Uporaba metodologije TAD in UML za razvoj aplikacije TRANZIT

Nadja Damij
Šišenska 27, 1000 Ljubljana
nadja.damij@uni-lj.si

Izvleček

Članek obravnava uporabo dveh objektno-usmerjenih metodologij. To sta TAD in UML. Metodologija TAD je primerna za uporabo na področjih prenove poslovnih procesov in razvoja informacijskih sistemov. Značilnost te metodologije je v tem, da za razvoj informacijskega sistema uporablja različne tabele. UML je znan kot zbirka osmih diagramov, s katerimi si pomagamo pri načrtovanju objektno-orientiranih projektov. Za implementacijo korakov obeh metodologij obravnavamo delovni proces Tranzit, ki je le eden izmed delovnih procesov, ki sestavljajo poslovni proces Trgovanje. Na koncu je podana tudi primerjava med omenjenima metodologijama.

Abstract

The aim of the article is to represent the use of two object-oriented methodologies: TAD and UML. TAD metodology is suitable for use in the field of business process reengineering and information system development. This metodology uses several specific tables to develop the information system. UML is known as a set of eight diagrams which could be used to design object-oriented projects. This article illustrates the use of both metodologies in developing a transit information system. The article also shows a comparison between both used metodologies.

1 Uvod

Članek obravnava uporabo objektno-orientiranih metodologij TAD in UML za razvoj informacijskega sistema. Za implementacijo korakov obeh metodologij obravnavamo problem tranzitnega poslovanja. Tranzit pomeni, da podjetje blaga ne skladišči, ampak dobavitelj blago dostavi direktno kupcu, ki ga je naročil. Kupec v podjetju naroči potrebno blago, kjer komercialist zapiše naročilo. Ker v podjetju želenega blaga na zalogi nimajo, kontaktirajo ustreznega dobavitelja. Tranzit se začne, ko kupec pošlje zahtevek za predračun komercialistu v podjetju, ki formira predračun in ga pošlje kupcu. Kupec predračun plača v celoti ali pa samo delno. Komercialist nato zaključi predračun in na podlagi zahtevka za naročilo formira naročilo kupca (vnese podatke v računalnik). Iz teh podatkov se formira naročilo dobavitelju, ki pošlje blago kupcu in račun podjetju. Komercialist kontrolira prejeti račun z naročilom dobavitelju. V primeru, da se prejeti račun ne ujema z naročilom, skupaj z dobaviteljem reši ugotovljene nepravilnosti, nato pa preda račun v oddelek Komercialne dokumentacije. Komercialist tudi napravi račun in ga pošlje kupcu. Na podlagi tega računa referent v oddelku Saldokonti sprejme kupčevo plačilo.

2 Metodologija TAD

Metodologija Tabular Application Development (TAD) je objektno orientirana metodologija, ki se uporablja predvsem na področju prenove poslovnih procesov in razvoja informacijskih sistemov. Metodologija predstavlja realni svet z uporabo različnih tabel.

Metodologija TAD je sestavljena iz šestih faz. Prva faza je definiranje problema, v okviru katere se kreira tabela entitet. Ta tabela prikazuje predvsem vitalne analize, ki jih zahteva management na različnih nivojih organizacije. V drugi fazi opišemo funkcioniranje sistema, na podlagi katerega razvijemo tabelo aktivnosti in tabelo nalog. Tretja faza je prenova poslovnih procesov, kjer se analizirajo obstoječi strateški, poslovni in operativni cilji ter se iščejo možnosti za uvedbo novih ciljev oziroma spreminjanje obstoječih. V četrti fazi se razvije objektni model. Sistem načrtujemo v peti fazi, rezultat katere je aplikacijski model. Zadnja faza pa se ukvarja z implementacijo sistema.

Definiranje problema

Definiranje problema se začne z intervjuji managementa, tako na strateškem, kot tudi poslovnem in operativnem nivoju. Rezultati intervjujev managementa so:

- definiranje strateških ciljev podjetja,
- določitev izhodov, ki so povezani z definiranimi strateškimi cilji podjetja,
- definiranje analiz za podporo odločanja,
- spoznavanje organizacijske strukture organizacije,
- definiranje poslovnih ciljev,
- definiranje operativnih ciljev.

TAD uporablja termin entiteta, ki pomeni *uporabnik, skupina uporabnikov* ali *izvor informacij*. Ločiti moramo med internimi in eksternimi entitetami. Interne entitete so tiste, ki so znotraj sistema. Eksterne entitete pa so tiste, ki niso del sistema, so pa s sistemom povezane. Za definiranje problema metodologija TAD uporablja tako imenovano tabelo entitet. Tabelo entitet razvijemo na podlagi rezultatov intervjujev z managementom. V stolpcih tabele entitet prikažemo interne entitete, v vrsticah pa prikažemo analize, ki so povezane s strateškimi, poslovnimi in operativnimi cilji. Neprazno križišče v tabeli entitet prikazuje, katera entiteta potrebuje določeno analizo. V skladu s prvo fazo TAD metodologije sem začela z organizacijo intervjujev z uporabniki sistema. Kot rezultat intervjujev sem definirala tabelo entitet (Tabela 1), ki prikazuje štiri entitete in analize, ki so pomembne na področju tranzita.

2.2 Funkcioniranje sistema

Ta faza je sestavljena je iz treh korakov: identifikacija aktivnosti, definiranje nalog ter identifikacija delovnih in poslovnih procesov.

Aktivnosti

Za identifikacijo aktivnosti, ki jih uporabniki izvajajo pri svojem delu, je potrebna nadaljnja organizacija intervjujev z vsemi uporabniki sistema. V ta namen metodologija TAD razvije posebno tabelo, ki se imenuje tabela aktivnosti. Tabela je strukturirana tako, da so v stolpcih predstavljene entitete, medtem ko so aktivnosti zapisane v vrsticah. Istočasno s tabelo aktivnosti razvijemo tudi tabelo nalog, ki podrobno opisuje posamezno aktivnost.

Neprazno križišče (i,j) v tabeli aktivnosti pomeni, da entiteta definirana v stolpcu j opravlja določeno nalogo znotraj aktivnosti i, kjer je i=1,...,št.aktivnosti in j=1,...,št.entitet. Posamezna aktivnost lahko vsebuje eno ali več nalog, ki jih lahko opravlja ena ali več entitet. V tabeli aktivnosti definiramo horizontalno in vertikalno povezavo. Horizontalna povezava pove, katere entitete sodelujejo pri izvajanju posamezne aktivnosti. V ta namen uporabljamo dve črki: S in T, kjer S pomeni vir (Source) in T pomeni cilj (Target). Črka S v križišču (i,j) pomeni, da je entiteta j izvorna entiteta za aktivnost i. To pomeni, da entiteta j izvaja neko nalogo v okviru aktivnosti i. Črka T v križišču (i,j) pomeni, da je entiteta j ciljna entiteta za aktivnost i. To pomeni, da sprejme nek vhod od dane izvorne entitete. Ena aktivnost ima lahko eno ali več izvornih in ciljnih entitet, zato črki S in T indeksiramo z indeksom izvorne entitete. Vertikalna povezava ponazarja vrstni red aktivnosti za vsako entiteto posebej. To pomeni, da v okviru vsakega stolpca definiramo vrstni red aktivnosti entitete, definirane v tem stolpcu. Pri tem uporabimo črki P in U. Črka P pomeni *predhodnik* (Predecessor), črka U pa ponazarja *naslednik* (Successor). Črka P v križišču (i,j) pomeni, da je aktivnost i predhodnica določene aktivnosti (določenih aktivnosti), označene s črko U v koloni j. Črka U v križišču (i,j) pomeni, da je aktivnost i naslednica druge aktivnosti (drugih aktivnosti), označene s črko P v koloni j. Katerakoli aktivnost ima lahko eno ali več predhodnic in ravno tako eno ali več naslednic. Zato sta črki P in U indeksirani z indeksom predhodne aktivnosti. Poudariti je potrebno, da vertikalno povezavo definiramo samo pri internih entitetah. Za razvoj tabele aktivnosti sem organizirala intervjuje s predstavniki naslednjih oddelkov: Komerciale, Programov, Vhodov za DDV, Komercialne dokumentacije in Saldokontov. V tabeli aktivnosti (Tabela 2) je predstavljenih sedem entitet in deset aktivnosti. Prvih pet entitet je internih, zadnji dve pa sta eksterni.

Prva aktivnost 'Zahteva za predračun' pomeni, da kupec zahteva od komercialista predračun. To označimo tako, da zapišemo S_6 v križišče (1,6) in T_6 v križišče (1,1). Druga aktivnost 'Formiranje predračuna' pomeni, da komercialist kreira predračun in ga pošlje kupcu, zato to označimo z zapisom S_1 v križišču (2,1) in T_1 v križišču (2,6). Tretja aktivnost 'Plačilo predračuna' pomeni, da kupec plača (delno ali v celoti) predračun in o tem obvesti komercialista. Na podlagi tega napišemo S_6 v križišče (3,6) in T_6 v križišče (3,1). Vse ostale aktivnosti označimo z uporabo istega koncepta. Vertikalno povežemo aktivnosti vsake interne entitete. Tako naprimer je v zvezi s prvo entiteto, prva aktivnost 'Zahteva za predračun' predhodnica druge aktivnosti 'Formiranje

	ENTITETE ANALIZE	Komercialist	Vodja programa	Direktor sektorja	Direktor za grosizem
ZAHTEVE					
POROČILA	Poročilo po dobaviteljih	*	*	*	
	Poročilo po artiklih na različnih nivojih OE			*	*
	Poročilo o prodaji po strankah			*	*
	Poročilo o prodaji po artiklih		*		*

Tabela 1: Tabela entitet za proces Tranzit

predračuna', kar označimo s P_1 v križišče (1,1) in U_1 v križišče (2,1). Druga aktivnost 'Formiranje predračuna' je predhodnica tretje aktivnosti 'Plačilo predračuna', zato zapišemo P_2 v križišče (2,1) in U_2 v križišče (3,1). Vse ostale aktivnosti vertikalno povežemo po istem konceptu.

2.2.2 Naloge

Vsaka aktivnost je sestavljena iz ene ali več nalog in zaseda eno vrstico v tabeli aktivnosti. Naloge določene aktivnosti so označene z nepraznimi križišči v vrstici te aktivnosti.

Za definiranje nalog razvijemo posebno tabelo, ki jo imenujemo tabelo nalog. Tabela nalog je organizirana tako, da so naloge podane v vrsticah, medtem ko so karakteristike nalog definirane v stolpcih tabele. Iz tega sledi, da vsaka naloga, ki je predstavljena z nepraznim križiščem v tabeli aktivnosti, zaseda eno vrstico v tabeli nalog. Vsaka naloga je predstavljena s šifro K_{ij}, kjer K pomeni nalogo (Task), i in j so indeksi vrstice in stolpca, kjer je naloga definirana v tabeli aktivnosti. V stolpcih tabele nalog definiramo lastnosti nalog kot so: opis, čas, pogoj in vhod/izhod. V stolpcu Opis podamo kratek opis naloge. Čas lahko uporabimo za določanje časa, potrebnega za izvajanje naloge. V stolpcu Pogoj lahko definiramo vse pogoje, ki so potrebni pri opisani nalogi. V stolpcu Vhod/Izhod pa definiramo vse vhode in izhode, ki so povezani z nalogo K_{ij}. Tabela 3 prikazuje tabelo nalog pri primeru Tranzit. Vsaka naloga je predstavljena skladno z zgoraj opisanim korakom.

2.2.3 Delovni procesi in poslovni procesi

V tem koraku analitik analizira tabelo aktivnosti in poskuša grupirati aktivnosti v ustrezne grupe. Vsaka grupa aktivnosti predstavlja določen delovni proces. Iz tega sledi, da je delovni proces zbirka ene ali več aktivnosti in zato zaseda eno ali več vrstic tabele aktivnosti. Analitik nadaljuje delo z grupiranjem delovnih procesov v ustrezne grupe. Vsaka od teh grup predstavlja določen poslovni proces. Skladno s tem korakom sem ugotovila, da tvorijo vse aktivnosti tabele 2 delovni proces Tranzit, kar sem definirala v drugem stolpcu tabele. Tranzit je eden od delovnih procesov poslovnega procesa *trgovanje*, kar sem definirala v prvem stolpcu tabele aktivnosti.

2.3 Prenova poslovnih procesov

Ta faza se ukvarja z možnostjo izboljšanja funkcioniranja sistema. Prenova poslovnih procesov je metoda za identificiranje, definiranje in implementiranje sprememb (Watson, 1994). To je možno doseči z vzpostavljanjem

Del. proces	Posl. proces	Entitete	1	2	3	4	5	6	7
		Aktivnosti	Komercialist	Vodja (Komerciala) (Programi)	Referent programa za DDV)	Referent (Vhod dokume.)	Referent (Komerc. konti)	Kupec (Saldo-	Dobavitelj
TRGOVANJE	TRANZIT	1. Zahteva za predračun	T_6 P_1					S_6	
		2. Formiranje predračuna	S_1 U_1 P_2					T_1	
		3. Plačilo predračuna	T_6 U_2 P_3					S_6	
		4. Zaključitev predračuna	S_1 U_3 P_4						
		5. Zahtevek za naročilo	T_6 P_5					S_6	
		6. Formiranje naročila	S_1 U_4 U_5 P_6	T_1 P_6					
		7. Podpis vodje	T_2 U_6 P_7	S_2 U_6					
		8. Pošiljanje naročila dobavit.	S_1 U_7 P_8						T_1
		9. Pošiljanje računa v odd. Vhodi za DDV			T_7 P_9				S_7
		10. Vnos računa v knjigo računov	T_3 P_{10}		S_3 U_9				

Tabela 2: Tabela aktivnosti za proces Tranzit

ustrezne povezave med strateškim, poslovnim in operativnim nivojem organizacije. Strateški nivo organizacije je predstavljen z analizami, definiranimi v tabeli entitet, in povezanimi s strateškimi cilji organizacije. Poslovni nivo organizacije je predstavljen z analizami, ki so defnirane v tabeli entitet, povezanimi s poslovnimi cilji. Poleg tega pa je poslovni nivo predstavljen z množico poslovnih procesov, definiranih v tabeli aktivnosti. Operativni nivo je ravno tako predstavljen z določenimi analizami v tabeli entitet, ki so povezane z operativnimi cilji. Ta nivo je tudi predstavljen z množico delovnih procesov, definiranih v tabeli aktivnosti.

Z namenom izvedbe te faze se ustanovi projektni tim. Tim je sestavljen iz analitika in predstavnikov vseh treh nivojev organizacije. Projektni tim analizira obstoječi strateški plan in obstoječe strateške cilje ter predlaga ustrezne spremembe oziroma nove strateške cilje, če je to potrebno. Ustrezno s spremembami na strateškem nivoju se predlagajo spremembe v poslovnih ciljih, po potrebi pa tudi novi poslovni cilji. Spremembe v poslovnem planu se morajo odražati na operativnem nivoju tako, da se izvedejo nujne spremembe v delovnih procesih ali da se uvedejo novi delovni procesi in nove aktivnosti. Ker se ta primer ukvarja s samo enim delovnih procesom, ni bilo ne potrebe ne možnosti za izvedbo te faze.

2.4 Objektni model

Četrta faza metode TAD obravnava razvoj objektnega modela. Ta faza je sestavljena iz dveh korakov. V prvem koraku identificiramo razrede objektov, njihove atribute in asociacije. Informacije o razredih dobimo z analizo tabel entitet, aktivnosti in nalog. Običajno začnemo z analizo tabele nalog. Posebno skrbno analiziramo vsak opis, podan v stolpcu Opis. Nadaljujemo z analizo vhodov in izhodov, definiranih v stolpcu Vhod/Izhod. Poleg tega analiziramo tabelo aktivnosti in tabelo entitet. Rezultat omenjenih analiz je množica razredov, njihovih atributov in asociacij, ki jih uporabimo za razvoj začetnega objektnega modela. V ta namen uporabimo tri vrste struktur, ki lahko eksistirajo med razredi in atributi. To so: Je-del (is-part-of), Je-povezan-z (is-associated-with) in Je (isa). Struktura Je-del se uporablja za prikaz, da je lahko neka stvar konstruirana oziroma kreirana iz določenih elementov oziroma komponent. Na primer: Ime Je-del Stranka in Naslov Je-del Stranka opisuje agregacijo atributov razreda Stranka. Nadalje NarociloArtikel Je-del Naročilo opisuje agregacijo razredov. Struktura Je-povezan-z se uporablja za prikaz obstoja določene asociacije med dvema razredoma. Na primer: Naročilo Je-povezan-z Stranka opisuje asociacijo med razredoma Naročilo in Stranka. Struktura Je se uporablja za opis dedovanja med razredi. Iz opisa sledi, da v prvem

	Karakteristike				
AKTIVNOST	**Koda naloge**	**OPIS**	**ČAS**	**POGOJ**	**VHOD / IZHOD**
1. Zahteva za predračun	$K_{1,1}$	Kupec pošlje zahtevek po predračunu komercialistu.			
2. Formiranje predračuna	$K_{2,1}$	Komercialist formira predračun in ga pošlje kupcu.			Predračun
3. Plačilo predračuna	$K_{3,1}$	Komercialist sprejme plačilo predračuna.			Predračun
4. Zaključitev predračuna	$K_{4,1}$	Komercialist zaključi predračun.			Predračun
5. Zahtevek za naročilo	$K_{5,1}$	Kupec pošlje zahtevek za naročilo komercialistu.			
6. Formiranje naročila	$K_{6,1}$	Komercialist formira naročilo in ga pošlje vodji programa.			Naročilo
7. Podpis vodje	$K_{7,2}$	Vodja programa podpiše naročilo in ga vrne komercialistu.			Naročilo
8. Pošiljanje naročila dobavitelju	$K_{8,1}$	Komercialist pošlje naročilo dobavitelju.		Podpisano naročilo	Naročilo
9. Pošiljanje računa v odd. Vhodi za DDV	$K_{9,3}$	Referent sprejme račun.		Dobaviteljev račun	Dobaviteljev račun
10. Vnos računa v knjigo računov	$K_{10,3}$	Referent zavede račun v knjigo računov in ga pošlje komercialistu.			Račun

Tabela 3: Tabela nalog za proces Tranzit

koraku razvijemo začetni objektni model z uporabo strukture Je-del in Je-povezan-z. V drugem koraku poskušamo organizirati razrede tako, da implementiramo dedovanje med razredi z uporabo Je strukture. Dedovanje je najzanimivejši element objektnega modeliranja. Dedovanje pomeni identifikacijo in definiranje sistema hierarhij med razredi tako, da podrazred lahko podeduje atribute in operacije nadrazreda. Skladno s prvim korakom te faze prikazuje slika 1 objektni model procesa Tranzit.

2.5 Načrtovanje

Peta faza se ukvarja z načrtovanjem sistema. Faza je sestavljena iz dveh korakov. V prvem identificiramo operacije objektnega modela. V drugem koraku razvije analitik tako imenovani aplikacijski model.

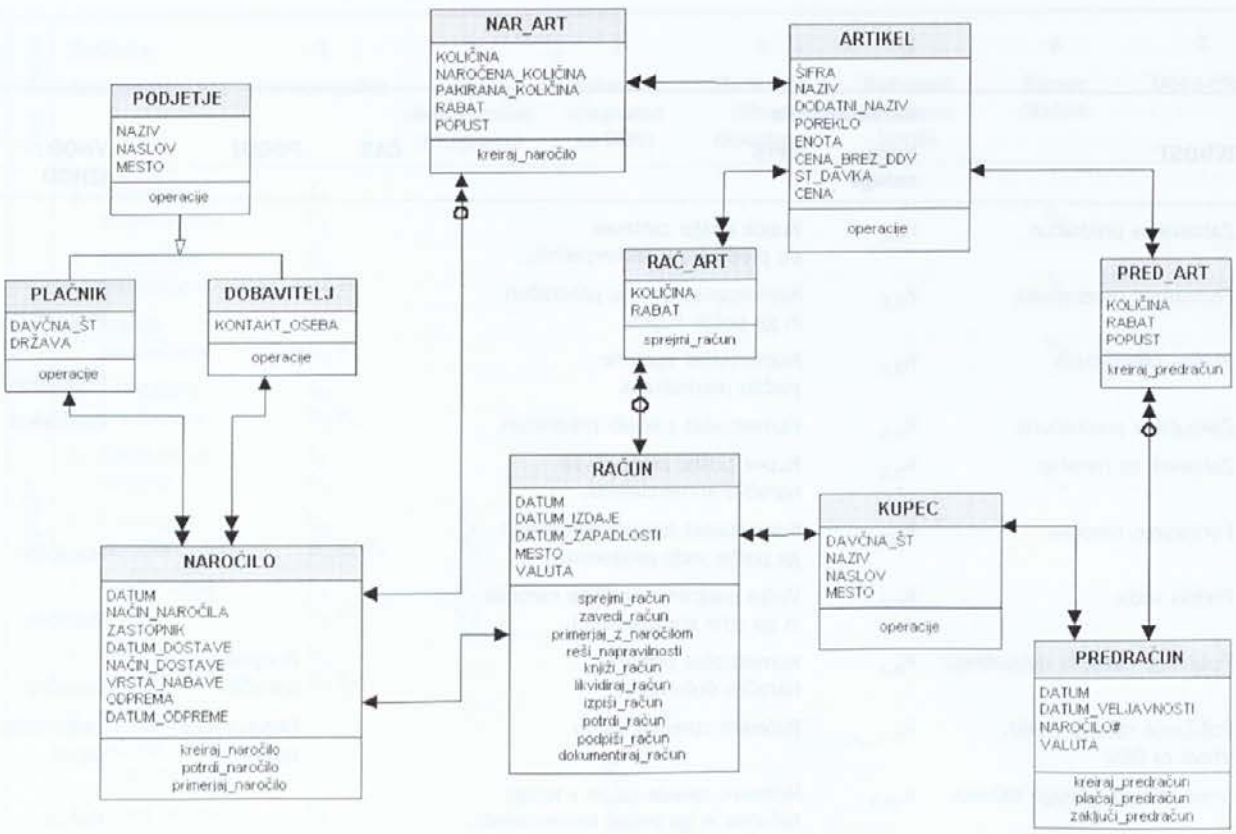
2.5.1 Operacije

Identifikacija operacij teče na podlagi opisov nalog v tabeli nalog. V ta namen mora analitik dodobra analizirati vsak opis obravnavane naloge in iz opisa identificirati operacije, ki so povezane z enim ali več razredov objektnega modela. Slika 1 prikazuje objektni model obogaten z operacijami, ki sem jih definirala z analizo opisov v tabeli 3.

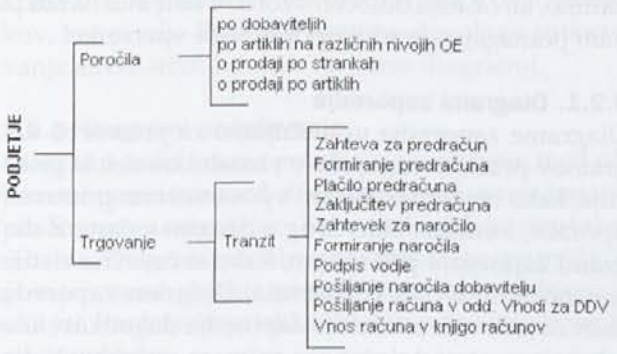
Aplikacijski model

Aplikacijski model predstavlja preslikavo informacij zbranih v tabelah. Aplikaciji model je sestavljen iz več delov. Prva dva dela sta izdelana iz informacij, registriranih v tabeli entitet. Prvi del prikazuje vitalne analize, drugi pa probleme za podporo odločanju.

Ostali deli aplikacijskega modela so izpeljani iz tabele aktivnosti. Za vsak poslovni proces kreiramo poseben del aplikacijskega modela. Slika 2 prikazuje aplikacijski model obravnavanega problema tranzita. Prvi del modela prikazuje poročila, ki so povezana s tranzitom, drugi pa prikazuje delovni proces Tranzit in njegove aktivnosti. Aplikacijski model predstavlja preslikavo informacij zbranih v tabelah. Aplikacijski model je sestavljen iz več delov. Prva dva dela sta izdelana iz informacij, registriranih v tabeli entitet. Prvi del prikazuje vitalne analize, drugi pa probleme za podporo odločanju. Ostali deli aplikacijskega modela so izpeljani iz tabele aktivnosti. Za vsak poslovni proces kreiramo poseben del aplikacijskega modela. Slika 2 prikazuje aplikacijski model obravnavanega problema tranzita. Prvi del modela prikazuje poročila, ki so povezana s tranzitom, drugi pa prikazuje delovni proces Tranzit in njegove aktivnosti.



Slika 1: Objektni model za proces Tranzit



Slika 2: Aplikacijski model za proces Tranzit

2.6 Implementacija

Zadnja faza se ukvarja z implementacijo sistema. Vhodi v to fazo so objektni model in aplikacijski model, ki jih analitik prevede v bazo podatkov in programsko kodo.

3 Modeliranje informacijskega sistema z jezikom UML

Prikazati želim razvoj informacijskega sistema za podporo delovnemu procesu Tranzit z drugega zornega kota. Odločila sem se za uporabo jezika UML (Unified Modeling Language), saj je le-ta standardni modelirni jezik. UML je jezik za specificiranje, vizualiziranje, konstruiranje in dokumentiranje programskih sistemov (UML version 1.1). UML je zbirka standardnih modelov, ki jih uporabljamo za načrtovanje objektno orientiranega projekta (Sturm, 1999). UML je samo modelirni jezik. UML je torej vizualni modelirni jezik, ki omogoča opisovanje modelov tako, da uporablja standardno notacijo in na ta način omogoča izmenjavo modelov (pogledov) med strokovnjaki. UML uporablja objektno-orientirane koncepte in je neodvisen od programskega jezika in razvojnega procesa. V splošnem jeziki za modeliranje uporabljajo diagrame za predstavitev entitet in njihovih povezav. Tako se tudi v jeziku UML za predstavitev uporablja različne grafe, ki jih uporabnik izbere sam, glede na to, kateri so zanj najbolj uporabni. Zaradi pomanjkanja prostora bom prikazala samo nekatere diagrame jezika UML.

3.1 Diagrami primerov uporabe

Diagram primerov uporabe je dokument, ki jasno opisuje zaporedje akcij akterja, ki uporablja sistem za izpolnitev procesa (Larman, 1998). Je prikaz uporabniških zahtev. Razvoj diagrama primerov uporabe predstavlja prvi korak pri uporabi UML-ja. Začnemo z vidika uporabnikov sistema in prevedemo uporabnikove potrebe v lahko razumljiv model. Ta diagram torej predstavlja, kako sistem ali del sistema deluje z vidika uporabnika.

Diagram primerov uporabe sestavljajo akter, primer uporabe in povezave med akterji. Akter je oseba ali drug eksterni sistem, ki ima določene interakcije z opazovanim sistemom. Delimo jih na primarne in sekundarne akterje. Primarni akter je tisti, ki začne uporabljati primer uporabe oziroma tisti, ki primer uporabe sproži, sekundarni akter pa v njem le sodeluje. Akter je torej vloga, ki jo oseba ali skupina igra v povezavi z uporabo obravnavanega sistema. Akterja predstavljamo z likom človeka. Primer uporabe izdela analitik na osnovi intervjujev z uporabniki na različnih organizacijskih nivojih in predstavlja dobro definirano delovanje sistema. Vsako dobro definirano delovanje definiramo kot primer uporabe, ki ga prikažemo v obliki elipse. Primer uporabe podrobno tekstovno opišemo. V ta namen uporabimo obrazec, ki je prikazan na sliki 3.

Primeri uporabe in akterji so med seboj povezani. V ta namen se uporabljajo asociacija, generalizacija, stereotip 'vsebuje' in stereotip 'razširja'. Stereotip 'vsebuje' pomeni, da pri uporabi enega primera uporabe lahko pride do uporabe drugega primera uporabe. Ta tip povezave uporabljamo, kadar se določeno delovanje ponavlja pri določenem številu primerov uporabe. Namen tega je, da odpravimo specifično podvajanje. Za to delovanje moramo kreirati nov primer uporabe,

PRIMER UPORABE: KREIRATI PREDRAČUN

Namen: Formirati predračun.

Primarni akter: Komercialist.

Sekundarni akter: /

Začetna točka: Prejeti zahtevek za predračun.

Končna točka: Predračun je kreiran ali razveljavljen.

Tok dogodkov: Komercialist prejme kupčev telefonski ali pisni zahtevek za predračun. Kreira predračun tako, da vpiše podatke o kupcu in o artiklih, ki jih kupec želi naročiti, ter izračuna znesek, ki naj bi ga kupec plačal.

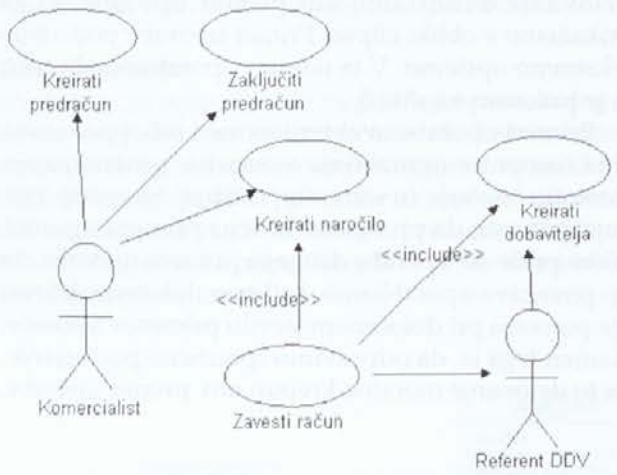
Alternativni tok dogodkov: /

Merljiv rezultat: Kreiran predračun.

Razširitev primera uporabe: /

Slika 3: Diagram primerov uporabe za proces Tranzit

ki ga imenujemo klientni primer porabe (client use case). Smer povezave vsebuje je usmerjena k klientnemu primeru uporabe. Stereotip 'razširja' se uporablja takrat, ko ima neki primer uporabe poleg normalnega toka dogodkov še alternativen tok dogodkov. V tem primeru definiramo poseben nov primer uporabe za alternativni tok dogodkov, ki ga povežemo z baznim s stereotipom 'razširja'. Asociacija je logična povezava med dvema akterjema, to je med akterjem in primerom uporabe. Prikazana je s puščico, na kateri lahko definiramo tudi ime asociacije. Zaradi omejenega prostora v nadaljevanju podajam samo opis primera uporabe 'Kreirati predračun' z uporabo prej omenjenega obrazca.



Slika 4 prikazuje del diagrama primerov uporabe za proces Tranzit.

3.2 Interakcijski diagrami

Naslednji korak pri načrtovanju sistema so interakcijski diagrami, kjer prevedemo narisane primere uporabe v diagrame, ki grafično prikažejo časovno zaporedje komunikacij, interakcij med akterji in komponentami, objekti primera uporabe in objekti samimi. Ti diagrami prikazujejo dinamično obnašanje sistema. Ločimo dva tipa interakcijskih diagramov in sicer diagrame zaporedja in diagrame sodelovanja.

Oba tipa diagramov predstavljata isto informacijo na različne načine. Pri diagramu zaporedij ima bistveno vlogo časovno zaporedje dogodkov, medtem ko ima diagram sodelovanja večji poudarek na povezavi med elementi sistema. S pomočjo teh diagramov lahko vidimo, ali obstaja določen vzorec v sistemu, hkrati pa nam pomagajo zgraditi uporabniške vmesnike.

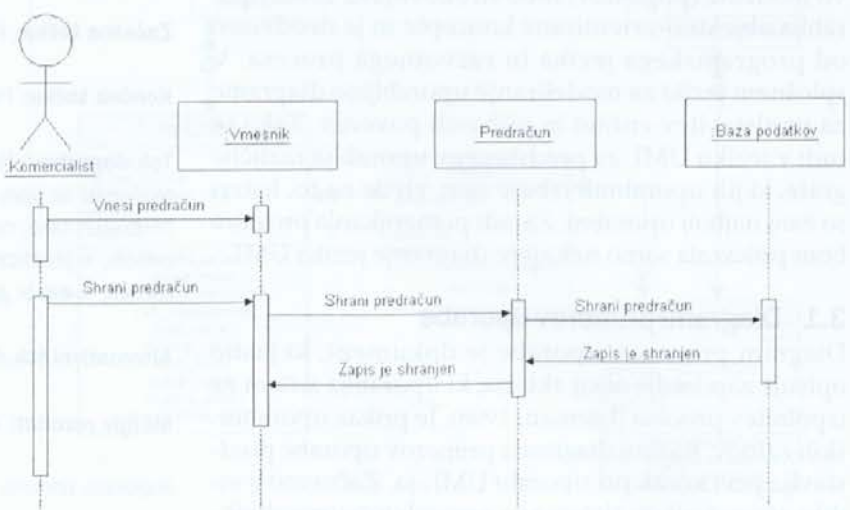
3.2.1. Diagrami zaporedja

Diagrame zaporedja uporabljamo za pretvorbo diagramov primerov uporabe v vizualni model, ki prikazuje, kako objekti, povezani s posameznim primerom uporabe, komunicirajo drug z drugim v času. Z diagrami zaporedja prikažemo, kako si časovno sledijo sporočila med objekti sistema. Diagram zaporedja služi za jasno identifikacijo zaporedja dogodkov, ki se odvijajo znotraj določenega primera uporabe. V diagramih zaporedja poleg akterja in objektov uporabljamo še naslednje elemente:

- življenjsko črto, ki izhaja iz vsakega akterja in objekta navpično navzdol, kjer se čas obstoja objekta meri od vrha diagrama navzdol.
- puščice, ki povezujejo življenjske črte, prikažemo med akterji in različnimi objekti. Na puščice napišemo imena sporočil. Sporočila si slede v časovnem zaporedju, ki ponazarja zaporedje dogodkov. Posebna vrsta sporočil so refleksivna sporočila, kjer objekt komunicira sam s sabo.
- pravokotnike, ki jih rišemo na življenjske črte, pomenijo pa aktivacije oziroma časovno obdobje, ko je akter ali objekt odgovoren za akcijo.

Dvopičje pred imenom objekta pomeni, da gre za objekt razreda in ne razred. Objekti sodelujejo drug z drugim, razredi pa ne. V diagramih zaporedja nastopajo samo objekti in akterji. Tako torej časovna premica teče od posameznega objekta in predstavlja obdobje, ko objekt obstaja. Sodelovanje med akterji in objekti ter objekti samimi prikazujejo sporočila, ki se kasneje, v diagramu razredov, pretvorijo v operacije razredov.

Slika 5 prikazuje diagram zaporedij za primer uporabe Kreirati predračun. Na diagramu ni narisan alternativni tok dogodkov, ki se sproži v primeru, ko



Slika 5: Diagram zaporedij Kreirati predračun

komercialist prekliče vnos predračuna v bazo podatkov, ker orodje Rational Rose ne dovoljuje prikazovanje alternativnih tokov na istem diagramu.

3.2.2 Diagrami sodelovanja

Diagrami sodelovanja poudarjajo povezave med objekti ter povezave med objekti in akterji, ne pa trajanje posamezne interakcije. Tako prikazujejo strukturo sistema. Diagrami sodelovanja prikažejo sporočila, ki potujejo med objekti ter med primarnimi akterji in objekti. Predstavljajo lahko celoten primer uporabe ali pa samo del.

3.3 Razredni diagram

Diagrami razredov se ukvarjajo s sistemskim načrtovanjem. Razred je opis skupine objektov, ki imajo enake atribute, operacije in relacije; torej razred predstavlja skupino objektov s podobnimi lastnostmi in skupnim obnašanjem. Vsak objekt je primerek nekega razreda. Razredi med seboj niso v interakciji, lahko pa so v medsebojni interakciji objekti iz različnih razredov. Diagram razredov torej prikazuje množico razredov in relacij med njimi. Predstavlja statičen pogled sistema. Diagram razredov prikazuje javne in privatne operacije ter lastnosti razredov. Izdelamo ga na podlagi informacij iz diagrama primera uporabe in diagramov zaporedja. Lastnosti posameznega razreda ugotovimo s pomočjo primerov uporabe, medtem ko nekatere operacije razberemo iz diagramov zaporedja. Razrede določimo iz opisov primerov uporabe. Povezave med razredi spadajo v tri kategorije:

Asociacija med dvema razredoma: en razred ne zahteva obstoja drugega razreda. Vsak razred ima enako odgovornost v partnerstvu. Predpostavimo jih s polno črto, ki povezuje oba razreda.

Generalizacija: pomeni, da je vsak objekt razreda posebna verzija objekta drugega razreda. Generalizacijo prikažemo s trikotnikom usmerjenim k staršev-skemu objektu.

Agregacija (sestavljena agregacija): je relacija med razredoma, kjer en razred ne more obstajati brez drugega razreda. To je ponavadi enosmerna relacija, ki je prikazana z rombom.

Lastnosti razredov dobimo iz tekstualnih opisov primerov uporabe. Operacije razredov dobimo iz analize vsakega zaporednega diagrama. Če objekt pošlje sporočilo drugemu objektu, to pove, da je potrebno definirati operacijo znotraj objekta, ki prejme sporočilo, pri čemer pa ima lahko več razredov enake operacije. Slika 1 prikazuje diagram razredov za proces Tranzit.

3.4 Aktivnostni diagrami

Aktivnostni diagrami so podobni diagramom poteka. Uporabljajo se lahko v različne namene in sicer za prikazovanje delovanja sistema, za opis primera uporabe ali za podoben opis operacij razrednega diagrama.

4 Primerjava metodologij

V ta namen bomo uporabili kriterije, ki jih v svojem prispevku Object-Oriented Technology predlaga avtor Josie Lam (Lam, 1997):

1. kriterij: ustreznost metodologije za definiranje zahtev in pokrivanje faz razvoja programske opreme.
2. kriterij: koliko vplivajo razvijalčeve izkušnje z metodologijo na uporabnost metodologije.
3. kriterij: kakšno podporo ima metodologija.
4. kriterij: enostavnost metodologije za uporabo in razumevanje.

V povezavi s prvim kriterijem ugotavljam, da metodologiji temeljita na različnih konceptih. UML pokriva vse faze razvoja programske opreme z uporabo različnih diagramov kot so diagrami primerov uporabe, zaporedja, sodelovanja, razredov, aktivnosti in drugi. TAD metodologija bazira razvoj programske opreme na tabelah entitet, aktivnosti in nalog. Poleg tega razvija model objektov in aplikativni model.

Drugi kriterij govori o vplivu izkušenj z metodologijo na uporabi metodologije na dejanskem primeru. Ugotavljam, da je za učinkovito uporabo ene ali druge metodologije potrebno nekaj izkušenj razvijalca, vendar mislim, da za uporabo UML potrebujemo veliko več izkušenj.

V zvezi s tretjim kriterijem ugotavljam, da ima UML dobro razvito podporo, ki razvijalca pripelje do implementacije, medtem ko je podpora metodologije TAD še v razvoju.

Glede na zadnji kriterij menim, da sta obe metodologiji jasni in razumljivi za uporabo. Moje mnenje pa je, da je uporaba tabel v metodologiji TAD enostavnejša in natančnejša, kot pa risanje diagramov pri UML-ju.

Slabost UML-ja je ta, da v končni fazi dobimo precejšnjo število diagramov. Vsak diagram predstavlja določeno delovanje sistema. Posledica tega je, da sistem postane neobvladljiv pri razvoju kakšnega večjega informacijskega sistema.

Prednost metodologije TAD je v tem, da tabela aktivnosti omogoča prikazovanje delovanja celotnega sistema, kar omogoča razvijalcu obvladovanje sistema. Nadaljna prednost je v enostavnem prehodu iz faze analize v fazo načrtovanja s prevedbo tabele entitet in aktivnosti v aplikacijski model.

5 Zaključek

Eden največjih problemov tradicionalnih metodologij razvoja informacijskih sistemov je v nepovezanosti, ko

projekt napreduje iz faze konceptualnega modeliranja, v fazo načrtovanja in kasneje v fazo implementacije.

Razvijalci UML-ja so vložili precej truda v izdelavo metodologije in orodja, ki omogoča spojitev korakov razvoja informacijskega sistema, vendar pa še zmeraj obstaja nepovezanost med različnimi koraki UML-ja.

Metodologija TAD je izjema v svoji edinstveni specifikaciji konstruktov modeliranja, v katerih uporablja enostavne tabele za popolno specifikacijo sistema tako v fazah analize kot načrtovanja in implementacije. Metodologija TAD povezuje tabele in elemente tabel v celovit načrt sistema. Uporablja iste elemente tako za prikaz strukture kot tudi za prikaz funkcionalnih odvisnosti končnega sistema. Zahteve uporabnikov, ki so definirane v prvi fazi, so ponovno uporabljene in preizkušene v zadnjih fazah metodologije. Ta natančna implementacija zahtev uporabnikov je vključena neposredno v postopek metodologije TAD.

6 Literatura in viri

Damij Talib:
Development of a Hospital Information System using the TAD Method. Journal of the American Medical Informatics Association, Chatsworth, 5(1998), 2, str. 16-20.

Damij Talib:
An Object-Oriented Methodology for Information Systems Development and Business Process Reengineering. Journal of Object-Oriented Programming, Chatsworth, 13(2000), 4, str. 23-34.

Watson Gregory H.:
Business Systems Engineering: Managing Breakthrough Changes for Productivity and Profit. John Wiley & Sons, New York, 1994, 287 str.

Sturm Jake:
VB6 UML Design and Development. Wrox Press, Birmingham, 1999, 581 str.

Rumbaugh James, Jacobson Ivar, Booch Grady:
The Unified Modeling Language Reference Manual. Addison Wesley Longman Inc., Reading, Massachusetts, 1998, 550 str.

Larman Craig:
Applying UML and Patterns. Prentice Hall, 1998, 507 str.

Unified Modeling Language version 1.1.
[URL: http://www.rational.com/uml/resources/documentation/formats.jsp], 15.05.2001.

Lam J.:
Object-Oriented Technology.
[URL: http://disc.cba.uh.edu/~rhirsch/spring97/lam1/hope.htm], 12.05.2001.

◆

Nadja Damij je študentka četrtega letnika na Ekonomski fakulteti v Ljubljani na smeri Poslovna informatika. V kratkem bo zagovarjala svoje diplomsko delo s področja objektno-orientiranega modeliranja.