UNIVERZA V LJUBLJANI

EKONOMSKA FAKULTETA

MAGISTRSKA NALOGA

ANALIZA UPORABE PRISTOPA K RAZVOJU PROGRAMSKIH REŠITEV NA OSNOVI MODELIRANJA POSLOVNIH PRAVIL

LJUBLJANA, SEPTEMBER 2010 JERNEJ IVANČIČ

IZJAVA

Študent Jernej Ivančič izjavljam, da sem avtor tega magistrskega dela, ki sem ga napisal v soglasju s svetovalcem prof. dr. Jurijem Jakličem, in da v skladu s 1. odstavkom 21. člena Zakona o avtorskih in sorodnih pravicah dovolim njegovo objavo na fakultetnih spletnih straneh.

V Ljubljani, dne 15. 8. 2010 Podpis:

KAZALO

UVOD 1
1 TEORETIČNA IZHODIŠČA 4
1.1 Razvoj programskih rešitev in poslovno modeliranje 4
1.1.1 Cikel razvoja programske rešitve 6
1.1.2 Informatizacija in avtomatizacija 6
1.1.3 Metodologije razvoja programskih rešitev 6
1.1.4 Razvoj programskih orodij in tehnologij 9
1.2 Poslovne zahteve in poslovni model 10
1.2.1 Opredelitev poslovne zahteve 10
1.3 Model poslovnih procesov 12
1.3.1 Poslovni procesi 12
1.3.2 Modeliranje poslovnih procesov 13
1.4 Poslovna pravila 14
1.4.1 Opredelitev poslovnih pravil 15
1.4.2 Delitev poslovnih pravil 16
1.5 Podatkovno modeliranje 18
2 ZNAČILNOSTI PRISTOPA POSLOVNIH PRAVIL 19
2.1 Osnovna načela pristopa poslovnih pravil 20
2.1.1 Osnovna vodila in opredelitev pristopa poslovnih pravil 21
2.1.2 Pristop poslovnih pravil v Zachmanovem ogrodju 21
2.1.3 Ostale prednosti in značilnosti pristopa poslovnih pravil 22
2.1.4 Taktični načrt 23
2.1.5 Kategorije projektov v prihodnosti 23
2.1.6 Tehnologija in pristop poslovnih pravil 25
2.2 Poslovno modeliranje pri pristopu poslovnih pravil 25
2.2.1 Model izrazov in dejstev 26
2.2.2 Poslovna pravila v vlogi kontrole 28
2.2.3 Poslovni proces oziroma potek dela 31
2.2.4 Predloge poslovnih pravil v naravnem jeziku 31
2.2.5 Informatizacija pri poslovnem pristopu 33
3 DRUGE METODOLOGIJE IN PRISTOPI RAZVOJA PROGRAMSKIH REŠITEV 34
3.1 Strukturni pristop 34
3.2 Objektno orientiran pristop in UML 35
3.2.1 Povzetek uporabnosti jezika UML 36
3.2.2 Informatizacija poslovnih pravil v UML 37
3.3 Storitveno usmerjena arhitektura 37
4 INFORMATIZACIJA POSLOVNEGA MODELA NA PODLAGI NOVEGA PRISTOPA POSLOVNEGA MODELIRANJA 40

4.1 Opis razvoja informatizacije programske rešitve na podlagi poslovnega modela ... 40
4.1.1 Zbiranje zahtev in določanje taktike ... 41
4.1.2 Poslovno modeliranje ... 42
4.1.3 Informatizacija ... 43
4.1.4 Testiranje in vpeljava ... 43
4.1.5 Opis razvoja programske rešitve s stališča tehnologije ... 44
4.2 Orodja za opis poslovnega modela ... 44
4.2.1 Opis orodja FactXpress ... 45
4.2.2 Opis orodja RuleXpress ... 47
4.2.3 Prednosti in novosti orodja RuleXpress ... 48
4.3 Orodja za informatizacijo ... 50
4.3.1 Opis orodja ILOG ... 50
4.3.2 Opis orodja RuleStudio in osnovnih gradnikov ILOG-a ... 51
4.3.3 Integracija ILOG in RuleXpress orodij ... 53
4.4 Analiza pristopa poslovnih pravil : problemi in rešitve ... 54
5 OPIS IN PREDSTAVITEV IZDELAVE PROTOTIPA ... 55
5.1 Taktični načrt projekta za obračunavanje ... 57
5.2 Programska rešitev za analizo odčitkov ... 59
5.2.1 Izdelava podatkovnega modela oziroma strukture ... 60
5.2.2 Skripta programske rešitve ... 64
5.2.3 Implementacija poslovnih pravil ... 64
5.3 Primeri poslovnih pravil pri obračunavanju ... 66
SKLEP ... 69
LITERATURA IN VIRI ... 72
PRILOGE

KAZALO SLIK

Slika 1: Razvoj programskih rešitev ... 8
Slika 2: Shematski prikaz procesa ... 13
Slika 3: Metamodel aktivnosti in poslovnih pravil ... 14
Slika 4: Delitev poslovnih pravil ... 17
Slika 5: Tehnološka klasifikacija poslovnih pravil ... 17
Slika 6: Razvoj programske rešitve po aktivnostih ... 19
Slika 7: Zachmanovo ogrodje ... 22
Slika 8: Primer grafične predstavitve modela dejstev ... 27
Slika 9: Aktivnosti in faze strukturnega pristopa ... 35
Slika 10: Pvezava SOA in drugih metodologij ... 38
Slika 11: Metamodel poslovnih pravil, procesov in storitev ... 39
Slika 12: Primer dekompozicije storitev in poslovnih funkcij ... 39
Slika 13: Faze razvoja programske rešitve ... 41

Slika 14: Aktivnosti poslovnega modeliranja 42
Slika 15: Postopek razvoja programske rešitve s poudarkom na poslovnih pravilih 44
Slika 16: Primer modela dejstev za osebo 46
Slika 17: Grafični uporabniški vmesnik orodja RuleXpress 48
Slika 18: Arhitektura ILOG paketa 51
Slika 19: Razvoj programske rešitve ali projekta v orodju ILOG 52
Slika 20: Grafični uporabniški vmesnik razvojnega okolja ILOG - RuleStudio 53
Slika 21: Shematski prikaz povezav orodij in tehnologije 54
Slika 22: Metamodel informatizacije programske rešitve 55
Slika 23: Ogrodje poslovnih procesov eTOM 56
Slika 24: Arhitektura obračunskega sistema 57
Slika 25: Poslovni proces obračunskega sistema 58
Slika 26: Grafični prikaz taktičnega načrta 60
Slika 27: Model dejstev za analizo odčitkov na podlagi zgodovine 62
Slika 28: Razredni diagram analize odčitkov na podlagi zgodovine 63

KAZALO TABEL

Tabela 1: Kronologija metodologij razvoja programskih rešitev 7
Tabela 2: Izvleček poslovnega pristopa iz Zachmanovega ogrodja 22
Tabela 3: Kategorije projektov v prihodnosti 24
Tabela 4: Predloge poslovnih pravil (angl. BRS rule speak) 32
Tabela 5: Dejavniki abstrakcije poslovnega modela 33
Tabela 6: Orodja za upravljanje in informatizacijo poslovnih pravil 45

KAZALO PRIMEROV

Primer 1: Pravilo MS Outlook, ki premakne sporočila v posebno mapo 18
Primer 2: Omejitveno poslovno pravilo 23
Primer 3: Nekaj primerov dejstev 27
Primer 4: Poslovna pravila glede na tip kontrole 28
Primer 5: Poslovno pravilo za opis deklarativnosti 29
Primer 6: Poslovno pravilo izdaje naročila in njegova grafična predstavitev 30
Primer 7: Skripta poteka dela ali postopka 31
Primer 8: Poročilo modela dejstev 46
Primer 9: Taktični načrt projekta za obračunavanje 58
Primer 10: Uporabniške zahteve programske rešitve analiza odčitkov 59
Primer 11: Taktični načrt programske rešitve analiza odčitkov 59
Primer 12: Primer izdelave fizičnih podatkovnih objektov 63
Primer 13: Skripta za analizo odčitka 64
Primer 14: Model dejstev za izračun temperaturnega primankljaja 64
Primer 15: Omejitvena in izpeljana poslovna pravila temperaturnega primankljaja 65

Primer 16: Implementirano poslovno pravilo v RuleStudio okolju 65
Primer 17: Izgled implementirane skripte za analizo odčitkov v RuleStudio 65
Primer 18: Kompleksno poslovno pravilo, ki določa davčno leto in mesec računa 67
Primer 19: Poslovno pravilo, ki omogoča seštevanje v zanki 67
Primer 20: Skupek primerov izbire ovrednotenja cene iz modelirnega vidika 68
Primer 21: Primer odločitvene tabele prikaza plačilnega inštrumenta 69

UVOD

Problematika in namen magistrskega dela

Pri **razvoju programskih rešitev**, katerega ključen del je **poslovno modeliranje**, velja omeniti težave, ki lahko nastanejo pri projektih. Veliko projektov danes ne dosega zastavljenih rokov in se roki neprestano prestavljajo. Zajem zahtev se v osnovi veliko spreminja, zato so potrebni popravki, spremembe zahtev ali pa nove zahteve, ki pa jih naročnik projekta ali sponzor pogosto razume kot vzdrževanje. Velikokrat se rešuje težave kasneje, ko pridejo na plano, saj ni nikoli časa za načrtovanje. Vprašanje, ki se postavlja, je, ali je to zaradi kompleksnosti računalniških orodij in tehnologije, ali se osnovne zahteve ne zapišejo dovolj natančno, preden se začne razvijanje programske rešitve? Težave lahko vodijo v neuspeh projektov in razlogi za to so lahko različni. Morgan (2002) pravi, da je ključen razlog za neuspešnost projektov upravljanje z zahtevami. Zahteve ali potrebe opisujejo le željeno delovanje poslovnega sistema, na drugi strani pa poslovni model, ki je rezultat poslovnega modeliranja, opisuje obstoječe in željeno delovanje poslovnega sistema.

Analiza zahtev in poslovno modeliranje v veliki meri vplivata na potek razvoja programskih rešitev, razvoj informacijskih sistemov in prenove poslovanja. Pri modeliranju je potrebno analizirati poslovne zahteve in poslovne procese uporabnikov in udeležencev poslovnega sistema. Poslovni model, ki predstavlja opis in delovanje poslovnega sistema v nekem okolju in je rezultat poslovnega modeliranja, mora predstavljati realno sliko sistema. Z vidika razvoja programskih rešitev lahko poslovno modeliranje delimo na tri področja, ki so med seboj prepletena: modeliranje poslovnih procesov, poslovnih pravil in podatkovnega modeliranja (Ross, 2003)

Poznamo več **pristopov oziroma metodologij** razvoja programskih rešitev, ki jih ponujajo in podpirajo različna interesna združenja in podjetja. V preteklosti je bila najbolj znana metodologija strukturnega razvoja programskih rešitev (Krisper, 2000), kjer je bil poudarek na podatkovnem modeliranju. Kasneje se je uveljavil objektno orientiran razvoj, ki temelji na objektno orientirani analizi, načrtovanju razredov in objekov ter je v današnjem času najbolj poznan.

Ključno vlogo pri izdelavi programskih rešitev in poslovnem modeliranju ima poslovni analitik. Poslovni analitiki morajo imeti različna znanja in veščine, predvsem pa morajo biti sposobni izdelati takšen poslovni model oziroma poslovne zahteve, ki pokriva vsa področja in vse faze razvoja programske rešitve. Končni cilj je programska rešitev, ki deluje ustrezno s specifikacijo in poslovnimi potrebami, podpira poslovne procese in je uporabna. Priporočljivo je, da je vsaj del razvojnega cikla programske rešitve avtomatiziran, od poslovne zahteve do informatizacije.

V zadnjem času je postal aktualen nov pristop (metodologija) razvoja programskih rešitev, ki poudarja pomembnost poslovnih pravil. Prednost novega pristopa je enostaven in razumljiv prehod v informatizacijo programske rešitve. Pristop predstavlja nov zgled pri razvoju poslovnih informacijskih sistemov in programskih rešitev. Nov pristop se ukvarja s poslovnim sistemom, pri čemer se poslovna pravila pojavljajo kot zmogljivo, učinkovito in fleksibilno orodje za opis ali obravnavo poslovnih sistemov (Ross, 2003).

Pri poslovnem modeliranju je pomembno vprašanje, kako podroben naj bo poslovni model. Morgan (2002) prikazuje primere elementov, ki jih pri poslovnem modeliranju potrebujemo. Število elementov je odvisno od poslovnega modela in nivoja podrobnosti, ki jih nameravamo vključiti, poznano pa je, da je največ elementov ravno poslovnih pravil (Morgan, 2002). Zato je pri razvoju programskih rešitev poslovna pravila potrebno postaviti v ospredje.

Med drugim je pomembno področje poslovnega modeliranja tudi opredelitev in upravljanje poslovnih procesov. Poslovni proces v organizaciji (podjetju) sestavljajo aktivnosti, kjer je s strukturo opisano njihovo logično zaporedje in medsebojna odvisnost ter katerih namen je doseganje želenega rezultata. Modeliranje poslovnih procesov omogoča enotno razumevanje in analizo poslovnih procesov, ki je osnova za temeljito razumevanje procesa. Preko poslovnih procesov je možno analizirati in povezati organizacijo.

Zmožnosti **informacijske tehnologije** se hitro razvijajo v zadnjih dveh desetletjih. Podjetja veliko vlagajo v informacijsko tehnologijo, zato da bi imeli na trgu vodilne položaje in tehnologije. Po drugi strani se podjetja sprašujejo, kako maksimalno izkoristiti nove tehnologije, da bi pridobili nove poslovne priložnosti in katere dele poslovnega sistema je smiselno avtomatizirati. Nove tehnologije, ki se danes veliko omenjajo, so orodja za upravljanje poslovnih procesov (angl. *bussiness process modeling*, okr. BPM) in orodja za upravljanje in izvajanje poslovnih pravil (angl. *rule engine, rule repository*), ki pa so tesno povezana z novim pristopom, ki ga predlaga Ross (2003).

V preteklosti se je poslovni sistem prilagajal informacijski podpori in informacijski tehnologiji. Uporabniki poslovnega sistema so uporabljali programske rešitve, ki so jih razvijali informatiki. Danes je trend, da se oba pola, informatiki in poslovni uporabniki, združijo in usmerijo k enemu cilju. Cilj je napisati zahteve, ki jih potrebuje poslovni sistem, in je v okviru informacijske tehnologije možno takšen poslovni sistem razviti. Postavljajo se vprašanja in izzivi, kako bi zmanjšali vrzel med informatiki in poslovnimi analitiki.

Pogosti problemi pri realizaciji programskih rešitev so sistemi, ki ne podpirajo poslovanja v celoti tako, kot bi morali. Do takšne situacije lahko pride zaradi različnih razlogov: ni ustrezne specifikacije poslovnih zahtev, razvijalci programskih rešitev napačno razumejo poslovne zahteve in poslovanje, ali pa se spremembe poslovanja in poslovnih zahtev

spreminjajo tako hitro, da razvijalci ne sledijo takšnemu tempu. Rešitev tega problema je v razumevanju obeh polov in pravi meri informacijske tehnologije, ki podpira poslovanje. Poslovno modeliranje mora zagotavljati model dejanskega poslovanja, njegove cilje, procese, vire in poslovna pravila.

V podjetju Gartner pravijo, da je poudarek na poslovnih procesih in storitveno orientiranih aplikacijah, ki vključujejo analizo poslovnih procesov, objektno orientirane analize in podatkovnega modeliranja. Te tehnike pa vodijo v posebne celovite programske pakete (Blechar, 2007).

Tehnologije, ki omogočajo podporo poslovnih pravil, so pomembne pri upravljanju poslovnih procesov zato, ker omogočajo fleksibilnost in prilagodljivost pri komponiranju kompleksnih odločitev in transparentnosti procesa uporabniku. Sistem za upravljanje poslovnih procesov mora vključevati poslovna pravila (Blechar, 2007).

Osnovni namen magistrske naloge je poiskati rešitve, kako iz opisa poslovnega modela izvesti informatizacijo na enostaven način in brez nepotrebnih usklajevanj. Na slovenskem področju nisem zasledil razvoja programskih rešitev, ki bi v ospredje postavljal poslovna pravila, zato bom pri iskanju rešitve uporabil pristop razvoja programskih rešitev, ki ga predlaga Ross (2003). Problemska domena izdelava obračunske aplikacije je primerna za izdelavo poslovnega modela obračunavanja, zato predstavlja izziv razvoja na podlagi novega pristopa, s ciljem o zmanjšanju vrzeli med poslovnim modelom in informatizacijo programske rešitve.

Cilji magistrskega dela

Osnovni cilj magistrskega dela je prikazati probleme in rešitve pri informatizaciji programske rešitve na podlagi opisa poslovnega modela, ki v ospredje postavlja pomembnost poslovnih pravil. Opis poslovnega modela temelji na predpostavki, da je sestavljen iz treh področij: poslovni procesi (Kovačič, 2005), poslovna pravila (Ross, 2003), podatkovni model. Danes se uporabljajo objektno orientirani pristopi in vedno bolj so aktualni storitveno usmerjeni pristopi. Drugoten cilj magistrskega dela je raziskati povezavo med novim pristopom in obstoječimi pristopi oziroma metodologijami z vidika treh osnovnih področij poslovnih procesov, poslovnih pravil in podatkovnega modeliranja. Raziskal bom, kako so zapisane zahteve in kako se poslovni model informatizira ter na razumljiv način prikazal potencialne prednosti in slabosti.

Eden izmed ciljev je tudi izdelati prototipno aplikacijo za podporo obračunavanja in ugotoviti uporabnost orodja, ki omogoča vodenje repozitorija poslovnih pravil. Poslovno modeliranje bom izvedel s pomočjo obratnega inženirstva in ponovne dokumentacije že implementirane programske rešitve. V modulu obračun je zbrana poslovna logika za

izdelavo in izstavitev računa. Proces obračunavanja zbira podatke o strankah in storitvah ter jih preslika v končni izdelek - račun.

Na podlagi analize razvoja programske rešitve bom skušal odgovoriti na vprašanje o smiselnosti in uporabnosti novega pristopa, ki v ospredje postavlja poslovna pravila.

Delovna hipoteza:
Nov pristop razvoja programskih rešitev, ki ga predlaga Ross, omogoča enostavnejši in razumljivejši prehod iz poslovnega modela v informatizacijo programske rešitve. Orodje za modeliranje poslovnih pravil je učinkovito orodje in sredstvo za poslovno modeliranje.

Metoda dela

Teoretičen del vsebuje analiziranje primarnih in sekundarnih virov s področja poslovnega modeliranja in razvoja programskih rešitev, ki so objavljeni v člankih, revijah, raziskovalnih nalogah, svetovnem spletu.

V osrednjem delu magistrske naloge bom analiziral in predstavil, kaj so osnovne značilnosti novega pristopa razvoja programskih rešitev, ki ga predlaga Ross (2003). Prikazal bom, kakšne so povezave in vzporednice z drugimi področji modeliranja, poslovnimi procesi in podatkovnim modeliranjem v okviru drugih znanih pristopov: s strukturnim pristopom, objektnim pristopom in danes popularnim storitveno usmerjenim pristopom. V manjšem obsegu bom analiziral tudi vprašanja informatizacije omenjenih razvojnih pristopov in raziskal njihove lastnosti.

V praktičnem delu bom za programsko rešitev, ki podpira obračunavanje, izvedel celoten cikel razvoja programske rešitve, od zajema zahtev in izdelave poslovnega modela ter izvedbe izdelava prototipa programske rešitve na podlagi pristopa poslovnih pravil (angl. *busines rule aproach*). Na podlagi lastnih izkušenj iz podjetja, kjer sem zaposlen in delam kot razvijalec, poslovni analitik in preskuševalec, bom podal kritično oceno ustreznosti novega pristopa.

Pri opisu poslovnega modela bom uporabil orodje RuleExpress, ki ga predlaga združenje BRCommunity. Opis modela bo narejen v angleškem jeziku glede na to, da se programska koda piše v angleškem jeziku.

1 TEORETIČNA IZHODIŠČA

1.1 Razvoj programskih rešitev in poslovno modeliranje

Strukturni pristop k razvoju informacijskih sistemov je najstarejši, vendar še vedno pogosto uporabljen **proces razvoja informacijskih sistemov**. Zgleduje se po standardnih

postopkih razvoja tehničnih izdelkov, pri katerih si opravila v okviru aktivnosti sledijo zaporedno, med nekaterimi pa je možno ali celo zaželeno vzporedno opravljanje. Uvajati se je pričel v poznih 60. in v začetku 70. letih prejšnjega stoletja kot rezultat naporov, da se v razvoj informacijskih sistemov uvede red z doslednim izvajanjem analize in načrtovanja. Osnovni cilj je bil zmanjšanje stroškov izgradnje in uvajanja **informacijskih sistemov** s poudarkom na stroških njihovega vzdrževanja (Krisper, 2000).

Izraza razvoj programskih rešitev in razvoj informacijskih sistemov sta si po vsebini in pomenu zelo podobna. Informacijski sistem (angl. *information system*) je kombinacija informacijskih tehnologij, aktivnosti, uporabnikov, ki podpira poslovanje (SEI Report, 2010). Programska rešitev (angl. *software solution*) je računalniški program, prirejen zahtevam posameznega uporabnika (iSlovar, 2010), ki tudi lahko vsebuje vse elemente informacijskega sistema: procese, uporabnike, algoritme. Programska rešitev je lahko del celotnega informacijskega sistema in obratno, informacijski sistem si lahko predstavljamo kot množico programskih rešitev. V nadaljevanju bom uporabljal izraz programska rešitev, razen v primerih ko se bo le-ta nanašal na sistem informacijskih tehnologij, kjer bom uporabljal izraz informacijski sistem.

Proces razvoja programske rešitve (*angl. software life cycle*) je logično zaporedje aktivnosti za razvoj programske rešitve, ki običajno poteka v korakih po aktivnostih (FOLDOC, 2010): analiza zahtev, načrtovanje, implementacija, testiranje ali validacija in vzdrževanje. Običajno se pri analizi zahtev pripravijo uporabniške oziroma poslovne zahteve, zberejo se informacijske potrebe, ki so podlaga za podrobno načrtovanje oziroma tehnični načrt rešitve. V fazi implementacije se na podlagi podrobno razdelane rešitve implementira programska rešitev. Na koncu se programska rešitev validira oziroma preveri, ali deluje v skladu z zahtevami na testnih primerih.

Skozi čas se je oblikovalo nekaj razvojnih modelov oziroma **pristopov razvoja programskih rešitev**, ki se razlikujejo glede na zaporedje, povezovanje in kombiniranje izvajanja osnovnih aktivnosti analize, načrtovanja, implementacije in testiranja. Poznamo tri osnovne razvojne modele: zaporedni oziroma slapovni razvoj, spiralni in prototipni razvojni model (FOLDOC, 2010). V posebno kategorijo lahko umestimo sklop agilnih razvojnih modelov, sem sodijo: ekstremno programiranje, model RAD (angl. *rapid aplication development*) in model SCRUM, kjer je pri vseh poudarek na produktivnem in učinkovitem sodelovanju vseh deležnikov razvoja in pridobivanju povratnih informacij iz spremljanja napredka razvoja rešitve. Lahko pa se tudi odločimo za razvoj programske rešitve na podlagi interativno inkrementalnega modela, kjer posamezne zaključene funkcionalnosti dograjujemo postopoma (Centers for medicare & medicaid service, 2008).

Pri razvoju programskih rešitev je potrebno omeniti **formalizirane metode**. Formalizirane metode se uporabi, ko imamo probleme strukturirane do podrobnosti. Pristop je matematično korekten opis problema na nivoju analize zahtev, specifikacije in načrta.

Primeri formaliziranih metod so: Petrijeve mreže, B-metoda, avtomatična dedukcija ali drugi avtomati. V splošnem so formalne metode izvedljiva specifikacija. S problemom formalizacije razvoja programske rešitve se ukvarja področje modelno vodenega pristopa in jezik OCL (angl. *object coinstraint language*).

Pri formalizaciji ali strukturiranju problemov ne smemo pozabiti na **modelno vodeni razvoj** (angl. *model driven development,* okr. MDD) oziroma razvoj programske opreme na podlagi modelov, ki je ena izmed novejših smernic razvoja programske opreme. V ožjem smislu modelno vodeni razvoj pomeni izdelavo modelov, iz katerih se posredno ali neposredno dobi končne programske rešitve kot zamenjavo za pisanje klasične programske kode. V širšem smislu pa modelno vodeni razvoj zajema več podobnih filozofij, celovitih pristopov, dejanskih orodij in raznih drugih splošnih konceptov na temo razvoja na podlagi modelov (Kleppe, Warmer, Warmer, & Bast, 2003). Obstaja več različic in definicij modelno vodenega razvoja, od katerih je najbolj priznana, standardizirana in celovita različica modelno vodene arhitekture. Poslovni analitik je tisti, ki mora izpeljati prenovo sistema ali izdelati nov sistem. V procesu izdelave mora določiti, katero metodologijo programske rešitve bo uporabil in kako bo sistem integriral, kakšen bo poslovni model.

1.1.1 Cikel razvoja programske rešitve

Cikel razvoja programske rešitve (angl. *systems development life cycle*) je širši pojem od procesa razvoja programske rešitve, ki je del celotnega razvojnega cikla. Opredelimo ga lahko kot proces izdelave ali prenove sistema, modela in metodologije, ki ga deležniki uporabljajo pri razvoju informacijskega sistema. Cikel razvoja programske rešitve lahko podpira več različnih metodologij razvoja programskih rešitev. Cikel izvaja in uporablja poslovni analitik za razvoj informacijskih sistemov, vključno z zahtevami, testiranjem in šolanjem (FOLDOC, 2010).

1.1.2 Informatizacija in avtomatizacija

Informatizacija pomeni uvedbo informacijske tehnologije v delovni proces (iSlovar, 2010). V okviru magistrske naloge bom pojem povezoval kot uvedbo poslovnega modela preko razvoja v delujočo programsko rešitev. Pri uvedbi je seveda potrebna odločitev, v kateri tehnologiji bo programska rešitev delovala. Po drugi strani pa avtomatizacija govori, kako uporabiti stroje ali orodja, ki nam olajšajo prehod oziroma uvedbo programske rešitve v delovanje.

1.1.3 Metodologije razvoja programskih rešitev

Metodologija razvoja programske rešitve (angl. *software development methodology*) je ogrodje, ki se uporablja, da se proces razvoja programske rešitve izvaja načrtovano, strukturirano in kontrolirano. Vključuje določene opredelitve izdelkov, ki nastanejo tekom izvajanja. Nekatere metodologije so bolj primerne za večje projekte ali programske rešitve, druge za bolj specifične projekte (Centers for medicare & medicaid service, 2008).

V Tabeli 1 je kronološko prikazan seznam nekaterih metodologij oziroma pristopov razvoja programskih rešitev.

Tabela 1: Kronologija metodologij razvoja programskih rešitev

Desetletje	**Metodologija**
1970	Strukturno programiranje od leta 1969
1980	Metodologija strukturnega pristopa anlize in načrtovanja (angl. *structured systems analysis and design methodology*, okr. SSADM) od leta 1980
1990	Objektno orientirano programiranje (angl. *object oriented programing*, okr. OOP) je bilo razvito v zgodnjih 1960-ih letih in dokočno uveljavljeno v letih 1990
	Razvoj hitrih programskih rešitev (angl. *rapid aplication development*, okr. RAD) se je razvilo po letu 1991
	SCRUM razvojni model se je razvil po letu 1990
	Ekipni proces razvoja
2000	Ekstremno programiranje se je razvijalo po letu 1999
	RUP (angl. *rational unified process*) od leta 1998.
	Agile Unified Process (AUP) od leta 2005
	Integrated Methodology (QAIassist-IM) od leta 2007

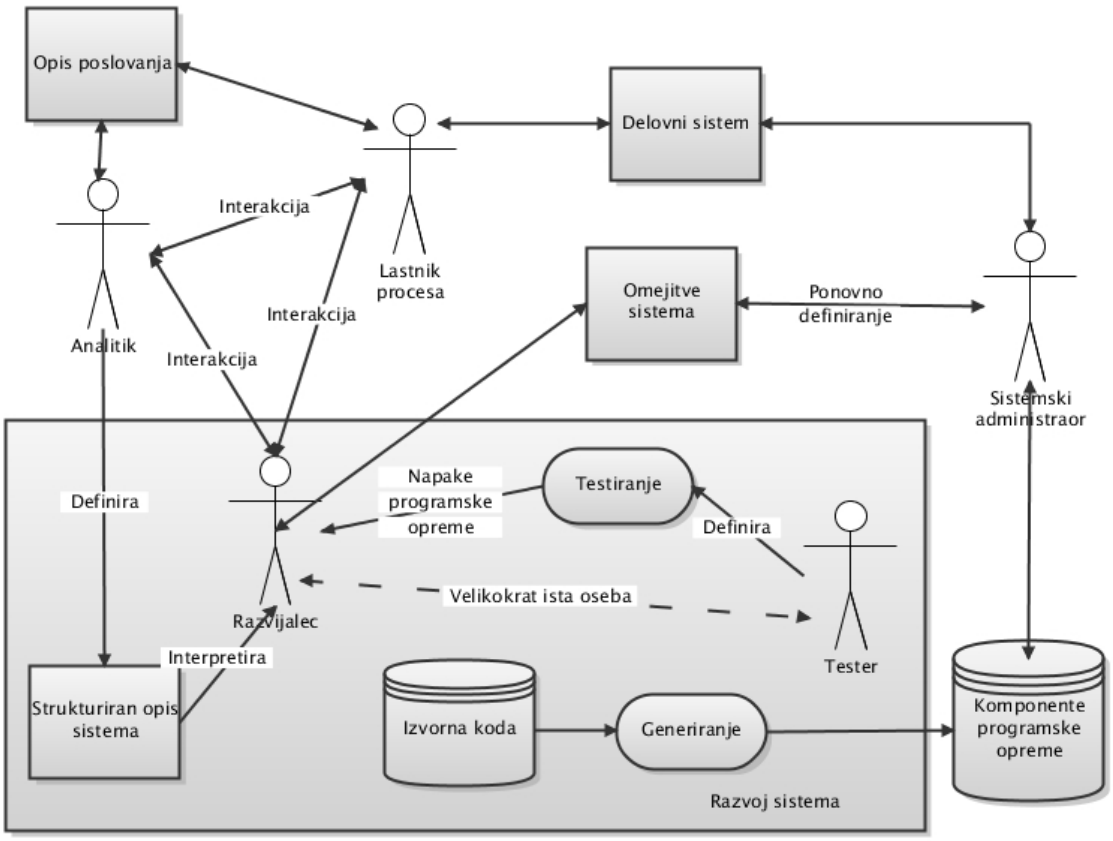
Vir: *Z. Qin, J. Xing, X. Zheng, Software architecture, 2008, str. 2.*

Danes se proces razvoja informacijskih sistemov redkeje izvaja, zato ker se lahko uporabniki odločijo za uporabo že obstoječih programskih rešitev, ki jih programske hiše ali spletni giganti ponujajo na trgu, ali pa uporabijo le del rešitve v novem, prenovljenem ali izgrajenem sistemu. Zato je cikel razvoja programske rešitve lahko zelo razpršen. Na Sliki 1 je prikazan razvoj programskih rešitev, kot ga je prikazal Krstov (2006).

S Slike 1 je razvidno, da je vpletenih najmanj 5 akterjev, ki morajo sodelovati s svojim strokovnim znanjem. Programska koda in komponente programske opreme so shranjene v repozitoriju, kjer se izvajajo.

Pred razvojem sistema oziroma programske rešitve mora analitik v sodelovanju z lastnikom poslovnega procesa opisati poslovanje. Poslovni analitik izdela strukturiran opis sistema oziroma programske rešitve, kadar so vsi problemi, ki nastanejo pri opisu poslovanja, razrešeni. Pri opisovanju poslovanja in strukturiranju problemov si lahko pomagamo z različnimi modeli in modelirnimi jeziki.

Slika 1: Razvoj programskih rešitev



Vir: L. Krstov, Modeliranje poslovnega pravila v modelih informacijskega sistema, 2006.

Modelirni jezik je umeten jezik, ki ga uporabljamo za opis sistema delovanja, izražanje informacij ali znanja v strukturi, ki je znana ali predpisana s pravili. Pravila uporabljamo za interpretacijo pomena komponent v strukturi. Modelirni jezik je lahko grafičen ali tekstovni. Grafični modelirni jeziki uporabljajo diagramske tehnike, kot so diagram podatkovnih tokov, BPMN (angl. *business process modeling notation*), IDEF (angl. *integration defiinition*), UML (angl. *unified modeling language*) (He, Ma, Shao & Li, 2007).

Proces izdelave poslovnega modela za programsko rešitev imenujemo **poslovno modeliranje** (angl. *business system design*). Poslovni model v magistrskem delu je mišljen kot model informacijskega sistema in njegovih elementov: poslovnih pravil, procesov in podatkov, ki so potrebni za informatizacijo programske rešitve. Pojma poslovni model ne smemo mešati z opisom poslovnega sistema, ki prinaša dodano vrednost podjetju in je del strategije podjetja. V tuji literaturi poslovno modeliranje v večini primerov spada pod področje modeliranja poslovnih procesov. Več o modeliranju poslovnih procesov bom opisal v poglavju 1.3. V magistrski nalogi bom poslovno modeliranje razdelil na tri podpodročja: modeliranje poslovnih pravil, poslovnih procesov in podatkovno modeliranje.

1.1.4 Razvoj programskih orodij in tehnologij

Za razvoj programske rešitve, ne glede na metodologijo razvoja je koristno imeti orodje, ki ga lahko učinkovito uporabljamo pri izdelavi programske rešitve. Nekatera orodja nam pomagajo voditi razvoj programske rešitve, druga orodja nam poenostavijo izdelavo načrta programske rešitve in poslovno modeliranje, kot npr. UML (angl. *unified modeling language*). V zadjem času so na voljo orodja za pisanje programske kode oziroma implementacijo programske rešitve, ki jim pravimo integrirana razvojna okolja (angl. *integrated development environment,* okr. IDE). Prav tako poznamo različna orodja za podoporo testiranju. Na voljo imamo tudi tako imenovana celovita orodja CASE (angl. *computer aided software environment),* ki nam pomagajo pri celotnem procesu razvoja programske rešitve. Lahko uporabimo tudi orodja, ki podpirajo formalizirane metode (MDD). Vsa omenjena orodja v kočni fazi pripeljejo do tega, da je program ali aplikacija napisana v določenem programskem jeziku. Po drugi strani pa se moramo zavedati, da s pomočjo programskih jezikov podajamo računalniku navodila, kako naj se program izvaja.

Za **podajanje navodil računalnikom** so se v zgodovini uporabljali različni programski jeziki. Na začetku se je računalnikom podajalo navodila v **strojnem jeziku**. Prva izboljšava je bil zbirni jezik, ki je predstavljal nivo nad strojnim jezikom. **Zbirni jezik** ima na voljo nabor ukazov, ki se preslikajo v strojno kodo. Po zbirnem jeziku so se pojavili **programski jeziki tretje generacije**. Izdelovanje programov je postalo enostavnejše in je postalo poznano širši množici ljudi. **Opearacijski sistemi**, ki predstavljajo nivo nad strojno opremo, so pripomogli za lažje delo s strojnimi napravami preko standardiziranih funkcij. Prevajalniki, ki so pravajali programsko kodo, so se poenostavili. Naslednji velik preskok je spremenil način razmišljanja iz strukturnega proceduralnega programiranja v **objektno usmerjeno programiranje** (izdelovanje programov in programskih rešitev). Dobra objektno usmerjena programska koda je za razliko od strukturirane veliko bližje človeškemu načinu razmišljanje (Greenfield & Short, 2004).

Od objektno usmerjenega pristopa naprej se je pojavilo kar nekaj pomembnih in uporabnih idej na temo možnosti ponovne uporabe, organizacije programske kode obsežnih projektov, standardizacije prijemov in podobno: uporaba komponent, načrtovalski in drugi vzorci, porazdeljeno procesiranje, večnivojski modeli, ogrodja, napredna orodja in podobno. Pomembna je postala tudi **vmesna programska oprema** (angl. *middleware*), ki povezuje različne programske komponente z aplikacijami. V splošnem se danes pri razvoju programske opreme teži k uporabi že izumljenih in v praksi preizkušenih konceptov ter vnaprej izdelanih komponent. Pisanje programov od začetka je le še stvar entuziastov, informacijska podjetja si tega ne morejo več privoščiti, saj se kompleksnost programske opreme povečuje, čas in dovoljeni stroški razvoja pa stalno zmanjšujejo (Vidrih, 2008).

Vzporedno z razvojem zapisa, v katerim računalniku dajemo navodila, je računalniška znanost napredovala tudi v drugih vidikih. Tako smo iz začetnega strojno usmerjenega

obdobja uporabe na strojno opremo vezanega strojnega in zbirnega jezika prešli na aplikacijsko usmerjeno obdobje z od strojne opreme neodvisnimi programskimi jeziki tretje generacije do operacijskih sistemov in objektno usmerjenih jezikov, kar je skupaj omogočilo razvoj širšega kroga obsežnejših aplikacij. Naslednje veliko obdobje, katerega konec se morda približuje, je poslovno-celovito usmerjeno obdobje, ki veliko manjših neodvisnih aplikacij zamenjuje s konceptom celovite povezane informacijske rešitve za celotno poslovanje. Pojavi se mnogo novih dobrih idej, kot so komponentno usmerjeni razvoj, vzorci, porazdeljeno procesiranje, vmesna programska oprema ter arhitekture in drugi koncepti za podporo izgradnji celovitih informacijskih rešitev (Vidrih, 2008).

Razmišljanje, da je možno razvoj programske rešitve dvigniti na še višji abstraktni nivo ni tako tuje. Dejtvo je, da je tehnologijo, ki je na voljo, potrebno izkoristiti. Danes imamo boljšo tehnologijo: močnejšo strojno opremo (angl. *hardware*), boljšo komunikacijo in programsko opremo kot kadarkoli v zgodovini. Na trgu se že pojavljajo orodja za novo generacijo programskega jezika, kot npr. orodja upravljanje poslovnih pravil. Poslovni uporabniki bodo s pomočjo nove generacije programskih jezikov lažje razmišljali, kako podpreti poslovne procese in upoštevati pravila podjetja ter imeli možnost dogovarjanja in modeliranja v naravnem jeziku. Poslovni svet narekuje obnašanje in omejitve izvajanja informacijskega sistema. Potrebno je tudi omeniti, da nekatere programske rešitve migrirajo na svetovni splet s pomočjo storitev v oblaku (angl. *cloud computing*) in s storitveno usmerjeno arhitekturo (angl. *service oriented architecture*, okr. SOA).

1.2 Poslovne zahteve in poslovni model

Pri razvoju nove programske rešitve ali pri prenovi je potrebno najprej imeti poslovne potrebe ali opis željenega poslovnega modela. Poslovne zahteve, ki so izražene potrebe, zapiše poslovni analitik. Rezultat poslovnega modeliranja je poslovni model ali specifikacija. Poslovne zahteve se lahko tudi spreminjajo, zato je pomembno upravljanje z zahtevami tako pri razvoju programske rešitve, kot tudi kasneje (Ross, 2003). V nadaljevanju sledi opis zahtev in njihov zajem.

1.2.1 Opredelitev poslovne zahteve

Zahteva je izjava, ki identificira sposobnosti ali funkcije, ki jih potrebuje sistem, da se izpolni potreba stranke. Funkcionalne zahteve opredeljujejo kaj, kako dobro in pod kakšnimi pogoji je treba enega ali več vhodov pretvoriti v enega ali več realizacij glede na mejo, da bi zadovoljili strankine potrebe. Standard CMMI (angl. *capability maturity model integration)* (Software engineering institute, 2006) pravi, da je zahteva: 1) pogoj ali zmožnost, ki jih potrebujejo uporabniki za reševanje problema ali doseganje cilja; 2) pogoj ali zmožnost, da izdelek izpolnjuje pogodbe, standarde, ali specifikacijo. Zahteve morajo sporočati, kaj mora narediti sistem, vendar pa ne sme določati, kakšna naj bo programska rešitev v tehnološkem smislu.

Delitev poslovnih zahtev glede na uporabnike, ki bodo zapisane zahteve pri svojem delu (razvojem ciklu) uporabljali, je naslednja:

- uporabniške zahteve so zahteve, kot jih vidi končni uporabnik oziroma naročnik programske rešitve;
- poslovne zahteve so zahteve, ki so rezultat podrobne analize in jih mora sistem zagotoviti za podporo poslovnim procesom in normalno poslovanje;
- tehnične zahteve so podrobno opisane zahteve, ki izhajajo iz poslovnih zahtev in so običajno izražene z modelirnimi tehnikami, formulami ali pa so podrobno opisane v strukturiranem jeziku.

Uporabniške zahteve je potrebno določiti pred razvojno fazo, tudi pred študijo izvedljivosti ali konceptualno fazo analize projekta. Posamezne faze izdelave zahtev se lahko razdeli na opisovanje (zbiranje, razumevanje, pregledovanje in artikuliranje potrebe zainteresiranih strani), analizo (preverjanje doslednosti in popolnosti), specifikacijo (dokumentiranje zahtev) in potrjevanje (Bahill & Dean, 2009).

Kaj loči dobre zahteve od slabih? Organizacija IEEE (angl. *Institute of electrical and electronics engineers*) pravi, da morajo biti zahteve nedvoumne, popolne, pravilne, sledljive, spremenljive, razumljive, preverljive ter ovrednotene na pomembnost in stabilnost. Bahill (2009) pravi, da načrtovanje testov med pisanjem programske kode vodi v manjše število napak. Zato mora imeti vsaka zahteva proceduro (test) za preverjane. O kvaliteti zahtev so veliko pisali tudi drugi avtorji (Hooks & Farry, 2001; Young, 2001).

Poslovni analitiki se med drugimi želijo prepričati, da definirajo aplikacijo na takšen način, ki ustreza potrebam končnih uporabnikov. To pomeni, da morajo dokumentirati prave zahteve, pravilno poslušati uporabnike in pridobiti povratne informacije stranke. Hkrati morajo posredovati celoten sklop jasnih zahtev tehničnim arhitektom in programerjem. Če analitik nima na voljo dovolj orodij in sposobnosti, je verjetnost, da bo dokumentiral napačne zahteve, velika. Čas, ki ga lahko zapravi poslovni analitik za dokumentiranje nepotrebnih zahtev, ne vpliva samo na zajem zahtev, to vpliva tudi na ostale razvojne cikle. Čas se izgublja s tem, ko morajo programerji kodirati nepotrebne zahteve, testerji izdelovati in dokumentirati nepotrebne teste. Strokovnjaki ocenjujejo, da je 10 do 40 % funkcij v novih aplikacijah nepotrebnih ali ostanejo neizkoriščene. Do prihranka dela bi prišlo, če bi zmanjšali nepotrebne funkcije že za eno tretjino (International institute of business analysis, 2006).

Zahteve je najbolje zapisati z izjavami, ki so izražene v naravnem jeziku. Za dodatno razlago in predstavitev delovanja programske rešitve se priporoča diagramska tehnika primerov uporabe, ki jo omogoča modelirni jezik UML (angl. *unified modeling language*). Kako se zahteve izražajo, je odvisno od razvojne metode (Bahill & Dean, 2009).

1.3 Model poslovnih procesov

Pri razvoju programske rešitve opisu uporabniških zahtev in znanim potrebam sledi opis poslovnih procesov in modeliranje poslovnih procesov. Zato bom v nadaljevanju na kratko opredelil poslovne procese in modeliranje poslovnih procesov.

1.3.1 Poslovni procesi

Poslovni procesi v podjetjih so velikokrat nepregledni in neprilagodljivi ter s tem obremenjujoči v poslovnem ter informacijskem pogledu. Procesi potekajo skozi različne organizacijske enote in so obremenjeni z vsemi problemi, ki nastanejo ob prehodu iz ene organizacijske enote v drugo. Splošne pomanjkljivosti izvajanja poslovnih procesov v večini organizacij so neenotnost, nepoznavanje celotnega procesa s strani izvajalcev, podvajanje dela ter razmeroma dolgotrajno čakanje na birokratske aktivnosti. Takšno stanje je neprimerno, zato je treba poslovne procese najprej poenotiti, včasih tudi na novo opredeliti ali pa jih radikalno spremeniti, kar imenujemo prenova poslovnih procesov (Kovačič, 2005).

V splošnem je proces:

- celota del, delovanja za dosego kakšnega cilja (SSKJ, 2010);
- zaporedje operacij, ki so izvedena za doseg nekega cilja (Dictionary.com, 2010).

Za poslovni proces se pojavlja veliko definicij različnih avtorjev, med njimi so tudi naslednje:

- poslovni proces opredeljujemo kot skupek logično povezanih izvajalskih in nadzornih postopkov ter aktivnosti, katerih posledica oziroma izid je načrtovani izdelek ali storitev (Jacobson, 1999);
- poslovni proces je zbirka aktivnosti, ki prejme enega ali več tipov vhodov in kreira izhod, ki stranki prenese neko vrednost (Baloh, Indihar Štemberger, & Vrečar, 2002);
- poslovni proces je set logično povezanih nalog, ki se izvajajo s ciljem doseči poslovni rezultat (Davenport & Short, 1990);
- poslovni proces je organizirana skupina povezanih dejavnosti, ki delujejo skupaj, da bi ustvarile rezultat, ki ima vrednost za stranko (Hammer, 2003);
- poslovni proces je skupek aktivnosti, ki kot vložek sprejemajo dane vire v podjetju in zagotavljajo rezultat, ki je v skladu s poslovnimi cilji podjetja (Harrington, 1991).

Če povzamemo in združimo naštete definicije, velja, da je poslovni proces sklop aktivnosti, dejavnosti in opravil, ki dosegajo želene poslovne rezultate (Medeot, 2007). Vhodne elemente preko izvajanja procesa pretvorimo v izhodne elemente, ki so rezultat izvajanja procesa, kakor je razvidno na Sliki 2.

Slika 2: Shematski prikaz procesa

1.3.2 Modeliranje poslovnih procesov

Model poslovnega procesa je predstavljen v grafični obliki in ima vhode, izhode in dogodke, ki prožijo izvajanje procesa. Obstajajo različni nivoji podrobnosti in različne stopnje abstrakcije. Procese in podprocese na ravni izvajanja pojasnjujejo postopki in delovni procesi.

Modeliranje poslovnih procesov (angl. *business proces modeling*, okr. BPM) lahko definiramo kot proces formaliziranja in predstavitve poslovnih procesov v neki združbi oziroma poslovnem sistemu. Rezultat takšnega modeliranja so različni modeli, ki so uporabljeni za analizo dogajanja v poslovnem sistemu. Modeliranje poslovnih procesov je v prvi vrsti aktivnost, ki se dogaja na strateškem nivoju upravljanja združbe. Rezultat modeliranja je namenjen vodstvenemu kadru z namenom boljšega upravljanja in organiziranja dela. Gre torej za poslovne uporabnike, ki dogajanje v združbi predstavijo v obliki procesnih modelov. Po drugi strani pa je modeliranje poslovnih procesov namenjeno tudi njihovi informatizaciji. Uspešna implementacija informacijske tehnologije za podporo poslovnih procesov je neposredno odvisna od dobrih procesnih modelov, ki predstavljajo osnovo za uspešno analizo in izdelavo specifikacij informacijskega sistema.

Poslovna logika oziroma poslovna pravila se večinoma smatra kot del procesa (Standeker, 2010). V magistrski nalogi bom področja poslovnih procesov in poslovnih pravil obravnaval ločeno.

Za modeliranje poslovnih procesov se uporablja različne modelirne ali dokumentacijske tehnike. Naštete so samo najbolj pogosto uporabljene:

- Petrijeve mreže kot formalizirane in popolnoma strukturirane tehnike. So formalen matematični jezik, ki ga lahko uporabljamo za opis diskretnih sistemov.
- BPMN (angl. *business process modeling notation*) diagramska tehnika je grafična notacija za modeliranje poslovnih procesov. Skrb za razvoj notacije trenutno nosi konzorcij OMG (angl. *object modeling group*).
- Diagrami EPC (angl. *event driven proces chain*) je razvita tehnika v sklopu pristopa ARIS (angl. *architecture of integrated information systems*) za potrebe ERP (angl. *enterprise resource planning*) sistema za SAP podjetje.
- Jezik UML, ki je zelo bogat z notacijami in številnimi elementi.
- Diagram poteka (angl. *flowchart*), ki je ena najenostavnejših in tudi široko uporabljenih diagramskih tehnik.

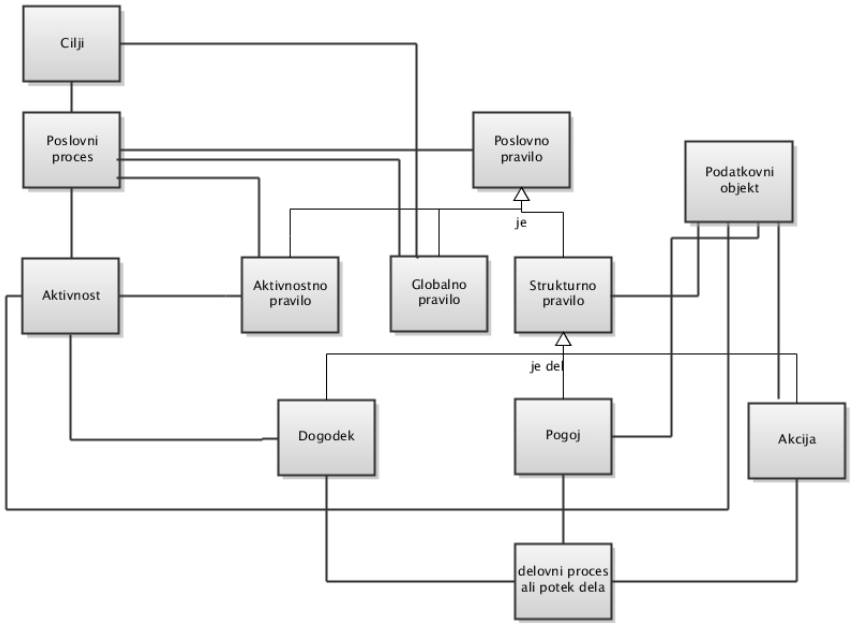
Veliko se govori o modeliranju in razvoju poslovnih procesov, ki obljubljajo bolj dinamično, fleksibilno organizacijo in izboljšave delovnih procesov. V ta namen je bil razvit **kognitiven pristop** kontrole in dinamike upravljanja procesov. Poslovni procesi niso predstavljeni proceduralno oziroma zaporedno, ampak so predstavljeni kot množica poslovnih pravil, ki sami določajo potek. Takšen pristop ponuja boljšo prilagodljivost poslovnemu sistemu v primerjavi s kompleksno procesno porazdelitvijo in povezavami. Proces se lahko izvaja v realnem času na podlagi odločitev, ki jih usmerjajo pravila. Novost se v literaturi se naslavlja kot **prehod iz procesne logike v poslovno logiko** (Wang & Wang, 2006)**.**

1.4 Poslovna pravila

Poslovna pravila se lahko navezujejo na poslovne procese na tri možne načine, in sicer (Tibco, 2006) :

- poslovna pravila lahko prožijo proces;
- poslovna pravila so lahko algoritem za odločanje znotraj posamezne aktivnosti v procesu;
- poslovna pravila se lahko uporabi v namene opazovanja oziroma nadzorovanja, na podlagi katerega lahko proži izboljšanje oziroma uredi obstoječe stanje;

Slika 3: Metamodel aktivnosti in poslovnih pravil



Vir: A. Kovačič, Business renovation: business rules (still) the missing link, 2004.

Slika 3 prikazuje metamodel poslovnih pravil. Vse relacije med entitetami so m:n. Globalna pravila so agregirana pravila, ki odražajo dinamiko poslovnih procesov in sledijo ciljem poslovnega sistema. Na desni strani je poudarek na strukturiranih pravilih, ki se v

programski kodi izvajajo in morajo biti matematično definirana. Na Sliki 3 je tudi razvidna povezava med poslovnim svetom in dejansko implementacijo informacijskega sistema oziroma delovnih procesom. Jasno je, da so povezovalni elementi poslovna pravila različnih kategorij (delitev v poglavju 1.4.2.). Slika 3 tudi prikazuje, kako različna poslovna pravila se uporabljajo v poslovnem procesu in aktivnostih.

1.4.1 Opredelitev poslovnih pravil

Veliko je različnih definiciji poslovnih pravil, ker lahko na poslovna pravila gledamo z različnih vidikov, kot so poslovni vidik, informacijski vidik in vidik omejitve poslovanja. V nadaljevanju magistrske naloge je naštetih nekaj definicij, opredelitev in kategorizacij poslovnih pravil. Za lažjo razlago so ponekod podani primeri.

Ena od definicij poslovnih pravil pravi, da so to postopki, s katerimi so opisani obnašanje, delovanje in odločanje posameznikov v podjetju. Poslovna pravila so opredeljena s poslovno politiko podjetja ali globalnimi poslovnimi pravili ter s pravnimi normami in regulativami. Poslovna pravila so namenjena uporabi izvajalcev in odločevalcev aktivnosti poslovnih procesov (Kovačič, 2005).

Poslovno pravilo je lahko tudi izjava ali stavek, ki definira ali omejuje določen vidik poslovanja nekega organizacijskega sistema z namenom, da vanj vpelje ustrezno poslovno obnašanje oziroma ga nadzira ali nanj vpliva. Poslovna pravila so zahteve, ki izhajajo iz poslovnih ciljev in usmeritev organizacijskega sistema (Krisper, 2000).

Poslovno pravilo iz informacijskega stališča je izjava, ki definira in omejuje določen vidik poslovanja neke organizacije z namenom, da vanje vpelje ustrezno poslovno obnašanje oziroma ga nadzira in nanj vpliva (Business Rule Group, 2010).

Poslovno pravilo je koncept (predikat), ki je predstavljen z izrazom, dejstvom ali pravilom. Po sistemski definiciji je poslovno pravilo atomski element uporabne poslovne logike napisan deklerativno (Ross, 2003).

Poslovna pravila predstavljajo osnovno znanje neke organizacije, ki ga imajo ali udeleženci poslovanja ali oblikovalci politike. Usmerjajo izvajanje in opredeljujejo strukturo ter organizacijo poslovanja (Krstov, 2006).

Poslovna pravila (Vantheienen, 2007) definirajo vsebino in omejitev poslovnih konceptov, reakcije na poslovne dogodke, omejitve in predpogoje za aktivnosti, pravice in obveznosti in podobno. Zato poslovna pravila vodijo in omejujejo poslovne procese, in sicer ne le kalkulacij in pravne veljavnosti, ampak tudi zaporedja in časovno usklajenost aktivnosti.

Iz zgoraj opisanih definicij poslovnih pravil lahko povzamemo, da razlikujemo dva načina predstavitve poslovnih pravil. V prvi vrsti so poslovna pravila v obliki deklerativnih stavkov, ki jih uporabljajo poslovni analitiki in poslovni uporabniki. Poslovna pravila so izražena v strukturiranem jeziku v konceptualnem katalogu ali slovaju. Druga vrsta opisa pravil so poslovna pravila z informacijskega ali tehničnega stališča, ki so v končni fazi prevedena poslovna pravila, ki se implementriana v programski kodi in se izvajajo v določenem okolju. Podobno kot opredelitev poslovnega pravila poznamo tudi več različnih delitev poslovnih pravil.

1.4.2 Delitev poslovnih pravil

Zaradi velikega števila definicij poslovnega pravila obstaja veliko različnih delitev poslovnih pravil. V tem poglavju bom izpostavil nekaj različnih pogledov na poslovna pravila.

Poslovno pravilo spada v eno izmed naštetih kategorij (Business Rule Group, 2010):

- Definicija izraza (angl. *term*) ali pojma: je najosnovnejši element, ki ga opišemo v naravnem jeziku. Opisuje vsebino in kako ljudje razumejo izraz. Običajno je opisan v poslovnem slovarju ali konceptualnem katalogu.
- Dejstva (angl. *fact*), ki povezujejo izraze: določajo strukturo organizacije in relacije med izrazi. Izrazijo se lahko v naravnem jeziku kot stavki ali z grafično predstavitvijo.
- Omejitve (angl. *constraint or action assertion*) so definirane omejitve poslovanja, ki so povezane z omejitvami pri vnašanju podatkov.
- Izpeljave (angl. *derivation*), ki določajo, kako se znanje transformira iz ene oblike v drugo obliko.

Poslovna pravila lahko opredelimo na treh nivojih (Kovačič, 2005), in sicer:

- globalna pravila: poslovna pravila kot del poslovne politike organizacije;
- aktivnostna pravila: poslovna pravila na ravni posameznih aktivnosti poslovnega procesa;
- strukturna pravila: poslovna pravila za izvajanje logike uporabniških programov in orodij za krmiljenje delovnih procesov.

Predstavljena klasifikacijska shema (Slika 4) temelji na predlogu Martina in O'della (1994), kjer so poslovna pravila razdeljena v 3 skupine: definicije, omejitve in izpeljave.

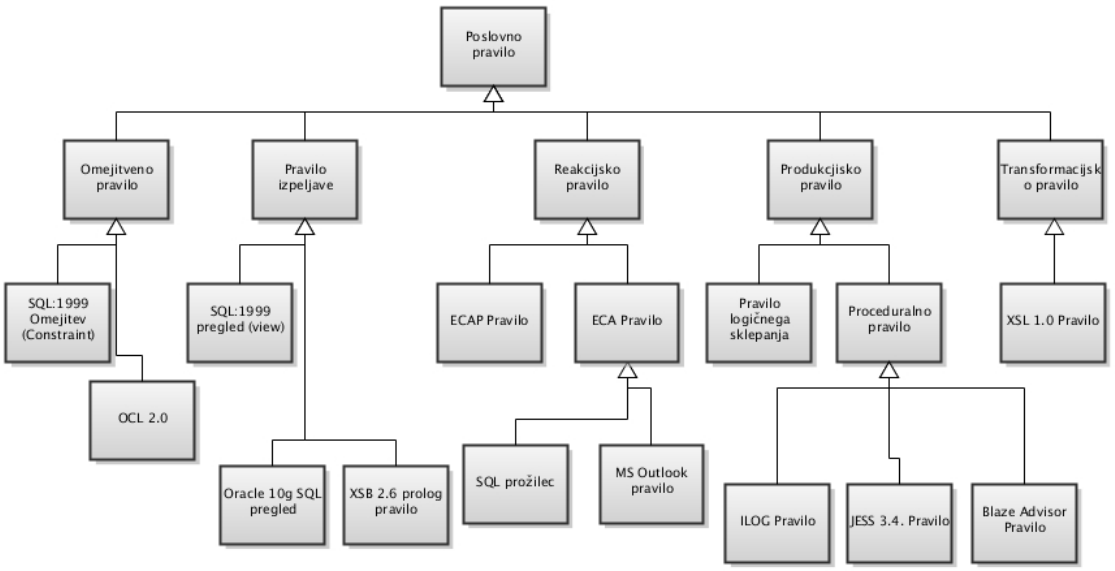
Slika 4: Delitev poslovnih pravil

Vir: J. Martin, J.O'dell, Object-oriented methods, 1994.

Ross (2003) poslovna pravila loči v 3 osnovne funkcionalne kategorije: omejitveno pravilo, izpeljava, prožilec.

Zanimiv je tudi pogled z informacijskega stališča (Slika 5) (OMG available specification, 2009), ki je nastal zaradi opisa in opredelitve produkcijskih pravil (angl. *production rule representation*, okr. PRR). S Slike 5 lahko tudi razberemo, kje v informacijskem sistemu so poslovna pravila informatizirana.

Slika 5: Tehnološka klasifikacija poslovnih pravil

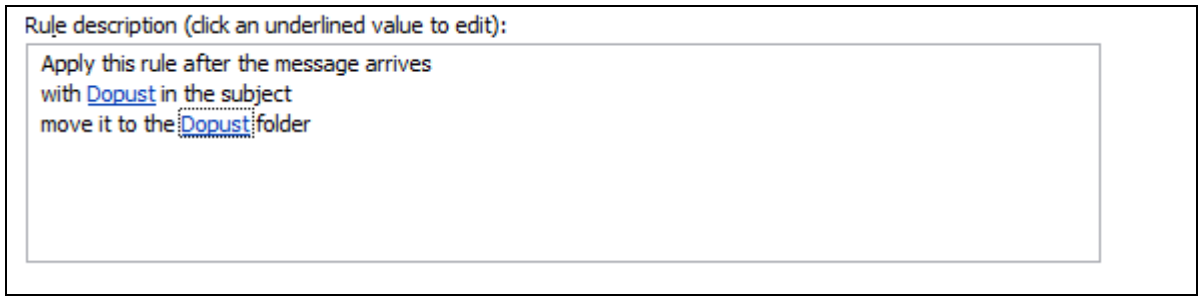


Vir: OMG interno gradivo, OMG available specification, production rule representation - PRR 1.0, 2009.

Specifikacija metamodela produkcijskih pravil je nastala iz različnih razlogov, ki so: i) potreba po definiciji produkcijskih pravil; ii) potreba po določitvi standardov in normativov za poslovna pravila, ki jih implementirajo različna podjetja v svojih rešitvah; iii) potreba po definiciji načina izvajanja poslovnih pravil.

Na Sliki 5 so tudi opisana druga pravila poleg produkcijskega pravila. Omejitveno pravilo in pravilo izpeljave se običajno informatizira z orodji v relacijskih podatkovnih bazah (npr. Oracle PLSQL orodja). Pravilo dogodek, pogoj in akcija (angl. *event, condition, action*, okr. ECA) je reakcijsko pravilo, katere implementacije lahko nadjemo tako v SQL (angl. *structure query language)* prožilcih, kot tudi v bolj poznanem orodju MS Outlook. XLS (angl. *extensible stylesheet language*) pravila so običajno koristna pri sinhronizaciji in transformacijah.

Primer 1: Pravilo MS Outlook, ki premakne sporočila v posebno mapo



1.5 Podatkovno modeliranje

Podatkovno modeliranje je način raziskovanja podatkov in podatkovne strukture. Podobno kot drugi modeli se lahko uporablja za različne namene: od konceptualnih (poslovnih), preko logičnih do fizičnih podatkovnih modelov. Pri objektno orientiranem pristopu je podatkovno modeliranje podobno izdelavi razrednemu diagramu. Pri podatkovnem modeliranju definiramo entitetne tipe, pri objektnem pa razrede. Entitenim tipom in razredom določimo atribute podatkov in medsebojne relacije ali asociacije (agragacije, generalizacije, dedovanja ...) (Agiledata.org, 2006).

Tradicionalno podatkovno modeliranje se razlikuje od objektnega v tem, da se fokusira na podatke. Po drugi strani lahko z objektnim pristopom opišemo obnašanje in podatke. Razvijalci, ki se ukvarjajo s podatkovnim modeliranjem, podatke strukturirajo veliko bolje kot razvijalci razrednih diagramov (Agiledata.org, 2006).

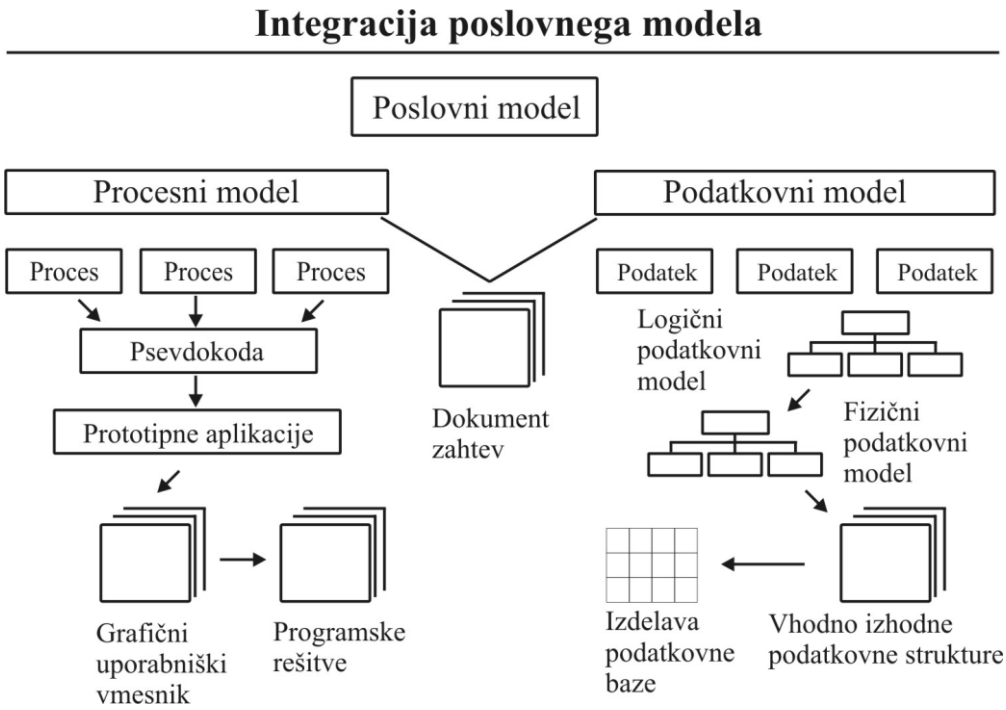
Podatkovni model lahko določa poslovna pravila na sledeč način: ima določene izraze in dejstva. Po drugi strani lahko prikazuje poslovno pravilo iz kategorije izpeljava (angl. *derivations*) in lahko definira omejitve (angl. c*onstrain or assertion*). Ne more pa definirati, da se lahko določena ažurirajo pod določenimi pogoji (Hey, 2004).

Podatki se povezujejo s poslovnim procesom. Poslovni proces preoblikuje podatke, pri tem pa se upošteva poslovna logika (kontrola).

Slika 6 prikazuje razvoj programske rešitve z vidika izvajanja aktivnosti modeliranja poslovnega procesa in podatkovnega modeliranja. Na levi strani se iz dokumenta o

zahtevah določi poslovni proces, ki se razdeli na podprocese. Napiše se psevdokoda oziroma poslovna pravila in se definira grafični uporabniški vmesnik, ki na koncu rezultira v programski rešitvi. Na desni strani Slike 6 so opisane aktivnosti od konceptualnega podatkovnega modela preko logičnega podatkovnega modela in fizičnega podatkovnega modela ter vsebinsko napolnjena podatkovna baza, v katero se preko izvorne strukture migrira podatke.

Slika 6: Razvoj programske rešitve po aktivnostih



Vir: Wikipedia, Business process modeling, 2010.

2 ZNAČILNOSTI PRISTOPA POSLOVNIH PRAVIL

V 2. poglavju bom zajel osnovne principe, nekatere povzetke in načela, ki jih poudarja nov pristop razvoja programskih rešitev na podlagi poslovnih pravil (Ross, 2003). V nadaljevanju bom pristop razvoja programskih rešitev na podlagi poslovnih pravil naslavljal kot **pristop poslovnih pravil (okr. PPP)** razvoja programskih rešitev ali pa samo **poslovni pristop**, kot ga naslavlja tudi Bajec (2003). Pristop poslovnih pravil, ki ga predlaga Ross (2003), nima jasno metodološko opredeljenega procesa kot druge metodologije npr. UML. Je osnovan na Zachmanovem ogrodju arhitekture podjetja in predlaga določena načela in izdelke, ki bi jih poslovni uporabniki, analitiki in informatiki morali spoštovati in imeti pri projektnem vodenju informacijskega projekta v prihodnosti (več o projektih v razdelku 2.1.5).

Osnovne ideje in koncepti novega pristopa razvoja programskih rešitev izvirajo iz sredine 90. let prejšnjega stoletja, mnoge tehnike in metodologije pa so bile testirane v poznih 90.

letih in v začetku drugega tisočletja. Zanimivo je, da pristop poslovnih pravilni nastal na podlagi nove generacije programske opreme ali novih tehnologij. Podpornik poslovnega pristopa je svetovno združenje **BRSolutions**, ki si prizadeva izboljšati poslovanje podjetij. **Združenje** vsebuje posameznike, ki imajo veliko izkušenj iz prakse. Njihov poglavitni cilj je ponuditi podjetjem najboljši pristop k razvoju programskih rešitev, ki vključuje avtomatizirane sisteme za razvoj programskih rešitev. **Poslovni pristop** predstavlja novo paradigmo pri **poslovnem modeliranju** (angl. *business system design*) in razvoju (angl. *development*). Obstajajo tudi sorodna združenja in skupnosti, kot so BRCommunity in BusinesRuleGroup, ki prav tako delujejo s ciljem ozaveščati ljudi o rešitvah s pomočjo poslovnih pravil.

2.1 Osnovna načela pristopa poslovnih pravil

Vodilo **poslovnega sistema** bi moralo biti poslovna potreba ali poslovna priložnost. Poslovneži bi morali poslovne potrebe izražati na diskreten in konkreten način. Z upoštevanjem teh načel v praksi se je potrebno lotiti novega pristopa izdelave poslovnih sistemov, ki vplivajo na vlogo **poslovnih analitikov** (angl. *business professional*) in **informatikov** (angl. *IT professional*).

Glede na to, da se v prvi vrsti pristop poslovnih pravil osredotoča na poslovni del problema, lahko proces imenujemo **poslovno analiziranje** ali **poslovno modeliranje**, katerega rezultat je **poslovni model**. Osnovno sredstvo, ki je postavljeno v ospredje poslovnega pristopa, je **poslovno pravilo**.

Osnovni načela poslovnega pristopa, ki se nanašajo na poslovna pravila, so:

- poslovna pravila naj bi bila zapisana in eksplicitna;
- poslovna pravila naj bi bila izražena v naravnem jeziku;
- poslovna pravila naj bi bila samostjna in ločena od programske kode, procedur in poslovnega procesa;
- poslovna pravila naj bi se gradila na dejstvih, dejstva naj bi bila predstavljena iz osnovnih izrazov;
- poslovna pravila naj bi vplivala na dinamiko sistema na željen način;
- poslovna pravila naj bi bila identificirana za pomembne poslovne faktorje;
- poslovna pravila naj bi bila dostopa pooblaščenim deležnikom;
- poslovna pravila naj bi bila zapisana samo enkrat;
- poslovna pravila naj bi specificirali ljudje, ki posedujejo ustrezno poslovno znanje;
- poslovna pravila naj bi se upravljala.

2.1.1 Osnovna vodila in opredelitev pristopa poslovnih pravil

Pristop na podlagi poslovnih pravil poudarja **poslovno voden pristop** (angl. *business driven aproach*) razvoja programskih rešitev, kjer so poslovna pravila osnovni element kontrole. Pomembnost poslovnih pravil je najlažje prikazati na praktičnih primerih.

Kaj pomeni poslovno voden pristop? Poslovni sistemi ne obstajajo zato, da bi upravljali in uporabljali določeno strojno in programsko opremo, ravno nasprotno, strojna in programska oprema obstaja zato, da podpira poslovni sistem. Vsak informacijski projekt bi moral koristiti poslovanju, seveda pa je to lažje napisati in težje realizirati.

Pristop poslovnih pravil ima organiziran načrt za poslovno voden pristop in določena poglavja, ki jim je smiselno slediti. Zanje velja:

- poslovni in informacijski projekti naj bodo združeni v en sam projekt, saj so povezani na eni strani zaradi potrebe po avtomatizaciji poslovanja, po drugi strani pa ima IT direkten vpliv na poslovanje.
- poslovni svet ima znanje za reševanje problemov, informatiki, lahko pomagajo pri razvoju rešitev s svojim znanjem o načrtovanju in implementaciji;
- poslovne zahteve naj bodo v aplikaciji enakovredne obveščanju o napakah;
- najprej se je potrebno vprašati o poslovnih zahtevah;
- zajemanje poslovnega znanja zahteva podrobnejše poglobitve poslovanja in zavezanost pravilnemu izrazoslovju, zato je potrebno poslovnim rešitvam omogočiti dovolj časa;
- pridobivanje informacij o poslovnem sistemu na pravem mestu ob pravem času;
- kompleksna poslovna vprašanja ali probleme je potrebno ustrezno strukturirati.

2.1.2 Pristop poslovnih pravil v Zachmanovem ogrodju

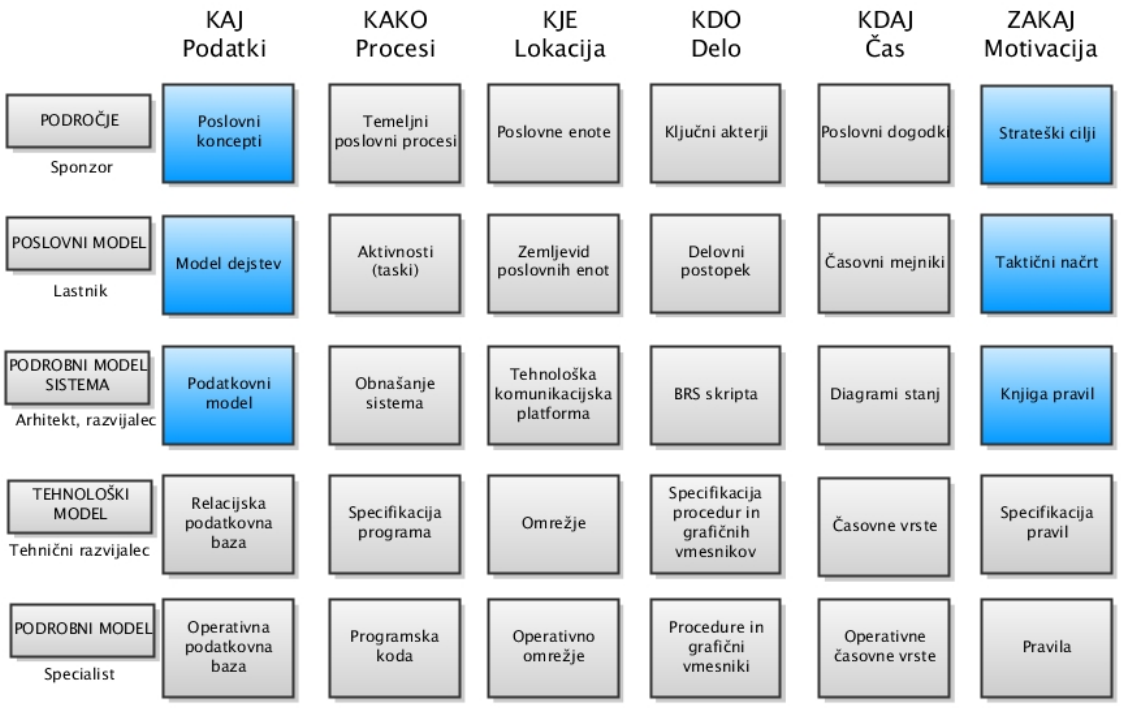
Pri poslovnem pristopu je poudarek na izdelavi poslovnega modela v sodelovanju z lastniki poslovnih procesov, operacijskimi vodji, ki skrbijo, da se poslovanje učinkovito izvaja, in strokovnjaki posameznega področja. Poslovni model mora biti predstavljen kot načrt (angl. *blueprint*), ki ga lahko berejo poslovni uporabniki in informatiki. Poslovni model mora biti dobro razdelan, da lahko kompleksne rešitve obvladujemo (Ross, 2003).

Poslovni model mora imeti ustrezno strukturo. Pristop poslovnih pravil se zgleduje po Zachmanovem ogrodju, predvsem se nanaša na prvo in drugo vrstico ogrodja (Slika 7). Po drugi strani pa se izdelki bolj nanašajo na delovanje kot na organizacijo podjetja. Povezava poslovnega pristopa in Zachmanovega ogrodja je na kratko opisana v Tabeli 2.

Tabela 2: Izvleček poslovnega pristopa iz Zachmanovega ogrodja

	Faktor	**Opis**
Motivacija	Zakaj	Izdelava vzpostavitvenega dokumenta oziroma kratka verzija načrta. Izdelek je taktični načrt. Opisuje cilje in sredstva, kako se bomo lotili in reševali probleme.
Funkcija	Kako	Izdelava zaporedja opravil in skupine opravil.
Struktura	Kaj	Lahko tudi naslavljamo kot podatkovni faktor. Izdelava poslovnega slovarja, konceptualnega kataloga in modela dejstev.
Ljudje	Kdo	Določanje organizacijskih odgovornosti in vlog.
Časovnica	Kdaj	Časovnica prikazuje zgodovino in upravljanje osnovnih konceptov.
Prostor	Kje	Idelava zemljevida povezav različnih poslovnih mest: komunikacija in transport.

Slika 7: Zachmanovo ogrodje



Vir: Interno gradivo BRSolution, Busines rule speak, 2010.

2.1.3 Ostale prednosti in značilnosti pristopa poslovnih pravil

Pristop na podlagi poslovnih pravil ima to prednost, da se **poslovna pravila izražajo na deklarativen način v naravnem jeziku**. To prednost je možno izkoristiti pri šolanju uporabnikov na način, da poslovna pravila spoznavajo z uporabo aplikacije. Namesto obvestila o napaki lahko prikažemo poslovno pravilo, ki smo ga prekršili, kar je prikazano s Primerom 2. V primeru, da zaposleni pri uporabi aplikacije poslovno pravilo prekrši, se mu na zaslon izpiše omenjeno pravilo.

Primer 2: Omejitveno poslovno pravilo

Hitro naročilo ne sme presegati pet izdelkov.

Na poslovna pravila lahko gledamo tudi z vidika poslovne zahteve. Poslovni analitik definira novo poslovno pravilo, ki se zaposlenemu ali uporabniku programske rešitve prikaže na ekranu brez dejanske medsebojne komunikacije, sestankov, ozaveščanja in posebnega šolanja.

Pomembno vlogo pri novem pristopu ima **poslovni analitik**, ki je v prvi vrsti človek, ki rešuje poslovne probleme in pozna poslovanje do te mere, da lahko razloži delovanje poslovnega sistema.

2.1.4 Taktični načrt

Izdelek na strateškem ali taktičnem nivoju predstavlja osnovne cilje in globalna poslovna pravila za poslovno in programsko rešitev. Konceptualni pregled identificira, kakšna je taktika in kakšna so globalna poslovna pravila, ki jih potrebujemo za doseganje cilja ter tveganja. Taktični načrt (angl. *policy charter*) vzpostavlja pristop, kako se bomo lotili poslovnih problemov, poslovnih procesov, delovnih tokov in drugih izdelkov pri razvoju programske rešitve ali informacijskega sistema.

Taktičnemu načrtu lahko tudi rečemo načrt uresničitve poslovnih ciljev ali pa konceptualni načrt informatizacije. Pri iskanju slovenskega prevoda je najbližje izraz načrt bitke (angl. *battle plan*). Njegovo vsebino lahko razčlenimo v naslednje korake (Gladys, 1998):

Korak 1: Opredeli poslovne cilje v okviru projekta.
Vprašanje: Kakšne poslovne učinke želimo za področje doseči?
Korak 2: Določi taktiko, kako bomo zadostili poslovnim ciljem.
Vprašanje: Kaj je potrebno postoriti za vsakega izmed poslovnih ciljev?
Korak 3: Določi tveganja, ki jih lahko sprožijo taktične usmeritve.
Vprašanje: Kakšna so tveganja, če začnemo izvajati taktične aktivnosti?
Korak 4: Določi nadaljnje ukrepe, taktiko in politiko, ki zmanjšujejo tveganje.
Vprašanje: Kaj lahko naredimo, da zmanjšamo tveganje?
Korak 5: Ponavljaj korak 3 in 4, dokler tveganje ni sprejemljivo.
Vprašanje: Ali poslovanje lahko deluje nemoteno z določenim tveganjem?
Korak 6: Specificiraj politiko oziroma globalna poslovna pravila za taktiko.
Vprašanje: Ali se lahko taktika poistoveti s politiko? V nasprotnem primeru določi ustrezno politiko, če zahteva ni usklajena.

2.1.5 Kategorije projektov v prihodnosti

Projekti informatizacije poslovnih procesov v današnjem času po vsebini niso več izdelava informacijskega sistema, temveč prenova informacijskega sistema, saj vsak poslovni

sistem že ima informacijsko podporo svojim poslovnim procesom. Pri informacijskih projektih se zato srečujemo z dvema izzivoma: tehnološkim, ki pomeni integracijo nove informacijske rešitve v obstoječe okolje, in poslovnim, ki vključuje spremembe in optimizacijo poslovnega procesa zaradi možnosti nove informacijske rešitve (Keber, 2003).

Tabela 3: Kategorije projektov v prihodnosti

Angl. izraz	**Slovenski pomen**	**Motivacija za projekt**	**Predosti poslovnega pristopa**
Reengineering	Prenova poslovnih procesov	Poslovni procesi niso zastavljeni optimalno.	1. Poslovna pravila so pomemba pri ponovnem strukturiranju poslovnega problema in pripravi optimalnih rešitev. 2. Poslovna pravila podrobneje dopolnjujejo delovne tokove in poslovne procese s poslovno logiko.
Revitalization	Upravljanje s spremembami	Neusklajene obstoječe programske rešitve, ki niso usklajene z globalnimi poslovnimi pravili na najvišjem nivoju.	Izboljšati vrzel globalnih poslovnih pravil in tistih, ki so zapisani v programski kodi.
Redeployment	Prehod na spletne programske rešitve	Ponujanje storitev na svetovnem spletu.	Zajem poslovne logike, ki je implementirana v obstoječih programskih rešitvah.
Recapture	Obratno inženirstvo in ponovna dokumentacija	Projekti se običajno izdelujejo zaradi strahu pred izgubo poslovne logike.	Strukturiran pristop zajem in hranjanje poslovnega znanja in poslovnih pravil.
Reempowerment	Izboljšati moč CRM-ja	Izboljšati sodelovanje z večjimi, pomembnimi strankami.	Zapisovanje in upravljanje poslovnih pravil je hitro in učinkovito. S pravili upravljajo poslovni uporabniki.

Vir: R. Ross, Principal of business rule approach, 2003, str. 11.

Vemo, da je programska oprema že prisotna v skoraj vseh poslovnih sistemih in je v nekaterih tudi nujno potrebna za uspešno delovanje poslovnega sistema. Razvoj novih programskih rešitev bo v prihodnosti odvisen od kvalitete obstoječih rešitev in motivacije za nove programske rešitve. V Tabeli 3 so zbrani in kategorizirani projekti, ki bodo aktualni v prihodnosti (Ross, 2003). V zadnjem stolpcu Tabele 3 je zapisano, zakaj je koristno uporabiti pristop poslovnih pravil. Omeniti moram, da sem se v magistrski nalogi

v praktičnem delu odločil za izdelavo projekta obratnega inženirstva in ponovne dokumentacije projekta, pri katerem sem sodeloval v podjetju, kjer sem zaposlen.

2.1.6 Tehnologija in pristop poslovnih pravil

Zanimivo je, da pri uveljavitvi poslovnega pristopa informacijska tehnologija nima posebnega vpliva, z razliko od objekto orientiranega pristopa, ki se je razvil iz že uveljavljenega objektno orientiranega programiranja (angl. *object oriented programming*). Poslovne sisteme morajo voditi poslovne potrebe in ne tehnologija, ki je na voljo na trgu (Ross, 2003). Avtor trdi, da smo na meji novega vala tehnologij, ki zajemajo in zbirajo bazo znanja o poslovanju. Poslovna pravila, ki zajemajo poslovno logiko poslovanja, so en korak do uresničitve tega cilja.

Tehnologija, ki bi omogočala arhitekturo programske rešitve za izvajanje poslovnih pravil, se v preteklosti ni uveljavila. Bili so redki poizkusi v 80. letih prejšnjega stoletja na področju ekstremnega programiranja in umetne inteligence. Takrat je bila računalniška arhitektura monolitna oziroma strogo proceduralna. Danes lahko poslovno logiko vkomponiramo v storitve ali knjižnice, kar omogoča bolj fleksibilno arhitekturo informacijskih sistemov, zato je onemogočena osnovna ovira za razcvet vkomponiranih strojev poslovnih pravil in orodij za podporo odločanju.

2.2 Poslovno modeliranje pri pristopu poslovnih pravil

Zanimiva je analogija poslovnega pristopa s človeškim telesom, ki jo je omenjal Ross (2003). Avtor je primerjal mehanično delovanje človeškega telesa s poslovnim sistemom in informatizacijo. S pomočjo analogije s človeškim telesom si lažje razlagamo delovanje poslovnega sistema in področja poslovnega modeliranja.

Podobno kot človeško telo deluje tudi poslovni sistem. **Strukturo** poslovnega sistema sestavljajo **osnovni koncepti** in logične povezave med njimi. **Moč** ali delovanje poslovnega sistema predstavljajo **poslovni procesi**, ki delujejo na omenjenih osnovnih konceptih. **Kontrola** poslovnega sistema so **poslovna pravila**, ki usmerjajo poslovni proces z odločitvami.

Struktura predstavlja osnovno znanje o poslovanju in poslovnem sistemu. Osnovno znanje je sestavljeno iz osnovnih konceptov in njihovih logičnih povezav, Osnovni koncept mora biti jasen in ustrezno definiran za poslovno rabo. Osnovne koncepte moramo poimenovati v **izraze** (angl. *term*) ali pojme. **Dejstva** (angl. *fact*) so standardni stavki, ki so izraženi na podlagi izrazov. Model izrazov in dejstev predstavlja ogrodje poslovnega sistema, okoli katerega se kasneje organizirajo druge komponente in nosi težo organizacije, ki se kasneje implementira v podatkovno bazo. Običajno se sestavi v grafično obliko, kjer je viden celoten sistem izrazov in njihovih povezav (dejstev).

Poslovni proces je komponenta, ki je v poslovnem sistemu največkrat vidna ali pa se najbolj pogosto omenja, ker definira tisto, kar mora poslovni sistem delati (npr. proces naročila za stranko). Moč zagotavljajo poslovni procesi, ki delujejo na znanih izrazih in dejstvih. Medtem ko na eni strani model dejstev omogoča strukturo, poslovni procesi omogočajo **izvajanje aktivnosti ali delovanje**. Poslovni sistem ni samo seznam poslovnih procesov

Poslovna pravila zagotavljajo kontrolo delovanja poslovnega sistema, ki vključujejo poslovne procese, ki delujejo na določen način, da koristijo poslovnemu sistemu (poslovanju). Poslovni procesi morajo delovati skladno in organizirano, v nasprotnem primeru poslovni sistem ne deluje optimalno. V nekaterih primerih pride do posledic izbube strank, kar pomeni zmanjšanje kapacitete delovanja poslovnega sistema.

Dokler poslovanje sistema teče nemoteno, se ne oziramo na kontrolo ali poslovna pravila in se lahko osredotočamo na druge prioritete. **Neodvisnost poslovnih pravil in poslovnih procesov je ena izmed ključnih načel poslovnega pristopa.**

2.2.1 Model izrazov in dejstev

Model dejstev je v očeh informatikov zelo podoben konceptualnemu podatkovnemu modelu ali razrednem diagramu na najvišjem nivoju. Vendar obstajajo nekatere razlike, ki jih je potrebno izpostaviti. Večina informatikov uporabi konceptualni podatkovni model za načrt izdelave strukture podatkovne baze ali pa za načrt izdelave podatkovnega skladišča. Kasneje po izdelavi in implemetaciji se poslovni uporabniki redko sklicujejo na podatkovni model, čeprav se nekateri zavedajo pomembnosti terminologije in poslovnega slovarja. Na drugi strani pa mora biti model dejstev osnovno držalo (sredstvo) za komuniciranje. Model dejstev mora predstavljati poslovni slovar celovito za vsa poslovna pravila. Za urejanje poslovnega slovarja je potrebno imeti upravljalca poslovnega slovarja, ki opravlja vlogo koordinatorja s poslovnim znanjem iz poslovne strani. V današnjem času le-ti redko obstajajo, če pa so, so to informatiki, ki poznajo podatkovni model. Za pristop poslovnih pravil je pomembno, da morajo model dejstev sestaviti poslovni uporabniki, ki posedujejo znanje o poslovnem sistemu. Čeprav služi model dejstev za načrt podatkovne baze, to ni njegov osnovni namen. Dejstvo je, da so se poslovni uporabniki v preteklosti izogibali in bali podatkovnega modela, zato je model dejstev zastavljen tako, da ni toliko tehničen. V primeru, da imamo ustrezno poslovno znanje, ne sme biti težko razumljiv, saj je v nasprotnem primeru prišlo do nerazumevanja poslovanja.

Izraz je beseda ali fraza, ki je izražena v naravnem jeziku (npr. angleščina). Za poslovni svet ima izraz pomen, ki ne sme biti podan nedvoumno v določenem kontekstu. Običajno so to samostalniki ali pa samostalnik s pridevnikom, kot naprimer: stranka, status, račun, pomembna stranka, plačilo. Izrazi ne smejo biti opisani tehnično, ampak mora biti opisan njihov poslovni pomen.

Pri poslovnem pristopu množici izrazov pravimo konceptualni katalog ali poslovni slovar. Pri izdelavi podatkovnega modela se običajno izdelujejo etitepni tipi ali razredi, ki vključujejo izraze. Pri poslovnem pristopu obstaja možnost sklicevanja na posamezno instanco razreda (objekt) ali entitete in ne na celotni entitetni tip, npr. entitetni tip spol imai ima običajno dve instanci, moški in ženski in vsi predstavljajo izraze pri poslovnem pristopu. Pristop poslovnih pravil omogoča model dejstev razširiti z modelom instanc izrazov. Standardizacija teh izrazov je pomemba pri prenašanju znanja in avtomatizaciji znanja.

Dejstva so enostavni deklerativni stavki, ki povezujejo ustrezne izraze. Dejstva predstavljajo vse, kar poslovanje ve o izrazih. Izražena so z glagoli.

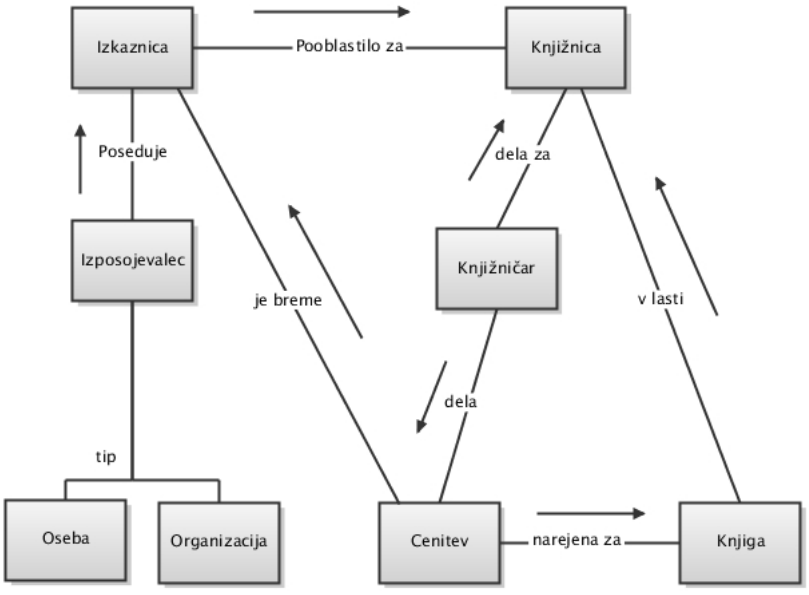
Primer 3: Nekaj primerov dejstev

Naročnik izda naročilo. Menedžer je kategorija zaposlenih.

Nekaj osnovnih ugotovitev glede modela dejstev:

- model dejstev razširja osnovni poslovni slovar;
- struktura dejstev je logična povezava dejstev: osebek-povedek-predmet;
- model dejstev predstavlja skupno poslovno znanje;
- dejstva predstavljajo, kaj je možno vedeti in ne omejujejo poslovnega sistema;
- model dejstev povezuje izraze med seboj, ampak ne določa zaporedja izvajanja.

Slika 8: Primer grafične predstavitve modela dejstev



Vir: R. Ross, Principal of business rule approach, 2003, str. 63.

Grafična predstavitev modela dejstev je bolj pregledna in je dobrodošla pri velikih kompleksnih poslovnih sistemih, primer je prikazan na Sliki 8.

Zanimivo je, da ko končni uporabnik postavlja **zahteve** za poslovni sistem, običajno le-te nanaša na izgled zaslonske maske ali katero funkcijo naj se izdela, ne glede na to, pa so izrazi in dejstva tudi zahteve, ki so pomembne. Seveda niso to edine zahteve, je pa to načrt, kjer je zbrano osnovno znanje poslovnega sistema, ki pove, katere podatke lahko imamo in kako jih lahko povezujemo. Kot je že bilo večkrat omenjeno, so osnovni koncepti oziroma zahteve podlaga za druge vrste zahtev, kot so poslovna pravila.

2.2.2 Poslovna pravila v vlogi kontrole

Pravila so vsem znana pri običajnem življenju, saj jih moramo poznati pri igranju različnih iger, živimo v pravnem sistemu, kjer moramo spoštovati določena zakonska pravila, otrokom postavljamo določena pravila. Vendar poslovni analitiki in informatiki v poslovnem svetu redko najdemo poslovna pravila eksplicitno izražena in so tudi redko sprejeta kot konstrukt ali element. Velikokrat poslovna pravila za informatike pomenijo elemente za ekspertne sisteme in umetno iteligenco.

Nekateri poslovni uporabniki in analitiki so se naučili proceduralnega razmišljanja izdelave zahtev, zato je lahko razmišljanje na podlagi poslovnih pravil zanje preveč abstraktno ali tuje, za kar so krive skoraj vse metodologije. Po drugi strani pa je razmišljanje o pravilih v organizaciji aktivnosti zelo naravno, saj si je težko razližiti katerokoli igro brez pravil.

V nadaljevanju je naštetih nekaj poslovnih pravil glede na tip kontrole, ki jo naslavljajo.

Primer 4: Poslovna pravila glede na tip kontrole

Omejitev: Naročnik ne sme plačati več kot tri hitra plačila na njegov kreditni račun.
Hevristično pravilo: Naročnik, ki ima VIP status, mora dobiti naročilo takoj.
Izračunljivo pravilo: Naročnikov letni prihodek je izračunan kot vsota prodanih izdelkov v času letnega poročila podjetja.
Inferenca: Naročnik je velika stranka v primeru, da plača več kot pet zneskov večjih od 1000 evrov.
Pravilo s časovno opredelitvijo: Naročnik se arhivira v primeru, da ne plača nobenega zneska v roku 36. mesecev.
Prožilec: Pošlji sporočilo, ko je pošiljka oddana na pošto.

Eno izmed ključnih načel poslovnega pristopa je, da **morajo biti poslovna pravila izdelana na modelu dejstev in konceptualnem katalogu**.

Osvoboditev pravil od poslovnih procesov nadeja nove priložnosti, ki so:

- konsistenca: eno poslovno pravilo ima lahko več ažurirnih podatkov. Poslovno pravilo je implementirano na enem mestu in je konsistentno v vseh procedurah, kjer se uporablja.
- prilagodljivost: poslovno pravilo je lahko definirano na enem mestu in se uporablja v različnih procedurah, kar pomeni, da ga je lažje najti in tudi spreminjati.
- prenova: poslovni procesi in procedure so narejeni kot odgovor na poslovni dogodek. Deklerativna pravila so specificirana brez dogodkov, služijo samo kot osnova za kontrolo delovanja.

2.2.2.1 Poslovna pravila in dogodki

Poslovni sistemi so izvajali validacijo na spremembah podatkov že od vsega začetka izdelovanja programskih rešitev. Na žalost je bila validacija proceduralna, ker tako delajo tradicionalni programi. Pristop poslovnih pravil pravi, da morajo biti poslovna pravila izražena deklarativno in neodvisno od procesov ali procedur. To nas na nek način sili v to, da se ukvarjamo s poslovnim problemom (Ross, 2003).

Obrazložitev deklarativnosti je pomemben del poznavanja poslovnih pravil, zato bom na kratko poskušal na primeru orisati definicijo. Množica pravil je zapisana deklarativno v primeru, da zaporedje, ki ga podamo stroju za izvajanje poslovnih pravil, poda vedno isti rezultat in je rezultat neodvisen od zaporedja izvajanja poslovnih pravil (Ross, 2003).

Primer 5: Poslovno pravilo za opis deklerativnosti

15 % popust moramo dodeliti **stranki** v primeru, da velja dvoje: **Naročilo** je za **dobro stranko** in **Znesek naročila** je večji od 1000 evrov.

Verjetno obstajajo še druga pravila, ki določajo, kaj je dobra stranka in pravilo za določanje skupnega zneska naročila. Jasno je, da se morajo ta pravila ovrednotiti pred zgoraj opisanim pravilom.

Ravno zato, ker so pravila zapisana deklerativno, lahko stroj za izvajanje poslovnih pravil odloči pravilni vrstni red oziroma zaporedje. Uporabnik lahko stroju za izvajanje poslovnih pravil poslovna pravila poda v poljubnem zaporedju, kjer bo dobil vedno enak rezultat.

Za razumevanje delovanja vrednotenja poslovnih pravil je potrebno podrobno preučiti relacijo med dogodkom in pravilom. Intiutivno je znano, da se poslovna pravila prožijo, kadar se dogodek izvede (če ni dogodkov, ni potrebno izvajati nobenih pravil). Na dogodke lahko gledamo iz dveh perspektiv, in sicer:

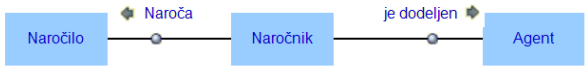
1. Poslovna perspektiva: za poslovanje je dogodek nekaj, kar povzroča ustrezen poslovni odziv tudi, če je trivialen. Primer: Naročnik izda naročilo. To je dogodek, ki zahteva organiziran odgovor. Običajno te odgovore vnaprej pripravimo v delovne procese, skripte, procedure ali postopke ravnanja.

2. Informacijska perspektiva: za poslovni sistem je dogodek nekaj, kar je potrebno zapisati ali shraniti. Ker je podatek, je lahko pomemben pri nadaljnih aktivnostih ali pa v tem trenutku. Takšen zapis je osnovan na modelu dejstev. Glede na to, da je model dejstev lahko razredni diagram ali podatkovna baza, ga lahko v implementacijskem smislu poenostavimo. Podatke se v sistemu ažurira (angl. *create, read, update, delete, okr.* CRUD) in se izvede dogodek. Tem dogodkom bom v nadaljevanju opisal kot ažuriranje podatka.

Kako se dogodki povezujejo s poslovnimi pravili, je najbolje opisati na primeru (Primer 6).

Primer 6: Poslovno pravilo izdaje naročila in njegova grafična predstavitev

Agent mora biti dodeljen naročniku v primeru, da naroča naročilo.



Poslovno pravilo iz Primera 6 se sproži, kadar naročnik naroča naročilo. Obstajajo pa tudi drugi dogodki, pri katerih se proži pravilo, kot je npr. ko agent zapušča podjetje in je zbrisan iz registra agentov. Oba dogodka sta za podjetje lahko pomembna, čeprav je prvi dogodek bolj intiutiven.

Za poslovno pravilo opisano v Primeru 6 velja:

- poslovno pravilo je deklerativno, saj ne sodeluje v procesu ali proceduri;
- prav tako ni vezan na noben dogodek ali ažuriranje podatka;
- pravilo ni proceduralno v smislu: če naročnik naroča, potem je potrebno...

Vsako pravilo lahko tipično sproži enega ali več dogodkov. Primer poslovnega pravila je: Naročnik mora imeti naslov. Pravilo se proži, kadar se izdela nova instanca naročnika ali pa kadar se briše nročnikov naslov.

Odgovoriti moramo tudi na vprašanje, kaj se zgodi v primeru, ko se pravilo prekrši? Dogodek je sprožil pravilo, ki se ovrednoti in pravilo pri kršitvi izvede 2 stvari: i) ne glede na dogodek se pravilo mora sprožiti, da se omejitev ali test lahko aplicira; ii) pri predpostavki, da pravilo preprečuje napako, se vrne spročilo v obliki poslovnega pravila.

Ključni izdelek poslovnega pristopa je **knjiga pravil** (angl. *rule book*). Knjiga pravil lahko igra aktivno vlogi pri razvoju programske rešitve ali sistema. Tudi po vpeljavi programske rešitve je lahko koristen pripomoček. Potrebno je poudariti, da knjiga pravil ne dodaja dodatne komplekstnosti poslovnega sistema, poslovni sistemi so že sami po sebi lahko kompleksni. Knjiga pravil se lahko primerja z drugimi izdelki, kot so model delovnih

procesov, primeri uporabe in druge tehnike pri izdelavi specifikacije in uporabniških zahtev.

2.2.3 Poslovni proces oziroma potek dela

Spomnimo, da v splošnem proces vzame vhodni podatek, nato sledi nekemu algoritmu in na koncu producira izhodni podatek kot rezultat. Poslovni procesi so postali preveč komplesni za spreminjanje. Pristop poslovnih pravil oblublja, da bodo poslovna pravila revolucionarno spremenila potek dela ali delovni proces.

Iz vidika poslovnih procesov so današnji izzivi povezani s spreminjanjem poslovnih procesov in prilagajanju zaposlenih novim vlogam in zadolžitvam. Zaposleni morajo pri teh spremembah iti skozi proces uvajanja čim bolj efektivno in z minimalnim posredovanja drugih zaposlenih. Šolanje je drago in časovno zahtevno, zato je rešitev računalniško vodeno šolanje (Ross, 2003).

Pri poslovnem pristopu je potrebno vedeti, da poslovnim procesom oziroma postopkom pravimo **skripte**. Skripte opisujejo potek dela na operativnem nivoju. Priporočajo, da je skripta **predpisan seznam zahtev**, ki ga lahko imenujemo tudi serija korakov za doseganje željenega cilja, ki ne vključuje poslovnih pravil. Predpisani seznam pomeni, da lahko sledimo zahtevam za doseganje rezultata, ni pa nujno. Lahko pa obstaja druga skripta, ki dosega enak rezultat. Zahteva je v tem primeru zahteva za akcijo, ki od programa nekaj zahteva. Ta progam je lahko: DBMS (anlg. *database management system*), zaslonska maska, spletna storitev, integracija ali stroj za izvajanje poslovnih pravil.

Primer 7: Skripta poteka dela ali postopka

Najdi referenčni odčitek. Poišči zgodovinsko analizo. Izračunaj pričakovano porabo.

Upoštevanje tudi drugih načel je ključnega pomena. Skripte se morajo sklicevati na že znana dejstva in izraze, ki so v skupni rabi. Skripte lahko ponovno uporabimo v drugih primerih. Pri poteku dela lahko sodelujejo ljudje oziroma akterji. Torej lahko sklepamo, da poslovna pravila omogočajo prilagajanje in sodelovanje, ki vključuje stroje in ljudi.

2.2.4 Predloge poslovnih pravil v naravnem jeziku

Pri izdelavi knjige pravil je priporočeno uporabljati predloge različnih vrst poslovnih pravil, ki so zbrane v Tabeli 4.

Predloge poslovnih pravil so razdeljene po različnih kategorijah poslovnih pravil. V ospredju poslovnega pravila je subjekt, ki je običajno izraz ali dejstvo na kateremu se poslovno pravilo ovrednoti. Veliko pravil ima dodano opcijsko komponento, ki določa pod

katerimi pogoji se lahko pravilo vrednoti. Za lažje razumevanje so zapisani tudi primeri poslovnih pravil.

Tabela 4: Predloge poslovnih pravil (angl. BRS rule speak)

Kategorija	**Namen in neformalni opis**	**Subjekt mora biti**	**Predloga**	**Primer**
Omejitev	Omejitev za vzdrževanje pravilnosti stanja (konsistence) podatkov	Izraz ali dejstvo	<subjekt>mora imeti<dejstvo>[pogoj]	Naročilo mora imet datum prejema.
			<subjekt>samo<dejstvo>[pogoj]	Nižji uradnik lahko dela samo v pisarni v prvem nadstropju
Izjava dovoljenja	Politika, ki omogoča poslovno prakso		lahko ali ni nujno	Stranka ni nujno, da kupi izdelek.
				Kreditno plačilo lahko sprejemo tudi, če je znesek računa manjši od 100 evrov.
Izračunljivo pravilo	Izračunljiva matematična formula	Izračunljiv izraz	<subjekt>se izračuna kot <matematična formula>[pogoj]	Znesek računa se izračuna kot seštevek vseh bruto postavk računa.
Izpeljava	Izjava, ki določa logično izpeljajo (da/ne)	Izpeljan izraz	<subjekt>pomeni/je<logična izpeljava>[pogoj]	Dobra stranka je stranka, ima na računu več kot 1000 evrov.
				Analiziran odčitek pomeni odčitek, ki ga analiziramo.
Pravilo sklepanja ali inferenca	Pravilo, ki sklepa na podlagi možice dejstev.	Izraz	<subjekt>je<logični stavek>[pogoj]	Oseba je deklina v primeru, da je ženskega spola in je njena starost manjša od 18 let.
Omogočanje pravila	Pravilo, ki lahko vključi drugo pravilo pod določenimi pogoji. Primerno za vračanje napak.	Pravilo	<naziv-pravila>ne velja/se ne sme prožiti<pod pogojem>	Pravilo-definicija-delkine ne velja za ameriške državljane.
				Pravilo-določanje-dobre-stranke se ne sme sprožiti ob petkih.
Omogočanje procesa	Pravilo, ki omogoči proces pod določenimi pogoji.	Proces, skripta ali procedura	<subjekt>mora biti omogočen/neomogočen<pod pogojem>	Proces-pakiranja-izdelka mora biti onemogočen v primeru, da nimamo vseh sestavhin delov izdelka.
Omogočanje podatka	Pravilo, ki briše podatek pod določenimi pogoji	Podatek	<subjekt>moramo brisati<pod pogojem>	Stranko moramo brisati pod pogojem, da eno leto ni aktivna.
Implicitno pravilo	Pravilo, ki nastavi podatek na novo vrednost	Izraz ali dejstvo	<izraz>mora biti določen<izraz/vrednost>	Letna naročnina mora biti določena na 5 evrov, pod pogoejm, da je stranka aktivna vsaj eno leto.
Predstavitveno pravilo	Pravilo, ki predsatavi podatke v določeni obliki - lahko tudi na ekranu	Izraz ali dejstvo	<subjekt>se mora prikazati <na mediju><pod pogojem>	Pogodba se mora prikazati na ekranu, v primeru, da je pretekla veljavnost.
Prožilec procesa	Pravilo, ki proži proces pod določenimi pogoji	Proces, skripta ali procedura	<subjekt>se mora prožiti<pod pogojem>	Proces-izedlave-računa se mora prožiti, ko dobimo odčitek odjemnega mesta.
Prožilec pravila	Pravilo, ki proži drugo pravilo pod določenimi pogoji	Pravilo	<naziv-pravila>se mora prožiti<pod pogojem>	Izračun-tveganja se mora prožiti, kadar stranka preseže limit.

Vir: R. Ross, Bussines rule approach, 2003, str. 158.

2.2.5 Informatizacija pri poslovnem pristopu

Informatiki se lahko vprašajo, koliko poslovnih pravil ima informacijski sistem implementiranih, kako težko je poslovno pravilo spremeniti in katera pravila iz specifikacije so implementirana? Pogosto so to kompleksna vprašanja. Zato se stvari poenostavijo, kadar imamo poslovna pravila shranjena v podatkovni bazi. V teh primerih lahko s poslovnimi pravili upravljamo in jih lažje spreminjamo. Repozitorij poslovnih pravil je koristen v primerih, ko imamo tudi shranjene druge metapodatke o poslovnih pravilih, kot naprimer: kdo in kdaj je izdelal poslovno pravilo, zakaj smo poslovno pravilo izdelali in kakšen je status, ali je v produkciji, ali je poslovno pravilo zastarelo.

Povezanost modela dejstev z ostalimi izdelki poslovnega pristopa lahko opišemo preko:

- konceptualnega kataloga, ki je seznam vseh izrazov in definicij, ki zajemajo znanje podjetja. Vsak izraz, ki se uporablja v modelu dejstev, bi moral imeti definicijo v konceptualnem katalogu.
- taktičnega načrta, ki je načrt, ki vsebuje ključne cilje in poslovna pravila ter nakazuje, kako bomo te cilje dosegli in zmanjšali tveganje. Vsak element je implicitno zgrajen na osnovnem znanju, ki mora biti strukturirano in standardizirano. Taktični načrt je izvor izrazov ali konceptov, ki morajo biti v modelu dejstev.
- modela delovnih tokov (procesov), ki opisuje potek dela v poslovnem procesu. Izvajanje poteka dela producira neko znanje, ki mora bazirati na izrazih in dejstvih. Modeli delovnih tokov so tudi izvor izrazov in osnovnih konceptov, ki morajo biti vključeni v model dejstev.
- knjige pravil, ki vsebuje poslovna pravila in predstavlja centralni izdelek. Vsako pravilo mora biti odvisno od izraza ali dejstva definiranega v modelu dejstev.

Tabela 5: Dejavniki abstrakcije poslovnega modela

Poslovni aspekt	**Poslovna komponenta**	**Izdelek poslovnega modela**	**Aplikacijska komponenta**
Znanje	Izrazi in dejstva	Model dejstev	Podatki
Transformacija	Poslovni procesi	BPM	Procesi
Povezovanje	Poslovne povezave	Omrežni načrt	Povezave med stroji
Sodelovanje	Vloge in delovni produkti	Organizacijski delovni procesi	Zaslonske maske ali grafični uporabniški vmesniki
Uprizoritev	Mejniki	Poslovni mejniki	Stanja ali statusi
Vodenje	Poslovni cilji in taktika	Taktični načrt	Pravila

Vir: R.Ross, Principal of business rule approach, 2003, str. 193.

Cilj razvoja je zagotoviti prilagodljivo in fleksibilno poslovanje. Zato mora vsebovati poslovni model, najboljšo možno poslovno rešitev in sponzorja projekta, ki nadzira projekt z njegovim minimalnim vložkom časa. Aplikacijske komponetne morajo biti integrirane v

poslovanje. Intergracija mora imeti načrt in od vrha navzdol (angl. *top-down*) poslovni model, ki mora pokrivati celotno sfero poslovanja, kar je prikazano v Tabeli 5.

Prednost poslovnega pristopa od drugih pristopov in metodologij je v tem, ker teži k temu, da se v prvi vrsti reši poslovni problem in da se naredijo izdelki od vrha navzdol.

3 DRUGE METODOLOGIJE IN PRISTOPI RAZVOJA PROGRAMSKIH REŠITEV

V tretjem poglavju bom raziskal vzporednice poslovnega pristopa in povezave z drugimi motodologijami razvoja programske rešitve. V ospredju sta v preteklosti najbolj poznan **strukturni pristop** in v današnjem času najbolj razširjen in poznan **objektno orientiran pristop** (angl. *object oriented approach*, okr. OOA). Raziskal bom tudi, kako se pristop poslovnih pravil povezuje s **storitveno usmerjeno arhitekturo** (angl. *service oriented architecure*, okr. SOA). Pri raziskovanju me je najbolj zanimalo, kako so osnovni koncepti, ki vključujejo 3 koncepte poslovnega modeliranja (struktura, moč in kontrola) in načela povezani z drugimi metodologijami in pristopi. Dotaknil se bom tudi vprašanja, ali pristop poslovnih pravil izpolnjuje pogoje formaliziranih metod.

Predmet povezovanja je v prvi vrsti upravljanje z zahtevami in poslovno modeliranje. Ne moremo pa mimo dejstva, da je pri končni rešitvi potrebno programsko rešitev implementirati s pomočjo določene tehnologije. Zato se bom na kratko dotaknil tudi informatizacije razvoja programske rešitve.

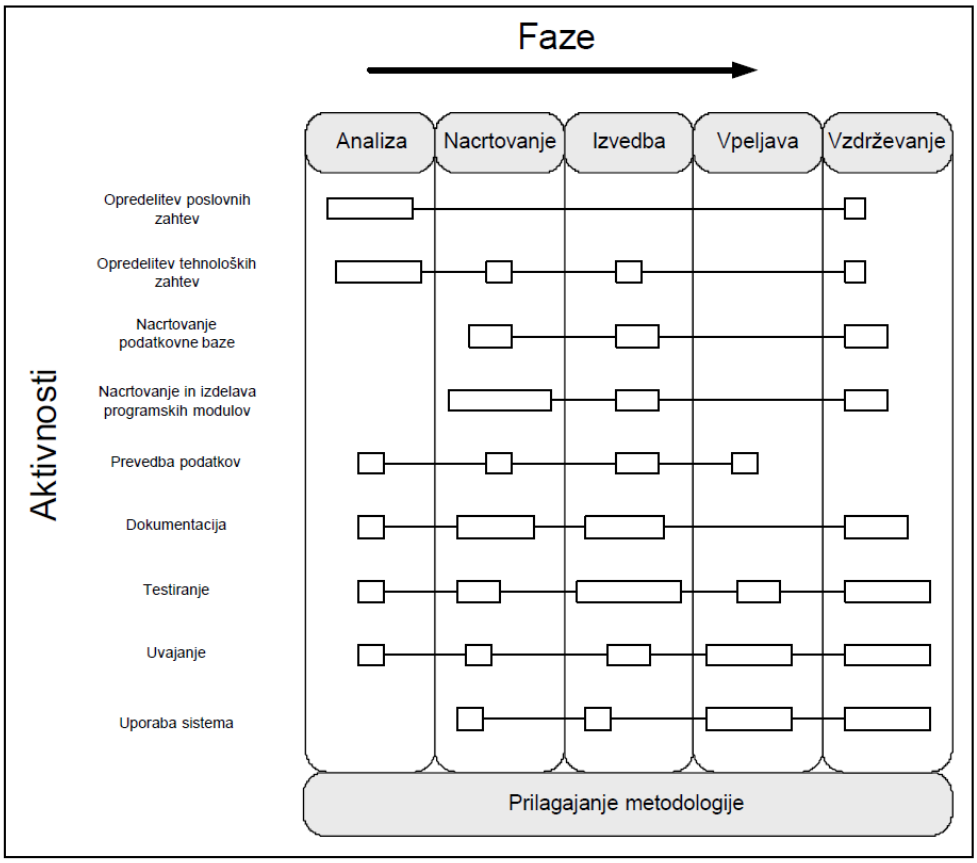
Kljub določenim prednostim, ki jih prinaša uvedba in uporaba metodologij, pa se njihova uporaba ne povečuje. Še bolj zaskrbljujoče je dejstvo, da uporaba metodologij v nekaterih organizacijskih sistemih celo upada. Fitzgerald (1998) je v svoji raziskavi ugotovil, da 60 % podjetij, ki se ukvarjajo z razvojem informacijskih sistemov sploh ne uporablja nobene metodologije. Od preostalih 40 % podjetij pa jih le 6 % dosledno sledi navodilom uporabljene metodologije. Do podobnih rezultatov je prišla tudi raziskava, ki je bila opravljena na vzorcu angleških podjetij (Avison & Fitzgerald, 2003) in ugotavljata, da se metodologije niso uporabljale kar v 43 % podjetij. Preostalih 57% podjetij je uporabljalo določen metodološki pristop, od tega jih je le 11 % uporabljalo v naprej opredeljeno neprilagojeno komercialno metodologijo, 30 % podjetij je uporabljalo komercialno metodologijo, ki so jo prilagodili svojim potrebam, 59 % od omenjenih podjetij pa je uporabljalo unikatne lastne metodologije, razvite v lastni hiši.

3.1 Strukturni pristop

Strukturni pristop izgradnje informacijskih sistemov ima predvidene aktivnosti faze in izdelke kot je prikazano na Slika 9. Uporablja tehnike izdelave relacijskih modelov pri podatkovnem modeliranju in diagrame podatkovnih tokov za opis procesnih modelov. Tehnike modeliranja procesne logike nakazujejo, da se poslovna logika implementira v

poslovnem procesu, kjer se uporablja strukturiran jezik. Prav tako predvideva odločitvene tabele, ki so na nek način skupine poslovnih pravil.

Slika 9: Aktivnosti in faze strukturnega pristopa



Vir: M. Krisper, EMRIS Strukturni pristop, 2000, str. 4.

Pri poslovnih pravilih se obravnava samo naslednje kategorije poslovnih pravil: definicije, omejitve in izpeljave. Pri zapisovanju poslovnih pravil uporablja diagram dekompozicije in ni repozitorija poslovnih pravil. Kasneje je izboljšana verzija sturkturnega pristopa predlagala odločitvena pravila, ki bi sugestirala pri procesu izdelave programske rešitve (Krisper, 2000). Danes sturkturni pristop v fazi analize zajema ali predvideva izdelavo modela poslovnih pravil. Večinoma so mišljena omejitvena poslovna pravila. Izrazi in dejstva so zabeležena v globalnem podatkovnem modelu.

Strukturni pristop je še vedno koristen za določene projekte. Po drugi strani pa se je nekoliko bolje uveljavil objektno orientiran pristop.

3.2 Objektno orientiran pristop in UML

Objektno orientiran pristop se ukvarja z modeliranjem objektov in njihovim povezovanjem. Vsak objekt predstavlja neko entiteto, ki ima določene atribute, stanje, definira neko obnašanje v sistemu in pripada nekemu razredu. Obstaja veliko različnih

diagramskih tehnik oziroma grafičnih predstavitev za uporabo objektno orientiranega pristopa, med njimi se je najbolje uveljavil UML (angl. *unified modified language*).

Zgodovinsko gledano so se objektno orientirane metodologije začele pojavljati vzporedno z objektno orientiranim programiranjem, izraz programiranje pa predstavlja generacijo programske kode. Razvoj se je nadaljeval z objektno orientiranim načrtovanjem, nato se je začela pojavljati objektno orientirana analiza. Izraz objektno orientirana metodologija združuje vse 3 dejavnosti in vključuje celotno filozofijo razvoja sistema od analize zahtev, načrtovanja sistema, programiranja, do testiranja delovanja sistema. Glavne prednosti objektnega pristopa sta ponovna uporabnost in nadgradnja programske kode (Fetih, 2004).

Poslovna pravila so v UML jeziku vgrajena v grafični sintaksi ali semantiki, ki se imenuje omejitveno pravilo v OCL (angl. *object coinstraint language*). Omejitveno pravilo je stavek zapisan v zavitih oklepajih. OCL je lahko tudi opomba, ki je povezana z modelom. Na opombi se uporablja poseben stereotip z angleškim izrazom *business rule* (Penker & Eriksson, 2000).

OCL, ki ga je razvilo podjetje IBM kot jezik za poslovno modeliranje, je deklerativen formaliziran jezik. OCL ni programski jezik in je neodvisen od tehnologije. Ne podpira izvajanje akcije v primerih, da je ovrednoteno pravilo kršeno pri določenih pogojih.

Poslovna pravila, ki ne morejo biti izražena na tradicionalen način z enostavnimi pravili, imenujemo **mehka logika** (angl. *fuzzy logic*). Mehka logika se predstavi s funkcijo, ki vključuje različna poslovna pravila ali stavke in se izogiba striktni definiciji, ki jih zahteva OCL ali drugi programski jeziki. Mehka logika se lahko sestavlja v kompleksne programske komponente.

3.2.1 Povzetek uporabnosti jezika UML

Od vseh diagramov UML-ja so se za namene modelno vodenega razvoja izkazali najbolj uporabni razredni diagrami ter diagrami prehajanja stanj, oboji dopolnjeni z jezikom OCL za podajanje omejitev ali določeno konkretno akcijsko semantiko. Uporabni so tudi komponentni in namestitveni diagrami, vendar ti dve vrsti nimata takega pomena kot razredni diagrami in diagrami prehajanja stanj, s katerimi podrobno opišemo sistem. Od teh dveh tipov diagramov k dvigu nivoja abstrakcije doprinesejo le diagrami prehajanja stanj. Drugi diagrami so za direktno uporabo v modelno vodenem pristopu manj primerni. Še vedno pa jih lahko uporabimo v njihove originalne namene, torej za opisovanje sistema. Uporabnost UML diagramov precej poveča uporaba jezika OCL za določanje omejitev, poizvedb in v omejeni obliki tudi splošnejših postopkov z dopolnjevanjem omejenih izraznih možnosti UML diagramov (Vidrih, 2008).

Vidrih (2008) je nakazal, da UML diagrami za modeliranje določenih vidikov sistema ne delujejo najbolj primerno in hkrati je ugotovili, da tudi različica UML-ja 2.x, razen akcijske semantike brez konkretnega zapisa in manjših izboljšav, nima bistvenega doprinosa k modelno vodenemu pristopu.

3.2.2 Informatizacija poslovnih pravil v UML

OMG (angl. *object management group*) je izdelal specifikacijo SBVR (angl. *semantic semantics of business vocabulary and business rules*) z namenom za standardizacijo poslovnih pravil, ki bi se lahko implementirala v različnih programskih orodjih. SBVR je izdelan na predikatni in modelni logiki in je neodvisen od informacijske rešitve. Ideja poslovnega pristopa je, da se pretvori semantika definirana s SBRV v razvoj programske rešitve, da se informatizira. Problem je, da imajo poslovna pravila, ki se izvajajo v stroju za izvajanje poslovnih pravil, obliko in predstavitev, ki je odvisna od tehnološke platforme. OMG dela na izdelavi specifikacije oziroma standarda prezentacije produkcijskih pravil (angl. *production rule representation*, okr. PRR). RuleML dela na specifikaciji, ki bi transformirala in izmenjala poslovna pravila med različnimi programskimi paketi in semantičnim opisom.

S pomočjo razširitve UML za poslovna pravila lahko kategorizacija in strukturiranje UML omejitev omogoča pretvorbo naravnega jezika v OCL in druge jezike poslovnih pravil (Nemuraite, Ceponiene, & Vedrickas, 2009).

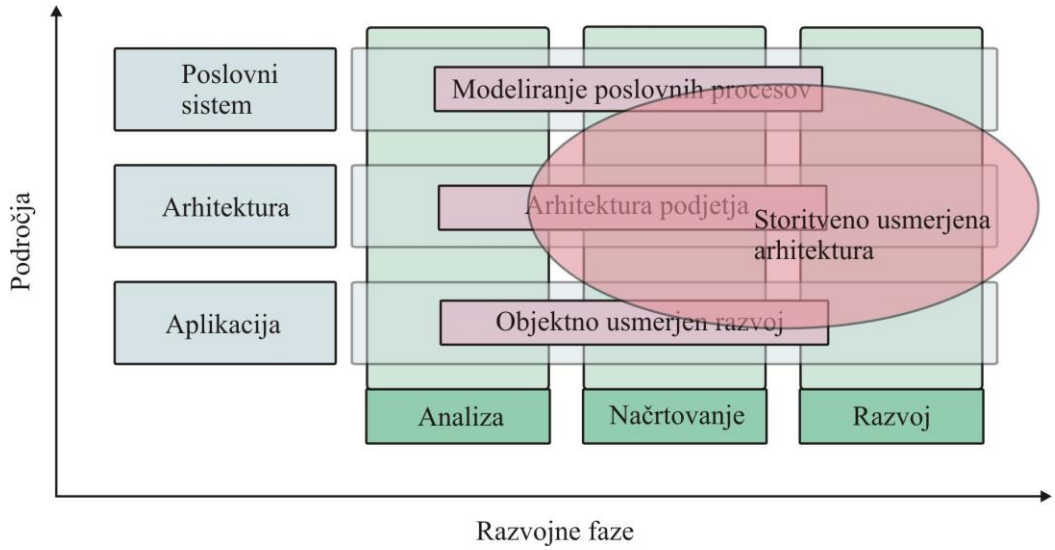
Omejitvena poslovna pravila je možno opisati z uporabo OCL jezika. Implementacija se lahko izvede v objektnih podatkovnih bazah, obstaja pa tudi translacija OCL opisov v SQL (angl. *structured query language*) stavke in prožilce, ki se izvajajo na transakcijskih podatkovnih bazah (Demuth, Hussmann, & Loecher, 2001).

3.3 Storitveno usmerjena arhitektura

Storitveno usmerjena arhitektura (SOA) je pristop, ki stremi k upravljanju kompleksnosti informacijskih sistemov, poslovnih procesov, podatkovnih baz..., z namenom ponovne uporabe, lažje integracije ter predvsem lažjega razvoja posameznih delov informacijskega sistema, brez negativnih vplivov na razvite informacijske rešitve (Medeot, 2007).

V določenih primerih objekto usmerjen razvoj in modeliranje poslovnih procesov ne zadostuje za celovit opis poslovnega sistema in je potrebno vpeljati storitveno usmerjeno arhitekturo (Zimmermann, 2004). V praksi se je pokazalo, da je predvsem v primerih, kjer je potrebno povezovati neodvisne funkcije ali aplikacije med seboj, primeren hibridni pristop, ki ga prikazuje Slika 10.

Slika 10: Pvezava SOA in drugih metodologij



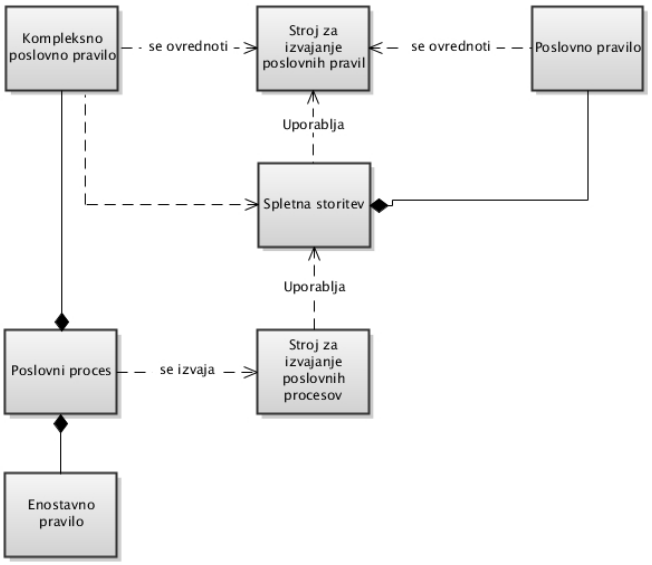
Vir: O. Zimmermann, Elements of service-oriented analysis and design, 2004.

Metodologija objektega pristopa je dobra na začetku razvoja programske rešitve, ko postavljamo osnovno strukturo in konceptualni katalog kot tudi pri analizi in v končni fazi v razvoju, kjer imamo definirane razredne diagrame. Arhitekturni pristop Zachmanovega ogrodja (poglavje 2.1.2) je koristen pri planiranju omrežja lokacij in drugih virov celotnega podjetja (angl. *enterprise*). Tudi modeliranje poslovnih procesov je pomembno, saj prikazuje celotne (angl. *end-to-end*) poslovne procese in delovanje podjetja.

Pri informatizaciji poznamo 2 komplementarna pristopa reševanja problemov: BPM (angl. *business process modeling*) in BRM (angl. *business rule management*). Ideja pristopa SOA je, da poslovno logiko loči od procesne, kjer se z dodajanjem načel SOA pristopa gradi programske rešitve in informacijske sisteme, da postanejo bolj agilni.

Na Sliki 11 je razvidna uporaba stroja za izvajanje poslovnih pravil, ki ga uporablja storitev, ki se izvaja v stroju za izvajanje poslovnega procesa (angl. *service orchestration*). V primerih, kjer imamo kompleksno poslovno logiko, ki se s časom spreminja bolj pogosto, jo lahko zapakiramo v posebno storitev, ki se izvaja v stroju za izvajanje poslovnih pravil. To storitev lahko pokliče stroj za izvajanje poslovnih procesov. Klici storitev na daljavo preko omrežja so lahko tudi potratni in zmogljivostno vprašljivi, zato obstajajo nekateri paketi, ki vključujejo vse omenjene funkcionalnosti v enem paketu (Lublinsky & Le Tien, 2007).

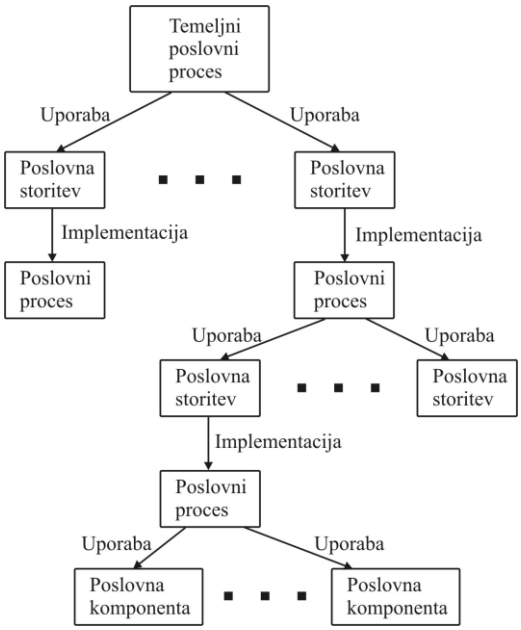
Slika 11: Metamodel poslovnih pravil, procesov in storitev



Vir: B. Lublinsky & D. Tien, Implementation of business rules and business processes in SOA, 2007.

Eden glavnih razlogov za uporabo storitveno usmerjene arhitekture je, da skušamo narediti aplikacije bolj prilagodljive in zmanjšamo vpliv sprememb. Tipično se to naredi tako, da ločimo poslovno logiko ali atribute, ki se najbolj pogosto spreminjajo od tistih, ki so stabilni. To lahko naredimo s tehniko dekompozicije ali razbitja problema na podprobleme (strukturiranje problema). To nas vodi v dekompozicijo storitev, kar lahko vključuje tudi spletne storitve, in poslovnih procesov.

Slika 12: Primer dekompozicije storitev in poslovnih funkcij



Vir: B. Lublinsky & D. Tien, Implementation of business rules and business processes in SOA, 2007.

Glede na dekompozicijo se v SOA implementaciji uporabljajo sledeče povezave med poslovnimi pravili in poslovnimi procesi:

- poslovni procesi implementirajo poslovne storitve;
- poslovna pravila so vkopomonirana v poslovnih komponentah, ki so del poslovne storitve;
- poslovne komponente so odvisne od drugih komponent.

Dekompozicija ne obravnava poslovnih pravil direktno, saj se poslovna pravila tudi pogosto spreminjajo.

Na spletno storitev lahko gledamo tudi kot na knjižnico, ki ima neko funkcionalnost in jo lahko uporabimo za nek namen. V spletni storitvi so zapakirana poslovna logika ali poslovna pravila, ki nas v danem trenutku ne zanimajo, pomembno je, da se reši poslovni problem. Tehnologija klicanja spletnih knjižnic je v današnjem času že na voljo.

4 INFORMATIZACIJA POSLOVNEGA MODELA NA PODLAGI NOVEGA PRISTOPA POSLOVNEGA MODELIRANJA

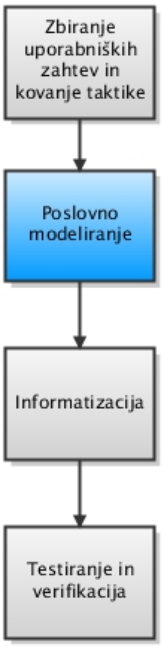
V tem poglavju je prikazan potek dela poslovnega modeliranja in informatizacije poslovnega modela, ki je del praktičnega dela magistrske naloge. Predstavljeni sta tudi 2 ključni orodji, ki sem ju pri tem uporabljal, RuleXpress in ILOG. Glede na to da pristop poslovnih pravil nima podrobno opisane metodologije razvoja programske rešitve, kot sicer druga 2 pristopa, strukturni in objektni, sem pri praktični rešitvi razvoja razvil svoj pristop programske rešitve z upoštevanjem načel poslovnega pristopa.

4.1 Opis razvoja informatizacije programske rešitve na podlagi poslovnega modela

V tem poglavju bom opisal razvoj programske rešitve projekta obratnega inženirstva, ponovne dokumentacije poslovnih pravil in razvoj programske rešitve **analize odčitka na podlagi zgodovine**, ki sem jo razvil sam na podlagi praktičnih iizušenj in novih spoznanj.

Programsko rešitev sem razvijal v štirih fazah (Slika 13). Projektne faze so izpeljane iz osnovnih metodologij z upoštevanjem osnovnih principov in načel poslovnega pristopa.

Slika 13: Faze razvoja programske rešitve



4.1.1 Zbiranje zahtev in določanje taktike

V fazi zbiranja uporabniških zahtev poslovni uporabniki definirajo svoje potrebe in želje, kako naj izgleda rešitev, kaj vse je potrebno upoštevati pri rešitvi, da bodo poslovni uporabniki njihove stranke zadovoljni.

V tej aktivnosti poteka spoznavanje problematike in problemske domene, kjer poteka sodelovanje zaposlenih na različnih sestankih. Pregleda se dokumentacija, obstoječa programska koda in intervjuja se ključne uporabnike. Po potrebi se izdela primere uporabe na najvišjem nivoju in temeljne procese za problemsko domeno ali področje.

Na podlagi zbranih osnovnih informacij se izdela taktični načrt (angl. *policy charter*) programske rešitve. Taktični načrt mora imeti jasne cilje, ne sme biti preveč obsežen, običajno je napisan na eni do petih straneh, kar pa je lahko odvisno od velikosti projekta, ali gradimo informacijski sistem ali zaključeno manjšo programsko rešitev (glej poglavje 2.1.4).

Po izdelavi taktičnega načrta, se le-ta predstavi sponzorju projekta in ključnim udeležencem. V primeru odobritve se lahko prične naslednja aktivnost, ki je podrobno poslovno modeliranje problemskega področja.

4.1.2 Poslovno modeliranje

Poslovno modeliranje je izredno pomembna faza, kateri sem tudi sam posvetil največ pozornosti in ga glede na druge metodologije uvrščamo v fazo analize in načrtovanje. Poslovno modeliranje sem razdelil v podaktivnosti, ki so prikazane na Sliki 14.

Slika 14: Aktivnosti poslovnega modeliranja

V prvi aktivnosti prikazani na Sliki 14 poslovni uporabniki ali poslovni analitiki izdelujejo poslovni slovar, izoblikujejo terminologijo iz obstoječe programske kode, dokumentacije ali pa s pogovori in sodelovanjem. Definirajo in izoblikujejo se osnovni koncepti in izrazi. Struktura programske rešitve oziroma osnovno ogrodje je določeno z modelom dejstev, ki sem ga v našem primeru razvil ali izdelal brez sodelovanja poslovnih uporabnikov na podlagi obstoječega podatkovnega modela.

Delovanje programske rešitve se izdeluje v tretji aktivnost, kjer se določi zaporedje korakov ali delovni tok dogodkov (angl. *workflow*). Izdela se seznam postopkov izvajanja obliki skript brez poslovnih pravil.

Na tem delu je potrebno omeniti relacijo med poslovnim procesom in postopkom izvajanja ali potekom klicev. V osnovni definiciji so si lahko pojmi podobni, jih pa lahko ločimo glede na različne nivoje abstrakcije. Temeljne poslovne procese definiramo pri zbiranju zahtev. Pri podrobnem modeliranju pa modeliramo podprocese ali delovni tok dogodkov oziroma potek klicev programskih funkcij, ki imajo v osnovi isto nalogo, vhodni podatek transformirajo v izhod in lahko spremenijo stanje sistema.

V zadnjem koraku glede na postopek izvajanja izdelujemo poslovna pravila, ki jih razdelimo v skupine in shranjujemo v repozitorij poslovnih pravil.

Pri izdelavi skript in poslovnih pravil je priporočljivo uporabljati prenosti deklerativne definicije poslovnih pravil oziroma že prej omenjeni kognitiven pristop razvoja. Kognitiven pristop ni smiselen na najvišjem nivoju temeljnih poslovnih procesov je primeren delovenem toku dogodkov.

Po zaključeni zadnji aktivnosti se iz repozitorija poslovnih pravil izvozi dokumentacijo, izdela poročilo, ki predstavlja poslovni model in se ponovno pregleda s strani sponzorja (naročnika). Za nadaljevanje razvoja programske rešitve, se morajo ključni uporabniki s poročilom stirnjati. Po potrditvi se lahko začne informatizacija.

V primeru, da izdelujemo informacijski sistem se projekt razdeli na samostojno zaključene programske rešitve in je priporočljivo iterativno inkrementalno potrjevanje in informatizacija posameznih zaključenih sklopov. Pri takšnih primerih pa moramo paziti na spremembe osnovne strukture oziroma poslovnega slovarja je potrebno v primeru vpliva stukture na drugo programsko rešitev redefinirati in se sprmembami strinjati na višjih ravneh podjetja. To je potrebno zato, ker lahko sprememba strukture vpliva na druge že izdelane in vpeljane programske rešitve.

4.1.3 Informatizacija

Informatizacija poslovnega modela se prav tako razvija postopoma. Potrjena poslovna pravila se preda razvijalcem, ki pravila prevedejo v jezik, ki ga razume stroj za izvajanje poslovnih pravil. Razvijalci tudi skrbijo za objavo poslovnih pravil v realno okolje. Pogled iz vidika uporabe orodij je prikazan v razdelku 4.1.5.

Hitrost izdelane rešitve bo odvisna tudi od tehnologije, ki jo uporabljamo. V primeru, da obstaja avtomatiziran prenos poslovnih pravil, je rešitev lahko hitreje implementirana. Velikokrat se lahko pripeti, da v fazi informatizacije ugotovimo dodatne stvari, ki niso dovolj podrobno definirane, specificirane ali modelirane do potankosti. Tem primerom se lahko izognemo z učinkovitim poslovnim modeliranjem. Ravno te pomankljivosti pristop poslovnih pravil odpravlja.

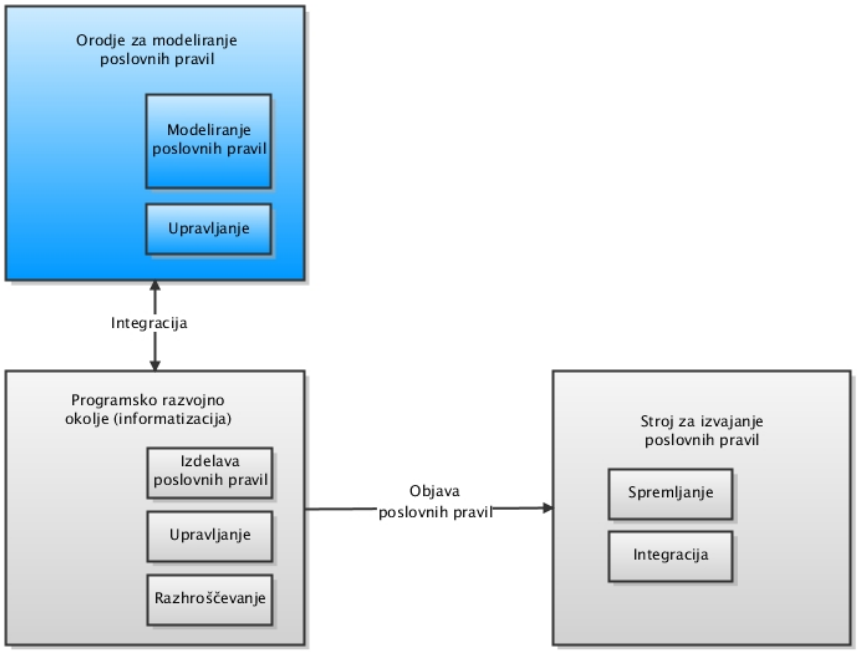
4.1.4 Testiranje in vpeljava

V fazi testiranja se uporabijo primeri uporabe, ki jih izdelamo v fazi poslovnega modeliranja in jih uporabimo za testne primere. Po drugi strani pa testiramo, ali so se poslovna pravila pravilno ovrednotila. In nenazadnje tudi pregledamo, ali so grafični uporabniški vmesniki v skladu s specifikacijo in željami končnih uporabnikov.

4.1.5 Opis razvoja programske rešitve s stališča tehnologije

Pogled iz vidika orodij, ki sem jih uporabljal pri informatizaciji programske rešitve, prikazje Slika 15. Aplikacijo za modeliranje poslovnih pravil (RuleXpress) je potrebno integrirati s programskim razvojnim okoljem, kjer se opis poslovnih pravl prilagodi glede na tehnologijo, ki jo uporabljamo (ILOG). Poslovna pravila se v produkcijsko okolje objavi na stroj za izvajanje poslovnih pravil. V stroju za izvajanje poslovnih pravil moramo spremljati delovanje poslovnih pravil in skrbeti za integracijo s razvojnim programskim okoljem.

Slika 15: Postopek razvoja programske rešitve s poudarkom na poslovnih pravilih



4.2 Orodja za opis poslovnega modela

Na svetovnem trgu je na voljo kar nekaj orodij, ki podpirajo izdelavo oziroma modeliranje poslovnih pravil. Po drugi strani obstaja manj orodij, ki podpirajo sistematičen način opisovanja, shranjevanja in upravljanja poslovnih pravil. Glede na to, da je v ospredju magistrske naloge predvsem poslovno modeliranje, sem se osredotočil samo na tista orodja, ki podpirajo poslovno modeliranje osnovnih poslovnih pravil. Tabela 6 opisuje nekatera orodja po sledečih kategorijah, povzeto po (Bajec, 2003):

- stroj za pravila je kategorija orodja, ki omogoča izvajanje poslovnih pravil;
- orodje za modeliranje poslovnih pravil je kategorija orodij, ki omogoča učinkovito upravljanje poslovnih pravil;
- celovito orodje ali rešitve je programski paket ali sistem paketov, ki omogoča celoten razvoj od upravljanja poslovnih pravil do izvajanja.

Glede na raziskano področje orodij za poslovno modeliranje, ki bi omogočalo izdelavo poslovnih pravil in repozitorij, sem se odločil, da bom v magistrski nalogi za poslovno modeliranje uporabil orodje RuleXpress, ki ga opisujem v 4. poglavju. Orodje razvijajo ljudje, ki so postavili temelje združenja BRCommunity in ga trži podjetje RuleArts. Del orodja RuleXpress je orodje FactXpress, ki pa se lahko uporablja samostojno.

Tabela 6: Orodja za upravljanje in informatizacijo poslovnih pravil

Podjetje	**Orodje**	**Kategorja**	**Kratek opis**
Usoft		Celovite rešitve	Promovira opis po naravnem jeziku na deklarativen način.
Jboss	Drools	Stroj za pravila	Stroj za izvajanje poslovnih pravil, ki temelji na aplikacijskem strežniku jboss.
IBM	ILOG	Stroj za pravila in modeliranje poslovnih pravil	Stroj za izvajanje poslovnih pravil in orodje za upravljanje s poslovnimi pravili.
RuleArts	RuleXpress	Modeliranje poslovnih pravil	Orodje za upravljanje poslovnih pravil, izrazov, dejstev in integracija z drugimi sistemi.
Open Rules	Open Rules	Celovite rešitve	Odprtokodni sistem za upravljanje poslovnih pravil.
Business rule solution	BRS RuleTrack	Modeliranje poslovnih pravil	Podpora za upravljanje poslovnih pravil in izgradnja poslovnega slovarja.
FICO	Blaze Advisor	Celovito orodje	Celovito orodje za upravljanje poslovnih pravil.

4.2.1 Opis orodja FactXpress

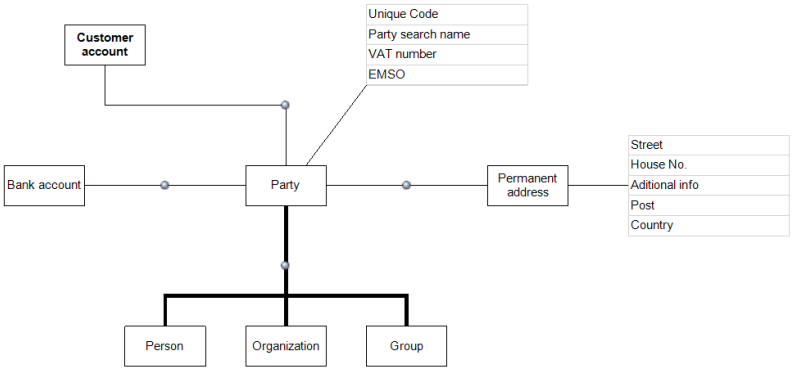
FactXpress je orodje za izdelavo modela izrazov in dejstev v grafični obliki. Orodje je integrirano z RuleXpress orodjem, kar pomeni, da se lahko izrazi in dejstva sinhronizirajo s poslovnim slovarjem, ki je definiran v RuleXpress. Primer modela dejstev opisuje Slika 16, ki je prikazana v angleškem jeziku, ker je programiranje v praktičnem delu magistrske naloge potekalo v angleškem jeziku.

Slika 16 prikazuje osnovna dejstva in izraze o osebi in si jo lahko razlagamo na sledeči način:

- osebi (angl. *party*) pripada stalni naslov, bančni račun (angl. *bank account*) in naročniški račun (angl. *customer account*);
- stalni naslov ima lastnosti: ulica, hišna številka, pošta in država;

- oseba pripada eni izmed kategorij: fizična oseba (angl. *person*), pravna oseba (angl. *organization*) ali skupina (angl. *group*);
- oseba ima sledeče lastnosti: EMŠO (angl. *EMSO*), davčna števila (angl. *vat number*), identifikacijsko številko (angl. *unique code*) in iskalne znake (angl. *party search name*).

Slika 16: Primer modela dejstev za osebo



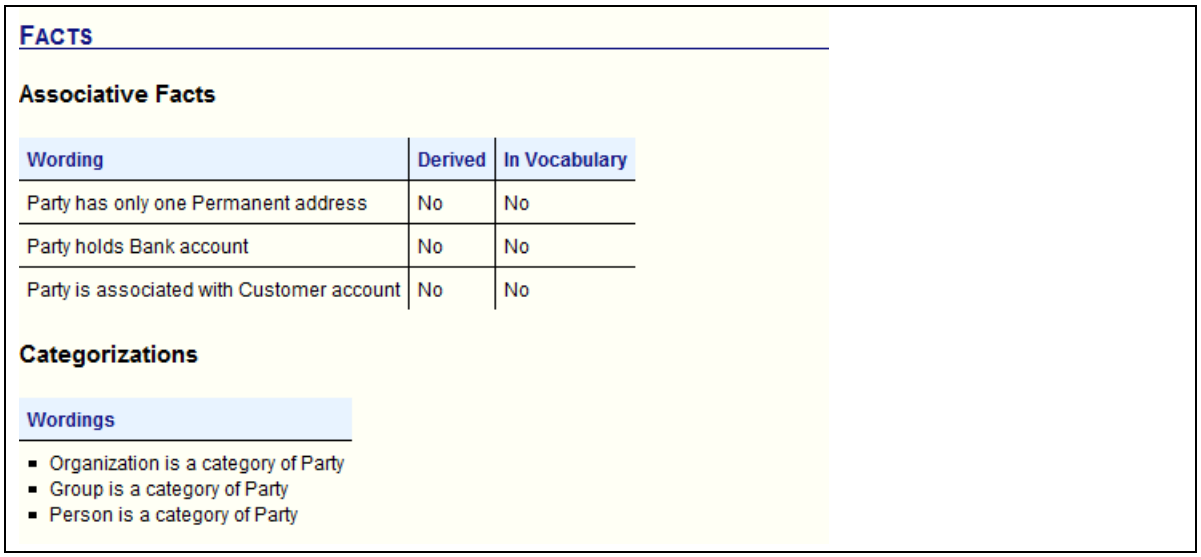
Iz modela dejstev lahko izdelamo poročilo, ki tabelarično prikaže osnovne izraze, njihove tipe, ali so izpeljani in ali se uporabljajo v poslovnem slovarju. Prav tako vidimo dejstva zapisana v strukturiranem naravnem jeziku in različne kategorizacije izrazov. Kompletno poročilo za programsko rešitev, ki sem jo izdelal v okviru magistrske naloge je tudi v prilogah. Podan je ilustrativni primer (Primer 8) kot izvleček.

Primer 8: Poročilo modela dejstev

TERMS

Terms

Signifier	Concept Types	Derived	In Vocabulary
Aditional info	Property	No	No
Bank account	Concept	No	No
Country	Property	No	No
Customer account	Concept	No	Yes
EMSO	Property	No	No
Group	Category	No	No
House No.	Property	No	No
Organization	Category	No	No
Party	GeneralConcept	No	No



FACTS

Associative Facts

Wording	Derived	In Vocabulary
Party has only one Permanent address	No	No
Party holds Bank account	No	No
Party is associated with Customer account	No	No

Categorizations

Wordings
▪ Organization is a category of Party ▪ Group is a category of Party ▪ Person is a category of Party

4.2.2 Opis orodja RuleXpress

RuleXpress je orodje, ki omogoča izdelavo poslovnega slovarja, vodenje evidence (repozitorja) poslovnih pravil in učinkovito upravljanje poslovnih pravil. Namenjeno je poslovnim analitikom in poslovnim uporabnikom za upravljanje sprememb poslovne logike. Med drugim lahko orodje uporabljajo zaposleni, ki oblikujejo slovar in poslovanje podjetja.

Osnovni namen orodja je, da poslovni uporabniki uporabljajo enotno besedišče in slovar. Uprabnik se lahko do repozitorija poslovnih pravil poveže lokalno ali pa preko strežnika, kjer lahko več uporabnikov dostopa do repozitorija. Zelo koristna funkcionalnost je podpora za skupinsko delo in dodeljevanje nalog ali opravil odgovornim in znalcem za določeno področje.

Na voljo so vsi osnovni gradniki in funkcionalnosti, ki so potrebni za izdelavo poslovnega modela, ki so:

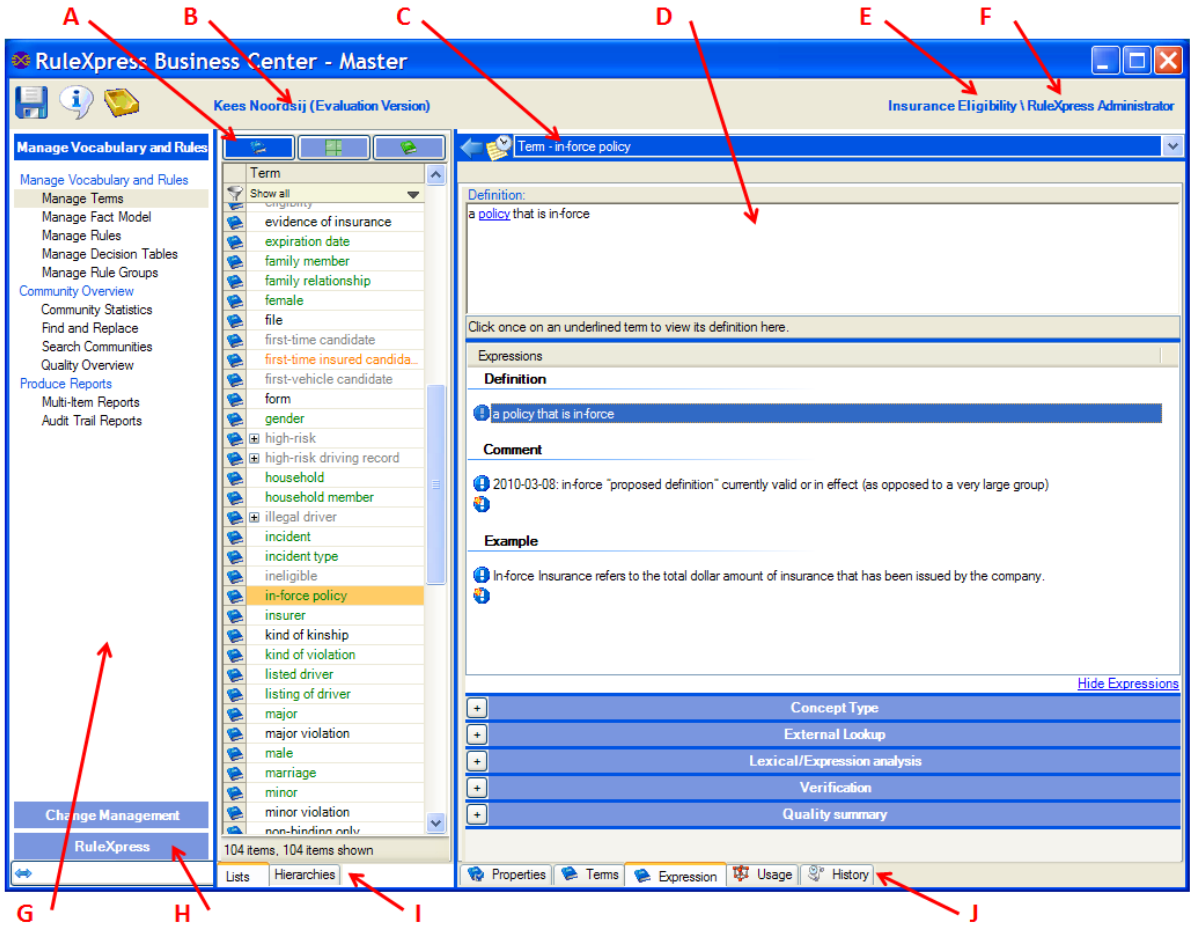
- izrazi;
- dejstva ali model dejstev;
- poslovna pravila;
- odločitvene tabele;
- skupine pravil.

Poleg osnovnih funkcionalnosti orodje omogoča izdelavo:

- različnih poročil;
- uvoz izrazov in poslovnih pravil iz drugih sistemov (npr. iz Excel-a);
- upravljanje z uporabniki in skupinami, ki jih lahko razdelimo po različnih projektih;
- interno komuniciranje s sporočili.

Slika 17 prikazuje izgled orodja in kratek opis ali osnove grafičnega uporabniškega vmesnika.

Slika 17: Grafični uporabniški vmesnik orodja RuleXpress



Legenda:

- A – pomočnik za prikazovanje seznamov;
- B – licenčne informacije;
- C – zgodovina navigacije;
- D – editor za urejanje različnih elementov;
- E – informacija, v kateri trenutni skupini delujemo;
- F – trenutno logirani uporabnik;
- G – možna izbira nalog ali aktivnosti;
- H – navigacija po različnih nalogah;
- I – zavihek za pomoč urejanje seznamov;
- J – urejanje zavihkov.

4.2.3 Prednosti in novosti orodja RuleXpress

Najbolj pogosto omenjene prednosti uporabe orodja RuleXpress so zbrane v naslednjih točkah iz strani uporabnikov orodja oziroma strank:

- Enostavno in učinkovito upravljanje z izrazi
  Upravljanje z izrazi omogoča enostavno navigacijo med izrazi, ugotavljanje, kje se uporabljajo in urejanje definicije izraza. Uporabniki lahko izraze pregledujejo pri izdelavi poslovnih pravil, ki bazirajo na omenjenih izrazih. Avtomatično prepoznavanje in sledljivost izrazov omogoča enostavno uporabo v poslovnih pravilih. Nenazadnje je olajšana izdelava poslovnega slovarja, možnost izdelave sinonimov in identifikacije hononimov.
- Enostavna uporaba za poslovne uporabnike
  Grafični uporabniški vmesnik je prilagojen in enostaven za poslovne uporabnike. Zaposleni, ki nimajo tehnične izobrazbe, lahko upravljajo s poslovnimi pravili. RuleXpress se izogiba prikazovanju psevdokode in podpira pisanje v lokalnem jeziku, velja samo za večje države, slovenščine ne podpira.
- Kvaliteta izrazov in poslovnih pravil
  Vgrajeno je preverjanje kvalitete poslovnih pravil in izrazov. Preverjanje pomaga pri kvaliteti in pravilnosti poslovnih pravil in izrazov. Preverjamo lahko, ali poslovna pravila res temeljijo na izrazih in dejstvih, ali so osnovni koncepti podrobno opisani, ali so izpeljani izrazi ustrezno definirani.
- Sledljivost in poročanje
  Orodje shranjuje vse podatke in relacije potrebne za sledljivost, ki so poslovna pravila in izrazi. Sledljivost omogoča repozitorij, iz katerega lahko pridobimo poročilo o spremembah. Shranjuje se, kdo in kdaj je spremenil določen izraz oziroma poslovno pravilo. Pri potrjevanju pravil se dodaja opombe in razloge za spremembo.
- Učinkovito preiskovanje poslovnih pravil
  Orodje RuleXpress omogoča napredno preiskovanje poslovnih pravil in njihovih relacij. Preiskovanje deluje po ključnih besedah tako po izrazih, kot po poslovnih pravilh in drugih elementih. Rezultate lahko tiskamo in delimo z drugimi za lažje sodelovanje.
- Razvoj poslovnih pravil je avtomatiziran
  Različne skupine in uporabniki lahko izdelajo, spreminjajo in potrjujejo poslovna pravila skozi razvojni cikel poslovnih pravil. Fleksibilno poročanje omogoča pregled po posameznih korakih razvoja. Poslovna pravila lahko imajo določene statuse, kot so izdelano pravilo, pravilo v pregledovanju, aktivirano pravilo, pravilo v produkciji, ali pa se statuse nastavi po potrebi.
- Referenčna knjižnica
  RuleXpress lahko služi kot referenčna knjižnica. Omogoča shranjevanje poslovnih pravil, izrazov, podpornih dokumentov, evidenc in implementacijskih detajlov.
- Možnost povezovanja
  Pomembna je tudi lastnost povezovanje z drugimi sistemi. Uvozimo lahko poslovna pravila in izraze iz drugih sistemov tako, da lahko uporabimo obstoječe opise poslovnih pravil in izrazov. Omogočen je tudi izvoz elementov v XML (angl.

extensible markup language) obliko, ki jo lahko uvozimo v produkcijske sisteme in stroje za izvajanje poslovnih pravil.

4.3 Orodja za informatizacijo

Podobno kot druga orodja obstaja na svetovnem trgu na voljo več različnih programskih orodij ali paketov za informatizacijo programske rešitve. Raziskal sem 3 takšna orodja: OpenRules, Drools, ILOG.

OpenRules ponuja celo paleto rešitev za upravljanje poslovnih pravil (angl. *business rule management system*, okr. BRMS). Programski paket je odprtokodni in v njemu lahko razvijamo programske rešitve, ki bazirajo na osnovi poslovnih pravil. Pri razvoju poslovnega modeliranja se uporablja MS Excel orodja ali preglednice podjetja Google, za programsko razvojno okolje pa se uporablja orodje Eclipse IDE. Repozitorij poslovnih pravil je shranjen v preglednicah MS Excel, ki jih lahko kategoriziramo po zavihkih (Business Rules Repository, 2010).

Drools je zelo popularen in prav tako odprtokodni sistem za upravljanje poslovnih pravil. Ponuja 5 produktov za vse aktivnosti, ki so potrebne za informatizacijo programske rešitve. Za poslovno modeliranje lahko uporabniki uporabljajo spletni uporabniški vmesnik. Poslovna pravila se vnašajo in urejajo v produktu, ki se imenuje Guvnor. Sinhronizacija poslovnih pravil je implementirana med Guvnor orodjem in programskim razvojnim ogrodjem Eclipse (Drools Guvnor - JBoss Community, 2010).

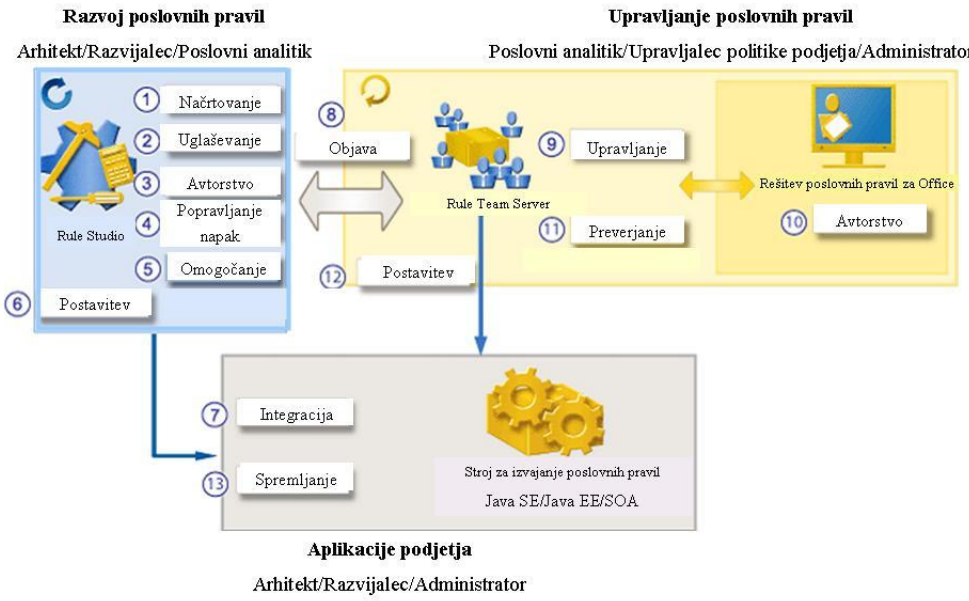
Tudi ILOG je orodje, ki omogoča modeliranje poslovnih pravil in ima stroj za izvajanje poslovnih pravil. V osnovi je zelo podoben Drools-u: ima spletno aplikacijo, kjer lahko modeliramo in upravljamo poslovna pravila, jih integriramo v RuleStudio, ki je del programskega razvojnega okolja Eclipse. Razlikuje se v tem, da je to komercialno plačljivo orodje. Potrebno je tudi omeniti, da ga je pred leti kupila ena izmed treh velikih programskih hiš IBM, zato se sedaj paket imenuje WebSphere ILOG JRules BRMS.

V magistrski nalogi sem za razvoj programske rešitve uporabil ILOG komercialno orodje, ki omogoča lažji razvoj izbrane programske rešitve.

4.3.1 Opis orodja ILOG

Paket ILOG vključuje več različnih programskih orodij. Za poslovno modeliranje je na voljo spletni strežnik z aplikacijo TeamServer, ki omogoča urejanje poslovnih pravil preko spletnega vmesnika. Za nameščanje in vzdrževanje spletnega oziroma aplikacijskega strežnika je priporočljivo znanje o omrežnih aplikacijah. TeamServer lahko integriramo na različne aplikacijske strežnike, kot so tomcat, jboss, orcale weblogic. Obstaja tudi nadgradnja ali podaljšek aplikacije TeamServer, ki omogoča uporabo programov MS Office. Arhitekturo ILOG paketa podrobneje opisuje Slika 18.

Slika 18: Arhitektura ILOG paketa



Vir: IBM, interno gradivo, 2009.

Nameščanje in spoznavanje celotnega paketa ILOG ni tako enostavno, saj je potrebno povezati več programskih komponent: RuleStudio, Eclipse, TeamServer, tomcat, program MS Office. Osrednjo programsko orodje je RuleStudio, kjer poteka razvoj poslovnih pravil in informatizacija. RuleStudio je del oziroma razširitev integriranega razvojnega okolja Eclipse, podobno kot Drools in OpenRules. Uporabniki so lahko poslovni analitiki, razvijalci ali pa arhitekti.

4.3.2 Opis orodja RuleStudio in osnovnih gradnikov ILOG-a

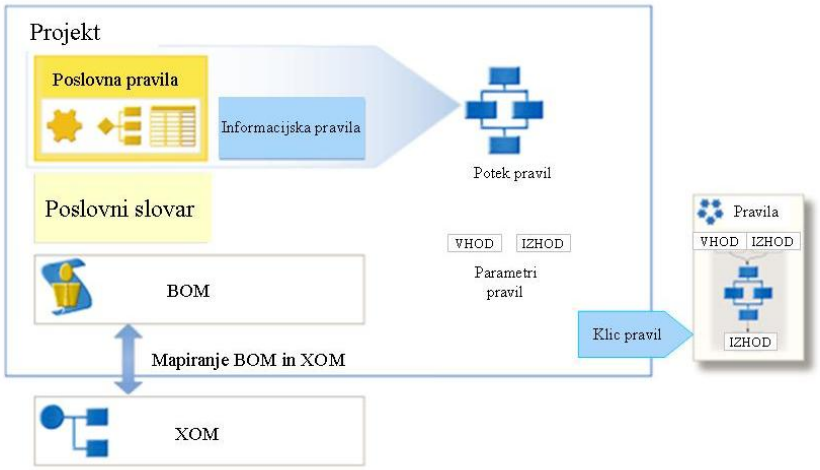
RuleStudio omogoča izdelavo in kategorizacijo programskih rešitev, ki jih lahko razdelimo modularno po projektih. Vsako programsko rešitev (v nadaljevanju projekt) lahko zapakiramo v storitev, ki jo lahko integriramo kot knjižnico v druge javanske projekte ali pa kot spletno storitev. Za uporabo orodja RuleStudio je potrebno razumevanje nekaterih osnovnih gradnikov, ki so prikazani na Sliki 19:

- projektna mapa;
- XOM (angl. *extended object model*);
- BOM (angl. *bussiness object model*);
- potek (angl. *flow*) ali skripta;
- prikazovalnik gradnikov;
- predloga za izdelavo dejstva.

Projektna mapa nam služi kot vodilo pri izdelavi projekta. Dejansko nas vodi pri izdelavi ostalih gradnikov. Omejuje nas v tolikšni meri, da ne moremo narediti nedovoljene akcije

(npr. ne moremo izdelovati poslovnih pravil, če nimamo definiranega poslovnega slovarja). Potek projekta prikazuje Slika 19.

Slika 19: Razvoj programske rešitve ali projekta v orodju ILOG



Vir: IBM, interno gradivo, 2009.

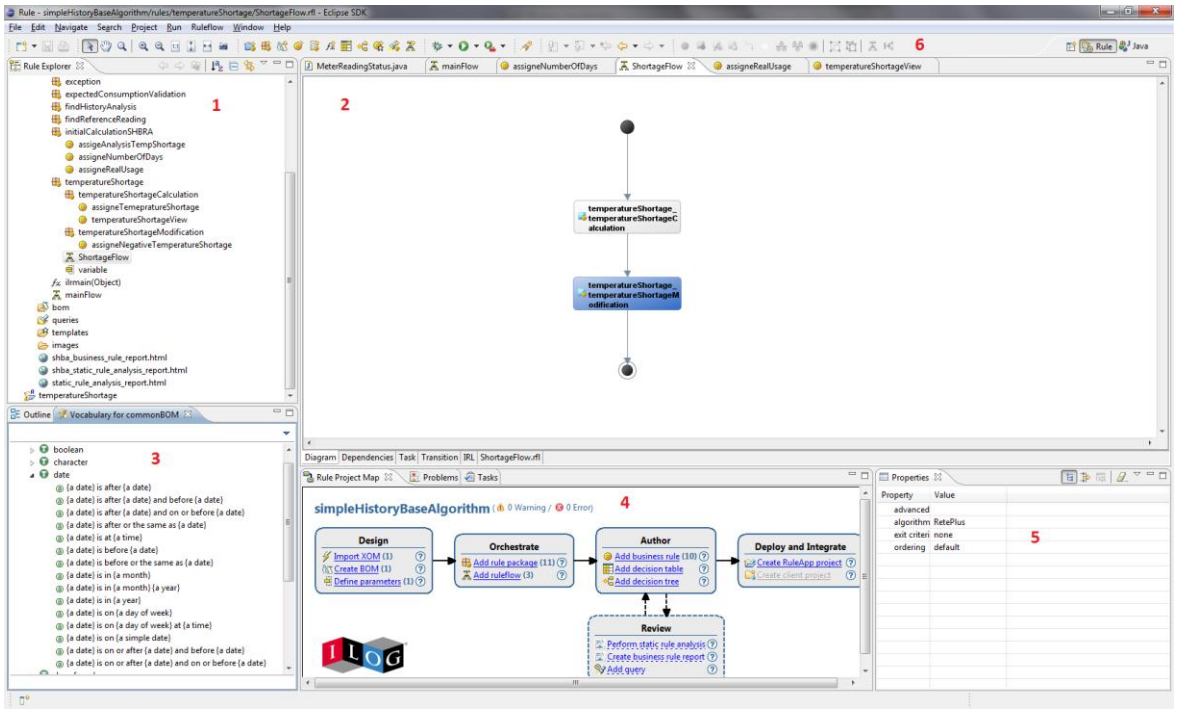
Izrazi in dejstva iz poslovnega modela se preslikajo v razredni diagram. Na podlagi razrednega diagrama se izdela ustrezne javanske razrede. Javanske razrede običajno implementiramo v ločenem projektu in jih uvozimo v projekt poslovnih pravil preko XOM gradnika. Nato izdelamo mapiranje med XOM in BOM modelom, ki dejansko predstavlja poslovni slovar oziroma osnovno strukturo projekta. Pomemben element javanskih razredov so tudi konstruktorji in njegove metode. Metode razreda definirajo obnašanje in dejstva. Iz metode se sestavi predloga ali stavek (angl. *template*), ki lahko nekaj določa, medtem ko so izrazi podatki (angl. *placeholder*), ki se vrinejo v predlogo.

Glede na to, da ima java v osnovnih knjižnicah že vgrajene nekatere predloge stavkov za vrsto stvari, kot na primer logične operatorje in časovne operatorje, je delo z orodjem ILOG oljašano. Potek pravil je potrebno organizirati v zaporedje. Dejansko se izdela graf poteka, ki ima lahko tudi vgnezdene poteke. V zadnjih elementih grafa poteka ali v listu drevesa so paketi, ki imajo seznam poslovnih pravil. Pakete imenujejo tudi taski. Pravila v taskih se vrednotijo ali izvajajo po znanem algoritmu imenovanem RETE algoritem.

Potrebno je tudi omeniti testiranje in odpravljanje napak. Razhroščevanje poteka zelo enostavno, saj se po korakih spremlja delovanje in klicanje poslovnih pravil po diagramu poteka. Za testiranje lahko definiramo primere uporabe, kar lahko storimo tudi s pomočjo MS Excel dokumenta, in na drugi strani preverjamo rezultate, ali so se poslovna pravila ovrednotila, tako kot si želimo oziroma kot smo napovedali.

Grafični uporabniški vmesnik je dokaj zapleten in si ga lahko poljubno prilagajamo, kar je funkcionalnost orodja Eclipse. Kratka razlaga privzete nastavitve prikazuje Slika 20.

Slika 20: Grafični uporabniški vmesnik razvojnega okolja ILOG - RuleStudio



Legenda:

1- Pregledovalnik paketov pravil ali razredov;
2- Prikazovalnik gradnikov (odvisno od izbranega gradnika, trenutno je prikazan potek taskov);
3- Pregledovalnik poslovnega slovarja;
4- Projektna mapa;
5- Lastnosti izbranega elementa v prikazovalniku gradnika;
6- Orodna vrstica.

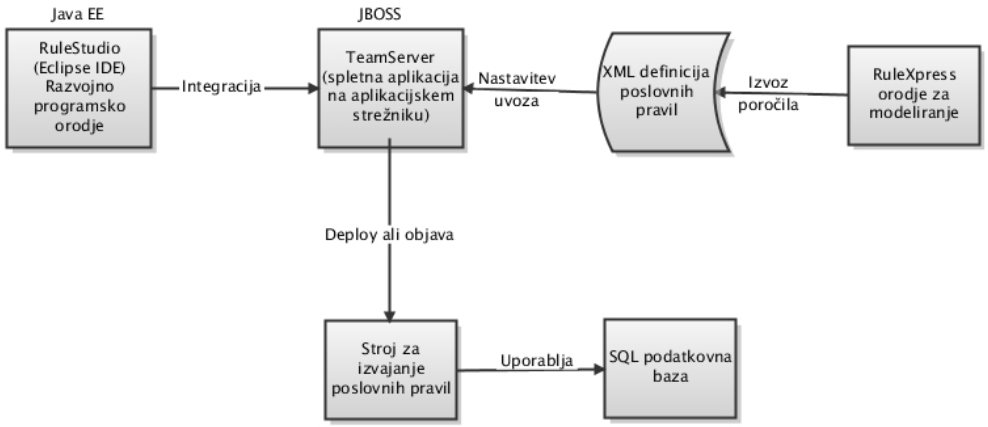
4.3.3 Integracija ILOG in RuleXpress orodij

Pri praktičnem delu magistrske naloge sem imel v načrtu povezati oziroma integrirati orodje za poslovno modeliranje RuleXpress in orodje za informatizacijo poslovnih pravil ILOG. Kljub temu, da so obstajala podrobna navodila povezovanja obeh orodij, moram omeniti, da se je pri tem delu pojavilo veliko težav, ki so omenjene v nadaljevanju.

Orodje RuleXpress ima možnost izvoza poslovnih pravil v XML strukturi, ki je definirana. Na drugi strani pa lahko nastavimo TeamServer aplikacijo tako, da vanjo lahko uvozimo XML strukturo. Uvoz je sicer bil uspešen, vendar se pri podrobnem pregledu podatki niso ustrezno nastavili. Zato sem transformacijo v TeamServer in RuleStudio naredil ročno s prepisom poslovnih pravil.

Slika 21 prikazuje shematski prikaz povezav različnih orodij in tehnologij, ki sem jih uporabil pri razvoju programske rešitve analiza odčitka na podlagi zgodovine.

Slika 21: Shematski prikaz povezav orodij in tehnologije



4.4 Analiza pristopa poslovnih pravil : problemi in rešitve

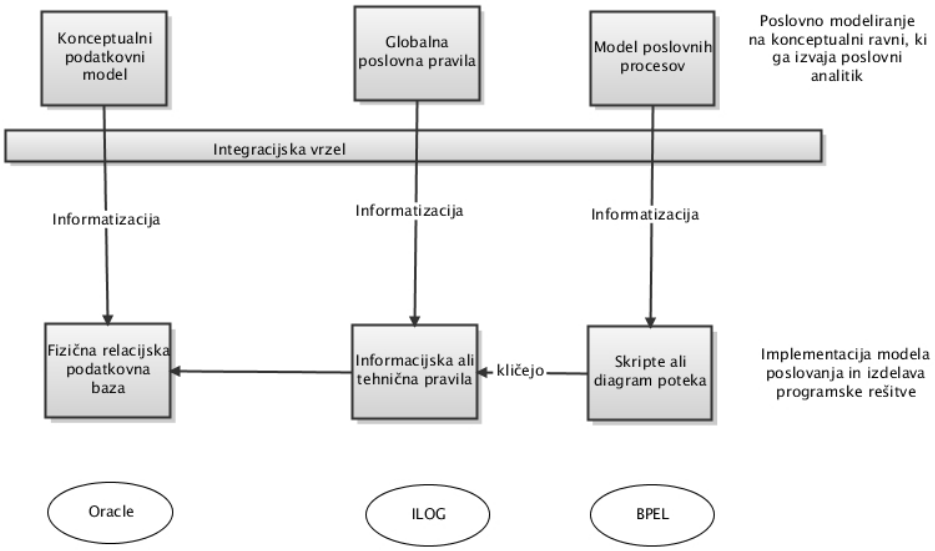
Prvi del razvoja programske rešitve je potekal brez večjih problemov. Zbiranje zahtev in določanje taktike ni bilo zahtevno. Taktični načrt se je iskazal kot učinkovito orodje za sprejemanje odločitev o nadaljevanju razvoja programske rešitve.

Tudi v nadaljevanju pri izdelavi poslovnega modela ni bilo večjih težav. Časovno bolj zahtevno je bilo prilagajanje na uporabniški vmesnik orodja RuleXpress. Zanimivo je, da je sponzor programske rešitve potrdil izdelani in predlagani poslovni model in ni imel pripomb.

Integracija poslovnega modela in razvojnega programskega orodja ni v celoti uspela, saj ILOG ni znal pravilno interpretirati poslovnih pravil. Predvidevam, de je težava v obratnem načinu predstavitve predikatne logike. ILOG uporablja programerski način logike, kar pomeni najprej preverjanje pogoja (angl. *if*), potem (angl. *then*) izvrševanje akcije. Pravila sem pisal po načelih poslovnega pristopa, ki subjekt daje v ospredje in je struktura poslovnega pravila naslednja: najprej se zapiše akcija, nato sledi zapis pogoja. Menim, da je rešitev povezovanja v standardizaciji poslovnih pravil, kjer se že pojavljajo rešitve, saj je združenje OMG že pripravilo specifikacijo SBRV in PRR, medtem ko združenje RuleML pripravlja translacijske jezike v obliki XML shem.

Tudi druga področja poslovnega modeliranja imajo problem integracijske vrzeli. Slika 22 prikazuje, kako se 3 področja poslovnega modeliranja informatizirajo v konkretne rešitve v določenih tehnologijah.

Slika 22: Metamodel informatizacije programske rešitve



Zanimivo je, da kljub vsej tehnologiji in transformacijskih mapiranjih še vedno najdemo anomalije v programskih rešitvah med definicijo poslovnega modela ali poslovnega slovarja, fizično relacijsko podatkovno bazo in mapiranjem v razredni diagram oziroma objektni svet. Poslovni analitiki dobro vedo, kaj pomeni dodajanje novega izraza oziroma atributa ali tabele v programsko rešitev. Sam sem moral najrej popraviti poslovni model, nato sem dodal atribut na tabelo in na koncu še popravil mapiranje v javanskem razredu. To dokazuje, da tudi pri podatkih obstaja integracijska vrzel in ni avtomatiziranega prehoda v informatizacijo.

5 OPIS IN PREDSTAVITEV IZDELAVE PROTOTIPA

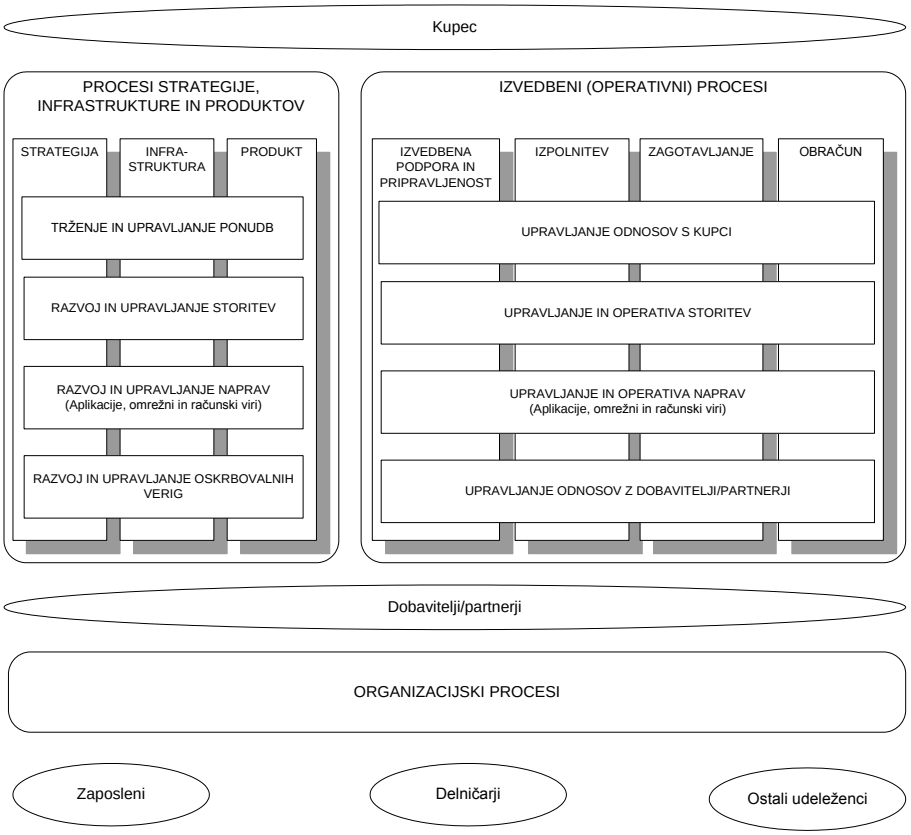
V okviru magistrskega dela je bil cilj analizirati pristop poslovnih pravil razvoja programske rešitve. Zadani cilj se lahko izpolni samo tako, da se izdela prototipni projekt in razvije programsko rešitev. Izbral sem prototipni projekt obratnega inženirstva in ponovne dokumentacije za sistem obračunavanja. Pri izvajanju projekta sem upošteval načela in vodila poslovnega pristopa. V tem poglavju bom predstavil rezultate praktičnega dela.

Obračunski sistem je vpet v širše ogrodje poslovnih procesov eTOM (angl. *enhanced telecom operations map*). Mednarodno združenje ponudnikov informacijsko-telekomunikacijskih (okr. ITkT) storitev TeleManagement Forum (TMF) je v okviru pobude NGOSS (angl. *new generation operations support system*) oblikovalo ogrodje poslovnih procesov ponudnika storitev imenovano eTOM. Ogrodje se uporablja pri prenovi poslovnih procesov (angl. *business process reengineering*, BPR) ponudnikov

internetnih storitev, storitev stacionarnih, mobilnih govornih in podatkovnih komunikacij itd. V nadaljevanju sledi opis ogrodja eTOM in njegovo uporabo pri informatizaciji in prenovi poslovnih procesov ponudnikov storitev. Pri tem se eTOM ne omejuje zgolj na ponudnike ITkT storitev, temveč razlaga tudi možnost uporabe pri ponudnikih komunalnih storitev, ki so ponudniki oskrbe s plinom, daljinskim ogrevanjem, električno energijo in pitno vodo (Keber, 2003).

Ogrodje eTOM, prikazano na Sliki 23, razdeli poslovni sistem na strateške, organizacijske in operativne (izvedbene) skupine procesov. Osrednja točka modela so operativni procesi, ki zadevajo kupce: te procese imenujemo FAB (angl. *fulfillment, assurance, billing*). Gre torej za procese izpolnitve zahteve kupca (dodelitev produkta), zagotavljanja dogovorjene ravni storitve in obračunavanja (u)porabljenih produktov. Operativna podpora in pripravljenost (1. steber operativnega dela modela) sta ločeni od FAB procesov, da bi poudarili, da ti procesi zagotavljajo podporo in avtomatizacijo FAB procesov. Levi del modela na Sliki 23 predstavlja strategijo in upravljanje življenjskega cikla (infrastrukture in produktov). Procesi, opisani v tem delu, niso namenjeni neposredni podpori kupcem in so bistveno drugačni od operativnih procesov predstavljenih na desnem delu Slike 23. Levi in desni del modela se razlikujeta tudi po življenjskih ciklih procesov, ki pada od leve proti desni.

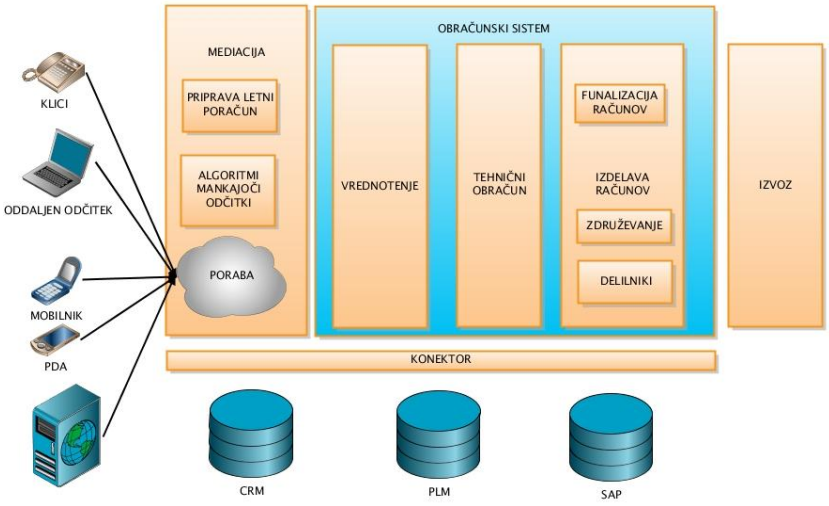
Slika 23: Ogrodje poslovnih procesov eTOM



Vir: B. Keber, M. Krisper & T. Gornik, Ogrodje poslovnih procesov ponudnikov storitev, 2003, str. 4.

Obračunavanje (angl. *billing*) je proces, ki zagotavlja pravočasno in natančno izdelavo računov, obveščanje o porabi pred izdajo računa in za obdelavo plačil. Proces skrbi tudi za reševanje težav pri obračunavanju in podpira predplačniške storitve. Arihtekturo obračunskega sistema prikazuje Slika 24. Obračunski sistem se preko konektorjev povezuje z drugimi zunanjimi sistemi, kjer pridobiva podatke. Iz sistema CRM (angl. *custom related management*) pridobiva podatke o strankah in dodeljenih produktih. Iz produktnega kataloga pridobiva podatke o sestavi paketov za obračunavanje in druge tehnične podatke, ki so za obračun pomembni. V večini primerov je potrebno iz finančnega sistema na račune povezovati finančne vrstice, kot so obresti in preplačila, zato je možno povezovanje s SAP sistemom.

Slika 24: Arhitektura obračunskega sistema



Vir: Marand d.o.o., Arhitektura obračunskega sistema (interno gradivo), 2010.

Pred začetkom projekta obratnega inženirstva in ponovne dokumentacije sistema za obračunavanje je bilo potrebno zapisati motivacijo za projekt in uporabniške zahteve. Dejstvo je, da je pri obstoječi programski rešitvi nastalo veliko projektne dokumentacije v različnih zapisnikih. Spremembe zahtev in poslovnih pravil so bile zelo intenzivne, zato ni več centralnega pogleda na implementirana poslovna pravila. Manjkal je tudi popoln slovar pojmov in definicij osnovnih konceptov, saj je bil obstoječi slovar pojmov zastarel.

5.1 Taktični načrt projekta za obračunavanje

V okviru zajema zahtev in določanja taktike sem izdelal taktični načrt, ki je opisan okrnjeno zaradi poslovnih skrivnosti.

Primer 9: Taktični načrt projekta za obračunavanje

Poslovni cilji
Zajeti ogrodje dejstev in poslovnih pravil.
Zagotoviti upravljanje s poslovnimi pravili.
Zmanjšati stroškov sprememb, ki jih je pri projektih s področja energetike veliko.
Deliti znanje na spletu (angl. *on-line*) z deležniki.

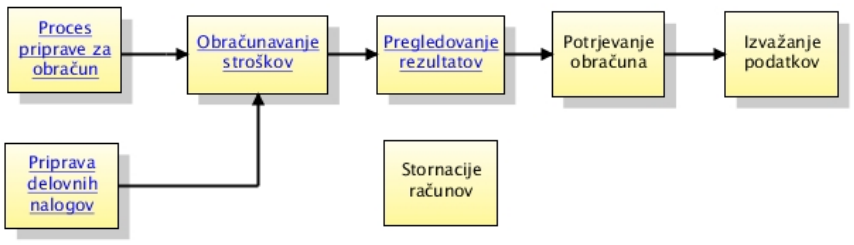
Taktika
Pregled dokumentacije in programske kode in zapisati poslovna pravila.
Uporaba orodja RuleXpress za specifikacijo poslovnih pravil.
Izdelava modela dejstev, ogrodja za poslovna pravila in slovar.
Zajemanje poslovnih pravil iz programske kode za obračunavanje.

Tveganje
Celotne poslovne logike ni možno zapisati.

Zmanjšanje tveganja
V primeru, da celotne poslovne logike ni možno zapisati, bomo zajeli samo enostavne rešitve, ki se najbolj spreminjajo.

Obračunski sistem je povezan s temeljnimi poslovnimi procesi na operativni ravni. Proces obračunavanja se izvaja enkrat mesečno in je v grobem opisan na Sliki 25. Proces obračunavanje stroškov je podproces operativnega procesa obračunavanja in se deli naprej na podprocese: vrednotenje porabe, tehnični obračun in fakturiranje ali izdelava računov. Podprocesi pa imajo definirane svoje postopke izvajanja.

Slika 25: Poslovni proces obračunskega sistema



Potrebno je tudi omeniti, da se v podrobnosti projekta ponovne dokumentacije za sistem obračunavanja ne bom spuščal, ker je projekt zaupne narave. Bom pa prikazal manjšo programsko rešitev, ki sem jo razvil v celoti in se nahaja v okviru procesa priprave na obračun in se uporablja v aktivnosti generiranja računskih odčitkov in pri izvajanju analize odčitkov.

5.2 Programska rešitev za analizo odčitkov

Pri projektu sem ugotovil slabosti pri analiziranju odčitka, zato sem izvedel prenovo. Za podprojekt analize odčitkov na podlagi zgodovine za sistem toplote sem izvedel celoten razvojni cikel programske rešitve »Vgradnja algoritmov za analizo odčitka na podlagi zgodovine odčitkov«, bom v nadaljevanju naslavljal kot analiza odčitka na podlagi zgodovine. Na začetki je bilo potrebno je določiti uporabniške zahteve ali potrebe (Primer 10).

Primer 10: Uporabniške zahteve programske rešitve analiza odčitkov

Osnovni namen algoritma je preverjanje pravilnosti vnešenih odčitkov. Upoštevaj posebnosti odjemnih mest odjema toplote, da so odčitki odčitujejo mesečno. Izdelati algoritem, ki bo v 80 % pravilno napovedal porabo za en mesec. Algoritem mora pri izračunu upoštevati enoletno zgodovino porabe. Algoritem mora biti enostaven za razumevanje, da ga je možno razložiti končnim uporabnikom. Algoritem deluje samo na vrednostih glavnih odčitkov merilnih naprav. Glavni odčitek je tisti, ki se v danem trenutku obračunava. Primer: Na toplotnem števcu je glavni odčitek odčitek energije in ne odčitek vodnega dela.

Najprej sem izdelal konceptualni ali taktični načrt, ki je prikazan s Primerom 11, in poslovni slovar. Uporabljeni izrazi in dejstva so zapisani v poglavju 5.2.1.

Primer 11: Taktični načrt programske rešitve analiza odčitkov

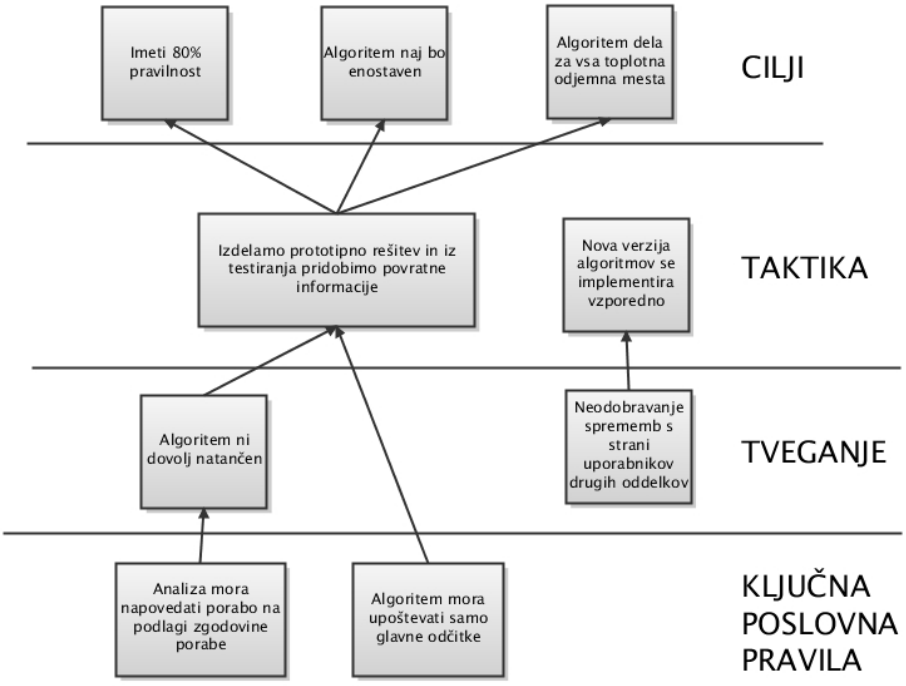
Cilji Izhajajo iz uporabniških zahtev. Izdelati algoritem, ki bo analiziral odčitke za sistem toplote na podlagi zgodovinske porabe. Izdelati analizo, ki bo v 80 % pravilno napovedala porabo. Obstoječ algoritem ima 40 % pravilnost. Algoritev mora biti enostaven za uporabo in razumevanje. Obstoječ algoritem ima veliko fizikalnih lastnosti, izračunljivih enačb in je preveč zapleten. Algoritem mora delovati za vsa odjemna mesta na toplotnem sisitemu. Razvoj algoritma analize na podlagi zgodovine moramo izdelati v enem mesecu. **Taktika** Izdelamo prvo prototipno rešitev, ki je funkcionalna v enem tednu. Testiramo rešitev in na podlagi povratnih informacij izdelamo dve iteraciji možnih izboljšav. **Tveganje**

1. Lahko se pripeti, da algoritem ne bo dovolj natančen.
2. Uporabniki iz drugega oddelka organizacije obstoječega algoritma analize ne odobravajo sprememb, ker so navajeni na trenuten potek dela.

Ukrepi za zmanjšanje tveganja
1. Pred izdelavo rešitve izdelamo 5 testnih primerov. Razmislimo o možnih izboljšavah in podamo predloge, predno se izvede razvoj.
2. Novo verzijo algoritmov uporabljamo vzporedno v aplikaciji. Obstoječa verzija deluje brez sprememb.

Taktični načrt se lahko predstavi tudi v grafični obliki (Slika 26), ki je mnogim deležnikom lažje razumljiv.

Slika 26: Grafični prikaz taktičnega načrta



Omenim naj nekaj statističnih podatkov za projekt analiza odčitkov. Pri izgradnji modela dejstev sem definiral približno 132 izrazov in približno 59 dejstev. Določil sem 5 skupin poslovnih pravil, kar vodi v 19 različnih poslovnih pravil, med njimi so izpeljave, omejitvena praila in izračunljva pravila.

5.2.1 Izdelava podatkovnega modela oziroma strukture

Podatkovni model je bil izpeljan iz razrednega diagrama, ta pa iz modela dejstev. Fizični podatkovni model se je gradil na podlagi podatkovne podpore, ki jo je konkretna rešitev potrebovala.

Na Sliki 27 je prikazan model dejstev izdelan v RuleXpress orodju za programsko rešitev analiza odčitkov. Model dejstev se sinhronizira s poslovnim slovarjem, ki ga lahko uporabljamo pri izdelavi poslovnih pravil in skripte, ki opisuje potek.

Osnovni koncepti prikazani na Sliki 27 so:

- odjemno mesto (angl. *site*) je prevzemno mesto odjemalca, kjer je nameščena merilna naprava in se meri predani zemeljski plin;
- odčitek (angl. *meter reading*) je datumsko in vrednostno opredeljen izraz, ki ga producira merilna naprava;
- analiza odčitkov na podlagi zgodovine (angl. *simple heat meter reading analisys*) je po algoritmu analiziran odčitek.

Osnovna dejstva prikazana na Sliki 27 so:

- odjemno mesto (angl. *site*), ki ima lahko več odčitkov (angl. *meter reading*);
- vsak odčitek (angl. *meter reading*) bi moral imeti analizo odčitka (angl. *simple heat meter reading analysis*).

Odjemno mesto je lahko v kategoriji tipa porabe odjemnega mesta (Slika 27):

- konstantne moči in poraba odjemnega mesta (angl. *constant powers*);
- dinamične moči in poraba odjemnega mesta (angl. *variable powers*).

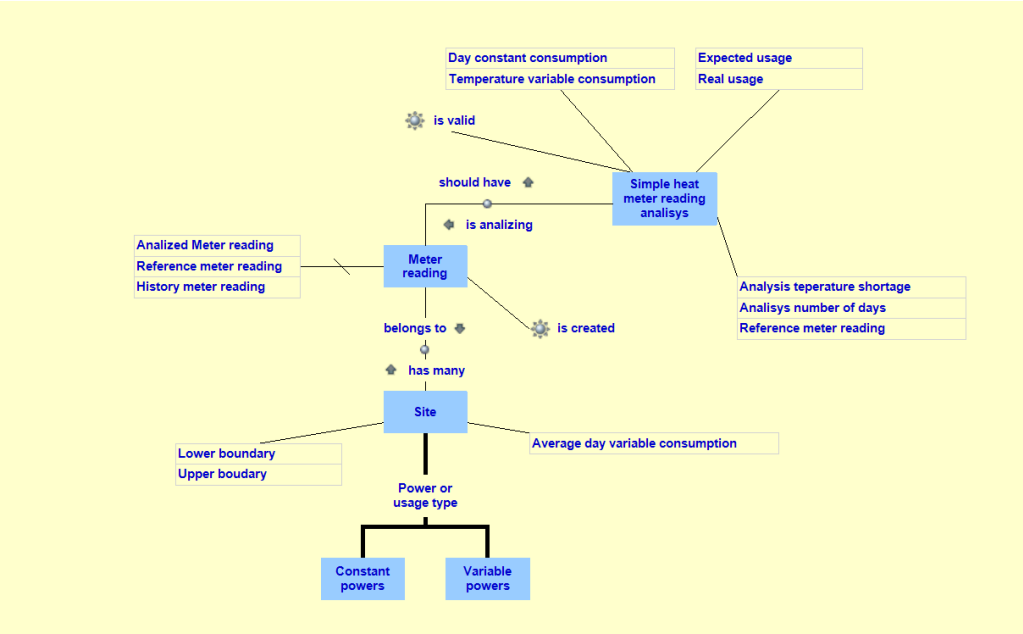
Odčitek lahko nastopa v treh primerih (Slika 27):

- analiziran odčitek (angl. *analized reading*);
- referenčni odčitek (angl. *referenced reading*);
- zgodovinski odčitek (angl. *history reading*).

Vsaka analiza ima sledeče lastnosti (Slika 27):

- število dni analize (angl. *analisys number of days*);
- stopinjski dnevi analize (angl. *analisys temperature shortage*);
- referenčni odčitek (angl. *reference meter reading*);
- realno porabo (angl. *real usage*);
- pričakovano porabo (angl. *expected usage*).

Slika 27: Model dejstev za analizo odčitkov na podlagi zgodovine



Ostale podrobnosti modela dejstev za programsko rešitev analize odčitkov so zbrane v Prilogi 1.

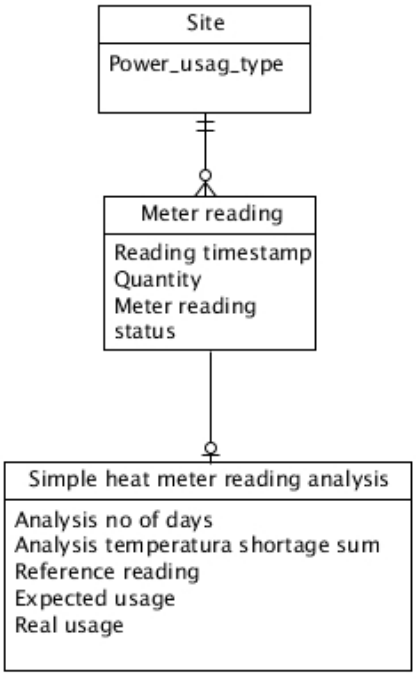
Model dejstev za delovanje rešitve seveda ne zadošča. Za shranjevanje dejanskih podatkov oziroma primerkov je potrebno imeti podatkovno podporo, ki omogoča persistenco podatkov. Zato sem potreboval 3 tabele, in sicer za odjemno mesto in odčitek, ki sta že prisotni v obstoječi rešitvi, ter tabelo za analizo, ki jo je bilo potrebno dodati, njen izvedbeni SQL stavek pa je prikazan v Primeru 12.

Razrede sem na podlagi modela dejstev izdelal v razvojnem programskem orodju Eclipse. Razredni diagram prikazan na Sliki 28 služi kot osnova za BOM. Mapiranje med razrednim in relacijskim podatkovnim modelom je bilo potrebno sprogramirati.

Primer 12: Primer izdelave fizičnih podatkovnih objektov

```
-- Create table
create table HISTORY_BASED_READING_ANALYSIS
(
  HISTORY_BASED_ANALYSIS_ID     NUMBER not null,
  DAY_CONSTANT_CONSUMPTION      NUMBER(20,10),
  TEMPERATURE_VAR_CONSUMPTION   NUMBER(20,10),
  ANALYSIS_NUMBER_OF_DAYS       NUMBER(10),
  ANALYSIS_TEMPERATURE_SHORTAGE NUMBER(20,10),
  EXPECTED_CONSUMPTION          NUMBER(20,10),
  CONSUMPTION                   NUMBER(20,10),
  NOTE                          VARCHAR2(2000)
 );
-- Create/Recreate primary, unique and foreign key constraints
alter table HISTORY_BASED_READING_ANALYSIS
  add primary key (HISTORY_BASED_ANALYSIS_ID)
  using index
  (
    initial 64K
    minextents 1
    maxextents unlimited
  );
alter table HISTORY_BASED_READING_ANALYSIS
  add foreign key (HISTORY_BASED_ANALYSIS_ID)
  references METER_READING_ANALYSIS (METER_READING_ANALYSIS_ID);
```

Slika 28: Razredni diagram analize odčitkov na podlagi zgodovine



Na Sliki 28 ni razvidno, da je tip porabe (angl. *power usage type*) odjemnega mesta (angl. *site*) enumeracija, ki je statična in je definirana s statičnimi instancami razreda.

5.2.2 Skripta programske rešitve

Izdelava skripte ali poteka delovanja za preverjanje analize odčitka na podlagi zgodovine je bila enostavna, kar je prikazano v Primeru 13.

Primer 13: Skripta za analizo odčitka

1. Poišči referenčni odčitek analize.
2. Izračunaj temperaturni primankljaj obdobja za dnevno porabo glede na trenutni in referenčni odčitek.
3. Izračunaj dnevno konstantno porabo in stopinjsko porabo.
4. Pridobi podatek o zgodovinski dnevni porabi in zgodovinski stopinjski porabi.
5. Izračunaj pričakovano porabo.
6. Primerjaj dejansko porabo referenčnega odčitka in izračunano pričakovano porabo.

Dejansko se potek dela razvija vzporedno s podatkovnim modelom, saj moramo pri izdelavi rešitve razmišljati tudi o nekaterih dejstvih, kot so kako bomo lahko pridobivali podatke in jih preiskovali. Potrebno je poudariti, da je s tem samo definiran potek dela, ki še nima določenih poslovnih pravil. Poslovna pravila, ki določajo poslovno logiko, so implemenitrana ločeno.

5.2.3 Implementacija poslovnih pravil

V tem poglavju bom predstavil nekaj primerov implementacije poslovnih pravil iz programske rešitve analiza odčitkov na podlagi zgodovinske porabe. Glede na to, da je sistem obračunavanja lažje razumljiv, bom v naslednjem poglavju (5.3) podal tudi nekaj poslovnih pravil iz sistema obračunavanja.

Na podlagi modela dejstev iz Primera 14 je nastal model poslovnih pravil zapisanih v Primeru 15, ki so se kasneje implementirala v razvojnem programskem okolju RuleStudio. Njihova sintaksa je prikazana v Primeru 16.

Primer 14: Model dejstev za izračun temperaturnega primankljaja

Temperature day
is a day that has properties of temperature.
• Average temperature is property of Temperature day
• Date is property of Temperature day
• Max temperature is property of Temperature day
• Min temperature is property of Temperature day
• Temperature shortage is property of Temperature day
• Temperature day is associated with Date

Primer 15: Omejitvena in izpeljana poslovna pravila temperaturnega primankljaja

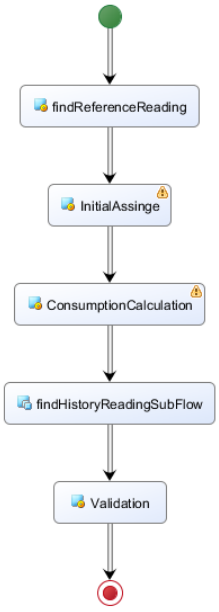
Temperature shortage boundary constant is set to 20.
Temperature shortage must be calculated as **Temperature shortage boundary constant** minus **Average temperature**.
Temperature shortage must be set to 0 only if **Temperature shortage** is less then 0.

Primer 16: Implementirano poslovno pravilo v RuleStudio okolju

```
definitions
    set 'd' to a temperature day ;
then
   set the temperature shortage of d to 20 - the average temperature of d ;

definitions
    set 'd' to a temperature day ;
if
    the temperature shortage of d is less than 0
then
   set the temperature shortage of d to 0 ;
```

Primer 17: Izgled implementirane skripte za analizo odčitkov v RuleStudio



Legenda:

1. FindReferenceReading - Poišči referenčni odčitek;
2. InitialAssigne - Izvedi začetne izpeljave ali priredtivene stavke;
3. ConsumptionCalculation - Izračunaj porabo;
4. FindHistoryReadingSubFlow - Najdi zgodovinsko primerjavo (potek ima dodaten potek);
5. Validation – Omejitvena pravila validacije podatkov.

Skupine teh pravil, ki so delno prikazani v Primeru 16, je potrebno za ustrezno delovanje programske rešitve postaviti v potek dela, ki ga navadno določi arhitekt ali razvijalec programske rešitve. V programski rešitvi analize odčitkov sem se odločil za potek dela, ki je prikazan v Primeru 17. Ostale skupine in primeri poslovnih pravil so prikazani v Prilogi 3.

5.3 Primeri poslovnih pravil pri obračunavanju

Pri projektu obratnega inženirstva in ponovne dokumentacije sem pridelal približno 200 izrazov, 50 poslovnih pravil in 5 odločitvenih tabel. Proces poslovnega modeliranja ni v celoti zaključen, ker je preobsežen in ga bo potrebno zaključiti po manjših zaključenih korakih do ustreznega nivoja podrobnosti. Ugotovil sem tudi, da ni učinkovito celotne programske rešitve implementirati s poslovnimi pravili, ampak je še vedno včasih koristno del reštev napisati v javanski programski kodi. Meja med javanskim jezikom in jezikom pravil ni jasna določena, o tem se odloča arhitekt ali razvijalec.
V nadaljevanju bom predstavil primer poslovnih pravil pri finalizaciji računa. Finalizacija računa je aktivnost pri zaključevanju računa, kjer se račun opremi s potrebnimi podatki, ki se prikažejo na izpisu računa. Eno od pravil finalizacije računa je zapisano v Primeru 18.

V Primeru 18 so najprej definirane spremenljivke, in sicer:

- izpeljan račun i (angl. *invoice*);
- 'datum davka' (angl. *tax date*) je izpeljan iz datuma storitve (angl. *effective service time*). Izpeljava datuma storitve je določena v drugem pravilu.
- 'datum odprte davčne periode' (angl. *open tax period date*) je izpeljan iz vrednosti 'odprta davčna perioda' (ang. *OpenTaxPeriodDate*), ki je izpeljana iz globalnih nastavitev (angl. *global parameter view*).

V preostalem delu Primera 18 so skrita poslovna pravila, ki so tipa logične izpeljave, kot so:

- 'davčni mesec računa' (angl. *tax month*) je mesec od 'datuma davka', v primeru da je davčni mesec večji ali enak 'datumu odprte davčne periode';
- 'davčno leto računa' (angl. *tax year*) je leto od 'datuma davka', v primeru da je davčni mesec večji ali enak 'odprti davčni periodi';
- 'davčni mesec računa' je mesec od 'datum odprte davčne periode', v primeru da je davčni mesec manjši 'datumu odprte davčne periode';
- 'davčno leto računa' je leto od 'datum odprte davčne periode', v primeru da je davčni mesec manjši ali enak 'odprti davčni periodi'.

Eno pravilo v Primeru 18 velja vedno in bi ga lahko postavili kot samostojnega, to je:
'datum pošiljanja računa' (angl. *post date*) je zadnji dan 'davčnega meseca računa'.

Primer 18: Kompleksno poslovno pravilo, ki določa davčno leto in mesec računa

```
Definitions
   set i to an invoice ;
   set 'tax date' to the calendar date of 'effective service time' ;
   set 'open tax period date' to the OpenTaxPeriodDate value of 'global parameter view' ;
if
   'tax date' is before 'open tax period date'
then
   set the tax month of i to the month of 'open tax period date' ;
   set the tax year of i to the year of 'open tax period date' ;
   set the post date of i to last day of the tax month of i in the tax year of i ;
else
   set the tax month of i to the month of 'tax date' ;
   set the tax year of i to the year of 'tax date' ;
   set the post date of i to last day of the tax month of i in the tax year of i ;
```

Drugo izmed pravil, ki jih predstavljam, je pravilo seštevanja in je prikazano v Primeru 19. V definiciji poslovnega pravila v Primeru 19 je zapisano:

- račun je spremenljivka i (angl. *invoice*);
- postavka računa je spremenljivka ii (angl. *invoice item*).

V definiciji Primera 19 je tudi skrito dejstvo, da ima račun več postavk računa. To pomeni, da se za vrednotenje poslovnega pravila pripravi seznam primerkov, ki ima toliko primerkov, kolikor je številčno postavk računa.

Poslovno pravilo iz Primera 19 se glasi:
'skupni znesek računa' (angl. *invoice amount*) je 'znesek računa' sešteto z 'zneskom postavke računa' (angl. *invoice item amount*) odšteto z 'zneskom popusta postavke računa' (angl. *invoice item discount*) prišteto z davkom računa (angl. *tax amount*).

Primer 19: Poslovno pravilo, ki omogoča seštevanje v zanki

```
Definitions
   set i to an invoice ;
   set ii to an invoice item in the items of i ;
then
   set the total amount of i to the total amount of i + the charged amount of ii - the
discounted amount of ii + the tax amount of ii ;
```

Pravilo se izvede za vsak primerek postavke računa. Rezultat je sešteti znesek računa. Na nek način je to programska zanka, ker se pravila izvajajo za vsako instanco objekta po RETE algoritmu.

Da ne bom prikazoval le implementacijskih pravil, podajam še sklop poslovnih pravil iz modelirnega vidika s Primerom 20. V njem je prikazano ovrednotenje cene za določenega naročnika.

Primer 20 lahko razumemo:

- Cena 78434 mora biti ovrednotena v primeru, da velja: Naročnikov produkt ima ponudbo TOPGO in naročnikov produkt ima napravo VODOMER, ki ima veljavno dodeljeno ponudbo na datum računa do.
- Znesek diskriminirane cene 78434 se izbere glede na diskriminator, ki je izpeljan iz velikosti naprave od naročnikovega produkta.
- Cena 78432 mora biti ovrednotena v primeru, da velja: Naročnikov produkt ima ponudbo TOPGO in naročnikov produkt ima napravo SESTAVLJIVI TOPLOTNI ŠTEVEC.
- Cena 78432 mora biti ovrednotena v primeru, da velja: Naročnikov produkt ima ponudbo TOPGO in naročnikov produkt ima napravo MODULARNI TOPLOTNI ŠTEVEC.

Primer 20: Skupek primerov izbire ovrednotenja cene iz modelirnega vidika

Price 78434 must be charged if all of the folowing is true.
Customer (product) has TOPGO product offering.
Customer (product) has WATER_METER resource only if APO has site valid on invoice to timestamp.

Price 78432 must be charged if all of the folowing is true.
Customer has TOPGO product offering.
Customer has COMPACT_HEAT_METER resource.

Price 78432 must be charged if all of the folowing is true.
Customer has TOPGO product offering.
Customer has MODULAR_HEAT_METER resource.

Charge amount discriminator for price 78434 must be agregated resource size.

Zelo uporabno orodje so odločitvene tabele, kjer obstaja en ali več pogojev, ki določajo akcije ali nadaljni tok dogodkov. Primer 21 prikazuje poslovno pravilo v obliki odločitvene tabele, ki določa odločitev, ali naj se plačilni inštrument na položnici prikaže. Primer 21 si lahko razlagamo: Prikaži plačilni inštrument (angl. *payment instrument*) na računu samo v

primeru, da velja oboje: Račun je tipa račun (angl. *debit*) in račun ima trajnik (angl. *direct debit*).

Primer 21: Primer odločitvene tabele prikaza plačilnega inštrumenta

Invoice type	direct debit	Payment instrument
credit		No
debit	Yes	No
	No	Yes

Obstajajo še drugi zanimivi pogledi poslovnih pravil, ki jih sam nisem podajal. Na kakšen način jih bomo izrazili, je odvisno od razvijalca in poslovnega problema. V primeru, da analitik opazi potrebo ali pa ve, da se bodo programske komponente veliko spreminjale, je vpeljava poslovnih pravil smiselna.

SKLEP

Iz lastnih delovnih izkušenj lahko napišem, da je vsekakor zahteve težko zapisati na enostaven in nedvoumen način pri pogovorih in komunikaciji s končnimi uporabniki. Probleme in zahteve je možno kasneje popolnoma strukturirati, vendar se za to porabi veliko časa in vprašanje je, ali bodo uporabniki potrdili zahteve izražene v diagramskih tehnikah ali pa izjave zapisane s poslovnimi pravili.

Od vseh primerov razvojnih metodologij razvoja programskih rešitev menim, da je pristop na podlagi poslovnih pravil zelo blizu idealnim zahtevam, ki bi jih s kombinacijo izdelave primerov uporabe in upravljanjem poslovnih procesov lahko v prihodnosti uporabljali. Ne glede na pristop je lahko dokumentiranje zahtev problematično in velikokrat je sponzorju brez izobrazbe s področja informatike zapis UML dokumentacije z različnimi diagramskimi tehnikami težko razumljiv. Ravno zato menim, da je taktični načrt zapisan v grafični obliki boljša izbira za zapis in predstavitev potreb in načrta, ki prikazuje potek projekta.

Pri projektu obratnega inženirstva in ponovne dokumentacije sistema za obračunavanje sem prišel do zelo koristih ugotovitev. Marsikje sem pri strukturiranju problema opazil, da smo izdelali preveč kompleksno poslovno logiko. Zato sem poslovno logiko z uporabo odločitvenih tabel poenostavil in naredil bolj pregledno. Prednost odločitvenih tabel je v tem, da je poslovna logika napisana na enem mestu, kjer je jasno napisano, katere odločitve podpiramo in katere ne.

Drugo pomembno vprašanje, ki sem ga zasledil in tudi uporabil, so generične rešitve shranjevanje podatkov, ki so v določenih primerih tudi uporabne. Potrebno pa se je zavedati, da bo večji del poslovne logike zapisane v programski kodi, zato potrebujemo pri

generičnih rešitvah dober pregled programske kode. Stroji za izvajanje poslovnih pravil, ki imajo v paketih podano rešitev za pregled poslovnih pravil, kot so spletne aplikacije, so primerno orodje za pregled programske kode oziroma implementirane poslovne logike.

Prednost uporabe orodja RuleXpress je tudi pri predaji poslovnega modela programerjem, ki izvajajo informatizacijo. Dejstvo je, da je poslovni problem strukturiran do takšnega nivoja, da dodatna vprašanja in pojasnila niso potrebna. Pojavi se lahko nekaj težav pri obrnjeni logiki razmišljanja ECA (angl. *event condition action*), saj je programiranju bližje kot pa stavki v naravnem jeziku.

Vsekakor je bila pri poslovnem modeliranju pomembna uspešna izdelava poslovnega slovarja in zajetje poslovne logike, kar vodi v to, da se bom tudi v prihodnosti pri razvoju programske rešitve skušal držati skupnega besedišča in poslovnega slovarja.

Orodje RuleXpress je zelo uporabno orodje za izdelavo in shranjevanje poslovnih pravil v repozitorij. Njegova poglavitna prednost je delovanje v različnih skupinah in dobro usklajevanje sprememb in upravljanje tako izrazov kot dejstev ter poslovnih pravil. Predvsem bi bilo orodje zelo koristo za večja podjetja, kjer bi se kopičilo znanje ljudi iz različnih področij. V okviru praktičnega dela sem orodje uporabljal izključno za potrebe ponovne dokumentacije in repozitorija poslovnih pravil.

S pomočjo repozitorija poslovnih pravil, ki ga omogoča orodje RuleXpress, lahko sedaj hitro pridemo do poslovne logike programske rešitve, kjer se razbere delovanje določenega dela programske rešitve. Pridobljene informacije se zato lahko poda naročniku na licu mesta. Še vedno pa obstaja problem ali bojazen, kadar so poslovna pravila v repozitoriju, niso pa objavljena ali implementirana v delovnem toku, ki nas ob določenem trenutku zanima. Menim, da se bo sčasoma tudi problem prisotnosti poslovnih pravil v repozitoriju, a odsotnost njihove implementacije v repozitoriju, rešil.

Pri podprojektu izdelave programske rešitve analize odčitkov na podlagi zgodovine, kjer sem za poslovno modeliranje uporabljal orodje RuleXpress, je sponzor projekta poslovni model sprejel in potrdil, vendar se moramo zavedati, da izkazovanje zaupanja ne nujno pomeni popolnega razumevanja poslovnega modela s strani sponzorja.

Malo več kritičnih besed lahko namenim sami informatizaciji poslovnih pravil. Veliko težav sem imel pri integraciji poslovnega pravila v razvojno programsko orodje za upravljanje poslovnih pravil, nasprotno pa pri objavi poslovnih pravil v stroj za izvajanje poslovnih pravil nisem imel toliko težav. Zaključim lahko, da je integracija poslovnega modela in avtomatiziran prehod v informatizacijo v tem trenutku nemogoč, se pa pojavljajo nove rešitve tega problema. Velike organizacije kot so OMB in RuleML se trudijo izdelati specifikacije, ki bi združile in integrirale semantično razlikovanje, kjer so na voljo SBRV in RuleML prevajalniki.

Sama implementacija je bila časovno potratna, deloma verjetno tudi zato, ker sem moral spoznati in nastaviti razvojno programsko okolje Eclipse ter izdelati razrede in metode ročno po specifikaciji poslovnega modela.

Razširjenost uporabe pristopa poslovnih pravil je težko ovrednotiti, saj o tem ni na voljo nobenih podatkov. Lahko pa omenim, da sem v tuji literaturi zasledil, da so se s poslovnimi pravili začele ukvarjati velike organizacije in podjetja, kar nakazuje na pomembnost uporabe poslovnih pravil. Vendar pa nisem nikjer zasledil, da bi izrecno izjavili, da uporabljajo pristop poslovnih pravil. V končni fazi ni cilj združenja BRSolutions, ki je pristop poslovnih pravil razvil, promovirati pristop poslovnih pravil, ampak je njihov cilj promovirati načela pristopa poslovnih pravil ter s tem izboljšati poslovanje v podjetji s pomočjo poslovnih pravil, s čimer bi zadržali znanje v podjetju.

Pri razvoju programskih rešitev in uporabi različnih metodologij bi se lahko pridružil ugotovitvam, ki jih je povzel Bajec (2003). V praksi se metodologije ne uporabljajo v celoti kot bi morale, saj se podjetja skušajo držati osnovnih načel različnih metodologij. Težava nastane, ker se v praksi pojavi nekakšen hibrid vseh metodologij in se skušamo tega držati. Seveda je izbira metodologije tudi odvisna od velikosti, tipa projekta, od zaposlenih in od stranke. Zanimiva je tudi opazka, da pristop poslovnih pravil nima čisto jasno določenega postopka in izdelkov, kot ga ima strukturni pristop ali pa objektno orientiran pristop. Ima pa jasno določena načela in v ozadju podpornike svetovnega formata. Tudi sam sem sestavil svoj okvir za lastne potrebe, kateremu se bom v prihodnosti prilagajal. To pa ne pomeni, da je pristop poslovnih pravil pri drugih projektih neuporaben. Odločitev o izbiri pristopa se prepušča poslovnemu analitiku ali pa sponzorju projekta.

V zaključku lahko povem, da mi ni uspelo zavreči nobenega načela ali trditve, ki jih pristop poslovnih pravil izpostavlja. Menim, da so načela osnovana na trdnih temeljih. Moram pa omeniti, da sem imel kar nekaj težav, ker sem moral spremeniti način razmišljanja izdelovanja poslovnih pravil. Na začetku mi je bila bližja programerska definicija poslovnega pravila, kjer velja: pri proženju dogodka pod določenim pogojem se zgodi akcija. Pri poslovnih pravilih pa je v ospredju subjekt v povezavi z dejstvom, ki se ovrednoti pod določenimi pogoji. Prednost se mi zdi v deklarativni definiciji poslovnih pravil in dejstev v naravnem jeziku, za katero menim, da je bolj razumljiva v primerjavi z diagrami.

LITERATURA IN VIRI

1. Agiledata.org. (2006). *Data modeling 101*. Najdeno 18. maja 2010 na spletnem naslovu http://www.agiledata.org/essays/dataModeling101.html
2. Avison, D. E., & Fitzgerald, G. (2003). Where now for development methodologies? *Communications of the ACM*, 46(1), 78–82.
3. Bahill, A. T. (2009). Discovering system requirements. V A.P. Sage & W.B.Rouse (ur.), *Handbook of system engineering and management* (str. 205–266). New York: John Wiley & Sons.
4. Baloh, P., Indihar Štemberger, M., & Vrečar, P. (2002). *Poslovna informatika - dodatno študijsko gradivo, naloge in vodnik po predmetu.* Ljubljana: Ekonomska fakulteta.
5. Blechar, M. J. (2007). *Magic quadrant for business process analysis tools, 2H07-1H08*. Najdeno 5. septembra 2010 na spletnem naslovu http://mediaproducts.gartner.com/reprints/microsoft/vol2/article3/article3.html
6. Business process modeling - Wikipedia. (2010). Najdemo 18 aprila 2010, na spletnem naslovu http://en.wikipedia.org/wiki/Business_modeling
7. Business Rule Group. (2010). *Defining 'business rule'*. Najdeno 25. julija 2010 na spletnem naslovu http://www.businessrulesgroup.org/defnbrg.shtml
8. Centers for medicare & medicaid service. (2008). *Selecting a development approach*. Najdeno 5. julija na spletnem naslovu https://www.cms.gov/SystemLifecycleFramework/Downloads/SelectingDevelopmentApproach.pdf
9. Davenport, T. H., & Short, J. E. (1990). *The new industrial engineering: Information technology and business process redesign.* Cambridge: Sloan Management Review.
10. Demuth, B., Hussmann, H., & Loecher, S. (2001). OCL as a specification language for business rules in database applications. V *«UML» 2001 – The unified modeling language. Modeling languages, concepts, and tools* (str. 104–117). Berlin, Heidelberg: Springer.
11. *Dictionary.com* (b.l.), Proces. Najdeno 5. maja 2010 na spletni strani http://dictionary.reference.com/browse/process
12. *Drools Guvnor - JBoss Community*. Najdeno 5. avgusta 2010 na spletnem naslovu http://www.jboss.org/drools/drools-guvnor.html
13. Fetih, M. (2004). *Razvoj programske rešitve za elektronski števec električne energije s pomočjo jezika za modeliranje UML* (magistrsko delo). Ljubljana: Ekonomska fakulteta.
14. *FOLDOC* (b.l.), Cikel razvoja programske rešitve. Najdeno 5. julija 2010 na spletni strani http://foldoc.org/Systems+Development+Life+Cycle
15. *FOLDOC* (b.l.), Proces razvoja programske rešitve. Najdeno 5. avgusta 2010 na spletni strani http://foldoc.org/software+development+lifecycle

16. Gladys, S. L. (1998). Business knowledge - packaged in a policy charter policy charter as a deliverable. *BRCommunity*. Najdeno 7. avgusta 2010 na spletnem naslovu http://www.brcommunity.com/a385.php
17. Greenfield, J., & Short, K. (2004). *Software factories: assembling applications with patterns, models, frameworks, and tools*. Anaheim: Wiley.
18. Hammer, M. (2003). *The agenda: what every business must do to dominate the decade.* New York: Three Rivers Press.
19. Harrington, H. J. (1991). *Business process improvement: the breakthrough strategy for total quality, productivity, and competitiveness*. New York: McGraw-Hill Professional.
20. He, X., Ma, Z., Shao, W., & Li, G. (2007). A metamodel for the notation of graphical modeling languages. *Proceedings of the 31st Annual international computer software and applications conference* (str. 219–224). B.k.: IEEE Computer Society.
21. Hey, D.C., (2004. 1. januar). Modeling business rules: what data models do. *The Data Administration Newsletter.* Najdeno 2. avgusta 2010 na spletnem naslovu http://www.tdan.com/view-articles/5174/
22. Hooks, I. F., & Farry, K. A. (2001). *Customer-centered products: creating successful products through smart requirements management*. New York: AMACOM Div American Mgmt Assn.
23. IBM. (2009). *IBM WebSphere ILOG JRules V7.0.2 Getting started* (interno gradivo). B.k.: IMB.
24. *iSlovar (b.l.),* Informatizacija. Najdeno 5. septembra 2010 na spletni strani http://www.islovar.org/iskanje_enostavno.asp
25. *iSlovar* (b.l.), Programska rešitev. Najdeno 5. avgusta 2010 na spletni strani http://www.islovar.org/iskanje_enostavno.asp
26. International institute of business analysis. (2006). *A guide to the business analysis body of knowledge*. Najdeno 13. julija 2010 na spletnem naslovu http://www.theiiba.org/AM/Template.cfm?Section=Body_of_Knowledge
27. Jacobson, I. (1999). *The unified software development process*. Reading: Addison-Wesley.
28. Keber, B., Krisper, M., Gornik, T., (2003). *Ogrodje poslovnih procesov ponudnikov storitev*. Ljubljana: Marand d.o.o.
29. Kleppe, A. G., Warmer, J., Warmer, J. B., & Bast, W. (2003). *MDA explained: the model driven architecture: practice and promise*. Boston:Addison-Wesley.
30. Kovačič, A. (2005). *Management poslovnih procesov*. Ljubljana: Založba GV.
31. Kovacic, A. (2004). *Business renovation: business rules (still) the missing link.* Business Process Management Journal 10:158–170.
32. Krisper, M. (2000). *EMRIS Strukturni pristop,* Najdeno 29. marca 2010 na spletni strani http://www2.gov.si/mju/emris.nsf
33. Krstov, L. (2006). *Modeliranje poslovnega pravila v modelih informacijskega sistema* (doktorska dizertacija). Maribor: Ekonomska poslovna fakulteta.

34. Lublinsky, B., & Le Tien, D. (2007, 12. marec). Implementation of business rules and business processes in SOA. *InfoQ*. Najdeno 18. maja 2010 na spletnem naslovu http://www.infoq.com/articles/business-rules-processes
35. Marand d.o.o. (2010). *Arhitektura obračunskega sistema* (interno gradivo). Ljubljana: Marand d.o.o.
36. Martin, J., O'dell, J. (1994). *Object-Oriented Methods*. New York: Prentice Hall PTR
37. Medeot, T. (2007). *Uporaba sodobnih pristopov pri upravljanju poslovnih procesov* (diplomsko delo). Ljubljana: Ekonomska fakulteta.
38. Morgan, T. (2002). *Business rules and information systems: aligning it with business goals*. Boston: Addison-Wesley Professional.
39. Nemuraite, L., Ceponiene, L., & Vedrickas, G. (2009). Representation of business rules in UML&OCL models for developing information systems. V J. Stirna & A. Persson (ur.), *The practice of enterprise modeling* (str. 182–196). B.k.: International Federation for Information Processing.
40. OMG (2009). *OMG available specification, production rule representation - PRR 1.0.* (interno gradivo). Najdeno 25. julija 2010 na spletnem naslovu http://www.omg.org/spec/PRR/1.0/
41. OpenRules. (2010). *Business rules repository*. Najdeno 5. avgusta 2010 na spletnem naslovu http://openrules.com/RulesRepository.htm
42. Penker, M., & Eriksson, H. (2000). *Business modeling with UML: business patterns at work*. New York: Wiley.
43. Ross, R. (2003). *Principles of the business rule approach*. Boston: Addison-Wesley.
44. *SEI Report* (b.l.), Informacijski sistem. Najdeno 5. septembra 2010 na spletni strani http://web.archive.org/web/20070903115947/http://www.sei.cmu.edu/publications/documents/03.reports/03tr002/03tr002glossary.html
45. Software engineering institute. (2006). *CMMI for development, version 1.2*. Najdeno 15. avgusta 2010 na spletnem naslovu http://www.sei.cmu.edu/library/abstracts/reports/06tr008.cfm
46. *SSKJ* (b.l.) Proces. Najdeno 5. maja 2010 na spletni strani http://bos.zrc-sazu.si/cgi/a03.exe?name=sskj_testa&expression=proces&hs=1
47. Standeker, M. (2010). *Obravnava in modeliranje ad-hoc poslovnih procesov* (magistrsko delo). Ljubljana: Fakulteta za računalništvo in informatiko.
48. Tibco (2006). Enhancing BPM with a Business Rule Engine. Najdeno 15. Avgusta 2010 na spletnem naslovu http://www.tibco.com/resources/software/bpm/bpm_rules_wp.pdf
49. Vantheienen, J. (2007). How business rules define business processes. *BRComunity*. Najdeno 5. septembra 2010 na spletnem naslovu http://www.brcommunity.com/b336.php
50. Vidrih, D. (2008). *Modelno vodeni razvoj: Ovrednotenje konceptov in razvoj podpornih orodij* (magistrsko delo). Ljubljana: Fakulteta za računalništvo in informatiko.

51. Wang, M., & Wang, H. (2006). From process logic to business logic - A cognitive approach to business process management. *Information & Management*, 43(2), 179–193.
52. Young, R. R. (2001). *Effective requirements practices*. Boston: Addison-Wesley Professional.
53. Zimmermann, O. (2004, 2. julij). Elements of service-oriented analysis and design. *IBM*. Najdeno 5. septembra na spletnem naslovu http://www.ibm.com/developerworks/library/ws-soad1/

PRILOGE

KAZALO PRILOG

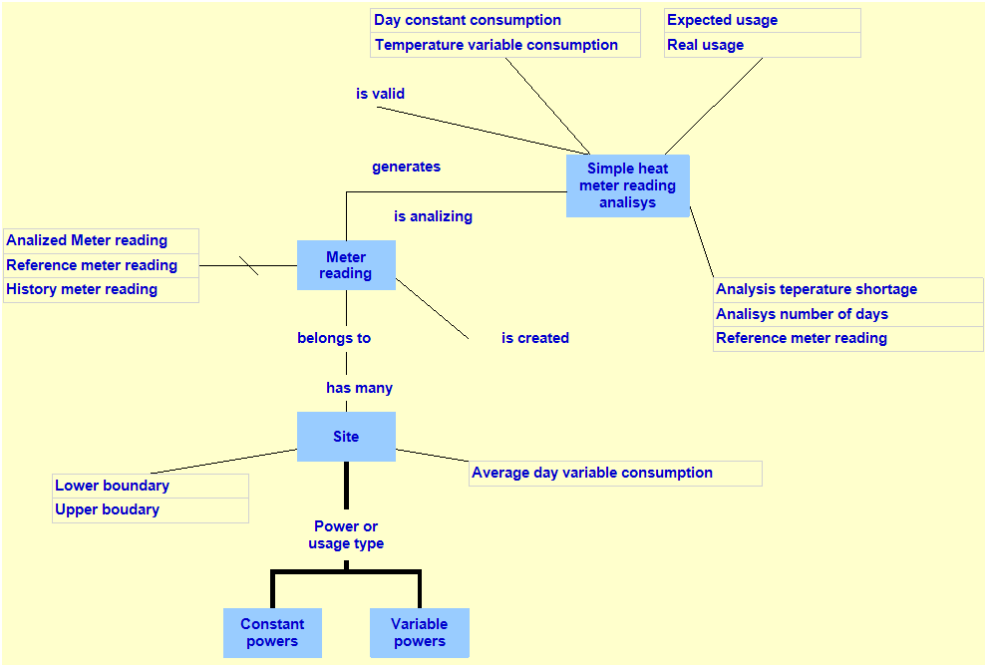
Priloga 1: Seznam uporabljenih kratic..... 1
Priloga 2: Model dejstev programske rešitve za analizo odčitkov..... 2
Priloga 3: Model poslovnih pravil programske rešitve za analizo odčitkov..... 2
Priloga 4: Poročilo informatizirane rešitve za analizo odčitkov..... 16

Priloga 1: Seznam uporabljenih kratic

Okrajšava	Angleški izraz	Slovenski pomen
BOM	Business object modeling	Poslovni slovar v ILOG implementaciji
BPEL	Business process execution languange	Jezik za izvajanje poslovnih procesov
BPM	Business process management	Upravljanje poslovnih procesov
BPMN	Business process modeling notation	Notacija za diagramsko tehniko modeliranje poslovnih procesov
BRM	Business rule management	Upravljanje poslovnih pravil
BRMS	Business rule management system	Sistem za upravljanje poslovnih pravil
CRM	Customer relation management	Upravljanje s strankami
CRUD	Create, read, update, delete	Izdelava, branje, ažuriranje, brisanje
ECA	Event, condition, action	Dogodek, pogoj, akcija
EPC	Event, driven, process, chain	ARIS diagramska tehnika za modeliranje poslovnih procesov
ERP	Enterprise, resource, planning	Celovite programske rešitve
FAB	Fullfilment, assurance, billing	Izpolnjevanje, zagotavljanje, obračunavanje
IDE	Integrated development environment	Itegrirano razvojno programsko okolje
IEEE	Institute of electrical and electronics engineers	Mednarodna neprofitna organzacija za razvoj napredne tehnologijo povezane s elektriko.
IS	Information system	Informacijski sistem
IT	Information technology	Informacijska tehnologija
MDA	Model driven architecture	Modelno voden pristop razvoja programskih rešitev
MDD	Model driven development	Modelno voden razvoj programskih rešitev
OCL	Object constraint language	Omejitveni jezik v okviru UML-ja
OMG	Object modeling group	Združenje, ki se ukvarja s standardizacijo različnih metodologij
PRR	Production rule representation	Specifikacija združenja OMG za standardizacijo poslovnih pravil
RAD	Rapid aplication development	Medtodologija za hiter razvoj programskih rešitev
SCRUM	Ni kratica	Metodologija za razvoj programskih rešitev
SOA	Service oriented architecture	Storitveno usmerjena arhitektura programske rešitve
SQL	Structured query language	Jezik za programiranje v ralacijskih podatkovnih

		bazah
UML	Unified modeling language	Orodje za razvoj programskih rešitev na za objektno orientiran pristop. Uniformni modelirni jezik.
XSL	Extensible stylesheet language	Jezik za transformacijso XML podatkovnih struktur.
XML	Extensible markup language	ML je preprost računalniški jezik podoben HTML-ju, ki nam omogoča format za opisovanje strukturiranih podatkov ali arhitektura za prenos podatkov in njihovo izmenjavo med več omrežji.
XOM	Extended object modeling	Model za definicijo vmesnikov

Priloga 2: Model dejstev programske rešitve za analizo odčitkov



Priloga 3: Model poslovnih pravil programske rešitve za analizo odčitkov

RULEXPRESS™

Title Complete RuleXpress Report

Date 2010-06-17

Member JPE

Shortcut to Report Items

- Rules
- Rule Groups
- Terms
- Facts
- Wordings

- Decision Tables

Community: CRM JPE production

Description

Rules

Analized meter reading	
Statement	Analized reading quantity means reading quantity of analized meter reading.
B. Management	
Quality Grade	10
DCC calculation	
Statement	Day constant consumption must be calculated as real usage divided by Analisys number of days.
B. Management	
Quality Grade	10
Expected constant calculation	
Statement	Constant expected consumption must be calculated as Analysis number of days belongs to Analized Meter reading multiply Day constant consumption belongs to History meter reading
B. Management	
Quality Grade	10
Expected consumption	
Statement	Expected usage must be between Lower boundary and Upper boudary
B. Management	
Quality Grade	10
Expected dynamic calculation	
Statement	Dynamic expected consumption must be calculated as Analysis teperature shortage belong to Analized Meter reading multiply Temperature variable consumption belongs to History meter reading

B. Management	
Quality Grade	10
Expected usage constraint	
Statement	Expected usage can be calculated only if history meter reading has simple heat meter reading analysis
B. Management	
Quality Grade	10
Firs reading status	
Statement	First meter reading on site must have Real meter reading status of instance priklop
B. Management	
Quality Grade	10
Hitrory meter reading	
Statement	History meter reading means next meter reading from one year old analized meter reading.
B. Management	
Quality Grade	10
Lower boundary calculation	
Statement	Lower boundary must be calculated as Real usage multipy Site lower boundary constant.
B. Management	
Quality Grade	10
Minimim temp shortage	
Statement	Temeprature shortage must not be negative it is set to 0.
B. Management	
Quality	10

Grade	
MR constraint	
Statement	Meter reading must have meter reading status
B. Management	
Quality Grade	10
Real usage calculation	
Statement	Real usage must be calculated as Analized reading quantity minus Reference reading quantity
B. Management	
Quality Grade	10
Reference reading quantity	
Statement	Reference reading quantity means reading quantity of reference meter reading
B. Management	
Quality Grade	10
Rule 10021	
Statement	Reference meter reading must be at least 5 days older than analized meter reading
B. Management	
Quality Grade	10
set meter reading status	
Statement	Meter reading analysis that is not valid must set meter reading status to in verification
B. Management	
Quality Grade	10
simple reading analysis	

Statement	Meter reading must have Simple heat meter reading analysis only if meter reading status is of category Real meter reading status and not instance of priklop Meter reading may generate Simple heat meter reading analysis only if meter reading status is of category Real meter reading status and not instance of priklop
B. Management	
Quality Grade	10
Temperature shortage calculation	
Statement	Temeprature shortage must be calculated as Temperature shortage boundary constant minus Average temeprature.
B. Management	
Quality Grade	10
TVC calculation	
Statement	Temperature variable consumption must be calculated as real usage divided by Analisys teperature shortage
B. Management	
Quality Grade	10
Upper boundary calculation	
Statement	Upper boudary must be calculated as Real usage multiply Site upper boundary constant
B. Management	
Quality Grade	10

Rule Groups

Expected consumption		
B. Management		
Quality Grade	10	
Uses Rule(s)		
Expected consumption		

Expected dynamic calculation		
Expected constant calculation		
Meter reading calculation		
B. Management		
Quality Grade	10	
Uses Rule(s)		
Upper boundary calculation		
Lower boundary calculation		
DCC calculation		
TVC calculation		
Real usage calculation		
Hitrory meter reading		
Reading analisys constraint		
B. Management		
Quality Grade	10	
Uses Rule(s)		
simple reading analysis		
Expected usage constraint		
Rule 10021		
Temperature shortage		
B. Management		
Quality Grade	10	
Uses Rule(s)		
Minimim temp shortage		
Temperature shortage calculation		

Terms

Term	Definition
Analisys number of days	
Analized Meter reading	
Analized reading quantity	
Analysis number of days	
Analysis teperature shortage	
Average day variable consumption	
Average temeprature	Average temperature of day 10

Billing code	
Calculated meter reading status	
Catalog element	
Constant expected consumption	
Constant powers	
Corrected volume	
Day	Is 24 hours long timeline
Day constant consumption	
Dynamic expected consumption	
Energy	
Entity catalog type	
Expected usage	
Gas	
Gas meter reading	
Global parameters	
Heat Heatmeter	
Heat meter reading	
Heat Watermete	
History meter reading	
Izklop	
Javljen	
Lower boundary	
Max temeprature	Maksimal temperature of day 10
Meter reading (Reading)	
Meter reading status	
Meter reading type	
Min temeprature	Minimal temeprature od day. 10
Odcitovalec	
Power or usage type	
Priklop	
Reading quantity	
Reading timestamp	
Real meter reading status	
Real usage	Is usage between two meter readings that are real and not calculated. 10
Reference meter reading	
Reference reading quantity	
Selitev	
Simple heat meter reading analisys	
Site	
Site lower boundary constant	constat is 0,7.
Site upper boundary constant	constant is 1,2
Skladisce	
Temeprature shortage	Is calculated average temeprature. 10
Temperature shortage boundary	

constant	

Facts

Site has many Meter reading Meter reading belongs to Site		
B. Management		
Quality Grade	10	
Meter reading generates Simple heat meter reading analisys		
B. Management		
Quality Grade	10	
Analisys number of days is property of Simple heat meter reading analisys Simple heat meter reading analisys has Analisys number of days		
B. Management		
Quality Grade	10	
Analysis number of days is property of Simple heat meter reading analysis Simple heat meter reading analysis has Analysis number of days		
B. Management		
Quality Grade	10	
Analysis teperature shortage is property of Simple heat meter reading analisys Simple heat meter reading analisys has Analysis teperature shortage		
B. Management		
Quality Grade	10	
Analysis teperature shortage is property of Simple heat meter reading analysis Simple heat meter reading analysis has Analysis teperature shortage		
B. Management		
Quality Grade	10	
Average day variable consumption is property of Site Site has Average day variable consumption		
B. Management		
Quality Grade	10	
Average temeprature is property of Day Day has Average temeprature		

B. Management		
Quality Grade	10	
Corrected volume is property of Gas meter reading Gas meter reading has Corrected volume		
B. Management		
Quality Grade	10	
Day constant consumption is property of Simple heat meter reading analisys Simple heat meter reading analisys has Day constant consumption		
B. Management		
Quality Grade	10	
Day constant consumption is property of Simple heat meter reading analysis Simple heat meter reading analysis has Day constant consumption		
B. Management		
Quality Grade	10	
Energy is property of Heat meter reading Heat meter reading has Energy		
B. Management		
Quality Grade	10	
Expected usage is property of Simple heat meter reading analisys Simple heat meter reading analisys has Expected usage		
B. Management		
Quality Grade	10	
Expected usage is property of Simple heat meter reading analysis Simple heat meter reading analysis has Expected usage		
B. Management		
Quality Grade	10	
Lower boundary is property of Site Site has Lower boundary		
B. Management		
Quality Grade	10	
Max temeprature is property of Day Day has Max temeprature		

B. Management		
Quality Grade	10	

Min temeprature is property of Day Day has Min temeprature

B. Management		
Quality Grade	10	

Reading quantity is property of Meter reading Meter reading has Reading quantity

B. Management		
Quality Grade	10	

Reading timestamp is property of Meter reading Meter reading has Reading timestamp

B. Management		
Quality Grade	10	

Real usage is property of Simple heat meter reading analisys Simple heat meter reading analisys has Real usage

B. Management		
Quality Grade	10	

Real usage is property of Simple heat meter reading analysis Simple heat meter reading analysis has Real usage

B. Management		
Quality Grade	10	

Reference meter reading is property of Simple heat meter reading analisys Simple heat meter reading analisys has Reference meter reading

B. Management		
Quality Grade	10	

Reference meter reading is property of Simple heat meter reading analysis Simple heat meter reading analysis has Reference meter reading

B. Management		
Quality Grade	10	

Temeprature shortage is property of Day Day has Temeprature shortage

B. Management		
Quality Grade	10	
Temperature variable consumption is property of Simple heat meter reading analisys Simple heat meter reading analisys has Temperature variable consumption		
B. Management		
Quality Grade	10	
Temperature variable consumption is property of Simple heat meter reading analysis Simple heat meter reading analysis has Temperature variable consumption		
B. Management		
Quality Grade	10	
Upper boudary is property of Site Site has Upper boudary		
B. Management		
Quality Grade	10	
Volume 1 is property of Heat meter reading Heat meter reading has Volume 1		
B. Management		
Quality Grade	10	
Volume 2 is property of Heat meter reading Heat meter reading has Volume 2		
B. Management		
Quality Grade	10	
Volume is property of Gas meter reading Gas meter reading has Volume		
B. Management		
Quality Grade	10	
Entity catalog type is associated with Catalog element		
B. Management		
Quality Grade	10	
Meter reading is created		
B. Management		
Quality Grade	10	

Simple heat meter reading analisys is valid		
B. Management		
Quality Grade	10	
Simple heat meter reading analysis is valid		
B. Management		
Quality Grade	10	
Meter reading status is analizing Meter reading Meter reading must have Meter reading status		
B. Management		
Quality Grade	10	
Simple heat meter reading analysis is analizing Meter reading Meter reading must have Simple heat meter reading analysis		
B. Management		
Quality Grade	10	
Meter reading should have Simple heat meter reading analisys		
B. Management		
Quality Grade	10	
Calculated meter reading status is categorized as Calculated meter reading status Calculated meter reading status is a category of Meter reading status		
B. Management		
Quality Grade	10	
Meter reading is categorized as Gas meter reading Gas meter reading is a category of Meter reading		
B. Management		
Quality Grade	10	
Meter reading is categorized as Heat meter reading Heat meter reading is a category of Meter reading		
B. Management		
Quality Grade	10	
Meter reading status is categorized as Real meter reading status Real meter reading status is a category of Meter reading status		

B. Management		
Quality Grade	10	
Site is categorized as Constant powers Constant powers is a category of Site		
B. Management		
Quality Grade	10	
Site is categorized as Gas Gas is a category of Site		
B. Management		
Quality Grade	10	
Site is categorized as Heat Heatmeter Heat Heatmeter is a category of Site		
B. Management		
Quality Grade	10	
Site is categorized as Heat Watermete Heat Watermete is a category of Site		
B. Management		
Quality Grade	10	
Site is categorized as Variable powers Variable powers is a category of Site		
B. Management		
Quality Grade	10	
Global parameters has the instance Site lower boundary constant Site lower boundary constant is an instance of Global parameters		
B. Management		
Quality Grade	10	
Global parameters has the instance Site upper boundary constant Site upper boundary constant is an instance of Global parameters		
B. Management		
Quality Grade	10	
Global parameters has the instance Temperature shortage boundary constant Temperature shortage boundary constant is an instance of Global parameters		

B. Management		
Quality Grade	10	
Meter reading has the instance Analized Meter reading Analized Meter reading is an instance of Meter reading		
B. Management		
Quality Grade	10	
Meter reading has the instance Analized reading quantity Analized reading quantity is an instance of Meter reading		
B. Management		
Quality Grade	10	
Meter reading has the instance History meter reading History meter reading is an instance of Meter reading		
B. Management		
Quality Grade	10	
Meter reading has the instance Reference meter reading Reference meter reading is an instance of Meter reading		
B. Management		
Quality Grade	10	
Meter reading has the instance Reference reading quantity Reference reading quantity is an instance of Meter reading		
B. Management		
Quality Grade	10	
Real meter reading status has the instance Izklop Izklop is an instance of Real meter reading status		
B. Management		
Quality Grade	10	
Real meter reading status has the instance Javljen Javljen is an instance of Real meter reading status		
B. Management		
Quality Grade	10	
Real meter reading status has the instance Odcitovalec Odcitovalec is an instance of Real meter reading status		

B. Management		
Quality Grade	10	
Real meter reading status has the instance Priklop Priklop is an instance of Real meter reading status		
B. Management		
Quality Grade	10	
Real meter reading status has the instance Selitev Selitev is an instance of Real meter reading status		
B. Management		
Quality Grade	10	
Real meter reading status has the instance Skladisce Skladisce is an instance of Real meter reading status		
B. Management		
Quality Grade	10	

Priloga 4: Poročilo informatizirane rešitve za analizo odčitkov

Business Rule Report

Generated on 30.6.2010 9:31

Parameters

Project: simpleHistoryBaseAlgorithm

name	readingTimestamp
type	java.util.Date
verbalization	the reading timestamp
direction	IN
defaultValue	

Rule Artifacts

Project: simpleHistoryBaseAlgorithm

Package: consumptionCalculation

Name: calculateSpecificConsumption

Documentation:

Definition:

definitions

set 'shbmr' to a simple history base analysis ;

then

set the day constant consumption of **shbmr** to the real usage of **shbmr** / the analysis number of days of **shbmr** ;

set the temperature variable consumption of **shbmr** to the real usage of **shbmr** / the analysis temperature shortage of **shbmr** ;

Properties:

type	model.brl.ActionRule
locale	en_US

Package: default

Name: ilrmain

Documentation:

Definition:

```
System.out.println("Start");
averageTemperature.TemperatureDay  oneDay = new
averageTemperature.TemperatureDay(common.DateUtil.makeDate(2009,3,20),17
);
System.out.println(oneDay);

view.SiteReadingView srview = new view.TestSiteReadingViewImpl();
//System.out.println(srview.findMeterReadings("55"));

view.TemperatureDayView tdview = new view.TemperatureDayViewImpl();
System.out.println(tdview.temperatureDayDatas);


tdview.registerTemperatureDay(DateUtil.makeDate(2009,6,20),4);
System.out.println(tdview.temperatureDayDatas);
execute();
```

Properties:

type	model.rule.Function

Package: expectedConsumptionValidation

Name: calculateExpectedUsage

Documentation:

Definition:

definitions

set 'shbmr' to a simple history base analysis ;

set 'hmra' to a simple history base analysis ;

then

set the expected usage of **shbmr** to the analysis number of days of **shbmr** * the day constant consumption of **hmra** + the analysis temperature shortage of **shbmr** * the temperature variable consumption of **hmra** ;

Properties:
type model.brl.ActionRule
locale en_US

Name: validateExpectedConsumption
Documentation:
Definition:
definitions
set 'shbmra' to a simple history base analysis ;
set 'ru' to the real usage of **shbmra ;**
set 'eu' to the expected usage of **shbmra ;**
if
eu is between (**ru** * 0.7) and (**ru** * 1.3)
then
make it **shbmra** is valid reading that **shbmra** is valid reading ;
Properties:
type model.brl.ActionRule
locale en_US

Package: findReferenceReading

Name: findReferenceReading
Documentation:
Definition:
Properties:
type model.brl.ActionRule
locale en_US

Package: initialCalculationSHBRA

Name: assigeAnalysisTempShortage
Documentation:
Definition:
definitions
set 'shbra' to a simple history base analysis ;
set 'amrd' to the analized reading of **shbra ;**
set 'rmrd' to the reference meter reading of **shbra ;**
set 'analizeDay' to a temperature day
where the day of **this temperature day** is after or the same as the reading timestamp of **amrd ;**
then
set the analysis temperature shortage of **shbra** to **analizeDay** .calculateTemperatureShortage(the reading timestamp of **rmrd**) ;
Properties:
type model.brl.ActionRule

locale en_US

Name: assigneNumberOfDays
Documentation:
Definition:
definitions
set 'shbra' to a simple history base analysis ;
set 'amr' to a meter reading from the analized reading of **shbra ;**
set 'rmr' to a meter reading from the reference meter reading of **shbra ;**
then
set the analysis number of days of **shbra** to **amr** .diference(**rmr**) ;
Properties:
type model.brl.ActionRule
locale en_US

Name: assigneRealUsage
Documentation:
Definition:
definitions
set 'mr' to a meter reading ;
set 'shbra' to a simple history base analysis ;
set 'rmr' to a meter reading from the reference meter reading of **shbra ;**
then
set the real usage of **shbra** to the reading quantity of **mr** - the reading quantity of **rmr** ;
Properties:
type model.brl.ActionRule
locale en_US

Package: temperatureShortage.temperatureShortageCalculation

Name: assigneTemepratureShortage
Documentation:
Definition:
definitions
set 'd' to a temperature day ;
then
set the temperature shortage of **d** to 20 - the average temperature of **d** ;
Properties:
type model.brl.ActionRule
locale en_US

Name: temperatureShortageView
Documentation:
Definition:

Properties:
type model.brl.ActionRule
locale en_US

Package: temperatureShortage.temperatureShortageModification

Name: assigneNegativeTemperatureShortage
Documentation:
Definition:
```
definitions
  set 'd' to a temperature day ;
if
  the temperature shortage of d is less than 0
then
  set the temperature shortage of d to 0 ;
```
Properties:
type model.brl.ActionRule
locale en_US

Ruleflows

Project: simpleHistoryBaseAlgorithm

Package: default

Name: mainFlow
Documentation:
Definition:

Tasks:
Properties:
type model.ruleflow.RuleFlow
locale en_US

Package: findHistoryAnalysis

Name: findHistoryReadingSubFlow
Documentation:
Definition:

Tasks:
Properties:
type model.ruleflow.RuleFlow

locale en_US

Package: temperatureShortage

Name: ShortageFlow
Documentation:
Definition:

Tasks:
Properties:
type model.ruleflow.RuleFlow
locale en_US
mainFlowTask true

Packages

Project: simpleHistoryBaseAlgorithm

Package: default

Name: consumptionCalculation
Documentation:
Properties:
type model.base.RulePackage

Name: exception
Documentation:
Properties:
type model.base.RulePackage

Name: expectedConsumptionValidation
Documentation:
Properties:
type model.base.RulePackage

Name: findHistoryAnalysis
Documentation:
Properties:
type model.base.RulePackage

Name: findReferenceReading
Documentation:
Properties:

type model.base.RulePackage

Name: initialCalculationSHBRA
Documentation:
Properties:
type model.base.RulePackage

Name: temperatureShortage
Documentation:
Properties:
type model.base.RulePackage

Package: findHistoryAnalysis

Name: findFirstHistoryAnalysisFromDate
Documentation:
Properties:
type model.base.RulePackage

Name: findHistoryYearAnalysis
Documentation:
Properties:
type model.base.RulePackage

Package: temperatureShortage

Name: temperatureShortageCalculation
Documentation:
Properties:
type model.base.RulePackage

Name: temperatureShortageModification
Documentation:
Properties:
type model.base.RulePackage