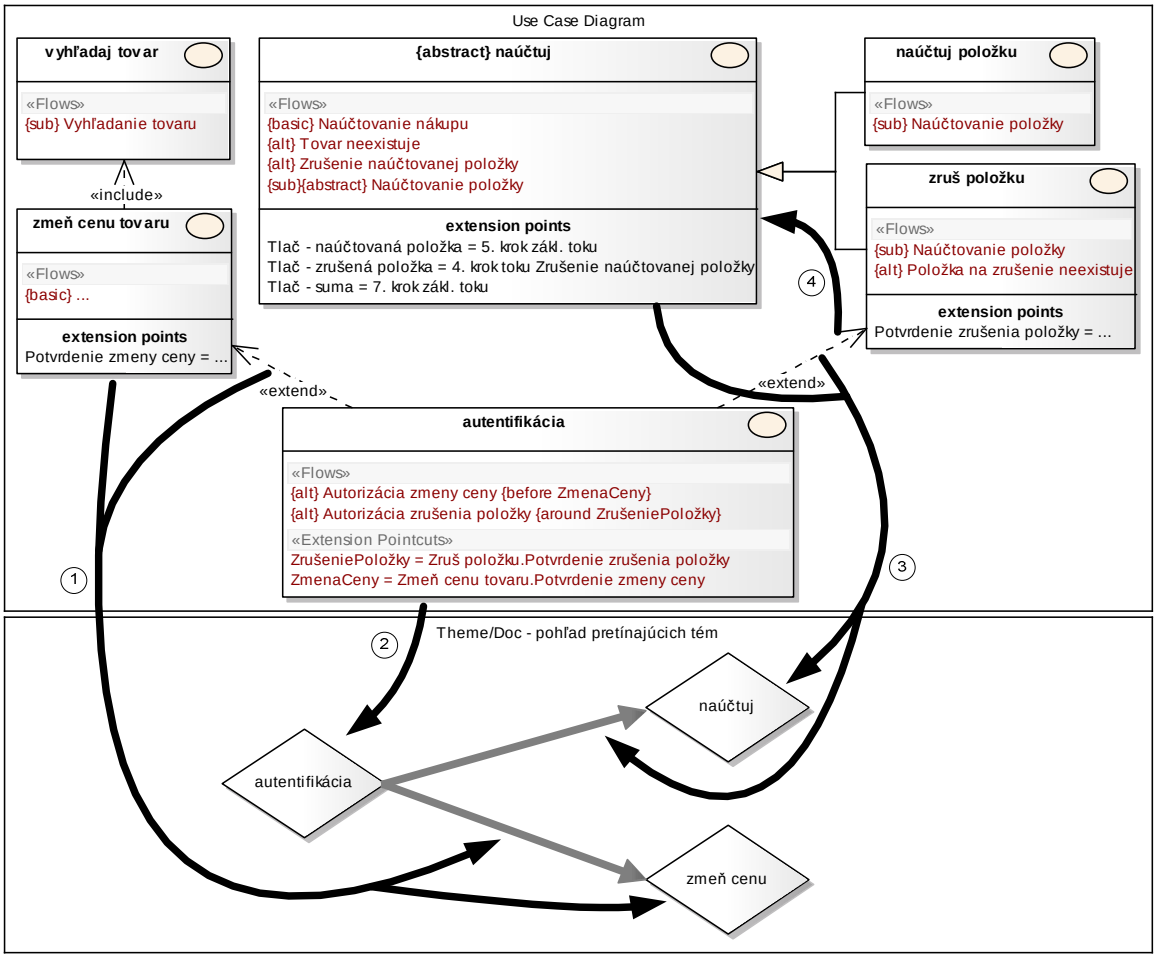
5.2 Transformácia vzťahov

5.2.1 Pohľad pretínajúcich tém

Vzťahy medzi témami v pohľade pretínajúcich tém sú určené pomocou aspect/based väzieb alebo odročených väzieb. V prípadoch použitia sú to väzby include, extend a generalizácia. Väzby include a generalizácia boli už premietnuté do zoskupenia v predošlej časti. Ostáva už iba väzba extend.

Keďže základné a pretínajúce témy zodpovedajú základným a rozširujúcim prípadom použitia, tak potom väzba extend medzi prípadmi použitia zodpovedá aspect/based väzbe medzi témami. Väzby medzi prípadmi použitia sú presne určené, preto pri transformácii týchto väzieb na väzby medzi témami, by nemali vznikať odročené väzby. Podľa príkladu na obrázku (obr. 5.4) to znamená, že v pohľade pretínajúcich tém medzi témami *autentifikácia* a *zmeň cenu* vznikne aspect/based väzba (obr. 5.4, šípky č. 1 a 2).

Hoci podtémy nie sú v pohľade pretínajúcich tém zobrazené, je nutné sledovať väzby extend aj medzi prípadmi použitia zodpovedajúcich témam a prípadmi použitia zodpovedajúcich podtémam. Pri zoskupení sa totiž všetky väzby medzi podtémami a okolitými témami prenesú na tému, pod ktorou sú tieto podtémy zoskupené. V diagrame prípadov použitia to teda vyzerá, akoby sa všetky väzby extend medzi prípadmi použitia, ktoré sa zobrazia do normálnych tém a prípadmi použitia, ktoré sa zobrazia do podtém, presunuli na prípady použitia, ktoré sa zobrazia ako témy, pod ktorými sú tieto podtémy zoskupené. V príklade na obrázku (obr. 5.4) sa teda bude väzba extend medzi prípadmi použitia *autentifikácia* a *zruš položku* brať tak, akoby to bola väzba medzi prípadmi použitia *autentifikácia* a *naúčtuj* (obr. 5.4, šípka č. 4). Preto sa táto väzba prevedie na aspect/based väzbu medzi témami *autentifikácia* a *naúčtuj* v pohľade pretínajúcich tém (obr. 5.4, šípky č. 2 a 3).

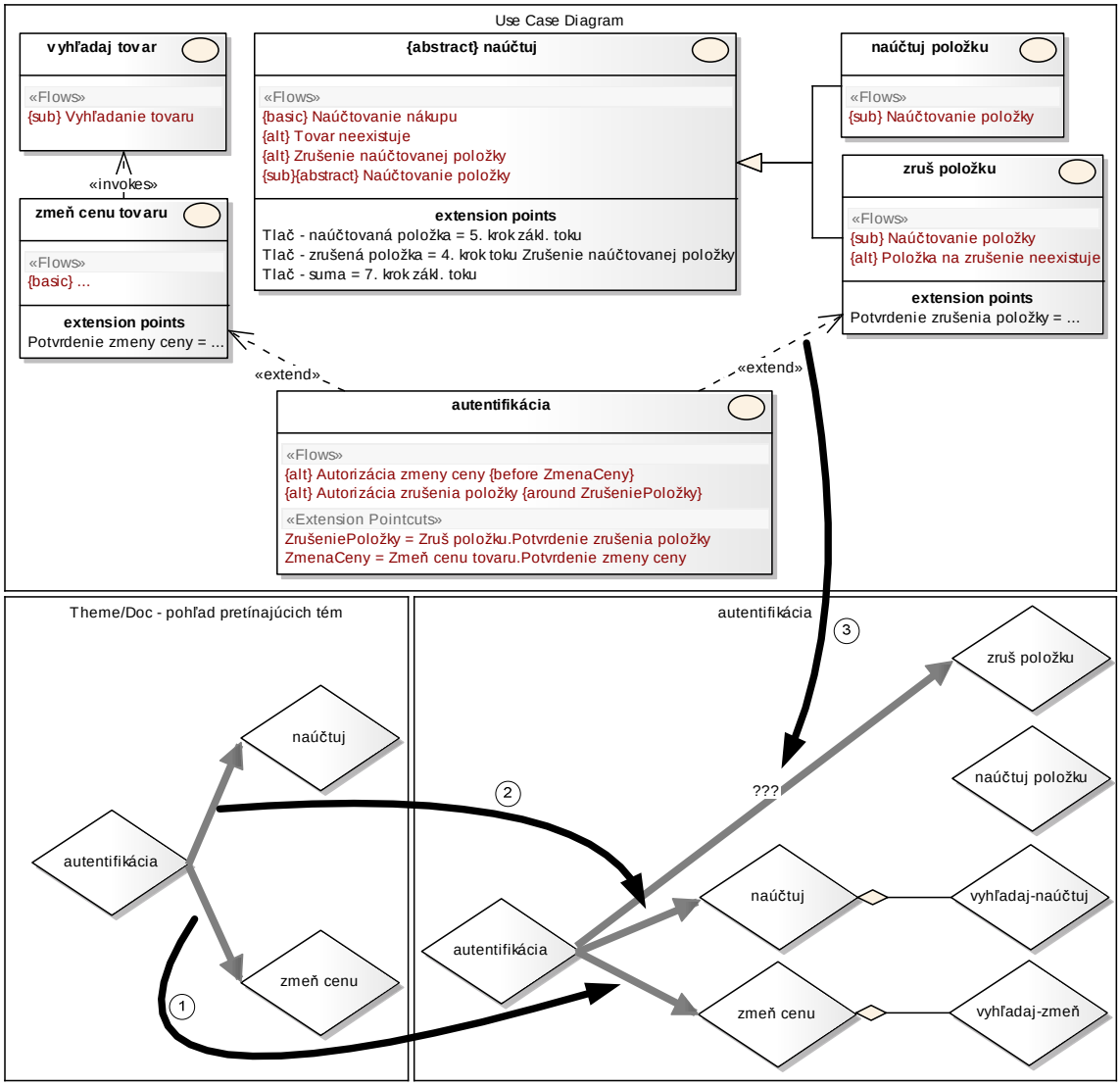


Obr. 5.4: Transformácia väzieb extend na aspect/based väzby v pohľade pretínajúcich tém.

5.2.2 Individuálny pohľad tém

Po doplnení väzieb v pohľade pretínajúcich tém je potrebné upraviť aj individuálny pohľad tém. Téma *autentifikácia* sa po transformácii vzťahov stala aspektová, preto musia byť v individuálnom pohľade tejto témy aj témy, ktoré pretína, čiže témy *naúčtuj* a *zmeň cenu* (obr. 5.5, šípky č. 1 a 2). Spolu s nimi sú zobrazené aj podtémy, ktoré zoskupovali.

Podobne ako pri opačnom prevode aj tu nastáva otázka, či by v individuálnom pohľade témy *autentifikácia* nebolo lepšie prepojiť aspect/based väzbu priamo s podtémou *zruš položku*, tak ako to v skutočnosti je (obr. 5.5, šípka č. 3), pretože ináč sa informácia o tejto skutočnosti z Theme/Doc modelu vytratí. Podľa pravidiel zoskupovania a vytvárania individuálnych pohľadov to nie je možné a Theme/Doc sa tvári, akoby bolo zakázané prepájať témy s podtémami aspect/based väzbou, resp. že je zakázané vytvárať podtémy z tém, ktoré sú pretínané.



Obr. 5.5: Doplnenie aspect/based väzieb do individuálneho pohľadu témy autentifikácia.

5.3 Pripojenie požiadaviek k témam

Pre vytvorenie plnohodnotného Theme/Doc modelu je potrebné transformovaný model ešte doplniť o požiadavky a entity (kľúčové slová).

Prípady použitia v UML notácii ani ich textový opis neobsahujú požiadavky, ktoré s daným prípadom použitia súvisia. Obsahujú však iné prvky, pomocou ktorých je možné sa pokúsiť o vytvorenie požiadavky aplikovaním reverzného inžinieringu.

Reverzný inžiniering je ťažké automatizovať aj na nižších úrovniach vývoja softvéru, preto nie je možné očakávať, že vytvoríme všeobecne aplikovateľný automatický postup reverzného inžinieringu vo fáze analýzy vzhľadom na to, že pôjde najmä o hľadanie sémantiky.

5.3.1 Pohľad pretínajúcich tém

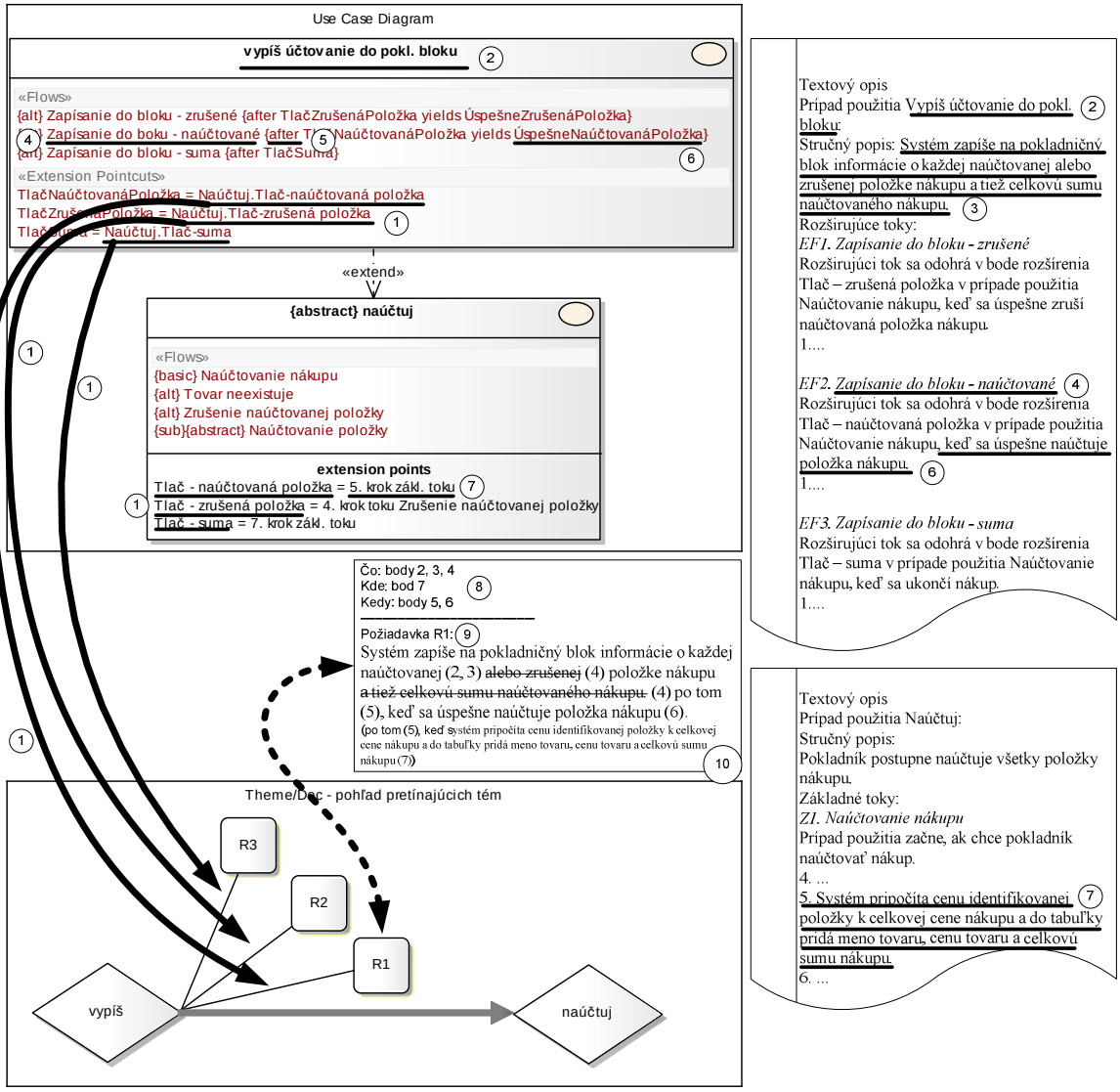
O požiadavkách v pohľade pretínajúcich tém vieme povedať, že ku každej aspektovej téme a každej osamostatnenej základnej téme musí byť pripojená minimálne jedna požiadavka. Aspektová téma preto, lebo musí nejakú požiadavku pretínať, inač by nebola aspektová. Analogicky z toho vyplýva, že pri každej aspect/based väzbe, vychádzajúcej z tejto témy, musí byť pripojená minimálne jedna požiadavka. O osamostatnenej základnej téme to platí preto, lebo ak by nesúvisela so žiadnou požiadavkou, tak by nemohla ani vzniknúť. V prípade základnej témy, ktorá je prepojená s aspektovou témou, to nemusí platiť. Ak s takouto základnou témou je prepojená jedna požiadavka, tak tá bude presunutá k súvisiacej aspketovej téme. Preto takáto základná téma nemusí mať žiadnu pripojenú požiadavku. Taktiež platí, že pri zoskupovaní tém sa všetky požiadavky, ktoré boli pripojené k zoskupovaným témam, pripoja k téme, pod ktorou sú zoskupené. V prípade, že sa nájdu prislúchajúce požiadavky danej témy, tak už nebude dôležité, či budú tieto požiadavky spojené v jednej väčšej požiadavke, alebo bude každá samostatne prepojená s danou témou.

Požiadavky by mali byť niekde spísané, keďže sa podľa nich vytváral diagram prípadov použitia. Preto jedným z možných spôsobov ako by sa dalo postupovať, je určiť každému prípadu použitia prislúchajúcu požiadavku a tú potom pripojiť k téme, ktorá vznikla transformáciou z daného prípadu použitia. Ale problémom je, že požiadavky v pohľade pretínajúcich tém nie sú tie isté ako požiadavky, ktoré boli zadefinované na začiatku. Požiadavky v pohľade pretínajúcich tém prešli operáciami ako spájanie, rozdeľovanie a odstraňovanie s cieľom získať také požiadavky, ktoré by neboli zdieľané viacerými témami. Preto je potrebné na základe diagramu prípadov použitia, resp. ich textových opisov, vytvoriť nové požiadavky tak, aby každá téma disponovala vlastnou nezdieľanou a podľa možností nepretínajúcou požiadavkou. Keďže téma zodpovedá prípadu použitia, preto informácie na vytvorenie požiadavky súvisiacej s danou témou je najlepšie hľadať práve v prípade použitia, z ktorého bola prevedená.

Veľký vplyv na spätné vytvorenie požiadaviek ma fakt, či k prípadom použitia je k dispozícii ich textový opis, alebo je nutné sa uspokojiť iba so samotným diagramom. Ak je k dispozícii popis, tak to ako by mohla požiadavka vyzerať, je možné zistiť z krátkych popisov prípadov použitia a z popisov tokov. V prípade, že textový opis nie je k dispozícii, tak je možné vychádzať len z názvu prípadu použitia, názvov jeho tokov, bodov rozšírenia a podmienky rozšírenia. Samozrejme v takom prípade sa body rozšírenia neodvolávajú na scenár, ale slovne opisujú, kde nastáva rozšírenie. No takáto situácia by teoreticky nemala nastať, keďže textový opis je podstatou prípadov použitia.

Pripojenie požiadaviek k aspektovým témam

Požiadavky pripojené k aspektovým témam by mali byť najmä tie, ktoré súvisia s aspect/based väzbou. Takéto požiadavky priamo súvisia s bodovými prierezmi rozširujúceho prípadu použitia, z ktorého bola daná aspektová téma prevedená. Tie nám hovoria, koľkokrát daná aspektová téma pretína iné základné témy a tiež odkazujú na body rozšírenia, ktoré zase hovoria, na ktorom mieste dochádza k rozšíreniu, resp. pretínaniu. K dispozícii je aj podmienka rozšírenia, ktorá sa nachádza v názve rozširujúceho toku za slovom *yields*. Táto podmienka zase hovorí, kedy dochádza k rozšíreniu, resp. pretínaniu. A kedy relatívne k miestu rozšírenia je zase definované v názve rozširujúcich tokov pomocou slov *after*, *before*, *around*. Všetky tieto prvky môžu byť priamo alebo nepriamo definované aj v textovom opise prípadu použitia.



Obr. 5.6: Konštrukcia a priradenie požiadavky k aspektovej téme.

Na obrázku (obr. 5.6) je zobrazený postup konštrukcie a pripojenia požiadavky k aspektovej téme *vypíš*. Bodové prierezy rozšírenia v súvisiacom prípade použitia hovoria o výskyte troch pretínaní touto témou. Rozhodli sme sa, že každé pretínanie bude zachytené v jednej požiadavke. Zaujímať nás budú požiadavky súvisiace s aspect/based väzbou, ktorá smeruje k téme *naúčtuj*, ktorých počet zodpovedá počtu rozširujúcich bodových prierezov, ktoré odkazujú na prípad použitia, z ktorého vznikla téma *naúčtuj*. V tomto prípade sú to všetky tri spomínané bodové prierezy, čo

znamená, že k téme *vypíš* budú k aspect/based väzbe pripojené tri požiadavky (obr. 5.6, body č. 1).

Každá požiadavka pripojená k aspektovej téme, resp. aspect/based väzbe, by mala určovať (obr. 5.6, bod č. 8):

- Čo má systém robiť
- Kde dochádza k pretínaniu
- Kedy dochádza k pretínaniu

Predstavu o tom, čo robí, je možné získať z názvu súvisiaceho prípadu použitia (obr. 5.6, bod č. 2) spolu s krátkym textovým popisom prípadu použitia (obr. 5.6, bod č. 3), názvu toku využívajúceho daný bodový prierez a textový opis tohto toku (obr. 5.6, bod č. 4). Textový opis prípadu použitia v krátkosti opisuje správanie celého prípadu použitia, no my potrebujeme opis prípadu použitia len pre jedno rozšírenie. Na zúženie opisu len na jedno rozšírenie je vhodné použiť opis toku súvisiaceho s daným bodovým prierezom, pomocou ktorého sa vyberú tie časti opisu, ktoré s daným rozšírením, t. j. tokom, súvisia, resp. nesúvisia, alebo sa podľa neho doplnia nové informácie. V našom príklade postačí aj dostatočne výstižný názov toku, podľa ktorého boli zo vznikajúcej požiadavky odstránené informácie o výpise na pokladničný blok pri zrušení položky (táto informácia bude v požiadavke *R2*) a pri celkovej sume (táto informácia bude v požiadavke *R3*) (obr. 5.6, bod č. 9 – preškrtnutý text). Týmito krokmi je možné spätne vytvoriť tú časť požiadavky, ktorá obsahuje informácie určujúce čo má systém robiť z hľadiska danej témy a daného pretínania (*Systém zapíše na pokladničný blok informácie o každej naúčtovanej položke nákupu*).

Informáciu o tom, kedy dochádza k pretínaniu, je možné doplniť z podmienky rozširujúceho toku, ktorá sa nachádza v deklarácii rozširujúceho toku v diagrame alebo v textovom opise (obr. 5.6, bod č. 6) a zo sufixu deklarácie rozširujúceho toku (obr. 5.6, bod č. 5). V našom príklade z toho vyplýva, že to čo má systém urobiť, sa urobí po tom, keď sa úspešné naúčtuje položka nákupu.

Miesto, kde ma dôjsť k rozšíreniu, je definované v bodovom priereze buď priamo určením kroku v scenári, alebo prostredníctvom odkazu na bod rozšírenia, ktorý slúži ako rozhranie medzi rozširovaným a rozširujúcim prípadom použitia. Väčšinou sa používa odkaz na meno bodu rozšírenia, takže rozširujúci prípad použitia nevidí skutočné miesto rozšírenia. V tom prípade sa vychádza iba z názvu tohto odkazu. Ak v bodovom priereze je určený priamo krok v scenári rozširovaného prípadu použitia, tak je možné to zohľadniť v požiadavke a podľa daného kroku scenára bližšie rozpísať miesto, kde dochádza k rozšíreniu (obr. 5.6, body 7 a 10).

Výsledkom týchto krokov, je vytvorenie požiadavky, ktorá znie: *Systém zapíše na pokladničný blok informácie o každej naúčtovanej položke nákupu po tom, keď sa úspešne naúčtuje položka nákupu* (obr. 5.6, bod č. 9 – čísla v zátvorkách za textom znamenajú body v obrázku, na základe ktorých tento text vznikol, resp. bol odstránený). Tá sa pripojí k téme *vypíš*, k aspect/based väzbe, ktorá smeruje k téme *naúčtuj*. Takto sa postupne vytvoria požiadavky pre každý bodový prierez rozšírenia. V pohľade pretínajúcich tém sa často zobrazujú len odkazy na požiadavky, aby sa

ušetrilo miesto, no nie je to podmienkou. V každom prípade, celé znenie požiadavky bude potrebné minimálne v individuálnom pohľade témy.

V prípade, že prípad použitia prislúchajúci aspektovej téme rozširuje prípad použitia, ktorý prislúcha podtéme (napr. prípad použitia *autentifikácia* a prípad použitia *zruš položku* na obrázku obr. 5.5), tak sa postupuje rovnako s prihliadnutím na to, že sa vytvorená požiadavka pripojí k aspektovej téme, k aspect/based väzbe, ktorá smeruje k téme, ktorá danú podtému zoskupuje (téma *naúčtuj*).

K aspektovým témam môžu byť okrem požiadaviek, ktoré priamo súvisia s pretínaním, pripojené aj požiadavky, ktoré súvisia len s témou ako takou, t. j. nie sú zdieľané s inou témou. V súvisiacom prípade použitia sa to prejaví tak, že bude obsahovať okrem alternatívnych tokov závislých na bodových prierezoch aj toky, ktoré s bodovými prierezmi nesúvisia. Najčastejšie pôjde o základný tok. Vtedy sa bude postupovať tak, akoby sa hľadali a pripájali požiadavky k základným témam.

Pripojenie požiadaviek k základným témam

Pri definovaní požiadaviek k základným témam sa postupuje obdobne, ako pri aspektových témach, s tým rozdielom, že už nie je dôležité kedy a kde, ale iba čo robí prislúchajúci prípad použitia. Pri ich definovaní sa vychádza najmä z názvu prípadu použitia, názvov tokov a z ich textových opisov podobne ako pri aspektových témach. Môže byť definovaná jedná požiadavka pre celú tému, resp. prípad použitia alebo v prípade, že sú toky prípadu použitia dostatočne špecifické, tak môžu byť osobitne pre každý tok definované samostatné požiadavky.

Výsledné požiadavky, na rozdiel od požiadaviek pre aspektovú tému, by mali byť definované tak, aby neboli zdieľané s inou témou. Pretože tie budú pripojené k základným témam v pohľade pretínajúcich tém, kde všetky požiadavky, nesúvisiace s aspect/based väzbami, nesmú byť zdieľané.

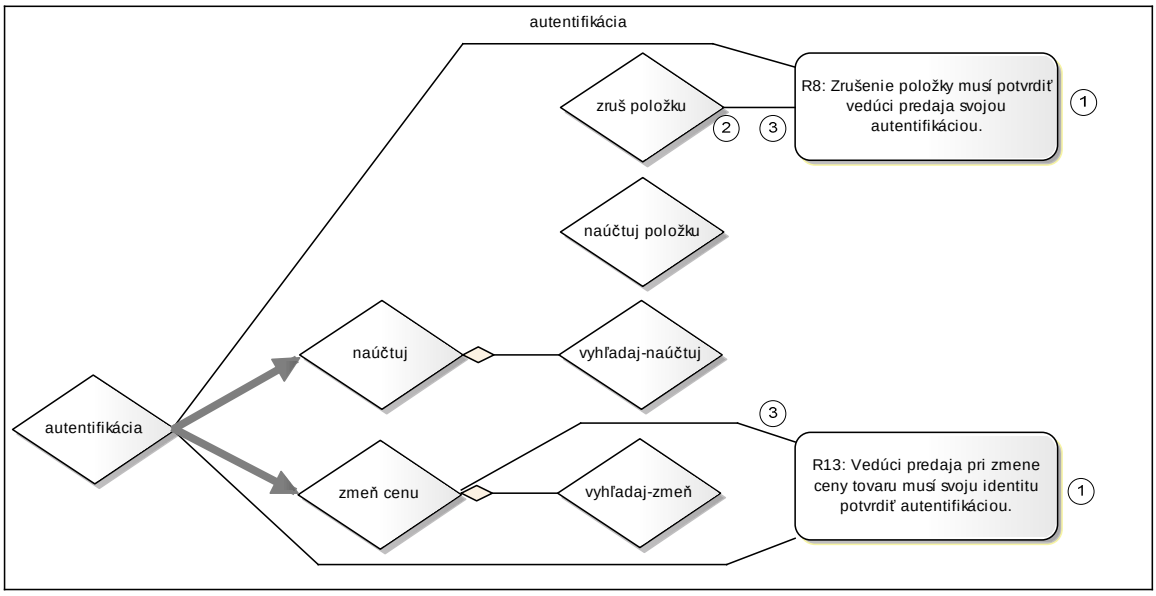
Požiadavky sa definujú aj pre podtémy, ale v pohľade pretínajúcich tém budú všetky požiadavky podtém pripojené k téme, pod ktorou sú dané podtémy zoskupené. Požiadavky, ktoré patria pod to isté zoskupenie sa môžu spojiť do jednej spoločnej požiadavky. Keďže vystupujú pod spoločným zoskupením, tak jej zdieľanie nie je problém.

Keďže pri definícii požiadavky vychádzame z prípadov použitia, tak by nemali vznikať osirotené požiadavky. Každá požiadavka súvisí s nejakým prípadom použitia, preto by zároveň mala súvisieť aj s nejakou témou. Ale problém môže nastať pri tzv. infraštruktúrnych prípadoch použitia, pomocou ktorých môžu byť zachytené aj niektoré nefunkcionálne požiadavky. Theme/Doc sa nefunkcionálnymi požiadavkami v analýze nezaoberá, ale ich odročuje do fázy návrhu. Teda ak sa z infraštruktúrneho prípadu použitia spätne vytvorí nefunckionálna požiadavka, tak nielenže ju nie je kam pripojiť, keďže infraštruktúrny prípad použitia nemá žiadnu analógiu v Theme/Doc modeli, ale sa navyše musí označiť za odročenú.

5.3.2 Individuálny pohľad tém

Doplnenie požiadaviek do individuálnych pohľadov tém už nevyžaduje žiadne transformácie modelu prípadov použitia. Stačí vychádzať z Theme/Doc pravidiel pre vytváranie individuálnych pohľadov. Jedným z nich je napríklad zobrazenie celej požiadavky, nielen odkazu (obr. 5.7, bod č. 1). Ďalej platí, že v individuálnom pohľade témy sú všetky jej podtémy nezoskupené a sú v ňom zobrazené všetky požiadavky, ktoré boli pripojené k danej téme v pohľade pretínajúcich tém, pričom požiadavky, ktoré v skutočnosti súvisia s podtémou, sú pripojené k danej podtéme a nie k téme, pod ktorou boli zoskupené v pohľade pretínajúcich tém (obr. 5.7, bod č. 2).

Požiadavky môžu byť spojené aj s témami, s ktorými neboli v pohľade pretínajúcich tém spojené, ak s danými témami majú nejaký súvis. Napríklad požiadavky aspektovej témy môžu byť v individuálnom pohľade spojené tiež so základnými témami, ktoré aspektové témy pretínajú, ak tieto požiadavky súvisia s týmto pretínaním (obr. 5.7, bod č. 3). No v celom individuálnom pohľade témy sú vždy zobrazené len požiadavky, ktoré patria danej téme.



Obr. 5.7: Pripojenie požiadaviek v individuálnom pohľade témy autentifikácia.

5.4 Entity

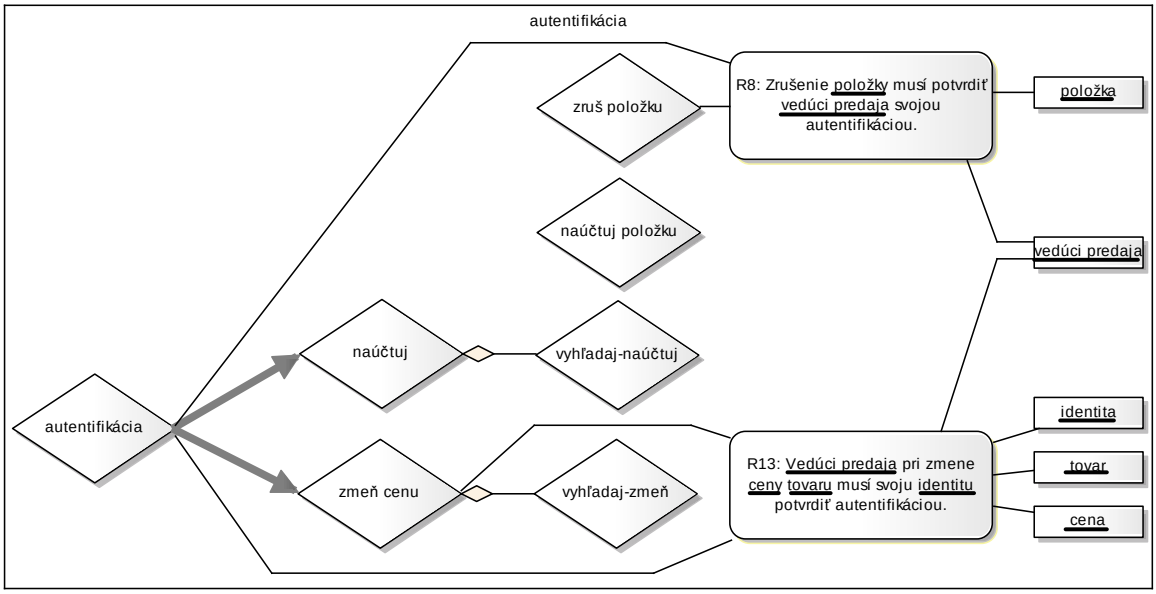
K plnohodnotnému Theme/Doc modelu je potrebné doplniť už len entity. Entity sú také kľúčové slová, ktoré majú potenciál byť objektmi budúceho systému, čiže predstavujú predbežnú štruktúru systému.

Model prípadov použitia nedisponuje žiadnymi prvkami, ktoré by určovali štruktúru budúceho systému, preto sa pri pridávaní entít bude vychádzať len z

Theme/Doc pravidiel. V Theme/Doc sa na začiatku procesu rozhodnutia o témach robí zoznam entít, kedy sa zo zoznamu požiadaviek vyberú všetky podstatné mená (postup zdola nahor). Potom sa tieto entity zakresľujú do individuálnych pohľadov tém, kde sú prepojené s požiadavkami, v ktorých sa vyskytujú.

Výhodou individuálneho pohľadu tém je, že sa v ňom nachádza aj plné znenie požiadaviek, ktoré súvisia s danou témou. To umožňuje identifikovať entity súvisiace s danou témou priamo v tomto pohľade. Stačí, ak sa zo všetkých požiadaviek, uvedených v individuálnom pohľade tém, vyčlenia všetky podstatné mená, ktoré majú predpoklad stať sa objektmi systému, do samostatného uzla (obr. 5.8, podčiarknuté slová). Tieto uzly sa následne väzbou prepoja s požiadavkami, v ktorých boli dané entity nájdené.

Je viac ako pravdepodobné, že niektoré entity sa budú nachádzať v individuálnych pohľadoch viacerých tém. Vtedy sa jedná o zdieľanie konceptu, čo je vlastne pretínanie záležitosti z hľadiska jej štruktúry. Tento jav sa taktiež rieši aspektovo-orientovaným prístupom, ale až vo fáze návrhu.



Obr. 5.8: Doplnenie entít do individuálneho pohľadu témy autentifikácia.

5.5 Granulácia

Zmena granulácie nie je povinná a bude uvedená len pre úplnosť porovnania s opačným procesom transformácie.

Theme/Doc priamo podporuje zmenu granulácie, či už sa jedná o zjemnenie alebo zhrubnutie. Používajú sa na to operácie ako zoskupovanie, zjednocovanie (pre zhrubnutie tém) a rozdeľovanie tém (pre zjemnenie granulácie), ktoré sú definované

v Theme/Doc. Tieto operácie sa vykonávajú ešte v procese rozhodnutia o témach, aby v pohľade pretínajúcich tém už bolo o granulácii rozhodnuté. Pri použití týchto operácií na konečný model Theme/Doc, je potrebné všetky pohľady znova prepracovať, keďže pri zmene granulácie požiadavky môže dôjsť aj k zmene granulácie pripojených požiadaviek a k presunom aspect/based väzieb na novovytvorené témy.

Ak má Theme/Doc model aj po zmene granulácie odzrkadľovať štruktúru modelu prípadov použitia, tak potom je potrebné pri granulácii vychádzať zo súvisiacich prípadov použitia. Ak sa bude vyžadovať jemnejšia granulácia, tak je možné postupovať tak, že sa namiesto celého prípadu použitia prevedú na témy jeho alternatívne toky. Do aspektových tém sa prevedú tie alternatívne toky, ktoré sa vo svojej definícii odvolávajú na bodový prierez rozšírenia. Ostatné alternatívne toky budú reprezentované ako základné témy. Konštrukcia požiadavky pre takéto témy bude vychádzať nie z popisu celej témy, ale z popisu daného toku.

Najhrubšie zrno granulácie v diagrame prípadov použitia je práve prípad použitia. Ten zodpovedá téme v Theme/Doc. Preto v prípade, že sa bude vyžadovať hrubšia granulácia tém, ktorá bude odzrkadľovať štruktúru prípadov použitia, tak je potrebné zmeniť granuláciu na hrubšiu najprv v samotnom modeli prípadov použitia a potom takýto model normálnym postupom transformovať na Theme/Doc model.

5.6 Zhrnutie postupu transformácie

V tejto časti budú zhrnuté kroky transformácie modelu prípadov použitia do modelu v notácii Theme/Doc:

1. Každý základný prípad použitia sa prevedie na základnú tému a každý rozširujúci na aspektovú tému v pohľade pretínajúcich záležitostí.
2. Do individuálnych pohľadov týchto tém sa doplnia príslušné podtémy zoskupené podľa pravidla úzko súvisiaceho správania, ktoré zodpovedajú konkrétnym prípadom použitia väzby generalizácie v modeli prípadov použitia.
3. Do individuálnych pohľadov týchto tém sa doplnia príslušné podtémy zoskupené podľa pravidla degradácie, ktoré zodpovedajú podtokom v modeli prípadov použitia. V prípade, že sa jedná o podtok vyčlenený v samostatnom inclusion prípade použitia, tak sa prevedená podtéma znásobí, ku každej sa do názvu pridá identifikátor, ktorý ich rozlíši a pripoja sa k súvisiacim témam, ktoré zodpovedajú prípadom použitia, ktoré tento inclusion prípad použitia zahŕňajú.
4. Individuálny pohľad aspektových tém sa doplní o témy, ktoré pretínajú, spolu s aspect/based väzbou, ktorá ich spája.
5. Ku každej téme sa pripojí požiadavka, ktorá sa spätne zrekonštruuje podľa súvisiaceho prípadu použitia, z jeho názvu, textového popisu, z názvov tokov a ich textových popisov. Je možné vytvoriť jednu požiadavku pre podľa prípadu použitia alebo ju podľa tokov prípadu použitia rozdeliť na viacero menších požiadaviek. V prípade aspektovej témy sa pri rekonštrukcii vychádza tiež z bodových prierezov, z podmienky rozširujúceho toku a zo sufixu (after, before, around) rozširujúceho toku, ktoré doplňujú požiadavku o čas a miesto, kedy

dochádza k pretínaniu. Pri aspektových témach sa rekonštruuje samostatná požiadavka pre každý rozširujúci tok. V pohľade pretínajúcich tém sa požiadavky podtém pripoja k téme, pod ktorou sú tieto podtémy zoskupené. V individuálnych pohľadoch tém sa pri pripájaní požiadaviek k témam postupuje podľa pravidiel definovaných v Theme/Doc.

6. Do individuálnych pohľadov tém sa doplnia entity, ktoré sa získajú vytiahnutím takých podstatných mien z požiadaviek nachádzajúcich sa v týchto pohľadoch, pri ktorých je vysoká pravdepodobnosť, že sa stanú objektmi systému.
7. Podľa potreby sa zmení granulácia tém. Ak pôjde o zjemnenie, tak sa za témy nebudú považovať prípady použitia, ale ich alternatívne toky. V opačnom prípade je potrebné najprv zmeniť granuláciu samotného modelu prípadov použitia a ten potom normálnym spôsobom transformovať.
8. V individuálnych pohľadoch sa vyznačia všetky uzly, ktoré sa nachádzajú vo viacerých individuálnych pohľadoch súčasne, čím sa zjednoduší odhalenie zdieľaného konceptu.

6 Porovnanie prípadov použitia a prístupu Theme/Doc

Táto kapitola sa venuje vzájomnému porovnaniu prístupu Theme/Doc a prípadov použitia. Vychádzať sa bude nielen z ich predstavenia z úvodu práce, ale tiež z poznatkov získaných pri ich vzájomnej transformácii.

6.1 Porovnanie notácií

Najviac viditeľný rozdiel medzi týmito dvomi prístupmi je možné spozorovať v ich notácií (tab. 3).

Tab. 3: Porovnanie notácie prípadov použitia a notácie Theme/Doc.

Notácia prípadov použitia	Notácia Theme/Doc
Prípady použitia	Témy
Abstraktný prípad použitia	Podtémy
Väzby include, extend, generalizácia	Požiadavky
Toky	Odročené požiadavky
Typy tokov	Aspect/based väzba
Body rozšírenia	Odročená väzba
Bodové prierezy rozšírenia	Entity
Podmienka rozširania	Tri typy diagramov (pohľadov)
Sufixy after, before, around	
Účastníci	
Textový opis	

Ako je možné vidieť z tabuľky (tab. 3), tak neexistuje ani jeden prvok, ktorý by sa nachádzal v oboch notáciách.

Ale pri hľadaní analógie medzi prípadmi použitia a témami počas transformácie sa ukázalo, že hoci prvky *témy* a *prípady použitia* majú rôzne názvy, tak ich význam je v skutočnosti rovnaký. Ako prípady použitia, tak aj témy sú vlastne črty alebo funkcionality systému, ktoré majú pre nejakého účastníka dôležitý význam.

Transformácia tiež odhalila jednoznačnú analógiu medzi aspect/based väzbou a väzbou extend.

Taktiež bola odhalená analógia medzi podtémami zoskupenými na základe úzkej súvislosti a generalizáciou v modeli prípadov použitia. Rozdiel je ale v tom, že síce je možné takéto podtémy transformovať na konkrétne prípady požitia a hlavnú tému na všeobecný prípad použitia, no model Theme/Doc neumožňuje definovať hlavnú tému ako abstraktnú.

Podobne bola odhalená analógia medzi podtémami zoskupenými na základe degradácie témy na podtému a podtokmi. S týmto tiež súvisí prekvapivé zistenie, že Theme/Doc neumožňuje znovupoužitie týchto podtém, narozdiel od prípadov použitia,

kde je možné vyčleniť takýto podtok do samostatného inclusion prípadu použitia, na ktorý sa ostatné prípady použitia odkazujú. Preto pri transformácii Theme/Doc modelu do modelu prípadov použitia je možné takéto znovupoužitie odhaliť len z dostupných plných znení požiadaviek v individuálnych pohľadoch tém. Pri opačnej transformácii je potom potrebné inclusion prípad použitia rozdeliť na samostatné podtémy.

Pri transformáciách sa v Theme/Doc objavil problém so zoskupovaním podtém, ktoré boli pretínané. Pri zoskupení sa aspect/based väzba prenesie na tému, pod ktorou je zoskupená, pričom sa tak stráca informácia o jej pretínaní. V Individuálnom pohľade, kde sú tieto podtémy opäť rozoskupené, už táto informácia chýba.

Odhalením týchto analógii je možné zmierniť rozdiely v notácii týchto modelov, no ostávajúce rozdiely, ktoré sú stále ľahko identifikovateľné.

S každým prípadom použitia je spojený aj jeho textový opis (čo sa týka prípadov použitia ako takých, nie ich vyjadrenie pomocou UML), ktorý obsahuje scenár, takže je možné pre každý prípad použitia poznať aj jeho správanie pri jeho realizácii. To umožňuje už v tejto fáze určiť, ako a kde sa budú navzájom prípady použitia ovplyvňovať, t.j. v prípade väzieb extend je možné určiť body rozšírenia. V diagrame prípadov použitia sa do samotného prípadu použitia zapisujú aj všetky toky scenára a v rozširujúcich prípadoch použitia aj bodové prierezy rozšírenia.

Čo sa týka Theme/Doc modelu, tak ten ani v jednom zo svojich pohľadov neponúka bližší pohľad na to, kde presnejšie sa budú toky aspektových a základných tém pretínať, čo samozrejme súvisí s tým, že teoreticky nepoznajú scenár. V Theme/Doc scenáre síce nie sú spísané, no pri určovaní aspect/based väzieb, musí existovať aspoň približná predstava o tom, ako sa dané témy správajú, inak by nebolo možné sa rozhodnúť, ktorá téma je základná a ktorá aspektová. No aj napriek tomu je možné konštatovať, že prípady použitia vyjadrujú správanie systému lepšie než Theme/Doc model.

Na druhej strane, vďaka existencii entít v individuálnych pohľadoch tém, je možné pomocou Theme/Doc modelu lepšie zobraziť štruktúru systému. Prístup aspektovo-orientovaného vývoja pomocou prípadov použitia sa s entitami bližšie zaoberá až v diagramoch realizácie prípadov použitia.

Viditeľným rozdielom je aj možnosť použitia tzv. odročených väzieb v Theme/Doc. Sú to bližšie nešpecifikované väzby, o ktorých sa môže rozhodnúť v neskorších fázach, keď bude k dispozícii dostatok informácií na ich rozhodnutie. Analógiou k tejto väzbe by mohla byť nejaká bližšie nešpecifikovaná závislosť, ktorá sa s obľubou používa v prípadoch použitia (najčastejšie sa označuje obyčajnou asociáciou alebo závislosťou bez určeného smeru). Pri aspektovo-orientovanom vývoji pomocou prípadov použitia sú ale takéto bližšie nešpecifikované väzby zakázané.

Ďalším viditeľným rozdielom je to, že Theme/Doc pracuje vo svojich diagramoch, resp. pohľadoch aj so samotnými požiadavkami. Tie sú dôležité pri určovaní pretínajúcich tém.

Na druhú stranu, model prípadov použitia, si zo zoznamu požiadaviek vyberá účastníkov a priraďuje ich k samotným prípadom použitia, ktoré sú pre daného účastníka dôležité. Nedá sa povedať, že pri aspektovo-orientovanom vývoji softvéru pomocou prípadov použitia už ďalej nebudú potrebné požiadavky zo zoznamu

požiadaviek, ale samotné prípady použitia ich v tejto fáze nepotrebujú. Tie sa môžu zísť vo fáze realizácie prípadov použitia, kedy entity, ktoré kolaborujú pri realizácii prípadu použitia, môžu byť identifikované v scenári prípadu použitia, v diagrame domény problému alebo v samostatných požiadavkách. Theme/Doc si tieto požiadavky prenáša prostredníctvom pohľadov až do procesu plánovania návrhu.

Celkovo sa dá povedať, že čo sa týka notácie, tak model prípadov použitia disponuje širším vyjadrovacím aparátom než Theme/Doc model. Dôvod tohto rozdielu treba hľadať najmä v tom, že prípady použitia sa často používajú aj ako komunikačný prostriedok medzi analytikmi, resp. konzultantmi a ostatnými účastníkmi. Preto musí navyše disponovať vyjadrovacím aparátom aj pre túto stránku použitia diagramov prípadov použitia.

6.2 Iné rozdiely a podobnosti

Rozdiely je možné vidieť aj v miestach, ktoré nesúvisia s notáciou, ako je napríklad inžiniering požiadaviek (Requirements Engineering).

Prípady použitia sa inžinieringom požiadaviek nezaoberajú a určenie záležitostí, ktoré budú modelované pomocou prípadov použitia, nechávajú na intuícii analytika. Navyše vytvorený model prípadov použitia neuchováva žiadne referencie na požiadavky, z ktorých vznikol. Preto je možné vytvoriť model prípadov požiadaviek aj bez spísaných požiadaviek, napríklad pri ústnom pohovore. Stačí, ak analytik správne zachytí všetky podstatné záležitosti.

Súčasťou prístupu Theme/Doc je aj inžiniering požiadaviek, ktorého cieľom je v zozname požiadaviek správne identifikovať témy a základné entity (proces rozhodnutia o témach). Tie sú v Theme/Doc modeli priamo prepojené s požiadavkami, s ktorými súvisia a preto je tento model závislý na zozname požiadaviek. Nie je preto možné použiť Theme/Doc prístup, pokiaľ nie je k dispozícii zoznam požiadaviek. Nie je ale dôležité, aby boli požiadavky v tomto zozname nejako správne štruktúrované. Inžiniering požiadaviek v Theme/Doc umožňuje vychádzať z jednotlivých slov a definuje kroky, ktorými je možné upraviť požiadavky do vhodnejšej podoby, tak aby sa minimalizovalo pretínanie. Navyše je možné tento inžiniering požiadaviek čiastočne automatizovať.

Ďalším takýmto zásadným rozdielom je v prípade prípadov použitia aj možnosť modelovania niektorých nefunkcionálnych požiadaviek pomocou tzv. infraštruktúrnych prípadov použitia. V Theme/Doc prístupe sa takéto požiadavky odročujú do neskorších fáz.

Ak sa témy identifikujú nezávislé od prípadov použitia, tak pohľad tém a vzťahov má obyčajne jemnejšiu granuláciu než diagram prípadov použitia, t.j. Theme/Doc obsahuje viac tém s menším záberom a diagram prípadov použitia obsahuje menej prípadov použitia, no s väčším záberom. Granulácia závisí od viacerých faktorov a nikde nie je stanovené, aká granulácia je ta správna. Všetko závisí od veľkosti projektu, zvoleného prístupu analýzy a taktiež od zvyklostí analytika. Keďže Theme/Doc umožňuje použiť nástroje, ktoré automatizujú identifikáciu prvotných akcií na základe slovies v požiadavkách, tak je zrejmé, že v Theme/Doc sa najviac využíva práve postup zdola nahor, t.j. identifikované akcie sa potom zjednocujú

a zoskupujú, až kým sa nedosiahne požadovaná granulácia. Preto majú témy tendenciu byť jemnejšej granulácie. V modeli prípadov použitia sa prípady použitia identifikujú priamo, podľa subjektívnych pocitov analytika tak, aby bol čitateľný pre všetkých zainteresovaných. Málokedy sa stane, že by oba modely skončili na rovnakej úrovni granulácie, no v prípade analýzy to nie je až také dôležité. Dôležitejšie je, aby tam nič podstatné nechýbalo bez ohľadu na to, či je to jasne viditeľné v diagrame alebo je to skryté niekde v scenári. Rozhodnutie o konečnej granulácii záležitostí v programe je aj tak v rukách návrhára.

Nemalú rolu pri výbere prístupu bude zohrávať aj softvérová podpora týchto prístupov. Prípady použitia sa už dlho tešia nielen softvérovej, ale z časti aj UML podpore. Prvky, ktoré pridal I. Jacobson síce nie sú priamo zahrnuté v UML špecifikácii, ale je možné si ich podľa UML špecifikácie dodefinovať. Na druhú stranu použitie use case slice techniky v ďalších fázach nie je podporované nielen UML nástrojmi, ale je problém jej netypické konštrukcie zakresliť aj pomocou všeobecných nástrojov na kreslenie diagramov.

V prípade Theme/Doc je situácia opačná. Hoci jediná netypická vec sa zdá byť použitie kosoštvorcov na modelovanie tém, tak je to práve to, čo robí najväčšie problémy. Je problém nájsť nástroj, ktorý umožňuje zakresľovať kosoštvorce spolu so všetkými typmi väzieb, s ktorými Theme/Doc pracuje. Je preto potrebné hľadať také UML nástroje, kde je možné si definovať vlastné tvary. Taktiež ešte stále nie je k dispozícii už dlho avizovaný nástroj na automatický inžiniering požiadaviek, ako ho definuje Theme/Doc. Naopak, vo fáze návrhu sa používa Theme/UML technika, ktorá používa iba prvky podporované UML štandardom.

6.3 Témy a prípady použitia v praxi

Momentálne je možné sa častejšie stretnúť s témami, než s prípadmi použitia v aspektovo-orientovanej analýze. Témy sa s obľubou objavujú v odborných článkoch, ktoré sa venujú riešeniu pretínajúcich záležitostí v špecifických oblastiach (napr. webové služby). Je to hlavne kvôli jednoduchšej notácii a jednoduchšiemu použitiu oproti prípadom použitia. Pretože v oblastiach, kde sa aplikovanie aspekto-orientovaného vývoja ešte len analyzuje, je hlavným cieľom zachytenie a jednoduché modelovanie pretínajúcich záležitostí, pričom sa nedbá na detaily, ako sú napr. bodové prierezy rozšírenia, body spájania, toky alebo textový opis. Často sa potom stáva, že v prípade potreby detailnejšieho opisu pretínania sa neprechádza na prípady použitia, ale sa upraví prístup založený na témach, ako je napr. doplnenie väzby generalizácie (Zhang, et al., 2009). Treba si ale uvedomiť, že takéto modely nie sú vhodné pre komunikáciu medzi zainteresovanými osobami alebo ako súčasť zmluvy, pretože neobsahujú dostatok informácií.

Výhodou prípadov použitia je ich všeobecné akceptovanie aj medzi zákazníkmi a neraz sa stávajú predmetom zmluvy a následného testovania výsledného produktu. No niektorí odborníci považujú modelovanie pretínajúcich záležitostí pomocou prípadov použitia, tak ako to definoval I. Jacobson, za nie je príliš vhodný spôsob. Poukazuje sa na to, že mechanizmus rozširovania pomocou rozširujúceho prípadu použitia môže byť použitý aj v prípade, ak sa nejedná o pretínanie, napr. ak rozširujúci

prípad použitia reprezentuje komplexný alternatívny priebeh jedného špecifického základného prípadu použitia. Taktiež sa poukazuje na to, že podľa UML špecifikácie musia byť možné body rozšírenia definované v rozširovanom prípade použitia a v rozširujúcom prípade použitia by mali byť referencie na tieto body rozšírenia. Keďže pretínanie by malo nastať, bez toho aby pretínajúca záležitosť bola nejakým spôsobom ovplyvnená, tak potom by sa ani nemalo stávať, že je potrebné definovať body rozšírenia v pretínajúcej záležitosti. Miesto pretínania by sa malo definovať len rozširujúcom prípade použitia.

Keďže podľa UML špecifikácie sa pri vytvorení extend závislosti vytvoria aj body rozšírenia v rozširovanom prípade použitia, tak jediným riešením je vytvorenie nového stereotypu závislosti *crosscuts* a vytvorenie tzv. *crosscutting* prípadu použitia, ktorý obsahuje pretínajúce správanie. Pri použití tejto závislosti a prípadu použitia sa miesta pretínania definujú iba v crosscutting prípade použitia alebo sa definujú mimo modelu prípadov použitia (Sousa, et al., 2004). V prípade odborných článkov sa dáva prednosť práve tomuto spôsobu aspektovo-orientovanej analýzy pomocou prípadov použitia.

Aj napriek spomenutým nedostatkom, prístupy Theme a aspektovo-orientovaný vývoj pomocou prípadov použitia sú tie, ktoré poskytujú najkompletnejšie riešenie AOSD záležitostí.

7 Záver

V tejto práci bolo predstavené AOP a dva AOSD prístupy vo fáze analýzy. Prvým je prístup založený na prípadoch použitia a druhým je prístup Theme/Doc, ktorý je založený na témach. Už pri ich opise boli postrehnuté niektoré podobné, ale i rozdielne črty a tiež analógie. Preto sa pristúpilo k navrhnutiu postupu transformácie z modelu prípadov použitia do modelu v notácii Theme/Doc a naopak, s cieľom podrobne preskúmať tieto črty a analógie, prípadne objaviť nové a na základe poznatkov z týchto transformácií podrobne analyzovať jednotlivé prístupy.

Jedným z najdôležitejších výsledkov týchto transformácií bolo ukázanie, že väzba extend medzi prípadmi použitia je analógiou k aspect/based väzbe medzi témami, z čoho ďalej vyplynulo, že základné a aspektové témy zodpovedajú základným a rozširujúcim prípadom použitia a naopak. Taktiež boli objavené analógie medzi zoskupovaním podtém na základe úzkej súvislosti a generalizáciou v modeli prípadov použitia a tiež medzi zoskupovaním na základe degradácie témy na podtému a podtokmi v modeli prípadov použitia. Prekvapením bolo zistenie, že pri dodržiavaní pravidiel ako ich definuje Theme/Doc, nie je možné znovupoužitie podtém.

Hlavné odlišnosti týchto prístupov sú jasne viditeľné v notácii. V prvom rade je to notácia pohľadov v Theme/Doc, ktorá nepozná účastníkov. V Theme/Doc nie sú scenáre, ani toky a ani body rozšírenia, resp. bodové prierezy rozšírenia. V modeli prípadov použitia, na rozdiel od prístupu Theme/Doc, zase nie sú k dispozícii entity a požiadavky, s ktorými daný prípad použitia súvisí. Vďaka prítomnosti scenára , model

prípadov použitia lepšie zobrazuje správanie systému. Naopak, vďaka prítomnosti entít, model v notácii Theme/Doc podáva lepší prehľad o štruktúre systému.

Výhodou prípadov použitia je ich všeobecné akceptovanie aj medzi zákazníkmi a neraz sa stávajú predmetom zmluvy a následného testovania výsledného produktu. Prípady použitia sú súčasťou UML špecifikácie, preto majú značnú podporu v CASE nástrojoch. Na druhú stranu, prípady použitia sú náročnejšie na správne použitie, ktoré vyžaduje skúsenosti.

Hlavnou výhodou Theme/Doc je možnosť automatizovať niektoré kroky inžinieringu požiadaviek (Requirements Engineering), hoci dopredu ohlásený oficiálny nástroj na prácu s Theme/Doc ešte stále nie je k dispozícii. Taktiež je výhodou jednoduchšia notácia a samotné použitie. Nevýhodou je užší vyjadrovací aparát, ktorý znemožňuje použitie modelov tohto prístupu na komunikáciu so zákazníkom. Jedná sa hlavne o chýbajúci scenár.

Aj napriek tomu, že tieto prístupy poskytujú momentálne najlepšiu podporu aspektovo-orientovanému vývoju softvéru, tak kvôli ich niektorým nedostatkom je možné často natrafiť na ich modifikácie.

V ďalšej práci by bolo vhodné pokračovať v analýze týchto prístupov vo fáze návrhu a navrhnúť transformáciu use case slice prístupu do Theme/UML prístupu a naopak. Výhodné by tiež bolo pokúsiť sa o priamy prechod z aspektovo-orientovanej analýzy pomocou prípadov použitia do aspektovo-orientovaného návrhu pomocu Theme/UML prístupu. Zabezpečila by sa tak UML podpora či už pre aspektovo-orientovanú analýzu alebo aspektovo-orientovaný návrh.

Použitá literatúra

1. Clarke, S., Baniassad, E.: Aspect-Oriented Analysis and Design: The Theme Approach. Addison Wesley Professional, marec 2005. 400 s. ISBN 0-321-24674-8.
2. Jacobson, I., Ng, P.: Aspect-Oriented Software Development with Use Cases. Addison Wesley Professional, december 2004. 464 s. ISBN 0-321-26888-1.
3. Jacobson, I.: Use Cases and Aspects – Working Seamlessly Together. In: *Journal of Object Technology*, vol. 2, júl-august 2003, no. 4, pp. 7-28. <http://www.jot.fm/issues/issue_2003_07/column1> (13.5.2009)
4. Övergaard, G., Palmkvist, K.: Mistake: Functional Decomposition. In: *Use Cases Patterns and Blueprints*. Addison Wesley Professional, november 2004.
5. Pawlak, R., Younessy, H.: On Getting Use Cases and Aspects to Work Together. *In: Journal of Object Technology*, vol. 3, január-február 2004, no. 1, pp. 15-26. <http://www.jot.fm/issues/issue_2004_01/column2> (13.5.2009)
6. Rosenberg, D., Stephens, M.: Use Case Modeling. *In: Use Case Driven Object Modeling with UML: Theory and Practice*. Apress, 2007, s. 49-82.
7. Sousa, R., Soares, S., Borba, P., Castro, J.: Separation of crosscutting concerns from requirements to design: Adapting the use case driven approach. In: *Proc. Early Aspects Workshop at AOSD 2004*, 2004. <http://trese.cs.utwente.nl/workshops/early-aspects-2004/Papers/SousaEtAl.pdf> (13.5.2009)
8. Zhang, Y., Yang, H., Chen, J., Meng, X.: Themes4BPEL: An efficient aspect-oriented web service composition design approach. In: *Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on*, vol. 1, február 2009. <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4810011&isnumber=4809880> (13.5.2009)

A.Elektronický nosič – DVD ROM

Priložený DVD nosič obsahuje dva súbory typu *pdf*. Jeden s názvom *text.pdf*, ktorý obsahuje elektronickú verziu tohto dokumentu a druhý s názvom *priloha.pdf*, ktorý obsahuje prílohu článku. Tieto súbory sa nachádzajú v koreňovom adresári.

B.Transforming Aspect-Oriented Analysis in Theme/Doc into Use Case

Na základe výsledkov tejto diplomovej práce bol napísaný nasledujúci článok:

Pavol Michalco: Transforming Aspect-Oriented Analysis in Theme/Doc into Use Case. In.: *IIT.SRC 2009: Student Research Conference*. Nakladateľstvo STU, 2009.

Transforming Aspect-Oriented Analysis in Theme/Doc into Use Cases

Pavol MICHALCO*

Slovak University of Technology
Faculty of Informatics and Information Technologies
Ilkovičova 3, 842 16 Bratislava, Slovakia
pavol.michalco@gmail.com

Abstract. This paper is concerned with the aspect-oriented software development in the analysis phase. Two approaches to aspect-oriented analysis are briefly introduced – aspect-oriented software development with use cases and Theme/Doc approach. These two approaches are analyzed to prove that base and aspect themes in the Theme/Doc correspond to extended and extension use cases and vice versa. Based on this analysis, a process of transformation from Theme/Doc into use cases is proposed.

1 Introduction

Dividing program into loosely coupled modules is rather difficult. The reasons are called crosscutting concerns. It is not possible to divide these concerns into separate modules in object-oriented programming, so tangled and scattered code occurs.

The solution at the implementation level is a new approach to the programming called aspect-oriented programming (AOP). New constructions have been introduced into programming languages that have made possible to insert behavior from another module into the module in operation and express required code without tangled or scattered code. Such constructions are aspects, advices, pointcuts, join points and others that allow to specify crosscutting at a higher level of abstraction.

Moving AOP principles to the earlier phase development software (analysis, design) is referred to as aspect-oriented development software (AOSD). There are several AOSD approaches, but currently the most comprehensive (in the sense of AOP

* Master degree study programme in field: Software Engineering
Supervisor: Dr. Valentino Vranić, Institute of Informatics and Software Engineering, Faculty of Informatics and Information Technologies STU in Bratislava

IIT.SRC 2009, Bratislava, April 29, 2009, pp. 1–8.

issues support in software analysis and design) approaches are considered to be aspect-oriented software development with use cases (AOSD/UC) and Theme.

In this paper we will focus on the analysis phase, which in the case of AOSD/UC means use cases and in the case of Theme means Theme/Doc tool. The basic goal of the analysis in AOSD is to discover so-called early crosscutting concerns in requirements, and their modularization and subsequent visual modeling. Already a cursory comparison of these approaches reveals a parallel between use cases and Theme/Doc, therefore in this paper we will analyze the relationship between these approaches and the possibility of transformation of a Theme/Doc model into a use case model.

2 Aspect-oriented software development with use cases

Ivar Jacobson was first to point out that there is a link between use cases and aspect-oriented programming [2]. Concerns as such are represented by use cases, while crosscutting concerns are represented by extension use cases. But for better support of aspect-oriented programming Ivar Jacobson has had to extend the UML notation of use cases with new features – extension pointcuts and types of flows (see Figure 1).

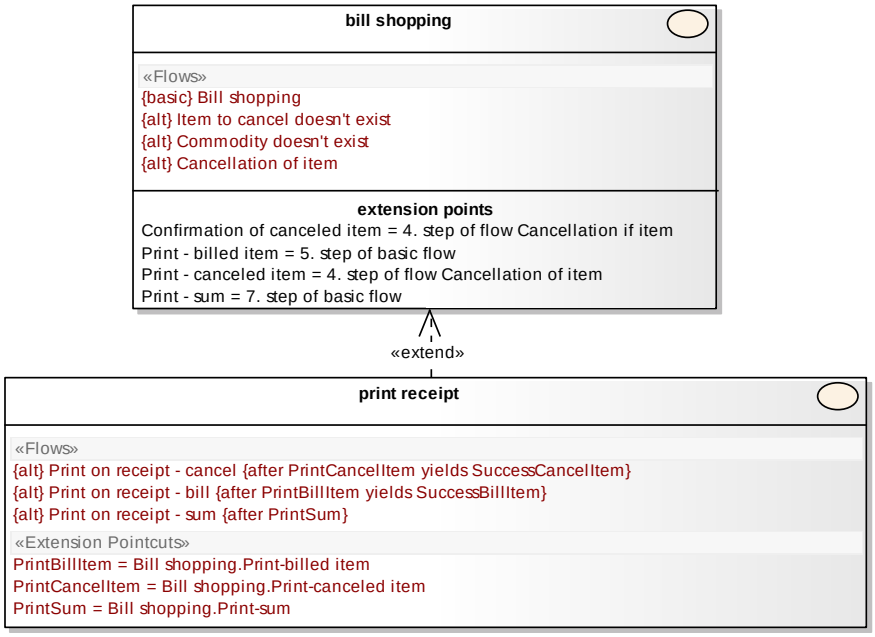


Figure 1. Use case diagram in compliance with I. Jacobson's definition of use case.

3 Theme/Doc

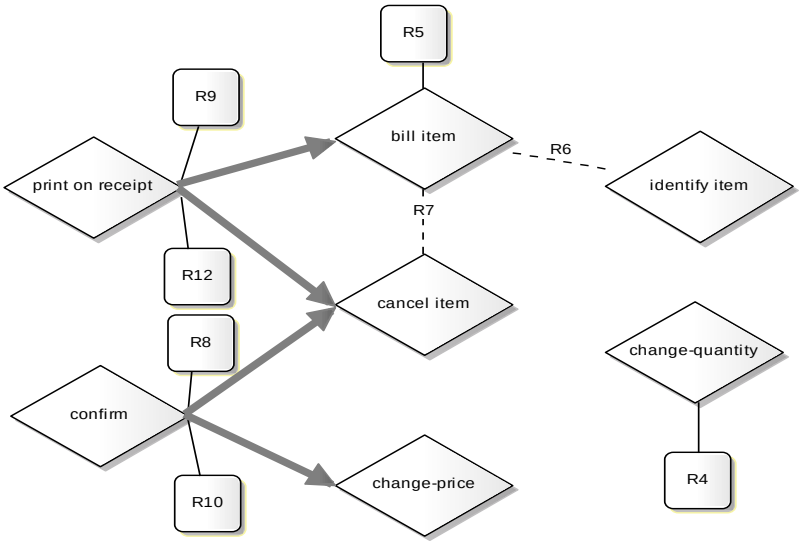


Figure 2. Crosscutting theme view in the process of determining theme responsibilities.

In Theme/Doc there are three processes, whose aim is to identify and model concerns, which will be later used in the design (using the procedures and tools called Theme/UML):

1. Deciding the themes – identifying potential themes and entities and mapping requirements to concerns in the system
2. Determining theme responsibilities – determine aspect-based relationship
3. Design planning – individual theme preparation for design

Each process also has its own view, i.e. a particular diagram type.

Concerns are represented using so-called themes, which are divided into base and aspect themes. Aspect themes correspond to crosscutting concerns that are to be realized aspects. In the crosscutting theme view (see Figure 2) themes are displayed as diamond, which are linked with requirements (squares with description *R#*) that relate to the theme. Aspect-based relationships between the base and aspect themes are marked with thick arrows.

4 Use cases and themes

Even at a simple glance of use case diagrams and views in Theme/Doc of the same system, we can see some similarities between themes and use cases and between some relationships among them. More light on these similarities will be shed if we look better at how to actually identify the themes and use cases.

In use case identification we start from specified requirements and we are looking for the functionality important for stakeholders.

The identification of themes can proceed in three ways [1]:

1. Identification of actions (verbs) in the requirements and subsequent splitting, unifying, grouping, deleting actions and requirements to obtain key functionality
2. By direct identification of the key functionality in the requirements declared as a theme
3. From the use case model

From comparison of the process of identification of use cases and identification of themes, we see that use cases represent the same as themes, i.e. some key functionality.

In themes we also have to decide which themes are base and which one aspect. Similarly, in use cases we have to decide which of use cases are base, extension, including (include inclusion use case), inclusion and so on. How to decide which themes are base and which one aspect is determined by four conditions [1].

1. The requirement cannot be split to untangle the themes.
2. One theme dominates the requirement – a theme that a requirement is most "about" (or is dominated by) is the one that may be the aspect.
3. The base triggers aspect – if two themes are mentioned in a requirement, the one that is being triggered is the aspect, and the one responsible for the triggering is the base.
4. The dominant theme is triggered in multiple situations.

The first point is simply that we first try to untangle themes by splitting the request in which these tanglings occur. If this is not possible, then we have to decide whether the theme is a base or aspect theme. And this is achieved by points 2 and 3. The most important point is that we work with characteristics that are typical for the relationships between extended and extension use cases:

- The dominant (aspect) theme is the one that must have the information about the requirement's behavior (and about the other non-dominant themes related to this requirement) to know when to run (in the diagram the arrow is directed from the aspect theme to the base theme) – the extension use case must be aware of the extended use case to know when to run, i.e. the extension use case depends on the related extended use case (in the diagram the arrow is directed from extension use case to the extended use case).
- An aspect theme is activated by the base theme's behavior – an extension use case is activated by the behavior of the extended use case.
- A base theme is independent of the aspect theme – an extended use case is independent of the extension use case.

This shows that the decision process for aspect and base themes is the same as the decision process for extension and extended use cases. From the fact that the identification of themes is identical to the identification of use cases, it follows that base and aspect themes correspond to extended and extension use cases and vice versa.

5 The transformation

Based on previous sections, the transformation process of Theme/Doc into use cases can be proposed. Since base and aspect themes correspond to extended and extension use cases, respectively, we can declare all base themes in the crosscutting theme view as base use cases and their related aspect themes as extension use cases. We will get a model which will include base and extension use cases. That means that only extend dependency will be in the use case diagram, and no include nor generalization relationships. It is necessary to precisely specify the rest types of relationships between use cases. Only stereotyped dependencies (include, extend) not a general one is allowed. The model will also miss some elements such as flows, extension points, pointcuts and others. Thus the transformation process consists of several steps to get full-blown use case diagram:

1. Transform base themes into base use cases and aspect themes into extension use cases and mark extend dependencies
2. Factor out include dependencies and create related flows
3. Factor out generalization and create related flows
4. Create extension pointcuts, which reflect crosscutting in requirements that associate with aspect theme
5. For each extension pointcuts create related extension points without references to the scenario
6. Create a scenario, create the rest of flows based on this scenario and set extension points' references to this scenario

In the following sections we will take a closer look especially at points 2 and 3.

5.1 Include dependency

The include dependency connects inclusion use cases, which contain subflows, with use cases that reuse these subflows in their own flows. A subflow is analogy to the subtheme in the Theme/Doc. If we want to capture a decision to demote behavior from its own theme to being a method, then we can use subthemes, which are grouped under one larger theme. This means that the theme that behaves toward other theme like its method, can be grouped together under this other theme and in the theme-relationship view and crosscutting theme view is displayed only one large theme. The grouping occurs in the process of deciding in the themes and determining theme responsibilities and then we work only with themes and larger themes, under which subthemes are grouped together.

Subthemes will be not visible until we come to the design planning process. In this process we are always using individual theme view where all subthemes are again displayed as themes and aggregated into a larger theme, under which these subthemes have been grouped together. Thus in the process of transformation into use case model we will use individual theme views to search for subthemes. Then large themes can be transformed into use cases (see Figure 3, edges number 1) and subthemes can be transformed into subflows of such use cases.

We can factor out these subflows into own inclusion use cases, if they are multiple used by other use cases. This would mean that a subtheme that is grouped under several themes could be transformed into an inclusion use case. But Theme/Doc does not support subthemes reusing, so one subtheme cannot be subtheme of several themes, which use the same behavior. Equal behavior of such subthemes can be identified in related requirements, which are available in the individual theme view. In Figure 3 we can see such subthemes. Despite different names, the behavior of these subthemes is identical (underlined in the requirements). Accordingly, these subthemes can be transformed into the inclusion use case (see Figure 3, edge number 2).

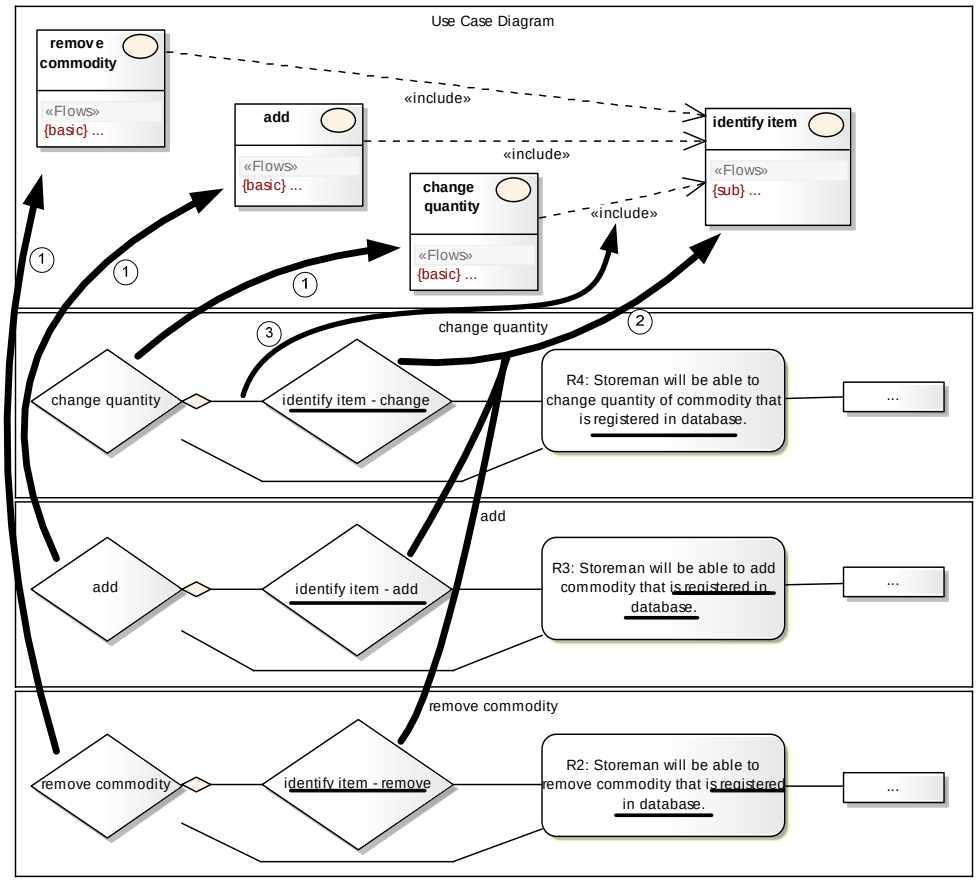


Figure 3. Analogy between the grouping of themes and include relationship in the use case model.

5.2 Generalization relationship

Use case generalization expresses relationship between use cases with similar behavior, which is generalized by the one parent use case. A parent use case is usually

abstract, which means it has some abstract flows, which must be specified by flows in linked child use cases. Similarly to the include relationship, Theme/Doc does not directly support generalization relationship, so at first sight we have nothing, from which generalization links could be obtained to draw into use case diagram.

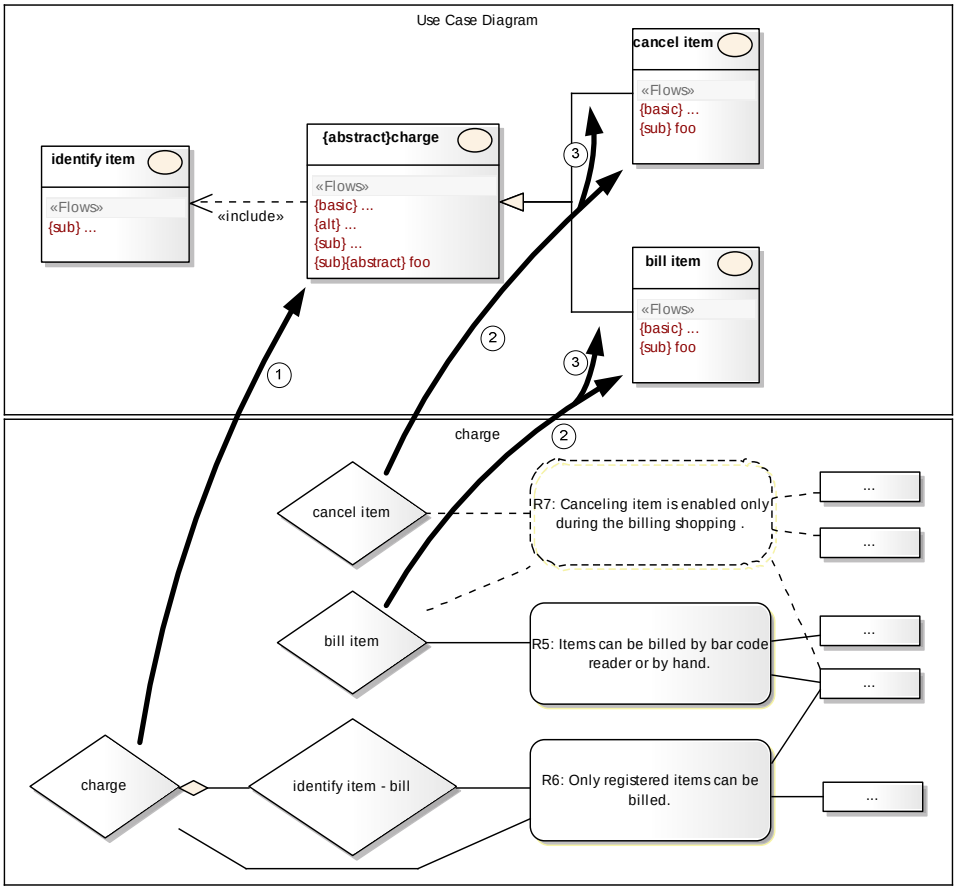


Figure 4. Analogy between the grouping of themes and generalization relationship in the use case model.

In the previous subhead we have mentioned grouping themes on basis of demotion from theme to method. But this is not the only reason for grouping. The grouping may also occur on the basis of closely related behavior. Such grouping actually corresponds to the generalization in the use case model. Closely related themes are grouped under one common general theme. Thus in the process of transformation into the use case model we must check individual theme views and all themes that have been grouped on the basis of closely related behavior transform into concrete use cases (see Figure 4,

edges numbers 2 and 3). And themes under which such themes have been grouped transform into general use cases (see Figure 4, edge number 1).

5.3 Other elements

After transformation of themes into use cases and the adding relationships, we are still missing elements such as flows and their types, extension points, extension pointcuts and actors in use case model.

Some flows were inserted into use cases while we were obtaining relationships, but certainly not all flows. We are missing some basic and alternative flows that derive from the scenario, which is not supported by Theme/Doc. Similarly for extension pointcuts, where for each requirement that associates with aspect theme (i.e. crosscutting occurs there) we can insert extension pointcut into related use case, which corresponds to this aspect theme. However, this way avoids the possibility of reusing extension pointcuts. The biggest problem is extension point. We could for each extension pointcut insert related extension point into the diagram, but we wouldn't know where these extension points refer to because extension points refer directly to the position in the scenario. Theme/Doc also doesn't support actors. Overall, we can say that the use case model has a broader expression apparatus than Theme/Doc.

6 Conclusions and further work

In this article we have shown that base and aspect themes correspond to extended and exclusion use cases and vice versa. We have presented possibilities of transformation of the model in the Theme/Doc notation into the use case model, but we believe that it is unable to create a full-blown use case diagram from model in the Theme/Doc notation. It is mainly due to the fact that the use case modeling has a broader expression apparatus than Theme/Doc and that use cases are based on text description.

The top advantage of use case modeling is use case diagram, which is understandable for most stakeholders and also the possibility to apply the use case driven development issues, for example use case driven testing. But the Theme approach has a better support of aspect-oriented design using Theme/UML tool. Thus further work will be devoted to propose a process of transformation from use cases into Theme/Doc.

Acknowledgement: This work was partially supported by the Science and Technology Assistance Agency under the contract No. VG1/0508/09.

References

[1] Clarke, S., Baniassad, E.: Aspect-Oriented Analysis and Design: The Theme Approach. Addison Wesley Professional (2005), ISBN 0-321-24674-8.
[2] Jacobson, I., Ng, P.: Aspect-Oriented Software Development with Use Cases. Addison Wesley Professional (2004), ISBN 0-321-26888-1.