UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Sašo Thorževskij

VKLJUČITEV POSLOVNIH UPORABNIKOV V ŽIVLJENJSKI CIKEL POSLOVNIH PRAVIL

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

Ljubljana, 2010

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01625/2009
Datum: 15.12.2009

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **SAŠO THORŽEVSKIJ**

Naslov: **VKLJUČITEV POSLOVNIH UPORABNIKOV V ŽIVLJENJSKI CIKEL POSLOVNIH PRAVIL**

INVOLVING BUSINESS USERS IN BUSINESS RULES LIFECYCLE

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Nenehne spremembe v poslovanju silijo organizacije v pogoste spremembe informacijskih sistemov. Eden izmed pristopov, ki olajša njihovo pogosto spreminjanje, temelji na razdvojitvi poslovnih pravil od informacijskega sistema.

V diplomskem delu preučite možnosti za vključitev poslovnih uporabnikov v življenjski cikel poslovnih pravil z uporabo izbranega sistema za upravljanje poslovnih pravil, pri čemer mora biti poslovnim uporabnikom poleg samega dela s poslovnimi pravili omogočeno tudi njihovo testiranje in postavitev na izvajalno okolje.

Mentor:

prof. dr. Marko Bajec

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Franc Solina

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Sašo Thorževskij

VKLJUČITEV POSLOVNIH UPORABNIKOV V ŽIVLJENJSKI CIKEL POSLOVNIH PRAVIL

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

Mentor:
izr. prof. dr. Marko Bajec

Ljubljana, 2010

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani **Sašo Thorževskij**,

z vpisno številko **63000301**,

sem avtor diplomskega dela z naslovom:

Vključitev poslovnih uporabnikov v življenjski cikel poslovnih pravil

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom **izr. prof. dr. Marka Bajca**;
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela;
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 14. 6. 2010 Podpis avtorja:

Zahvala

„*Tiha hvaležnost nikomur ne koristi*.“
~ G. B. Stern ~

Izr. prof. dr. Marko Bajec;
Mateja, Neva, Sergej;
Borut, Matic, Nina, Robert, Tjaša;
Anja, Uroš.

Hvala vsem!

Za Tita in Matejo.

Kazalo

POVZETEK 1
ABSTRACT 2
UVOD 3
1 POSLOVNA PRAVILA 5
1. 1 Kaj so poslovna pravila? 5
1. 1. 1 Kaj poslovna pravila niso? 5
1. 1. 2 Evolucija definicije poslovnih pravil 6
1. 1. 3 Definicija 8
1. 2 Klasifikacija poslovnih pravil 8
1. 2. 1 Definicije 9
1. 2. 2 Omejitve 10
1. 2. 3 Izpeljave 10
2 SISTEMI ZA UPRAVLJANJE POSLOVNIH PRAVIL 11
2. 1 Upravljanje poslovnih pravil 11
2. 1. 1 Življenjski cikel poslovnih pravil 13
2. 2 Kaj so sistemi za upravljanje poslovnih pravil 14
2. 3 Komponente 15
2. 3. 1 Domensko specifični jeziki 16
2. 4 Prednosti 17
3 TESTIRANJE POSLOVNIH PRAVIL 19
3. 1 Splošno o testiranju 19
3. 1. 1 Kategorije in metode testiranja 21
3. 1. 1. 1 Metode bele skrinjice 23
3. 1. 1. 2 Metode črne skrinjice 23
3. 1. 2 Postopek testiranja 23
3. 2 Delta testiranje 25
3. 2. 1 Ogrodje 26
3. 2. 2 Kvalitativno delta testiranje 27
3. 2. 3 Kvantitativno delta testiranje 28
3. 2. 4 Prednosti 28
4 UPORABLJANI PRODUKTI IN TEHNOLOGIJE 30
4. 1 IBM WebSphere ILOG JRules 30
4. 1. 1 Moduli 30
4. 1. 1. 1 JRules 31
4. 1. 1. 2 Rule Team Server 32
4. 1. 1. 3 Decision Validation Services 32
4. 1. 1. 4 Rule Solutions for Office 32
4. 1. 2 Koncepti 32
4. 1. 2. 1 Izvajalni objektni model 33
4. 1. 2. 2 Poslovni objektni model 33
4. 1. 2. 3 Projekti poslovnih pravil 34
4. 1. 2. 4 Poslovna in tehnična pravila 35
4. 1. 2. 5 Množice poslovnih pravil 36
4. 1. 2. 6 Aplikacije poslovnih pravil 37

4. 2 N-nivojska arhitektura 38
4. 3 Ogrodje Spring 40
4. 4 JUnit in Unitils 42
5 OBRAVNAVANI PROBLEM 44
5. 1 Obstoječi in želeni postopek upravljanja poslovnih pravil 44
5. 2 Orodja za delo s poslovnimi pravili za poslovne uporabnike 46
5. 2. 1 Rule Team Server 46
5. 2. 1. 1 Funkcionalnosti 48
5. 2. 1. 2 Pravice uporabnikov in varnost 51
5. 2. 2 Decision Validation Services 52
5. 2. 2. 1 Scenario Service Provider 53
5. 2. 2. 2 Decision Warehouse 53
5. 2. 2. 3 Osnovni pojmi 53
5. 2. 2. 4 Možnosti podajanja vhodnih podatkov 57
5. 2. 2. 5 Postopek testiranja 58
5. 2. 2. 6 Delta testiranje 58
5. 2. 2. 7 Vzpostavitev testnega okolja 59
5. 3 Alternativna možnost 60
6 VPELJAVA ORODIJ 62
6. 1 Rule Team Server 62
6. 2 Decision Validation Services 62
6. 2. 1 Dostop do podatkov 63
6. 2. 2 Postavitev testnega izvajalnega okolja 65
ZAKLJUČEK 66
LITERATURA IN VIRI 68
SEZNAM SLIK IN TABEL 70
Seznam slik 70
Seznam tabel 71

Seznam uporabljenih kratic in simbolov

API	Application Programming Interface – aplikacijski programski vmesnik
BAL	Business Action Language – domensko specifični jezik za pisanje poslovnih pravil
BOM	Business Object Model – poslovni objektni model
BRMS	Business Rules Management System – sistem za upravljanje poslovnih pravili
DAO	Data Access Object – načrtovalski vzorec za potrebe dostopa do podatkov
DSL	Domain Specific Language – domensko specifični jezik
DTO	Data Transfer Object – načrtovalski vzorec za potrebe prenosa podatkov
DVS	Decision Validation Services – modul v ILOG-u, ki omogoča testiranje poslovnih pravil
EAR	Enterprise Archive Resource – datotečni format, ki se uporablja za pakiranje javanskih strežniških aplikacij
EJB	Enterprise JavaBeans – komponenta J2EE arhitekture, ki omogoča modularno zgradbo strežniških aplikacij
ILOG	Blagovna znamka podjetja IBM
IRL	ILOG Rule Language – namenski jezik za pisanje tehničnih poslovnih pravil
KPI	Key Performance Indicators – ključni kazalniki uspeha
ORM	Object Relational Mapping – programska tehnika, ki omogoča preslikavo podatkov v relacijskih tabelah v enakovreden objektni model
RES	Rule Execution Server – ILOG-ovo izvajalno okolje za poslovna pravila
RTS	Rule Team Server – modul v ILOG-u, ki omogoča delo s poslovnimi pravili poslovnim uporabnikom
SSP	Scenario Service Provider – izvajalno okolje za testiranje v ILOG-u
SUPB	Sistem za upravljanje s podatkovno bazo
SUPP	Sistem za upravljanje poslovnih pravil
XML	eXtensible Markup Language – format za opisovanje strukturiranih podatkov
XOM	eXecution Object Model – izvajalni objektni model

Povzetek

Organizacije delujejo v izredno dinamičnih okoljih, ki terjajo nenehne spremembe v poslovanju. Slednje vključujejo tudi spremembe na področju poslovne informatike, ki predstavlja osnovo za kakovostno in učinkovito delovanje organizacije. Zaradi potrebe po nenehnem spreminjanju in prilagajanju informacijskih sistemov se je razvil nov pristop njihove zasnove in izgradnje, ki temelji na razdvojitvi poslovnih pravil od preostalega informacijskega sistema. V tem primeru je skrb za izvajanje poslovnih pravil največkrat prepuščena sistemom za upravljanje s poslovnimi pravili. Ti poleg izvajalnega okolja vsebujejo tudi orodja za razvoj in upravljanje poslovnih pravil.

V obstoječih aplikacijah poslovnih pravil je skrb za pravila med njihovim življenjskim ciklom v celoti na ramenih razvijalcev. V prihodnosti želimo skrb za poslovna pravila deloma prenesti tudi na poslovne uporabnike in s tem skrajšati čas, potreben za vpeljavo novih različic poslovnih pravil v izvajanje.

V diplomskem delu se podrobneje seznanimo s poslovnimi pravili in spoznamo osnovne koncepte sistemov za upravljanje s poslovnimi pravili, ki zagotavljajo ustrezna orodja za podporo upravljanju. Dodatno je predstavljen postopek testiranja poslovnih pravil, ki se zaradi njihove spremenljive narave razlikuje od običajnega postopka testiranja. V ta namen spoznamo postopek delta testiranja, ki nam olajša testiranje neprestanih sprememb v pravilih.

Nato na izbranem sistemu za upravljanje poslovnih pravil, to je IBM WebSphere ILOG JRules, predstavimo možnosti za neposredno vključitev poslovnih uporabnikov v delo s poslovnimi pravili in s tem v njihov življenjski cikel. Izbrana orodja podrobneje predstavimo in se med pregledom funkcionalnosti prepričamo, ali ustrezajo našim potrebam.

Za konec se osredotočimo na posledice, ki jih ima vpeljava predstavljenih orodij za obstoječe aplikacije poslovnih pravil in predstavimo možnosti za njihovo rešitev, s čimer se omogoči uporaba predstavljenih orodij in poslovnim uporabnikom zagotovi vključenost v celotni življenjski cikel poslovnih pravil.

Ključne besede

poslovna pravila, sistem za upravljanje poslovnih pravil, življenjski cikel poslovnih pravil, testiranje poslovnih pravil, ILOG, delta testiranje

Abstract

Organizations operate in dynamic environments, which require continuous modifications of business policies. The latter also implies changes in business informatics, the basis for effective and prosperous operation. To answer to the constant need for modifications and adaptations, a new approach to information systems design and implementation has been developed, based on separating business rules from the rest of the information system. In such solutions, business rule execution is entrusted to special business rule management systems, which incorporate development and management tools, including the execution environment.

In existing business rules applications, developers are responsible for maintenance of the business rule logic throughout the system lifecycle. In the future we would like to encourage business users to maintain their rules by themselves, and thereby shorten the time needed to deploy new business rules into execution.

The Thesis focuses on business rules, introduction of the basic concepts of business rule management systems, and providing appropriate tools for business support. An alternative method for testing business rules, which differs from the standard testing procedure due to their changing nature, is also presented. This delta testing technique simplifies the testing of continuous change in business rules.

The IBM WebSphere ILOG JRules business rule management system is then used to introduce the possibilities of direct involvement of business users into the process of applying and managing business rules and thus into the lifecycle of business rules. The functionalities of the selected tools, which are presented in detail, are reviewed in order to assess if they meet our demands.

Finally, we review the consequences of the usage of the featured tools in existing business rules applications. We present the possibilities to remedy these consequences, and thereby enable the integration of business users into the entire lifecycle of business rules.

Key words
business rules, business rule management systems, business rules lifecycle, business rules testing, ILOG, delta testing

Uvod

„Ne preživijo najmočnejši, niti najbolj inteligentni, temveč tisti, ki so najbolj dovzetni za spremembe.”

L. C. Megginson

Zgornji citat, ki ga pogosto v zmoti pripisujejo Charlesu Darwinu, se ne navezuje le na živalske in rastlinske vrste, temveč ga v zadnjih letih pogosto uporabljajo tudi v poslovnem okolju. Organizacije namreč delujejo v izredno dinamičnih okoljih, v katerih so izpostavljene nenehnim poslovnim pritiskom. Če želijo ostati konkurenčne, se morajo na pritiske pravočasno odzvati z ustreznimi spremembami v svojem poslovanju, ki med drugim zadevajo tudi spremembe na področju poslovne informatike. Ta namreč predstavlja podlago za kakovostno in učinkovito delovanje organizacije. Sprememba poslovanja v večini primerov pomeni tudi spremembo obstoječega informacijskega sistema ter razvoj dodatnih aplikacij za podporo spremembam poslovanja.

Nenehne spremembe poslovanja tako silijo raziskovalce in praktike k iskanju novih pristopov, ki bi olajšali nenehno spreminjanje in prilagajanje informacijskih sistemov. Eden izmed pristopov, ki olajša spremembe informacijskih sistemov, temelji na razdvojitvi poslovnih pravil od informacijskega sistema ter njihovi samostojni obravnavi. Podrobneje ga bomo spoznali v nadaljevanju.

Omenjeni pristop s pridom uporabljamo tudi v podjetju Optilab, d. o. o., pri razvoju programskih rešitev za zunanje naročnike in tržišče. Naše rešitve temeljijo na platformi J2EE in večnivojski arhitekturi, za razvoj in izvajanje aplikacij poslovnih pravil pa uporabljamo WebSphere ILOG JRules, sistem za upravljanje s poslovnimi pravili podjetja IBM.

Do sedaj se je ILOG izkazal za odličnega, vendar smo uporabljali le omejen nabor njegovih funkcionalnosti: razvojna orodja za razvoj aplikacij poslovnih pravil in izvajalno okolje oz. stroj za izvajanje poslovnih pravil za potrebe izvajanja aplikacij poslovnih pravil. Za začetni razvoj in kasnejše upravljanje poslovnih pravil, to je urejanje in brisanje obstoječih poslovnih pravil ter dodajanje novih poslovnih pravil, so skrbeli razvijalci, sistemski arhitekti pa so skrbeli za integracijo aplikacij poslovnih pravil v celovite rešitve. Poslovni uporabniki tako niso imeli neposrednega vpogleda v obstoječa poslovna pravila, prav tako niso imeli možnosti njihovega neposrednega urejanja in so se morali v primeru kakršnihkoli sprememb poslovnih pravil obrniti na razvijalce, ki so potrebne spremembe implementirali.

V prihodnje želimo postopek skrajšati in skrb za poslovna pravila deloma prenesti tudi na poslovne uporabnike. Poleg tega, da bi poslovnim uporabnikom omogočili dodajanje, urejanje in brisanje poslovnih pravil – kar je ena izmed prednosti pristopa, ki temelji na ločitvi poslovnih pravil od preostalega informacijskega sistema – jim želimo omogočiti tudi neposredno dodajanje novih različic aplikacij poslovnih pravil v izvajanje na izvajalno okolje.

Da bi jim to lahko omogočili, moramo zagotoviti ustrezna orodja za preverjanje pravilnosti dodanih ali popravljenih poslovnih pravil in preverjanje njihovega vpliva na ostala poslovna pravila in celotno aplikacijo poslovnih pravil.

V diplomskem delu bomo tako spoznali, kako lahko poslovnim uporabnikom omogočimo delo s poslovnimi pravili in zagotovimo testiranje aplikacij teh pravil ter kako oboje vpliva na obstoječe programske rešitve. Preden se lotimo obravnavane tematike, si bomo za lažje razumevanje najprej pogledali nekaj osnovnih dejstev o poslovnih pravilih in kako jih kategoriziramo. Nadalje se bomo osredotočili na sisteme za upravljanje poslovnih pravil, v sklopu katerih bomo obdelali tudi postopek upravljanja teh pravil, ter testiranje poslovnih pravil.

1 Poslovna pravila

„Hitrost sprememb se v prihodnosti ne bo zmanjšala. V večini panog se bo v naslednjih desetletjih konkurenca kvečjemu pohitrila."

John. P. Kotter

Nenehne spremembe v poslovanju organizacij, v želji da ostanejo konkurenčne, terjajo tudi nenehne spremembe poslovnih okolij in posledično pravil, na katerih temelji delovanje organizacij. V zadnjem času so se začela poslovna pravila v organizacijah spreminjati tako hitro, da jim razvoj in posodabljanje informacijskih sistemov, v katerih so ta pravila zajeta, enostavno več ne moreta slediti. Zaradi hitrih sprememb poslovnih pravil je treba zagotoviti tako zasnovo informacijskih sistemov, ki bo omogočala njihov neprestani razvoj in spremembe, skladno s spremembami organizacijskega okolja.

V ta namen se je razvil nov pristop zasnove in izgradnje informacijskih sistemov, ki omogoča ločitev poslovnih pravil, vsebovanih v informacijskem sistemu, od preostalega, aplikacijskega dela, informacijskega sistema. S takšno zasnovo informacijskih sistemov poslovna pravila vzdržujemo ločeno od aplikacijskega dela. S tem omogočimo njihovo hitro spreminjanje in posledično hitro in učinkovito prilagoditev informacijskih sistemov na spremembe v organizacijskem okolju, hkrati pa se izognemo nepotrebnih posegom v aplikacijski del informacijskih sistemov.

1. 1 Kaj so poslovna pravila?

Na nek način vsakdo, ki se vsaj malo ukvarja s poslovanjem oz. izhaja iz nekega organizacijskega okolja, ve, kaj so poslovna pravila. Poslovna pravila so pravila neke organizacije, ki jo vodijo skozi vsakodnevna opravila. Brez samih poslovnih pravil bi se morali v organizacijah odločati v hipu ter v primeru vsake odločitve odločati „ad-hoc". Če bi organizacije odločale na tak način, bi jim vsaka odločitev vzela veliko časa, dodatno pa bi bili rezultati odločitev med seboj nekonsistentni. Ravno zaradi tega ima vsaka uspešna organizacija določena poslovna pravila [6].

1. 1. 1 Kaj poslovna pravila niso?

Preden definiramo poslovna pravila in spoznamo njihov razvoj skozi čas, si poglejmo, kaj poslovna pravila niso [6, 10]:

- Poslovna pravila niso programska oprema. Pogosto so implementirana v različnih programih, pri čemer je to le ena izmed možnih rešitev. Lahko so podprta ročno (kar

ni najbolj učinkovito, vendar včasih drugačna rešitev ni možna), lahko so implementirana kot posamezna pravila ali množice le-teh v sklopu sistemov za upravljanje poslovnih pravil ali v sklopu sistemov za podporo odločanju. Bistvo tega je, da poslovna pravila izhajajo iz poslovanja (na kar namiguje že beseda *poslovna*), programska oprema oz. informacijski sistemi pa podpirajo njihovo izvajanje.

- Poslovna pravila niso procesi. Roger T. Burlton je izrazil bistvo poslovnih pravil v stavku: „*Potrebno je ločiti znanje od toka.*", pri čemer sta *znanje* in *tok* dva različna pojma. Pri tem poslovna pravila predstavljajo znanje, ki usmerja tok poslovnih procesov [3].

Kaj torej so poslovna pravila?

Pri razvoju informacijskih sistemov se že dolgo časa zavedajo obstoja treh elementov: podatkov, procesov in pravil. Medtem ko so prva dva združili z uporabo objektno-orientiranega pristopa, so pravila pogosto zanemarili in jih pustili vsebovane v sami programski kodi [15]. Na težave je prvi opozoril Daniel S. Appleton leta 1984 v članku z naslovom „*Business Rules: The Missing Link"* [1]. Poudaril je, da razvijalci informacijskih sistemov težko zagotovijo enotne programske rešitve, če se v organizaciji uporabljajo termini, ki nimajo enotnega imena, oz. se njihov pomen razlikuje med oddelki. Manjkajoči člen naj bi bila ravno poslovna pravila, ki med drugim enolično definirajo pomen vsakega poslovnega termina znotraj organizacije. Pri tem je Appleton termin poslovno pravilo definiral kot „*omejitev, ki obstaja znotraj poslovne ontologije"*. V naslednjih letih so poslovna pravila pridobila tako na pomembnosti, kakor tudi na popularnosti. Prepoznali so jih kot posebne koncepte, ki igrajo ključno vlogo pri razvoju aplikacij, ki so podvržene spremembam in morajo biti zaradi tega prilagodljive. O poslovnih pravilih se je pisalo vedno več, o tej tematiki je bilo tudi vedno več raziskav. Kakor se je s časom spreminjalo dojemanje poslovnih pravil, tako se je s časom spreminjala tudi njihova definicija. Preden podamo definicijo poslovnih pravil, se ustavimo še pri evoluciji slednje.

1. 1. 2 Evolucija definicije poslovnih pravil

Trenutno ne obstaja splošno sprejeta definicija termina poslovno pravilo. Pri pregledu literature, ki se ukvarja s to tematiko, hitro ugotovimo, da ni enotne definicije poslovnega pravila, temveč je definicija odvisna od avtorja in obdobja, v katerem je nastala. Preden podamo lastno definicijo oz. izpostavimo najpogosteje uporabljeno, se zadržimo še pri evoluciji definicije in s tem tudi samem razumevanju poslovnih pravil skozi čas [7, 10].

Leto	*Avtor*	*Definicija*
1984	Daniel S. Appleton	*„… Eksplicitno navedena omejitev, ki obstaja znotraj poslovne ontologije."*
1987	Ronald G. Ross	*„… So specifična pravila (oziroma politika poslovanja), ki določajo … delovanje podjetja in ga razlikujejo od drugih ... Ta pravila upravljajo spremembe v statutu. "*
1994	Ronald G. Ross	*„… Poslovno pravilo predstavlja izjavo o poslovnem obnašanju …"*
1995	GUIDE Business Rules Project Report	*„… (Poslovno pravilo) je izjava, ki definira ali omejuje nekatere vidike poslovanja ... z namenom zagotavljanja poslovne strukture ali nadzora oziroma vpliva na poslovno obnašanje. Poslovnega pravila ne moremo razčleniti na bolj podrobna poslovna pravila ... V primeru nadaljnje razčlenitve izgubimo pomembne podatke o poslovanju."*
1997	Ronald G. Ross	*„Pogoj, dejstvo ali pravilo, ki predstavlja trditev … "*
1998	Business Rules Group	*„Smernica, ki nadzoruje ali vpliva na poslovno obnašanje. Take smernice obstajajo za podporo poslovne politike, ki nastane kot odziv na poslovna tveganja, grožnje ali priložnosti."*
2000	Ronald G. Ross in Gladys S. W. Lam	*„Nedeljiv, jasno opisan del poslovne logike, ki ga lahko ponovno uporabimo."*
2000	Business Rules Group	*„Poslovno pravilo je izjava, ki definira ali omejuje nek vidik poslovanja. Namenjeno je opredelitvi poslovne strukture oziroma vplivu ali nadzoru poslovnega obnašanja. Poslovna pravila … so atomarna – ne moremo jih členiti dalje."*
2001	M. Chisholm	*„Izjava, ki podatke oziroma informacije, ki jih organizacija poseduje, pretvori v druge podatke oziroma informacije ali pa jih uporabi za proženje akcij."*
2002	Barbara von Halle	*„… (Poslovna pravila so) pogoji, ki usmerjajo poslovne dogodke tako, da so sprejemljivi za podjetje."*
2002	Tony Morgan	*„Poslovno pravilo je jedrnata izjava o vidiku podjetja ... Je omejitev, ki določa, kaj se mora zgoditi in kaj se ne sme. V vsakem trenutku mora biti mogoče ugotoviti, ali je pogoj, ki ga določa omejitev, resničen tudi v logičnem smislu. Če ni, je treba izvesti ustrezne popravke. Tako interpretacijo lahko iz perspektive programske opreme opišemo kot boolean, zaradi tega se tudi pogosto uporablja izraz poslovna logika."*
2003	Ronald G. Ross	*„… poslovna pravila so grajena neposredno iz pogojev in dejstev. Pravila določajo, kaj se mora in česa se ne sme, in s tem dajejo smisel že obstoječemu modelu pogojev in dejstev in katalogu konceptov. V mnogih poslovnih okoljih se lahko srečamo tudi z več sto ali celo več tisoč pravili. V takih primerih je nemogoče doseči konsistenco brez uporabe skupne baze pogojev in dejstev."*
2003	EMRIS	*„1. Poslovno pravilo je 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.* *2. Poslovna pravila so zahteve, ki izhajajo iz poslovnih ciljev in usmeritev organizacijskega sistema."*

Tabela 1. 1: Evolucija definicije poslovnega pravila

1. 1. 3 Definicija

Kot smo že omenili, danes še vedno ni sprejeta neka splošno veljavna definicija, ki bi natančno opredeljevala poslovno pravilo. Če si ponovno podrobno ogledamo tabelo 1. 1, ugotovimo, da večina definicij omenja besede, kot so omejitev, upravljanje, definiranje, nadzorovanje, pogoji; slednjim pa sledi zveza poslovno obnašanje ali poslovni dogodki. Povedano drugače: „*Poslovna pravila definirajo poslovno obnašanje, ga pogojujejo in omejujejo ter nadzorujejo, skratka nanj vplivajo."*

Izpostavimo še definicijo, ki jo v strokovni literaturi danes najpogosteje slišimo [7, 12, 19]:

„*Poslovno pravilo je 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.*"

1. 2 Klasifikacija poslovnih pravil

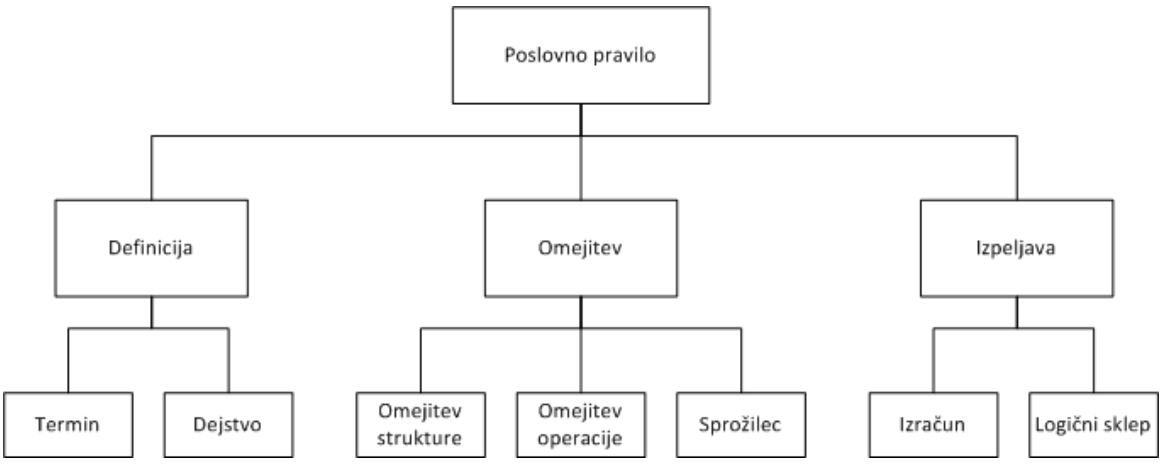
Poslovna pravila lahko razdelimo na različne skupine, pri čemer ugotovimo podobno kot pri njihovi definiciji. Poznamo vsaj deset različnih klasifikacij, ki poslovna pravila razvrščajo po različnih kriterijih, kljub temu pa ni neke standardne klasifikacije. Podrobna analiza klasifikacij pokaže, da so si med seboj podobne in praviloma zajemajo vsaj tri kategorije poslovnih pravil [2]:

- Pravila, ki predstavljajo dejstva iz poslovnega okolja; na primer definicije poslovnih pojmov, razmerja med pojmi ipd.
- Pravila, ki postavljajo omejitve v zvezi s strukturo poslovnega okolja.
- Pravila, ki postavljajo omejitve v zvezi z izvajanjem poslovnih procesov ali pa prožijo posamezne aktivnosti znotraj poslovnega procesa.

V strokovni literaturi je pogosto uporabljena kategorizacija neodvisne organizacije Business Rules Group, ki se zavzema za izdelavo standardov o naravi in strukturi poslovnih pravil. Leta 2000 so izdali končno različico poročila, v katerem opisujejo kategorije poslovnih pravil in njihov vpliv na samo strukturo podatkov [19]. V omenjenem poročilu ločijo tri glavne kategorije poslovnih pravil, ki jih navajata tudi [2, 7]:

- strukturne trditve ali definicije,
- trditve o delovanju ali omejitve,
- izpeljave.

Posamezne kategorije se delijo nadalje na podkategorije, kar prikazuje slika 1. 1.



Slika 1. 1: Kategorije poslovnih pravil [2, 7]

1. 2. 1 Definicije

Definicije predstavljajo posebno kategorijo poslovnih pravil, saj njihova vsebina ni tipična za poslovna pravila. Ob omembi besede pravilo si ponavadi predstavljamo neke omejitve, na kar napeljuje že sama definicija poslovnega pravila. Definicije predstavljajo posebno kategorijo, saj ničesar ne omejujejo, temveč definirajo koncepte in dejstva iz poslovnega okolja. Povedo, da v poslovnem okolju nekaj obstaja bodisi kot pomemben koncept bodisi v razmerju do drugih konceptov. Delijo se na:

- **termine** (besede ali fraze, ki imajo za poslovni sistem specifičen pomen; delimo jih lahko na poslovne termine in skupne termine),
- **dejstva** (opredeljujejo razmerja med termini iz poslovnega okolja in lahko zajemajo tudi več terminov. Obstaja več klasifikacij dejstev. Lahko jih delimo na osnovna dejstva, ki predstavljajo nekaj, kar vemo o poslovnem okolju; in izpeljana dejstva, ki jih je mogoče izpeljati iz osnovnih dejstev. Druga možna delitev dejstev upošteva možna razmerja med termini in kot taka loči med [19]:
    - termini, ki so *atributi* drugih terminov,
    - termini, ki *generalizirajo* (specializirajo) več drugih terminov,
    - termini, ki so *v razmerju* z drugimi termini, pri čemer so možna naslednja razmerja: agregacija, vloga in asociacija).

Kategorija poslovnega pravila	Primer
definicija – termin	Stranka je oseba, s katero se sklene pogodba o najemu avtomobila.
definicija – dejstvo	Model avtomobila (npr. Volkswagen Golf) pripada neki avtomobilski kategoriji (npr. Kategorija D)

Tabela 1. 2: Primeri definicij (povzeto po [19])

1. 2. 2 Omejitve

Naslednja kategorija poslovnih pravil so omejitve. Omejitve se ukvarjajo z dinamičnimi vidiki poslovanja, natančneje z določanjem predpisov, ki morajo veljati v zvezi s strukturo ali delovanjem organizacijskega sistema. Omejitve se delijo na:

- **omejitve strukture** (določila, ki se nanašajo na strukturne lastnosti organizacije; pogosto nastopajo v povezavi z dejstvi in določajo omejitve v zvezi z njimi),
- **omejitve operacij** (določajo pogoje, ki morajo biti izpolnjeni pred ali po operaciji, da bi se ta pravilno izvedla; so ključne za izvedbo operacije in neodvisne od dogodka, ki jih proži),
- **sprožilce** (za razliko od omejitev operacij njihova vsebina določa operacije, ki se sprožijo, če se zgodi določen dogodek in so poleg tega izpolnjeni vsi potrebni pogoji; odvisni so od sprožitvenega dogodka, saj pogoje preverjajo le v primeru, če se določen dogodek dejansko zgodi).

Kategorija poslovnega pravila	Primer
omejitev – omejitev strukture	Stranka ima lahko več rezervacij avtomobilov v prihodnosti, vendar ima lahko istočasno v najemu samo en avtomobil.
omejitev – omejitev operacije	Strankam, ki so na črni listi, se vozil ne sme dati v najem.
omejitev - sprožilec	Če stranka zamuja z vračilom avtomobila, se do šest ur zamude obračuna po urni tarifi najema; za več kot šest ur zamude se zaračuna celoten dan.

Tabela 1. 3: Primeri omejitev (povzeto po [19])

1. 2. 3 Izpeljave

Zadnja kategorija poslovnih pravil so izpeljave. Nastanejo kot logični sklepi ali matematični izračuni iz terminov, dejstev, drugih izpeljav in celo omejitev. Izpeljave se delijo na:

- **izračune** (algoritmi, ki predpišejo postopek izračuna za neko poslovno kategorijo).
- **logične sklepe** (temeljijo na logičnem sklepu in so največkrat oblike *„če velja A, potem velja tudi B”*).

Kategorija poslovnega pravila	Primer
izpeljava – izračun	Znesek najema avtomobila je enak zmnožku zneska dnevnega najema avtomobila in števila dni najema.
izpeljava – logičen sklep	Znesek dnevnega najema avtomobila je odvisen od kategorije, ki ji avtomobil pripada.

Tabela 1. 4: Primeri izpeljav (povzeto po [19])

2 Sistemi za upravljanje poslovnih pravil

2. 1 Upravljanje poslovnih pravil

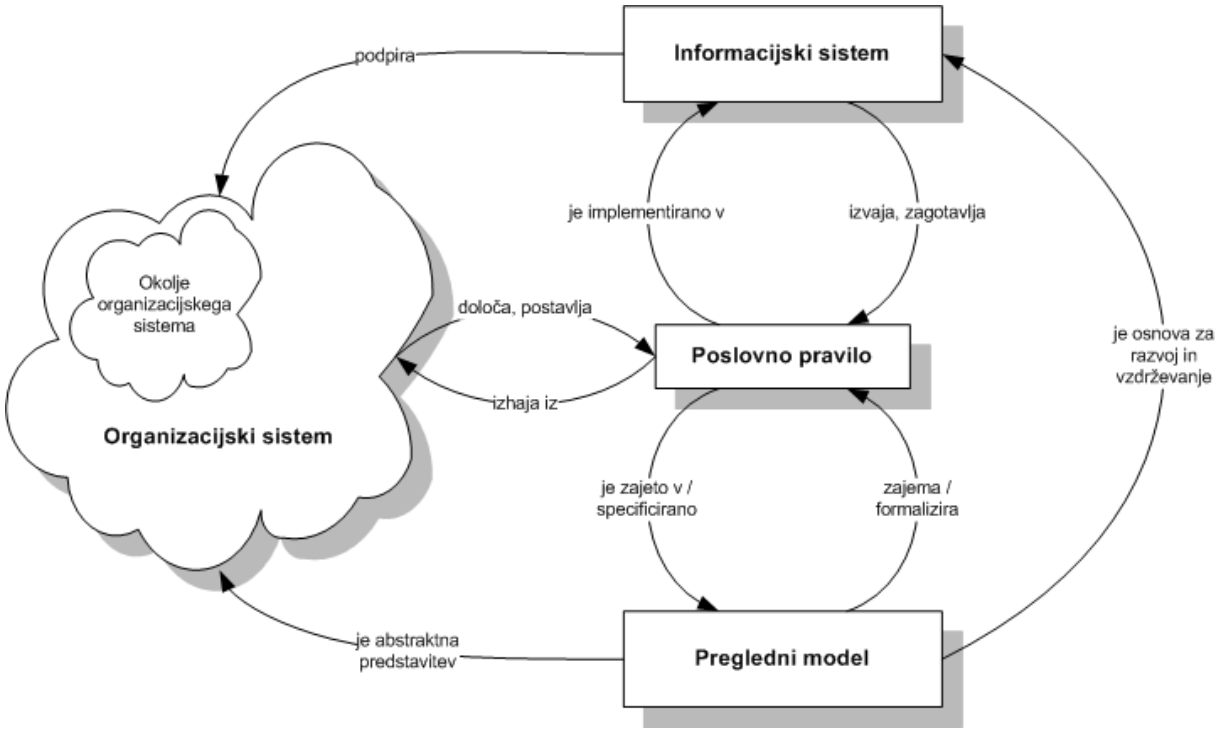
Iz definicije poslovnega pravila je razvidno, da so le-ta prisotna v organizacijah že od nekdaj. Še pred pojavom informacijskih sistemov so se v organizacijah srečevali s pravili in omejitvami, ki so jih vodila pri doseganju ciljev. S pojavom informacijskih sistemov so začeli pravila in omejitve vključevati v informacijske sisteme. Pri tem je že leta 1984 Appleton [1] opozoril na težave zaradi neenotnega pojmovanja terminov znotraj organizacij oz. različnega pomena med oddelki. Iz tega je razvidno, da zavedanje o obstoju poslovnih pravil ni dovolj, temveč je treba nad njimi imeti vzpostavljen ustrezen nadzor. Ta je še posebej pomemben v primeru, ko poslovna pravila ločimo od preostalega informacijskega sistema. Zaradi neprestanih sprememb, katerim so poslovna pravila podvržena, lahko le z ustreznim nadzorom nad njimi omogočimo hitro in učinkovito spreminjanje poslovnih pravil, zajetih v informacijskem sistemu in dodajanje novih pravil. Pri nadzoru poslovnih pravil ločimo med obvladovanjem poslovnih pravil (ang. *Business Rules Governance*) in upravljanjem poslovnih pravil (ang. *Business Rules Management)* [17].

Oboje zajema poslovna pravila v celotni organizaciji, pri čemer se obvladovanje poslovnih pravil izvaja na višjem nivoju kot upravljanje. Za učinkovito obvladovanje poslovnih pravil je treba določiti interesne skupine, vloge posameznikov v procesu obvladovanja poslovnih pravil in njihove odgovornosti, ter procese obvladovanja poslovnih pravil. Dodatno je treba vzpostaviti učinkovito vodenje, primerne organizacijske strukture in ustrezno komunikacijo med njimi. Izdelan način obvladovanja poslovnih pravil omogoča učinkovito upravljanje z njimi.

Za razliko od obvladovanja poslovnih pravil je upravljanje poslovnih pravil odgovorno za vsakodnevno izvajanje organizacijskih procesov, povezanih s poslovnimi pravili. Kot tako vsebuje aktivnosti odkrivanja poslovnih pravil v organizaciji in njihovo analizo ter avtorstvo, validacijo in postavitev v informacijski sistem.

- **Odkrivanje** (ang. *Discovery*) omogoča identifikacijo poslovnih pravil v organizaciji. Poslovna pravila črpa iz dokumentacije in drugih virov izsledljivih informacij.
- **Analiza** (ang. *Analysis*) se ukvarja z analizo poslovnih pravil z namenom njihovega optimalnega delovanja.
- **Avtorstvo** (ang. *Authoring*) zajema implementacijo poslovnih pravil, vključenih v informacijski sistem.
- **Validacija** (ang. *Validation*) se ukvarja s testiranjem implementiranih poslovnih pravil, s čimer zagotavlja njihovo pravilno delovanje.
- **Postavitev** (ang. *Deployment*) predstavlja dejansko vključitev implementiranih in validiranih poslovnih pravil v informacijski sistem.

Podobno kot predstavlja izdelan način obvladovanja poslovnih pravil osnovo za učinkovito upravljanje poslovnih pravil, predstavlja učinkovit način upravljanja poslovnih pravil osnovo za njihovo implementacijo v informacijskem sistemu. Pri tem predstavljajo poslovna pravila, zajeta v informacijskem sistemu, le del poslovnih pravil, ki veljajo v celotnem organizacijskem sistemu, kar prikazuje tudi konceptualni model poslovnih pravil na sliki 2. 1.



Slika 2. 1: Konceptualna shema poslovnih pravil [7]

Kot je razvidno iz konceptualnega modela, predstavlja t. i. pregledni model poslovnih pravil abstraktno predstavitev organizacijskega sistema in kot tak zajema vsa identificirana poslovna pravila. Pregledni model predstavlja osnovo za razvoj in vzdrževanje informacijskega sistema, ki zajema le neko podmnožico poslovnih pravil. Pri slednjih moramo tako ločiti med:

- poslovnimi pravili na nivoju celotne organizacije in
- poslovnimi pravili, zajetimi v informacijskem sistemu.

V sklopu upravljanja poslovnih pravil se tako aktivnosti odkrivanja in analize navezujeta na poslovna pravila na nivoju celotne organizacije, medtem ko se avtorstvo, validacija in postavitev poslovnih pravil navezujejo le na poslovna pravila, vključena v informacijski sistem.

Kot smo že omenili, učinkovito obvladovanje in upravljanje nista toliko pomembna zaradi začetne identifikacije in implementacije poslovnih pravil v informacijskem sistemu, kakor zaradi kasnejših sprememb, katerim so poslovna pravila neprestano podvržena.

2. 1. 1 Življenjski cikel poslovnih pravil

Dejstvo, da so poslovna pravila v organizacijah podvržena neprestanim spremembam, ne pomeni le, da se pogosto spreminjajo, temveč da se tudi neprestano pojavljajo nova poslovna pravila, prav tako pa obstoječa pravila izgubljajo na veljavi (niso več aktualna). Zato lahko za poslovna pravila rečemo, da imajo svoj življenjski cikel. To še posebej velja za pravila, vsebovana v informacijskem sistemu.

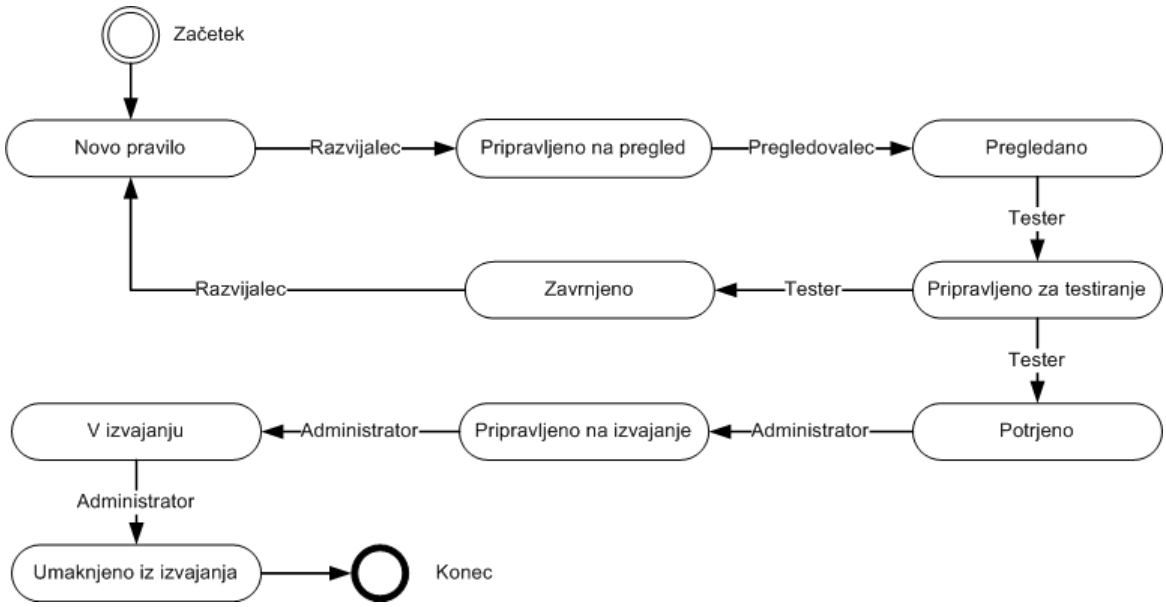
Če potegnemo vzporednico med poslovnimi pravili in človeškim življenjem, lahko rečemo, da pojav novega poslovnega pravila predstavlja njegovo rojstvo; ko je poslovno pravilo dovolj zrelo (implementirano in validirano), ga vključimo med obstoječa pravila v informacijskem sistemu; ko izgubi na veljavi, pa konča svojo življenjsko pot in ga odstranimo iz informacijskega sistema. Življenjski cikel poslovnih pravil v neki organizaciji je določen v sklopu upravljanja poslovnih pravil in se tako razlikuje od organizacije do organizacije. V tem ciklu poslovnih pravil, zajetih v informacijskem sistemu, v splošnem ločimo naslednja stanja [17]:

- **Novo pravilo**: poslovno pravilo je identificirano in ga razvijalec lahko implementira.
- **Pripravljeno na pregled**: poslovno pravilo je implementirano in čaka na pregledovalca, da preveri njegovo ustreznost (skladnost s predvidenim poslovnim obnašanjem).
- **Pregledano**: poslovno pravilo je pregledano s strani pregledovalca.
- **Pripravljeno na testiranje**: poslovno pravilo je pripravljeno na testiranje. V tem stanju bo ostalo, dokler ga tester ne potrdi ali zavrne.
- **Zavrnjeno**: poslovno pravilo ni uspešno prestalo testiranja.
- **Potrjeno**: poslovno pravilo je uspešno prestalo testiranje.
- **Pripravljeno na izvajanje**: zadnje stanje pred postavitvijo poslovnega pravila v izvajalno produkcijsko okolje.
- **V izvajanju**: administrator je postavil novo različico aplikacije poslovnih pravil, ki vključuje tudi novo poslovno pravilo, v izvajanje na produkcijsko okolje.
- **Umaknjeno iz izvajanja**: poslovno pravilo ni več veljavno in je kot tako umaknjeno iz aplikacije poslovnih pravil, ki se izvaja na produkcijskem okolju.

Kot je razvidno iz opisa posameznih stanj, je v življenjski cikel poslovnih pravil vključenih več različnih vlog:

- **Razvijalci** skrbijo za implementacijo poslovnih pravil.
- **Pregledovalci** skrbijo za pregled implementiranih poslovnih pravil, to je njihovo skladnost s predvidenim poslovnim obnašanjem. Največkrat izvirajo iz vrst poslovnih uporabnikov, ki dobro poznajo poslovni sistem in pravila ter imajo ustrezno tehnično znanje za razumevanje implementiranih poslovih pravil.
- **Testerji** izvajajo testiranje poslovnih pravil (preverjajo, ali se poslovna pravila izvajajo skladno s pričakovanji).
- **Administratorji** imajo ustrezne pravice in znanje za postavitev aplikacij poslovnih pravil na produkcijsko izvajalno okolje.

Slika 2. 2 prikazuje navedena stanja, skupaj s prehodi med njimi in vlogami, odgovornimi za prehode.



Slika 2. 2: Življenjski cikel poslovnih pravil

Kot smo predhodno omenili, se dejanski življenjski cikel poslovnih pravil razlikuje med organizacijami. Zgornja slika tako prikazuje splošni življenjski cikel poslovnih pravil, iz katerega lahko organizacije izhajajo pri definiranju lastnega življenjskega cikla poslovnih pravil, ki je prilagojen njihovim zahtevam in pričakovanjem. Pri tem upoštevajo različne dejavnike, kot npr. uporabljana razvojna orodja, število testnih okolij, izvajalno okolje, uporabnike, ki so vključeni v delo s poslovnimi pravili, vrste pravil, ki se pojavljajo v organizaciji itd. Dodatno lahko definirajo več različnih življenjskih ciklov, npr. lastnega za vsako vrsto poslovnih pravil, ki se pojavljajo v organizaciji.

2. 2 Kaj so sistemi za upravljanje poslovnih pravil

Sisteme za upravljanje poslovnih pravil (ang. *Business Rule Management System* ali krajše BRMS; v nadaljevanju SUPP) marsikdo enači s stroji za izvajanje poslovnih pravil, ki omogočajo ločeno izvajanje deklarativno opisanih poslovnih pravil od preostalega informacijskega sistema. Vendar predstavljajo stroji za izvajanje poslovnih pravil oz. t. i. izvajalno okolje le eno izmed komponent SUPP. Osnovni namen SUPP je, kot že samo ime pove, podpora upravljanju poslovnih pravil. V ta namen vsebujejo SUPP programska orodja, ki omogočajo ločitev poslovnih pravil od preostalega informacijskega sistema ter njihovo upravljanje. SUPP tako omogočajo definiranje, namestitev, izvajanje in spremljanje izvajanja poslovne logike (aplikacij poslovnih pravil) ter obvladovanje njene kompleksnosti znotraj organizacij, v kar sodi tudi obvladovanje življenjskega cikla poslovnih pravil.

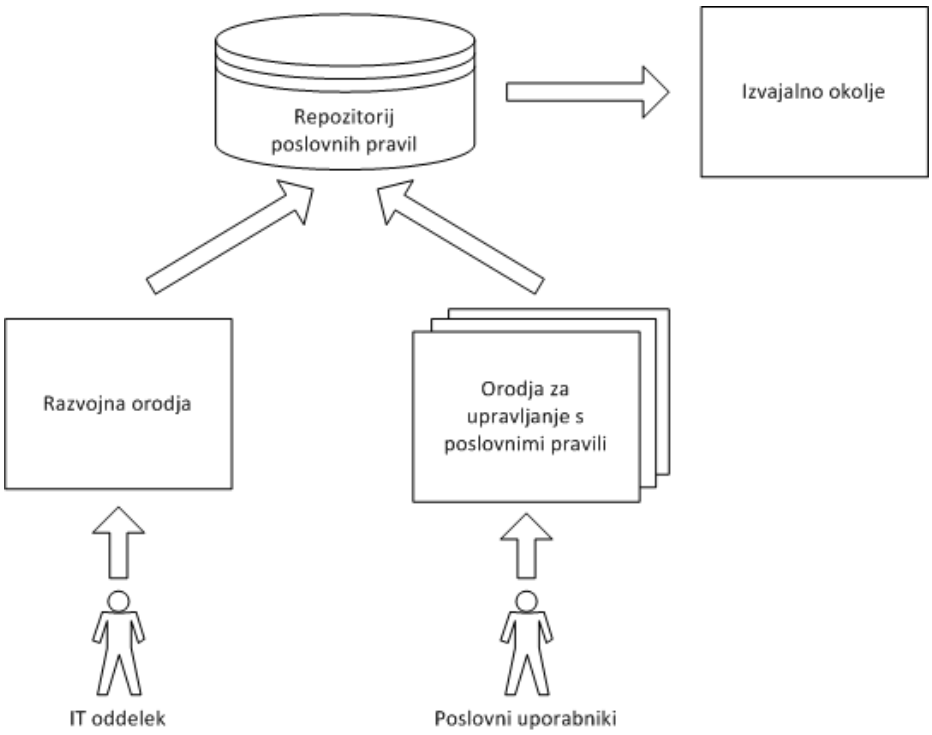
V ta namen vsebujejo SUPP naslednje funkcionalnosti in odgovornosti:

- poslovna pravila hranijo in upravljajo v repozitoriju poslovnih pravil, ki predstavlja politiko in postopke organizacije,
- ločujejo poslovno logiko (poslovna pravila oz. aplikacije poslovnih pravil) od preostale logike informacijskih sistemov,
- omogočajo integracijo z ostalimi aplikacijami, tako da lahko poslovna pravila uporabljamo za vse vrste odločitev,
- združujejo poslovna pravila v neodvisne vendar povezljive množice ter izvajajo sklepanje znotraj posamezne ali več množic,
- omogočajo poslovnim analitikom in uporabnikom ustvarjanje, razumevanje in vzdrževanje poslovnih pravil brez potrebe po tehničnem znanju in posledično obsežnemu izobraževanju,
- avtomatizirajo in olajšujejo poslovne procese,
- ustvarjajo inteligentne aplikacije, s katerimi lahko uporabniki sodelujejo prek naravnih, razumljivih in logičnih dialogov.

2. 3 Komponente

Z namenom zagotavljanja omenjenih funkcionalnosti SUPP v osnovi vsebujejo naslednje komponente [13]:

- repozitorij poslovnih pravil, ki omogoča hranjenje in upravljanje poslovnih pravil ločeno od preostalega informacijskega sistema,
- razvojna orodja, ki omogočajo razvoj in upravljanje poslovnih pravil tako informatikom kot poslovnim ekspertom,
- izvajalno okolje za poslovna pravila.



Slika 2. 3: Osnovne komponente sistema za upravljanje poslovnih pravil

V sklopu orodij za razvoj in upravljanje poslovnih pravil SUPP pogosto uporabljajo domensko specifične programske jezike. Slednji omogočajo bolj natančno izražanje odločitvene logike, pri čemer uporabljajo sintakso poslovnega besednjaka. Dodatno SUPP omogočajo grafično predstavitev poslovnih pravil v obliki odločitvenih tabel, dreves in tokov.

2. 3. 1 Domensko specifični jeziki

Domensko specifični programski jeziki (ang. *Domain Specific Programming Language* ali DSL) so, za razliko od splošno uporabljanih programskih jezikov (ali jezikov GPL), kot so C, java, C#, COBOL, itd., razviti za točno določeno področje uporabe, v našem primeru torej za razvoj poslovnih pravil. Tako se lahko osredotočajo izključno na semantiko tega področja, dodatno pa so visoko deklarativni in opisujejo, kaj se mora zgoditi, namesto kako se nekaj zgodi (slednje zagotavlja njihov interpreter).

Njihove prednosti so [14]:

- **Lažje programiranje**: zaradi višje stopnje abstrakcije, ki se močno navezuje na problemsko domeno in definira, kaj je treba storiti in ne kako, so domensko specifični jeziki lažji za razvoj in samo razumevanje tako s strani razvijalcev kot tudi specialistov obravnavanega področja. Posledično to pomeni tudi krajši čas, potreben za razvoj in cenejše vzdrževanje.
- **Sistematična ponovna uporaba**: za razliko od jezikov GPL, kjer je ponovna uporaba že obstoječih knjižnic prepuščena na izbiro razvijalcem, domensko specifični jeziki

razvijalcem vsiljujejo ponovno uporabo knjižnic, ki jih uporabljajo pri svoji implementaciji. Ker so definirani za točno določeno področje uporabe oz. problemsko domeno, zajemajo in posledično ponovno uporabljajo specifično znanje o tej problemski domeni.

- **Lažja verifikacija**: zaradi že omenjenih lastnosti domensko specifičnih jezikov (navezanost na problemsko domeno, deklarativnost) pogosto že sama validacija (glej poglavje 3. 1) zagotavlja, da bomo dobili pravilne rezultate.
- **Povečano sodelovanje**: uporaba iste poslovne semantike znotraj organizacije pospešuje izmenjavo informacij in zmanjšuje tveganje za razhajanje med dejansko implementacijo poslovne logike in pričakovanji poslovnih uporabnikov.

Domensko specifični jeziki sistemov za upravljanje s poslovnimi pravili tako pogosto omogočajo t. i. verbalizacijo ali ubeseditev objektov, njihovih atributov in metod, ki nastopajo v pravilih. Ubeseditev omogoča zapis poslovnih pravil v jeziku, ki je zelo podoben naravnemu jeziku, s čimer je poslovnim uporabnikom njihovo razumevanje močno olajšano.

Za lažje razumevanje podajamo primer poslovnega pravila, zapisanega v javi, in istega poslovnega pravila zapisanega v jeziku BAL (ang. *Business Action Language*). BAL je domensko specifični jezik, ki se uporablja v SUPP IBM WebSphere ILOG JRules za pisanje poslovnih pravil.

Pravilo	Strankam, ki so na črni listi, se vozil ne sme dati v najem.

Java	`Customer customer = contract.getCustomer();`
	`if (customer.isBlacklisted()) {`
	`  contract.setApproved(false);`
	`}`

BAL	**`definitions`**
	`  set customer to the customer of the contract;`
	`if`
	`  customer is blacklisted`
	`then`
	`  refuse the contract;`

Tabela 2 .1: Zapis poslovnega pravila v naravnem jeziku, javi in BAL-u

2. 4 Prednosti

Orodja za upravljanje poslovnih pravil, ki so namenjena poslovnim uporabnikom; uporaba domensko specifičnih jezikov in grafična predstavitev poslovnih pravil so ključne prednosti SUPP, ki omogočajo delo s poslovnimi pravili tudi poslovnim uporabnikom. Možnost

neposredne vključenosti poslovnih uporabnikov v delo s poslovnimi pravili, ki predstavljajo osnovo poslovne logike, predstavlja tudi eno glavnih vodil pri razvoju poslovnih pravil. SUPP tako nudijo podporo naslednjim določilom Manifesta poslovnih pravil [20]:

- *„Pravila morajo biti za poslovne uporabnike izražena deklarativno v naravnem jeziku."*
- *„Poslovna pravila morajo biti izražena na tak način, da lahko poslovni uporabniki validirajo njihovo pravilnost."*
- *„Neposredno izvajanje poslovnih pravil v stroju za izvajanje poslovnih pravil je boljša strategija za njihovo implementacijo kot prepisovanje poslovnih pravil v proceduralno obliko."*
- *„Sistem, ki skrbi za izvajanje poslovnih pravil, mora omogočati razlago, kako je do neke odločitve prišlo oz. zakaj se je neka akcija zgodila."*
- *„Poslovna pravila naj izvirajo iz poslovnih uporabnikov z ustreznim znanjem."*
- *„Poslovni uporabniki morajo imeti na razpolago orodja za formulacijo, validacijo in upravljanje poslovnih pravil."*
- *„Poslovni uporabniki morajo imeti na razpolago orodja, ki jim omogočajo preverjanje medsebojne konsistence poslovnih pravil."*
- *„Pravila in zmožnost njihovega učinkovitega spreminjanja so ključni za izboljšanje poslovne prilagodljivosti."*

Uporaba SUPP za upravljanje poslovnih pravil tako prinaša organizacijam vrsto prednosti:

- boljšo uravnanost in razumevanje med poslovni uporabniki in IT oddelkom ter zmanjšano ali celo ukinjeno odvisnost od IT oddelka za spremembe poslovne logike,
- hitrejši razvoj in vzdrževanje poslovne logike ter povečan nadzor nad razvito poslovno logiko,
- jasnejšo revizijo,
- višji nivo ponovne uporabljivosti poslovne logike,
- konsistenco terminov na nivoju celotne organizacije,
- izboljšano učinkovitost poslovnih procesov zaradi večje avtomatizacije odločanja.

3 Testiranje poslovnih pravil

„Testiranje programa lahko pokaže samo prisotnost napak, ne pa njihove odsotnosti."
Edsger Dijkstra

Medtem ko se razvoj informacijskih sistemov z ločenimi poslovnimi pravili od preostale logike ne razlikuje toliko od razvoja klasičnih informacijskih sistemov, pa to ne velja za postopek testiranja. Pri razvoju klasičnih informacijskih sistemov predstavlja testiranje le eno izmed faz oz. aktivnosti v razvoju. Ko je sistem dokončno razvit in vpeljan v delovanje oz. prevzet s strani naročnika, je njegov razvoj končan in testiranje ni več potrebno, z izjemo kasnejših nadgradenj oz. dodajanj funkcionalnosti.

Z ločitvijo poslovnih pravil od preostalega informacijskega sistema pa želimo omogočiti njihovo prilagajanje na spremembe, kar pomeni da se poslovna logika, to so poslovna pravila oz. aplikacije poslovnih pravil, neprestano spreminja tudi po vpeljavi sistema v delovanje. Ob vsaki spremembi poslovnih pravil je treba slednja stestirati in tako preveriti pravilnost njihovega delovanja, kar pomeni, da se testiranje neprestano izvaja. Ker se struktura uporabljanih podatkov načeloma ne spreminja, temveč se spreminja le vsebina poslovnih pravil, ni treba testirati celotnega informacijskega sistema, temveč le poslovno logiko. Če se spremeni tudi struktura uporabljanih podatkov, vpliva slednja tudi na strukturo preostalega informacijskega sistema, zato je treba ponovno stestirati celoten sistem. Spremembe strukture podatkov so sorazmerno redke v primerjavi s spremembami same vsebine poslovnih pravil, zato lahko nanje gledamo tudi kot na nadgradnjo celotnega informacijskega sistema, pri kateri je treba v vsakem primeru ponovno stestirati delovanje celotnega sistema.

V nadaljevanju bomo najprej spoznali splošna dejstva o testiranju, nato pa se bomo seznanili s postopkom delta testiranja, ki olajša testiranje poslovnih pravil.

3. 1 Splošno o testiranju

Slovar slovenskega knjižnega jezika (SSKJ) glagol testirati opredeljuje kot:

1. opraviti postopek za ugotavljanje določenih lastnosti, sposobnosti, znanja koga, preizkusiti: testirati kandidate pred sprejemom v delovno razmerje; opraviti postopek za ugotavljanje česa sploh: testirati vinjene voznike
2. opraviti postopek za ugotavljanje ustreznosti, učinkovitosti česa: testirati avtomobile, pralne stroje / testirati zanesljivost kake teorije

Pri testiranju programske opreme veljata obe definiciji:

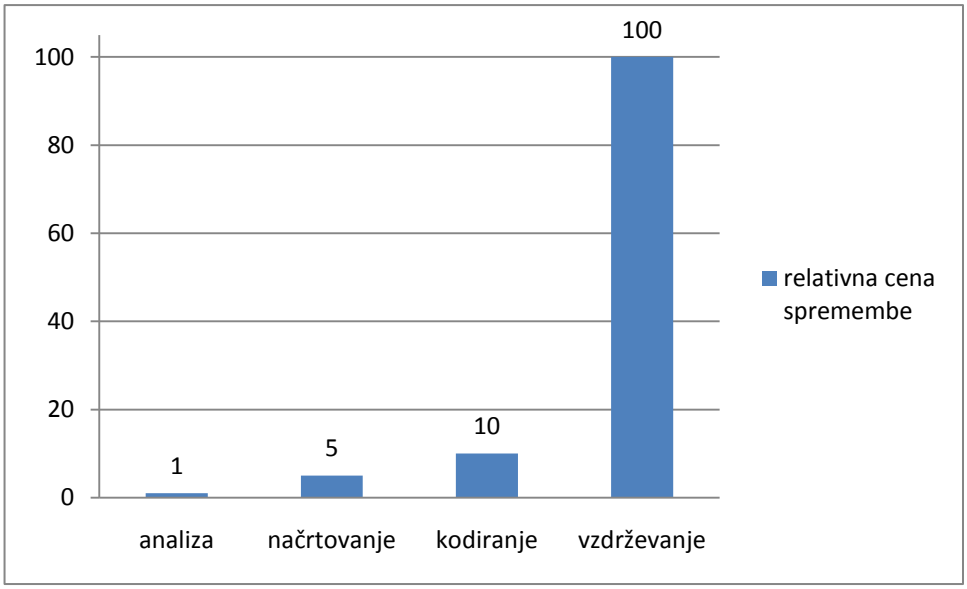
- Iz prve definicije izhaja, da je testiranje programske opreme ugotavljanje njenih lastnosti in preverjanje, ali le-te ustrezajo podanim zahtevam. Temu postopku pravimo **validacija** in preverja, ali gradimo pravi produkt.
- Iz druge definicije izhaja, da je testiranje programske opreme ugotavljanje ustreznosti njenega delovanja in preverjanje zanesljivosti. Temu postopku pravimo **verifikacija** in preverja tehnično pravilnost produkta.

Poleg verifikacije in validacije programske opreme je namen testiranja tudi [16]:

- izboljšanje kvalitete programske opreme
  S tem, ko programsko opremo testiramo, naletimo na napake v programski kodi, ki jih moramo nato odpraviti. Postopku odprave napak pravimo razhroščevanje (ang. *debugging*) in zajema iskanje vzrokov za napake in njihovo odpravo. S tem, ko napake odkrijemo in jih odpravimo, naredimo programsko kodo bolj kakovostno. Izboljšanje kakovosti programske opreme je tako ne samo namen testiranja, temveč tudi njegova posledica.

- preverjanje oziroma ocenjevanje njene zanesljivosti
  Tudi za oceno zanesljivosti programske opreme lahko rečemo, da je posledica testiranja. Med testiranjem je namreč programska oprema podvržena različnim testom, s katerimi ocenjujemo njeno zanesljivost.

Testiranje je tako osrednja aktivnost, ki preverja kakovost programske opreme. Prisotno mora biti skozi celoten razvojni cikel, saj je treba pomanjkljivosti odkrivati sproti in napake preprečevati vnaprej. Če so v programski opremi pomanjkljivosti, velja, da čim kasneje jih bomo zaznali, tem dražje bo njihovo dodajanje. Slika 3. 1 prikazuje ceno sprememb programske opreme od faze analize do faze vzdrževanja.



Slika 3. 1: Ocena sprememb v programski opremi od faze analize do faze vzdrževanja [8]

Kot je razvidno s slike, lahko dodajanje neke funkcionalnosti v fazi vzdrževanja stane celo stokrat več, kakor če bi jo upoštevali že v fazi analize. Podobno prikazuje tabela 3. 1 stroške odpravljanja pomanjkljivosti v posamezni fazi razvoja v odvisnosti od faze pojavitve.

		Faza odkritja				
		Zajem zahtev	*Arhitekturno načrtovanje*	*Kodiranje*	*Testiranje*	*Vzdrževanje*
Faza pojavitve	*Zajem zahtev*	1x	3x	5-10x	10x	10-100x
	Arhitekturno načrtovanje		1x	10x	15x	25-100x
	Kodiranje			1x	10x	10-25x

Tabela 3. 1: Stroški odprave pomanjkljivosti v fazah razvoja glede na fazo pojavitve [18]

Iz tabele je razvidno, da odprava pomanjkljivosti, nastalih v fazi zbiranja zahtev, stane v fazi arhitekturne zasnove 3-krat več, v fazi kodiranja 5- do 10-krat več, v fazi testiranja 10-krat več, v fazi vzdrževanja pa kar 10- do 100-krat več, kot če bi jo zaznali in odpravili že v fazi zbiranja zahtev.

Testiranje programske opreme pogosto enačimo s fazo testiranja, v kateri temu namenjena skupina uporabnikov (testerjev) preveri pravilnost nastalega izdelka, vendar je v tej fazi testiranje le najbolj izpostavljeno. Ker je testiranje ključen dejavnik, ki zagotavlja kakovost programske opreme, mora biti prisotno v vseh fazah razvoja programske opreme. V začetnih fazah razvoja je treba zbrane zahteve validirati, šele v fazi kodiranja lahko začnemo verificirati narejeno. Metode testiranja morajo biti prilagojene posameznim fazam razvoja in posledično izbrani metodologiji razvoja programske opreme. Izbrana metodologija razvoja namreč določa metodologijo testiranja.

Testiranje programske opreme je, če primerjamo ostale faze razvoja, neke vrste destruktivno delo. S testiranjem namreč skušamo programsko opremo „spraviti na kolena”, pokazati, da vsebuje napake ali ne ustreza zahtevam. Ker s testiranjem preverjamo delo, ki je bilo opravljeno v predhodnih fazah razvoja, so lahko analitiki, načrtovalci in koderji osebno prizadeti, če se v rezultatih njihovega dela odkrijejo pomanjkljivosti ali napake. Pride lahko do konflikta interesov, če testiranje izvajajo iste osebe, ki so sodelovale tudi v aktivnostih predhodnih faz. Za testiranje lahko namreč izberejo take testne primere, ki ne preverijo delovanja programske opreme z vseh možnih vidikov.

3. 1. 1 Kategorije in metode testiranja

Kot smo že omenili, mora biti testiranje prisotno v vseh fazah razvoja programske opreme. Poznamo veliko tehnik in metod testiranja, ki služijo več namenom v različnih fazah in

aktivnostih razvoja. Testiranje programske opreme lahko razvrstimo v kategorije glede na več dejavnikov [16]:

- po fazi razvoja:
  - testiranje faze zbiranja zahtev,
  - testiranje faze načrtovanja,
  - testiranje faze kodiranja,
  - vrednotenje rezultatov testiranja,
  - testiranje faze namestitve,
  - sprejemno testiranje (v prisotnosti naročnika),
  - testiranje vzdrževanja.
- po namenu:
  - testiranje pravilnosti,
  - testiranje zmogljivosti,
  - testiranje zanesljivosti,
  - testiranje varnosti.
- po obsegu:
  - testiranje posameznih modulov,
  - testiranje komponent,
  - testiranje integracije,
  - sistemsko testiranje,
  - testiranje sistemske integracije.

V posameznih kategorijah uporabljamo različne metode testiranja, ki jih v osnovi delimo na dve skupini [8]:

- **Metode bele skrinjice**, ki pri načrtovanju testnih primerov upoštevajo notranjo strukturo kode. Drugo ime za metode bele skrinjice je tudi strukturna analiza.
- **Metode črne skrinjice**, pri katerih je notranja struktura kode zastrta. Izbiramo lahko le vhodne podatke in dobljene rezultate primerjamo s pričakovanimi. Metode črne skrinjice imenujemo tudi funkcionalna analiza.

Druga možna delitev metod testiranja je na:

- Statične metode, ki ne zahtevajo izvajanja programa.
- Dinamične metode, pri katerih izvajamo program in njegove rezultate primerjamo s pričakovanimi.

Statične metode predvidevajo branje dokumentov oziroma sledenje programski kodi, zato jih lahko imenujemo tudi ročne metode. Kot take so posebej primerne za začetne faze razvoja programske opreme, ko programska koda še ni razvita: testiranje oziroma verifikacija in validacija faze zbiranja zahtev in faze načrtovanja.

3. 1. 1. 1 Metode bele skrinjice

Po principu bele skrinjice testiramo predvsem posamezne module. Kot smo že omenili, imamo pri tem pristopu na vpogled notranjo strukturo modulov, na podlagi katere tudi načrtujemo testne primere. Med metode bele skrinjice tako sodita:

- **testiranje glavnih poti**
  Osnovni namen metode je, da se vsi ukazi izvedejo vsaj enkrat, kar pomeni, da s testnimi primeri pokrijemo celotno kodo.
- **testiranje zank**
  Izkušnje kažejo, da pri kodiranju zank še posebej pogosto naredimo napake, zato je smiselno sestaviti testne primere za njihovo testiranje, še posebej pri minimalnih in maksimalnih možnih ponovitvah.

3. 1. 1. 2 Metode črne skrinjice

Testiranje po principu črne skrinjice uporabljamo v kasnejših aktivnostih testiranja za preverjanje funkcionalnih zahtev. Metode črne skrinjice tako niso nadomestilo za metode bele skrinjice, saj z njimi odkrivamo druge vrste napak:

- napačne ali manjkajoče funkcije,
- napake uporabniškega vmesnika,
- napake pri dostopu do podatkov v podatkovnih bazah,
- nizko zmogljivost,
- napake pri inicializaciji in na koncu procesiranja.

Metode črne skrinjice se osredotočajo na maksimiranje učinkov testiranja z minimalnimi stroški, kar pomeni, da skušajo s čim manjšim številom testnih primerov odkriti čim več možnih napak. Z enim testnim primerom tako ne želimo preveriti le ene možne napake, temveč (če je le možno) za cel razred podobnih napak. Med metode črne skrinjice sodita:

- **metoda ekvivalentnih particij**
  Vhodne podatke razdelimo na particije ali razrede, pri čemer predpostavimo, da so vrednosti v posamezni particiji ekvivalentne. Potrebujemo le en testni primer za vsako ekvivalentno particijo, pri čemer lahko neka ekvivalentna particija predstavlja množico veljavnih ali neveljavnih podatkov.
- **analiza mejnih vrednosti**
  Izkušnje kažejo, da je običajno več napak pri mejnih vrednostih intervalov vhodnih podatkov kot na njihovi sredini. Analiza mejnih vrednosti tako predvideva testne primere, ki preverjajo mejne vrednosti.

3. 1. 2 Postopek testiranja

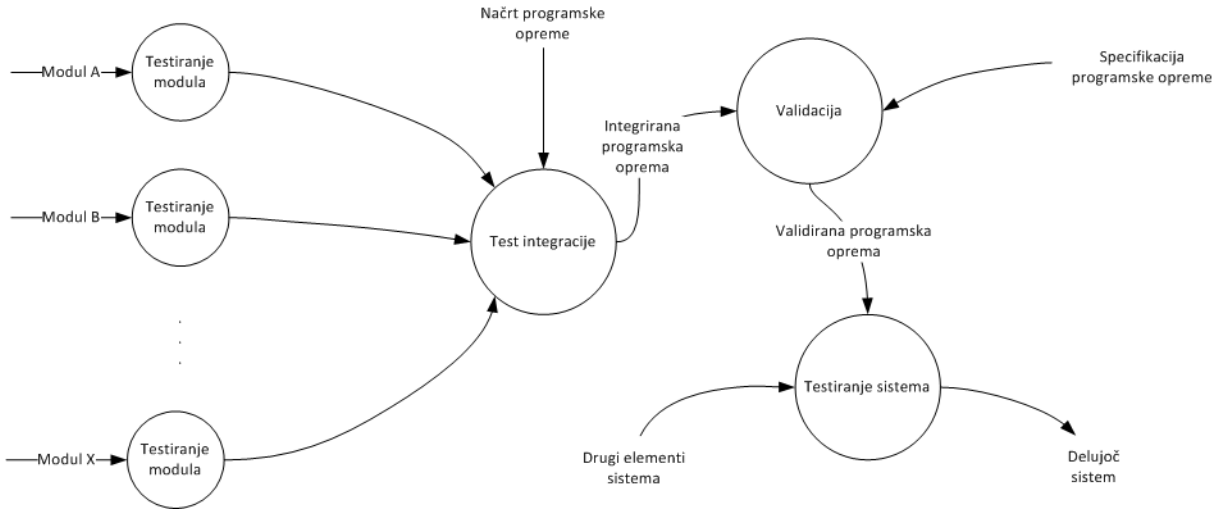
Omenili smo že, da je testiranje v veliki meri določeno z uporabljeno metodologijo razvoja. Pogosto uporabljana praksa je, da testiranje izvaja neodvisna skupina testerjev po končani fazi

kodiranja in pred namestitvijo izdelka oziroma fazo vzdrževanja. V primeru zamude predhodnih faz tako služi faza testiranja pogosto kot nekakšen izravnalnik (ang. *buffer*), kar ogroža čas, namenjen testiranju in posledično tudi uspešnost le-tega in odprave napak. Druga, dosti boljša možnost je, da s testiranjem začnemo takoj ob začetku projekta razvoja programske opreme in ga izvajamo nepretrgano do zaključka projekta. Poleg že opisanih praks se je v zadnjem času pojavila tudi tretja, tako imenovani *testno usmerjeni razvoj* (ang. *test-driven development*), ko razvijalci najprej napišejo teste in se šele nato lotijo kodiranja.

Ne glede na izbrano metodologijo razvoja in posledično testiranja je v splošnem strategija testiranja naslednja [8]:

„Testirati začnemo na nivoju posameznih modulov in postopoma napredujemo k integraciji celotnega sistema. Kot referenca za testiranje med integracijo modulov služi načrt programske opreme. Nato validacija na osnovi zahtev, zapisanih v specifikaciji, ugotavlja, ali jih sistem izpolnjuje. Končno sistemsko testiranje v simuliranih ali realnih razmerah preverja, ali razvita programska oprema pravilno deluje skupaj z drugimi sistemskimi sklopi. Za testiranje posameznega modula je v prvi vrsti zadolžen razvijalec modula. Pri velikih sistemih pa kasnejše testiranje izvaja posebna neodvisna skupina za testiranje."

Grafični potek testiranja programske opreme prikazuje slika 3. 2.



Slika 3. 2: Testiranje programske opreme [8]

Testiranje posameznih modulov poteka po principu bele skrinjice, medtem ko testiranje integracije poteka po principu črne skrinjice. Pri tem mora integracija potekati postopno z uporabo regresijskega testiranja. To pomeni, da vsakič, ko sistem razširimo za en modul, ponovimo testiranje. Ob hkratni združitvi več modulov ali celo celotnega sistema, bi namreč le stežka izolirali vzroke za napake. Na koncu sledi še testiranje celotnega sistema, v sklopu katerega nas ponavadi zanima hitrost programske opreme, zmožnost okrevanja sistema po izpadu, maksimalna obremenitev, varnost in občutljivost sistema, odziv sistema na vnos napačnih podatkov in ukazov itd.

3. 2 Delta testiranje

Testiranje klasičnih informacijskih sistemov in ostale programske opreme poteka tako, da se najprej določi načrt testiranja in testne nabore, s katerimi želimo testirati delovanje sistema. Za posamezne testne nabore se nato določi testne primere, ki predstavljajo osnovo za nadaljnji postopek testiranja (glej poglavje 3. 1. 2). Testni primeri vsebujejo dva nabora testnih podatkov: zbirko vhodnih testnih podatkov in pričakovane izhodne podatke oz. rezultate testiranja. Pri izvajanju nekega testnega primera se zbirka vhodnih podatkov pošlje v izvajanje v testirani sistem, dobljeni rezultati testiranja pa se primerjajo s pričakovanimi rezultati testiranja. Če se ujemajo, je testiranje uspešno in sistem deluje po pričakovanjih.

Čeprav začetna definicija potrebnih testnih primerov vzame veliko časa, saj jih morajo poslovni eksperti, ki poznajo domeno testiranega sistema, definirati ročno, se investicija kmalu povrne. Isti testni primeri se namreč uporabljajo za testiranje med celotnim razvojem sistema in se načeloma ne spreminjajo, če se ne spreminjajo funkcionalnosti sistema. V primeru sprememb funkcionalnosti sistema je treba ustrezno spremeniti testne primere, da testni podatki ustrezno odražajo spremembe v funkcionalnosti.

Kot smo že omenili na začetku 3. poglavja, navedeno ne velja pri razvoju informacijskih sistemov oz. programskih rešitev, ki temeljijo na ločitvi poslovnih pravil od preostalega sistema. Njihova prednost, ki izvira iz dejstva, da so poslovna pravila ločena od preostalega sistema in jih je kot take možno hitro spremeniti, se pri njihovem testiranju izkaže za slabost.

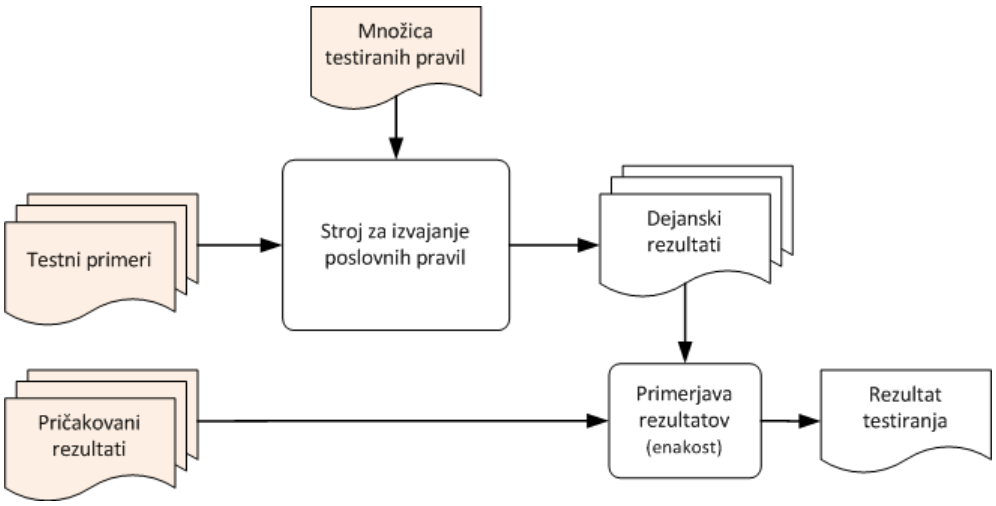
Pri poslovnih pravilih se namreč srečamo s težavo, kako zagotoviti hitro postavitev novih različic poslovnih pravil v izvajanje, hkrati pa jih tudi temeljito stestirati pred samo postavitvijo v izvajanje. Vsaka sprememba poslovnih pravil predstavlja spremembo poslovne logike, ki vpliva na pričakovane rezultate testiranja. V ta namen morajo pri vsaki spremembi poslovnih pravil poslovni eksperti, ki so definirali podatke za testne primere, slednje ponovno pregledati in določiti nove vrednosti pričakovanih rezultatov. To je pri veliki količini testnih primerov časovno potratno in podaljša čas, potreben za postavitev novih različic poslovnih pravil v izvajanje.

Kot rešitev opisane težave je Pierre Berlandier [11] opisal pristop, ki omogoča delno avtomatizacijo testiranja naraščajočih sprememb v poslovnih pravilih in dopolnjuje običajni postopek testiranja oz. uporabo pričakovanih rezultatov v testnih primerih. Zasnovan je na primerjavi rezultatov obstoječih, že stestiranih poslovnih pravil, ki so v izvajanju na izvajalnem okolju (imenovana tudi *branilna množica pravil* – ang. *Champion rule set*), in rezultatov novejše različice poslovnih pravil, ki jih trenutno testiramo (imenovana tudi *izzivalna množica pravil* – ang. *Challenger rule set*). Primerjava rezultatov branilne in izzivalne množice nam pove, ali je razlika v rezultatih med obema množicama sprejemljiva ali ne.

Ker pri testiranju primerjamo samo razliko med rezultati (od tod tudi ime **Delta testiranje**), je postopek testiranja bolj deklarativen in ožje usmerjen kot običajni postopek testiranja. Pri načrtovanju testnih primerov se – namesto „*Kakšne rezultate moramo pričakovati?*" – sprašujemo „*Kakšne spremembe v rezultatih moramo pričakovati glede na predhodno različico pravil?*". Dodatna prednost delta testiranja je v tem, da smo pri posodabljanju testnih primerov manj odvisni od poslovnih ekspertov, s čimer se skrajša čas za testiranje.

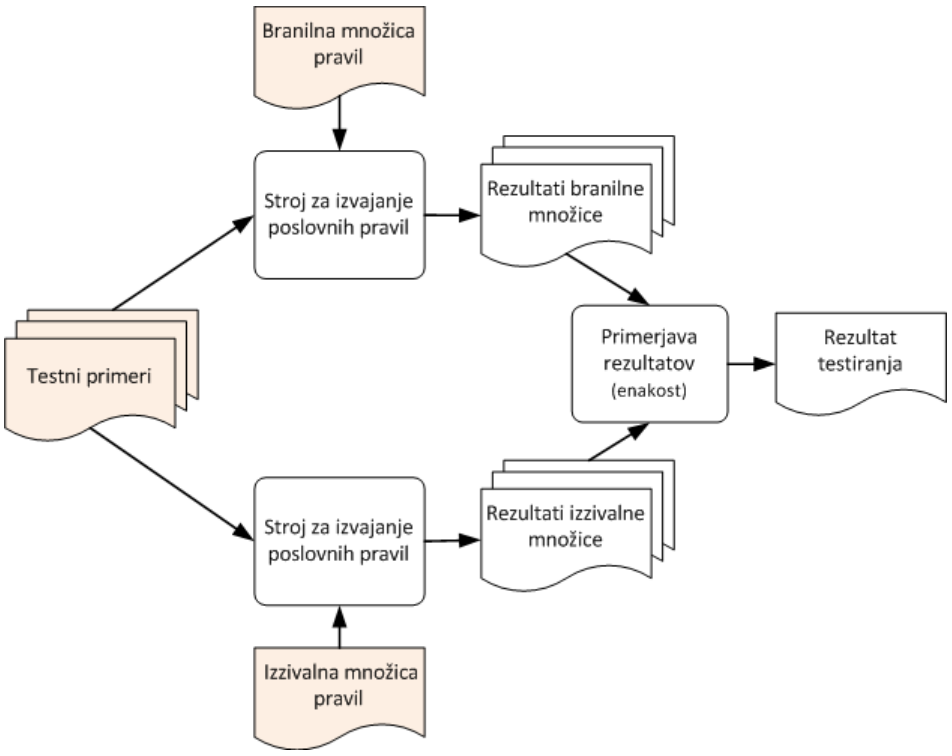
3. 2. 1 Ogrodje

Če poslovna pravila testiramo po običajnem pristopu, je testno izvajalno okolje sestavljeno iz enega samega stroja za izvajanje poslovnih pravil, ki skrbi za izvajanje testiranih poslovnih pravil. Dodatno potrebujemo dva nabora podatkov: testne primere, ki predstavljajo vhodne podatke za testirana poslovna pravila, in pričakovane rezultate, ki jih po izvajanju množice testiranih poslovnih pravil primerjamo z dejanskimi rezultati izvajanja. Slika 3. 3 prikazuje ogrodje običajnega postopka testiranja.



Slika 3. 3: Ogrodje običajnega postopka testiranja [11]

Delta testiranje – z razliko od običajnega postopka – predvideva uporabo dveh strojev za izvajanje poslovnih pravil, enega za branilno množico poslovnih pravil in drugega za izzivalno množico poslovnih pravil. Dodatno predvideva le en nabor testnih podatkov, ki ga predstavljajo testni scenariji. Po končanem izvajanju obeh množic poslovnih pravil nad istimi vhodnimi podatki se rezultati izvajanja obeh množic medsebojno primerjajo. Ogrodje delta testiranja prikazuje slika 3. 4.



Slika 3. 4: Ogrodje delta testiranja [11]

Medtem ko pri običajnem testiranju dobljene in pričakovane rezultate največkrat primerjamo z uporabo striktne enakovrednosti, je delta testiranje bolj prilagodljivo pri primerjavi rezultatov. Če za primerjavo rezultatov branilne in izzivalne množice poslovnih pravil uporabljamo striktno enakovrednost, govorimo o t. i. kvalitativnem delta testiranju; če pa pri primerjavi za izračun enakovrednosti uporabljamo predvidevanja ali funkcije za izračun enakovrednosti, govorimo o t. i. kvantitativnem delta testiranju [11].

3. 2. 2 Kvalitativno delta testiranje

Najlažji način primerjave rezultatov branilne in izzivalne množice poslovnih pravil pri delta testiranju predstavlja uporaba striktne enakovrednosti. Primerjava rezultatov nam pokaže, ali spremembe poslovnih pravil vplivajo na rezultate izvajanja testnih primerov, in razdeli testne primere v dve množici: množico testnih primerov, pri kateri se rezultati izvajanja branilne in izzivalne množice medsebojno ujemajo, in množico testnih primerov, pri kateri se rezultati izvajanja ne ujemajo.

Kvalitativno delta testiranje je kot tako primerno za:

- Preverjanje pravilnosti preurejanja (ang. *refactoring*) poslovnih pravil: med življenjskim ciklom poslovnih pravil lahko pride do potrebe po preureditvi njihove strukture, ki zajema spremembo tokov poslovnih pravil, paketov, v katerih se pravila nahajajo, mest sprožitve poslovnih pravil itd. Po končanem preurejanju poslovnih pravil je treba preveriti za vse obstoječe testne scenarije, ali so rezultati izvajanja

preurejenih poslovnih pravil (izzivalne množice poslovnih pravi) enaki rezultatom izvajanja pravil pred preurejanjem (branilna množica poslovnih pravil).
- Preverjanje stabilnosti sistema po posodobitvi poslovnih pravil: po spremembi poslovnih pravil skoraj vedno obstaja množica testnih primerov, na katere spremenjena poslovna pravila nimajo vpliva. Z izvajanjem kvalitativnega delta testiranja nad omenjeno množico testnih scenarijev preverimo, ali se posodobljena poslovna pravila izvajajo po pričakovanjih.

3. 2. 3 Kvantitativno delta testiranje

Za razliko od kvalitativnega delta testiranja predvideva kvantitativno delta testiranje podrobnejšo primerjavo rezultatov izvajanja branilne in izzivalne množice poslovnih pravil. Namesto uporabe striktne enakovrednosti pri primerjavi rezultatov se rezultati primerjajo na podlagi določenih predvidevanj, npr. na podlagi matematične funkcije, s katero lahko izrazimo spremembo v rezultatih branilne in izzivalne množice poslovnih pravil.

Za lažje razumevanje si poglejmo kvantitativno delta testiranje na primeru aplikacije za izračun kreditne sposobnosti.

Če v primeru omenjene aplikacije spremenimo osnovno obrestno mero, se sprememba obrestne mere odraža v končnem znesku za odplačilo in posledično mesečnem znesku odplačila kredita (ob istem obdobju odplačila). Pri primerjavi rezultatov izvajanja branilne množice poslovnih pravil (stara obrestna mera) in izzivalne množice poslovnih pravil (nova obrestna mera) mora primerjava omenjenih zneskov ustrezno odražati spremembo osnovne obrestne mere.

Čeprav znajo biti spremembe poslovnih pravil sorazmerno pogoste, so dostikrat ponovljive narave in jih lahko kategoriziramo v ustrezne množice ponavljajočih se sprememb. Po nekem številu sprememb poslovnih pravil imamo tako za vsako množico ponavljajočih se sprememb na voljo ustrezne funkcije za izračun enakovrednosti, ki so nam na voljo pri nadaljnjih spremembah poslovnih pravil.

3. 2. 4 Prednosti

Kot je razvidno iz opisa delta testiranja, slednji ne predstavlja zamenjave za običajni postopek testiranja poslovnih pravil, temveč predstavlja njegovo nadgradnjo, ki omogoča enostavnejše testiranje spremenljive narave poslovnih pravil. Začetno različico poslovnih pravil tako še vedno testiramo po običajnem postopku testiranja, pri katerem uporabljamo predhodno definirane testne scenarije in pričakovane rezultate.

Ko začetna različica poslovnih pravil uspešno prestane testiranje, vse nadaljnje spremembe poslovnih pravil, tj. njihove nadaljnje različice, testiramo z uporabo delta testiranja. S tem odpade potreba po neprestanem spreminjanju pričakovanih rezultatov s strani poslovnih ekspertov, saj kot osnovo za primerjavo uporabljamo rezultate izvajanja branilne množice poslovnih pravil (trenutno veljavne različice poslovnih pravil, ki je uspešno prestala predhodno testiranje). S tem, ko je začetna različica poslovnih pravil uspešno prestala testiranje, lahko tudi ustrezno razširimo nabor testnih primerov. Namesto da bi pri delta testiranju uporabljali samo predhodno definirane testne primere, lahko nabor podatkov za delta testiranje razširimo in ga izvajamo neposredno nad dejanskimi podatki v podatkovni bazi. S tem je omogočeno testiranje poslovnih pravil nad dosti večjo količino podatkov.

4 Uporabljani produkti in tehnologije

Kot smo omenili že v uvodu, v podjetju Optilab, d. o. o., razvijamo programske rešitve za detekcijo goljufij v zavarovalništvu, s katerimi smo prisotni na tržišču, prav tako pa razvijamo programske rešitve za znane naročnike. Pri razvoju smo se osredotočili izključno na rešitve, ki uporabljajo omenjeni pristop poslovnih pravil in temeljijo na platformi J2EE, ki je splošno razširjena platforma za razvoj strežniških aplikacij v javi. Kot sistem za upravljanje poslovnih pravil smo posledično izbrali IBM WebSphere ILOG JRules, ki to platformo podpira, prav tako pa je združljiv z WebSphere Application Serverjem, aplikacijskim strežnikom podjetja IBM, na katerem delujejo naše rešitve.

Preden se lotimo obravnavane tematike, se bomo podrobneje seznanili z orodji in tehnologijami, ki jih uporabljamo pri razvoju programskih rešitev. Pri tem se bomo osredotočili na tiste, ki so posredno ali neposredno povezani z obravnavano tematiko. Poleg samega SUPP so to n-nivojska arhitektura, na kateri temeljijo naše rešitve, in knjižnice, ki jih uporabljamo za testiranje.

4. 1 IBM WebSphere ILOG JRules

Leta 2009 je podjetje IBM kupilo ILOG, francosko podjetje, ki je bilo ustanovljeno leta 1987 v Parizu. ILOG razvija aplikacije za štiri različna področja: oskrbovalne verige, upravljanje poslovnih pravil, vizualizacijo in optimizacijo. Njihove rešitve podpirajo različne platforme, med drugim tudi C++, C#, .NET, Javo, AJAX, Adobe Flex/Air.

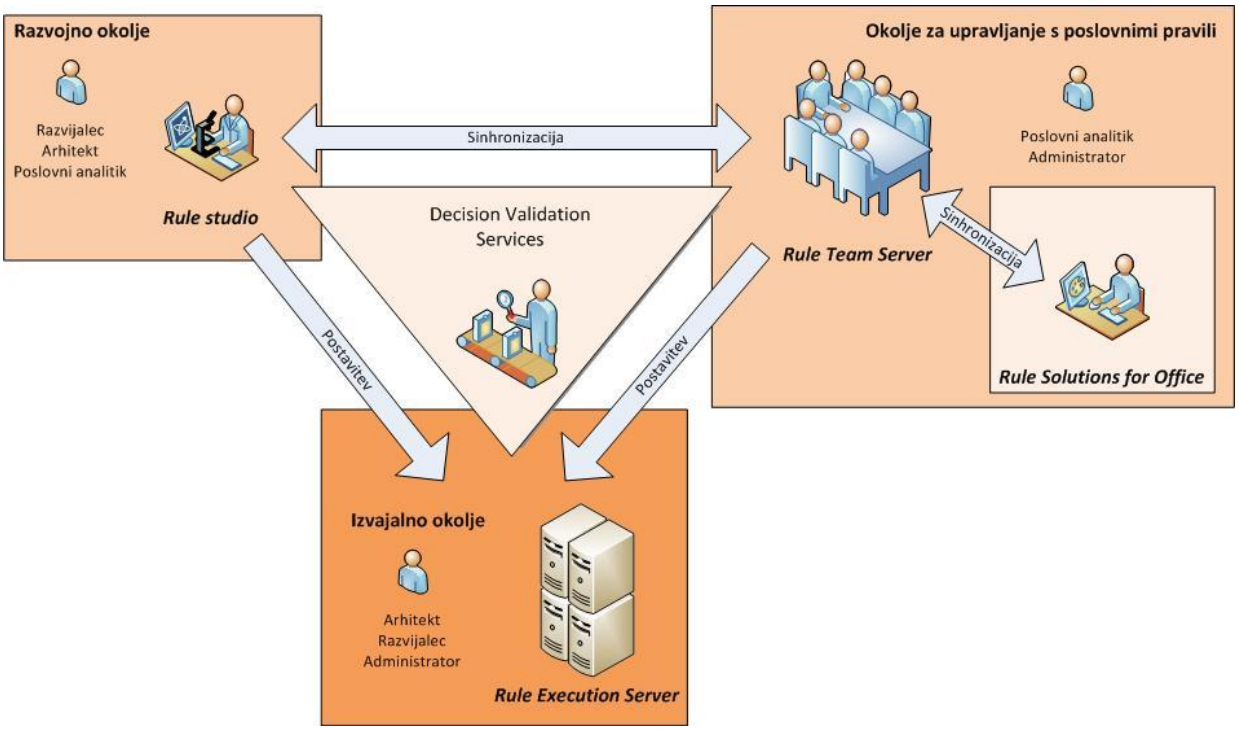
Že leta 1996 je ILOG na tržišče lansiral JRules, lasten sistem za upravljanje s poslovnimi pravili. Kot je razvidno že iz imena, ILOG JRules podpira javansko platformo. Njihov primarni trg za SUPP predstavlja finančni sektor, kjer jih uporabljajo med drugim tudi za razvoj sistemov za trgovanje, sistemov za izračun kreditnih ocen ter podobnih odločitvenih sistemov, poleg tega pa jih s pridom uporabljajo v sistemih za odkrivanje goljufij.

4. 1. 1 Moduli

IBM WebSphere ILOG JRules verzije 7 sestavlja več produktov oz. modulov, ki se med seboj razlikujejo po funkcionalnostih, ki jih nudijo uporabniku, in posledično tudi krogu uporabnikov, katerim so namenjeni:

- IBM WebSphere ILOG JRules
- IBM WebSphere ILOG Rule Team Server
- IBM WebSphere ILOG Decision Validation Services
- IBM WebSphere ILOG Rule Solutions for Office

Slika 4. 1 prikazuje vlogo posameznih produktov v sklopu celotnega sistema za upravljanje poslovnih pravil in uporabnike, katerim so namenjeni.



Slika 4. 1: ILOG JRules produkti in moduli (povzeto po [21])

V nadaljevanju se bomo na kratko zadržali pri poglavitnih značilnostih posameznih produktov.

4. 1. 1. 1 JRules

Drugo ime zanj je tudi JRules Core, saj prestavlja jedro ILOG-ovega SUPP in je sestavljen iz dveh modulov:

- **Rule Studio** je integrirano razvojno okolje za aplikacije poslovnih pravil, namenjeno tehničnim uporabnikom (razvijalcem in arhitektom aplikacij). Temelji na orodju Eclipse in poleg osnovnih funkcionalnosti tega okolja uporabniku nudi tudi dodatne funkcionalnosti, ki so specifične za razvoj poslovnih pravil in njihovih aplikacij. V njem lahko ustvarimo izvajalni in poslovni objektni model in ustrezne povezave med njima, ustvarimo lahko besednjak za pisanje poslovnih pravil, v njem lahko pišemo vse vrste pravil (poslovna in tehnična pravila, odločitvene tabele) ter jih testiramo in razhroščujemo. Dodatno omogoča postavitev novih različic poslovnih pravil oz. njihovih aplikacij na izvajalno okolje (Rule Execution Server), v povezavi z Rule Team Serverjem pa omogoča sodelovanje med razvijalci in poslovnimi analitiki.

- **Rule Execution Server** je izvajalno okolje za aplikacije poslovnih pravil. Omogoča več načinov klicanja aplikacij poslovnih pravil – sinhrono in asinhrono neposredno

klicanje aplikacij, posredno klicanje aplikacij, kot t. i. odločitvenih storitev (ang. *hosted decision services*) – ter upravljanje obstoječih preko administrativne konzole. Možno ga je povezati z razvojnimi orodji, s čimer se omogoči postavitev novih različic aplikacij neposredno iz teh orodij razvijalcem iz Rule Studia in poslovnim uporabnikom iz Rule Team Serverja.

4. 1. 1. 2 Rule Team Server

Za razliko od JRules Core je Rule Team Server v prvi vrsti namenjen poslovnim uporabnikom, ki aktivno sodelujejo pri razvoju poslovnih pravil. Kot že samo ime pove, gre za strežnik, ki omogoča sodelovanje (timsko delo) večjega števila uporabnikov. Ti lahko prek spletne konzole iščejo med že obstoječimi pravili po ključnih besedah, pravila lahko urejajo in dodajajo nova ter testirajo njihovo delovanje in vpliv na obstoječa pravila (v povezavi z Decision Validation Services). Strežnik omogoča nadzor nad dovoljenji za delo in skrbi za zaklepanje pravil med delom z njimi. Dodatno je uporabnikom z ustreznimi pravicami omogočena postavitev novih različic aplikacij poslovnih pravil na izvajalno okolje. Rule Team Server vsebuje tudi repozitorij, ki mu omogoča beleženje zgodovine pravil, upravljanje različic posameznih pravil in tudi celotnih projektov poslovnih pravil. Uporabniki lahko tako kadarkoli vrnejo posamezna pravila ali pa celoten projekt na predhodno verzijo. Dostop do repozitorija je omogočen tudi razvijalcem iz Rule Studia.

4. 1. 1. 3 Decision Validation Services

Vsebuje modul z enakim imenom, ki uporabnikom omogoča testiranje pravilnosti delovanja pravil, poleg tega pa omogoča tudi simuliranje sprememb poslovnih politik (v primeru spreminjanja obstoječih ali dodajanja novih poslovnih pravil). Za testiranje ga lahko uporabljajo tehnični in poslovni uporabniki, saj se modul ob namestitvi integrira v Rule Studio in Rule Team Server.

4. 1. 1. 4 Rule Solutions for Office

Rule Solutions for Office predstavljajo dodatek za programsko zbirko Microsoft Office 2007, ki omogoča poslovnim uporabnikom enostavno urejanje in dodajanje poslovnih pravil neposredno z uporabo programov iz omenjene zbirke, natančneje z uporabo programov Microsoft Word in Microsoft Excel.

4. 1. 2 Koncepti

Za lažje razumevanje si bomo na tem mestu pogledali nekatere koncepte WebSphere ILOG JRules, ki se uporabljajo pri razvoju poslovnih pravil in njihovih aplikacij. Nekatere izmed konceptov smo že omenili, s preostalimi pa se bomo srečali v nadaljevanju.

4. 1. 2. 1 Izvajalni objektni model

Izvajalni objektni model ali krajše XOM predstavlja objektni model, na podlagi katerega se poslovna pravila v ILOGu izvajajo.

Ločimo med:

- javanskim izvajalnim objektnim modelom, ki ga sestavljajo javanski razredi, in
- dinamičnim izvajalnim objektnim modelom, ki ga sestavljajo sheme XML.

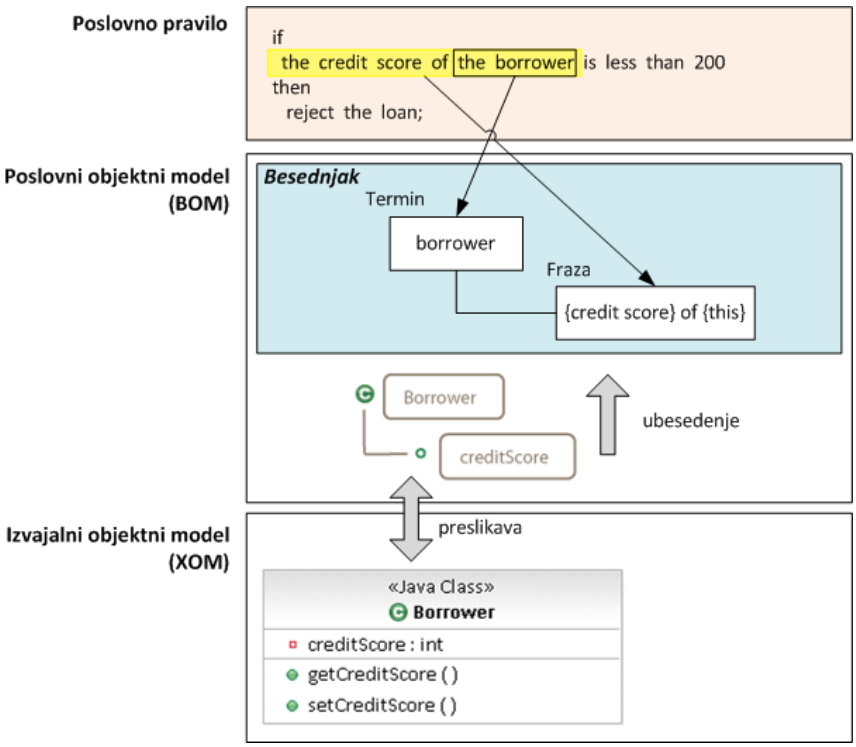
Izvajalni objektni model omogoča izvajalnemu okolju dostop do aplikacijskih objektov in metod, ki so lahko javanski objekti, podatki XML ali podatki iz drugih virov.

V diplomski nalogi bomo izvajalni objektni model enačili z javanskim objektnim modelom. V naših programskih rešitvah namreč uporabljamo izključno slednjega.

4. 1. 2. 2 Poslovni objektni model

Poslovni objektni model ali BOM predstavlja objektni model, ki ga uporabljajo razvijalci in poslovni uporabniki za pisanje poslovnih pravil v ILOG-ovem domensko specifičnem jeziku BAL. Vsebuje orodja, ki omogočajo vzpostavitev besednjaka, zelo podobnega naravnemu jeziku, ki se uporablja za opisovanje poslovne logike.

BOM je po zgradbi zelo podoben javanskemu objektnemu modelu, saj vsebuje razrede, razvrščene v posamezne pakete. Posamezni razredi vsebujejo atribute, metode in asociacije na druge razrede. Razredi in njihove metode ter atributi, definirani v BOM-u, predstavljajo osnovo za pisanje poslovnih pravil. BOM je definiran na podlagi XOM-a, ki predstavlja osnovo za izvajanje poslovnih pravil. Med izvajanjem se tako elementi BOM-a preslikajo v ustrezne elemente XOM-a, kar določa preslikava med objektnima modeloma (t. i. *BOM To XOM* Mapping). Slika 4. 2 prikazuje primer preslikave poslovnega pravila v ustrezne razrede BOM-a in XOM-a.



Slika 4. 2: Preslikava poslovnega pravila v BOM in XOM (povzeto po [21])

4. 1. 2. 3 Projekti poslovnih pravil

Projekti poslovnih pravil omogočajo upravljanje, razvoj in razhroščevanje elementov, ki sestavljajo poslovno logiko aplikacij poslovnih pravil. Predstavljajo poseben tip projektov v Eclipsu, za katere v Rule Studiu obstajata posebni perspektivi za delo z njimi: razvojna perspektiva in perspektiva za sinhronizacijo z Rule Team Serverjem.

V projektih poslovnih pravil definiramo strukturo elementov in vzpostavimo poslovni objektni model ter besednjak za delo z elementi. Za potrebe vzpostavitve BOM-a morajo vsebovati reference na javanske projekte, v katerih je definiran XOM. Dodatno lahko vsebujejo reference na druge projekte poslovnih pravil, kar omogoča razvijalcem uporabo elementov, definiranih v referenčnih projektih.

Projekti poslovnih pravil vsebujejo:

- Elemente pravil, to so: poslovna pravila, odločitvene tabele in drevesa, tehnična pravila, tokovi poslovnih pravil, funkcije, spremenljivke.
- Predhodno definirane poizvedbe, ki omogočajo iskanje po elementih projekta.
- Predloge poslovnih pravil, ki vsebujejo delno napisana poslovna pravila, s pomočjo katerih lahko kreiramo več poslovnih pravil s podobno strukturo.
- Definicijo BOM-a, ki določa njegovo strukturo, besednjak in preslikavo med elementi BOM-a in XOM-om.
- Definirane vhodne in izhodne parametre.

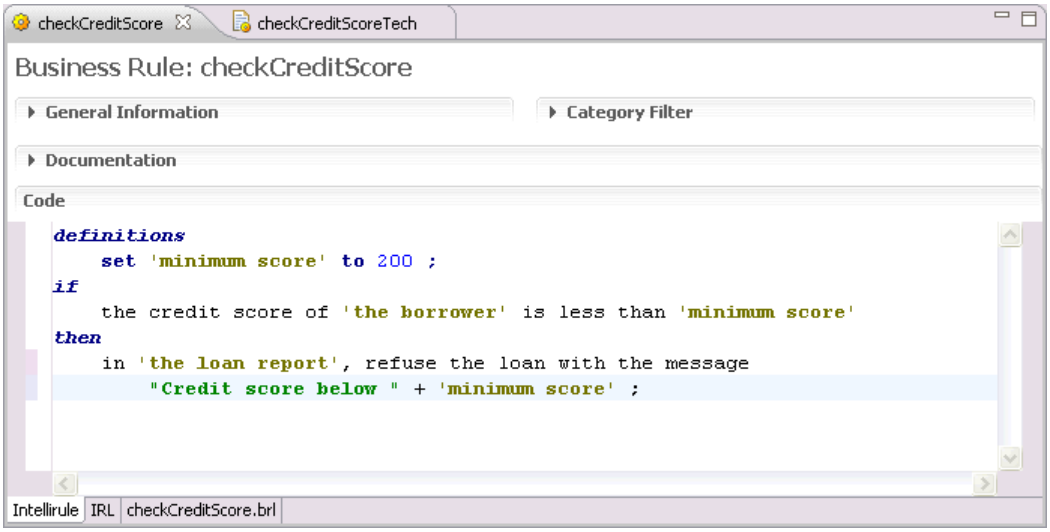
Projekte poslovnih pravil lahko razvijalci shranijo v repozitorij Rule Team Serverja, s čimer omogočijo delo z njimi tudi poslovnim uporabnikom. Slednji lahko uporabljajo vse elemente, definirane v sklopu nekega projekta, ne morejo pa spreminjati vhodnih in izhodnih parametrov nekega projekta poslovnih pravil, prav tako ne morejo spreminjati obstoječega BOM-a.

4. 1. 2. 4 Poslovna in tehnična pravila

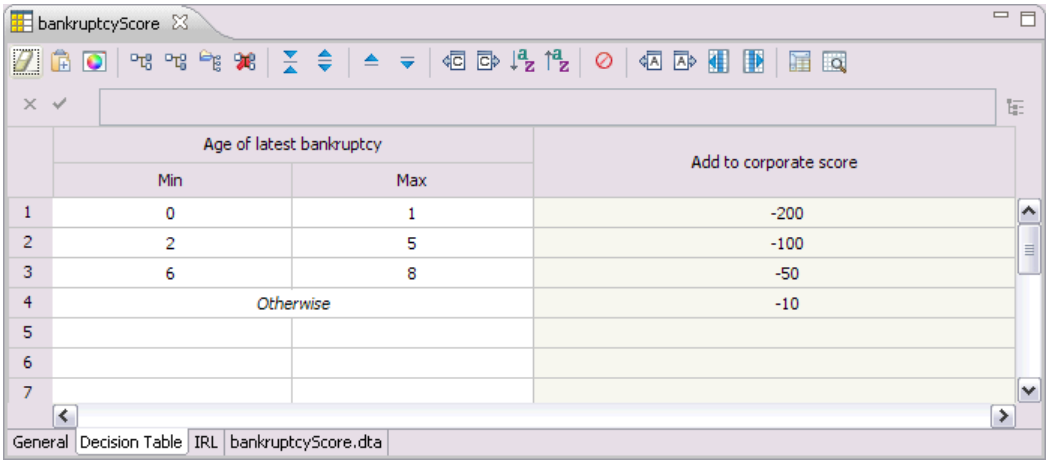
V ILOG-u poslovne politike izražamo z uporabo poslovnih in tehničnih pravil. Za njihov zapis uporabljamo jezik BAL, ILOG-ov domensko specifični jezik, ki uporablja v sklopu BOM-a definiran besednjak. Pravila, zapisana v BAL-u, so zelo podobna pravilom, zapisanim v naravnem jeziku, kar omogoča njihovo enostavno razumevanje. Med poslovna pravila sodijo:

- poslovna pravila,
- odločitvene tabele,
- odločitvena drevesa,
- predloge poslovnih pravil,
- poizvedbe nad poslovnimi pravili.

Sliki 4. 3 in 4. 4 prikazujeta primer poslovnega pravila in odločitvene tabele, zapisanih v jeziku BAL.



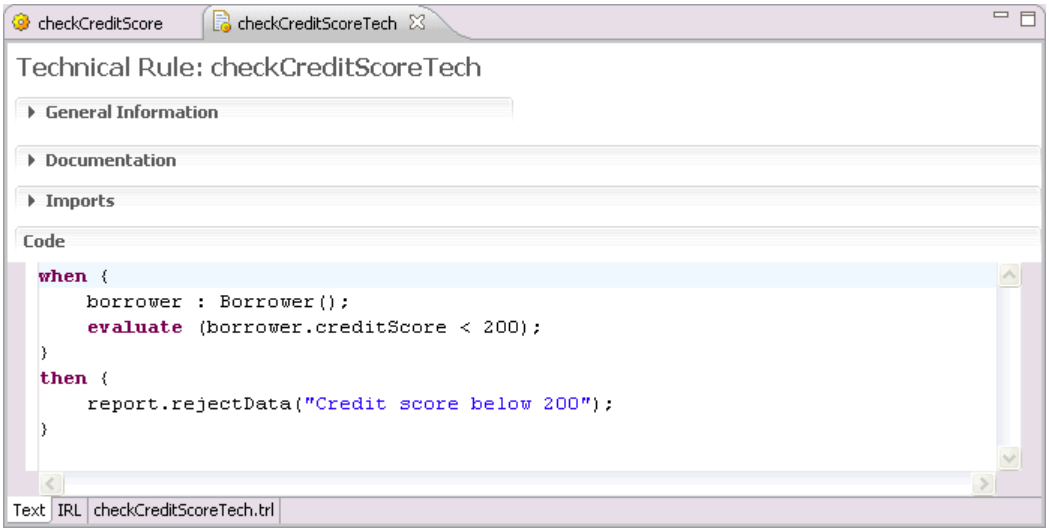
Slika 4. 3: Poslovno pravilo, zapisano v jeziku BAL



Slika 4. 4: Odločitvena tabela, zapisana v jeziku BAL

Za razliko od poslovnih pravil se za zapis tehničnih pravil uporablja jezik IRL, ki je podoben javanski kodi. Tehnična pravila se uporabljajo za zapis bolj kompleksnih zank ali aktivnosti, ki jih je težko izraziti v BAL-u. Med tehnična pravila sodijo: tehnična pravila, funkcije, tokovi poslovnih pravil, spremenljivke ter vhodni in izhodni parametri.

Tokove poslovnih pravil bi lahko sicer uvrstili tudi med poslovna pravila, saj omogočajo pisanje pogojev v odločitvenih zankah tudi v BAL-u, ne samo IRL-ju, prav tako pa so zaradi grafične predstavitve enostavno razumljivi. Za primerjavo poslovnih in tehničnih pravil slika 4. 5 prikazuje poslovno pravilo s slike 4. 3, zapisano kot tehnično pravilo.



Slika 4. 5: Tehnično pravilo, zapisano v jeziku IRL

4. 1. 2. 5 Množice poslovnih pravil

Poslovna pravila, ki se navezujejo na neko odločitev, so za potrebe izvajanja organizirana in shranjena v obliki množic poslovnih pravil (ang. *rulesets*). Množice poslovnih pravil tako

predstavljajo samostojne, izvršljive vsebnike (ang. *executable container*), ki predstavljajo neko odločitev.

Množice poslovnih pravil so zapakirane v obliki arhivskih datotek, ki omogočajo njihovo izvajanje. Vanje lahko izvozimo celotne projekte poslovnih pravil ali pa le neko podmnožico slednjih v sklopu nekega projekta poslovnih pravil. Podmnožico poslovnih pravil določimo s t. i. ekstraktorji (ang. *ruleset extractor*). Poleg samih pravil so v arhivu vsebovani tudi drugi elementi, potrebni za izvajanje na izvajalnem okolju; poleg poslovnih in tehničnih pravil, odločitvenih tabel in dreves vsebujejo tudi tokove poslovnih pravil, tehnične funkcije, množice spremenljivk, BOM, preslikavo med BOM-om in XOM-om ter vhodne parametre vsebovanih projektov poslovnih pravil, ki predstavljajo tudi vhodne parametre za množico poslovnih pravil.

Da bi jih lahko izvajali na izvajalnem okolju, je treba množice poslovnih pravil dodatno zapakirati v aplikacije poslovnih pravil.

4. 1. 2. 6 Aplikacije poslovnih pravil

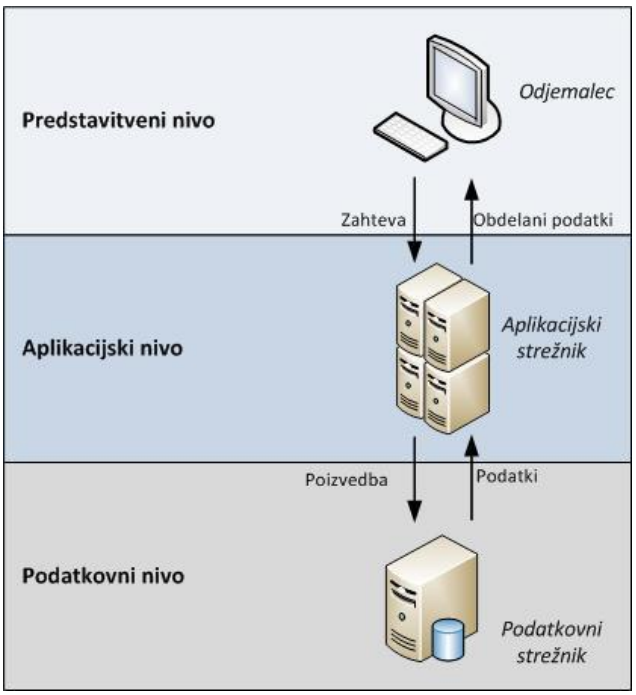
Aplikacije poslovnih pravil (ang. *RuleApps*) so vsebniki (ang. *containers*), ki se izvajajo na izvajalnem okolju. Predstavljajo komponente, ki jih lahko upravljamo in postavimo na izvajalno okolje ter vsebujejo eno ali več množic poslovnih pravil. Največkrat vsebujejo množice poslovnih pravil, ki si delijo isti objektni model. Kot smo omenili že v prejšnjem poglavju, je vsaka množica poslovnih pravil zapakirana v obliki arhivske datoteke, ki omogoča njeno izvajanje. Za lažjo predstavo lahko potegnemo vzporednico z J2EE arhitekturo: aplikacije poslovnih pravil predstavljajo aplikacijske arhive, tj. datoteke EAR, vsebovane množice poslovnih pravil pa predstavljajo posamezne module, tj. EJB JAR arhive.

Do vsake množice poslovnih pravil v sklopu neke aplikacije poslovnih pravil lahko dostopamo individualno z uporabo dostopne poti za množico poslovnih pravil (npr. *ime_aplikacije/ime_množice/*). Dostopna pot predstavlja vstopno točko za odjemalce, preko katere dostopajo do poslovne logike. Izvajalno okolje omogoča razlikovanje med posameznimi različicami aplikacij poslovnih pravil in prav tako omogoča razlikovanje med različicami množic poslovnih pravil, vključenih v neko različico aplikacije poslovnih pravil. Da bi dostopali do ustrezne različice množice poslovnih pravil v neki različici aplikacije poslovnih pravil, je treba pri dostopni poti navesti ustrezni različici aplikacije in množice poslovnih pravil (npr. *ime_aplikacije/1.0/ime_množice/1.3/*). Če različic ne navedemo, bomo dostopali do zadnje veljavne različice aplikacije poslovnih pravil in njenih množic, kar uporabljamo tudi v naših programskih rešitvah, ko dostopamo do aplikacij poslovnih pravil.

4. 2 N-nivojska arhitektura

N-nivojska arhitektura je arhitektura, ki se uporablja za modeliranje in razvoj programskih rešitev. Temelji na arhitekturi tipa odjemalec-strežnik. Za n-nivojsko arhitekturo je značilno, da so prikaz podatkov, procesiranje in uporaba poslovne logike ter dostop do podatkov logično ločeni procesi. Kot taka predstavlja nadgradnjo tronivojske arhitekture, ki je sestavljena iz treh nivojev, prikazanih na sliki 4. 6:

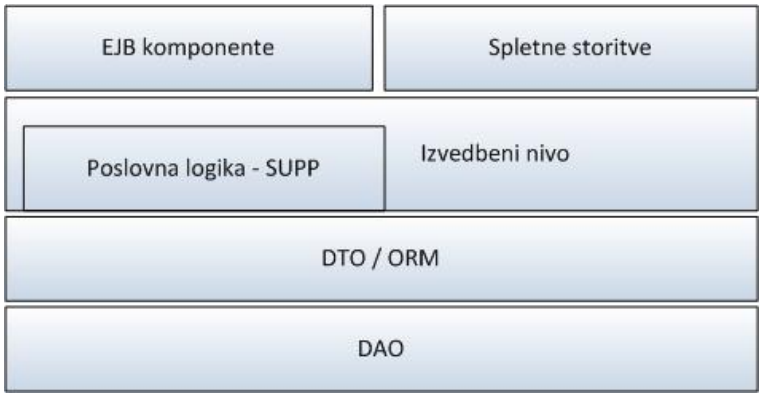
- Nivo odjemalca ali predstavitveni nivo: vrhnji nivo se izvaja na odjemalcu in skrbi za izvajanje uporabniškega vmesnika ter zahteva podatke od aplikacijskega nivoja.
- Nivo aplikacijskega strežnika ali aplikacijski nivo, tudi logični nivo: srednji nivo se izvaja na aplikacijskem strežniku in skrbi za procesiranje ter izvaja poslovno logiko.
- Nivo podatkovnega strežnika ali podatkovni nivo: najnižji nivo predstavlja podatkovni strežnik, ki je sestavljen iz podatkovne baze in sistema za njeno upravljanje (SUPB). Kot tak skrbi za shranjevanje podatkov, izvajanje poizvedb ter skrbi za integriteto in varnost podatkov.



Slika 4. 6: Tronivojska arhitektura

Medtem ko pri tronivojski arhitekturi odjemalec skrbi za izvajanje uporabniškega vmesnika, je pri n-nivojski arhitekturi to prepuščeno spletnemu strežniku, s čimer je odjemalčev sistem skoraj popolnoma razbremenjen, saj skrbi le za izvajanje spletnega brskalnika, s katerim uporabniki dostopajo do uporabniškega vmesnika.

Dodatno so naše programske rešitve na aplikacijskem nivoju razdeljene na več medsebojno povezanih podnivojev, ki jih prikazuje slika 4. 7.



Slika 4. 7: Delitev aplikacijskega nivoja

Najnižji nivo oz. nivo načrtovalskih vzorcev DAO (ang. *Data Access Object*) predstavljajo objekti za dostop do podatkov oziroma t. i. DAO-ti , ki omogočajo komunikacijo med aplikacijskim in podatkovnim nivojem. V njih so implementirani dostopni mehanizmi, potrebni za delo s podatkovno bazo. Prednost uporabe objektov DAO je v tem, da je dejanska implementacija dostopa do podatkovne baze višjim nivojem nepoznana, saj v ta namen uporabljajo izključno javanske vmesnike, ki jih objekti DAO zagotavljajo. Ker se vsebina teh vmesnikov načeloma ne spreminja, je sprememba dostopnih mehanizmov do podatkovne baze za višje nivoje transparentna.

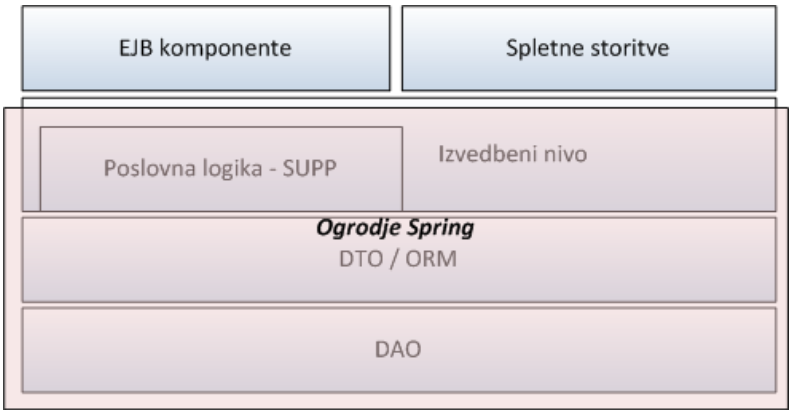
Nivo višje se nahaja nivo podatkovnih objektov. Sestavljajo ga objekti, ki temeljijo na načrtovalskem vzorcu DTO (ang. *Data Transfer Object*) in/ali objekti ORM (ang. *Object-Relational Mapping*). Objekti DTO predstavljajo ovoj za podatke (jih enkapsulirajo), ki jih vračajo objekti DAO, in se uporabljajo v izvedbenem nivoju in posredujejo predstavitvenemu nivoju. Objekti ORM se uporabljajo v primeru uporabe ogrodja Hibernate, ki olajšuje preslikavo tabel v relacijski podatkovni bazi v javanske objekte. Vsak ORM objekt predstavlja eno tabelo relacijske baze, vsi objekti skupaj pa tvorijo natančen objektni model podatkovne baze. V primeru uporabe ogrodja Hibernate objekti DAO posredujejo objekte ORM, ki se za potrebe višjih nivojev pretvorijo v ustrezne objekte DTO.

Predzadnji ali izvedbeni nivo sestavljata izvedbena in poslovna logika. Slednjo v našem primeru predstavljajo aplikacije poslovnih pravil; izvedbena logika pa vsebuje vse preostalo procesiranje. Vse potrebne podatke jim zagotavljajo objekti DTO in ORM.

Na najvišjem nivoju, ki ga lahko imenujemo tudi povezovalni nivo, so EJB komponente in spletne storitve, ki omogočajo komunikacijo med predstavitvenim in aplikacijskim nivojem. To pomeni, da sprejemajo zahteve predstavitvenega nivoja in jih posredujejo izvedbenemu nivoju. Ko se zahteve na izvedbenem nivoju obdelajo, jih posredujejo nazaj predstavitvenemu nivoju.

4. 3 Ogrodje Spring

Ogrodje Spring je odprtokodno ogrodje za izdelavo javanskih aplikacij in ga, kot je prikazano na sliki 4. 8, v naših programskih rešitvah uporabljamo skoraj na vseh podnivojih aplikacijskega nivoja.



Slika 4. 8: Ogrodje Spring na aplikacijskem nivoju

Deluje kot močno in stabilno lepilo, ki omogoča povezovanje in konfiguracijo posameznih nivojev aplikacije v eni ali več datotekah XML. Sestavljeno je iz več modulov, ki uporabnikom zagotavljajo veliko funkcionalnosti, vendar se na tem mestu ne bomo spuščali v podrobnosti ogrodja. Bolj se bomo posvetili le t. i. vstavljanju odvisnosti (ang. *Dependency Injection*).

Vstavljanje odvisnosti omogoča programskim komponentam oskrbovanje z zunanjimi odvisnostmi. Povedano z drugimi besedami: objektom, ki uporabljajo vmesnike drugih objektov (npr. vmesnike DAO), vstavljanje odvisnosti omogoča podajanje dejanske implementacije teh vmesnikov. Dejanska implementacija je tako za uporabnike vmesnikov neznana, poleg tega pa omogoča enostavno zamenjavo implementacije nekega vmesnika, saj je treba le popraviti nastavitvene datoteke XML. Za lažje razumevanje si poglejmo koncept vstavljanja odvisnosti na primeru.

Recimo, da imamo naslednji vmesnik DAO (z imenom *CustomerDao*):

```
public interface CustomerDao {
   public Customer getCustomer(int customerId);
   public void saveCustomer(Customer customer);
}
```

Pri tem razred *Customer* predstavlja objekt DTO, ki vsebuje vse podatke o stranki. Razred *CustomerDaoImpl* predstavlja implementacijo vmesnika *CustomerDao*:

```
public class CustomerDaoImpl implements CustomerDao {
   public Customer getCustomer(int customerId) {
      //implementacija metode
```

```
    }
    public void saveCustomer(Customer customer) {
       //implementacija metode
    }
}
```

Ker v razredu *CustomerLogic* potrebujemo dostop do podatkovne baze, uporabimo v ta namen vmesnik DAO:

```
public class CustomerLogic {
   private CustomerDao customerDao;

   public CustomerLogic() {
      ApplicationContext ctx = = new
ClassPathXmlApplicationContext(pot_do_konfiguracijske_datoteke);
      this.customerDao = (CustomerDao) ctx.getBean("customerDaoBean");
   }

   public void manipulateCustomerData(int customerId) {
      Customer customer = customerDao.getCustomer(customerId);
      //dodatna logika
      customerDao.saveCustomer(customer);
   }

   //dodatne metode

}
```

Kot je razvidno iz zgornje kode, vsebuje *CustomerLogic* privatno spremenljivko *customerDao* tipa *CustomerDao*. Ob inicializaciji novega objekta tipa *CustomerLogic* se dejanska implementacija spremenljivke *customerDao* prebere iz nastavitvene datoteke XML, kjer je njeno zrno definirano pod imenom *customerDaoBean.* Spodaj je prikazana definicija zrna s tem imenom:

```
<bean id="customerDaoBean" class="CustomerDaoImpl" />
```

Iz definicije je razvidno, da se zrno *customerDaoBean* sklicuje na razred C*ustomerDaoImpl*, ki predstavlja dejansko implementacijo vmesnika *CustomerDao*. Če bi želeli uporabljati drugo implementacijo tega vmesnika za dostop do podatkovne baze, bi morali samo ustrezno popraviti konfiguracijo zrna *customerDaoBean*, da bi se sklicevalo na ustrezno implementacijo.

Zrna, definirana v nastavitvenih datotekah, se vstavijo (zato tudi ime vstavljanje odvisnosti) šele po njihovi tvorbi, kar je ravno obratno od vstavljanja lastnosti ob tvorbi s konstruktorjem. Vstavljanje lastnosti tako predstavlja obliko inverzije nadzora (ang. *Inversion of Control*).

4. 4 JUnit in Unitils

Za potrebe testiranja programskih rešitev uporabljamo knjižnice Unitils, ki predstavljajo nadgradnjo knjižnic JUnit.

Knjižnice JUnit omogočajo izvajanje t. i. unit testov, ki omogočajo testiranje posameznih modulov po principu bele skrinjice. Kot taki omogočajo testiranje delovanja posameznih metod: preverjanje robnih pogojev, izvajanje negativnih in pozitivnih testov. Omogočajo tudi uporabo t. i. hlinjenih (ang. *mock*) razredov za potrebe vmesnikov DAO. To pomeni, da med izvajanjem unit testov ne potrebujemo dostopa do podatkovne baze, saj hlinjena implementacija vmesnikov DAO simulira obnašanje dejanske implementacije, ki dostopa do podatkovne baze, tako da vrača vrednosti, ki bi jih načeloma prebrali iz podatkovne baze. Unit teste izvajamo zaradi:

- testiranja pravilnega delovanja metod,
- večjega zaupanja v refaktorizacijo kode,
- izogibanja napakam pri usklajevanju kode,
- dodajanja testov v regresijske teste,
- zagotavljanja večje kakovosti implementacije metod.

Postopek testiranja z unit testi je sledeč:

- implementacija razreda,
- implementacija metode,
- implementacija mock razredov,
- izdelava unit test razreda,
- izdelava unit test metod: načrt možnih scenarijev in izdelava testnih metod.

Pri testiranju z unit testi poznamo dve vrsti testov: pozitivne in negativne teste. Prvi skušajo dokazati, da metoda ali modul delujeta po pričakovanjih. Negativni testi pa skušajo dokazati, da metoda ali modul ne delujeta tako, kot ne bi smela.

Knjižnice Unitils predstavljajo nadgradnjo knjižnic JUnit in omogočajo bolj zapleteno testiranje. Kot take jih uporabljamo za integracijske teste v sklopu integracijskega testiranja celotnih programskih rešitev. Prednosti knjižnice Unitils so:

- izboljšani ukazi za primerjavo pričakovanih in dejanskih vrednosti (v primerjavi z JUnit),
- omogoča integracijo ogrodij Hibernate in Spring,
- omogoča integracijo z različnimi podatkovnimi bazami (DB2, Oracle, MySQL),
- omogoča uporabo mock razredov,
- omogoča transparentno integracijo z JUnit3 in JUnit4.

Omenili smo že, da Unitils uporabljamo za izvajanje integracijskih testov, ki (za razliko od unit testov) testirajo integracijo med komponentami, pri čemer uporabljajo princip črne skrinjice. Pri tem se testira delovanje celotne komponente, pisanje in branje podatkov v/iz

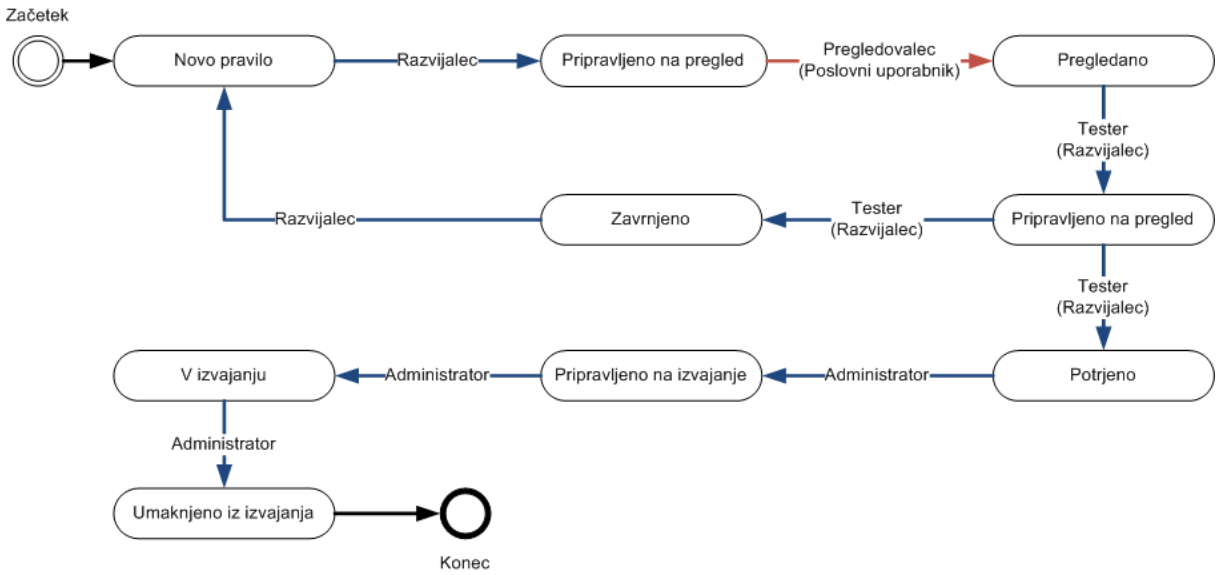
podatkovne baze ter komunikacija med komponentami. Postopek izvajanja integracijskih testov je sledeč:

- implementacija komponente,
- izdelava scenarijev,
- izdelava integracijskih testov: polnjenje testne integracijske podatkovne baze, zagon metode komponente in primerjava novih vrednosti v podatkovni bazi s pričakovanimi, izpraznitev testne integracijske podatkovne baze.

5 Obravnavani problem

5. 1 Obstoječi in želeni postopek upravljanja poslovnih pravil

Pri vzdrževanju aplikacij poslovnih pravil je sedaj večina dela na ramenih samih razvijalcev aplikacij, saj poslovnim uporabnikom ne omogočamo neposrednega vpogleda v poslovna pravila in njihovega urejanja in dodajanja. Pri samem razvoju aplikacij poslovnih pravil so slednji udeleženi pri zajemu in formalnem zapisu poslovnih pravil, ki jih nato razvijalci implementirajo in preverijo pravilnost njihovega delovanja (jih stestirajo), na koncu pa celotno aplikacijo skupaj z aplikacijo poslovnih pravil postavijo na produkcijsko okolje. Podoben postopek se uporablja pri vzdrževanju aplikacij poslovnih pravil v primeru urejanja obstoječih ali dodajanja novih poslovnih pravil, pri čemer se na produkcijsko okolje postavi le nova različica aplikacije poslovnih pravil. Obstoječi življenjski cikel poslovnih pravil prikazuje slika 5. 1, pri čemer so z rdečo puščico prikazane aktivnosti, v katerih sodelujejo poslovni uporabniki.



Slika 5. 1: Obstoječi življenjski cikel poslovnih pravil

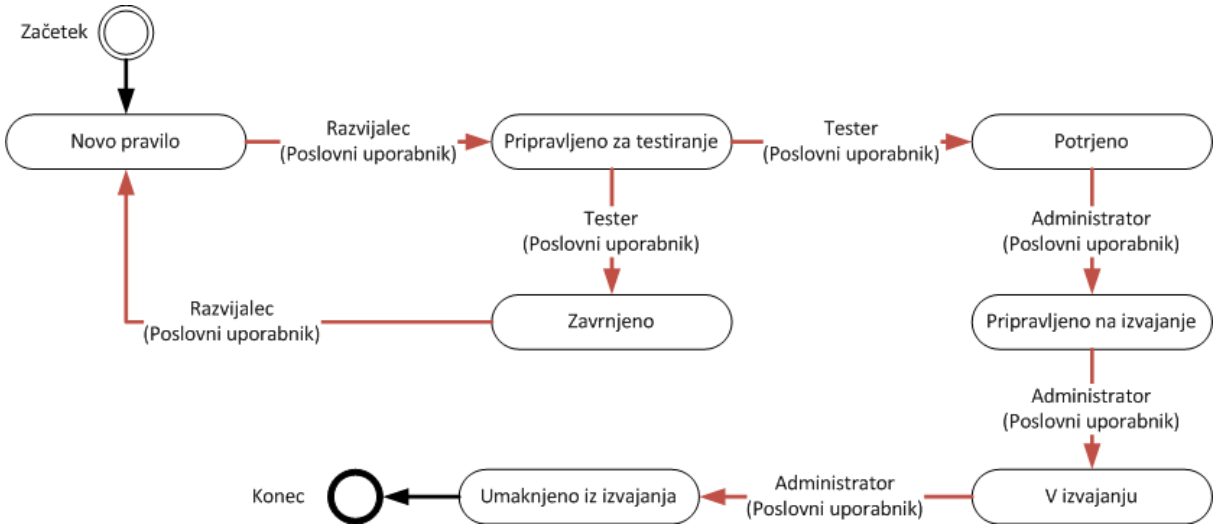
Če želijo poslovni uporabniki neko poslovno pravilo spremeniti, dodati novo pravilo ali odstraniti obstoječega, sporočijo želene spremembe razvijalcem, ki jih nato implementirajo. Poslovni uporabniki nato popravljena oz. dodana poslovna pravila pregledajo. Temu sledi testiranje poslovnih pravil, ki ga opravijo razvijalci, in postavitev nove različice aplikacije poslovnih pravil na produkcijsko okolje, kar opravi administrator.

Zaradi dejstva, da poslovni uporabniki izhajajo iz zunanjih organizacij, ki jim prodamo programske rešitve, je komunikacija med poslovnimi uporabniki in razvijalci otežena in časovno potratna. Zahvaljujoč uporabi SUPP in jeziku BAL za zapis poslovnih pravil, so

slednja zapisana v obliki, ki je zelo podobna naravnemu jeziku in posledično razumljiva tudi poslovnim uporabnikom. Njihovo urejanje in dodajanje sta zelo intuitivna in ne zahtevata veliko tehničnega znanja. V prihodnosti želimo zato poslovnim uporabnikom omogočiti neposredno delo s poslovnimi pravili in jih tako aktivno vključiti v njihovo upravljanje.

Z aktivno vključitvijo poslovnih uporabnikov v upravljanje s poslovnimi pravili želimo omogočiti njihovim organizacijam čimvečjo prilagodljivost na spremembe na tržišču in posledično konkurenčno prednost. Da bi to dosegli, ni dovolj, da omogočimo poslovnim uporabnikom samo dodajanje novih poslovnih pravil in urejanje obstoječih, temveč jim moramo omogočiti tudi samostojno testiranje poslovnih pravil in postavitev novih različic aplikacij poslovnih pravil na produkcijsko okolje.

S tem bi poslovne uporabnike vključili v celoten življenjski cikel poslovnih pravil, kar prikazuje slika 5. 2, pri čemer rdeče puščice predstavljajo aktivnosti, v katere so vključeni poslovni uporabniki.



Slika 5. 2: Želeni življenjski cikel poslovnih pravil

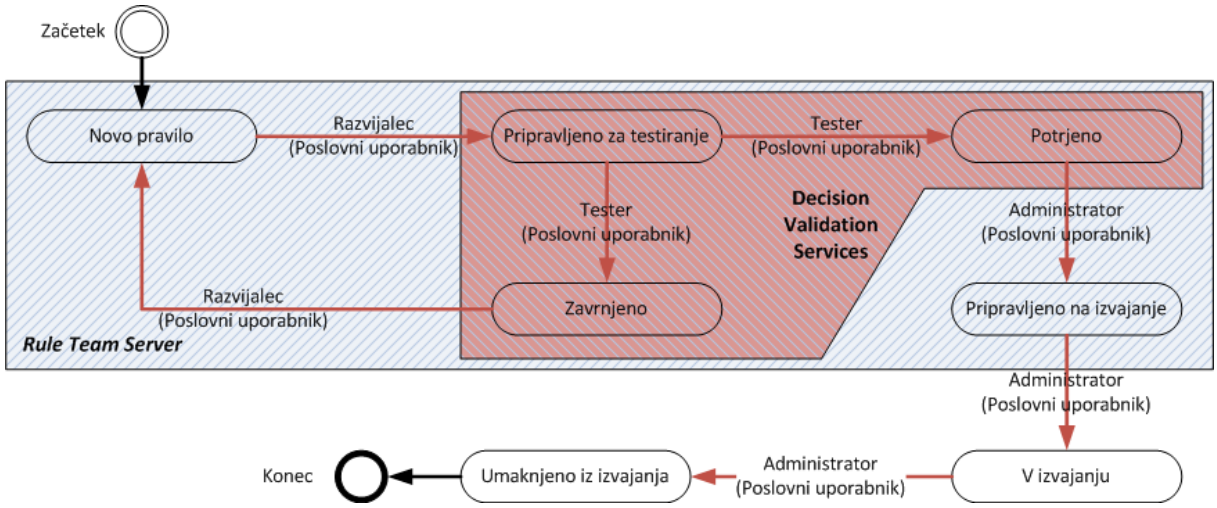
Kot je razvidno s slike, so poslovni uporabniki (v tem primeru) vključeni v celoten življenjski cikel poslovnih pravil. S tem, ko jim omogočimo razvoj poslovnih pravil, odpade potreba po pregledu pravil, saj že med samim razvojem poskrbijo za njihovo skladnost s predvidenim poslovnim obnašanjem (glej poglavje 2. 1. 1). Dodatno jim z zagotovitvijo testiranja aplikacij poslovnih pravil in njihovo postavitvijo v produkcijsko okolje omogočimo tudi brisanje neaktualnih poslovnih pravil in postavitev tako popravljene aplikacije poslovnih pravil nazaj na produkcijsko okolje.

Vprašanji, ki se nam pri tem zastavljata, sta, kakšne možnosti imamo za aktivno vključitev poslovnih uporabnikov v delo s poslovnimi pravili oz. na kakšen način jim to lahko zagotovimo ter kakšne posledice ima to na obstoječe programske rešitve. Odgovora nas čakata v nadaljevanju.

5. 2 Orodja za delo s poslovnimi pravili za poslovne uporabnike

Ena izmed možnosti, s katerimi bi poslovnim uporabnikom omogočili delo s poslovnimi pravili, bi bil razvoj lastnih programskih rešitev, s katerimi bi jim zagotovili delo s poslovnimi pravili v njihovem celotnem življenjskem ciklu. Pri tem se postavlja vprašanje smiselnosti razvoja lastnih programskih rešitev, saj za razvoj poslovnih pravil in njihovih aplikacij uporabljamo že omenjeni WebSphere ILOG JRules (v nadaljevanju ILOG). ILOG kot SUPP poleg razvojnih orodij in izvajalnega okolja vsebuje tudi orodja, ki poslovnim uporabnikom omogočajo delo s poslovnimi pravili. To so (že omenjeni) Rule Team Server, Decision Validation Services in Rule Solutions For Office.

Rule Team Server in Rule Solutions For Office omogočata poslovnim uporabnikom urejanje in dodajanje poslovnih pravil, pri čemer je možno poslovna pravila sinhronizirati med obema orodjema, kar prikazuje slika 4. 1. Dodatno Rule Team Server omogoča postavitev novih različic aplikacij poslovnih pravil na izvajalno okolje. Decision Validation Services pa poslovnim uporabnikom omogoča testiranje poslovnih pravil in simulacijo njihovih sprememb v sklopu Rule Team Serverja. Slika 5. 3 prikazuje, katere aktivnosti v življenjskem ciklu poslovnih pravil pokriva posamezno orodje, namenjeno poslovnim uporabnikom.



Slika 5. 3: Življenjski cikel poslovnih pravil in ILOGova orodja za poslovne uporabnike

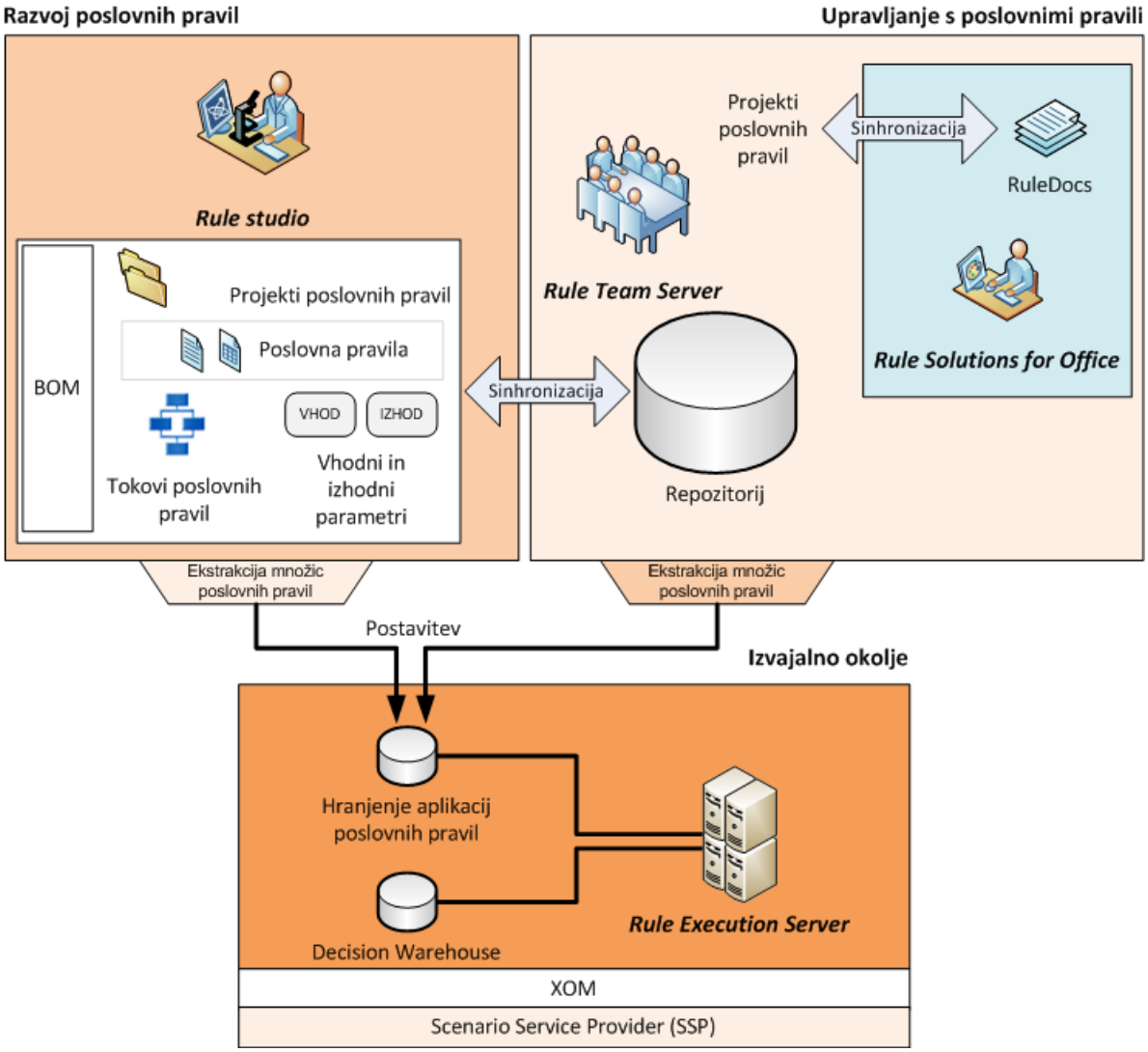
V nadaljevanju bomo podrobneje opisali obe orodji ter funkcionalnosti, ki jih nudita poslovnim uporabnikom. Prav tako si bomo pogledali možnost alternativne rešitve, kar pomeni, da poslovnim uporabnikom omogočimo takojšnje delo s poslovnimi pravili z uporabo Rule Studia.

5. 2. 1 Rule Team Server

Kot smo že omenili v poglavju 4. 1. 1. 2, predstavlja Rule Team Server (v nadaljevanju RTS) samostojni modul ILOG-a, ki omogoča poslovnim uporabnikom delo s poslovnimi pravili.

Podobno kot izvajalno okolje se ga namesti na aplikacijski strežnik, v našem primeru je to IBM WebSphere Application Server, uporabniki pa do njega dostopajo prek spletnega vmesnika. RTS je tako aplikacija tipa odjemalec-strežnik, ki prek spletnega vmesnika omogoča poslovnim uporabnikom dodajanje novih pravil, urejanje in brisanje obstoječih, njihovo organiziranje, iskanje obstoječih poslovnih pravil, postavitev novih različic aplikacij poslovnih pravil na izvajalno okolje itd.

Z uporabo RTS-ja poslovni uporabniki dostopajo do posameznih projektov poslovnih pravil, ki so shranjeni v repozitoriju. Za razvoj samih projektov so še vedno zadolženi razvijalci. Razvoj projektov poslovnih pravil poteka v Rule Studiu, ki poleg shranjevanja aplikacij pravil v repozitorij omogoča tudi njihovo naknadno sinhronizacijo med repozitorijem in Rule Studiem. Na ta način imajo razvijalci v vsakem trenutku dostop do trenutno aktualnih različic poslovnih pravil in drugih elementov v posameznih projektih. Dodatno omogoča RTS poslovnim uporabnikom kreiranje in postavitev aplikacij poslovnih pravil na izvajalno okolje, to je Rule Execution Server. Opisano povezavo med posameznimi orodji prikazuje slika 5. 4.

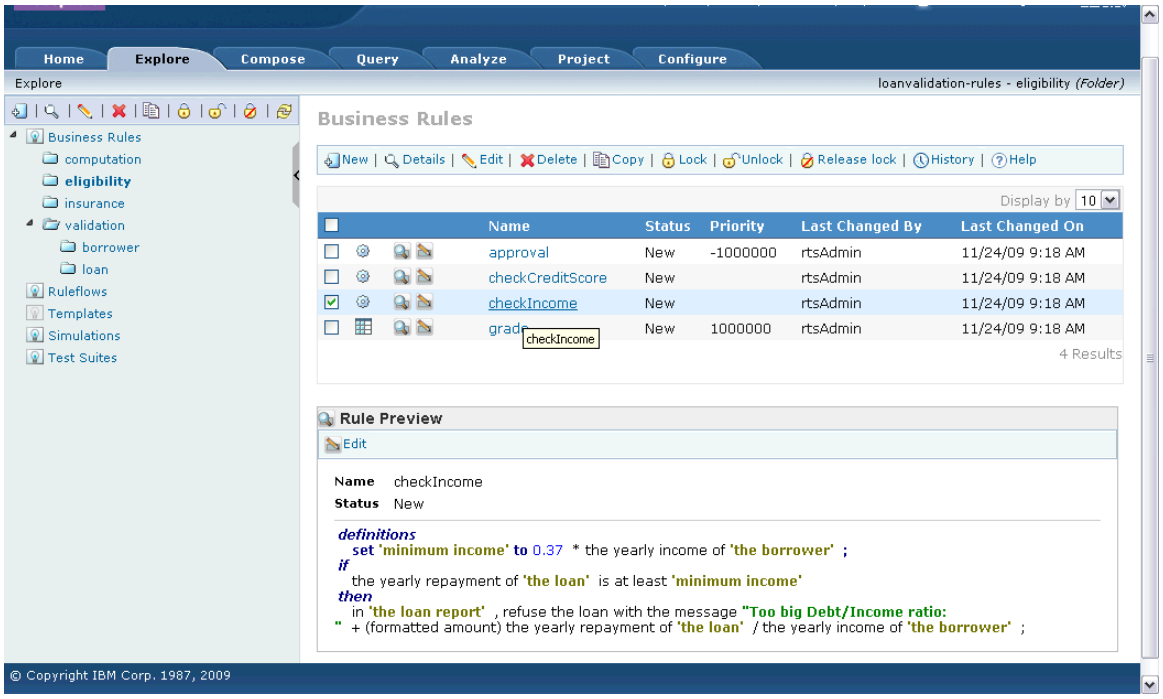


Slika 5. 4: Povezava Rule Studia, Rule Team Serverja in Rule Execution Serverja (povzeto po [21])

5. 2. 1. 1 Funkcionalnosti

Vse funkcionalnosti RTS-ja so poslovnim uporabnikom dostopne prek spletnega vmesnika. Spletni vmesnik je razdeljen na posamezne zavihke, ki vsebujejo posamezne sklope funkcionalnosti:

- **Zavihek *Home*** jim omogoča izbiro med tistimi projekti poslovnih pravil v repozitoriju, za katere imajo dodeljene pravice za delo z njimi.
- **Zavihek *Explore*** jim omogoča navigacijo po izbranem projektu poslovnih pravil in njegov podroben pregled, kar prikazuje slika 5. 5.

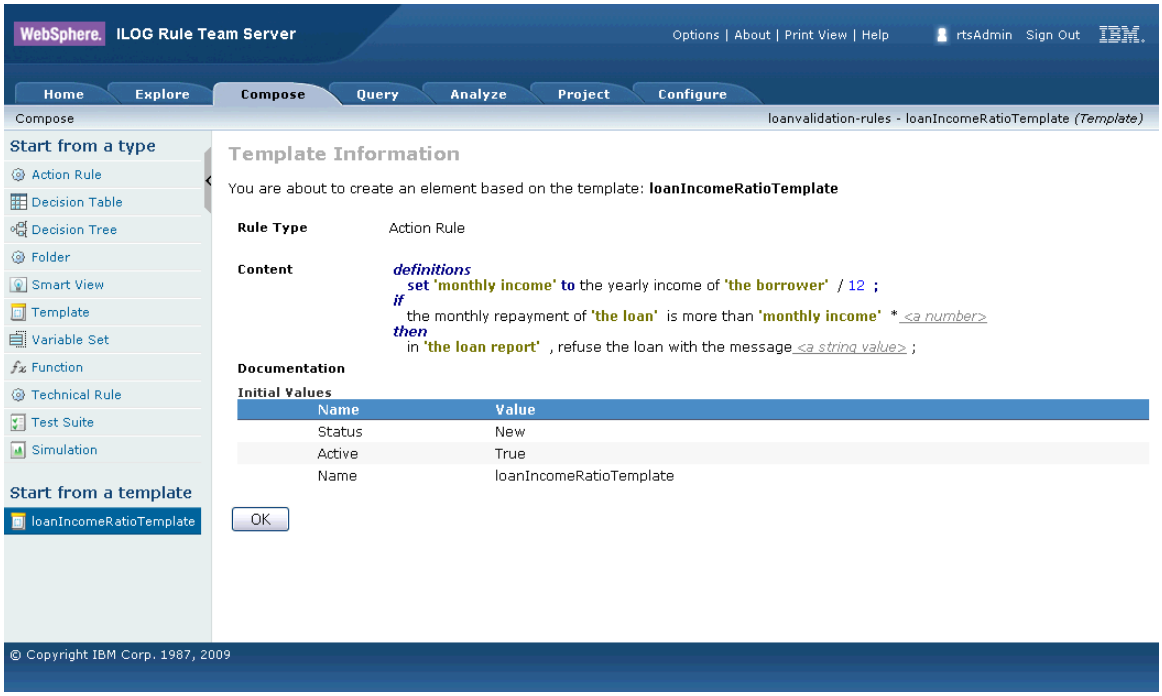


Slika 5. 5: Navigacija po izbranem projektu v zavihku Explore

Uporabniki dostopajo do elementov projekta prek menija na levi strani, pri čemer so elementi urejeni v naslednje osnovne skupine: poslovna pravila (*Business Rules*), v katerih so prikazane vse vrste poslovnih in tehničnih pravil (glej poglavje 4. 1. 2. 4), tokovi poslovnih pravil (*Ruleflows*), predloge poslovnih pravil (*Templates*) ter simulacije (*Simulations*) in zbirke testov (*Test Suites*) za potrebe testiranja projektov poslovnih pravil. Posamezne skupine se lahko nadalje delijo na podmape, pri čemer posnemajo strukturo projektov v Rule Studiu. Ob kliku na želeno mapo se uporabniku na desni strani prikažejo vsi elementi v tej mapi, ob kliku na posamezni element pa se prikažejo njegove podrobnosti.

- **Zavihek *Compose*** jim omogoča kreiranje novih elementov v sklopu izbranega projekta. Pri tem je omogočeno kreiranje praktično vseh elementov, ki so vsebovani v projektih poslovnih pravil. Poleg kreiranja poslovnih in tehničnih pravil, odločitvenih dreves in tabel, je omogočeno kreiranje novih map, v katerih so vsebovana poslovna pravila, kreirajo lahko tudi nove predloge za poslovna pravila, množice spremenljivk, ki jih uporabljajo, in tudi funkcije, če jih potrebujejo. V kolikor je nameščen modul

Decision Validation Services za potrebe testiranja, imajo tudi možnost kreiranja novih testnih zbirk in simulacij.



Slika 5. 6: Predloga poslovnega pravila v Rule Team Serverju

Za večino poslovnih uporabnikov je še posebej zanimiva možnost kreiranja podobnih poslovnih pravil neposredno iz predlog poslovnih pravil, kar prikazuje slika 5. 6. V tem primeru jim ni treba pisati celotnega poslovnega pravila od začetka, temveč v predlogo preprosto vnesejo potrebne vrednosti ter tako dopolnjeno predlogo ustrezno poimenujejo in shranijo v želeno mapo.

- **Zavihek *Query*** jim omogoča iskanje poslovnih pravil in drugih elementov v sklopu izbranega projekta, pri čemer imajo na voljo ogromno kriterijev za iskanje. Poslovna pravila lahko iščejo po njihovih lastnostih, npr. datumu zadnje spremembe, datumu začetka veljavnosti, imenu, statusu itd. Dodatno lahko poslovna pravila iščejo glede na njihovo vsebino, prav tako pa lahko iščejo po dokumentaciji poslovnih pravil.
  Iskanje po elementih projekta je še posebej uporabno, ko imajo poslovni uporabniki v sklopu izbranega projekta opraviti z večjo količino poslovnih pravil in ne vedo točno, kje v projektu se neko poslovno pravilo nahaja.
  Poizvedbe lahko tudi shranijo, s čimer so na voljo tudi drugim uporabnikom. Shranjene poizvedbe je mogoče uporabiti tudi za ekstrakcijo množice poslovnih pravil. Če pri ekstrakciji uporabimo neko poizvedbo, se v množico poslovnih pravil izvozijo le tista pravila, ki so določena s samo poizvedbo.
- **Zavihek *Analyze*** sestavljajo trije sklopi, ki uporabnikom omogočajo analiziranje izbranega projekta, izdelavo poročil o projektu, kreiranje predlog za testne scenarije ter pregled zbirk testov in scenarijev, ki se trenutno izvajajo na RTS-ju.
  Analiza projekta omogoča odkrivanje pomanjkljivosti v posameznih poslovnih pravilih ter medsebojni vpliv poslovnih pravil. Kot taka pokaže na:

  - poslovna pravila, ki niso vsebovana v tokovih poslovnih pravil,
  - poslovna pravila, ki se nikoli ne izvedejo (predpogoji niso nikoli izpolnjeni),
  - poslovna pravila, ki kršijo omejitve v podatkih,
  - poslovna pravila, ki vsebujejo enakovredne pogoje, vendar imajo različne akcije,
  - poslovna pravila, ki so enakovredna (imajo identične pogoje in akcije),
  - poslovna pravila, ki so v konfliktu (vsebujejo pogoje, ki se ne izključujejo, obenem pa vsebujejo akcije, ki se med seboj izključujejo),
  - poslovna pravila, ki napravijo druga poslovna pravila odvečna (kadar imata dve pravili isto akcijo, pogoji enega pravila pa so vsebovani v pogojih drugega pravila).

  V sklopu izdelave poročil je uporabnikom omogočena izdelava podrobnih poročil o projektu v html obliki ali izdelava poročil o obstoječih poslovnih pravilih in aktivnostih tokov poslovnih pravil v obliki preglednice v Excelu. Podrobna poročila o projektu vsebujejo grafični prikaz in opis posameznih poslovnih pravil (akcijskih in tehničnih pravil, odločitvenih tabel in dreves) ter tokov poslovnih pravil. Poročila o obstoječih pravilih in aktivnostih tokov pa vsebujejo samo seznam in lastnosti vseh obstoječih poslovnih pravil ter aktivnosti v sklopu tokov poslovnih pravil.
- **Zavihek *Project*** vsebuje napredne možnosti in omogoča upravljanje izbranega projekta. Možnosti so razdeljene v štiri sklope: upravljanje nastavitev izbranega projekta, pregled BOM-a, generiranje množic poslovnih pravil ter delo z Rule Solutions For Office.

  V sklopu slednjega je omogočen izvoz pravil v RuleDocs, to je dokumente formata Microsoft Office Word za poslovna pravila ter Microsoft Office Excel za odločitvene tabele. Če imajo poslovni uporabniki nameščen Rule Solutions For Office, lahko pravila v omenjenih programih urejajo in nato uvozijo nazaj v Rule Team Server in integrirajo z obstoječimi poslovnimi pravili.
- **Zavihek *Configure*** prav tako vsebuje napredne možnosti in omogoča upravljanje RTS-ja. Možnosti so razdeljene v tri sklope: postavitev, varnost in administracija.
  V sklopu postavitve omogoča upravljanje aplikacij poslovnih pravil, njihovo kreiranje in postavitev na izvajalno okolje ter upravljanje z izvajalnimi okolji (pregled in brisanje povezav z obstoječimi izvajalnimi okolji ter dodajanje povezav na nova).
  V sklopu varnosti omogoča upravljanje varnostnih nastavitev izbranega projekta in urejanje pravic uporabnikov za delo s projekti.
  V sklopu administracije omogoča diagnostiko RTS-ja in spreminjanje njegovih nastavitev ter brisanje izbranega projekta poslovnih pravil iz repozitorija.

Vse navedene funkcionalnosti niso na voljo vsem uporabnikom RTS-ja, temveč le tistim, ki imajo ustrezne pravice za delo. V ta namen morajo uporabniki pri prijavi v RTS vnesti svoje uporabniško ime in geslo, na podlagi katerega jih RTS identificira in omogoči ustrezne funkcionalnosti.

5. 2. 1. 2 Pravice uporabnikov in varnost

Pravice uporabnikov za delo v RTS-ju so definirane na nivoju skupin, ki jim uporabniki pripadajo. V osnovi so vsi uporabniki razdeljeni v tri skupine:

- **rtsUser:** navadni uporabniki.
- **rtsConfigManager:** dodatno imajo na voljo upravljanje konfiguracije.
- **rtsAdministrator**: skrbniki Rule Team Serverja.

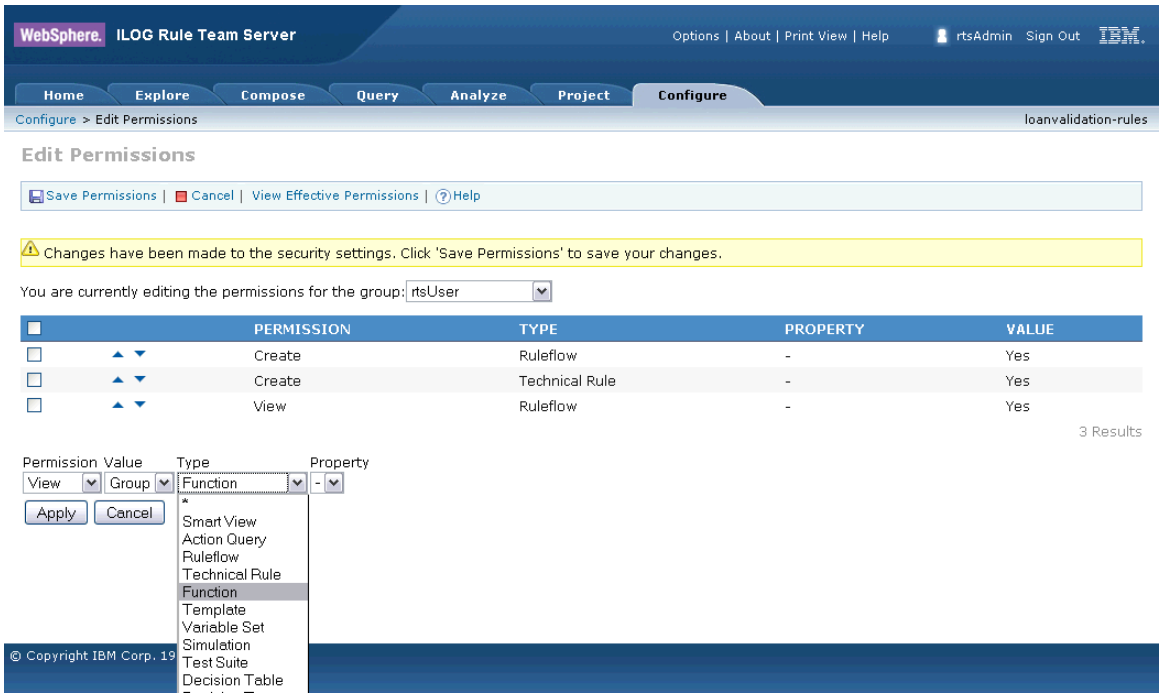
Uporabniki iz skupine *rtsUser* imajo na voljo zavihke *Explore, Compose, Query, Analyze* in nekatere možnosti v sklopu zavihka *Project.* Uporabniki iz skupine *rtsConfigManager* imajo na voljo iste zavihke kot uporabniki iz skupine *rtsUser*, dodatno pa imajo na voljo še več možnosti v sklopu zavihka *Project*, ki zadevajo nastavitve projekta. Uporabniki iz skupine *rtsAdministrator* imajo na voljo vse možnosti v vseh zavihkih.

Če osnovne tri skupine ne zadostujejo, je možno definirati dodatne skupine in jim dodeliti ustrezne pravice za delo. Ustrezne pravice jim dodelijo administratorji, to so uporabniki, ki pripadajo skupini *rtsAdministrators*. Poleg tega lahko administratorji spreminjajo pravice preostalih dveh osnovnih skupin.

Pravice nad elementi lahko administratorji določijo zelo podrobno. V sklopu posamezne skupine za vsak element projektov poslovnih pravil omogočijo ustrezne pravice, prav tako lahko pravice onemogočijo. Možne pravice za posamezne elemente projektov poslovnih pravil so naslednje:

- pregled (*View*),
- kreiranje (*Create*),
- posodabljanje (*Update*),
- brisanje (*Delete*).

Posamezna pravica nad elementom je lahko uporabnikom omogočena (vrednost *Yes*), onemogočena (vrednost *No*) ali pa je določena s skupino, ki ji uporabnik pripada (vrednost *Group*). Posameznim elementom projektov poslovnih pravil je namreč mogoče določiti, kateri skupini uporabnikov RTS-ja pripadajo. Tako lahko do njih dostopa in jih ureja le točno določena skupina uporabnikov. Slika 5. 7 prikazuje primer nastavljanja pravic za skupino *rtsUser*.



Slika 5. 7: Nastavljanje pravic uporabniški skupini

Projekti poslovnih pravil v repozitoriju so v osnovi dostopni vsem uporabnikom RTS-ja, ki imajo nad projekti vse pravice za delo. Da bi uveljavili pravice, določene znotraj posameznih uporabniških skupin, morajo administratorji za vsak projekt posebej omogočiti varnost. Istočasno, ko omogočijo varnost na projektu, izberejo tudi skupine uporabnikov, ki lahko do projekta dostopajo. S tem je projekt poslovnih pravil viden izključno izbranim skupinam uporabnikov, ki imajo nad projektom le pravice, določene z uporabniško skupino, ki ji pripadajo.

5. 2. 2 Decision Validation Services

Kot smo omenili v poglavju 4. 1. 1. 3, je modul Decision Validation Services (v nadaljevanju DVS) namenjen testiranju in simulaciji množic poslovnih pravil. V ta namen ga lahko uporabljajo razvijalci in poslovni uporabniki. Ob namestitvi se namreč DVS integrira v:

- Rule Studio: dodatni vtičniki za Eclipse omogočijo razvijalcem razvoj elementov DVS in delo z njimi, omogočeno pa jim je tudi lokalno poganjanje testov in simulacij.
- Rule Team Server: s tem je poslovnim uporabnikom omogočeno poganjanje testov in simulacij ter posledično verifikacija poslovnih pravil.
- Rule Execution Server: Scenario Service Provider omogoča izvajanje testov in simulacij, dodatno pa je omogočena revizija sledi izvajanj (ang. *execution* traces) testov in simulacij, ki se shranjujejo v Decision Warehouse v sklopu RES.

5. 2. 2. 1 Scenario Service Provider

Scenario Service Provider (v nadaljevanju SSP) je komponenta DVS, ki omogoča izvajanje testnih scenarijev in simulacij na izvajalnem okolju. Pri tem deluje popolnoma ločeno od izvajalnega okolja, saj ne testira aplikacij poslovnih pravil, postavljenih na izvajalnem okolju, temveč testira izključno množice poslovnih pravil. Slednje mu posredujemo skupaj z vhodnimi podatki (testnimi scenariji) in preostalimi podatki, potrebnimi za testiranje. Primer testiranja z RTS prikazuje slika 5. 12 v poglavju 5. 2. 2. 5. Da bi se neka množica poslovnih pravil lahko izvedla, mora imeti SSP dostop do XOM-a, nad katerim je definiran BOM podane množice poslovnih pravil. Razvijalci morajo tako omogočiti SSP-ju dostop do XOM-ov vseh projektov poslovnih pravil, ki se bodo testirali z uporabo RTS-ja. Za razliko od RTS-ja, kjer se testi izvajajo na izvajalnem okolju, razvijalci v Rule Studiu teste izvajajo lokalno, kjer imajo na voljo vse potrebne projekte in izvajalno okolje.

Po končanem izvajanju množice poslovnih pravil SSP posreduje nazaj vse podrobnosti izvajanja: seznam sproženih tokov poslovnih pravil in njihovih aktivnosti, seznam sproženih poslovnih pravil, trajanje izvajanja itd.

5. 2. 2. 2 Decision Warehouse

Decision Warehouse omogoča shranjevanje sledi izvajanj množic poslovnih pravil v podatkovno bazo v sklopu izvajalnega okolja. Sled posameznega izvajanja vsebuje podatke o tem, kako je do neke odločitve, tj. rezultata izvajanja poslovnih pravil, prišlo. Kot taka zabeleži podatke o: toku poslovnih pravil, ki se je izvajal; poti znotraj toka, ki je privedla do odločitve, tj. katere aktivnosti v sklopu toka poslovnih pravil so se izvedle; in poslovnih pravilih, ki so se sprožila med izvajanjem. Vse te podrobnosti omogočajo uporabnikom natančno rekonstrukcijo poteka izvajanja poslovnih pravil za neko odločitev.

5. 2. 2. 3 Osnovni pojmi

Preden si pogledamo postopek testiranja v DVS, se bomo seznanili z osnovnimi pojmi, s katerimi se srečamo pri testiranju poslovnih pravil z DVS. Slednji omogoča izvajanje poslovnih pravil za tipične primere uporabe za potrebe:

- **Testiranja**: omogoča verifikacijo pravilnosti poslovnih pravil s primerjavo pričakovanih rezultatov z dejanskimi rezultati izvajanja poslovnih pravil.
- **Simulacije**: omogoča oceno vpliva sprememb poslovnih pravil.

Testiranje z DVS lahko izvajajo poslovni uporabniki v RTS-ju, pa tudi razvijalci v Rule Studiu, kjer imajo na voljo dodatne funkcionalnosti za potrebe razhroščevanja. Rule Team Server razlikuje med testi in simulacijami, ki so poslovnim uporabnikom razvidni kot elementi projektov poslovnih pravil, in sicer kot *Test Suite* in *Simulation* (glej sliko 5. 5). Rule Studio za razliko od RTS-ja ne razlikuje med testi in simulacijami, lahko pa razvijalci v njem kreirajo ključne kazalnike uspeha (KPI-je) za potrebe izvajanja simulacij v RTS-ju.

Pojmi, s katerimi se srečamo pri testiranju v DVS, so:

- scenariji,
- testiranje,
- simulacije,
- poročila.

Scenariji

Scenariji predstavljajo resnične ali namišljene primere uporabe, ki jih uporabljamo za verifikacijo izvajanja poslovnih pravil. Vsak scenarij vsebuje vse podatke, ki so potrebni za izvedbo poslovnih pravil. Nek scenarij lahko tako enačimo s testnim primerom, podatke, ki jih vsebuje, pa lahko enačimo z vhodnimi podatki za ta testni primer (glej poglavje 3. 2). V primeru prošnje za kredit bi kot vhodne podatke za izračun kreditne sposobnosti potrebovali podatke o stranki in želenem kreditu. Podatki o stranki vsebujejo ime stranke, kreditno zmožnost in letni prihodek, podatki o kreditu pa dobo odplačila, znesek kredita in letno obrestno mero.

Priporočeni format za vhodne podatke so preglednice formata Microsoft Excel, pri čemer je priporočena uporaba Microsoft Office Excel 2003. V tem formatu vsaka vrstica predstavlja en scenarij, stolpci pa predstavljajo posamezne vrednosti podatkov.

Za zgoraj opisani primer podatkov (za prošnjo za kredit) prikazuje slika 5. 8 preglednico v Excelu, ki vsebuje dva scenarija, za katera želimo testirati delovanje poslovnih pravil:

- *Big Loan*, s katerim želimo testirati delovanje poslovnih pravil v primeru visokega zneska kredita.
- *Small Loan*, s katerim želimo testirati delovanje poslovnih pravil v primeru nizkega zneska kredita.

5		the borrower				the loan		
6	Scenario ID	first name	last name	credit score	yearly income	duration	amount	rate
9	Big Loan	John	Smith	600	80000	24	500000	5
10	Small Loan	John	Smith	600	80000	24	25000	5

Scenarios HELP

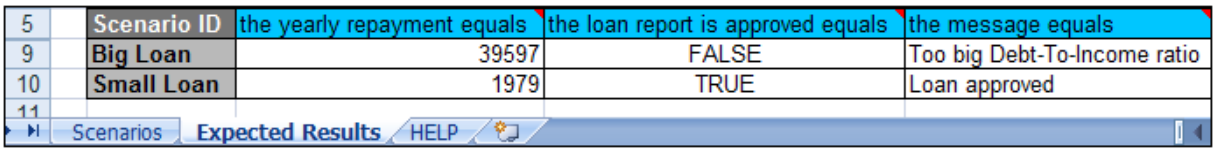
Slika 5. 8: Preglednica v Excelu z dvema testnima scenarijema

Testiranje

Testiranje omogoča verifikacijo pravilnosti izvajanja poslovnih pravil s primerjavo pričakovanih rezultatov izvajanja scenarijev z dejanskimi rezultati izvajanja.

Če za podajanje podatkov o posameznih scenarijih uporabljamo preglednice v Excelu, se preglednica s pričakovanimi rezultati nahaja v isti datoteki kot podatki o scenarijih, vendar v novem delovnem listu (list *Expected Results* na sliki 5. 9). Struktura podatkov je podobna kot pri vhodnih podatkih za scenarije: vrstice predstavljajo posamezne scenarije, stolpci pa

posamezne pričakovane vrednosti. Slika 5. 9 prikazuje primer pričakovanih rezultatov za scenarije na sliki 5. 8.



5	Scenario ID	the yearly repayment equals	the loan report is approved equals	the message equals
9	Big Loan	39597	FALSE	Too big Debt-To-Income ratio
10	Small Loan	1979	TRUE	Loan approved
11				

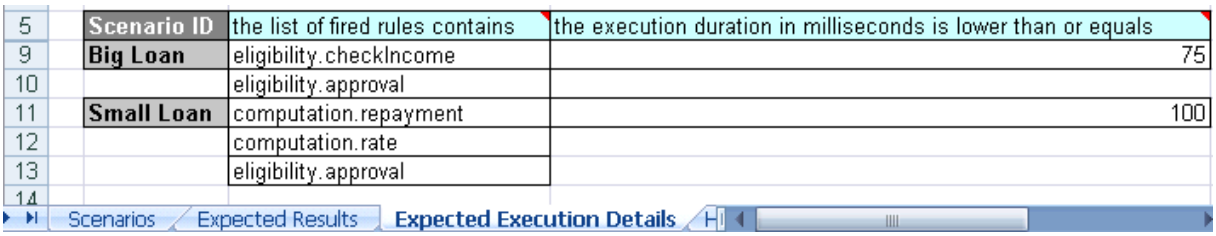
Scenarios | Expected Results | HELP

Slika 5. 9: Preglednica s pričakovanimi rezultati za testne scenarije

V primeru, prikazanem na sliki 5. 9, želimo preveriti tri različne vidike prošnje za kredit oz. tri izhodne podatke:

- Kakšen bo letni znesek plačila kredita (stolpec *the yearly repayment equals*).
- Ali bo kredit odobren (stolpec *the loan report is approved equals*).
- Kakšno sporočilo bo vsebovano v zahtevku za kredit (stolpec *the message equals*).

Poleg pričakovanih rezultatov lahko testi vsebujejo tudi delovni list s preglednico o pričakovanih podrobnostih izvajanja (ang. *Expected Execution Results*), ki vsebuje bolj tehnične podrobnosti izvajanja poslovnih pravil. To so npr. seznam poslovnih pravil, sproženih med izvajanjem scenarija, ali čas trajanja izvajanja posameznega scenarija. Slika 5. 10 prikazuje preglednico s pričakovanimi podrobnostmi izvajanja za scenarije s slike 5. 8.



5	Scenario ID	the list of fired rules contains	the execution duration in milliseconds is lower than or equals
9	Big Loan	eligibility.checkIncome	75
10		eligibility.approval	
11	Small Loan	computation.repayment	100
12		computation.rate	
13		eligibility.approval	
14			

Scenarios | Expected Results | Expected Execution Details

Slika 5. 10: Preglednica s pričakovanimi podrobnostmi izvajanja testnih scenarijev

Seznam podanih poslovnih pravil se preverja v podanem vrstnem redu, kar je še posebej primerno za preverjanje, ali se poslovna pravila prožijo v pričakovanem vrstnem redu. Če se dejanski vrstni red proženja poslovnih pravil za scenarij ne ujema s pričakovanim, se test izvede neuspešno, kar je razvidno s poročila o izvajanju testnega scenarija (glej sliko 5. 11).

Simulacije

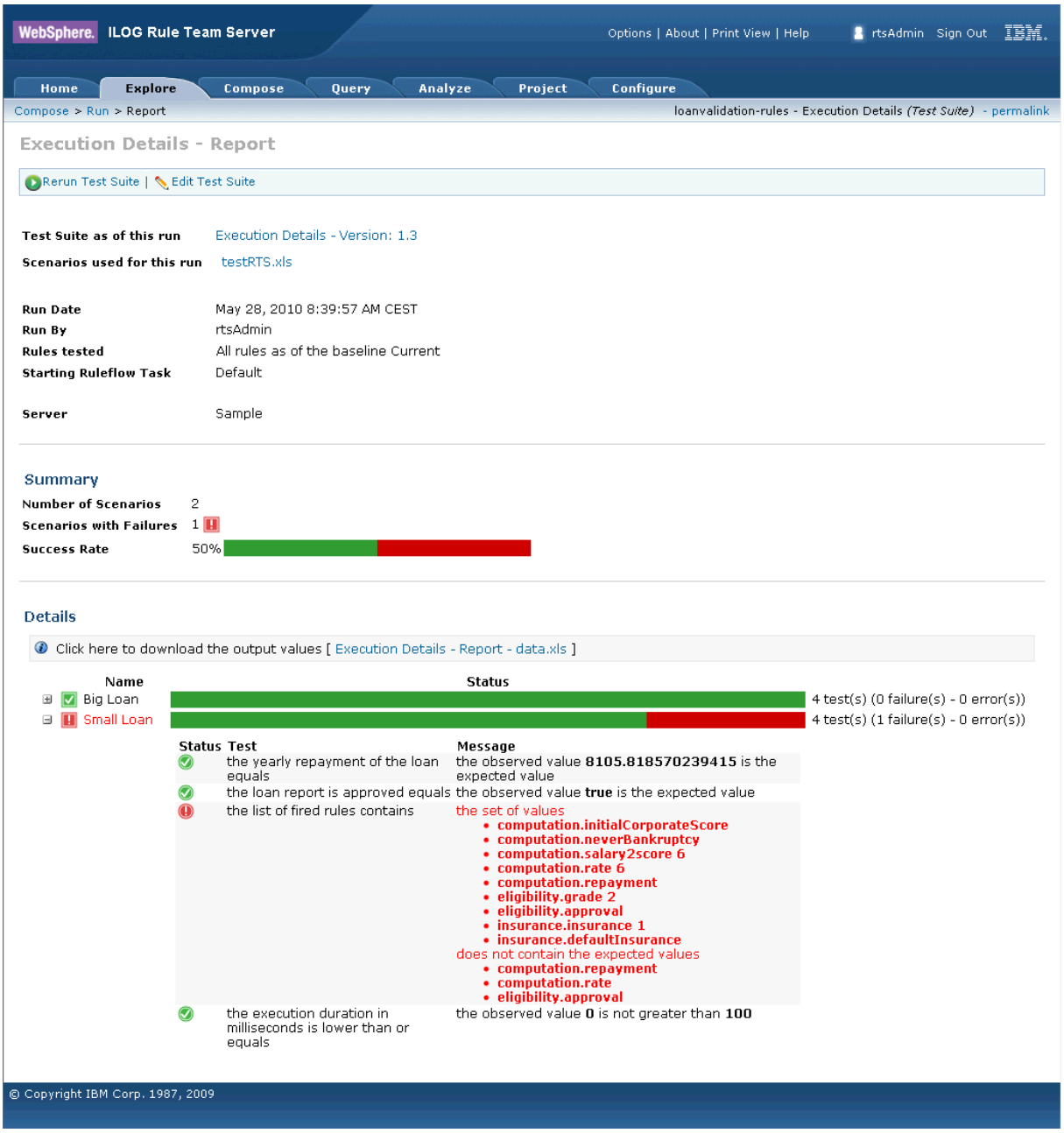
Simulacije omogočajo izvajanje „kaj-če" analiz, s katerimi poslovni uporabniki ocenijo, kakšen vpliv na organizacijo bi imele spremembe poslovnih pravil. Podatki, potrebni za izvajanje simulacij, se ujemajo s podatki posameznih scenarijev in največkrat združujejo podatke več scenarijev ali celo zgodovinske podatke o poslovanju.

Po končani simulaciji je poslovnim uporabnikom na voljo poročilo, ki prikazuje interpretacijo rezultatov na podlagi podanih ključnih kazalnikov uspeha. Ključne kazalnike uspeha

definirajo razvijalci v Rule Studiu, in jih nato objavijo na RTS-ju, kjer so dostopni poslovnim uporabnikom pri poganjanju simulacij.

Poročila

Ob izvajanju testov in simulacij v Rule Team Serverju se v izvajanje SSP-ju pošljejo: množica poslovnih pravil, ki jih želimo testirati, ter scenariji z vhodnimi podatki in pričakovani rezultati. SSP vrne Rule Team Serverju vse podatke, ustvarjene med izvajanjem. Poročila skrbijo za prikaz pridobljenih podatkov v obliki, primerni za poslovne uporabnike.



Slika 5. 11: Poročilo o izvajanju testnih scenarijev

Poročila so sestavljena iz več delov. Poleg splošnih podatkov o izvajanju, ki se nahajajo na vrhu poročil, se v sklopu *Summary* nahaja povzetek o izvajanju testov, to je število izvedenih scenarijev ter njihova uspešnost. V sklopu *Details* pa se nahajajo podrobni rezultati testov vsakega scenarija. Rezultat testiranja scenarija je lahko:

- Uspešen (*Successful*): pričakovani rezultati se ujemajo z dejanskimi rezultati izvajanja.
- Neuspešen (*Failure*): pričakovani rezultati se ne ujemajo z dejanskimi rezultati izvajanja.
- Napaka (*Error*): kadar se scenariji ne morejo izvesti, na primer zaradi napačnih vhodnih podatkov (napačno formatirane preglednice s scenariji) ali napake med izvajanjem.

5. 2. 2. 4 Možnosti podajanja vhodnih podatkov

Do sedaj smo kot edino možnost za podajanje scenarijev omenili preglednice v Excelu, vendar obstajajo tudi drugi načini, s katerimi lahko zagotovimo scenarije za testiranje. Podajanje scenarijev v obliki preglednic v Excelu je primerno za projekte poslovnih pravil, ki kot vhodne podatke pričakujejo razmeroma preprost in majhen objektni model ter pri testiranju uporabljajo do nekaj tisoč testnih scenarijev. V primeru kompleksnejših objektnih modelov ali večjega števila testnih scenarijev je mogoče predloge preglednic prikrojiti ali uporabiti drug format za vhodne podatke (datoteke XML, podatkovna baza), kar storijo razvijalci v Rule Studiu.

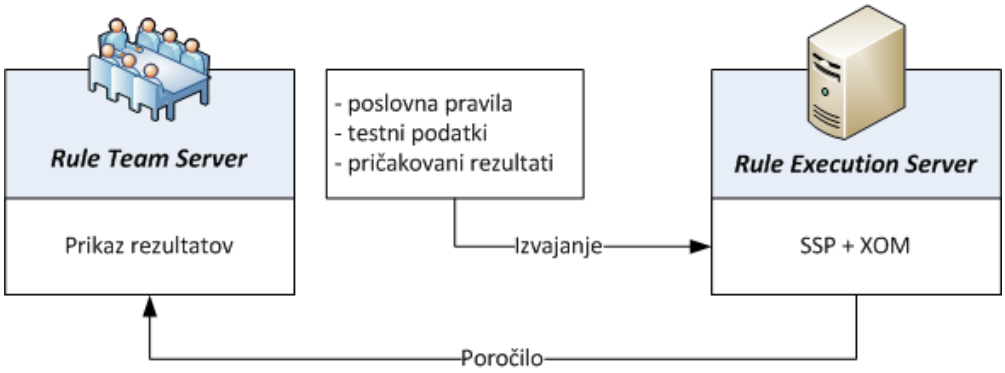
Namesto da bi morali poslovni uporabniki pri definiranju scenarijev na roke vnašati vse potrebne podatke v Excelove preglednice, jim razvijalci predhodno pripravijo ustrezne predloge s podatki v Rule Studiu, ki so jim nato na voljo v RTS-ju. Pri kreiranju predlog s podatki lahko razvijalci uporabijo DVS-jev vmesnik API, s katerim preglednice predhodno napolnijo s podatki, ki se nahajajo v podatkovni bazi. Poslovni uporabniki nato spreminjajo samo vrednosti tistih podatkov, ki so relevantni za posamezne testne scenarije, kar jim omogoča fokusiranje na samo definiranje testnih scenarijev.

Druga možnost je prikrojitev predlog v Excelu na tak način, da zahtevajo vnos samo minimalne količine podatkov, potrebnih za izvajanje testnih scenarijev. Podatke, ki za poslovne uporabnike niso zanimivi oz. se ne spreminjajo pri različnih scenarijih, zagotovijo razvijalci. Poslovni uporabniki morajo tako zagotoviti le minimalno količino podatkov, potrebnih za posamezne testne scenarije.

Tretja možnost je, da se za podajanje podatkov posameznih scenarijev namesto Excelovih preglednic uporabljajo podatki, ki so shranjeni v podatkovni bazi. Pri tem SSP dostopa izključno do podatkov v podatkovni bazi. V ta namen je treba SSP ustrezno prikrojiti, da bo lahko dostopal do podatkovne baze, in mu tako omogočiti dostop do podatkov posameznih testnih scenarijev. Slabost te možnosti je v tem, da so scenariji predhodno definirani v podatkovni bazi in poslovni uporabniki nimajo vpliva nanje oz. ne morejo prilagajati vrednosti določenih podatkov za posamezne teste.

5. 2. 2. 5 Postopek testiranja

Da bi izvedli testni scenarij, poslovni uporabniki ustvarijo nov testni nabor, v katerem določijo množico poslovnih pravil, ki jih želijo testirati, ter vir scenarijev (Excelova datoteka, podatkovna baza). Tako definiran testni nabor pošlje RTS v izvajanje na SSP. Ko se testni scenarij izvede, pošlje SSP nazaj v RTS podrobnosti izvajanja, ki jih slednji prikaže v obliki poročila (glej sliko 5. 11). V poročilu se primerjajo pričakovani rezultati z dejanskimi rezultati izvajanja, od njihovega (ne)ujemanja pa je odvisen rezultat testnega scenarija. Grafični prikaz postopka testiranja prikazuje slika 5. 12.



Slika 5. 12: Postopek testiranja v Rule Team Serverju

Ko poslovni uporabniki pregledajo poročilo, lahko ustrezno popravijo poslovna pravila ali spremenijo podatke testnega scenarija ter ponovno zaženejo testni nabor. Proces lahko ponavljajo, dokler niso zadovoljni z rezultati. Vsako izvajanje testnega nabora se shrani v repozitorij RTS-ja, s čimer je omogočena primerjava izvajanj med seboj.

Če poslovni uporabniki pri izvajanju naletijo na napako, ki je ne morejo oz. ne znajo odpraviti, pošljejo testni nabor v obliki arhiva razvijalcem. Razvijalci arhiv uporabijo za testiranje v Rule Studiu, kjer imajo na voljo orodja za razhroščevanje ki jim nudijo več nadzora nad izvajalnim okoljem in lahko izvajanje poslovnih pravil podrobneje analizirajo.

5. 2. 2. 6 Delta testiranje

Čeprav lahko na prvi pogled iz opisa podajanja testnih naborov (glej prejšnje poglavje) zaključimo, da DVS omogoča le testiranje z uporabo običajnega postopka testiranja (glej poglavje 3. 2), pri čemer pred testiranjem nove različice množice poslovnih pravil poslovni uporabniki sami popravijo pričakovane rezultate, to ni tako. DVS namreč omogoča testiranje množic poslovnih pravil tako po običajnem postopku testiranja kot z uporabo prilagojenega delta testiranja, pri čemer se pri delta testiranju branilna in izzivalna množica ne izvajata sočasno, ampak je dovolj, da testiramo samo izzivalno množico, to je množico spremenjenih poslovnih pravil, s čimer odpade potreba po dveh testnih izvajalnih okoljih, kot ju predvideva delta testiranje (glej poglavje 3. 2. 1).

Da bi spremenjeno množico poslovnih pravil lahko testirali po postopku delta testiranja, moramo v testnem naboru, s katerim testiramo spremenjeno množico pravil, kot pričakovane rezultate podati dejanske rezultate izvajanja branilne množice poslovnih pravil, to je množice trenutno veljavnih pravil, ki se izvaja na produkcijskem okolju in je ustrezno stestirana. Če nad spremenjeno množico poslovnih pravil izvajamo kvalitativno delta testiranje, se morajo dejanski rezultati testiranja ujemati s pričakovanimi rezultati. Če nad spremenjeno množico poslovnih pravil izvajamo kvantitativno delta testiranje, se nam bodo vsa neujemanja med pričakovanimi in dejanskimi rezultati prikazala v poročilu, ki ga nato podrobno pregledamo in se odločimo, ali so neujemanja in ujemanja rezultatov skladna s spremembami poslovnih pravil ali ne. Če se neujemanja v rezultatih skladajo s spremembami poslovnih pravil, lahko rezultate izvajanja zdajšnje izzivalne množice poslovnih pravil uporabimo pri nadaljnjem testiranju kot rezultate izvajanja branilne množice pravil oz. kot pričakovane rezultate izvajanja pri testnih naborih nove različice množice poslovnih pravil.

5. 2. 2. 7 Vzpostavitev testnega okolja

Pri opisu postopka testiranja v poglavju 5. 2. 2. 5 smo se osredotočili na poslovne uporabnike, saj je to tematika diplomskega dela, dodatno pa predstavlja postopek testiranja za poslovne uporabnike nadgradnjo postopka testiranja, kot ga izvajajo razvijalci. Za razliko od razvijalcev, ki teste poganjajo lokalno v Rule Studiu, poslovni uporabniki teste izvajajo neposredno na izvajalnem okolju, kar prikazuje slika 5. 12.

Da bi poslovnim uporabnikom omogočili izvajanje testov v RTS-ju, ga je treba predhodno ustrezno nastaviti in vzpostaviti ustrezno testno izvajalno okolje. Za vzpostavitev testiranja v RTS-ju z uporabo DVS je tako treba:

1. Ustrezno definirati projekte poslovnih pravil z ustreznimi vhodnimi parametri, ki predstavljajo vhodne podatke za testne primere.
2. Projekte poslovnih pravil postaviti na RTS oz. jih shraniti v njegov repozitorij, kjer so na razpolago poslovnim uporabnikom.
3. Validirati BOM in kreirati ustrezne Excelove predloge za vnos testnih podatkov ali definirati druge načine dostopa do podatkov testnih scenarijev. V primeru, da se za definiranje testnih scenarijev uporabljajo preglednice v Excelu, je treba preveriti, da predloge vsebujejo ustrezna polja za posamezne vrednosti podatkov.
4. Postaviti testno izvajalno okolje.
5. SSP-ju omogočiti dostop do XOM-a.

Izpolnitev vseh potrebnih pogojev pa ni brez posledic za obstoječe programske rešitve oz. aplikacije poslovnih pravil. Preden podrobneje opišemo potrebne posege, si bomo pogledali še alternativno možnost, s katero lahko poslovnim uporabnikom zagotovimo takojšnje delo z obstoječimi aplikacijami poslovnih pravil, njihovo testiranje in postavitev na izvajalno okolje.

5. 3 Alternativna možnost

Kot alternativno možnost, ki ne zahteva posegov v obstoječe programske rešitve oz. aplikacije poslovnih pravil, lahko poslovnim uporabnikom zagotovimo delo s poslovnimi pravili z uporabo razvojnega orodja Rule Studio. Slednji nudi poslovnim uporabnikom vse potrebno za delo s poslovnimi pravili, prav tako pa je omogočeno testiranje aplikacij poslovnih pravil in njihova postavitev na izvajalno okolje.

Prednost take rešitve je v tem, da bi imeli poslovni uporabniki takoj možnost dela s poslovnimi pravili, vendar ima kar nekaj pomanjkljivosti:

- Od poslovnih uporabnikov zahteva dobro poznavanje razvojnih orodij.
- Zahteva dobro poznavanje strukture programske rešitve.
- Poleg urejanja poslovnih pravil omogoča poslovnim uporabnikom tudi urejanje preostale kode, ki jo imajo na vpogled.
- Obstoječi postopek testiranja zahteva od njih veliko tehničnega znanja, dodatno pa ne omogoča testiranja aplikacij poslovnih pravil po principu bele skrinjice.

Rule Studio je osnovan na orodju Eclipse in je kot tak v prvi vrsti namenjen razvijalcem. Poleg funkcionalnosti, ki omogočajo delo s poslovnimi pravili, vsebuje tudi preostale funkcionalnosti, potrebne razvijalcem pri razvoju programskih rešitev, ki za poslovnega uporabnika nimajo dodane vrednosti in predstavljajo tveganje, da bi slednji zaradi njih vplival na preostalo kodo. Kot smo že omenili, so poslovna pravila med razvojem shranjena v enem ali več projektih poslovnih pravil, ki poleg samih pravil vsebujejo tudi preostale elemente in kodo, ki je za poslovne uporabnike nepomembna. Dodatno se projekti poslovnih pravil navezujejo na druge javanske projekte, zaradi česar bi morali poslovni uporabniki v primeru uporabe Rule Studia zelo dobro poznati samo projektno strukturo programskih rešitev.

V primeru uporabe Rule Studia za delo s poslovnimi pravili bi poslovnim uporabnikom omogočili dostop ne samo do poslovnih pravil, temveč tudi do preostale, za poslovne uporabnike nepomembne kode, kar nam ni v interesu. Ne samo, da bi poslovne uporabnike količina vseh nepotrebnih podatkov zmedla, s tem bi povečali verjetnost za napake v sistemu v primeru njihovega urejanja ali brisanja preostale kode, dodatno pa bi imeli vpogled v uporabljeno tehnologijo in arhitekturo sistema, česar si ne želimo.

Dodatno težavo v primeru uporabe Rule Studia predstavlja obstoječi postopek testiranja. Poslovna pravila oz. aplikacije poslovnih pravil trenutno testiramo v sklopu testiranja celotnih programskih rešitev, pri čemer se aplikacije poslovnih pravil testirajo po principu črne skrinjice z uporabo integracijskih testov. Pred testiranjem neke programske rešitve podatke, potrebne za obdelavo, zapišemo v podatkovno bazo z uporabo Unitils, po končanem izvajanju pa rezultate, zapisane v podatkovni bazi, primerjamo s pričakovanimi rezultati. To zahteva od razvijalcev in testerjev zelo dobro poznavanje postopka testiranja in strukture podatkovne baze, saj je treba v primeru popravka vhodnih podatkov popraviti XML skripte s podatki, ki jih uporabljajo Unitils za vnos podatkov v podatkovno bazo.

Tudi če bi postopek testiranja ustrezno spremenili, da bi omogočal testiranje izključno aplikacij poslovnih pravil ali celo množic poslovnih pravil po principu bele skrinjice, je z našega stališča uporaba Rule Studia za poslovne uporabnike nesprejemljiva.

6 Vpeljava orodij

Kot smo ugotovili v prejšnjem poglavju, je z našega vidika uporaba Rule Studia – za poslovne uporabnike za delo s poslovnimi pravili – nesprejemljiva. Da bi uporabnikom lahko zagotovili delo s poslovnimi pravili in njihovo testiranje, jim moramo tako omogočiti uporabo RTS in DVS. Vpeljava omenjenih orodij v postopek upravljanja poslovnih pravil pa ni brez posledic za obstoječe programske rešitve. V nadaljevanju si bomo za vsako od omenjenih orodij pogledali, kakšne posledice ima njegova vpeljava na obstoječe aplikacije poslovnih pravil.

6. 1 Rule Team Server

Vpeljava RTS v postopek upravljanja poslovnih pravil ne predstavlja večjih težav, saj ne zahteva posegov v obstoječe aplikacije poslovnih pravil.

Da bi poslovnim uporabnikom omogočili delo z RTS, je treba slednjega namestiti na ustrezni aplikacijski strežnik, ga ustrezno nastaviti in uporabnikom dodeliti potrebne pravice za delo. Da bi jim omogočili dostop do potrebnih projektov poslovnih pravil, jih je treba shraniti v repozitorij RTS, dodatno pa je treba na projektih omogočiti varnost, s čimer preprečimo uporabnikom brez potrebnih pravic dostop do njih. S tem, ko projekte shranimo v repozitorij, omogočimo poslovnim uporabnikom urejanje in brisanje obstoječih poslovnih pravil, prav tako pa imajo možnost dodajanja novih pravil.

6. 2 Decision Validation Services

Vpeljava DVS predstavlja veliko večjo težavo, saj ne zadostuje samo namestitev DVS, temveč je treba vzpostaviti tudi ustrezno testno okolje. V ta namen moramo izpolniti vseh pet pogojev, potrebnih za vzpostavitev testiranja v RTS, ki so navedeni v poglavju 5. 2. 2. 7.

Nekatere izmed pogojev že izpolnjujemo ali pa je njihova izpolnitev trivialna. Za obstoječe aplikacije poslovnih pravil namreč že imamo definirano strukturo projektov poslovnih pravil, prav tako imamo definiran XOM za njihovo izvajanje. Kot smo zapisali v prejšnjem poglavju, vpeljava RTS v upravljanje obstoječih projektov poslovnih pravil ne predstavlja težav, saj je poleg namestitve RTS treba uporabnikom le omogočiti dostop do obstoječih projektov. Težava nastane, ko želimo uporabnikom omogočiti ustrezen način podajanja testnih scenarijev. V tem primeru se izkaže, da je treba v obstoječih aplikacijah poslovnih pravil spremeniti vhodne parametre, da bodo poslovnim uporabnikom omogočali ustrezen način podajanja testnih scenarijev. Da bi jim to omogočili, moramo spremeniti zasnovo aplikacij

pravil, predvsem način dostopa do podatkov. Za konec nam preostane le še vzpostavitev ustreznega testnega okolja, v sklopu katerega moramo SSP-ju omogočiti dostop do XOM-a za potrebe izvajanja poslovnih pravil.

6. 2. 1 Dostop do podatkov

Obstoječe aplikacije poslovnih pravil med samim izvajanjem dostopajo neposredno do podatkovne baze z uporabo vmesnikov DAO. Vhodne podatke tako sestavljajo le potrebni osnovni objekti ter vmesniki DAO, potrebni za dostop do podatkovne baze. Dejanske implementacije vmesnikov oz. njihova zrna se pred izvajanjem aplikacij poslovnih pravil preberejo iz XML nastavitvenih datotek ogrodja Spring (glej poglavje 4. 3). Takemu načinu podajanja vhodnih podatkov je prilagojen tudi obstoječi način testiranja aplikacij poslovnih pravil. Trenutno se namreč aplikacije pravil testirajo po principu črne skrinjice v sklopu testiranja celotne programske rešitve. Pred testiranjem vse podatke, potrebne za izvedbo testov, prepišemo iz XML skript s podatki v podatkovno bazo z uporabo Unitils. Pred začetkom izvajanja aplikacij poslovnih pravil se iz Springovih XML nastavitvenih datotek preberejo vsa potrebna zrna z implementacijami vmesnikov DAO, ki se nato skupaj z osnovnimi objekti, ki vsebujejo izključno identifikatorje zapisov v podatkovni bazi, pošljejo kot vhodni podatki aplikacijam poslovnim pravil. Aplikacije pred začetkom izvajanja poslovnih pravil z uporabo ustreznih metod vmesnikov DAO (oz. vstavljenih implementacij) iz podatkovne baze pridobijo vse potrebne podatke. Po končanem izvajanju poslovnih pravil posreduje aplikacija pravil dobljene rezultate krovni aplikaciji, ki jih zapiše v podatkovno bazo. Po končanem testiranju se z uporabo Unitils rezultati izvajanja v podatkovni bazi primerjajo s pričakovanimi rezultati, ki so zapisani v ustreznih XML skriptah.

Iz opisa obstoječega postopka testiranja je razvidno, da bi lahko že sedaj poslovnim uporabnikom omogočili preverjanje dobljenih rezultatov s pričakovanimi, ki bi jih ustrezno definirali v Excelovih preglednicah ali podatkovni bazi. Večjo težavo predstavlja način podajanja vhodnih podatkov, saj se vstavljanje ustreznih zrn izvede izven aplikacij poslovnih pravil na nivoju celotne programske rešitve, prav tako pa njihove implementacije niso vključene v sam XOM, saj so za aplikacije poslovnih pravil oz. njihov BOM nepomembne. Dodatna slabost omenjenega podajanja vhodnih podatkov je, da se podatki berejo neposredno iz podatkovne baze, na njeno vsebino pa poslovni uporabniki nimajo neposrednega vpliva.

Da bi uporabnikom lahko zagotovili spreminjanje podatkov testnih scenarijev pred izvajanjem testov, jim moramo omogočiti, da aplikacijam oz. množicam poslovnih pravil kot vhodne podatke posredujejo vsaj nek osnoven nabor podatkov, če že ne vseh potrebnih.

Vsi objekti, ki jih potrebujemo kot vhodne podatke, so že vsebovani v BOM-u in XOM-u, tako da objektnih modelov ni treba spreminjati. Namesto da aplikacijam poslovnih pravil med izvajanjem kot vhodne podatke podamo le identifikatorje zapisov v podatkovni bazi, moramo vse potrebne podatke predhodno prebrati iz baze in jim jih posredovati kot vhodne parametre.

Omenjeni način podajanja vhodnih podatkov priporočajo tudi najboljše prakse razvoja aplikacij poslovnih pravil. S tem se izognemo neposrednemu dostopu aplikacij do podatkovne baze in v primeru naših arhitekturnih rešitev tudi podajanju ustreznih implementacij vmesnikov DAO kot vhodnih podatkov. Slednji na ta način predstavljajo le poslovno relevantne objekte, ki se pojavljajo v poslovnih pravilih in so kot taki blizu poslovnim uporabnikom.

Da bi poslovni uporabniki lahko samostojno podajali potrebne podatke oz. testne scenarije, morajo razvijalci predhodno validirati BOM in kreirati ustrezne predloge preglednic v Excelu za vnos testnih scenarijev. Oboje storijo z uporabo čarovnikov v Rule Studiu. Validacija BOM-a preveri, ali za vse potrebne vhodne podatke obstajajo ustrezni konstruktorji, z uporabo katerih se iz podanih testnih scenarijev ustvarijo ustrezni objekti, ki se uporabljajo med izvajanjem množic poslovnih pravil. Če ustrezni konstruktorji ne obstajajo, morajo razvijalci slednje implementirati.

Ko je validacija BOM-a uspešna, kar pomeni, da so na voljo vsi potrebni konstruktorji, lahko razvijalci kreirajo ustrezne predloge preglednic v Excelu. V osnovi predloge vsebujejo vse objekte in njihove atribute, ki se pričakujejo kot vhodni parametri aplikacije poslovnih pravil, prav tako pa vsebujejo tudi objekte, ki se navezujejo na osnovne vhodne objekte. Če je objektni model preveč kompleksen ali pa vsi podatki niso potrebni za testiranje različnih scenarijev, lahko predloge ustrezno popravijo, da omogočajo vnos le minimalnega nabora podatkov. Pri tem morajo preostale podatke zagotoviti na drug način z ustrezno prikrojitvijo SSP-ja (glej poglavje 5. 2. 2. 4).

Preostale podatke lahko tako neposredno zakodirajo v sam XOM, od koder se posredujejo množicam poslovnih pravil med izvajanjem testnih scenarijev, ali pa tako prikrojijo SSP, da potrebne testne scenarije prebere iz podatkovne baze. V tem primeru SSP vse podatke o testnih scenarijih prebere iz baze, poslovni uporabniki pa v sklopu testnih zbirk SSP-ju posredujejo le pričakovane podatke, za kar imajo na voljo ustrezne predloge preglednic v Excelu. Slabost takega načina podajanja testnih scenarijev je zopet v tem, da so podatki shranjeni v podatkovni bazi, na vsebino katere poslovni uporabniki nimajo neposrednega vpliva.

Ko razvijalci definirajo ustrezne predloge preglednic v Excelu, jih morajo še napolniti z ustreznimi testnimi scenariji, da tega ni treba poslovnim uporabnikom. Ustrezne testne scenarije definirajo skupaj z uporabniki, podatke o njih pa nato vnesejo v predloge preglednic. Če se podatki o testnih scenarijih že nahajajo v podatkovni bazi, jim jih ni treba prepisovati na roke, temveč jih lahko v preglednice v Excelu prečrpajo z uporabo DVS-jevega vmesnika API. Preglednice s podatki o vseh testnih scenarijih nato shranijo v RTS, kjer so na voljo poslovnim uporabnikom za definiranje testnih naborov, prav tako pa jih lahko uporabljajo razvijalci za potrebe testiranja poslovnih pravil v Rule Studiu.

6. 2. 2 Postavitev testnega izvajalnega okolja

Da bi z DVS lahko testirali množice poslovnih pravil, je treba poleg testnih scenarijev zagotoviti tudi ustrezno testno izvajalno okolje, ki poleg samega izvajalnega okolja (strežnika RES) vsebuje tudi SSP s potrebnimi XOM-i.

Pri testnem izvajalnem okolju lahko omogočimo testiranje na obstoječem izvajalnem okolju, ki predstavlja tudi produkcijsko okolje, ali pa v ta namen postavimo ločeno izvajalno okolje, namenjeno izključno testiranju. Ker v primeru testiranja z DVS ne testiramo neposredno aplikacij poslovnih pravil, temveč testiramo množice pravil z uporabo SSP-ja, lahko uporabimo že obstoječe izvajalno okolje, saj za testiranje skrbi ločena komponenta, tj. SSP.

Pri testiranju celotnih aplikacij poslovnih pravil je treba testno različico aplikacije postaviti na izvajalno okolje, s čimer ogrozimo trenutno veljavno različico aplikacije poslovnih pravil. Naše programske rešitve namreč dostopajo do zadnje različice aplikacije (glej poglavje 4. 1. 2. 6), kar bi v tem primeru pomenilo, da bi dostopale do testne različice aplikacije.

DVS testira izključno množice poslovnih pravil, ki so za potrebe izvajanja vsebovane v aplikaciji poslovnih pravil. Ker se množice pravil pošljejo v izvajanje SSP-ju, ki predstavlja ločeno komponento v sklopu izvajalnega okolja, ni nevarnosti, da bi ogrozili trenutno veljavno različico aplikacije. Brez nevarnosti lahko zato kot testno izvajalno okolje uporabimo obstoječe produkcijsko izvajalno okolje, kar je povezano tudi z manjšimi stroški, saj je treba za vsako postavitev izvajalnega okolja pridobiti ustrezno licenco.

Množice poslovnih pravil, ki se pošljejo v testiranje na SSP, ne vsebujejo XOM-a, temveč vsebujejo le BOM in preslikavo med BOM-om in XOM-om (glej poglavje 4. 1. 2. 5). V ta namen je treba SSP-ju omogočiti dostop do ustreznih XOM-ov.

Pri razvoju aplikacij poslovnih pravil uporabljamo javanski XOM, ki je vsebovan v ločenih javanskih projektih. Da bi SSP-ju omogočili dostop do potrebnega XOM-a, morajo administratorji pred namestitvijo SSP-ja v njegovo namestitveno datoteko dodati vse razrede javanskega XOM-a, kar storijo z uporabo čarovnika v Rule Studiu. Tako dopolnjen SSP nato namestijo na testno izvajalno okolje oz. v našem primeru neposredno na obstoječe izvajalno okolje, kjer nato omogoča izvajanje testov. V primeru spremembe XOM-a je treba SSP-ju omogočiti dostop do spremenjenih razredov, kar zopet storijo administratorji po že opisanem postopku, pri čemer že nameščeni SSP s staro različico XOM-a odstranijo in na testno izvajalno okolje namestijo SSP z novo različico XOM-a.

Postavitev testnega izvajalnega okolja tako ne predstavlja večjih težav, saj lahko v ta namen uporabimo obstoječe izvajalno okolje, na katerega namestimo SSP s potrebnimi XOM-i. Če ne bi bilo treba spremeniti zasnove obstoječih aplikacij poslovnih pravil za potrebe podajanja vhodnih parametrov, kar ima za posledico tudi spremembo XOM-a, bi lahko poslovnim uporabnikom omogočili testiranje obstoječih aplikacij poslovnih pravil.

Zaključek

Z izpolnitvijo vseh potrebnih pogojev za vpeljavo Decision Validation Services za potrebe testiranja poslovnih pravil omogočimo testiranje množic in posledično aplikacij poslovnih pravil po principu bele skrinjice. Skupaj z vpeljavo Rule Team Serverja v proces upravljanja poslovnih pravil pa omogočimo poslovnim uporabnikom aktivno sodelovanje v življenjskem ciklu, kar je tudi naš cilj.

S tem, ko poslovnim uporabnikom omogočimo aktivno sodelovanje v življenjskem ciklu poslovnih pravil in posledično v procesu upravljanja z njimi, se pokažejo tudi vse prednosti ločitve poslovnih pravil od preostale logike programskih rešitev oz. informacijskih sistemov in uporabe sistemov za upravljanje poslovnih pravil. V primeru sprememb pravil tako ni treba popravljati celotnih programskih rešitev, temveč lahko poslovni uporabniki sami spremenijo potrebna pravila, jih stestirajo z uporabo običajnega postopka testiranja ali postopka delta testiranja in postavijo novo različico aplikacije poslovnih pravil v produkcijsko okolje. S tem skrajšamo čas, potreben za postavitev novih različic aplikacij v produkcijsko okolje, kar omogoča organizaciji hitrejše prilagajanje na spremembe v poslovanju. Dodatno lahko poslovni uporabniki sami, če imajo dovolj tehničnega znanja, definirajo nove projekte poslovnih pravil (seveda z nekaterimi omejitvami) nad obstoječim poslovnim objektnim modelom, s čimer lahko pokrijejo vedno večji delež poslovnih pravil v sklopu organizacije. Nadalje, možnost aktivnega sodelovanja poslovnih uporabnikov v procesu upravljanja poslovnih pravil pokaže tudi resnično prednost uporabe sistema za upravljanje poslovnih pravil, s čimer veliko lažje opravičimo sorazmerno visoke stroške nakupa licenc za uporabo.

Vpeljava Rule Team Serverja in Decision Validation Services pa ne prinaša prednosti samo poslovnim uporabnikom, temveč tudi razvijalcem med samim razvojem aplikacij poslovnih pravil. Z uporabo Rule Team Serverja lahko poslovnim uporabnikom omogočijo vpogled v nastajajoča poslovna pravila. Ker se za zapis slednjih uporablja domensko specifični jezik, lahko hitro opozorijo na morebitne napake v pravilih in jih tudi sami odpravijo, razvijalci pa nato spremembe uskladijo z Rule Studiem. Prav tako lahko Decision Validation Services s pridom izkoristijo za potrebe testiranja nastajajočih poslovnih pravil. Pri tem za testiranje uporabljajo lokalno testno okolje v sklopu Rule Studia, kar je še posebej pomembno na samem začetku razvoja, ko izvajalni objektni model še ni dokončno definiran in se pogosto spreminja. Če ne bi imeli možnosti lokalnega izvajanja, bi morali ob vsaki spremembi izvajalnega objektnega modela na izvajalno okolje ponovno namestiti Scenario Service Provider z ustreznim izvajalnim objektnim modelom, kar pa je časovno potratno in za razvijalce nesprejemljivo. Dodatno lahko testne scenarije poganjajo kot JUnit teste (glej poglavje 4. 4), pri čemer posamezni scenariji predstavljajo posamezne testne primere (ang. *TestCase*), pričakovani rezultati pa se preverjajo z uporabo ustreznih metod za preverjanje (t. i. *assert* metode).

Med pisanjem diplomskega dela smo v obstoječe programske rešitve že uvedli nekatere spremembe aplikacij poslovnih pravil. Do potrebnih podatkov tako več ne dostopamo med samim izvajanjem aplikacij poslovnih pravil, temveč jih iz podatkovne baze preberemo pred samim izvajanjem in aplikacijam v izvajanje podamo le ustrezne objekte z vsemi potrebnimi podatki. Kljub temu poslovnim uporabnikom še vedno nismo omogočili aktivnega dela s poslovnimi pravili, se pa pripravljamo na to, da jim bomo omogočili spreminjanje obstoječih poslovnih pravil z uporabo Rule Team Serverja.

O odzivu poslovnih uporabnikov na možnost dela s poslovnimi pravili tako še ne moremo govoriti, je pa ravno ta možnost vplivala na njihovo odločitev za uporabo pristopa poslovnih pravil in izbrani sistem za upravljanje poslovnih pravil.

Diplomsko delo smo začeli s citatom in ga tudi zaključujemo z njim:

„Korporacija je živi organizem, ki se mora neprestano leviti. Metode se morajo spreminjati. Fokus se mora spreminjati. Vrednote se morajo spreminjati. Rezultat vseh sprememb je preobrazba.”

Andrew Grove

Literatura in viri

[1] D. S. Appleton, „Business Rules: The Missing Link,” *Datamation*, New York, ZDA, 15. okt., 1984, str. 145–150.

[2] M. Bajec, M. Krisper, „A methodology and tool support for business rule management in organisations,” *Information Systems*, Let. 30, 2005, str. 423–443.

[3] R. T. Burlton, *Business Process Management: Profiting from Success*, Indianapolis: Sams Publishing, 2001.

[4] M. Chisholm, *Managing Reference Data in Enterprise Databases: Binding Corporate Data to the Wider World*, San Francisco: Morgan Kaufmann Publishers, 2001.

[5] I. Graham, *Business Rule Management and Service Oriented Architecture: A Pattern Language*, Chichester, Velika Britanija: John Wiley & Sons Ltd., 2006.

[6] R. G. Ross, *Principles of the Business Rules Approach*, Boston: Addison Wesley Professional, 2003.

[7] M. Silič, M. Colnar, M. Krisper, et al., „3. 1 Poslovna pravila,” v: *Enotna metodologija razvoja informacijskih sistemov – Strateško planiranje*, Ljubljana: Vlada Republike Slovenije, Center za informatiko, 2003, str. 64–71.

[8] F. Solina, „Poglavje 10: Testiranje,” v: *Projektno vodenje razvoja programske opreme*, Ljubljana: Založba FE in FRI, 1997, str. 175–190.

[9] B. Von Halle, *Business Rules Applied: Building Better Systems Using the Business Rule Approach*, New York: Wiley Computer Publishing, 2002.

[10] (Marec 2010) A Brief History of the Business Rule Approach [Online]. Dostopno na: http://www.brcommunity.com/b216.php.

[11] (Marec 2010) P. Berlandier. Changing the Rules of Testing ~ Testing Strategies for the Production Maintenance of Rapidly Evolving Business Rules Systems, 2006 [Online]. Dostopno na: http://www.brcommunity.com/b270.php.

[12] (Marec 2010) Business rule [Online]. Dostopno na: http://en.wikipedia.org/wiki/Business_rule.

[13] (Marec 2010) Bussiness rule management system [Online]. Dostopno na: http://en.wikipedia.org/wiki/Business_rule_management_system.

[14] (April 2010) Implementation of business rules and business processes in SOA [Online]. Dostopno na: http://www.infoq.com/articles/business-rules-processes.

[15] (Marec 2010) A. Kovačič. The Rule Transformation Approach to Business Renovation. Business Rules Journal, Vol. 4, Št. 8, 2003 [Online]. Dostopno na:

http://www.brcommunity.com/b162.php.

[16] (April 2010) J. Pan. Software Testing [Online]. Dostopno na: http://www.ece.cmu.edu/~koopman/des_s99/sw_testing/.

[17] (April 2010) Series: Business Rules Governance and Management [Online]. Dostopno na: http://www.primatek.ca/blog/series/governance-management/.

[18] (April 2010) Software testing [Online]. Dostopno na: http://en.wikipedia.org/wiki/Software_testing.

[19] (Marec 2010) the Business Rule Group: Defining Business Rules ~ What Are They Really? [Online]. Dostopno na: http://www.businessrulesgroup.org/first_paper/BRG-whatisBR_3ed.pdf.

[20] (April 2010) The Business Rules Manifesto [Online]. Dostopno na: http://www.businessrulesgroup.org/brmanifesto.htm.

[21] (Maj 2010) WebSphere ILOG JRules V7.0, 2009 [Dokumentacija].

Seznam slik in tabel

Seznam slik

Slika 1. 1: Kategorije poslovnih pravil [2, 7] 9
Slika 2. 1: Konceptualna shema poslovnih pravil [7] 12
Slika 2. 2: Življenjski cikel poslovnih pravil 14
Slika 2. 3: Osnovne komponente sistema za upravljanje poslovnih pravil 16
Slika 3. 1: Ocena sprememb v programski opremi od faze analize do faze vzdrževanja [8] 20
Slika 3. 2: Testiranje programske opreme [8] 24
Slika 3. 3: Ogrodje običajnega postopka testiranja [11] 26
Slika 3. 4: Ogrodje delta testiranja [11] 27
Slika 4. 1: ILOG JRules produkti in moduli (povzeto po [21]) 31
Slika 4. 2: Preslikava poslovnega pravila v BOM in XOM (povzeto po [21]) 34
Slika 4. 3: Poslovno pravilo, zapisano v jeziku BAL 35
Slika 4. 4: Odločitvena tabela, zapisana v jeziku BAL 36
Slika 4. 5: Tehnično pravilo, zapisano v jeziku IRL 36
Slika 4. 6: Tronivojska arhitektura 38
Slika 4. 7: Delitev aplikacijskega nivoja 39
Slika 4. 8: Ogrodje Spring na aplikacijskem nivoju 40
Slika 5. 1: Obstoječi življenjski cikel poslovnih pravil 44
Slika 5. 2: Želeni življenjski cikel poslovnih pravil 45
Slika 5. 3: Življenjski cikel poslovnih pravil in ILOGova orodja za poslovne uporabnike 46
Slika 5. 4: Povezava Rule Studia, Rule Team Serverja in Rule Execution Serverja (povzeto po [21]) 47
Slika 5. 5: Navigacija po izbranem projektu v zavihku Explore 48
Slika 5. 6: Predloga poslovnega pravila v Rule Team Serverju 49
Slika 5. 7: Nastavljanje pravic uporabniški skupini 52
Slika 5. 8: Preglednica v Excelu z dvema testnima scenarijema 54
Slika 5. 9: Preglednica s pričakovanimi rezultati za testne scenarije 55
Slika 5. 10: Preglednica s pričakovanimi podrobnostmi izvajanja testnih scenarijev 55
Slika 5. 11: Poročilo o izvajanju testnih scenarijev 56
Slika 5. 12: Postopek testiranja v Rule Team Serverju 58

Seznam tabel

Tabela 1. 1: Evolucija definicije poslovnega pravila 7
Tabela 1. 2: Primeri definicij (povzeto po [19]) 9
Tabela 1. 3: Primeri omejitev (povzeto po [19]) 10
Tabela 1. 4: Primeri izpeljav (povzeto po [19]) 10
Tabela 2 .1: Zapis poslovnega pravila v naravnem jeziku, javi in BAL-u 17
Tabela 3. 1: Stroški odprave pomanjkljivosti v fazah razvoja glede na fazo pojavitve [18] 21