UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

ALJAŽ ZRNEC

Odločitveni model za prilagajanje procesa razvoja informacijskih sistemov individualnim potrebam projektov

DOKTORSKA DISERTACIJA

Mentor: doc. dr. MARJAN KRISPER
Somentor: izr. prof. dr. VILJAN MAHNIČ

LJUBLJANA, 2006

PERFECTION IS ACHIEVED NOT WHEN THERE IS
NOTHING MORE TO BE ADDED BUT WHEN
THERE IS NOTHING LEFT TO TAKE AWAY

ANTOINE DE SAINT-EXUPERY

Kazalo vsebine

Kazalo slik

Kazalo tabel

Povzetek

Doktorska disertacija obravnava formalno opredelitev odločitvenega modela za prilagajanje procesa razvoja informacijskih sistemov potrebam individualnih projektov. Zmožnost prilagajanja razvojnega procesa potrebam projektov, ki se izvajajo v okrilju organizacijskega sistema, predstavlja enega od temeljev za zagotavljanje ustreznosti in sprejetosti metodologije, katere uporabo predpisuje organizacijski sistem.

Ideja o prilagajanju metodologij za razvoj informacijskih sistemov in s tem razvojnega procesa, ki ga ta predpisuje, ni nova. Kot odgovor na tehnično neustreznost metodologij se je razvilo ti. področje konstruiranja metodologij, ki skuša odpraviti težave, ki jih povzročajo togost, neprilagodljivost in tehnična neustreznost tradicionalnih metodologij. Pri tem se raziskave s tega področja osredotočajo predvsem na teoretične opredelitve, ki temeljijo na uporabi zapletenih matematičnih formalizmov, kar je tudi botrovalo temu, da se izsledki področja nikoli niso širše uveljavili v praksi. Poleg tega se pristopi konstruiranja metodologij skoraj ne dotikajo problematike socialne ustreznosti in s tem sprejetosti metodologije med njenimi uporabniki, kar jim daje dodaten negativen pridih z vidika praktične uporabe.

S pojavom agilnih pristopov postane ideja o prilagajanju metodologij veliko bolj zanimiva, saj pobuda tokrat pride s strani razvijalcev, ki delujejo v praksi. Ideja agilnih pristopov je zelo podobna ideji, na kateri temelji konstruiranje metodologij, le da se pri tem uporablja veliko bolj enostavne prijeme. V nasprotju z disciplino konstruiranja metodologij se agilne metodologije osredotočajo predvsem na prilagajanje procesa, poleg tega pa posvečajo veliko več pozornosti samim sociološkim vidikom njegove uporabe.

Ključ do uspeha pri uvajanju metodologij v organizacijske sisteme in njihovi kasnejši uporabi ter sprejetosti, vidimo zato predvsem v zmožnosti prilagajanja razvojnega procesa, ki ga predpisuje uporabljena metodologija, specifičnim potrebam projektov, ki jih organizacijski sistemi izvajajo. Zaradi tega mora metodologija, ki je prilagodljiva, definirati fleksibilen proces razvoja, ki ga je možno prilagajati razmeram, v kakršnih se projekti vzpostavljajo. Pri tem je potrebno postopek prilagajanja procesa podpreti z ustreznim programskim orodjem, ki omenjeni postopek prilagajanja avtomatizira in s tem olajša delo metodologa.

Namen doktorske disertacije je združiti teoretična znanja s področja konstruiranja metodologij in praktična znanja pridobljena na osnovi izkušenj iz podjetij za razvoj programske opreme. Na temelju tega podamo formalno opredelitev odločitvenega modela za prilagajanje procesa, ki v nadaljevanju disertacije služi opredelitvi izhodišč za izdelavo prototipa programskega orodja za podporo dela metodologa, ki izvaja prilagoditve procesa.

Pristop k prilagajanju, ki ga predstavi doktorska disertacija, temelji na iskanju prehoda - poti skozi graf osnovnega procesa. Predlagan pristop razširja idejo o prilagajanju, ki temelji na več možnih prehodih skozi dano metodologijo in predstavlja eno od oblik situacijskega konstruiranja metodologij. Razširitev pristopa predlaga odpravo vnaprej določenih poti skozi proces, ki ga uporabljena metodologija predpisuje. Pri tem je osnovni proces opredeljen kot proces, ki je predhodno že vpeljan v organizacijski sistem in predstavljen s pomočjo mešanega grafa. Gradnjo prilagojene različice procesa, ki ustreza potrebam danega projekta, usmerja odločitveni model z uporabo odločitvenih pravil. Odločitveni model mora za posamični element procesa, ki se nahaja v osnovnem procesu, podati odgovor, ali se ta vključi v prilagojeno različico procesa ali ne. Odločitvena pravila predstavljajo temelj tako za izgradnjo prilagojenih različic procesa, kot za zagotavljanje njihove celovitosti in odkrivanje odvisnosti med karakteristikami projekta. V okviru doktorske disertacije jih predstavimo z metamodelom in opredelitvijo njihove zgradbe.

Karakteristike projekta predstavljajo koncept, s katerim se predstavi lastnosti projekta. Disertacija poleg tehnoloških opredeli tudi sociološke karakteristike in zahteve naročnika. Z uporabo več tipov karakteristik se zagotovi, da prilagojena različica osnovnega procesa ustreza zahtevam projekta tako s tehničnega kot sociološkega vidika.

Formalno opredeljena odločitvena pravila in odločitveni model vzpostavijo teoretično osnovo za opredelitev izhodišč za izdelavo prototipa programskega orodja za prilagajanje procesa. Za predstavitev predvidene funkcionalnosti orodja se v okviru disertacije uporabijo opisi primerov uporabe. Orodje, ki ga predlagamo v okviru disertacije, predstavlja celovito rešitev za prilagajanje procesa potrebam individualnih projektov, saj poleg tehničnega vidika upošteva tudi socialni vidik metodologije, ki se uporablja v okviru celotnega organizacijskega sistema.

Decision model for information system development process adaptation to individual needs of the projects

Abstract

This dissertation presents formal definition of the decision model for information systems development process adaptation to individual needs of the projects. The ability of the software process to adapt to project specific needs, presents one of the most important factors for assuring methodology suitability and acceptance among software engineers in particular organizational system.

The idea of adaptation of information systems development methodology is not new. Inability of traditional methodologies to adapt to particular situational needs, their rigidness and technical unsuitability led to development of special scientific field, called method engineering. The field of method engineering is very complex, because it focuses mainly on theoretical point of view of methodology construction and adaptation, and it uses very complex mathematical formalisms. That is also the main reason why method engineering has never been widely acknowledged or practiced by software engineers. Method engineering also doesn't deal with social suitability and acceptance of the methodology among its users. That has been another reason why the field hasn't been accepted among the engineers.

With the emergence of the new approach, also called "agile approach", the idea of systems development methodology adaptation seems to become more attractive, as this time the initiative comes from practioners. The idea of agile approaches is very similar to the idea of method engineering, only that agile approaches are not so complex. In contrast to method engineering the focus in agile methodologies is on the adaptation of the process. Moreover, agile methodologies seem to give more attention to sociological aspects of the methodology, which have been rather ignored in method engineering.

The key to successful introduction of systems development methodologies into organizations, their later use and acceptance, lies in the ability of its prescribed software process to be adaptable to project specific needs. The methodology has to define flexible software process, capable of adapting to

specific situations, in which new projects emerge. The process adaptation has to be supported by software tool which automates the adaptation procedure and facilitates the work of method engineer.

The aim of this dissertation is to combine theoretical knowledge from method engineering and practical knowledge from organizations for software development. On this basis we formally define decision model for information systems development process adaptation to individual needs of the projects. The decision model serves as a basis for definition of the specification for software tool prototype.

The approach to process adaptation presented in this dissertation is based upon selecting the most suitable path through the given process, used in the organization. The information system development process used in the organization is defined as a base process. The approach extends the idea of choosing one of the possible paths through the given methodology, defined in situational method engineering field. Our extension of the approach suggests abolition of predefined paths. The construction of adapted instance of a base process is directed by decision model, on the basis of decision rules. For every process element, the decision model has to give an answer about its inclusion in adapted instance of the base software process. Decision rules are used for construction of adapted instances, assuring integrity of instances and definition of dependencies among project characteristics. In this dissertation we present decision rule metamodel and structure of proposed decision rules.

Project characteristics are used to describe projects. Beside technological characteristics, we have also defined sociological characteristics and customer requirements in this dissertation. Multiple types of characteristics ensure, that the adapted instance of the base process meets various technological and sociological needs of the project.

Formal definition of the decision model along with decision rules establish theoretical basis for the software tool prototype specification. The software tool is intended to help method engineer at his work. We used use cases to describe expected functionality of the software tool. The tool defined in the dissertation represents complete solution for process adaptation to project specific needs, because it considers technological and sociological point of view about base methodology use, used in the particular organization.

1 Uvod

1.1 Problemi razvoja informacijskih sistemov

Čeprav so postale informacije ena od najpomembnejših prvin poslovnega procesa v modernih organizacijskih sistemih, pa se na področju samega razvoja IS[1] srečujemo s številnimi težavami. Najbolj pereča je *nizka produktivnost in veliko število napak*, kar se odraža v IS, ki ne ustrezajo potrebam poslovanja. Nizko produktivnost označujemo tudi s pojmom "kriza programske opreme", kjer gre za dejstvo, da zahteve za izgradnjo novih ali izpopolnjenih IS rastejo hitreje kot pa zmožnosti za njihov dejanski razvoj. Vzroke za tako stanje lahko najdemo v vedno večjih stroških razvoja programske opreme (še posebej, če jih primerjamo z vedno nižjimi stroški strojne opreme), omejenih človeških in denarnih virih ter samo zmernih izboljšavah v produktivnosti razvojnega procesa.

Drugi problem, ki ga moramo upoštevati, predstavljajo projekti razvoja IS, ki so se končali kot popolni polom. Vzroki za neuspeh so bili različni, od prekoračitve proračuna, ki ga je imel projekt na voljo, do prekoračitve časovnih rokov, presenetljivo velikokrat pa je bila vzrok za neuspeh prenizka kakovost izdelane programske opreme in nezadovoljstvo uporabnikov. Gladden [Gladden 1982] je v svoji raziskavi ocenil, da se kar 75% projektov razvoja programske opreme nikoli ne konča, ali pa se izdelan sistem nikoli ne uporablja. Samo 16% razvojnih projektov pa naj bi bilo zaključenih po predvidenih finančnih in časovnih načrtih. V okviru iste raziskave, v katero je bilo vključenih 365 vodij iz področja IT[2], je bilo tudi ugotovljeno, da je bilo 31% projektov ukinjenih pred zaključkom, 53% pa jih je bilo sicer izpeljanih do konca, vendar so krepko presegli načrtovana finančna sredstva, poleg tega pa je bila funkcionalnost končnega izdelka manjša od načrtovane.

Z vidika poslovanja se pojavlja še dodaten problem, in sicer neskladnost med potrebami poslovnih sistemov in rešitvami, ki jih ponujajo izdelani IS. Medtem ko se v organizacijskih sistemih delež sredstev, ki je namenjen iskanju, beleženju, prečiščevanju in analizi podatkov povečuje, ostaja obstoj korelacije med uporabo informacijskih rešitev in učinkovitostjo organizacijskega sistema dvomljiv [Smith 1993] (glej razdelek 2.2.3.2 in razdelek 2.2.3.3). Naprimer, veliko ravnateljev IS se še vedno sooča s situacijami, ko ne morejo pridobiti ustreznih informacij za vodenje svojih

[1] *Informacijski sistem*
[2] *Informacijska tehnologija*

oddelkov. Od tod sledi, da je razvoj IS pred neprestanimi izzivi, ki jih postavlja dinamična narava poslovnih sistemov in načini organiziranja poslovnih aktivnosti, ki jih mora informatika v organizacijskih sistemih podpreti.

Vsi omenjeni problemi se še stopnjujejo z vedno večjo kompleksnostjo in obsežnostjo podporne programske opreme. Vsaka generacija programske opreme namreč prinese nove možnosti uporabe in razširi njen obstoječ nabor funkcionalnosti, kar vodi do obsežnejših sistemov, ki jih je težje načrtovati, izdelati in vzdrževati. Nadalje, zaradi velikega števila tehničnih novosti in inovacij, ki so na voljo (trinivojska arhitektura, porazdeljeni sistemi, objektno orientirani pristopi, modelno usmerjeno načrtovanje, elektronsko poslovanje itd.), novi tehnološki vidiki povzročajo preoblikovanje praks na področju razvoja IS. Na področju razvoja IS se je tako ukoreninilo splošno spoznanje, da *razvojne metodologije ne zadovoljujejo potreb organizacijskih sistemov, ne glede ali imamo v mislih tehnični, ekonomski ali socialni vidik uporabe metodologij*. Posledica tega je, da se tako organizacijski sistemi kot akademsko okolje srečujeta z izzivi na dveh področjih:

- opredelitve novih strategij za razvoj IS in
- izdelave podpornih orodij ter načinov dela.

1.2 Metodološka podpora razvoju informacijskih sistemov

Splošno sprejet pristop za reševanje omenjenih problemov predvideva uvedbo sistematičnih smernic in procedur za razvoj programske opreme v okviru organizacijskih sistemov. Znanje, ki je za to potrebno, se nahaja v obliki razvojnih metodologij, katere lahko na kratko opredelimo kot sistematičen pristop za celotno izvedbo vsaj ene razvojne aktivnost. Tako med praktiki kot teoretiki s področja metodoloških pristopov k razvoju IS obstaja enotno mnenje, da je za napake pri razvoju programske opreme kriva predvsem uporaba napačnih razvojnih pristopov. Metodologije predstavljajo možno rešitev tega problema. Tako je danes velik napor v smeri izboljševanja metodologij in razvojnih praks zaznati na vseh področjih razvoja IS: poslovnemu modeliranju, prenovitvi poslovnih procesov, razvoju arhitekture IS, analizi in načrtovanju ter implementaciji. Še zlasti velja, da k večji produktivnosti prispevajo izboljšave v zgodnejših fazah razvoja.

Cilj razvoja metodologij je zbrati čim več izkušenj s področja razvoja IS in jih uporabiti pri izgradnji sistematičnih razvojnih praks. Do teh izkušenj se je mogoče dokopati na več načinov:

- skozi udejstvovanje v razvojnem procesu,
- evaluacijo metodologij in
- spremljanje dejanske uporabe metodologij.

Na osnovi pridobljenih izkušenj metodologi promovirajo nove koncepte, prepričanja, modelirne jezike in postopke. V splošnem se od metodoloških pristopov pričakuje, da vodijo k boljšim rešitvam, ki so rezultat boljše upravljanega razvojnega procesa.

Paradoksalno, kljub vsem naporom vloženih v razvoj metodologij zgleda, da ne obstaja nobenega splošnega soglasja o tem, ali so metodologije sploh uporabne v okviru razvoja IS. Enega glavnih razlogov za pojav omenjenega protislovja predstavlja omejenost in ozka usmerjenost raziskav, saj obstaja predvsem na področju uporabe metodologij presenetljivo malo izkustvenega znanja. Velika večina raziskav se je usmerila predvsem na razvoj novih metodologij ali na razvoj ogrodij za analiziranje, primerjavo in izbiro ustreznih metodologij. Poleg tega glavnina izkustvenih študij o razvoju ali primerjavi metodologij temelji le na majhnem vzorcu primerov in omejenih izkušnjah, pridobljenih skozi njihovo uporabo.

Kljub temu, da sta učinek in uporabnost metodologij včasih lahko vprašljiva, pa je tržišče vseeno poudarjalo pomen uporabe razvojnih orodij in delovno storilnost. Tržišče z razvojnimi orodji, kot so naprimer orodja CASE[3] in generatorji kode, se je v zadnjih petnajstih letih stalno povečevalo. Čeprav se stopnja razširjenosti orodij CASE zvišuje počasneje, kot je bilo na začetku predvideno, je kmalu postalo jasno, da se bo njihova razširjenost v prihodnosti še poviševala.

1.3 Razvoj lastnih metodologij

Čeprav so na tržišču na voljo različne metodologije za razvoj IS, z njimi organizacijski sistemi ponavadi niso zadovoljni. Tako študije primerov kot različne ankete so pokazale, da organizacijski sistemi v večini težijo k razvoju lastnih metodologij ali pa se odločijo za prilagoditev metodologije, ki jo imajo na voljo, svojim konkretnim potrebam. Iz tega lahko sklepamo, da metodologije, ki vstopajo v organizacijski sistem "od zunaj", ne zadoščajo potrebam razvoja v organizacijskem sistemu, oziroma se z vidika uporabnikov metodologije smatrajo kot neuporabne. S tega stališča je torej edina možnost, ki preostane organizacijskemu sistemu, da sploh ne prične uporabljati nobene metodologije, kar z vidika tematike doktorske disertacije predstavlja najslabšo možnost.

[3] *Computer Aided Software Engineering*

Lahko pa poizkusi uporabiti katero drugo metodologijo, recimo komercialno, ki jo prilagodi lastnim potrebam, nadaljuje z uporabo obstoječe (neustrezne), ali pa razvije svojo lastno.

Metodologija razvita v organizacijskem sistemu obsega dele metodologije, ki se ne nahajajo v nobeni drugi metodologiji. Razvoj lastne metodologije običajno obsega kombiniranje ali prilagajanje obstoječih komercialnih metodologij, ki jih ima organizacijski sistem na voljo ali razvoj povsem lastnih metodoloških pristopov.

Pregled raziskav, ki so ugotavljale uporabo metodologij v organizacijskih sistemih, je pokazal, da je v devetdesetih letih prejšnjega stoletja le 38% [Hardy 1995] ali 36% [Yourdon 1992] organizacijskih sistemov od tistih, ki so uporabljali metodologije, uporabljalo lastne metodologije za razvoj IS, razvite v hiši. Te je bilo ponavadi možno prilagoditi tako potrebam celotnega organizacijskega sistema kot potrebam projektov.

Rezultat novejše raziskave [Bajec 2005a], ki je bila opravljena na vzorcu 40 podjetij za razvoj programske opreme v Sloveniji, kaže nekoliko drugačno - izboljšano sliko. S pomočjo ankete, ki je bila pripravljena za vodje oddelkov za informatiko, smo skušali ugotoviti, kako na problematiko gledajo tisti, ki v prej omenjenih podjetjih sprejemajo odločitve o tem, kakšne metodologije se bodo pri razvoju IS uporabljale ter kako podrobno. Namen vprašanja v anketi je bil izvedeti, kakšne metodologije se v slovenskih podjetjih uporabljajo. Gre za komercialne, prilagojene komercialne ali lastne metodologije.

Izmed 40 podjetij jih je 7 odgovorilo, da pri svojem delu uporabljajo komercialne pristope, 8 podjetij, da uporabljajo prilagojene komercialne pristope, kar 25 pa jih je odgovorilo, da uporabljajo lastne pristope, kar predstavlja 62.5% podjetij.

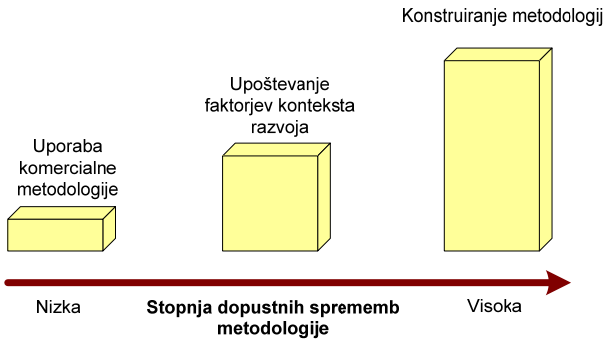
Iz raziskave lahko sklepamo, da se v podjetjih večinoma uporabljajo metodologije, ki nastanejo v okviru samih organizacij, na podlagi več let izkušenj ter se postopoma bogatijo z novimi izkušnjami in pridobljenim znanjem. Pri tem pa je potrebno posebej poudariti, da uporaba lastne metodologije še ne pomeni, da je metodologija dokumentirana, se pravi formalna, temveč le, da so v organizaciji vzpostavljena neka pravila, ki vsaj neformalno veljajo pri razvoju IS.

V okviru razvoja lastnih metodologij poznamo tri osnovne pristope, ki se med seboj razlikujejo glede na način izbora in razvoja lastne metodologije ter njene vpeljave v organizacijski sistem [Tolvanen 1998]. To so:

- uporaba komercialne metodologije (text-book approach),

- pristop z upoštevanjem specifičnih faktorjev konteksta razvoja (ang. contingency approach) in
- konstruiranje metodologij (ang. method engineering).

Omenjene pristope in njihovo stopnjo dopustnih sprememb metodologije prikazuje Slika 1:



Slika 1: Pristopi za razvoj lastnih metodologij

1.3.1 Uporaba komercialne metodologije

V organizacijskih sistemih poteka izbira in vpeljava metodologije ponavadi na temelju preizkušanja. Organizacijski sistem izbere metodologijo neposredno, tako da izbere eno izmed komercialnih metodologij, ki je dobro dokumentirana in ima dobro podporo s strani svetovalcev. Lahko pa izbere metodologijo posredno z uvedbo orodja CASE, ki implicitno tudi integrira določeno metodologijo. Izbrana metodologija se nato vpelje v organizacijski sistem brez modifikacij.

Omenjeni pristop temelji na principu, da so problemi in situacije, ki se pojavijo v okviru razvoja IS podobni, ali podobni vsaj v taki meri, da jih je možno analizirati in reševati z uporabo splošnih metodologij, ki so uporabne v "skoraj" vseh situacijah. Pristop predpostavlja, da je možno situacije iz prakse kategorizirati, tako da so problemi natančno opredeljeni in da jih je možno reševati na standardizirane načine. Z vidika tehnične racionalnosti lahko gledamo na metodologije kot na univerzalne inštrumente za reševanje problemov. Čeprav so se v okviru uporabe pokazale zahteve po prilagodljivost, pa omenjene metodologije ne vključujejo nobenih mehanizmov za prilagajanje različnim karakteristikam razvoja.

1.3.2 Upoštevanje faktorjev konteksta razvoja

Pristop za izbiro metodologije temelji na teoriji, ki zagovarja tezo, da ne obstaja splošna metodologija, ki bi bila primerna za uporabo v vseh okoliščinah (ang. contingency theory). Omenjeni pristop temelji na ugotovitvi, da je situacije iz prakse sicer mogoče klasificirati, vendar zanje uporaba splošnih metodologij ni ustrezna. Ker metodologije ne nudijo splošnih pravil, ki bi obravnavala različne situacije in odklone od v naprej predpostavljenih situacij, poizkušajo različna ogrodja, ki uporabljajo omenjeni pristop, vzpostaviti povezavo med metodološkimi potrebami in metodologijami, ki so na voljo [Tolvanen 1998]. Zagovorniki pristopa poizkušajo identificirati karakteristike za opisovanje problemske situacije, ki vplivajo na uporabo metodologije in na napovedovanje njene primernosti. Poznamo več skupin karakteristik: tehnične (tip IS, uporabljen programski jezik), organizacijske (razvojna kultura in njena zrelost) in človeške (nivo izkušenj, učenje). Osnovni namen ogrodij, ki temeljijo na omenjenem pristopu ni gradnja novih metodologij ampak izbira ustrezne metodologije iz portfelija razpoložljivih metodologij. Odklon v smeri izbiranja metodologij iz množice v naprej opredeljenih metodologij vodi v omejeno konstruiranje in izbiranje metodologij.

Pri uporabi tega pristopa v praksi se pojavi težava glede potrebe po poznavanju več metodologij, ki jih ima organizacijski sistem na voljo in med katerimi lahko izbere najustreznejšo, glede na trenutni kontekst situacije. V praksi je namreč zelo neobičajno, da bi v organizacijskem sistemu bilo na voljo več metodologij, še bolj neobičajno pa je, da bi bili razvijalci domači z vsemi od njih. Omenjenega problema se raziskovalci s področja metodologij zavedajo, zato so se odločili za bolj pragmatičen pristop. Namesto da se ustrezna metodologija izbere iz množice danih metodologij, mora biti metodologija dovolj obširna, tako da pokrije vse možne situacije, ki lahko nastopijo pri razvoju. To je tudi ideja iz katere izhajamo v nadaljevanju doktorske disertacije.

1.3.3 Konstruiranje metodologij

Ideja o prilagajanju metodologij potrebam posameznih projektov ni nova. Raziskave na tem področju segajo v devetdeseta leta prejšnjega stoletja, ko se je razvilo področje konstruiranja metodologij (ang. method engineering). Konstruiranje metodologij je inženirska disciplina, ki se ukvarja z načrtovanjem, konstruiranjem in prilagajanjem metodologij, tehnik in orodij za razvoj IS [Brinkkemper 1996a]. Iz te discipline se je razvila tudi posebna veja konstruiranja metodologij, ki se je ukvarjala s prilagajanjem metodologij konkretnim potrebam projektov - ti. situacijsko konstruiranje metodologij [Fitzgerald 2003; Ralyte 2003; Hofstede 1997a; Brinkkemper 1999; Brinkkemper 1996a; Hofstede 2001; Henderson 2003].

Raziskave na omenjenem področju pa so se soočale tudi s številnimi problemi [Hofstede 1997a; Highsmith 2002b; Tolvanen 1998]: velika prilagodljivost, ki jo omogočajo mehanizmi za konstruiranje ali prilagajanje metodologij, zahteva dobro poznavanje prednosti in slabosti različnih metodologij, poznavanje bistvenih lastnosti problemske domene, formalne načine za opis posameznih delov metodologije (fragmentov), zapletene postopke za iskanje fragmentov v repozitoriju in pomoč (sodelovanje metodologa) pri sestavljanju fragmentov metodologij v celoto.

Kljub temu, da so bile raziskave na področjih konstruiranja metodologij in situacijskega konstruiranja metodologij v akademskem okolju dolgo časa zelo intenzivne in so še danes, ga inženirji v praksi nikoli niso širše sprejeli [Bajec 2004c; Henderson 2003; Hofstede 1997a]. Omenimo naj tri poglavitne razloge za omenjeno stanje:

- previsoka kompleksnost samih postopkov konstruiranja in prilagajanja, ki so osredotočeni predvsem na metanivo predstavitve znanja o metodologijah,
- pomanjkanje ustreznih programskih orodij, ki bi postopek konstruiranja uspešno podprli,
- visoka cena, ki se odraža v presežku časa, denarja in ljudi, ki so potrebni za konstruiranje oziroma za prilagajanje metodologije pred samim začetkom izvajanja projekta razvoja IS.

1.4 Tematika doktorske disertacije

Ključ do uresničitve uspešne uporabe metodologije v organizacijskem sistemu vidimo predvsem v sami zmožnosti metodologije oziroma njenega razvojnega procesa za prilagajanje specifičnim potrebam projektov, ki se izvajajo v okviru organizacijskega sistema. Postopek prilagajanja je ponavadi zelo kompleksen, zato je obstoj ustreznega programskega orodja, ki bo omogočalo prilagajanje metodologije, predpogoj za njeno dejansko uporabo v okviru projekta. Omenjenih orodij kljub temu še vedno primanjkuje, oziroma so pomanjkljiva. Nobeno od njih ne ponuja celovite rešitve, ki bi omogočala prilagoditi metodologijo na način, da bi ta ustrezala potrebam projekta tako s tehničnega kot sociološkega vidika. To predstavlja povod, zaradi katerega smo se v okviru izdelave doktorske disertacije odločili posvetiti razvoju omenjenega orodja in predhodno sami teoretični opredelitvi odločitvenega modela za prilagajanje, na katerem orodje temelji.

Namen tematike doktorske disertacije je združiti teoretične izsledke raziskav s področja konstruiranja metodologij in znanje pridobljeno na osnovi izkušenj s področja agilnih

pristopov. Ker so agilni pristopi osredotočeni bolj na sam razvojni proces se bomo tudi v okviru tematike disertacije osredotočili na presek med obema področjema. Presek med področjem konstruiranja metodologij in agilnimi pristopi bomo tako opredelili kot znanje o prilagajanju procesa razvoja IS. To hkrati predstavlja temelj našega nadaljnjega dela, v okviru katerega se bomo torej osredotočili samo na razvojni proces in s tem posledično na opredelitev odločitvenega modela za prilagajanje procesa razvoja IS potrebam individualnih projektov.

Na obeh prej omenjenih raziskovalnih področjih se odvijajo intenzivne raziskave v smeri iskanja ustreznih smernic in pristopov, ki bi omogočili prilagoditi metodologijo oziroma njen razvojni proces potrebam konkretnih organizacijskih sistemov ali potrebam posamičnih projektov. Za raziskave s področja konstruiranja metodologij je značilno, da so izrazito teoretično obarvane. Predlagani pristopi za prilagajanje procesa, ki jih predlaga področje konstruiranja metodologij, stremijo predvsem k tehnični ustreznosti skonstruiranega procesa. Za agilne pristope pa je značilno, da črpajo znanje iz pridobljenih izkušenj o uporabi določenega razvojnega procesa. Pri tem zelo poudarjajo sociološko ustreznost uporabljenega procesa, ki ima močan vpliv na sprejetost s strani razvijalcev, ki proces uporabljajo.

Na podlagi povedanega predvidevamo, da lahko s povezavo znanj iz obeh področij opredelimo rešitev, ki bo celovita in bo omogočala prilagoditi proces, ki ga predpisuje uporabljena metodologija tako, da bo ta ustrezal zahtevam, tako s tehničnega kot sociološkega vidika.

1.5 Raziskovalni pristop in metode dela

Izdelava doktorske disertacije je potekala v okviru raziskovalnega dela pod okriljem projekta Centra odličnosti[4] (v nadaljevanju CO). Dva izmed ciljev raziskovalnega dela sta bila tudi opredelitev ustreznega metodološkega pristopa za prilagajanje razvojnega procesa konkretnim potrebam projektov in opredelitev zahtev za izdelavo prototipa programskega orodja za podporo prilagajanja.

Raziskovalno delo v okviru projekta je bilo organizirano v obliki vzajemnega raziskovanja, temelječega na znanstvenih metodah (ang. collaborative practice research), v okviru katerega smo uporabljali kombinacijo naslednjih pristopov:

[4] *Eden od ciljev projekta Centra odličnosti je bil tudi izboljšati metodologije razvoja informacijskih sistemov v slovenskih podjetjih. Projekt je bil sofinanciran s strani slovenskega Ministrstva za visoko šolstvo, znanost in tehnologijo, Evropske komisije in sodelujočih podjetij za razvoj programske opreme.*

- akcijskega raziskovanja (ang. action research) [Avison 1999, Baskerville 1999],
- eksperimentov in
- študije praks v slovenskih podjetjih za razvoj programske opreme, katera so sodelovala na projektu CO.

Za pridobivanje ocen obstoječega stanja na področju uporabe metodologij razvoja programske opreme v podjetjih, smo uporabili tehniko izvajanja intervjujev in tehniko anketiranja [Bajec 2005a]. Cilj tega je bil ugotoviti stopnjo prilagodljivosti razvojnega procesa, ki se uporablja v okviru posameznega podjetja.

Podatke, ki smo pridobili na osnovi ankete in intervjujev, smo dopolnili z akcijskim raziskovanjem. Za vsako sodelujoče podjetje je bila formirana delovna skupina, ki sta jo sestavljala dva raziskovalca in dva praktika. Namen delovne skupine je bil sodelovati na realnem projektu razvoja programske opreme, s čimer smo zagotovili neposredno pridobivanje informacij, ti. iz "prve roke". Praktika sta opravljala vlogo projektnega vodje in metodologa, medtem ko sta bila raziskovalca več ali manj samo opazovalca.

S pomočjo eksperimentov in študije na terenu (ang. field study) se je odvijalo pridobivanje novega znanja za podporo dela v praksi. Za ugotavljanje najvišje ravni razvoja (ang. state of the art) na področju konstruiranja metodologij in njihove uvedbe, smo uporabili metodo pregleda in analize literature [Glass 2002; Schwartz 2004]. V okviru te metode je bila izvedena analiza literature, ki se nanaša na naslednja področja:

- metodologije razvoja IS,
- konstruiranje metodologij,
- situacijsko konstruiranje metodologij,
- agilni pristopi in agilne metodologije ter
- modeliranje procesa.

Možne pristope za integracijo koncepta prilagodljivosti v obstoječe metodologije smo raziskali v sodelovanju z vodji projektov in metodologi. Ti so opredelili dejansko raven želene prilagodljivosti, ki naj bi jih njihove metodologije dosegale. Izboljšave obstoječih metodologij smo predlagali v obliki opredelitve pristopa in orodja za prilagajanje razvojnega procesa konkretnim potrebam projektov. V vsakem, na projektu CO, sodelujočem podjetju smo rezultate raziskave predstavili v obliki enodnevne delavnice.

1.6 Struktura doktorske disertacije

Disertacija poleg uvoda in zaključka zajema tri ključna poglavja:

- *Poglavje 2: Predstavitev raziskovalnega področja*: v drugem poglavju najprej predstavimo širše raziskovalno področje. V sklopu tega podamo najpomembnejše raziskave in dela s področja metodologij razvoja IS, ki so posredno ali neposredno povezana s temo disertacije. Med drugim predstavimo opredelitev pojma metodologije in pomen opredelitve zgradbe metodologije s pomočjo metamodela ter podamo pomen formalnega vidika metodologije za organizacijski sistem. Nadaljujemo s predstavitvijo ožjega raziskovalnega področja, v okviru katerega predstavimo raziskave s področja konstruiranja metodologij in modeliranja procesa. V zaključku poglavja podamo ugotovitve in izhodišča na katerih temelji delo v okviru disertacije.

- *Poglavje 3: Prilagajanje procesa razvoja IS*: tretje poglavje je namenjeno formalni opredelitvi odločitvenega modela za prilagajanje procesa individualnim potrebam projekta. V primerjavi z obstoječimi modeli za prilagajanje procesa razvoja programske opreme potrebam konkretnih projektov, ki razvojni proces obravnavajo z gledišča konstruiranja metodologij, izhajamo v disertaciji iz agilnih pristopov razvoja programske opreme, ki posvečajo več pozornosti samim razvijalcem, sam agilni pristop pa izhaja iz potreb realnega okolja. V tretjem poglavju podamo opredelitev modela postopka za prilagajanje razvojnega procesa, z vidika uporabe odločitvenih pravil, ki usmerjajo postopek konstruiranja prilagojenih različic procesa potrebam individualnih projektov. Tematika tretjega poglavja predstavlja temeljni prispevek k znanosti doktorske disertacije.

- *Poglavje 4: Programsko orodje*: drugi pomembnejši prispevek k znanosti v okviru disertacije predstavlja opredelitev izhodišč za izdelavo prototipa programskega orodja, ki predstavlja pripomoček za podporo dela metodologa pri izgradnji prilagojenih različic procesa. Delo je potekalo v okviru razvojno - raziskovalnega projekta z nazivom "Obvladovanje procesa razvoja pri razvoju rešitev za elektronsko poslovanje", katerega cilj je opredeliti pristop in izdelati celovito računalniško podporo za načrtovanje in razvoj metodologij, ki so tako tehnično kot tudi socialno ustrezne posameznim organizacijskim sistemom oziroma njihovim projektom. Opredelitve izhodišč za izdelavo programskega orodja temeljijo na izsledkih koncipiranja odločitvenega modela za prilagajanje procesa, ki ga obravnavamo v tretjem poglavju.

2 Predstavitev raziskovalnega področja

2.1 Uvod

Prilagajanje procesa razvoja IS ni novost na področju informacijskih znanosti, saj se raziskave na tem področju izvajajo že vrsto let. V tem poglavju so v okviru širšega raziskovalnega področja predstavljene metodologije razvoja IS, formalizacija njenih delov in razvoj metodologij skozi čas, s čimer hočemo podati splošen pregled nad dogajanjem na področju uporabe in prilagajanja metodologij. Povzetek dogajanja in opis najpomembnejših dosežkov iz področja konstruiranja metodologij ter samega razvojnega procesa programske opreme predstavljajo ožje področje doktorske disertacije, ki je predstavljeno v razdelku 2.3. V razdelku 2.4 so podane ugotovitve na katerih temelji delo doktorske disertacije ter cilji, ki jih v njenem okviru želimo doseči.

2.2 Širše raziskovalno področje

2.2.1 Uvod

Opis širšega raziskovalnega področja pričnemo s predstavitvijo metodologij razvoja IS. V okviru tega najprej podamo opredelitev pojma metodologije in nadaljujemo s predstavitvijo zgradbe metodologije, kjer podamo formalen način za opis zgradbe metodologije s pomočjo metamodela. V tem razdelku tudi že nakažemo pomen in možnosti uporabe metamodelov za potrebe kasnejšega prilagajanja procesa. Sledi umestitev metodologije v okolje in opis njunega medsebojnega vpliva. V nadaljevanju predstavimo proces formalizacije znanja o metodologiji in pomen formalizirane metodologije za organizacijski sistem. Podpoglavje zaključimo z pregledom razvoja metodologij skozi čas in predstavitvijo agilnih metodologij. V okviru tega predstavimo tudi opredelitev pojma agilnosti, kot ga uporabljamo v nadaljevanju disertacije.

2.2.2 Metodologije razvoja IS

2.2.2.1 Opredelitev pojma metodologija razvoja IS

Pojem metodologije razvoja IS različni avtorji razlagajo dokaj različno. Za izhodišče lahko privzamemo splošno opredelitev iz *Oxford Advanced Learner's Dictionary of*

Current English [Oxford 2000], ki besedo metodologija opredeli kot "skupek metod in principov za izvrševanje določene aktivnosti". Pri tem je metoda opredeljena kot "poseben način na katerega nekaj naredimo".

Ena izmed prvih opredelitev celotnega pojma metodologija razvoja IS, ki se uporablja še danes, izhaja iz leta 1983 in jo je oblikoval Maddison [Maddison 1983]: "Metodologija je priporočena zbirka filozofij, faz, postopkov, pravil, tehnik, orodij, dokumentacije, upravljanja in izobraževanja za razvijalce IS".

Avison kasneje poda dopolnjeno definicijo metodologije razvoja IS [Avison 2003]: "Metodologija razvoja IS je zbirka postopkov, tehnik, orodij in izdelkov, ki razvijalcem pomagajo pri razvoju novega IS. Metodologijo sestavljajo postopki, ki so sestavljeni iz podpostopkov, ki vodijo razvijalce pri izbiri tehnik, primernih v posameznih stopnjah projekta in tudi pomagajo načrtovati, upravljati, nadzorovati in ocenjevati projekte razvoja IS". Avison je naknadno svojo definicijo še razširil, tako da je dodal naslednje: "Metodologija razvoja IS je priporočen način za celoten razvoj ali le del razvoja IS, ki temelji na filozofiji in množici principov, ki podpirajo, opravičujejo in pojasnjujejo priporočila metodologije za določen kontekst. Navadno vključuje faze, postopke, naloge, pravila, tehnike, smernice, dokumentacijo in orodja. Poleg tega lahko vključuje tudi priporočila v zvezi z vodenjem in organizacijo dela ter identifikacijo in znanjem udeležencev v postopkih".

Pojem metodologije razvoja programske opreme definira tudi Cockburn [Cockburn 2002]. Opredeli jo kot "niz povezanih metod in tehnik", pri čemer je metoda definirana kot "sistematičen pristop k delu". Podobno kot Avison tudi Cockburn svojo definicijo metodologije kasneje razširi ter metodologijo razvoja programske opreme opredeli kot vse kar redno delamo, da bi dosegli končni rezultat – torej izdelano in delujočo programsko opremo pri končnem uporabniku. Pri tem ne gre samo za postopke, ki so ozko povezani z razvojem programske opreme, ampak tudi ostale podporne postopke, načine komunikacije med ljudmi, razporeditev delovnih mest razvijalcev, porazdelitev odločanja med člane skupine, kulturo organizacije itd. Iz povedanega lahko tudi povzamemo, da [Cockburn 2002]: "Metodologija je množica dogovorov (konvencij), s katerimi se (projektna) skupina/organizacija strinja".

Omeniti moramo še razliko med *procesom razvoja IS* in *metodologijo razvoja IS*, kar je še posebej pomembno za nadaljevane doktorske disertacije, v kateri se osredotočimo na razvojni proces. Čeprav se omenjena pojma pogosto zamenjuje, načeloma velja, da je proces razvoja programske opreme bolj ozko opredeljen in neposredno povezan z razvojem programske opreme. Proces je pravzaprav le nabor postopkov, aktivnosti, izdelkov itd. in ga ponavadi obravnavamo kot del metodologije. Metodologija

predstavlja širši pojem od pojma proces, saj poleg procesa obsega še dodatne koncepte, kot so filozofija, kultura, paradigma itd.

2.2.2.2 Zgradba metodologije

Metodologija in proces razvoja programske opreme, ki ga ta predpisuje, sta bila vedno predstavljena v obliki, ki je primerna za njene uporabnike (razvijalce). Metodologija predpisuje aktivnostih, tehnike in življenjski cikel razvoja programske opreme ter kako naj bi bili omenjeni procesni elementi časovno organizirani in kdo naj bi jih izvajal. Opisi metodologij se pogosto nahajajo v obliki knjig ali navodil, ki jim vodje projektov skrbno sledijo.

S prihodom orodij CASE se je znotraj posameznega orodja pojavila potreba po vzpostavitvi pravil, ki bi podpirala razvojni proces. S pomočjo teh pravil bi lahko naprimer opredelili, ali je možno dve aktivnosti izvesti zaporedno, ali je za izvajanje aktivnosti predpisana kakšna od tehnik itd. Taka pravila se običajno nahajajo v metamodelu metodologije.

Za opis zgradbe metodologij so metamodeli zelo uporabni, saj omogočajo predstaviti koncepte (gradnike metodologije), pravila in relacije med gradniki določene metodologije [Henderson 2005]. Pomembna lastnost metamodela je tudi ta, da lahko na njegovi osnovi opredelimo celo družno sorodnih metodologij. Čeprav je metodologijo možno opisati tudi brez eksplicitnega metamodela, pa je formalizacija metodologije z uporabo metamodela koristna, če hočemo preveriti njeno konsistentnost ali če planiramo njene razširitve ali prilagoditve.

Metamodele uporabljajo metodologi pri konstruiranju ali prilagajanju metodologije [Henderson 2005]. Na podlagi metamodela zgrajena metodologija nato služi kot osnova za prilagoditve metodologije na nivoju projektov. Na nivoju projekta prilagojena metodologija se nato uporabi za izdelavo programskih izdelkov. Metamodel metodologije postavlja omejitve, ki jih mora upoštevati metodolog pri izdelavi metodologije, ta pa naprej postavlja omejitve glede načina dela, ki ga predpisuje prilagojena metodologija na nivoju projekta. Odnos med modelirnimi nivoji predstavimo z relacijo "je primerek", kar bo podrobneje prikazano v nadaljevanju disertacije.

Uporabo metamodelov, ki podpirajo objektno orientirane razvojne procese, je začel prakticirati konzorcij OPEN, in sicer v sredini 1990ih let, rezultat česar predstavlja današnja oblika ogrodja procesa OPEN [Henderson 2005]. Kasneje mu je sledila še

skupina Object Management Group, ki je razvila visokonivojski standard za opredeljevanje procesa. Sočasno je potekal tudi razvoj metamodela OOSPICE, ki je bil izdelan s strani skupnosti za ocenjevanje zrelosti procesa in razvoj metamodela LiveNet.

Zaradi številnih avtorjev, ki so sodelovali pri snovanju omenjenih metamodelov, je že prišlo tudi do prenosa idej med njimi, zaradi česar Henderson in Gonzalez v svojem prispevku [Henderson 2005] predlagata osnovanje enotnega metamodela za metodologije razvoja programske opreme SMSDM[5]. Namen tega je opredelitev enotnega standarda, ki združuje vse najboljše lastnosti prej omenjenih metamodelov, katere so podrobno predstavljene v prispevku [Henderson 2005].

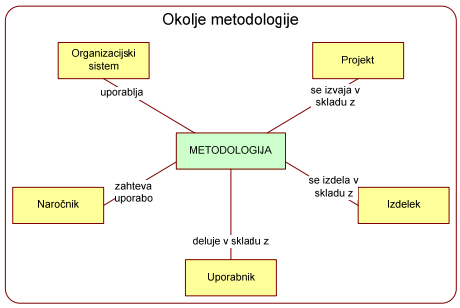
2.2.2.3 Umestitev metodologije v okolje

Metodologija razvoja programske opreme ne more obstajati sama zase in biti izolirana od okolice [Vavpotič 2005]. Metodologija se vedno nahaja v določenem okolju (organizacijskem sistemu), ki ga opredeljujejo entitete, ki so z metodologijo neposredno ali pa posredno povezane. Ko govorimo o metodologiji, jo vedno obravnavamo v kontekstu določenega okolja. Za entitete, ki se z metodologijo neposredno povezujejo velja, da vplivajo na metodologijo in/ali so od metodologije odvisne:

- Organizacijski sistem izvaja svoje delo v skladu z določeno metodologijo.
- Uporabnik metodologije je vsak posameznik, ki metodologijo uporablja pri izvajanju svojih delovnih nalog.
- Projekti se izvajajo v skladu z metodologijo, ki je sprejeta v organizacijskem sistemu.
- Naročnik vpliva na metodologijo, tako da zahteva uporabo določene metodologije.
- Rezultate projekta predstavljajo končni izdelki - dokumentirana programska koda, ki je izdelana v skladu z uporabljeno metodologijo.

Umestitev metodologije v okolje prikazuje razredni diagram na sliki (glej Slika 2):

[5] *Standard Metamodel for Software Development Methodologies*



Slika 2: Umestitev metodologije med entitete, ki opredeljujejo njeno okolje [Vavpotič 2005]

2.2.2.4 Medsebojni vpliv metodologije in okolja

Metodologija, ki se uporablja v nekem okolju, naj bi temu okolju v čim večji meri ustrezala, tako da se mu glede na njegove karakteristike prilagodi. Metodologijo je mogoče prilagajati potrebam celotnega organizacijskega sistema, potrebam individualnih projektov, posamičnim uporabnikom in zahtevam naročnika. Pravimo da našteti elementi vplivajo na metodologijo. Velja pa tudi obratno, in sicer, da metodologija vpliva na okolje. Metodologija ima lahko namreč vpliv na značilnosti organizacijskega sistema in projekte, ki jih ta izvaja, zahteve naročnika ter končni izdelek. V okviru disertacije nas bo še posebej zanimal vpliv lastnosti projektov, ki se izvajajo v okviru določenega organizacijskega sistema, na možnost prilagajanja metodologije oziroma procesa, ki ga ta opredeljuje.

2.2.2.5 Teža metodologije

Delitev metodologij na težke in lahke metodologije se je pojavila proti koncu 1990ih, ko so se kot kritika obsežnih formalnih metodologij začeli uveljavljati novi pristopi k razvoju programske opreme (glej razdelek 2.2.4.4).

Teža metodologije [Cockburn 2000a, 2000b in 2002] je opredeljena kot produkt njenega obsega in gostote:

$$TEŽA\ METODOLOGIJE = OBSEG\ x\ GOSTOTA$$

- **Obseg** metodologije je določen s številom različnih elementov, ki jih metodologija opisuje. Obseg opredeljuje, katere osnovne in podporne postopke neka metodologija obsega, katere vloge so v njej opisane, katere standarde upošteva itd.
- **Gostota** metodologije je opredeljena kot zahtevan nivo podrobnosti oziroma formaliziranosti opisa njenih elementov. Metodologije z višjo gostoto so bolj

formalne, njihovi elementi pa so opisani do višje stopnje podrobnosti. Metodologije z nižjo gostoto prepuščajo več stvari interpretaciji posameznika, ki metodologijo uporablja.

2.2.2.5.1 Značilnosti lahkih metodologij

Lahke metodologije so tiste z manjšim obsegom in predvsem z manjšo gostoto. Za njih je značilno, da se osredotočijo predvsem na razvoj programske rešitve, pri čemer pa ponavadi zapostavijo druge vidike sistema, kot npr. organizacijski, poslovni, sociološki itd. Lahke metodologije so primerne za okolja, kjer je ključni cilj razvoj programske rešitve. Temelj lahkih metodologij predstavljajo značilnosti, kot so dobro sodelovanje med razvijalci, disciplina pri delu ter znanje razvijalcev pa tudi drugih udeležencev na projektu. Pogosto uporabljajo posebne tehnike dela z ljudmi, kot so npr. programiranje v parih ali pa JAD (Joint Application Development) ipd. Posebnost lahkih metodologij je relativno visoka stopnja prilagodljivosti, kar jim omogoča sprotno odzivanje na spremembe v zahtevah oziroma v okolju (glej razdelek 2.2.2.3). Zaradi nižje stopnje formalnosti so primernejše za manjše projektne skupine, ki tako niso obremenjene s formalnostjo težkih metodologij, ki je pri večjih projektnih skupinah potrebna. Kot primere lahkih metodologij lahko navedemo: ASD - Adaptive Software Development, XP-Extreme Programming, FDD - Feature-Driven Development, Crystal Clear, SCRUM, DSDM - Dynamic System Development Method idr.

Čim lažja je metodologija, tembolj prilagodljiva je in manj optimizirana. Lahke metodologije so primerne za uporabo ko [Cockburn 2002; Highsmith 2002c]:

- je glavni (in dokončno opredeljen) cilj razvoj programske rešitve,
- imamo na voljo odgovorne, disciplinirane, izkušene ter motivirane razvijalce, ki so vešči s posebnimi tehnikami dela,
- stranka razume smisel lahkih metodologij in je pripravljena na sodelovanje,
- predvidevamo nepredvidljive in spreminjajoče zahteve glede programske rešitve,
- je cilj razvoja relativno *majhen* sistem, katerega *stopnja kritičnosti je nizka* in ga je *mogoče razviti z majhno razvojno ekipo*,
- itd.

2.2.2.5.2 Značilnosti težkih metodologij

Metodologije z večjim obsegom in večjo gostoto spadajo v skupino težkih metodologij. Temeljijo na večji stopnji formaliziranosti, njihovi procesi pa so optimizirani do precej višje stopnje kot so optimizirani procesi lahkih metodologij. Težke metodologije podrobno opisujejo izdelavo dokumentacije in pogosto predpišejo tudi uporabo

določenih orodij in tehnik. Primerne so predvsem za uporabo v večjih skupinah razvijalcev, pa tudi za izgradnjo kritičnih programskih rešitev, ki seveda zahtevajo višjo raven rigoroznosti. Nekatere težke metodologije se ukvarjajo tudi z dodatnimi vidiki razvoja programske opreme, kot sta npr. sociološki in organizacijski.

Za težke metodologije velja, da so relativno visoko optimizirane in da temeljijo na procesu, formalnosti in na dokumentaciji. Velja, da čim težja je metodologija, tembolj optimizirana je in manj prilagodljiva. Primeri težkih metodologij: IE, RUP, SSM, ETHICS, SSADM, EMRIS [Krisper 2004a in 2004b] (metodologija razvita za potrebe državne uprave) itd.

Težke metodologije so primerne za uporabo ko:

- cilj ni samo razvoj programske rešitve, ampak tudi prenova organizacijskega sistema,
- imamo prisotne manj izkušene razvijalce, pri katerih izkušnje in znanje nadomeščajo točno opredeljena formalna pravila,
- sam naročnik zahteva visoko stopnjo formalizma (izdelovanje dokumentacije),
- imamo dobro opredeljene in relativno stabilne zahteve,
- je cilj razvoja obsežnejši sistem, katerega stopnja kritičnosti je višja in zahteva izdelavo ustreznih načrtov in dokumentacije,
- itd.

2.2.3 Formalizacija metodologije in njen pomen

2.2.3.1 Uvod

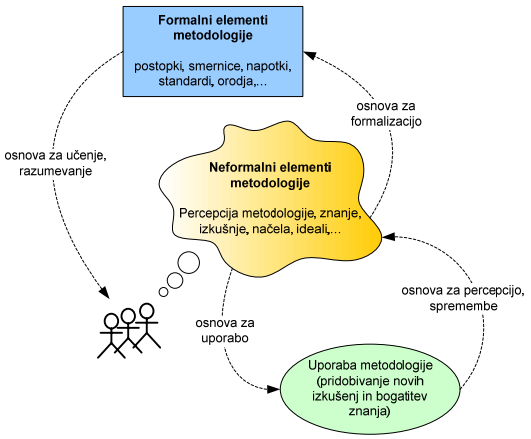
Metodologija razvoja IS, ki se uporablja v nekem organizacijskem sistemu ni vedno formalno zapisana. Tudi organizacijski sistemi, ki nimajo uvedene formalne metodologije, delujejo v skladu z določenimi neformalnimi pravili, ki temeljijo predvsem na pridobljenih izkušnjah in znanju članov razvojne skupine. V tem primeru govorimo o tacitnem[6] znanju zaposlenih v organizacijskem sistemu. Tudi v organizacijskih sistemih, ki imajo izdelano formalno metodologijo, razvijalci tej metodologiji pogosto ne sledijo v celoti, ampak uporabljajo lastne ali poenostavljene postopke, nastale na podlagi izkušenj. Lahko torej rečemo, da je *metodologija*

[6] *Skrito znanje, ki se nahaja v glavah zaposlenih.*

organizacijskega sistema tista metodologija, ki jo njeni uporabniki dejansko izvajajo oziroma jo imajo v glavah.

Temeljni elementi metodologije se torej skrivajo ravno v znanju in izkušnjah posameznikov. Znanje lahko tako opredelimo kot *eksplicitno*, ki ga je možno izraziti v neki formalni obliki in skrito ali *tacitno* znanje, ki je navadno subjektivne narave in prepleteno z izkušnjami, ideali, čustvi, intuicijo itd., zaradi česar ga je težko izraziti. Vendar pa se tudi skrito znanje lahko skozi uporabo rutinizira in postane tako izrazljivo. Tudi na področju metodologij ima omenjen fenomen svoj pomen. Dokumentirana metodologija je sprva osnova, iz katere se uporabniki učijo in razvijejo svojo percepcijo o načinu dela, odločanja, komunikaciji itd. Nato skozi uporabo pridobivajo nove izkušnje in bogatijo svoje znanje, kar vpliva tudi na metodologijo samo. Ko postane znanje, ki ga uporabljajo za izvajanje svojih nalog, dovolj rutinsko, pridobi obliko podatkov in tedaj lahko postane del formalne metodologije [Maier 2000]. Rečemo lahko torej, da se metodologija bogati skozi uporabo na osnovi skritega znanja njenih uporabnikov. Iz tega znanja se sčasoma lahko izluščijo formalne oblike (metode, postopki, smernice, napotki) ter postanejo del formalne metodologije. Pri tem se je potrebno zavedati, da je skrito znanje, ki je osnova za nastanek eksplicitnega znanja, vedno bogatejše kot pa stvaritev sama [Morabito 2001; O'Dell 1996]

Slika 3 prikazuje cikel, ki poteka od uporabe formalne metodologije kot osnove za učenje ter razumevanje metodologije do formalizacije elementov neformalne metodologije, ki se skozi uporabo izkažejo dovolj rutinski za standardizacijo.



Slika 3: Od neformalne do formalne metodologije [Bajec 2004a]

Kot smo videli, je mogoče formalizirati tudi neformalne postopke, ki jih razvijalci uporabljajo in z njimi nadgraditi obstoječo metodologijo. Pri tem je pomembno, da

pravilno postavimo mejo med formalnim in neformalnim delom metodologije. Če je formalni del metodologije preobsežen, ga težko vzdržujemo, zaradi česar metodologija hitro zastara in več ne ustreza dejanskim potrebam organizacijskega sistema, oziroma je razvijalci ne sprejmejo. Metodologija je namreč dinamična in se kot taka stalna spreminja. Po drugi strani pa je obsežna neformalna metodologija tvegana za organizacijski sistem, saj je popolnoma odvisna od njenih uporabnikov [Bajec 2004a in 2004b].

Številni organizacijski sistemi (podjetja, ki se ukvarjajo z razvojem IS) še vedno razvijajo programsko opremo na osnovi metodologij, ki niso formalno opredeljene, torej metodologij, ki niso dokumentirane. Čeprav so postopki, ki jih izvajajo, znani in ustaljeni, jih podjetja redko zapišejo eksplicitno. Še bolj poredko pa svoje metodologije osvežujejo v smislu zajema pridobljenih izkušenj in znanj, do katerih pridejo skozi uporabo metodologij.

Prava stopnja formalizacije metodologije nam lahko prinese precej prednosti. Razlogov za uvedbo formalne metodologije in prednosti, ki jih s tem pridobimo je opredelil Avison [Avison 2003]. Možne razloge za uvedbo metodologije smo združili v naslednje skupine:

- doseganje boljših poslovnih učinkov (izdelki ali storitve),
- izboljšanje razvojnega procesa (dvig storilnosti in izboljšan nadzor),
- standardizacija poslovanja,
- lažje delo s kadri in lažje uvajanje novih kadrov ter
- ustvarjanje vtisa na naročnika.

V teh skupinah so zajeti le najosnovnejši razlogi za vpeljavo formalne metodologije v organizacijski sistem, zato se moramo zavedati, da so možni tudi drugačni razlogi.

2.2.3.2 Doseganje boljših poslovnih učinkov

Prvi izmed naštetih razlogov za uvedbo formalne metodologije je pričakovanje, da bo končni poslovni učinek (programska rešitev) bolj kakovosten. Potrebno se je zavedati, da je zelo težko izmeriti vpliv uporabe metodologije na kakovost končnega izdelka. Kljub temu se je v praksi izkazalo, da so sistemi razviti z uporabo formalne metodologije pogosto bolj kakovostni.

2.2.3.3 Izboljšanje razvojnega procesa

Z uporabo metodologije se lahko doseže boljše upravljanje in nadzor nad projektom. Uporaba metodologije nam omogoča tudi lažji nadzor nad izdelki, ki v okviru razvoja nastajajo. Ustrezna metodologija naj bi ob enakem ali celo manjšem številu razvijalcev tudi pohitrila razvoj sistema [Cockburn 2002]. Po nekaterih trditvah naj bi se ob uporabi ustrezne metodologije znižala stopnja potrebnega znanja članov projektne skupine.

Kljub temu lahko najdemo tudi vrsto nasprotnih trditev. Tako lahko uporaba neprimerne metodologije ali pa neprimerna uporaba metodologije upočasni in podraži razvoj, povzroči, da ne razmišljamo o pravem cilju itd.

2.2.3.4 Standardizacija poslovanja

Uvedba standardov, ki jih lahko vključuje metodologija, organizaciji prinaša vrsto prednosti. Tako lahko zasnujemo skupno bazo znanja, zaposlene lažje razmeščamo na različne projekte, lažje preverjamo kakovost izdelkov, ki morajo ustrezati standardom, lažje vzdržujemo sistem itd.

2.2.3.5 Lažje delo s kadri in lažje uvajanje novih kadrov

Uporaba metodologije omogoča lažje delo s kadri. Opis in razmejevanje odgovornosti posameznih vlog nam namreč omogoča, da lahko natančno opredelimo naloge neke osebe.

Tudi uvajanje novih ljudi v metodologijo razvoja programske opreme je s pomočjo zapisanih postopkov in skupne baze znanja lažje. Prav tako je lažja zamenjava in nadomeščanje kadrov, saj je čas uvajanja novih kadrov krajši. Lažje je tudi nadomeščanje znanja (tacitno znanje), ki ga izgubimo z odhodom človeka.

Formalni opis metodologije je dobra osnova za poučevanje metodologije in izdelavo izobraževalnih tečajev. Ko poznamo standarde in tehnike, vloge in njihove odgovornosti itd. je veliko lažje oblikovati ustrezne tečaje in skupine, ki se jih bodo udeležile.

2.2.3.6 Ustvarjanje vtisa na naročnika

Dober vtis na potencialnega naročnika je mogoče ustvariti tudi z dobro opisano metodologijo. Na ta način dobi naročnik občutek varnosti. Prikaz napredovanja projekta

je povezan z ustvarjenjem dobrega vtisa na naročnika. S pomočjo formalnega opisa metodologije naročniku lažje predstavimo kje smo in kaj delamo. Pri tem je potrebno omeniti, da naročnik pogosto zahteva uporabo formalne metodologije.

2.2.4 Pregled razvoja metodologij skozi čas

Metodologije razvoja IS so se razvijale hkrati z razvojem IT, programskih jezikov itd. Tako lahko celotno zgodovino razvoja metodologij v grobem razdelimo na štiri obdobja, ki so predstavljena v nadaljevanju [Avison 2003a in 2003b; Eriksen 2000; Fitzgerald 1997; Vavpotič 2003]. Pri tem je potrebno opozoriti, da je delitev izvedena na podlagi časa nastanka metodologij in ne na podlagi časa njihove uporabe. Tako tudi danes mnogo organizacijskih sistemov ne uporablja nobene izmed formalnih metodologij.

2.2.4.1 Obdobje pred pojavom metodologij

Prvo obdobje uporabe metodologij sega v leta pred pojavom formalnih metodologij. Gre za obdobje, ki je trajalo od začetka uporabe računalniške tehnologije v poslovne (in tudi druge) namene pa do začetka 70ih let prejšnjega stoletja. IS iz tistega časa so bili razviti brez uporabe formalnih metodologij. Poudarek je bil na programiranju in reševanju tehničnih problemov, ki so izhajali iz močno omejenih zmogljivosti strojne opreme. Razvijalci so bili po eni strani dobro tehnično izobraženi ter so poznali vse podrobnosti strojne opreme in programskega jezika, po drugi strani so le redko dobro razumeli poslovanje in kontekst organizacije za katero so razvijali IS. Hkrati jim je ozko tehnično usmerjena izobrazba oteževala komunikacijo z ne-tehnično izobraženim naročnikom.

Proces razvoja je bil tipično individualno delo, ki je temeljilo na izkušnjah in sposobnostih posameznih programerjev. Vodenje in ocenjevanje projektov razvoja se je izvajalo po principu "čez prst". Posledica takega razvoja je bil nezadosten nadzor, rezultat pa aplikacije, ki pogosto niso ustrezale naročnikovim zahtevam.

2.2.4.2 Zgodnje obdobje metodologij

Zgodnje obdobje metodologij zaznamujejo leta od poznih 70ih do zgodnjih 80ih let prejšnjega stoletja. V tem času so se pričele pojavljati prve formalne metodologije, ki so se osredotočale predvsem na opredelitev posameznih faz in postopkov pri razvoju IS. Razbitje razvoja na več postopkov naj bi omogočilo večji nadzor in lažje upravljanje

projektov razvoja programske opreme. Leta 1970 je Winston W. Royce predstavil t.i. slapovni model razvoja poznan tudi kot SDLC[7]. SDLC je sestavljen iz več postopkov, ki si sledijo zaporedno brez učinkovitih možnosti vračanja v predhodne postopke. Izdelanih je bilo več različic SDLC, ki so vsebovale različne postopke, vendar so osnovni postopki vseh metodologij, ki temeljijo na SDLC, podobni. Najvidnejši avtorji tega obdobja so prej omenjeni W. Royce, B. W. Boehm, ki je predstavil spiralni model razvoja in E. Codd, ki je zasnoval relacijski podatkovni model.

2.2.4.3 Obdobje metodologij

Od sredine/konca 80ih pa do sredine/konca 90ih let prejšnjega stoletja se je pojavila vrsta novih pristopov k razvoju programske opreme, kot kritika slabosti in omejitev modela SDLC. V tem času se tudi prvič uporabi pojem "metodologija razvoja programske opreme" v današnjem pomenu. Večina metodologij, ki so nastale v zgodnejšem obdobju, se je opirala na eno ali nekaj povezanih tehnik, ki so predstavljale temelje metodologije. To je bila najpogosteje tehnika podatkovnega modeliranja ali tehnika diagramov podatkovnih tokov. Kasneje so avtorji šele začeli širiti obseg metodologij, z dodajanjem novih tehnik, postopkov in aktivnosti. Večino teh metodologij so kasneje avtorji še nadgrajevali in izpopolnjevali na podlagi izkušenj, ki so jih pridobivali ob uporabi metodologij na konkretnih projektih.

Tudi v svetovalnih hišah začnejo nastajati metodologije, ki predstavljajo osnovo za svetovanje podjetjem, ki se ukvarjajo z razvojem IS. Sčasoma so te hiše zaznale možnost, da bi metodologije lahko tržile v obliki komercialnih izdelkov. Komercialne metodologije zagotavljajo, da so vsi njeni deli zapisani, konsistentni, podrobno opredeljeni, vzdrževani in ažurni.

Obseg postopkov, ki jih metodologije podpirajo se je razširil. Bistvenega pomena je bila prav razširitev metodologij na področje strateškega planiranja. Predhodno so se metodologije uporabljale predvsem za izdelavo ožjih programskih rešitev, ki so ponavadi podprle le en ali samo del naročnikovega poslovnega procesa. Organizacijski sistemi pa nimajo le enega poslovnega procesa [Mihelčiči 1999 in 2000], ki bi ga bilo mogoče računalniško podpreti. Organizacijski sistemi, ki so posamezne probleme reševali ločeno, so kmalu ugotovili, da kljub temu, da so te posamezne probleme do neke mere rešili, obstoj velikega števila ločenih programskih sistemov ni vodil v harmonizacijo, niti k izboljšanju poslovnih procesov. Zaradi tega so pričeli IS

[7] *Systems Development Life Cycle*

obravnavati kot bistveni sestavni del organizacijskega sistema, ki ima lahko močan vpliv na njen uspeh ali neuspeh.

Poleg tega so metodologije poskušale zajeti tudi druge spremembe, do katerih je v tem času prišlo na področju razvoja programske opreme. Ena izmed bistvenih novosti je bila uporaba objektno usmerjenih programskih jezikov. Komponentni razvoj in ponovna uporaba, kot posledica objektne usmerjenosti, sta ponujala vrsto novih možnosti tudi pri razvoju poslovnih aplikacij. Metodologije so prevzele tudi nove objektne tehnike kot npr. jezik UML[8].

Do sprememb je prišlo tudi v življenjskem ciklu razvoja programske opreme. Večina komercialnih metodologij je prevzela iterativni in inkrementalni življenjski cikel. Za ključne avtorje iz tega obdobja lahko navedemo: G. Booch, C. Rumbaugh, I. Jacobson, J. Martin, E. Yourdon, E. Gamma, itd.

Komercialne metodologije so postajale vedno bolj obsežne in zahtevne za razumevanje. Opredeljevale so veliko število vlog, postopkov, faz, aktivnosti itd., ki pa pri manjših projektih sploh niso bile potrebne in so povzročale dodatno (nepotrebno) delo. Po mnenju številnih kritikov omenjenih metodologij, naj bi bili razvijalci, ki so te metodologije uporabljali, preveč zaposleni z izdelovanjem raznovrstne dokumentacije namesto, da bi se osredotočili na razvoj programske rešitve. Na podlagi množice kritik obsežnih metodologij se je na sredi/koncu 1990ih let začelo novo obdobje - obdobje ponovne ocenitve metodologij.

2.2.4.4 Obdobje ponovnega ovrednotenja metodologij

Za zadnje obdobje, ki traja od sredine/konca 1990ih let, je značilna močna kritika obsežnih formalnih metodologij (težke metodologije) razvoja IS. V tem času se je pojavila vrsta novih metodologij (lahke in agilne metodologije), ki v nasprotju s težkimi metodologijami ne skušajo do podrobnosti pokriti vseh postopkov razvoja programske opreme. Lahke metodologije navadno dajejo večji poudarek implementaciji in testiranju ter manjši poudarek analizi in načrtovanju.

Do sprememb je prišlo tudi pri uporabi metodologij v organizacijah. Tako so nekatere organizacije pričele posegati po lažjih metodologijah, ali pa so celo prenehale uporabljati formalne metodologije ter se vrnile k ad-hoc pristopom. Tudi komercialni ponudniki težkih metodologij so se pričeli zavedati prednosti lažjih metodologij, zato je

[8] *Unified Modelling Language*

marsikateri izmed njih pripravil lažjo različico svoje metodologije, namenjeno posebej za manjše projekte (kot primer glej vir [Hirch 2002]).

Vzrokov za zavračanje obsežnih formalnih metodologij je veliko. Po eni strani so posledica težav, s katerimi so se soočala nekatera podjetja, ki so poskušala takšne metodologije vpeljati, po drugi strani pa prenapihnjenih trditev njihovih ponudnikov o prednostih, ki naj bi jih njihove metodologije prinašale.

V obdobju ponovnega ovrednotenja metodologij ima še posebej velik pomen ti. trend agilnosti. Agilnost v slovenskem jeziku označujemo kot gibčnost ali prilagodljivost in priporoča prilagajanje metodologij potrebam organizacijskih sistemov in potrebam posameznih projektov. Sama ideja, ki jo promovirajo agilni pristopi je v precejšnji meri podobna ideji situacijskega konstruiranja metodologij, le da prihaja iz prakse. Glavni avtorji, ki jih lahko povežemo z obdobjem ponovnega ovrednotenja metodologij so: K. Beck, S. Ambler, A. Cockburn, J. Highsmith, G. Fitzgerald, Henderson itd.

Najpomembnejše vzroke za težave, ki nastajajo pri uporabi metodologij v današnjem času, lahko tako strnemo v naslednje točke:

- Slaba zasnova metodologij in s tem nezmožnost za prilagajanje dejanskim potrebam organizacijskih sistemov in potrebam samih projektov, ki se v okviru teh izvajajo.
- Uporaba metodologije, ki s tehničnega vidika ne ustreza potrebam organizacijskega sistema ali projekta [Vavpotič 2005].
- Uporaba metodologije, ki s socialnega vidika ne ustreza potrebam organizacijskega sistema ali projekta [Vavpotič 2005].
- Neustrezna in nepopolna vpeljava metodologije v organizacijski sistem ter kot posledica tudi napačna uporaba metodologije.

Vidimo, da vzroki za težave pri uporabi metodologij izvirajo predvsem iz tehnične in sociološke neustreznosti izbranih metodologij in nezmožnosti prilagajanja potrebam organizacijskih sistemov in projektov.

2.2.4.5 Trenutni trendi na področju razvoja metodologij

Trenutni trendi na področju razvoja metodologij za razvoj IS, ki jih navaja tudi Vavpotič [Vavpotič 2005] napovedujejo, oziroma dajejo že sedaj večji pomen analizi in načrtovanju, saj naj bi bil sam postopek prehoda iz analitičnih modelov v programsko kodo v veliki meri avtomatiziran. Za postopek pretvorbe iz modelov v programsko kodo naj bi skrbela visoko zmogljiva programska orodja, ki bi pri pretvorbi uporabljala

standardne gradnike (ogrodja, vzorci ipd.). Industrializacija procesa razvoja programske opreme (tovarne programske opreme) je trenutni trend, za katerega obstaja velika verjetnost, da bo imel velik vpliv na nadaljnji razvoj metodologij. Omenjeni trendi so se pojavili v sredini/koncu leta 2000 in še vedno trajajo.

Tabela 1 podaja strnjen pregled vseh, v predhodnih razdelkih, predstavljenih obdobij razvoja metodologij razvoja IS in v zadnjem stolpcu navaja trenutne trende in smernice na področju razvoja programske opreme.

Obdobje pred pojavom metodologij (do začetka 70')	**Zgodnje obdobje metodologij** (od začetka 70' do zgodnjih 80')	**Obdobje metodologij** (od sredine/konca 80' do sredine/konca 90')	**Obdobje ponovnega ovrednotenja metodologij** (od sredine/konca 1990)	**Obdobje industrializacije** (od sredine/konca 2000)
• ni formalnih metodologij • poudarek je na reševanju tehničnih problemov, ki izhajajo iz omejene zmogljivosti strojne opreme in na programiranju • pomanjkljiv zajem zahtev • nezadosten nadzor in vodenje projektov • ključna vloga pri razvoju PO[9] je programer	• slapovni razvoj (SDLC[10]) • metodologije so zasnovane na tehnikah podatkovnega modeliranja in procesnega modeliranja • večji poudarek na zajemu zahtev, analizi in načrtovanju • izdelava vrste postopkov za razvoj PO • ključna vloga postane sistemski analitik • celoten IS podjetja sestavlja več nepovezanih aplikacij	• iterativni in inkrementalni razvoj • RAD[11] (rapid application development) • oblikovanje metodologij kot komercialnih produktov • večanje obsega in gostote metodologij • metodologije pričnejo pokrivati tudi strateško načrtovanje • metodologije omogočajo izdelavo integriranih IS • vključevanje vrste novih tehnik in notacij • podpora objektno usmerjenim jezikom in komponentnemu razvoju	• močna kritika obsežnih formalnih metodologij razvoja • pojav vrste lahkih metodologij, ki dajejo večji poudarek programiranju in testiranju • nekatera podjetja opuščajo uporabo težkih metodologij • razvoj aplikacij za splet • pogosto spreminjanje zahtev med razvojem PO • hitro pojavljanje novih tehnologij • agilnost kot trend pri metodologijah razvoja PO	• industrializacija procesa razvoja programske opreme • tovarne programske opreme • uporaba standardnih gradnikov (ogrodja, vzorci) • velik pomen orodij za generiranje modelov in kode • ponovno večji pomen analize in načrtovanja ter manjši pomen programiranja • razvoj PO na višjem nivoju abstrakcije • MDD[12] in MDA[13]

Tabela 1: Značilnosti obdobij in metodologij razvoja IS skozi čas [Vavpotič 2005]

[9] *Programska oprema*
[10] *Systems Development Life Cycle*
[11] *Rapid Application Development*
[12] *Model Driven Development*
[13] *Model Driven Architecture*

2.2.5 Agilni pristopi

S pojavom agilnih pristopov [Cockburn 2002; Highsmith 2002a] sama ideja o konstruiranju metodologij sicer ni postala nič bolj izvedljiva. Konstruiranje metodologij je še vedno zelo kompliciran postopek. Je pa res, da je zanimanje s strani prakse postalo veliko večje, kar odpira mnoga vrata. Pobuda o konstruiranju je tokrat prišla s strani inženirjev, ki delujejo v praksi (realnem razvojnem okolju). Bistvo agilne metodologije se skriva v njeni zmožnosti za prilagajanje svojega obsega in vsebine različnim parametrom, kot so npr.:

- karakteristike organizacijskega sistema,
- karakteristike projektov,
- kultura organizacijskega sistema,
- izkušenost razvijalcev,
- nivo zrelosti procesa po modelu CMM[14],
- razpoložljiva razvojna orodja,
- želena kakovost,
- število ljudi v razvojni ekipi itd.

V nasprotju z disciplino konstruiranja metodologij, kjer osnovo za prilagajanje predstavlja metoda (v smislu zaključene enote, ki predstavlja del metodologije) ali njen izdelek, se agilne metodologije osredotočajo predvsem na prilagajanje procesa razvoja IS. Lahko bi torej govorili tudi o konstruiranju oziroma prilagajanju procesa. Razen tega tudi zgleda, da agilne metodologije posvečajo veliko več pozornosti sociološkim vidikom uporabe metodologije, kar je predvsem možno opaziti iz prispevkov pomembnejših avtorjev s tega področja [Beck 2001; Cockburn 1999, 2000a, 2000b in 2000c, Highsmith 2002a in 2002c]. Ti vidiki so bili na področju konstruiranja metodologij precej zapostavljeni. Zadnje raziskave s področja uspešnosti uvedbe in uporabe metodologij v organizacijskih sistemih za razvoj programske opreme, kažejo prav to, kako pomembna je za uspešno uporabo metodologij prav njena sociološka komponenta [Vavpotič 2005].

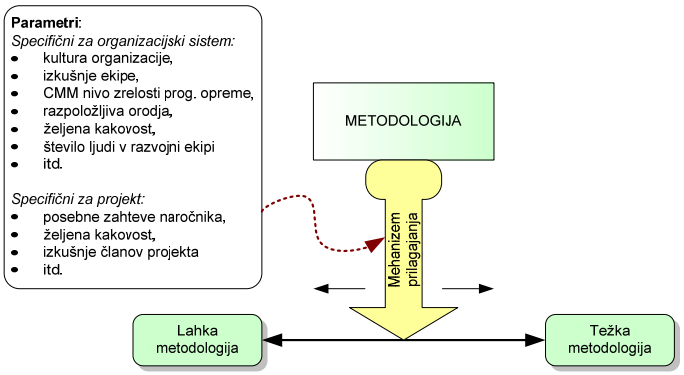
2.2.5.1 Povezava med pojmi lahka, težka in agilna metodologija

V literaturi se pojem agilne metodologije pogosto zamenjuje s pojmom lahke metodologije. Zagovorniki agilnega pristopa pogosto trdijo, da je potrebno agilni

[14] *Capability Maturity Model*

metodologiji, podobno kot npr. v metodologiji XP (eXtreme Programming), slediti v celoti [Henderson 2003]. V nasprotju z njimi Bajec [Bajec 2004b] zagovarja mnenje, da je *prilagodljivost* metodologije neodvisna od *teže* same metodologije. Tako težka kot lahka metodologija je lahko agilna, vse dokler omogoča "ad-hoc" prilagajanje obsega in svoje zgradbe iz posamičnih sestavnih delov metodologije, dejanskim potrebam projektov.

Temeljna lastnost agilne metodologije je torej zmožnost prilagajanja svojega obsega in vsebine različnim parametrom, npr.: karakteristikam projekta, kulturi organizacijskega sistema, izkušenosti razvijalcev, posebnim zahtevam naročnika, nivojem zrelosti modela CMM, razpoložljivim orodjem, želeni kakovosti, številu ljudi v razvojni ekipi itd. Rečemo lahko, da so agilne metodologije bolj osredotočene na sociološko komponento razvoja. Slika 4 prikazuje koncept agilnih metodologij, kot ga zagovarja v svojem prispevku Bajec. Kot je iz slike razvidno, se agilna metodologija s pomočjo mehanizma za prilagajanje v odvisnosti od predhodno naštetih parametrov, pozicionira na ustrezno mesto na skali, ki opredeljuje obseg in s tem tudi težo metodologije.



Slika 4: Koncept agilne metodologije [Bajec 2004b]

2.3 Ožje področje doktorske disertacije

2.3.1 Konstruiranje metodologij

2.3.1.1 Uvod

Kljub prednostim uvedbe in uporabe metodologij, je njihova dejanska uporaba bila (in je še vedno) na zelo nizki ravni. Metodologi so problem najprej začeli reševati s tehnološkega vidika uporabe metodologij. Kot odgovor na tehnično neustreznost

oziroma tehnične težave metodologij se je v prvi polovici 90ih let pojavilo t.i. področje *konstruiranja metodologij* (ang. method engineering). Konstruiranje metodologij skuša odpraviti težave, ki jih povzročajo slaba zasnova, neprilagodljivost, togost oziroma tehnična neustreznost tradicionalnih metodologij [Fitzgerald 2003; Ralyte 2003; Hofstede 1997a; Brinkkemper 1999; Brinkkemper 1996a; Hofstede 2001; Henderson-Sellers 2003]. V eni od prvih opredelitev področja konstruiranja metodologij [Kumar 1992] lahko zasledimo, da je cilj le tega izboljšati uporabnost metodologij. To naj bi dosegli z izdelavo metodologij prilagojenih specifičnim potrebam organizacijskih sistemov. Konstruiranje metodologij se osredotoča predvsem na *izbiro*, *shranjevanje* [Hofstede 1997a; Brinkkemper 1996a; Paige 1997], *iskanje* in *sestavljanje* fragmentov metodologij v nove metodologije, ki so prilagojene specifičnim potrebam organizacijskih sistemov ali projektov.

Prve raziskave, ki jih lahko zasledimo na področju konstruiranja metodologij, segajo v začetek 90ih let. Poleg avtorjev Kumar in Welke, ki v svoji raziskavi [Kumar 1992] postavljata izhodišča za *situacijsko konstruiranje metodologij* (ang. situational method engineering), lahko kot začetnike konstruiranja metodologij obravnavamo tudi avtorje Brinkkemper [Brinkkemper 1996b], Harmsen [Harmsen 1994] in Odell [Odell 1996]. Po začetnem obdobju raziskav na tem področju sledi obdobje umiritve oziroma streznitve, ko avtorji pričnejo ugotavljati, da so postopki konstruiranja metodologij zelo zahtevni in v praksi le delno izvedljivi [Hofstede 1997a]. Kljub temu pa se raziskave s tega področja nadaljujejo tudi danes, pri čemer se večina raziskav osredotoča na izdelavo prilagojenih metodologij [Fitzgerald 2003; Ralyte 2003; Henderson 2003; Bajec 2005b; Vavpotič 2005; Zrnec 2004a].

Kljub številnim raziskavam, se zahtevnejše oblike konstruiranja metodologij do danes sploh niso uveljavile v praksi. Situacijsko konstruiranje metodologij tako ostaja večinoma v domeni akademskih krogov. Poleg tega lahko v zadnjem času zasledimo vedno več raziskav, kot je npr. raziskava avtorja Vavpotiča [Vavpotič 2005], ki dokazujejo, da metodologija, ki je le tehnično prilagojena, ne zadošča. Slabost konstruiranja metodologij je namreč, da večinoma upošteva le tehnične vidike kot npr. konstruiranje različnih postopkov, tehnik, metod ipd., ki naj bi razvojnim ekipam omogočale lažje, hitrejše in bolj kakovostno delo. Ne upošteva pa socialnih vidikov razvojnih ekip. Posledica neupoštevanja socialnih vidikov v okviru metodologij je, kot danes ugotavljajo številni raziskovalci, nizka stopnja uporabe oziroma sprejetosti metodologij v organizacijskih sistemih [Huisman 2003; Fitzgerald 1998].

Kljub temu, da situacijsko konstruiranje metodologij v veliki meri ostaja v domeni teoretičnih raziskav, pa je potrebno omeniti podobnost s trendom agilnosti, ki izhaja iz

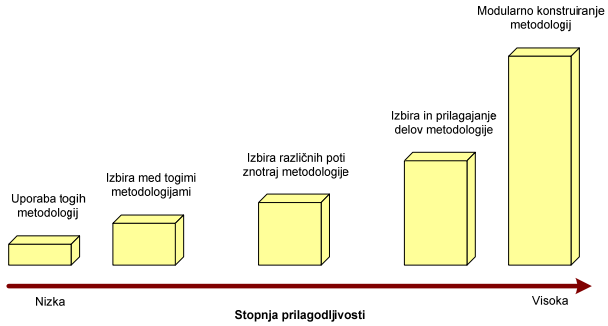
prakse in se je v praksi tudi razmeroma dobro uveljavil. Podobno kot situacijsko konstruiranje metodologij tudi agilni pristopi zagovarjajo uporabo metodologij prilagojenih potrebam posameznih projektov, le da je raven kompleksnosti postopka prilagajanja v okviru agilnih pristopov precej nižja kot pri konstruiranju metodologij. "Agilna" pravila za prilagajanje metodologij lahko zasledimo v raziskavah različnih avtorjev [Cockburn 2000a in 2002; Highsmith 2002b; Ambler 2002].

2.3.1.2 Predstavitev osnovnih pojmov

Kot v svoji raziskavi ugotavljajo Harmsen, Brinkkemper in Oei [Harmsen 1994], so tradicionalne metodologije razvoja programske opreme in IS po svoji naravi splošno namenske. Vsebujejo idealno množico metod, tehnik, smernic ipd., ki jim v realnosti ni mogoče v popolnosti slediti. Zato jih je smiselno prilagoditi določeni situaciji oziroma projektu. Namen konstruiranja metodologij je oblikovanje novih ali prilagojenih metodologij. Posebno zvrst področja predstavlja situacijsko konstruiranje metodologij, ki je usmerjeno v gradnjo in prilagajanje metodologij potrebam posameznih projektov. Poglejmo si opredelitev področja konstruiranja metodologij in specializirane veje tega področja, situacijskega konstruiranja metodologij, kot ju v svojem prispevku opredeljuje Brinkkemper [Brinkkemper 1996a].

Konstruiranje metodologij *je inženirska disciplina, ki se ukvarja s načrtovanjem, konstruiranjem in prilagajanjem metod*[15]*, tehnik in orodij za razvoj IS.*

Situacijsko konstruiranje metodologij *je posebna veja konstruiranja metodologij, ki se ukvarja s prilagajanjem metodologij potrebam konkretnih projektov.*



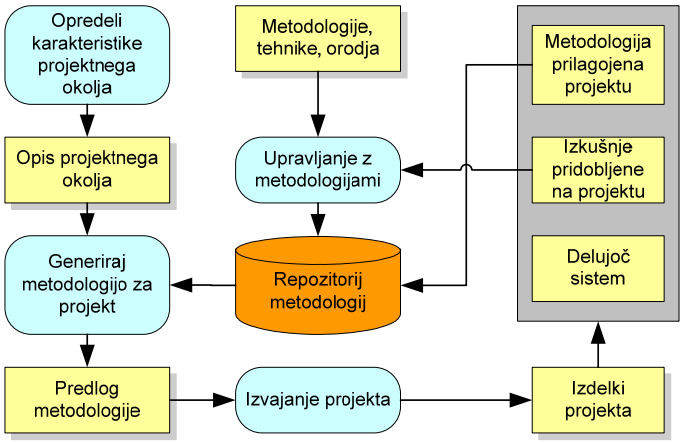
Slika 5: Stopnje prilagodljivosti različnih oblik situacijskega konstruiranja metodologij [Harmsen 1994]

[15] *Poseben način na katerega nekaj naredimo (glej razdelek 2.2.2).*

Glede na stopnjo prilagodljivosti lahko opredelimo več oblik situacijskega konstruiranja metodologij [Harmsen 1994], ki so prikazane tudi na sliki (glej Slika 5):

- uporaba togih metodologij: Ta oblika zahteva uporabo metodologije v točno taki obliki, kot je ta predpisana, z uporabo vseh standardov, tehnik, postopkov ipd. Ta oblika pravzaprav ne spada v sklop konstruiranja metodologij, saj ne dopušča nobenega prilagajanja, prikazana je zgolj kot eden izmed možnih ekstremov.
- izbira med togimi metodologijami: Za to obliko je značilno, da namesto uporabe ene toge metodologije lahko izbiramo med več različnimi metodologijami. Izbrana metodologija naj bi v čim večji meri ustrezala značilnostim projekta. Kljub temu pa izbrane metodologije dodatno ne prilagajamo, ampak jo le uporabimo.
- izbira različnih poti znotraj metodologije: Veliko metodologij ponuja več prilagodljivosti z zagotavljanjem možnosti izbire med več vnaprej določenimi potmi skozi metodologijo. Različne poti znotraj metodologije vključujejo le tiste dele metodologije, ki so za določen tip projekta najbolj primerne. Tipični sta npr. razvojni poti za klasični razvoj in hiter razvoj aplikacij. Nekatere metodologije v okviru izbire poti podpirajo izbor različnih vidikov, kot so npr. pilotni projekti, arhitektura, sistemi za upravljanje z znanjem, realnočasovni sistemi, objektno usmerjen razvoj. Glavna slabost teh metodologij je, da ne dopuščajo kombiniranja več opcij. Tako se naprimer lahko zgodi, da razvoj realnočasovnih sistemov za upravljanje z znanjem ni podprt.
- izbira in prilagajanje delov metodologije: Pri tej obliki gre za gradnjo visoko prilagojenih metodologij. Le te navadno temeljijo na referenčnih metodologijah, iz katerih izbiramo dele, ki ustrezajo projektu. Tipično se gradnja prične z določitvijo globalnega procesa, ki ga v nadaljevanju prilagajamo potrebam projekta oziroma podrobno definiramo njegove posamezne dele. Zaradi visoke kompleksnosti je pri tej obliki obvezna računalniška podpora.
- modularno konstruiranje metodologij: Gre za obliko z najvišjo stopnjo prilagodljivosti. Metodologijo izgradimo na podlagi vnaprej definiranih gradnikov – fragmentov oziroma elementov, ki so shranjeni v repozitoriju. Z uporabo odločitvenih pravil izberemo gradnike, ki najbolj ustrezajo značilnostim in potrebam projekta. Rezultat je učinkovita in skladna metodologija. Ta oblika konstruiranja metodologij ni mogoča brez dobre podpore programskih orodij.

Velik del raziskav s področja konstruiranja metodologij se ukvarja z modularnim konstruiranjem metodologij, ki ga prikazuje Slika 6.



Slika 6: Proces modularnega konstruiranja metodologij [Odell 1996]

Proces modularnega konstruiranja metodologij se lahko prične šele po vzpostavitvi repozitorija metodologij, ki vsebuje elemente različnih metodologij. Prva aktivnost je opredelitev karakteristik projektnega okolja, v katerem naj bi se metodologija uporabljala. Na podlagi opisa karakteristik projekta sledi generiranje prilagojene metodologije z uporabo elementov iz repozitorija metodologij. Rezultat generiranja je predlog metodologije, ki ga po potrebi dodatno prilagodimo projektu. Med izvajanjem projekta pridobivamo izkušnje in po potrebi še dodatno prilagajamo elemente metodologije. Rezultati izvajanja projekta so, poleg ciljnega sistema, še prilagojena metodologija in izkušnje. V okviru aktivnosti "upravljanje z metodologijami" pridobljene izkušnje ter izboljšave metodologije vključimo v repozitorij metodologij, s čimer poskrbimo za stalno izboljševanje metodologije.

Uporaba različnih strategij za razvoj metodologij pove tudi veliko o zrelosti procesa razvoja IS v neki organizaciji. Uporaba konstruiranja metodologij zahteva namreč dve stvari: prvič, da je organizacija zmožna razumeti svoje procese razvoja IS in drugič, da jih je zmožna meriti, s čimer bo lahko zagotovila boljše razvojne aktivnosti in smernice razvoja. Iz tega sledi, da organizacija, ki je uspešno prilagodila metodologijo svojim potrebam razvoja, uporablja razvojni proces, ki je zrelejši. Velja torej, da mora za uspešno prilagajanje, organizacijski sistem poznati svoj proces.

2.3.1.3 Nivoji predstavitve znanja konstruiranja metodologij

Zaradi razumevanja nivojev za predstavitev znanja o konstruiranju metodologij, v nadaljevanju najprej podajamo opredelitev metodologije, kot jo je z vidika konstruiranja metodologij za področje razvoja IS postavil Brinkkemper [Brinkkemper 1996a]:

"Metodologija predstavlja pristop, ki ga razvojni projekti uporabljajo pri razvoju informacijskih sistemov, temelji na določenem načinu mišljenja, je sestavljena iz množice usmeritev ter pravil in podpira sistematično izvajanje razvojnih aktivnosti, skladno s potrebami razvoja izdelka."

Lahko bi tudi rekli, da je metodologija sestavljena iz dela, ki predstavlja *opis njenih izdelkov*, ki nastanejo v njenem okviru in dela, ki *opisuje njen proces*. Del metodologije, ki opisuje izdelke, predstavlja v metodologiji uporabljene koncepte, relacije med njimi in omejitve, ki jim morajo zadostiti. Del metodologije, ki pa opisuje proces, opredeljuje način razvoja ustreznega izdelka, ki je opredeljen v okviru opisa prvega dela metodologije.

Na področju konstruiranja metodologij razlikujemo tri nivoje za predstavitev znanja o metodologijah. Najvišji nivo predstavlja metaznanje o metodologiji in je predstavljen z njenim metamodelom. Na tem nivoju se opredeli modele izdelkov in procesne modele metodologij v obliki metamodelov. V odvisnosti od razpoložljive semantike tega nivoja in njegove razširljivosti [Prakash 1997], lahko metamodel v okviru metodologije opredeljuje velik ali pa majhen niz procesov in izdelkov. Stopnja formalnosti procesnega modela še posebej močno vpliva na zmožnosti orodja za uprizoritev (ang. enactment) instance[16] (izdelavo primerka) metodologije.

Srednji nivo, ki predstavlja znanje o konkretni metodologiji, opredeljuje proces in izdelke z uporabo ustreznih modelov. Z izdelavo primerka metamodela metodologije se opredeli procesni model in modele izdelkov. Na tem nivoju se izvede konstruiranje metodologije, ali z uporabo strategij popolnoma "od začetka", ali s sestavljanjem kosov metodologij ali pa s prilagajanjem že obstoječih metodologij.

Ko je metodologija enkrat opredeljena, je pripravljena za uporabo. Uporaba metodologije nastopi na najnižjem nivoju, ki se imenuje nivo konkretizacije metodologije. Metodologija se uporabi na konkretnem projektu, kjer se njen procesni model izvede, na temelju modelov izdelkov pa se kreirajo izdelki metodologije.

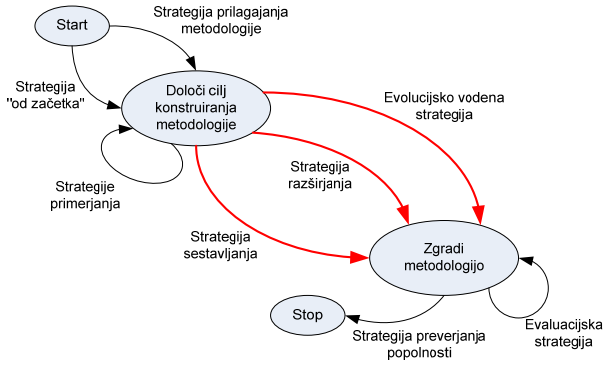
2.3.1.4 Pristopi in strategije situacijskega konstruiranja metodologij

V literaturi lahko zasledimo množico pristopov, ki se uporabljajo na področju situacijskega konstruiranja metodologij. Vsak pristop uporablja določeno strategijo za

[16] *Primerek*

gradnjo oziroma prilagajanje metodologije. Strategij je več in so v nadaljevanju predstavljene v okviru splošnega procesnega modela.

Splošni procesni model za situacijsko konstruiranje metodologij (ang. generic process model for SME[17]), katerega namen je integracija različnih strategij (in s tem tudi pristopov, ki strategije uporabljajo), kateri se uporabljajo na področju situacijskega konstruiranja metodologij, je v svojem prispevku [Ralyte 2003] predstavila avtorica Ralyte. Model je formaliziran z uporabo metamodela procesa, ki je predstavljen na sliki (glej Slika 7).



Slika 7: Splošni procesni model [Ralyte 2003]

Metamodel predstavlja procesni model, ki obsega nedeterministično ureditev namer in strategij. Namera predstavlja cilj, ki ga dosežemo z izvajanjem procesa, strategija pa način na katerega lahko namero dosežemo. Metamodel predstavlja torej označen usmerjen graf z vozlišči, ki označujejo namere in povezavami, ki označujejo strategije. Trojica <izvorna namera, ciljna namera, strategija> predstavlja sekcijo. Vsaki sekciji je pripisana smernica, ki zagotavlja nasvet za izpolnitev ciljne namere ob upoštevanju pripadajoče strategije.

Splošni procesni model vodi metodologa pri opredelitvi cilja v okviru gradnje metodologije in izbiri najustreznejše strategije za doseganje tega cilja. Na sliki (glej Slika 7) se nahajata dva temeljna cilja. Uresničitev prvega cilja ("Določi cilj konstruiranja metodologije") je odvisna od tega, kakšno metodologijo rabi določen projekt. Če metodolog ugotovi, da nobena obstoječa metodologija ne ustreza zahtevam projekta in da je potrebno zgraditi popolnoma novo metodologijo, se odloči za uporabo

[17] *Situational Method Engineering*

strategije "od začetka". Drugo strategijo uporabi v primeru, ko ugotovi, da se obstoječo metodologijo lahko samo prilagodi.

Za uresničitev drugega cilja predlaga model tri alternativne strategije, ki hkrati predstavljajo tudi glavne strategije na področju konstruiranja metodologij. Te strategije opredeljujejo načine za prilagajanje metodologij.

2.3.1.4.1 Strategija sestavljanja

Strategija sestavljanja (ang. assembly strategy), ki jo v svojem prispevku predstavi Ralyte [Ralyte 2001], predpostavlja način konstruiranja, ki temelji na izbiri in sestavljanju delov metodologije ("fragmentov" ali "kosov" metodologij), z namenom izdelati novo ali obogatiti obstoječo metodologijo, ki je namenjena konkretnemu projektu.

Pojem fragmenta metodologije sta predstavila Brinkkemper in Harmsen, in sicer kot ponovno uporabljiv del metodologije. Harmsen [Harmsen 1994] predlaga za potrebe zajema ustreznih delov metodologije dva tipa fragmentov, in sicer fragmente izdelkov in fragmente procesa, odvisno od vidika, ki ga pokrivajo. Fragmenti izdelkov modelirajo strukture posameznih izdelkov (diagrami, tabele, modeli), ki jih predpisuje metodologija razvoja, procesni fragmenti pa predstavljajo modele razvojnega procesa. Procesni fragmenti lahko predstavljajo različne visokonivojske strategije projekta (osnutek metodologije) ali pa bolj natančne procedure, ki podpirajo uporabo tehnik izdelave specifikacij.

Pojem *kos metodologije* (ang. method chunk) je prvič v svoji doktorski disertaciji predstavil avtor Plihon [Plihon 1996], naknadno pa je bil pojem prečiščen v delih avtoric Rolland in Ralyte [Ralyte 2001; Rolland 1998a]. Kos metodologije združuje procesni fragment in fragment izdelka, ki sta med seboj povezana v zaključen modul, z namenom poudariti njegovo konsistentnost in avtonomnost. Oboji, tako fragmenti metodologij kot kosi metodologij predstavljajo osnovne sestavne dele za gradnjo metodologij prilagojenih konkretni situaciji. Preden lahko pričnemo sestavljati fragmente ali kose v novo metodologijo pa jih je potrebno pridobiti iz repozitorija, kjer so shranjeni.

2.3.1.4.2 Strategija razširjanja

Strategijo razširjanja, ki jo je v prispevku [Deneckere 2001] predlagala Deneckere, temelji na razširjanju metodologije z uporabo razširitvenih vzorcev v odvisnosti od konkretne situacije. Strategija predvideva usmerjanje delovanja metodologa s pomočjo uporabe vzorcev za razširjanje metodologije. Ti vzorci nudijo pomoč pri identifikaciji

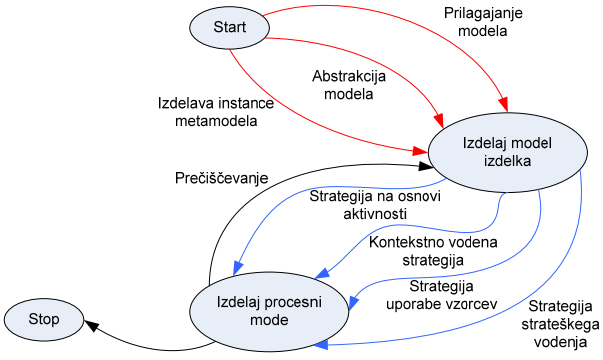
tipičnih situacij, v katerih nastopi potreba po razširjanju metodologije in zagotavljajo različne predloge za razširitve metodologije.

2.3.1.4.3 Evolucijsko vodena strategija

Evolucijsko vodena strategija, ki jo v prispevku [Mohamed 2004] predlagata Ralyte in Mohamed, temelji na uporabi tehnike metamodeliranja. Predstavlja najbolj splošno, od vseh v splošnem procesnem modelu predstavljenih strategij. Namenjena je pridobivanju novih metodologij, in sicer:

- na podlagi abstrakcije obstoječega modela,
- na podlagi izdelave primerka metodologije na osnovi izbranega metamodela ali,
- s prilagoditvijo obstoječega modela ali metamodela specifičnim okoliščinam.

Metamodeliranje je tehnika, ki omogoča zajeti znanje o metodologijah (glej razdelek 2.3.1.3). Predstavlja osnovo za razumevanje, primerjavo, evaluacijo in gradnjo metodologij. Eden od rezultatov skupnosti za konstruiranje metodologij je tudi opredelitev katerekoli metodologije kot skupka modela, ki opisuje izdelek in procesnega modela [Prakash 1999]. Model izdelka opredeljuje množico konceptov, njihovih lastnosti in relacij med njimi, ki so potrebne za opisovanje rezultata procesa. Procesni model pa obsega množico ciljev, aktivnosti in smernic za podporo doseganja ciljev procesa in izvajanja samih akcij. Iz tega sledi, da je gradnja metodologij, ki sledi tehniki meta modeliranja, osredotočena na opredelitve prej omenjenih modelov. To se zrcali tudi na sliki (glej Slika 8), ki predstavlja procesni model evolucijsko vodenega pristopa.



Slika 8: Procesni model evolucijsko vodene strategije [Ralyte 2003]

Procesni model evolucijsko vodenega pristopa vsebuje dve glavni nameri (dva cilja), in sicer: "Izdelaj model izdelka" in "Izdelaj procesni model". Pri tem mora biti procesni model prilagojen modelu izdelka, saj se skozi aktivnosti, akcije in korake, ki jih določa

procesni model, gradi, izboljšuje ali transformira izdelek, ki ga opredeljuje model izdelka. To je tudi vzrok, zakaj v modelu evolucijsko vodenega pristopa najprej nastopa namera "Izdelaj model izdelka".

Model evolucijsko vodenega pristopa predlaga tri alternativne strategije za doseganje cilja izdelave modela izdelka, in sicer strategijo abstrakcije modela, strategijo izdelave instance metamodela in strategijo prilagajanja modela. Za doseganje cilja "izdelave procesnega modela" pa evolucijsko voden pristop predlaga štiri strategije, in sicer: strategijo na osnovi aktivnosti, kontekstno vodeno strategijo, strategijo uporabe vzorcev in strategijo strateškega vodenja. Opisi vseh strategij se nahajajo v [Mohamed 2004].

2.3.1.5 Operatorji za sestavljanje metodologij

Vsi pristopi situacijskega konstruiranja metodologij temeljijo na metamodeliranju, ki ostaja temeljna tehnika na tem področju. V okviru pristopa sestavljanja, predstavlja vsak kos metodologije (ang. method chunk) primerek določenega dela metamodela modularne metodologije. Pristop razširjanja razširja model, ki sam po sebi predstavlja primerek določenega metamodela in predlaga vzorce za njegove razširitve. Vzorce se izdela na podlagi metavzorcev, ki so opredeljeni na metanivoju. Evolucijsko voden pristop pa že v osnovi temelji na metamodeliranju.

V okviru vseh omenjenih pristopov imamo opravka z opredelitvijo, izdelavo primerkov, transformacijami in sestavljanjem modelov in metamodelov. Ustrezne aktivnosti konstruiranja metodologij lahko posplošimo z uvedbo množice splošnih operatorjev [Ralyte 2003]. Ker vsi pristopi postavljajo striktno mejo med modelom izdelka in procesnim modelom, lahko operatorje razdelimo v skupino za gradnjo procesnega modela in skupino za gradnjo modela izdelka. Prva skupina operatorjev se nanaša na akcije, ki jih je potrebno izvesti na modelih izdelkov in obravnava elemente kot so: koncept, povezava, lastnosti. Druga skupina operatorjev se nanaša na akcije, ki jih je potrebno izvesti nad procesnimi modeli in obravnava elemente kot so: namere in strategije.

Na tem mestu naj še omenimo problem ujemanja z zahtevami. Posebnost pristopov situacijskega konstruiranja metodologij je namreč delovanje na osnovi zahtev, ki opredeljujejo želeno-ciljno metodologijo. Katerakoli tehnika za konstruiranje metodologij, ki jo uporabljajo omenjeni pristopi, mora upoštevati opredelitev zahtev glede ciljne metodologije in izbiro rešitev, ki tem zahtevam zadostijo. Zaradi tega je uporaba ustreznega mehanizma ujemanja med modelom zahtev in modelom rešitev bistvenega pomena. Metodologu bi morala biti na voljo možnost ugotavljanja

podobnosti med različnimi elementi procesnih modelov (namerami, sekcijami, celotnimi metamodeli) in modelov izdelkov (koncepti, povezave). Glede na dosedanje rezultate raziskav na tem področju lahko ugotovimo, da vsak pristop posebej predlaga svojo obliko reševanja problema ujemanja z zahtevami [Mohamed 2004; Ralyte 2003].

2.3.1.6 Orodja za podporo konstruiranju metodologij

2.3.1.6.1 Uvod

Na področju konstruiranja metodologij se veliko pozornosti posveča računalniškim orodjem, ki proces konstruiranja avtomatizirajo. Brez uporabe ustreznih programskih orodij naprednejše oblike konstruiranja metodologij v praksi niso izvedljive. V nadaljevanju zato navajamo tri skupine orodij, ki jih za potrebe situacijskega konstruiranja metodologij opredeljuje Arni-Bloch [Arni-Bloch 2005], za tem pa podrobneje predstavimo orodja CAME[18].

2.3.1.6.2 Repozitorij fragmentov

Prvo pomembnejše skupino orodij predstavlja repozitorij metodologij, ki ga pogosto označujemo tudi s pojmom baza metodologij in hrani fragmente oziroma kose metodologij, skupaj z njihovimi opisi. Največji izziv na področju razvoja teh orodij je opredelitev mehanizma za visokonivojsko klasifikacijo fragmentov / kosov metodologij, ki mora zagotoviti pravilen opis posameznega fragmenta / kosa metodologije, tako da vemo, čemu je ta namenjen, ne da bi morali pogledali v njegovo specifikacijo.

Vsebino repozitorija se predstavi z uporabo metamodela, katerega se opiše z uporabo določenega metamodelirnega jezika, ki se običajno opira na nek obstoječ semantični podatkovni model [Tolvanen 1998].

Kot primer repozitorija fragmentov metodologij lahko navedemo knjižnico komponent, ki se nahaja v okrilju ogrodja OPF - OPEN Process Framework [Henderson 2003]. Ker se OPF osredotoča na proces, se v prej omenjeni knjižnici nahajajo samo procesne komponente (fragmenti) metodologije. Ogrodje OPF opredeljuje 5 glavnih razredov procesnih komponent, ki so podkrepljeni z ustreznim metamodelom.

[18] *Computer Aided Method Engineering*

2.3.1.6.3 Orodja CAME

Drugo skupino orodij tvorijo ti. orodja CAME, katerih namen je metodologu olajšati delo pri gradnji metodologij in omogočiti izdelavo učinkovitih, prilagojenih in skladnih metodologij. Orodje CAME temelji na metamodelu metodologije in je namenjeno specifikaciji kosov metodologij, ki se lahko izvede z uporabo strategije "od začetka", s sestavljanjem ali prilagajanjem. V prvem primeru se procesni model in model izdelka določenega kosa metodologije opredeli z izdelavo primerka metamodela, ki ga orodje uporablja. V drugem primeru se kose metodologije sestavi glede na potrebe določene situacije. V tretjem primeru pa se kose metodologij priskrbi s prilagoditvijo kosov, ki pripadajo drugim metodologijam. Pomembna naloga orodij CAME je tudi zagotavljanje ponovne uporabe najboljših izkušenj pridobljenih na projektih oziroma integracija le teh v metodologijo.

Kot navaja Harmsen [Harmsen 1994], naj bi orodje CAME podpiralo naslednje funkcije:

- *opredelitev pravil in karakteristik za izbor ustreznih fragmentov metodologij*: Če želimo izbirati ustrezne fragmente, potrebujemo ustrezna pravila in karakteristike za izbor fragmentov, ki jih opredeli metodolog. Na podlagi lastnosti projekta nato orodje CAME iz baze fragmentov izbere ustrezne fragmente in jih sestavi v novo metodologijo.
- *shranjevanje fragmentov metodologij*: Baza fragmentov predstavlja repozitorij iz katerega lahko metodologi in orodja CAME izbirajo fragmente.
- *pridobivanje in kompozicija fragmentov*: V okviru gradnje metodologije morajo za uspešno pridobivanje fragmentov iz repozitorija biti na voljo ustrezne iskalne operacije. Zaradi kompleksnosti teh se lahko zgodi, da popolnoma avtomatiziran postopek gradnje metodologije sploh ne bo nikoli izvedljiv. Bolj realističen scenarij predvideva kombinacijo delno avomatiziranega postopka gradnje, v katerem sodeluje metodolog.
- *validacija in verifikacija zgrajene metodologije*: Orodje CAME mora po zaključku gradnje preveriti izdelano metodologijo glede konsistentnosti in kakovosti (ali metodologija ustreza predpisanim standardom).
- *prilagajanje izdelane metodologije*: Baza fragmentov naj bi hranila tudi podatke o pridobljenih izkušnjah o uporabi metodologij na prejšnjih projektih. Izkušnje naj bi se uporabljale za izboljševanje fragmentov, ažuriranje karakteristik za opis lastnosti projektov in ažuriranje pravil za izbor fragmentov.
- *integracija z metaCASE orodji*: Orodja CAME in CASE naj bi se sčasoma integrirala. Ko bi se izdelala nova metodologija, bi se hkrati izvedla tudi integracija

ustreznih podporni orodij. Tako prilagajanje CASE orodja zahteva njegovo konfiguracijo za potrebe podpore izdelane metodologije.

- *vmesnik za dostop do baze fragmentov*: Orodje CAME mora metodologu preko ustreznega vmesnika omogočiti iskanje fragmentov v repozitoriju.

Odell [Odell 1996] trdi, da orodja CAME mogoče razdeliti na štiri osnovne tipe:

1. Orodja za definiranje metodologije. Ta orodja omogočajo definiranje komponent metodologije glede na potrebe in značilnosti organizacijskega sistema. Komponente so definirane tako, da jih je mogoče uporabiti večkrat in v kombinaciji z različnimi drugimi komponentami.
2. Orodja za izgradnjo prilagojene metodologije. Ta orodja pomagajo pri izgradnji instance metodologije, ki je prilagojena zahtevam in značilnostim konkretnega projekta. Instanca je zgrajena z izbiro in povezovanjem komponent iz repozitorija, ki ustrezajo potrebam projekta.
3. Orodja za pomoč pri izvajanju projekta (metodologije). Med izvajanjem projekta je mogoče z orodji podpreti delovni tok. Ta tip orodij CAME zagotavlja, da so posamezne naloge dodeljene razvijalcem z ustreznimi znanji in da le ti pri tem uporabljajo primerna orodja.
4. Orodja za izboljševanje metodologije. Stalno izboljševanje je ključno za zagotavljanje uporabnosti metodologije. S pomočjo povratnih informacij, ki jih dobimo ob uporabi metodologije, lahko opredelimo, katere komponente metodologije so slabše in bi jih bilo potrebno izboljšati ali nemara celo ukiniti. Na podlagi povratnih informacij lahko opredelimo tudi nove komponente, ki temeljijo na najboljših izkušnjah.

V okviru doktorske disertacije nas najbolj zanimajo orodja iz tretje skupine - orodja za pomoč pri izvajanju projekta, saj je cilj doktorske disertacije tudi opredelitev izhodišč za izdelavo orodja za pomoč metodologu pri prilagajanju osnovnega procesa, ki se uporablja v okviru organizacijskega sistema.

2.3.1.6.4 Orodja za uprizoritev metodologije

V tretjo skupino se uvrščajo orodja, katerih naloga je uprizoritev skonstruirane metodologije. Orodje iz te skupine mora podpirati razvoj in izdelavo izdelkov v skladu s specifikacijami procesa metodologije. Orodje mora uporabnika metodologije usmerjati v uporabi izbranih kosov metodologije in nuditi različne možnosti za manipulacijo z izdelki, ki jih ti kosi opredeljujejo. Iz tega sledi, da mora orodje razumeti procesni model posameznega kosa metodologije in ga znati izvesti. Podobno lahko sklepamo, da

mora orodje za konstrukcijo izdelka razumeti model izdelka, ki ga predpisuje nek kos metodologije in omogočiti njegovo izdelavo.

2.3.1.7 Zaključna misel

Razdelek o konstruiranju metodologij zaključujemo z mislijo avtorja J. P. Tolvanen-a [Tolvanen 1998], ki pravi: najpomembnejše merilo za merjenje uspeha situacijskega konstruiranja metodologij ni to, kako dobro je metodologija lahko predstavljena, ampak kako se je izboljšala njena uporabnost. V okviru te misli je predstavljena tudi motivacija za naše nadaljnje delo v okviru disertacije.

2.3.2 Proces razvoja programske opreme

2.3.2.1 Uvod

V tem razdelku se bomo podrobno posvetili procesu razvoja programske opreme in njegovi opredelitvi. Proces razvoja programske opreme predstavlja kritičen dejavnik pri izdelavi kakovostne programske opreme, saj je njegov namen upravljanje in transformiranje uporabniških zahtev v programski produkt, ki naj bi tem zahtevam čimbolj ustrezal. V tem kontekstu predstavlja razvojni proces množico aktivnosti, ki jih mora izvesti organizirana skupina ljudi ob pomoči različnih orodij.

Prvotno se je vsak projekt razvoja programske opreme povezoval pretežno samo s pojmom *življenjskega cikla*[19]. Čeprav je bila ideja o procesu razvoja prisotna pri vseh teh projektih, pa sam koncept razvojnega procesa programske opreme ni bil natančno opredeljen. Skozi raziskave na področju programske opreme in njenega razvoja, je proces postopoma dobil svojo lastno identiteto oziroma opredelitev, ki jo povzemamo po [Acuna 1999].

Proces razvoja programske opreme (ang. software process) je delno urejena množica aktivnosti, katere se osredotočajo na upravljanje, razvoj in vzdrževanje IS. Lahko bi tudi rekli, da se razvojni proces osredotoča bolj na proces izdelave kot pa na same izdelke, ki pri tem nastanejo. Opredelitev procesa običajno navaja akterje, ki izvajajo aktivnosti, njihove vloge in izdelke, ki pri tem nastanejo. Organizacijski sistem lahko sam opredeli

[19] *Določa zaporedje izvajanja aktivnosti in postopkov. Poznamo več življenskih ciklov: slapovni, iterativni, inkrementalni, prototipiranje itd [Vavpotič 2003].*

svoj lasten način za izdelavo programske opreme in tako opredeli svoj lastni razvojni proces.

Metodologije razvoja IS navadno vsebujejo bolj ali manj jasno definiran proces, ki je sestavljen iz različnih postopkov, aktivnosti, vlog itd. Proces prikažemo z uporabo *procesnega modela*. Tega lahko obravnavamo kot skupek pravil, strategij, smernic, postopkov, aktivnosti, faz, iteracij itd. Poleg tega model prikazuje tudi vloge, ki izvajajo posamezne aktivnosti in izdelke, ki v aktivnostih nastanejo.

2.3.2.2 Osnovni elementi procesa

Osnovni elementi procesa sestavljajo proces oziroma procesni model. Gre za elemente, ki jih lahko zasledimo v večini procesov razvoja programske opreme. Elemente procesa je možno razdeliti na dve skupini, in sicer:

- skupni elementi, ki po večini nastopajo pri vseh procesih, in sicer: postopek, aktivnost, vloga in izdelek;
- elementi, ki nastopajo samo pri nekaterih procesih (glede na življenjski cikel - običajno gre za objektno usmerjene procese): faza, iteracija, inkrement.

V nadaljevanju so predstavljeni osnovni elementi procesa.

2.3.2.2.1 Postopek

Postopek predstavlja zaključeno množico aktivnosti. Aktivnosti so znotraj postopka običajno povezane. Pri tem so nekatere aktivnosti medsebojno neodvisne (vrstni red njihovega izvajanja ni pomemben), zaradi česar jih je možno izvajati vzporedno. Druge pa so med seboj odvisne, kar pomeni, da jih je mogoče izvesti le zaporedno.

Postopek se izjava v vrstnem redu, kot ga opredeljuje življenjski cikel procesa. Pri slapovnem razvoju se tako naprimer postopke izvaja zaporedno, pri čemer se vsak postopek pred začetkom naslednjega tudi zaključi. Pri iterativnem razvoju pa se postopkov ne zaključuje, ampak se jih v naslednji iteraciji ponovi. Pri iterativnem razvoju se postopke lahko izvaja tudi vzporedno, zaporedno pa se izvaja iteracije in faze.

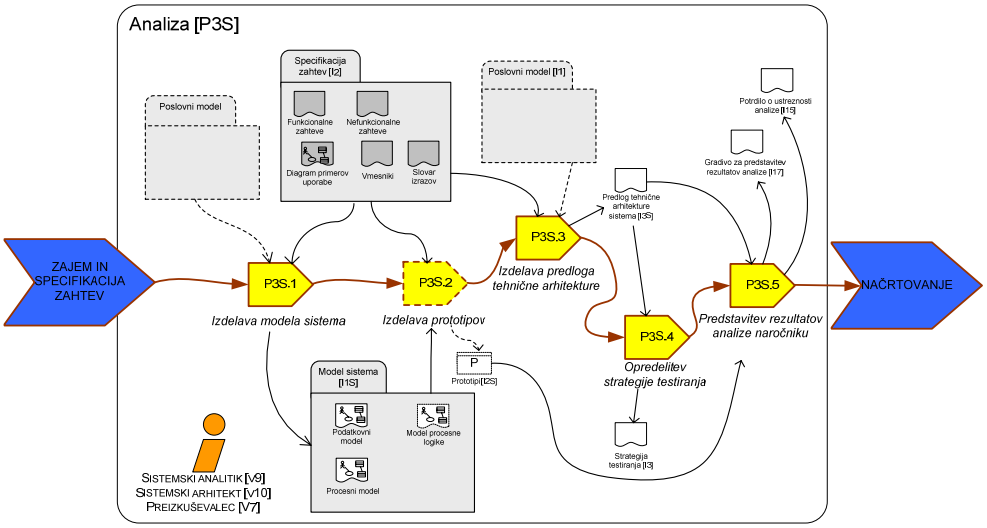
Postopki se delijo na *osnovne postopke*, ki opisujejo potek osnovnega procesa razvoja programske opreme in *podporne postopke*, ki opisujejo potek podpornega dela procesa in omogočajo izvajanje osnovnih postopkov. V doktorski disertaciji nas bodo s stališča

prilagajanja zanimali samo osnovni postopki zajetega procesa v organizacijskem sistemu, zato jih v nadaljevanju tudi podrobneje predstavljamo.

V okviru osnovnih postopkov se osredotočimo predvsem na opravila tehnične narave. Prvič so bili ti postopki podrobneje opredeljeni v okviru slapovnega razvoja, kasneje pa se je nabor postopkov razširil, na eni strani na področje poslovnega modeliranja in strateškega načrtovanja, na drugi strani pa na področje uvajanja in vzdrževanja. V nadaljevanju so našteti nekateri osnovni postopki, ki se pojavljajo pri večini procesov razvoja programske opreme:

- postopek zajema in specifikacije zahtev - pridobivanje uporabniških zahtev za nastajajoč sistem,
- postopek analize - analiza in izdelava konceptualnega načrta sistema,
- postopek načrtovanja - izdelava načrta celotnega sistema in podrobnega načrta posameznih elementov sistema,
- postopek izvedbe - realizacija načrta, kar pomeni razvoj elementov načrta v enem izmed programskih jezikov,
- postopek testiranja - testiranje nastalih programskih sklopov, integracije, celotnega sistema itd.

Predstavljeni postopki ne predstavljajo edino možnost za delitev osnovnih postopkov. Različne metodologije namreč lahko obsegajo tudi postopke poslovnega modeliranja, vzdrževanja, strateškega načrtovanja itd. Slika 9 prikazuje primer postopka, ki je prikazan v procesnem modelu:



Slika 9: Primer opisa postopka

2.3.2.2.2 Aktivnost

Aktivnost je del postopka, ki natančno opredeljuje način dela za izdelavo izdelkov. Izdelke, ki nastanejo v okviru neke aktivnosti, imenujemo izhodni izdelki in predstavljajo njen rezultat. Tiste izdelke, ki v aktivnost vstopajo, imenujemo vhodni izdelki in predstavljajo osnovo za izvajanje aktivnosti. Vsako aktivnost izvaja ena ali več vlog, pri čemer je pomembno, da se določi, katera vloga je odgovorna za izvedbo aktivnosti.

Posamezne dele aktivnosti se izvaja z uporabo različnih tehnik, ki so v okviru aktivnosti pogosto navedene. Poleg tega so lahko navedene tudi tehnike, ki so potrebne za izdelavo nekega izdelka (npr. diagramske tehnike).

2.3.2.2.3 Vloga

Vloga predstavlja element procesa, ki ima določene odgovornosti in izvaja določeno množico aktivnosti. Neko vlogo pri projektu ponavadi zasedajo (nastopajo v njej) člani projektne skupine. Vloga predstavlja pomen, ki ga ima oseba v nekem trenutku (in na določenem delovnem mestu) za projekt in ne dejanske osebe - člana projektne skupine. Velja, da lahko v neki vlogi nastopa več članov projekte skupine, po drugi strani pa tudi lahko en član opravlja več vlog v okviru projekta.

Za izvajanje neke vloge je pomembno, da ima natančno določene pristojnosti in odgovornosti glede izvajanja aktivnosti. Zahtevana znanja, ki so potrebna za izvajanje vloge, so lahko navedena v opisu vloge. Naloga vloge je, da s pomočjo izkušenj in znanj izvaja posamezne aktivnosti. Izkušnje in znanja potrebuje tudi za uporabo orodij in tehnik.

2.3.2.2.4 Izdelek

Izdelek predstavlja končni ali vmesni rezultat izvajanja aktivnosti. Izdelek se uporablja v okviru projekta, kjer služi za prenos podatkov med posameznimi aktivnostmi, postopki in vlogami. Izdelek je lahko tudi končni rezultat aktivnosti, postopka ali celotnega projekta.

Na eni strani se izdelek obravnava kot rezultat izvajanja aktivnosti (izhod), na drugi strani pa predstavlja vhod v drugo aktivnost. Pri izdelavi izdelka se uporabljajo različne tehnike in orodja.

Nek izdelek je lahko sestavljen tudi iz več podizdelkov. V primeru inkrementalnega razvoja se srečujemo z deli izdelkov, kar pomeni, da izdelek ni zaključen v celoti, temveč je zaključen le del izdelka, ki se nanaša na določen sklop. Drugače je pri

iterativnem razvoju, kjer se srečujemo z različicami istih izdelkov, ki so z vsako iteracijo bolj popolni. Metodologija razvoja lahko vključuje tudi primere izdelkov (ang. pattern) in predloge (ang. template), na podlagi katerih se kasneje izdelajo lastni izdelki. Poleg tega so lahko v okviru metodologije določeni tudi standardi in smernice za izdelavo posameznih izdelkov.

Zelo pomemben je tudi koncept sledljivosti, ki omogoča razbrati, na podlagi katerih izdelkov je nastal nek drug izdelek. Naprimer na osnovi katerih elementov načrta je nastala neka programska koda.

2.3.2.2.5 Iteracije in faze

Iteracija predstavlja en prehod skozi vse postopke procesa. Rezultat iteracije so različice izdelkov, ki lahko obsegajo tudi različico delujočega izdelka. Vsaka iteracija gre skozi vse postopke procesa, seveda z različnim poudarkom na posameznih postopkih.

Fazo sestavlja več iteracij s podobno vsebino oziroma iteracije s podobnim poudarkom na posameznih postopkih. Posamezne metodologije lahko določajo različne faze in jih tudi različno poimenujejo.

2.3.2.2.6 Inkrement

Pojem inkrement se uporablja pri inkrementalnem procesu. Podobno kot pri iteraciji se tudi v okviru inkrementa gre skozi vse postopke. Inkrement in iteracija se bistveno razlikujeta v tem, da se pri inkrementu vsi postopki izvedejo v celoti, vendar le za del (inkrement) sistema. Končni rezultat inkrementa je torej delujoč, zaključen in uveden del sistema.

2.3.2.2.7 Tehnika

Pomemben element procesa predstavljajo tudi tehnike. Kot je bilo omenjeno že v prejšnjih razdelkih vidimo, da se tehnika povezuje z naslednjimi elementi procesa: aktivnost, vloga, izdelek. Nekatere metodologije ločujejo proces razvoja od tehnik, medtem ko druge metodologije tehnike integrirajo, s čimer te postanejo neločljiv del procesa.

Tehnika, kot jo opredeljuje Brinkkemper [Brinkkemper 1996a], predstavlja proceduro, ki po možnosti predpisuje tudi notacijo, za izvedbo razvojne aktivnosti. Tehnika naj ne bi obsegala le vidik predstavitve konceptov, ampak tudi proceduralni vidik. Zaradi tega se v nekaterih virih uporablja tudi kot sinonim za metodo[Cockburn 2002]. Primeri tehnik so: podatkovno modeliranje z entitetnimi diagrami, intervjujanje z uporabo naravnega jezika, psevdo-kodiranje z akcijskimi diagrami. Tehnike je možno

klasificirati na več načinov: glede na stopnjo formalnosti uporabljene notacije (naravni jezik, strukturni diagrami), glede na tipe razvojnih aktivnosti, ki jih podpira (podatkovno modeliranje, procesno modeliranje itd.).

2.3.2.2.8 Orodje

Orodje predstavlja avtomatizirano sredstvo, ki podpira izvedbo dela razvojnega procesa. Nabor orodij za razvoj IS je zelo obširen. Nekatera podpirajo samo eno ali nekaj notacij, druga pa nudijo podporo celotnemu življenjskemu ciklu.

V metamodelu metodologije ni nujno, da je meta element orodje neposredno povezan z metaelementom proces, ampak je lahko povezan posredno, preko drugih metaelementov. Navajamo ga zato, ker ga v nadaljevanju doktorske disertacije obravnavamo kot enega izmed elementov procesa, ki se lahko povezuje samo z ostalimi metaelementi (tehnika, standard, izdelek), kot to opredeljuje metamodel zajete metodologije v organizacijskem sistemu. Uporaba določenega orodja je namreč lahko odvisna od standardov, uporabljene tehnike za izvajanje aktivnosti, izkušenj in znanj, ki jih mora posedovati vloga itd.

2.3.3 Modeliranje razvojnega procesa

2.3.3.1 Uvod

Modeliranje procesa razvoja programske opreme se nanaša na opredelitev procesa v obliki procesnega modela in opredelitev avtomatizirane podpore za modeliranje in izvajanje procesnih modelov med izvajanjem procesa. Finkelstein [Finkelstein 1994] v svojem prispevku opredeli procesni model kot opis procesa, ki je izražen v jeziku, primernem za modeliranje procesa.

2.3.3.2 Procesni model

Procesni model predstavlja abstrakten opis arhitekture, načrta ali opredelitve razvojnega procesa. Vsak opis procesnega modela na različnih nivojih podrobnosti opisuje organizacijo elementov izvršenega, tekočega ali predlaganega procesa in hkrati predstavlja opredelitev procesa, ki se bo uporabljala pri vrednotenju in izboljševanju procesa. Procesni model je mogoče analizirati in če je izvršljiv, z njim tudi simulirati proces, ki ga model opisuje. Procesni modeli se ponavadi uporabljajo za nadzor

(evaluacijo in izboljševanje) procesa v organizacijskem sistemu, lahko pa se uporabljajo tudi za potrebe teoretičnih raziskav in enostavnejšo avtomatizacijo procesa.

Colette Rolland [Rolland 1998b] v svojem prispevku opiše procesni model na naslednji način: Procesni model opisuje množico procesov, ki so si podobni. Zaradi tega predstavlja procesni model opis določenega tipa procesov (procesni model se nahaja na ravni tipa). Ker procesni model predstavlja tip procesa, posamezni primerek procesa predstavlja konkretno instanco tega procesnega modela. Isti procesni model se lahko uporabi večkrat za razvoj več aplikacij, zato pravimo, da opredeljuje več primerkov. S procesnim modelom je možno predpisati kako stvari morajo biti, oziroma naj bi se, oziroma bi bile lahko izvedene. Procesni model predstavlja bolj ali manj grobo predvidevanje o tem, kako naj bi potekal dejanski razvoj IS. Sam procesni model naj bi bil:

- *opisen*: Procesni model omogoča beleženje dejanskega dogajanja med izvajanjem procesa. Poleg tega omogoča predstaviti pogled opazovalca na način izvajanja procesa in določi izboljšave za povečanje obstoječe učinkovitosti.
- *predpisujoč*: Procesni model opredeli želene procese in kako naj bi se, oziroma bi se lahko, ali bi se utegnili izvajati. Določi tudi pravila, smernice in vzorce obnašanja, ki vodijo do želenih performans.
- *pojasnjevalen*: Procesni model zagotavlja razlago za principe, na katerih temelji proces, raziskuje in vrednoti več možnih smeri izvajanja, ki temeljijo na utemeljenih argumentih in vzpostavlja eksplicitno povezavo med procesom in zahtevami, ki jih mora zadovoljiti.

Vsak procesni model je podprt z metamodelom. Metamodel procesa pojasnjuje, kaj se dogaja v procesu razvoja, nad čem, kdaj in zakaj. Iz operativnega vidika pa naj bi metamodel zagotavljal smernice metodologom, ki se ukvarjajo s konstruiranjem metodologij in razvijalcem aplikacij [Rolland 1993].

2.3.3.3 Klasifikacija procesnih modelov

Prva klasifikacija procesnih modelov se nanaša na samo uporabo termina "procesni model". Glede na področje uporabe tega termina, Dowson [Dowson1988] razlikuje štiri opredelitve procesnega modela:

- aktivnostno usmerjena opredelitev [Feiler 1993]: *Procesni model predstavlja delno urejeno množico aktivnosti, katerih namen je doseči zastavljeni cilj.*

- izdelčno usmerjena opredelitev: *Procesni model predstavlja niz aktivnosti, ki zaporedoma izvajajo transformacije izdelka z namenom izdelati željeni končni izdelek.*
- odločitveno usmerjena opredelitev: *Procesni model predstavlja množico povezanih odločitev, ki se izvedejo z namenom opredelitve končnega izdelka.*
- kontekstno usmerjena opredelitev: *Procesni model predstavlja zaporedje kontekstov, ki prožijo zaporedne transformacije pod vplivom odločitev, ki se sprejmejo v okviru konteksta.*

Druga zanimiva klasifikacija procesnih modelov izhaja iz njihove razdrobljenosti. Razdrobljenost opredeljuje nivo podrobnosti v procesnem modelu in vpliva na vrsto navodil, razlag in možnosti sledenja, ki ga lahko procesni model zagotovi. Zahtevana stopnja razdrobljenosti procesnega modela je odvisna od situacije, za katero se model izdela [Rolland 1998b].

Tretja klasifikacija razlikuje procesne modele glede na prilagodljivost metodologij. Vsaka metodologija obsega bolj ali manj opredeljen proces. Razpon prilagodljivosti procesnih modelov tako prikazuje Slika 5 (glej razdelek 2.3.1.2).

2.3.3.4 Pristopi k modeliranju procesa

Obstajajo različni načini modeliranja procesa. Proces je možno modelirati na različnih nivojih abstrakcije (npr. generični modeli ali modeli izdelani po meri) in z različnimi cilji (izdelava opisnih modelov ali izdelava predpisujočih modelov). Ključne razlike med omenjenimi modeli so predstavljene v [McChesney 1991; Pfleeger 1998]. Procesni model podaja več tipov informacij, odvisno s katerega vidika gledamo na modeliranje procesa. Curtis [Curtis 1992] navaja naslednje vidike, ki jih lahko zasledimo v literaturi:

- *funkcionalni*: Določa kateri elementi procesa se vključijo v proces in kateri podatkovni tokovi so za omenjene elemente pomembni.
- *vedenjski*: Določa kdaj (glede na zaporedje) in pod kakšnimi pogoji se določen procesni element implementira.
- *organizacijski*: Določa kje in kdo v organizacijskem sistemu implementira elemente procesa.
- *informacijski*: Predstavlja informacije, ki se v procesu preoblikujejo in pojavijo na njegovem izhodu, vključno z njihovo strukturo in medsebojnimi povezavami.

Modeli so zgrajeni v skladu z jeziki, nivojem abstrakcije ali formalizmi, ki se uporabljajo za predstavitev določenega vidika. Nobena od naštetih abstrakcij:

proceduralni programski jeziki, analiza in načrtovanje, jeziki in pristopi umetne inteligence, dogodki in prožilci, nadzor delovnega toka, prehajanje stanj in Petrijeve mreže, funkcijski jeziki, formalni jeziki, podatkovno modeliranje, objektno modeliranje itd. ne pokriva vseh vidikov modeliranja. Zaradi tega v literaturi predstavljeni modeli uporabljajo jezike, ki temeljijo na več kot eni od naštetih abstrakcij. Ti modeli upoštevajo potrebo po integraciji več predstavitvenih paradigm, čeprav to vpliva na težjo opredelitev samega modela [Vasconcelos 1997].

Na eni strani imamo torej na voljo številne notacije za modeliranje procesa. Opis različnih notacij se nahaja v [SPC 1992]. Vendar pa ne obstaja nobenih podatkov o tem, katera notacija je v določenih okoliščinah primernejša za uporabo. Na drugi strani je bilo predlaganih veliko pristopov, ki za modeliranje procesa uporabljajo kombinacije več paradigm. Za te pristope sta značilni dve glavni karakteristiki:

- razen Petrijevih mrež, vsi uporabljajo za predstavitev tekstovno obliko,
- večina pristopov se nahaja že na nivoju programskih jezikov, zaradi česar je problematične situacije zelo težko modelirati, poleg tega pa silijo metodologa v poznavanje številnih tehnoloških vidikov modelirnih formalizmov.

2.3.4 Sklep

Na področju konstruiranja metodologij je bilo opravljenih veliko raziskav glede prilagajanja razvojnih metodologij. Kljub temu, da se izsledki te discipline nikoli niso uspešno uveljavljali v praksi, se je konstruiranje metodologij ohranilo vse do danes. Kot izrazito teoretično področje je prispevalo levji delež k odkrivanju novega znanja o prilagajanju metodologij. Raziskave konstruiranja metodologij so usmerjene predvsem v zagotavljanje teoretičnih temeljev za:

- opredelitev pristopov in uporabe strategij prilagajanja,
- obravnavanje različnih zornih kotov prilagajanja (izdelki, proces) in
- opredelitev funkcionalnosti orodij CAME.

Ključni problem, da se izsledki konstruiranja metodologij niso uveljavili v praksi, je bila kompleksnost obravnavanih postopkov za izvedbo konstruiranja, ki temeljijo na zapletenih matematičnih formalizmih. Večina pristopov konstruiranja metodologij obravnava proces konstrukcije predvsem na nivoju metamodeliranja. Z vidika teoretične obravnave tematike je to sicer popolnoma ustrezno in razumljivo, v praksi pa je tak pristop zelo slabo sprejet.

Kljub relativno slabi sprejetosti v praksi verjamemo, da lahko zaradi močne teoretične osnove mehanizmi konstruiranja metodologij predstavljajo zelo dobro izhodišče za opredelitev mehanizmov prilagajanja, ki bodo uporabni v realnem okolju (praksi).

2.4 Ugotovitve in izhodišča za opredelitev teme doktorske disertacije

2.4.1 Uvod

Na začetku podpoglavja podajamo najprej bistvene ugotovitve, do katerih smo prišli na podlagi analize širšega in ožjega raziskovalnega področja. Te ugotovitve nato uporabimo za grobo opredelitev namena doktorske disertacije in kasneje za podrobnejšo opredelitev ciljev, ki jih v okviru disertacije želimo doseči.

2.4.2 Dosedanje ugotovitve

2.4.2.1 Ugotovitve s področja konstruiranja metodologij

Na podlagi analize znanstvene literature s področja situacijskega konstruiranja metodologij lahko ugotovimo, da je bilo do danes na tem področju opravljenih zelo veliko raziskav (glej razdelek 2.3.4). Kljub temu pa se raziskave s področja situacijskega konstruiranja metodologij soočajo s številnimi težavami, ki jih v svojih prispevkih med drugim navajajo tudi Hofstede in ostali avtorji [Hofstede 1997a; Highsmith 2002b; Tolvanen 1998]. Nas so zanimali predvsem problemi, ki preprečujejo uveljavitev konstruiranja metodologij v praksi. V okviru teh smo identificirali naslednje probleme:

- pomanjkanje ustreznih orodij,
- visoke zahteve glede znanja metodologov,
- problem skladnosti med fragmenti metodologij,
- sprejetost področja med metodologi in razvijalci ter
- obravnava konstruiranja na višjem nivoju abstrakcije.

2.4.2.1.1 Pomanjkanje ustreznih orodij

Čeprav postaja teorija o situacijskem konstruiranju metodologij vse bolj trdna, imamo še vedno na razpolago zelo malo orodij, ki bi podpirala opredelitev ponovno uporabljivih kosov metodologij, njihovo shranjevanje v repozitoriju, sestavljanje itd. Še posebej je pri orodjih pereča težava uprizarjanja metodologije.

Da bi lahko v sklopu orodja implementirali vse naštete možnosti, je potrebno rešiti več problemov, in sicer:

- opredelitev primernih kosov metodologij: Bolj natančno želimo metodologijo prilagoditi konkretni situaciji, manjši morajo biti koščki metodologije (višja stopnja razdrobljenosti), kar ima za posledico večjo kompleksnost pri sestavljanju in shranjevanju v repozitoriju.
- opredelitev ustrezne klasifikacije kosov metodologij: Kot smo omenili že v uvodu razdelka 2.3.1.6 je potrebno opredeliti mehanizem za visokonivojsko klasifikacijo metodologij.
- opredelitev postopka za sestavljanje metodologije: Opredeliti je potrebno ustrezen postopek za sestavljanje procesnega modela in modelov izdelkov metodologije.
- uprizoritev metodologije: Orodje je možno uporabiti za podporo razvoja in izdelavo ustreznih izdelkov, v skladu s specifikacijo procesa metodologije.

2.4.2.1.2 Visoke zahteve glede znanja metodologov

Veliki problemi se navezujejo na potrebno znanje metodologov. Visoka stopnja prilagodljivosti pri konstruiranju ali prilagajanju metodologij namreč zahteva od metodologov, ki izvajajo prilagajanje metodologije naslednje veščine:

- izredno dobro poznavanje prednosti in slabosti različnih metodologij in tehnik razvoja,
- poznavanje bistvenih lastnosti obravnavane problemske domene,
- poznavanje formalnih načinov za opisovanje fragmentov metodologij,
- poznavanje zapletenih postopkov za iskanje fragmentov v repozitoriju fragmentov in
- sodelovanje metodologa pri sestavljanju fragmentov v celoto (na voljo nimamo popolnoma avtomatizirane podpore).

2.4.2.1.3 Problem skladnosti med fragmenti metodologij

Pri konstruiranju nove metodologije ima velik pomen zagotavljanje skladnosti [Rossi 2000; Brinkkemper 1996b] med posameznimi fragmenti, in sicer:

- zagotoviti je potrebno skladnost iz sintaktičnega (skladenjskega) vidika in
- zagotoviti je potrebno skladnost iz semantičnega (pomenskega) vidika.

Čeprav je skladnost možno zagotoviti z uporabo konsistenčnih pravil, o čemer razpravlja tudi Hofstede v svojem prispevku [Hofstede 1997a], pa ta pravila običajno zajamejo le sintaktični vidik integracije, medtem ko je obravnava semantičnega vidika bolj zapletena [Hofstede 1997a; Brinkkemper 1996a in1999; Karlson 2004; Nuseibeh 1996; Paige 1997 in 1999; Ralyte 2003; Song 1995; Harmsen 1996].

2.4.2.1.4 Sprejetost področja med metodologi in razvijalci

Kljub temu, da so bile raziskave na področju konstruiranja metodologij in na področju situacijskega konstruiranja metodologij v akademskem okolju zelo intenzivne, ga inženirji v praksi nikoli niso širše sprejeli [Bajec 2004c; Henderson 2003; Hofstede 1997a]. Razloge zakaj je temu tako, omenja že podpoglavje v uvodu doktorske disertacije (glej podpoglavje 1.3.3), glavni problem pa sta previsoka raven kompleksnosti in previsoka cena postopka konstruiranja oziroma prilagajanja metodologije, ki ga je potrebno izvesti pred samim začetkom izvajanja projekta razvoja IS [Henderson 2003].

2.4.2.2 Ugotovitve s področja agilnih pristopov

V tem razdelku podajamo nekaj značilnosti agilnih pristopov in imajo posreden vpliv tudi na nadaljnje delo v okviru snovanja pristopa za prilagajanje procesa v nadaljevanju disertacije.

S pojavom agilnih pristopov [Cockburn 2002; Highsmith 2002a] je postala ideja o konstruiranju metodologij veliko bolj zanimiva, saj je tokrat pobuda prišla s strani inženirjev, ki delujejo v praksi (realnem razvojnem okolju). Ideja agilnih pristopov je namreč zelo podobna ideji, na kateri temelji konstruiranje metodologij, le da se pri tem uporablja veliko bolj enostavne prijeme [Cockburn 2002]. V nasprotju z disciplino konstruiranja metodologij, kjer predstavlja osnovo za prilagajanje metoda ali njen izdelek, se agilne metodologije osredotočajo predvsem na prilagajanje procesa razvoja IS. Lahko bi torej govorili tudi o konstruiranju procesa. Pri tem se ne spuščajo na nivo podrobnih opredelitev procesnih fragmentov in na opredelitve tehnik za zajem formalne semantike fragmentov.

Razen tega tudi zgleda, da agilne metodologije posvečajo veliko več pozornosti sociološkim vidikom uporabe metodologije [Cockburn 1999, 2000a, 2000b in 2000c; Beck 2001], kateri so bili na področju konstruiranja metodologij precej zapostavljeni. Gre za upoštevanje samih lastnosti razvijalcev, ki so člani projektne skupine in pri svojem delu uporabljajo metodologijo. Poudarjanje sociološke komponente je razvidno prav v priporočilih in principih agilnih pristopov.

Z vplivom sociooloških dejavnikov na razvoj programske opreme, se je v okviru lahke metodologije Crystal Clear, v svojih prispevkih med drugim ukvarjal tudi avtor te metodologije Cockburn [Cockburn 1996, 1999 in 2000c]. Še bolj natančno pa se je tej tematiki posvetil Hofstede, ki je podrobno opredelili vplive kulturoloških razlik med razvijalci na delo v skupinah [Hofstede 1997b in 2001], ki veljajo na splošno in jih v svojih delih citirajo tudi teoretiki s področja organizacijskih znanosti.

S prilagajanjem metodologij razvoja IS in v okviru teh s prilagajanjem samega procesa razvoja, se ukvarja več avtorjev s področja agilnih pristopov. Med najbolj vidne prištevamo Henderson-a, Cockburn-a in Highsmith-a.

2.4.3 Izhodišča za opredelitev ciljev doktorske disertacije

Na podlagi ugotovitev, ki smo jih podali o vsebini raziskavah na področju konstruiranja metodologij in o vse bolj uveljavljenih agilnih pristopih, ki se uporabljajo v okviru uporabe metodologij razvoja, bomo v nadaljevanju podali izhodišča za opredelitev ciljev doktorske disertacije.

Večina znanstvenih prispevkov s področja konstruiranja metodologij obravnava pristop prilagajanja oziroma konstrukcije pretežno samo z gledišča vprašanja, kako skonstruirati metodologijo, ki bo tehnično ustrezna. Čeprav so rezultati raziskav iz tega področja sicer zelo vzpodbudni, se disciplina sama, zaradi problemov, ki smo jih navedli v okviru ugotovitev o konstruiranju metodologij (glej razdelek 2.4.2.1), nikoli ni uspešno uveljavila v praksi. Poleg tega se pristopi konstruiranja metodologij skoraj ne dotikajo problematike socialne ustreznosti in s tem sprejetosti metodologije med njenimi uporabniki, kar jim daje dodaten negativen pridih z vidika praktične uporabe.

Po drugi strani, smo na podlagi ugotovitev s področja agilnih metodologij spoznali, da moderni agilni pristopi dajejo veliko večji pomen enostavnosti postopkov prilagajanja, ki izvirajo iz uveljavljenih praks in se v nasprotju s pristopi konstruiranja metodologij osredotočajo bolj na prilagajanje procesa razvoja IS. Poleg tega smo tudi ugotovili, da agilni pristopi, v okviru prilagajanja procesa, dajejo sociološkemu vidiku prilagajanja

razmeroma velik, če ne kar glavni pomen [Cockburn 1996, 1999, 2000a, 2000b in 2000c; Highsmith 2002c].

Kljub vsemu v literaturi še vedno pogrešamo celovitejšo in podrobnejšo opredelitev samega pristopa k prilagajanju razvojnega procesa (agilnega pristopa) konkretnim potrebam projektov, ki se izvajajo v okviru organizacijskega sistema in ki bi podal formalno opredelitev postopka za prilagajanje procesa na temelju upoštevanja več tipov karakteristik in na nivoju samega elementa procesa [Zrnec 2004a in 2004b]. Pod različnimi tipi karakteristik imamo v mislih karakteristike, s katerimi lahko zajamemo tako tehnične lastnosti projekta kot sociološke značilnosti razvijalcev.

Ključ do uspeha pri uvajanju metodologij razvoja IS v organizacijske sisteme in njihovi kasnejši uporabi ter sprejetosti, vidimo zato predvsem v zmožnosti prilagajanja razvojnega procesa, ki ga predpisuje uporabljena metodologija, specifičnim potrebam projektov, ki jih organizacijski sistemi izvajajo. Zaradi tega mora metodologija, ki je prilagodljiva, definirati *fleksibilen proces razvoja*, ki *ga je možno prilagajati* razmeram, v kakršnih se projekti vzpostavljajo. Pri tem je potrebno postopek prilagajanja procesa podpreti z ustreznim programskim orodjem, ki bo avtomatiziralo sam postopek prilagajanja in s tem olajšalo delo metodologov.

Namen doktorske disertacije je podati formalno opredelitev odločitvenega modela za prilagajanje procesa razvoja IS individualnim potrebam projektov, ki se izvajajo v okviru opazovanega organizacijskega sistema. Pri tem temeljimo na dosedanjih ugotovitvah predvsem s področja konstruiranja metodologij in v manjšem obsegu na ugotovitvah s področja agilnih pristopov. Odločitveni model naj bi predstavljal temelje za izdelavo orodja za prilagajanje procesa, ki ga pri svojem delu uporabljajo metodologi in bi predstavljal ključen pripomoček pri njegovem delu v okviru aktivnosti prilagajanja procesa konkretnim potrebam projektov.

Opredelitev namena disertacije služi kot temelj za opredelitev ciljev doktorske disertacije, ki bodo natančneje predstavljeni v nadaljevanju.

2.4.4 Opredelitev ciljev doktorske disertacije

2.4.4.1 Predpostavke

Postopki konstruiranja metodologij so zaradi svoje velike prilagodljivosti zelo obširno in kompleksno področje. Hofstede je v svojem prispevku [Hofstede 1997a] celo postavil

pod vprašaj problem izvedljivosti konstruiranja metodologij. Kljub temu predpostavljamo, da so pridobljena dognanja s tega področja uporabna kot izhodišča za doktorsko disertacijo, če pri reševanju problema vpeljemo določene robne pogoje (omejitve), ki jih bomo predstavili v nadaljevanju. Z opredelitvijo omejitev želimo doseči dvoje:

- zagotovimo, da ostane reševanje problema obvladljivo in
- zožimo problemsko domeno na želeni obseg.

Opredelili smo naslednje robne pogoje oziroma predpostavke, na podlagi katerih temelji naše nadaljnje delo:

1. osredotočimo se na *prilagajanje* procesa: V okviru te predpostavke želimo poudariti, da se bomo v doktorski disertaciji osredotočili samo na postopek *prilagajanja* procesa in ne na sestavljanje (konstruiranje) procesa iz splošnih elementov, ki bi bili shranjeni v nekem repozitoriju. Prvi pristop - prilagajanje procesa predstavlja našo idejo o prilagajanju procesa individualnim potrebam projektov, medtem ko je ideja, na kateri temelji drugi pristop, uporabljena že na področju konstruiranja metodologij.
2. prilagajanje procesa je omejeno na en organizacijski sistem: S to predpostavko se omejimo na opazovanje procesa v enem organizacijskem sistemu. Gre namreč za to, da naš pristop za prilagajanja procesa temelji na uporabi odločitvenih pravil, ki usmerjajo izbiro ustreznih elementov na podlagi osnovnega procesa. Ta množica odločitvenih pravil pa je za vsak organizacijski sistem lahko precej različna, saj se tudi osnovni procesi teh organizacijskih sistemov lahko med seboj precej razlikujejo. Predpostavka je pomembna predvsem z vidika izdelave programskega orodja za prilagajanje procesa.
3. osnovo za prilagajanje predstavlja splošni proces v organizacijskem sistemu: Izhajamo iz procesa, ki je v organizacijski sistem predhodno že vpeljan in formaliziran - osnovni proces organizacijskega sistema. Ta proces opredeljuje množico vseh možnih elementov procesa in njihovih medsebojnih povezav, ki se med prilagajanjem lahko vključijo ali izključijo iz osnovnega procesa.
4. prilagajanje je omejeno samo na proces razvoja poslovnih aplikacij: Vse kar delamo upodablja nek proces, ne glede na to ali je proces dokumentiran ali ne, in ne glede na to, ali ga izvajamo precizno ali ne [Mejabi 1997]. Iz tega sledi, da proces označuje izvajanje zaporedja zelo različnih aktivnosti. Z vpeljavo pričujoče omejitve hočemo poudariti, da se bomo v doktorski disertaciji ukvarjali samo s prilagajanjem procesa razvoja programske opreme, čeprav bi teoretično lahko

obravnavali tudi širšo skupino procesov, ki niso neposredno povezani s področjem razvoja aplikacij.

5. prilagajanje procesa se mora izvajati na nivoju posameznega elementa procesa: S to omejitvijo želimo preprečiti preveliko razdrobljenost procesa. Elemente procesa, ki predstavljajo osnovne gradnike procesa, opredeli metamodel procesa, ki se ga skupaj s procesom predhodno zajame v obravnavanem organizacijskem sistemu (glej razdelek 3.2.2.2).

Če navedenih omejitev ne bi upoštevali, bi lahko problem prilagajanja procesa postal preveč kompleksen ali pa celo neobvladljiv.

2.4.4.2 Formalna opredelitev odločitvenega modela za prilagajanje procesa

Temeljni oziroma osrednji cilj doktorske disertacije predstavlja formalna opredelitev odločitvenega modela za prilagajanja procesa razvoja IS individualni potrebam projektov (1. in 4. predpostavka v razdelku 2.4.4.1). Odločitveni model temelji na uporabi odločitvenih pravil tipa if-then. Odločitvena pravila za vsak element osnovnega procesa opredelijo, ali se ta vključi v prilagojeno različico procesa ali ne. Pri tem izhajamo iz osnovnega procesa, zajetega v organizacijskem sistemu (3. predpostavka v razdelku 2.4.4.1), ki je sestavljen iz elementov, ki jih opredeljuje metamodel zajetega procesa. V podrobnosti prilagajanja posamičnih opisov elementov procesa se ne spuščamo, kar smo predpostavili že v okviru 5. predpostavke iz razdelka 2.4.4.1. Pri prilagajanju procesa je potrebno upoštevati lastnosti oziroma individualne potrebe vzpostavljenega projekta. Te se opredelijo z uporabo karakteristik, ki jih razdelimo v naslednje tri skupine:

- tehnološke karakteristike,
- sociološke karakteristike in
- zahteve naročnika.

Z upoštevanjem več skupin karakteristik hočemo zagotoviti ne samo tehnično prilagojenost predlagane različice procesa potrebam nekega projekta, ampak tudi upoštevanje sociološkega vidika prilagajanja in potreb samega naročnika. Tehnološko popolna metodologija še zdaleč ne zagotavlja, da bo metodologija tudi dejansko sprejeta med njenimi uporabniki. Z upoštevanjem sociološkega vidika v okviru prilagajanja procesa se zagotovi, da uporabniki prilagojene metodologije ne bodo zavrnili, s tem pa tudi tej metodologiji pripadajočega procesa. Sociološki vidik prilagajanja zagotavlja, da je prilagojena metodologija oziroma razvojni proces sestavljen iz elementov, ki so jih uporabniki pripravljeni uporabljati. Na ta način želimo zagotoviti celovito rešitev -

prilagojeno različico procesa, ki bo potrebam projekta prilagojena hkrati tako po tehnični kot po sociološki plati, kar trenutno pogrešamo pri vseh agilnih pristopih, ki se ukvarjajo s prilagajanjem procesa.

Rezultat prilagajanja procesa predstavlja prilagojena različica osnovnega procesa, ti. instanca procesa, ki ustreza karakteristikam projekta, za katerega je bila izdelana.

Odločitveni model bomo zasnovali na temelju predhodno predstavljenega inovativnega pristopa za prilagajanje razvojnega procesa, ki se uporablja v organizacijskem sistemu. Za pojasnjevanje delovanja odločitvenega modela bomo uporabili opisne slike in diagrame aktivnosti, ki jih bomo podrobno opisali. Predlagan pristop k prilagajanju procesa bomo predstavili z uporabo opisov in opisnih slik ter naslednjih formalnih metod: teorije grafov, teorije množic in metamodeliranja.

2.4.4.3 Opredelitev izhodišč za izdelavo programskega orodja za prilagajanje procesa

Drugi pomembnejši cilj doktorske disertacije predstavlja opredelitev izhodišč za izdelavo prototipa programskega orodja za podporo prilagajanja procesa, ki temelji na zasnovi teoretičnih opredelitev prej omenjenega odločitvenega modela. Programsko orodje naj bi kot pripomoček pri svojem delu uporabljal metodolog (ang. methodology engineer). Samo orodje naj bi podpiralo izvajanje naslednjih funkcij:

- vnašanje karakteristik konkretnega projekta: Orodje mora za vsako prilagajanje osnovnega procesa omogočiti vnos karakteristik projekta (tehnološke karakteristike, sociološke karakteristike, zahteve naročnika). Pri tem mora orodje dopuščati tudi vnos nepoznanih vrednosti karakteristik.
- delo z odločitvenimi pravili oziroma administriranje odločitvenih pravil: V okviru dela z odločitvenimi pravili mora orodje podpirati vnašanje novih in brisanje ter posodabljanje že obstoječih pravil v bazi znanja. Nova odločitvena pravila se vnaša po potrebi, na podlagi novega znanja, ki se pridobi skozi izkušnje pri uporabi odločitvenega modela.
- izpis instance procesa: Orodje mora omogočati izpis vseh elementov, ki tvorijo dobljeno instanco procesa v obliki seznama. Hkrati mora biti dosegljiva tudi informacija o tem, kako so elementi v instanci procesa med seboj povezani. Posamične elemente procesa mora biti možno pregledovati, kar je pomembno predvsem za uporabnike procesa - člane projektne skupine.
- pojasnjevanje vključenosti elementov procesa v instanco procesa: V okviru orodja mora biti podprta funkcija, ki omogoča pridobiti odgovor na to, na podlagi katerih

odločitvenih pravil je bil posamezni element vključen v instanco procesa - pojasnjevanje rezultata. S tem se zagotovi transparentnost odločitev, ki jih predlaga odločitveni model v okviru postopka prilagajanja.

Predlagano orodje predstavlja zelo močan pripomoček za pomoč pri delu metodologa. Še posebej, če je podprt z modulom za risanje procesa, ki omogoča preko grafične predstavitve osnovnega procesa organizacijskega sistema vnašati odločitvena pravila za izbor elementov in izris končne - prilagojene različice procesa.

Podrobnejše lastnosti orodja za podporo prilagajanja razvojnega procesa podajamo v poglavju o programskem orodju AMT - ang. Agile Methodology Toolset (glej poglavje 4).

Za predstavitev izhodišč za izdelavo prototipa orodja za podporo prilagajanja procesa bomo uporabili primere uporabe, strukturni diagram, podatkovne modele, opise in slike zaslonskih mask ter pseudokodo za opis programske logike aplikacije.

2.4.5 Umestitev teme doktorske disertacije v okvir scenarija za konstruiranje prilagodljivih metodologij razvoja IS

2.4.5.1 Uvod

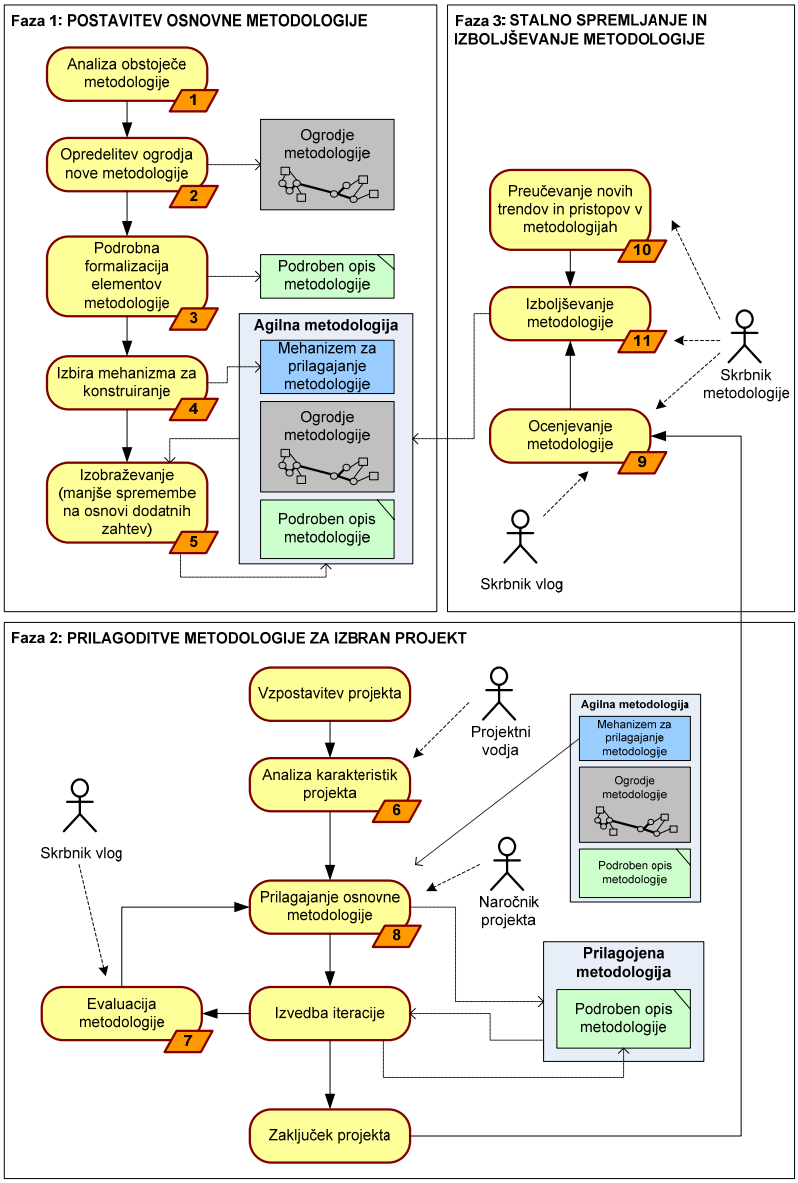
V tem razdelku bomo temo doktorske disertacije umestili v okvir scenarija, ki ga v svojem prispevku predlaga Bajec [Bajec 2004d]. Njegov predlog scenarija obravnava postopek konstruiranja prilagodljivih metodologij razvoja IS, pri čemer upošteva tudi sociološko komponento razvijalcev, ki sodelujejo pri razvoju. Pri tem avtor izhaja iz agilnih metodologij.

2.4.5.2 Predstavitev scenarija

Namen scenarija je opredelitev aktivnosti za konstruiranje prilagodljivih metodologij. Scenarij pomaga vzpostaviti metodologijo razvoja IS, ki najbolj ustreza organizacijskemu sistemu in opredeljuje, kako naj se organizira razvojni proces na način, da bo podpiral agilnost in "ad-hoc" prilagoditve konkretnim potrebam določene situacije ali projekta. Sam scenarij opredeljuje tudi dodatne aktivnosti in vloge, ki skrbijo za akumuliranje novega znanja, pridobljenega med uporabo metodologije. Scenarij sestavljajo naslednje tri faze:

- faza 1: postavitev osnovne metodologije,
- faza 2: prilagoditve metodologije za izbran projekt in
- faza 3: stalno spremljanje in izboljševanje metodologije.

Vsaka faza opredeljuje aktivnosti, ki jih je potrebno izvesti v okviru življenjskega cikla metodologije. Aktivnosti scenarija in faze prikazuje Slika 10. V nadaljevanju sledi opis aktivnosti posameznih faz in njihov namen. Aktivnosti scenarija, ki niso relevantne za doktorsko disertacijo, ne bomo podrobneje opisovali.



Slika 10: Scenarij za konstruiranje prilagodljivih metodologij za razvoj IS [Bajec 2004d]

2.4.5.2.1 Postavitev osnovne metodologije

Namen prve faze je izdelati agilno metodologijo, ki bo prilagojena zahtevam in potrebam organizacijskega sistema. V okviru prve aktivnosti se izvede analize obstoječe metodologije v organizacijskem sistemu. Na podlagi predpostavke, da je v okviru vsakega organiziranega dela mogoče identificirati elemente metodologije, scenarij v prvem koraku predlaga, da se izvede študija obstoječega procesa v organizacijskem sistemu. V okviru študije je potrebno ugotoviti, s katerimi elementi procesa so uporabniki (razvijalci) zadovoljni in kateri so tisti, ki jih vidijo kot problematične. Če imamo opravka z bolj izkušenimi razvijalci, nam omenjena analiza lahko pomaga tudi pri zajemu njihovega skritega znanja (pomen tega je opisan v razdelku 2.2.3). Pogovor z razvijalci lahko odkrije tudi pomembne sociološke elemente v okviru izvajanja procesa, kot so kultura organizacije, odnos posameznika do razvojnega procesa itd., kateri lahko bistveno omejijo bodočo metodologijo.

Po izvedbi analize obstoječega stanja, je potrebno opredeliti ogrodje nove metodologije, kar se izvede v okviru druge aktivnosti. Poleg izbranih elementov stare metodologije, za katere se izkaže, da so koristni, je pri izdelavi ogrodja potrebno upoštevati tudi karakteristike in zahteve organizacijskega sistema. To je še posebej pomembno v primeru, da obstoječi proces ni ustrezno opredeljen, ali če uporabniki z njim niso zadovoljni. Pri tem si pri vzpostavitvi ogrodja lahko pomagamo tudi z obstoječimi komercialnimi metodologijami oziroma procesi, ki so na voljo. Seveda je izbor razvojnega procesa, ki bi bil za organizacijski sistem najbolj ustrezen, zelo težak in zahteva dobro poznavanje različnih razvojnih procesov.

V okviru tretje aktivnosti se izvede podroben opis - formalizacija elementov metodologije. Pri tem ni priporočljivo opisati vsega v celoti, ampak glede na avtorja Cockburna [Cockburn 2002], dokumentirati samo tiste elemente, ki so pomembni in se po predvidevanjih ne bodo kmalu spremenili. V nasprotnem primeru lahko postane vzdrževanje metodologije zelo težko. Pri uporabi agilne metodologije se od uporabnikov pričakuje, da dobro poznajo svoje delo in da se jim ni potrebno postopkov učiti iz metodologije vsakič, ko morajo kaj narediti. V večini primerov naj bi šlo samo za vpoglede, in ne za študij celotne metodologije. Za neizkušene uporabnike pa naj bi metodologija vsebovala sklice na vire in literaturo, kjer se nahajajo podrobnejši opisi njenih elementov.

Četrta aktivnost skrbi za izbor mehanizma za konstruiranje metodologije. Pomembna lastnost agilne metodologije (kot jo navaja Bajec) je namreč njena zmožnost za prilagajanje svoje vsebine specifičnim potrebam projektov. Oblike implementacije takega mehanizma so zelo različne. Pri zelo preprostih gre samo za opredelitev

obveznih in neobveznih aktivnosti procesa. Pri kompleksnih implementacijah pa konstruiranje procesa temelji na uporabi mehanizmov umetne inteligence, ki za "ad hoc" konstruiranje primerka metodologije izhajajo iz metamodela metodologije.

Predpogoj za uspešno vzpostavitev in uporabo agilne metodologije so zadovoljni uporabniki, ki jo popolnoma razumejo in so pripravljeni uporabljati njene elemente. Zaradi tega je potrebno uporabnike predhodno poučiti o predpostavkah in dejstvih, na katerih nova metodologija temelji, kar se izvede v okviru pete aktivnosti. Ker se osnovni proces opredeli na temelju elementov obstoječe metodologije, s katerimi so uporabniki zadovoljni, naj ne bi prihajalo do večjih nesoglasij. Če vendarle obstajajo kakšna nesoglasja med udeleženci, se metodologijo ponovno preuči in ustrezno popravi. Pri tem je potrebno poudariti, da scenarij ne priporoča anarhije, vendar poudarja pomen poslušanja uporabnikov, ki bodo metodologijo dejansko uporabljali.

2.4.5.2.2 Prilagoditve metodologije za izbran projekt

Ko je nova metodologija v organizacijskem sistemu uvedena, je pripravljena za uporabo. Preden pa se jo dejansko uporabi na konkretnem projektu, jo je potrebno prilagoditi še glede na karakteristike tega projekta. Da bi bila nova metodologija čimbolj uporabna, scenarij v okviru druge faze predvideva izvajanje več metodoloških aktivnosti, ki se morajo izvesti v času trajanja projekta. Po vzpostavitvi projekta, morata projektni vodja in metodolog, v okviru šeste aktivnosti, preučiti karakteristike projekta in se na osnovi teh odločiti, kateri elementi osnovne metodologije so za projekt obvezni. Bolj kot je splošna metodologija obširna, bolj je potrebno zožiti množico njenih elementov, ki so za projekt pomembni. Na tem mestu igra ključno vlogo mehanizem za prilagajanje metodologije, ki mora upoštevati relacije med njenimi elementi.

Pomembno vlogo v šesti aktivnosti ima tudi naročnik projekta, ki lahko zahteva določene izdelke, ki morajo nastati med izvajanjem projekta.

Ko se metodologijo prilagodi potrebam projekta, postane na voljo uporabnikom - članom projektne skupine. Scenarij na tem mestu predlaga opredelitev nove vloge, ti. "skrbnika vlog", kateri vzpodbuja razvijalce k uporabi metodologije in njenemu neprestanemu izboljševanju, ki temelji na izkušnjah in znanju, pridobljenem med uporabo metodologije. Med samim izvajanjem iteracij lahko namreč skrbnik vlog spremeni opise metodologije, tako da doda nove smernice in priporočila, o katerih se je predhodno posvetoval s svojimi podrejenimi.

Nadaljnja uporaba metodologije temelji na izbranem življenjskem ciklu. Glede na to, da večina modernih pristopov razvoja programske opreme temelji na iterativnem pristopu,

scenarij po vsaki iteraciji predlaga sklic krajšega sestanka, na katerem se predebatira uporaba metodologije znotraj izvedene iteracije (sedma aktivnost). Člani projekte skupine se pogovorijo samo o bistvenih spremembah, ki bi lahko prispevali k višji učinkovitosti metodologije. Če so predlagane spremembe upravičene, se izvede popravek prilagojene metodologije (osma aktivnost). Pri tem je treba opozoriti, da se popravki izvedejo na metodologiji, ki je bila v osmi aktivnosti predhodno že prilagojena potrebam projekta in da se popravki ne odražajo na osnovni metodologiji, ki je bila za organizacijski sistem vzpostavljena v okviru prve faze scenarija.

2.4.5.2.3 Spremljanje in izboljševanje metodologije

Po zaključenem projektu metodolog, v okviru devete aktivnosti, skliče sestanek s skrbnikom vlog, na katerem se pogovorita o morebitnih spremembah osnovne metodologije in njihovi implementaciji, ki se izvede v okviru enajste aktivnosti. V okviru desete aktivnosti metodolog opravlja raziskave novih trendov in pristopov na področju metodologij, na podlagi katerih se v osnovno metodologijo tudi lahko vnesejo spremembe in izboljšave. Zelo pomembno je, da se pred vnosom sprememb v metodologijo, o tem doseže soglasje, pri čemer gre za upoštevanje socialnih lastnosti organizacijskega sistema. Metodolog nosi pri tem glavno odgovornost, da ostane metodologija socialno ustrezna.

2.4.5.3 Umestitev doktorske disertacije v okvir scenarija

Predlagan scenarij podpira številne aktivnosti za uvedbo prilagodljive - agilne metodologije v organizacijski sistem. Naše delo v okviru doktorske disertacije se osredotoča na vsebino četrte, šeste in osme aktivnosti, ki jih opredeljuje scenarij. V splošnem se tematika doktorske disertacije nanaša na formalno opredelitev odločitvenega modela za prilagajanje procesa razvoja IS konkretnim potrebam projektov, v nadaljevanju pa si poglejmo, kako se vsebina umešča v scenarij, ki ga predlaga [Bajec 2004d].

V četrti aktivnosti scenarija je potrebno izbrati mehanizem za konstruiranje agilne metodologije v organizacijskem sistemu. Doktorska disertacija sovpada s to aktivnostjo z vidika odločitve, da bomo prilagajanje osnovnega procesa v organizacijskem sistemu obravnavali s pomočjo kompleksnega mehanizma, ki temelji na uporabi odločitvenih pravil in ekspertnega sistema kar predstavlja inovativen pristop za prilagajanje razvojnega procesa.

V okviru formalne opredelitve odločitvenega modela, ki predstavlja prej omenjeni kompleksni mehanizem za konstruiranje agilne metodologije, obravnava disertacija tudi izbor in opredelitev ustreznih karakteristik projektov, s katerimi ne opisujemo samo tehnoloških lastnosti projektov ampak tudi sociološke lastnosti članov projektnih skupin. Obravnava karakteristik sovpada z vsebino šeste aktivnosti scenarija.

Jedro doktorske disertacije predstavlja opredelitev načina - odločitvenega modela za konstruiranje prilagojene različice osnovnega procesa v organizacijskem sistemu. Gre torej za samo opredelitev postopka prilagajanja procesa v okviru opredelitve zgradbe odločitvenega modela, s čimer tematika pokrije bistvo osme aktivnosti scenarija.

Tako kot predlaga scenarij, tudi v doktorski disertaciji obravnavamo prilagajanje osnovnega procesa, ki je bil predhodno uveden v organizacijski sistem. S tem si zagotovimo, da izhajamo iz procesa, katerega elementi so med uporabniki že sprejeti in formalizirani. Kako pridemo do osnovnega procesa, nazorno opisuje prikazan scenarij, zato to ni predmet naše obravnave.

3 Prilagajanje procesa razvoja IS

3.1 Uvod

Poglavje s svojo tematiko predstavlja jedro doktorske disertacije. Rdečo nit poglavja predstavlja opredelitev inovativnega pristopa za prilagajanje razvojnega procesa individualnim potrebam projektov v organizacijskem sistemu. Na začetku poglavja najprej podamo izhodišča, na katerih temeljimo pri opredelitvi pristopa. V okviru izhodišč podamo opredelitev in formalizacijo osnovnega procesa v organizacijskem sistemu. Sledi opredelitev karakteristik za opisovanje lastnosti konkretnih projektov in predstavitev ideje o prilagajanju procesa, na kateri temelji predlagan pristop v okviru te disertacije. Osrednji del poglavja predstavlja predstavitev odločitvenih pravil za gradnjo primerkov osnovnega procesa. V okviru tega predstavimo metamodel odločitvenih pravil in opredelimo njihovo zgradbo. V zaključku sledi formalna opredelitev odločitvenega modela za prilagajanje razvojnega procesa konkretnim potrebam projektov, ki predstavlja temeljni cilj in s tem glavni prispevek k znanosti v okviru doktorske disertacije.

3.2 Opredelitev in formalen opis osnovnega procesa

3.2.1 Uvod

Metodologija razvoja programske opreme opredeljuje sistematičen in koordiniran pristop - proces k razvoju IS. Vsak organizacijski sistem, ki se ukvarja z razvojem programske opreme, pri tem uporablja določeno metodologijo. Vprašanje je le, ali se ta metodologija nahaja v neformalni obliki in obstaja samo v obliki skritega znanja v glavah razvijalcev, ali gre za formalizirano metodologijo, z natančno opredeljenim razvojnim procesom, ki je nekje zapisana (papirni dokumenti, spletni dokumenti itd.).

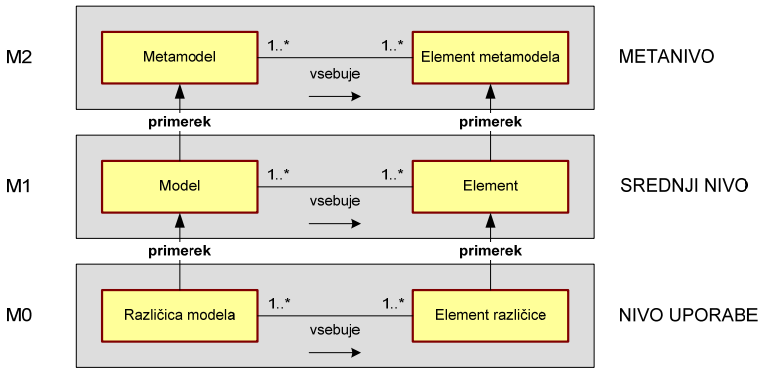
Osnovo za postopek prilagajanja procesa, ki ga predlagamo v doktorski disertaciji, predstavlja osnovni proces organizacijskega sistema, kot smo to navedli že v okviru omejitev, v razdelku 2.4.4.1. Zaradi tega mora biti osnovni proces predhodno formaliziran. Formalizacija osnovnega procesa se izvede v okviru formalizacije zajete metodologije organizacijskega sistema, v sklopu katere se opiše elemente procesa in izdela procesni diagram. V nadaljevanju bomo zato natančneje predstavili postopek

formalizacije osnovne metodologije in podali način za predstavitev procesa, ki ga uporabimo v nadaljevanju disertacije.

3.2.2 Formalizacija metodologije in opredelitev osnovnega procesa

3.2.2.1 Namen postopka formalizacije metodologije

Namen formalizacije metodologije v organizacijskem sistemu je zajeti znanje o aktivnostih, ki se izvajajo v okviru procesa, ki ga ta opredeljuje in o ostalih elementih, ki tvorijo metodologijo. Pri predstavitvi znanja o metodologiji izhajamo iz področja konstruiranja metodologij, ki opredeljuje tri nivoje za predstavitev znanja o metodologijah [Henderson 2005; Prakash 1997]. Omenjene tri nivoje v svojih delih zagovarjo avtorji s področja konstruiranja metodologij, kot so Brinkkemper [Brinkkemper 1999], Henderson-Sellers [Henderson 2005], Smolander [Smolander 1992] in drugi, zato tudi v disertaciji izhajamo iz omenjenih treh nivojev za opis znanja. Opis nivojev smo podali že v okviru razdelka 2.3.1.3, zato se bomo v nadaljevanju osredotočili predvsem na pomen, ki ga imajo omenjeni nivoji za predstavitev znanja o metodologiji, ki se nahaja v organizacijskem sistemu. Nivoje za opis znanja o metodologijah prikazuje Slika 11:



Slika 11: Trije nivoji predstavitve znanja o metodologiji

Metanivo predstavlja najvišji nivo za predstavitev znanja o metodologiji. V okviru tega nivoja se znanje o metodologiji predstavi z uporabo metamodela, ki ga tvorijo elementi metamodela. Pri izdelavi metamodela osnovne metodologije se v organizacijskem sistemu predstavi znanje o uporabljeni metodologiji, in sicer na nivoju M2.

Srednji nivo predstavi znanje o metodologiji v obliki modelov. Vsak model na tem nivoju predstavlja primerek metamodela iz nivoja M2. Prav tako velja, da elementi modelov predstavljajo primerke elementov metamodela. Z izdelavo primerka metamodela metodologije se opredeli procesni model in modele izdelkov. V organizacijskem sistemu se na tem nivoju (nivo M1) izvede konstruiranje splošne metodologije, ki obsega opis njenih elementov in izdelavo procesnega modela osnovnega procesa.

Nivo uporabe M0 (najnižji nivo) predstavi znanje o metodologiji na nivoju njene dejanske uporabe. V okviru tega nivoja se za posamezni projekt izdela različico metodologije, ki je prilagojena njegovim konkretnim potrebam. Različica metodologije predstavlja primerek osnovne metodologije organizacijskega sistema, hkrati pa elementi različice predstavljajo primerke elementov osnovne metodologije. Prilagojena metodologija se uporabi na konkretnem projektu, kjer se njen procesni model izvede, na temelju modelov izdelkov pa se kreirajo izdelki metodologije. V okviru obravnavanega organizacijskega sistema, govorimo na tem nivoju o prilagajanju osnovnega procesa konkretnim potrebam projektov, kar je tudi tema pričujoče doktorske disertacije.

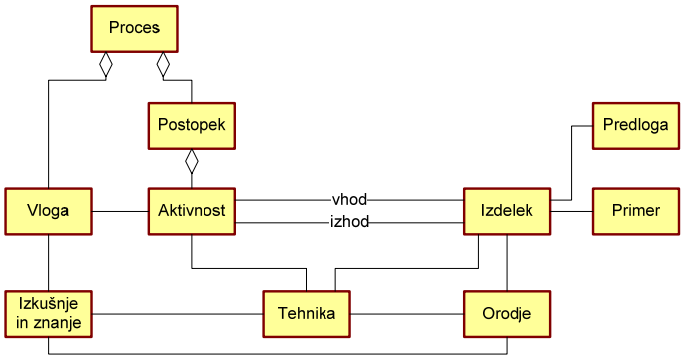
Postopek formalizacije osnovne metodologije zajema naslednje aktivnosti:

- izdelava metamodela osnovne metodologije,
- opis elementov metodologije,
- izdelava procesnega modela.

3.2.2.2 Izdelava metamodela osnovne metodolgije

Metamodel metodologije opredeljuje elemente (koncepte), ki nastopajo v osnovni metodologiji organizacijskega sistema in njihove medsebojne povezave. Povezave med elementi so lahko različnega tipa, in sicer: asociacija, agregacija in generalizacija [Vavpotič 2003]. Povezavi v metamodelu se lahko opredeli tudi številnost, ki določa, s koliko primerki določenega elementa, se primerek opazovanega elementa lahko povezuje [Henderson 2003]. Metamodel osnovne metodologije vsebuje tudi vse elemente procesa, ki sestavljajo osnovni proces. Na podlagi metamodela metodologije je razvidno, kateri metaelementi osnovnega procesa se med seboj lahko povezujejo in kateri ne, številnost povezav v metamodelu pa nam omogoča opredeliti omejitve, s pomočjo katerih lahko opredelimo pravila za zagotavljanje celovitosti prilagojene različice procesa (glej razdelek 3.5.3.6), ki jo dobimo v postopku prilagajanja osnovnega procesa.

Metamodel osnovne metodologije predstavlja tudi temelj za opredelitev elementov procesa, v smislu sestavnih delov iz katerih se gradi prilagojene različice procesa. Množico možnih elementov, ki se lahko pojavljajo v okviru procesnega modela, opredeljujejo elementi metamodela, ki predstavljajo koncepte iz katerih je zgrajen proces obravnavane metodologije. Primer elementov procesa prikazuje izsek metamodela metodologije na sliki (glej Slika 12).



Slika 12: Elementi procesa so opredeljeni z metaelementi metamodela metodologije

Za vsak element procesa obstaja ustrezna predstavitev v okviru treh nivojev predstavitve znanja o metodologijah. Na nivoju M2 je element procesa predstavljen z ustreznim elementom metamodela. Z izdelavo primerka tega metaelementa dobimo element procesa, ki je opredeljen z elementom procesnega modela na nivoju M1 (glej razdelek 2.3.2.2). Omenjeni element predstavlja sestavni del modela osnovnega procesa v organizacijskem sistemu (glej razdelek 3.2.3).

Z izdelavo primerka elementa modela osnovnega procesa dobimo element procesa, ki se uporabi v okviru dejanskega projekta. V tem primeru je element procesa opredeljen na nivoju uporabe. Primeri posameznih elementov procesa so: aktivnost, vhodni izdelek, izhodni izdelek, vloga, tehnika, orodje itd.

3.2.2.3 Opisi elementov metodologije

Formalni opisi elementov metodologije obsegajo tudi opise elementov procesa in ostalih elementov metodologije na določeni stopnji podrobnosti. Pri uporabi agilne metodologije se od uporabnikov pričakuje, da dobro poznajo svoje delo in da se jim ni potrebno postopkov vsakič učiti iz metodologije. V večini primerov naj bi šlo samo za vpoglede, in ne za študij celotne metodologije. Za neizkušene uporabnike pa naj bi metodologija vsebovala sklice na vire in literaturo, kjer se nahajajo podrobnejši opisi

njenih elementov. Pri opisovanju elementov je zato potrebno upoštevati, da ni priporočljivo opisati vsega in da naj se podroben opis izvede samo za najpomembnejše elemente, za katere se predvideva, da se ne bodo kmalu spremenili.

3.2.2.4 Procesni model

Opredelitev procesa navaja akterje, ki izvajajo aktivnosti, njihove vloge in izdelke, ki pri tem nastanejo. V okviru postopka formalizacije prikažemo proces z uporabo procesnega modela (glej razdelek 2.3.3). V našem primeru ta opredeljuje vse aktivnosti, ki se lahko izvajajo v okviru osnovne metodologije, pripadajoče vloge, vse vhodne in izhodne izdelke posameznih aktivnosti, uporabljene tehnike za njihovo izvedbo in orodja. Ker je metodologija izdelana za potrebe celotnega organizacijskega sistema, vsebuje njen procesni model veliko več elementov, kot pa se jih dejansko rabi v okviru posameznega projekta. Poleg tega ima lahko določen element procesa še več različic. Množico vseh možnih elementov, ki sestavljajo osnovni proces, tvorijo torej osnovni elementi procesa in njihove morebitne različice. Na podlagi tega lahko postavimo naslednjo opredelitev osnovnega procesa:

Osnovni proces predstavlja množica vseh aktivnosti, ki se izvajajo v okviru osnovne metodologije v opazovanem organizacijskem sistemu in tem aktivnostim pripadajočih elementov, ki so opredeljeni v metamodelu procesa (vloge, izdelki, orodja, tehnike itd.).

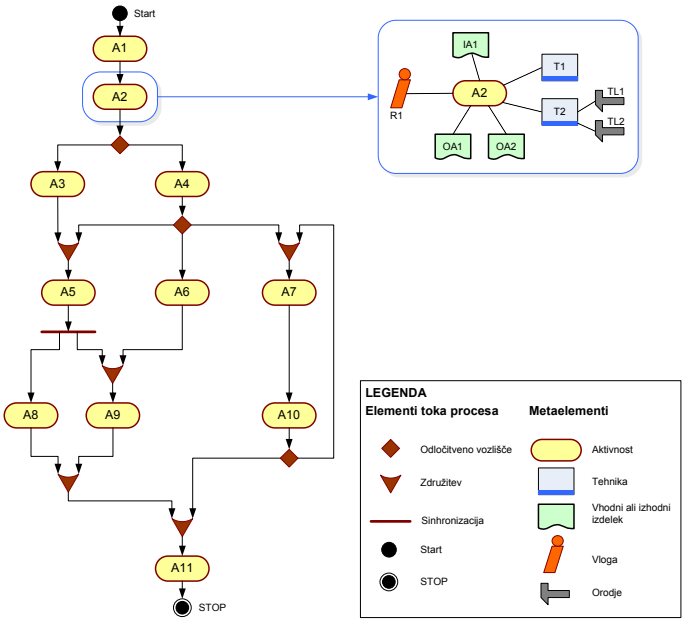
Lahko bi tudi rekli:

Osnovni proces organizacijskega sistema predstavlja celoten proces, ki ga opredeljuje agilna (prilagodljiva) metodologija razvoja IS, ki je bila predhodno vpeljana v organizacijski sistem.

3.2.3 Predstavitev procesa

Za formalni prikaz procesa lahko uporabimo različne diagramske tehnike: eEPC, diagram aktivnosti, diagram podatkovnih tokov itd. V okviru doktorske disertacije smo se odločili za uporabo razširjenega diagrama aktivnosti, ki smo ga opredelili kot skupek običajnega diagrama aktivnosti in množice pojasnjevalnih diagramov, ki se vežejo na posamezne aktivnosti. Diagram aktivnosti prikazuje vse aktivnosti, ki se lahko izvajajo v okviru splošnega procesa v organizacijskem sistemu. V posameznem pojasnjevalnem diagramu pa so prikazani ostali elementi procesa, ki se z določeno aktivnostjo povezujejo neposredno ali posredno. Vse možne elemente, ki se lahko pojavijo v

razširjenem diagramu aktivnosti, oziroma sestavljajo splošni proces, opredeljuje metamodel tega procesa. Najpogostejši predstavniki ostalih elementov procesa so: vloge, izdelki, tehnike, orodja. Primer razširjenega diagrama aktivnosti s samo enim pojasnjevalnim diagramom prikazuje Slika 13:



Slika 13: Razširjen diagram aktivnosti

V diagramu aktivnosti so prikazani tudi ti. kontrolni elementi, ki ne spadajo v množico običajnih elementov procesa, obravnavanih v disertaciji, ampak so pomembni za modeliranje toka procesa. Zaradi tega se tudi ne nahajajo v metamodelu splošnega procesa. V diagramu aktivnosti se lahko pojavijo kontrolni elementi, ki so prikazani v tabeli (glej Tabela 2):

Naziv	Simbol	Pomen
odločitev		Element se uporabi za modeliranje izbire nadaljnjega toka procesa, ki se opredeli z logičnimi pogoji.
sinhronizacija		Element pozna dve različici, in sicer vejitev in združitev. Če gre pri sinhronizaciji za vejitev procesnega toka, se izvajanje aktivnosti v vseh tokovih, ki izhajajo iz simbola, prične v istem časovnem trenutku. Sinhronizacija - združitev modelira združitev več vhodnih procesnih tokov. Aktivnosti v izhodnem procesnem toku se pričnejo izvajati šele, ko so vse aktivnosti vhodnih tokov zaključene.
združitev		Element omogoča modeliranje združitve dveh ali več tokov procesa. Aktivnosti v izhodnem toku se lahko pričnejo izvajati takoj, ko se zaključi izvajanje aktivnosti v vsaj enem vhodnem procesnem toku.

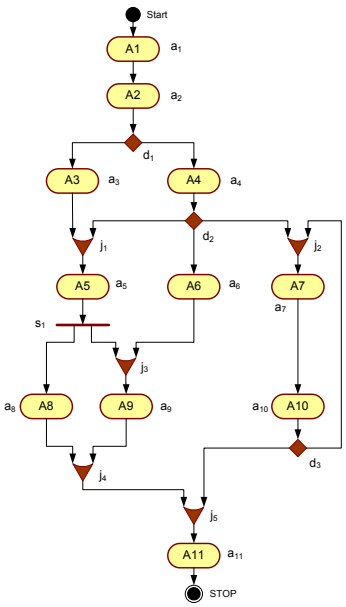
Tabela 2: Kontrolni elementi in njihov pomen

3.2.3.1 Formalen opis diagrama aktivnosti osnovnega procesa

Diagram aktivnosti lahko formalno opišemo s pomočjo elementarne teorije množic. Formalen opis diagrama se tako sestoji iz naslednjih komponent [Hofstede 1998]:

1. množice vseh objektov H, ki predstavlja unijo med množico aktivnosti A in množico kontrolnih elementov C. Množica kontrolnih elementov C predstavlja unijo med množico sinhronizacijskih elementov S, množico odločitvenih elementov D in množico združitvenih elementov J. Elemente množic označujemo enako kot imena pripadajočih množic, vendar z malimi tiskanimi črkami.
2. relacije $Trig \subseteq H \times H$, ki opredeljuje možne usmerjene povezave med elementi v diagramu aktivnosti, oziroma podaja informacijo o tem, kateri element je predhodnik drugega elementa.
3. funkcije $TName : A \rightarrow N$, ki vsaki aktivnosti iz množice aktivnosti A priredi ime iz množice imen N.
4. podmnožice SA množice A, ki vsebuje začetne aktivnosti procesa. Začetna aktivnost predstavlja aktivnost, v kateri se lahko proces prične izvajati.

Diagram aktivnosti iz slike (glej Slika 14) bi z uporabo zgoraj navedenih komponent formalizirali na naslednji način:



Slika 14: Označeni elementi diagrama aktivnosti

$H = \{a_1, a_2, a_3, ..., a_{11}, s_1, d_1, d_2, d_3, j_1, ..., j_5\}$;
$A = \{a_1, a_2, a_3, ..., a_{11}\}$;
$S = \{s_1\}$;

$D = \{d_1, d_2, d_3\}$;
$J = \{j_1, j_2, j_3, j_4, j_5\}$;
a_1 *Trig* a_2, a_2 *Trig* d_1, d_1 *Trig* a_3, a_3 *Trig* j_1 *itd.;*
$TName(a_1) = A1$, $TName(a_2) = A2$, $TName(a_3) = A3$ *itd.;*
$SA = \{a_1\}$;

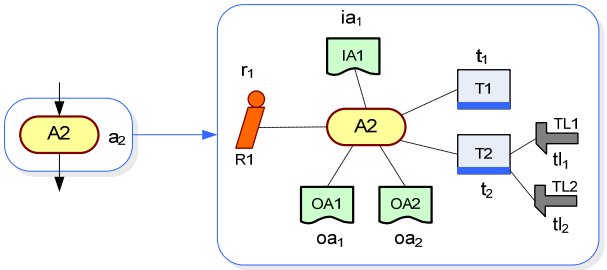
Diagram aktivnosti osnovnega procesa je torej sestavljen iz elementov procesa, kontrolnih elementov in povezav med naštetimi elementi. Povezave med elementi so *fiksne* - določijo se med vzpostavitvijo prilagodljive metodologije v organizacijskem sistemu in se kasneje *ne* spreminjajo več.

3.2.3.2 Formalen opis pojasnjevalnih diagramov

Tudi pojasnjevalne diagrame, ki opredeljujejo, kateri elementi procesa se povezujejo s posamezno aktivnostjo, je možno formalizirati, in sicer z uporabo naslednjih komponent:

1. množice vseh objektov E, ki predstavlja unijo med elementom aktivnost, na katero se pojasnjevalni diagram nanaša in množico preostalih elementov procesa ME, ki so neposredno ali posredno povezani z aktivnostjo. Množica elementov ME predstavlja unijo med množico vlog R, množico izdelkov AR, množico tehnik T in množico orodij TL. Množica AR predstavlja unijo množice vhodnih izdelkov aktivnosti IAR in množice izhodnih izdelkov aktivnosti OAR. Elemente množic označujemo enako kot imena pripadajočih množic, vendar z malimi tiskanimi črkami.
2. relacije $Con \subseteq E \times E$, ki opredeljuje povezave med elementi pojasnjevalnega diagrama z elementom aktivnost oziroma med elementi pojasnjevalnega diagrama samega.
3. funkcije $TName : R \cup AR \cup T \cup TL \rightarrow N$, ki vsakemu elementu iz unije ME priredi ime iz množice imen N.

Primer pojasnjevalnega diagrama, ki pripada aktivnosti A2 prikazuje Slika 15, formalizirali pa bi ga na naslednji način:



Slika 15: Pojasnjevalni diagram

$E = a_2 \cup ME;$
$R = \{r_1\};$
$IAR = \{ia_1\};$
$OAR = \{oa_1, oa_2\};$
$T = \{t_1, t_2\};$
$TL = \{tl_1, \{tl_2\};$
r_1 Con a_2, ia_1 Con a_2, oa_1 Con a_2, oa_2 Con a_2, t_1 Con a_2, t_2 Con a_2, tl_1 Con a_2, tl_2 Con a_2;
$TName(r_1) = R1$, $TName(ia_1) = A1$, $TName(oa_1) = OA1$, $TName(oa_2) = OA2$,
$TName(t_1) = T1$, $TName(t_2) = T2$, $TName(tl_1) = TL1$, $TName(tl_2) = TL2$;

3.3 Karakteristike projektov

3.3.1 Uvod

Za opisovanje lastnosti oziroma značilnosti razvojnih projektov, ki se izvajajo v okviru obravnavanega organizacijskega sistema, lahko uporabimo številne karakteristike. Pristopi za prilagajanje metodologij so do nedavnega dajali večji pomen tehnološkim karakteristikam projektov, zaradi česar so bile izdelane metodologije samo tehnološko ustrezne in so bile še vedno slabo ali celo nesprejete s strani njenih uporabnikov (razvijalcev). Avtorji s področja uporabe metodologij so kasneje ugotovili, da je pri vpeljavi in uspešni uporabi metodologije ključnega pomena tudi upoštevanje njenega socialnega vidika.

Na podlagi tega smo v doktorski disertaciji izhajali iz teze, da za opisovanje lastnosti projektov potrebujemo več tipov karakteristik. Opisovanje lastnosti projektov tako s tehničnega kot sociološkega vidika nam omogoča, da bo prilagojena metodologija poleg tehnične ustreznosti, dosegla tudi željeno sprejetost [Vavpotič 2005] med uporabniki metodologije. Karakteristike projektov smo opredelili na osnovi različnih znanstvenih in strokovnih virov:

- obstoječi modeli za izbiro metodologij ali projektov [Ambler 1998; Avison 2003b; Cockburn 2000a; McConnell 1996; Vavpotič 2002],
- raziskave, ki primerjajo različne metodologije in njihovo primernost za projekte [Budlong 1996; Cockburn 1999, 2000a; Fitzgerald 1998, 1999; Garmus 1996; Hardy 1995; Huismann 2002; Miller 2001; O. Agency 1995].

Na podlagi navedenih skupin virov smo opredelili karakteristike, ki jih njihovi avtorji v svojih raziskavah najpogosteje navajajo pri opisovanju značilnosti projektov. Množica vseh identificiranih karakteristik je zelo obširna, zato smo pri opredelitvi upoštevali le tiste, ki se v prispevkih pojavljajo največkrat, oziroma prestavljajo določen presek med

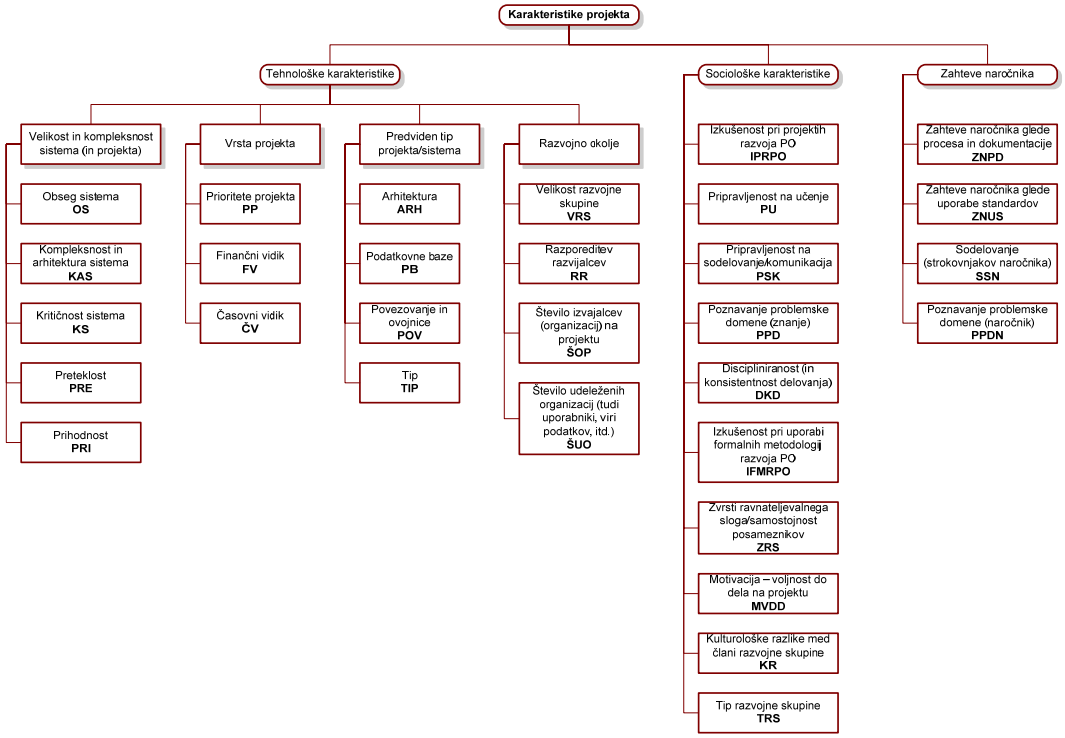
viri. Ugotovili smo, da je obseg identificiranih tehnoloških karakteristik, glede na v nadaljevanju obravnavani nadaljnji dve skupini karakteristik, relativno velik. Vzrok za tako stanje je dejstvo, da se je do nedavnega večina raziskav s področja prilagajanja metodologij osredotočala predvsem na tehnološke karakteristike projektov [Fitzgerald 2003; Garmus 1996; Henderson 2003; Hirch 2002], medtem ko je sociološki vidik razvoja programske opreme ostal relativno zapostavljen [Fitzgerald 2003; Henderson 2003; Highsmith 2002b]. Razmere so se začele spreminjati šele pred nekaj leti, s pojavom modernih agilnih pristopov, ki so sociološkemu vidiku prilagajanja procesa dale vodilni pomen [Cockburn 1996, 1999, 2002a, 2002b in 2002c; Highsmith 2002c], zaradi česar smo na podlagi virov lahko opredelili tudi množico sociooloških karakteristik razvijalcev, ki delujejo v okviru projektov.

3.3.2 Uporabljene skupine karakteristik

Ker v literaturi še vedno pogrešamo celovitejšo in podrobnejšo opredelitev samega pristopa prilagajanja razvojnega procesa (agilnega pristopa razvoja programske opreme) konkretnim potrebam projektov, ki bi agilnost obravnaval na podlagi več karakteristik in na nivoju samega elementa procesa [Zrnec 2004a], smo v disertaciji opredelili tri skupine (tipe) karakteristik, s katerimi lahko predstavimo bistvene lastnosti projekta. To so:

- tehnološke karakteristike,
- sociološke karakteristike in
- zahteve naročnika.

Skupine, podskupine in posamezne karakteristike, ki smo jih identificirali na podlagi analize literature in predstavljajo ti. največji presek med različnimi viri, so podrobno opredeljene v nadaljevanju in jih prikazuje Slika 16.



Slika 16: Skupine karakteristik, karakteristike in podkarakteristike projektov

3.3.3 Formalni vidik karakteristik in njihova predstavitev

3.3.3.1 Uvod

V nadaljevanju so podrobno opredeljene karakteristike, ki jih v disertaciji uporabljamo za opis značilnosti projektov. Nabor karakteristik smo opredelili na temelju znanstvenih in strokovnih prispevkov s področja metodologij razvoja IS. Pri določanju nabora smo želeli zagotoviti, da bodo opredeljene karakteristike čimbolj splošne. Na ta način zagotovimo, da je množica karakteristik uporabna za opis širšega spektra projektov. Vse v nadaljevanju prikazane karakteristike projektov tvorijo ti. množico osnovnih karakteristik *MOK*. Za vsako karakteristiko smo opredelili pripadajočo zalogo vrednosti D_i in podali splošen opis zanjo. Podskupin v okviru tehnoloških karakteristik na tem mestu nismo posebej opisovali, saj nimajo nobenega pomena za vsebino disertacije. Predstavljajo le grupirni mehanizem, za jasnejšo predstavitev celotne strukture karakteristik.

3.3.3.2 Formalizacija projektnih karakteristik

Za formalen opis množice karakteristik in preslikav iz množice projektov v zaloge vrednosti posameznih karakteristik, smo uporabili teorijo množic:

Množica karakteristik *MOK* je opredeljena kot:

(1) $MOK = \{k_1, k_2, k_3,...k_n\}$;

kjer je k_i i-ta karakteristika projekta, n pa število vseh karakteristik v osnovni množici karakteristik.

Množica projektov je opredeljena kot:

(2) $P = \{p_1, p_2, p_3,...p_m\}$;

kjer je s p_i označen i-ti projekt, ki se izvaja v organizacijskem sistemu, m pa označuje število projektov, za katere se je izvedla prilagoditev procesa.

Vsaka karakteristika k_i opredeljuje predpis:

(3) $k_i: P \rightarrow D_i$;

kjer k_i predpisuje preslikavo projekta iz množice *P* v zalogo vrednosti D_i, i-te karakteristike.

Zaloga vrednosti i-te karakteristike je opredeljena kot:

(4) $D_i = \{d_{i,1}, d_{i,2}, d_{i,3},...d_{i,k}\}$;

kjer D_i predstavlja končno množico vrednosti, ki jih karakteristika k_i lahko zavzame, k v indeksu pa število vseh možnih vrednosti v zalogi vrednosti.

Sedaj lahko predpis (3) nadomestimo z naslednjim zapisom:

(5) $d_{i,z} = k_i(p_j), \quad z \in 1..k \text{ in }, \ j \in 1..m$;

kjer $d_{i,z}$ predstavlja z-to vrednost iz zaloge vrednosti D_i, i-te karakteristike.

Vsak projekt je torej mogoče opisati kot končen seznam funkcij, od katerih vsaka opredeljuje preslikavo projekta v svojo zalogo vrednosti:

(6) $p_i = k_1(p_i), k_2(p_i), k_3(p_i),... k_n(p_i)$;

oziroma če predpis k_i nadomestimo z $d_{i,x}$, to je dejansko vrednostjo določene karakteristike, dobimo:

(7) $p_i = d_{1,x}, d_{2,x}, d_{3,x},... d_{n,x};$

kjer x predstavlja indeks dopustne vrednosti karakteristike.

Z drugimi besedami, projekt p_i opišemo z naborom (vektorjem) vrednosti karakteristik.

Množico osnovnih karakteristik *MOK* se lahko kasneje tudi razširi z novimi karakteristikami ali pa se obstoječe karakteristike dopolni, odvisno od potreb, ki se v okviru konkretnega organizacijskega sistema nanašajo na prilagajanje procesa (glej razdelek 3.3.4). Posledično s tem se število komponent vektorja vrednosti karakteristik za določen projekt poveča ali pa zmanjša.

3.3.3.3 Opredelitev tehnoloških karakteristik

Tehnološke karakteristike projektov opisujejo tehnične vidike projekta, za katerega se izvaja prilagajanje procesa. Tehnološke karakteristike smo zaradi sorodnosti združili v štiri podskupine karakteristik, kar je razvidno iz slike (glej Slika 16) in jih navajamo v nadaljevanju disertacije.

3.3.3.3.1 Obseg sistema

D = {Majhen, Srednji, Velik};

Obseg sistema lahko ocenimo na več načinov, kot so naprimer uporaba funkcijskih točk, predvideno število vrstic programske kode, predvideno število razredov itd [Highsmith 2002; Vavpotič 2003].

3.3.3.3.2 Kompleksnost in arhitektura sistema

D ={Majhna, Srednja, Velika};

Pri kompleksnosti in arhitekturi sistema se ocenjuje število zunanjih vmesnikov, število arhitekturnih nivojev, predvideno število podprtih konfiguracij strojne opreme (strojnih platform), predvideno število podprtih konfiguracij podporne programske opreme (programskih platform), število uporabnikov in število različnih tipov uporabnikov, število namestitev odjemalca in število namestitev strežnika itd [Firesmith 2004].

3.3.3.3.3 Kritičnost sistema

D = {Udobje, Denar, Nenadomestljiv denar, Življenje}, [Cockburn 2002];

Karakteristika temelji na predpostavki, da je za bolj kritičen sistem, katerega odpoved ima lahko hujše posledice, potrebna bolj rigorozna metodologija [Cockburn 2000a]. Alistair Cockburn trdi, da ima karakteristika v praksi ponavadi samo dve vrednosti, Nekritično za življenje in Kritično za življenje [Highsmith 2002]. Jim Highsmith predlaga poleg omenjene še eno kategorizacijo, ki kritičnost sistema opiše z naslednjimi vrednostmi:

D = {Življensko kritičen, Kritičen projekt (mission critical), Nekritičen projekt (mission support), Zabava (entertainment), [Highsmith 2002].

3.3.3.3.4 Preteklost

D = {Nov sistem, Nov sistem na osnovi starega (star sistem je strukturen), Nov sistem na osnovi starega (star sistem je OO), Prenova (star sistem je strukturen), Prenova (star sistem je OO)};

Ta karakteristika pove, ali že obstaja starejši sistem, ki je osnova novemu in v kakšnem razvojnem okolju (s kakšnim pristopom) je razvit (strukturnem ali objektnem).

3.3.3.3.5 Prihodnost

D = {Vzdrževanje, Obsežne nadgradnje, Nove različice na tej osnovi};

Ta karakteristika določa kakšna je predvidena prihodnost sistema. V primeru, da nameravamo sistem samo vzdrževati bo verjetno zadostovala manj obsežna dokumentacija kot v primeru, ko predvidevamo nove različice ali obsežnejše nadgradnje [Vavpotič 2003].

3.3.3.3.6 Prioritete projekta

D = {Čas končanja projekta, Sledljivost projekta, Ponovljivost projekta}, [Cockburn 1999, 2000a in 2002];

Ta karakteristika podaja, kaj je za nek projekt prednostno. To je lahko čim krajši čas za dokončanje projekta oziroma čim večja produktivnost, pri čemer je stopnja tolerance za izdelke visoka in dokumentacija minimalna. Lahko je prioriteta sledljivost posameznim delom in izdelkom projekta, kar naprimer pomeni, da mora biti v vsakem trenutku možno ugotoviti, na podlagi katere zahteve je nastal nek izdelek; dokumentacija mora

biti popolnejša! V primeru, da bomo enak ali podoben projekt ponovili, pa je priporočena še popolnejša dokumentacija, ki bo temelj tudi za naslednji projekt.

3.3.3.3.7 Finančni vidik

D = {Predvidena močno omejena količina sredstev, Predvidena zadostna količina sredstev};

V primeru omejene količine sredstev bomo morda morali omejiti funkcionalnost sistema [Charvat 2003; Firesmith 2004].

3.3.3.3.8 Časovni vidik

D = {Zaključka ni mogoče preložiti (predvidljiv urnik), Zaključek je mogoče preložiti (spremenljiv urnik)}, [Firesmith 2004; Vavpotič 2003; Brinkkemper 1996b];

V primeru, ko zaključka projekta ni mogoče preložiti, je potreben predvidljiv plan projekta, v katerem ne bo prišlo do večjih sprememb.

3.3.3.3.9 Arhitektura

D = {Več nivojska, Odjemalec-strežnik, Samostojne aplikacije}, [Vavpotič 2003];

Karakteristika opredeli arhitekturo, katero bo podprl proces razvoja IS.

3.3.3.3.10 Podatkovne baze

D = {Relacijske, Hierarhične, Objektne, Objektno relacijske}, [Vavpotič 2003];

Karakteristika opredeli vrsto podatkovne baze, katero bo podprl proces razvoja IS.

3.3.3.3.11 Povezovanje in ovojnice

D = {Povezovanje (EDI, XML), Ovojnice (CORBA, DCOM)};

Karakteristika opredeli način povezovanja med aplikacijami, katerega bo podprl proces razvoja IS.

3.3.3.3.12 Tip

D = {Spletne storitve, MIS (OLAP, podatkovna skladišča), Posebne aplikacije (sistemske, v realnem času)};

Karakteristika opredeli tip aplikacije, katero bo podprl proces razvoja IS.

3.3.3.3.13 Velikost razvojne skupine

D = {Majhna, Srednja, Velika};

Ta karakteristika označuje velikost celotne razvojne skupine, se pravi število vseh članov v razvojni skupini, tudi članov drugih organizacij, ki na projektu sodelujejo [Charvat 2003].

Razvoj metodologije je funkcija velikosti teama [Highsmith 2002]. Alistair Cockburn trdi, da se s podvojitvijo razvojnega teama komuniciranje tako poveča, da lahko med teamom 20 in teamom 40 ljudi opazimo velike razlike. Od tu tudi njegova klasifikacija velikosti teamov glede na število razvijalcev: 6, 20, 40, 100, 200 in 500 razvijalcev.

Velikost razvojne skupine vpliva na razporeditev razvijalcev [Highsmith 2002]. Pojavi se vprašanje razdalje med razvijalci, ki lahko zavzame več dimenzij: kilometri, časovni pasovi, kulture, različne organizacije.

3.3.3.3.14 Razporeditev razvijalcev

D = {Skupen prostor, Več prostorov, Več stavb, Več lokacij};

Vsi razvijalci so lahko v skupnem prostoru, v več ločenih prostorih, različnih stavbah ali pa celo po več ločenih lokacijah [Ambler 2002]. Razporeditev razvijalcev ima vpliv na vrsto komunikacije med njimi in s tem posredno tudi na učinkovitost izmenjave informacij.

3.3.3.3.15 Število izvajalcev (organizacij) na projektu

D = {1, 2 do 3, 4 do 6, 7 in več};

Gre za število vseh organizacij, ki na projektu sodelujejo kot izvajalci. Večje število izvajalcev vpliva na način in formalnost komuniciranja med njimi.

3.3.3.3.16 Število udeleženih organizacij (tudi uporabniki, viri podatkov itd.)

D = {1, 2 do 3, 4 do 6, 7 in več};

V tem primeru gre za število organizacij, ki sodelujejo na projektu pa niso izvajalci. Pri tem gre poleg uporabnikov tudi za vire podatkov, podjetja za zagotovitev infrastrukture ipd.

3.3.3.4 Opredelitev sociоloških karakteristik

V znanstveni literaturi, ki se ukvarja s področjem agilnih metodologij in prilagajanjem razvojnega procesa konkretnim potrebam projektov, je zaznati trend, ki nakazuje naraščanje števila interdisciplinarnih raziskav na področju metodologij razvoja IS, ki posegajo na področje sociologije [Becker 2002; Domino 2002, Singh 2003, Weinberg 1998]. Tudi Alistair Cockburn v svojem prispevku [Cockburn 2000c] trdi, da so ljudje in s tem njihove lastnosti (znanje, konsistentnost delovanja, izkušnje itd.) na prvem mestu po jakosti vpliva na uspešnost izvedbe projektov, zato jim je na področju raziskav o konstruiranju in prilagajanju metodologij potrebno posvetiti največjo pozornost. V zaključku prispevka gre celo tako daleč in predlaga, da naj bi preučevanje karakteristik razvijalcev postalo primarno področje razvoja programske opreme za nadaljnjih 20-50 let. Na podlagi vsega povedanega in tudi na podlagi lastnih izkušenj z razvojem IS, smo tudi sami prepričani, da je pri opredelitvi postopka za prilagajanje procesa razvoja konkretnim potrebam projektov smiselno upoštevati tudi sociološko komponento razvijalcev, ki jo zajamemo s sociološkimi karakteristikami.

V okviru sociоloških karakteristik opazujemo te z vidika celotne razvojne skupine. To pomeni, da z njimi opišemo sociološke lastnosti celotne skupine in se ne spuščamo na nivo posameznikov v njej. Nekatere, v nadaljevanju naštete sociološke karakteristike, ki so po vplivu na prilagoditve razvojnega procesa pomembne, v svoji magistrski nalogi (na podlagi analize literature) predstavi že Vavpotič [Vavpotič 2003]. Te smo na podlagi nadaljnje analize literature dopolnili še s tremi karakteristikami:

- motivacija - voljnost do dela na projektu,
- kulturološke razlike med člani razvojne skupine in
- tip razvojne skupine.

Sociološke karakteristike, ki jih po vplivu na prilagajanje procesa, avtorji navajajo kot najvidnejše, podajamo v nadaljevanju.

3.3.3.4.1 Izkušenost pri projektih razvoja programske opreme

D = {Slaba, Srednja, Dobra};

Pri tej karakteristiki gre za oceno povprečne izkušenosti članov skupine na projektih s podobno problematiko [Cockburn 2002; Vavpotič 2003]. Oceno lahko dobimo tudi s pomočjo ankete med potencialnimi člani razvojne skupine. Izkušenost ni enako formalnosti procesa.

3.3.3.4.2 Pripravljenost na učenje

D = {Nizka, Srednja, Visoka};

Pri tej karakteristiki gre za oceno, ali so člani projektne skupine pripravljeni sprejemati novo znanje. Oceno se da pridobiti tudi z anketo med potencialnimi člani skupine [Vavpotič 2003].

3.3.3.4.3 Pripravljenost na sodelovanje/komunikacija

D = {Nizka, Srednja, Visoka};

S pomočjo te ocene se poda stopnjo pripravljenosti za sodelovanje med člani skupine ter njihov način komuniciranja [Cockburn 2000b, 2003; Olson 2003/2004; Vavpotič 2003]. Oceno za to karakteristiko lahko dobimo s pomočjo ankete med potencialnimi člani skupine ali pa jo poda vodja projekta, ki je že delal s to skupino.

3.3.3.4.4 Poznavanje problemske domene (znanje)

D = {Slabo poznavanje, Srednje dobro poznavanje, Zelo dobro poznavanje};

Stopnjo poznavanja lahko ocenimo na podlagi informacije o tem, na koliko podobnih projektih je projektna skupina že sodelovala [Cockburn 1996, 2002; Highsmith 2002; Vavpotič 2003; Zrnec 2004a].

3.3.3.4.5 Discipliniranost (in konsistentnost delovanja)

D = {Nizka, Srednja, Visoka};

Ta karakteristika opisuje stopnjo, do katere projektna skupina spoštuje standarde, da deluje v skladu z navodili [Cockburn 2000, 2000b in 2000c; Vavpotič 2003; Zrnec 2004a], da razvijalec prostovoljno izbere način dela, katerega rezultat so skladni in ustrezni izdelki (npr. enoten način pisanja kode s komentarji, upoštevanje skupnih standardov za dokumentiranje itd.), ali člani projektne skupine tudi sami storijo očitno potrebne stvari ali pa za vse potrebujejo eksplicitna navodila.

3.3.3.4.6 Izkušenost pri uporabi formalnih metodologij razvoja PO

D = {Slaba, Srednja, Dobra};

Karakteristika ocenjuje izkušenost članov projektne skupine z uporabo metodologij razvoja programske opreme [Vavpotič 2003].

3.3.3.4.7 Zvrsti ravnateljevalnega sloga /samostojnost posameznikov

D = {Avtokratski, Srednji, Liberalni};

Od zvrsti ravnateljevalnega sloga je precej odvisna izbira aktivnosti in formalnost njihovih opisov [Vavpotič 2003; Mihelčič 1999].

3.3.3.4.8 Motivacija - voljnost do dela na projektu

D = {Nizka, Srednja, Visoka};

Mihelčič [Mihelčič 1999] opredeli voljnost do dela kot stopnjo želje in pripravljenosti posameznika vložiti napor za uresničitev neke ravni učinkov. Pri tem je motivacija v resnici orodje za spreminjanje možne voljnosti v resničnost. Karakteristika podaja oceno za raven motiviranosti članov razvojne skupine za delo na projektu. Avtorja, ki v svojih delih tudi omenjata motivacijo sta še Cockburn in DeMarco [Cockburn 2000b, DeMarco 1999].

3.3.3.4.9 Kulturološke razlike med člani razvojne skupine

D = {Majhne, Srednje, Velike};

Olson v okviru prispevka o kulturnih razlikah med člani razvojne skupine [Olson 2003/2004] opredeli njihov vpliv na način komuniciranja v porazdeljeni skupini. Pri tem obravnava tudi pet značilnosti kultur, ki jih v svojih delih opredeljuje priznani avtor Hofstede [Hofstede 2001]. Navedena karakteristika podaja oceno za kulturne razlike med člani razvojne skupine. Kulturne razlike imajo velik vpliv na vrsto in način komunikacije in s tem na stopnjo formalnosti potrebne metodologije.

3.3.3.4.10 Tip razvojne skupine

D = {skupina za reševanje ponavljajočih problemov; skupina za reševanje novih problemov; inovativna skupina; raziskovalna se skupina}

Glede na potrebe in cilje razvojnih skupin, Armour v svojem prispevku [Armour 2001] opredeli štiri različne tipe razvojnih skupin. Pri tem pride do ugotovitve, da mora biti razvojni proces, za različne tipe skupin drugačen, če hočemo, da bodo pri svojem delu uspešne. Armour opredeli naslednje zahteve za vsak tip skupine:

- skupina za reševanje ponavljajočih problemov: Ključni cilj je sledenje planu. Skupina potrebuje za delo dobro opredeljene vloge in proces. Ker so rešitve problemov znane, je zaporedje aktivnosti do potankosti predpisano.

- skupina za reševanje novih problemov: Ključni cilj je rešitev problema. Skupina potrebuje dobro opredeljene vloge, med člani mora biti vzpostavljeno zaupanje. Proces dopušča majhna odstopanja.
- inovativna skupina: Ključni cilj je razvoj novega programskega produkta. Skupina pri delu ne sme biti omejena z vidika aktivnosti procesa. Dopustna so večja odstopanja od predvidenih aktivnosti procesa.
- raziskovalna skupina: Ključni cilj je razvoj pristopov in procesov, ki prispevajo novo znanje. Uspešno delo temelji na uporabi konsistentnih modelov in jezikov za sporazumevanje. Aktivnosti procesa niso opredeljene.

3.3.3.5 Zahteve naročnika

Sodobne metodologije za razvoj IS temeljijo na intenzivnem sodelovanju projektne ekipe z naročnikom projekta. Zaradi tega je pri opisu projekta potrebno zajeti tudi zahteve, ki jih poda naročnik glede uporabljenega procesa, ker imajo te neposreden vpliv na vključevanje oziroma izključevanje nekaterih aktivnosti v oziroma iz različice procesa, ki jo bomo prilagajali. V disertaciji smo opredelili naslednje karakteristike, s katerimi opredelimo zahteve naročnika:

3.3.3.5.1 Zahteve naročnika glede procesa in dokumentacije

D = {Lahke, Srednje, Rigorozne};

Karakteristika govori o tem, kakšen proces razvoja zahteva naročnik. Npr. naročnik lahko v okviru procesa zahteva izdelavo obsežne dokumentacije ali pa ga dokumentacija (razen uporabniške) sploh ne zanima.

3.3.3.5.2 Zahteve naročnika glede uporabe standardov

D = {Osnovne, Srednje, Visoke};

Karakteristika govori o tem, kakšne zahteve ima naročnik glede uporabe standardov pri razvoju programske opreme [Firesmith 2004]. Npr. ali naročnik zahteva ustrezanje standardu ISO/IEC 12207:1995 [Theunissen 2003].

3.3.3.5.3 Sodelovanje (strokovnjakov naročnika)

D = {Slabo, Srednje, Dobro};

Karakteristika govori o tem, kako dobro so strokovnjaki s strani naročnika pripravljeni sodelovati pri razvoju programske rešitve. Vrednost karakteristike je odvisna od

vrednosti karakteristike, ki sledi v nadaljevanju - poznavanje problemske domene (naročnik).

3.3.3.5.4 Poznavanje problemske domene (naročnik)

D = {Slabo, Srednje, Dobro};

Karakteristika podaja oceno za raven poznavanja problemske domene s strani naročnika [Firesmith 2004].

3.3.4 Ažuriranje nabora obstoječih karakteristik

Nabor karakteristik, ki smo ga opredelili v predhodnem razdelku, predstavlja splošen nabor karakteristik, ki je primeren za širšo množico razvojnih projektov. Če hočemo nabor razpoložljivih karakteristik iz množice *MOK* prilagoditi zahtevam za opis projektov v določenem organizacijskem sistemu, je potrebno obstoječi nabor ažurirati. Ažuriranje nabora karakteristik obsega dvoje, in sicer:

- spreminjanje obsega nabora karakteristik v množici *MOK* in
- posodabljanje zalog vrednosti D_i, posameznih karakteristik.

Pojem ažuriranja nabora bomo natančneje pojasnili v nadaljevanju opisa. Obseg in vrsta upoštevanih karakteristik sta odvisna od številnih dejavnikov, ki vplivajo na njihov izbor, in sicer od:

- organizacijskega sistema, v katerem se izvaja prilagajanje procesa,
- preferenc glede na katere želimo proces bolj prilagoditi s tehničnega ali sociološkega vidika [Vavpotič 2005],
- zahtevane stopnje prilagojenosti procesa njegovim uporabnikom itd.

Pri spreminjanju osnovne množice karakteristik *MOK* je potrebno zagotoviti, da spremenjen nabor karakteristik MOK_{mod} še vedno zadošča zahtevanim lastnostim, ki veljajo za nabore [Bohanec 1995], in sicer:

- polnost,
- operativnost,
- razstavljivost,
- neredundantnost,
- minimalnost,
- ortogonalnost itd.

3.3.4.1 Spreminjanje obsega nabora obstoječih karakteristik

Osnovno množico karakteristik *MOK* (glej razdelek 3.3.3) je mogoče razširiti z novimi karakteristikami ali pa skrčiti, tako da iz obstoječega nabora odvzamemo določene karakteristike. Obseg novega nabora karakteristik je odvisen predvsem od vrste organizacijskega sistema, njegovih potreb in zahtev glede standardov, ki jim želijo v organizacijskem sistemu zadostiti. Nabor se razširi oziroma skrči s tistimi karakteristikami, za katere metodolog v organizacijskem sistemu presodi, da jih je potrebno dodati oziroma odvzeti, glede na preference v konkretni organizaciji. Za spremenjen nabor velja sledeče:

Pri dodajanju karakteristik velja: $MOK_{\text{mod}} = MOK \cup \{k_1, k_2, ...k_x\}$, kjer so k_1 do k_x oznake za karakteristike, x pa število vseh novih karakteristik.

Pri odvzemanju karakteristik velja: $MOK_{\text{mod}} = MOK - \{k_1, k_2, ...k_y\}$, kjer so k_1 do k_y oznake za karakteristike, ki pripadajo osnovni množici karakteristik *MOK* in so bile izločene iz osnovnega nabora karakteristik.

3.3.4.2 Posodabljanje zalog vrednosti karakteristik

Posamične zaloge vrednosti karakteristik iz množice *MOK*, je možno tudi posodabljati. Postopek posodabljanja karakteristike je opredeljen kot sprememba zaloge vrednosti D_i, ki pripada karakteristiki k_i. Iz tega sledi, da se lahko zaloga vrednosti določene karakteristike razširi ali zmanjša.

Pri razširjanju zaloge vrednosti i-te karakteristike velja: $D_{i,nova} = D_i \cup \{d_{i,1}, d_{i,2}, ...d_{i,x}\}$, kjer so $d_{i,1}$ do $d_{i,x}$ oznake novih vrednosti v zalogi vrednosti D_i, x pa število vseh novih vrednosti.

Pri zmanjševanju zaloge vrednosti velja: $D_{i,nova} = D_i - \{d_{i,1}, d_{i,2}, ...d_{i,y}\}$, kjer so $d_{i,1}$ do $d_{i,y}$ oznake obstoječih vrednosti v zalogi vrednosti D_i in so bile izločene iz osnovnega nabora karakteristik.

Na spremembo zaloge vrednosti, določene karakteristike iz osnovne množice karakteristik *MOK*, vplivajo predvsem zahteve, ki se nanašajo na želeno raven podrobnosti glede prilagojenosti procesa razvoja IS konkretnim potrebam projektov. O spremembi zaloge vrednosti neke karakteristike se odloča metodolog, ki ima izkušnje z vplivom karakteristik na izbor elementov procesa, in sicer na osnovi izkušenj,

pridobljenih skozi uporabo, v nadaljevanju disertacije predstavljenega, pristopa za prilagajanje procesa.

3.4 Pristop k prilagajanju procesa

3.4.1 Uvod

V tem podpoglavju je na konceptualnem nivoju predstavljen pristop za prilagajanje osnovnega procesa v organizacijskem sistemu, potrebam individualnih projektov, ki se v okviru obravnavanega organizacijskega sistema izvajajo. Opredelitev pristopa predstavlja jedro pričujoče disertacije, saj uvaja novo idejo o načinu prilagajanja razvojnega procesa. Pri opredelitvi pristopa smo izhajali iz dosedanjih ugotovitev o načinih prilagajanja metodologij potrebam organizacijskih sistemov ali konkretnih projektov in iz zastavljenih ciljev disertacije, ki so podani v okviru razdelka 2.4.4.

3.4.2 Splošna predstavitev ideje o prilagajanju procesa

3.4.2.1 Osnova za prilagajanje

Po uspešno izvedeni aktivnosti uvedbe prilagodljive metodologije v organizacijski sistem, je splošni proces razvoja IS formalno opisan, njegovi elementi pa naj bi bili s strani uporabnikov metodologije tudi sprejeti. Osnovni proces predstavimo z uporabo razširjenega diagrama aktivnosti (glej razdelek 3.2.3). Iz opredelitve osnovnega procesa sledi, da so v tem diagramu predstavljene vse aktivnosti, ki jih osnovni proces obsega in vsi ostali elementi procesa, ki se neposredno ali posredno povezujejo s posameznimi aktivnostmi. Aktivnosti in ostali elementi procesa, ki jih obravnavamo pri prilagajanju procesa so opredeljeni v okviru metamodela zajetega procesa, ki se izdela hkrati z zajemom in vzpostavitvijo prilagodljive metodologije v organizacijskem sistemu.

3.4.2.1.1 Formalni opis razširjenega diagrama aktivnosti z grafom

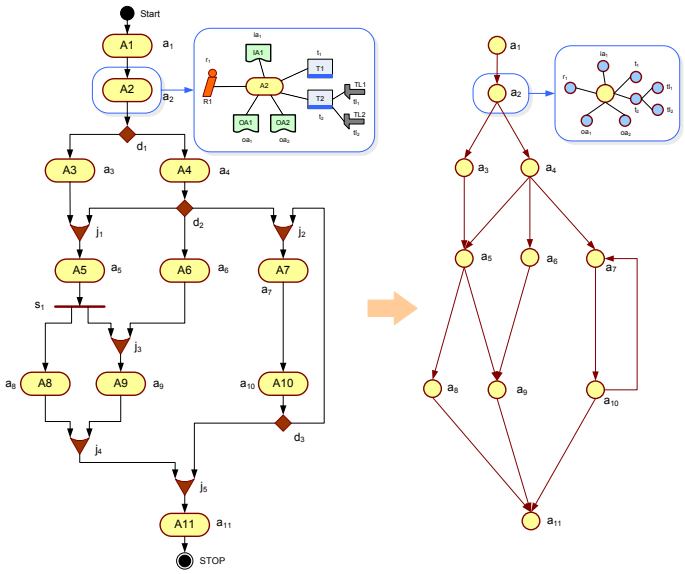
Diagram aktivnosti, ki se uporabi za opis splošnega procesa, vsebuje poleg elementov procesa, ki so opredeljeni že na nivoju metamodela, tudi ti. kontrolne elemente za prikazovanje vejanja in združevanja procesnega toka (glej razdelek 3.2.3). Ker kontrolni elementi nimajo vpliva na način prilagajanja procesa, kot ga predlagamo v okviru obravnavanega pristopa in hkrati niso opredeljeni kot elementi procesa, v okviru metamodela v organizacijski sistem uvedene metodologije, je smiselno, da diagram

aktivnosti in hkrati tudi pojasnjevalne diagrame predstavimo z uporabo enotnega grafa. Razširjen diagram aktivnosti zato predstavimo in formalno opišemo z uporabo ti. mešanega grafa [West 2000], za katerega je značilno, da poleg usmerjenih vsebuje tudi neusmerjene povezave.

Mešan graf je opredeljen z urejeno trojko $G = (V, E, A)$, kjer predstavlja:

- V množico vseh vozlišč v grafu,
- E množico neurejenih parov medsebojno različnih vozlišč - neusmerjenih povezav in
- A množico urejenih parov vozlišč - usmerjenih povezav.

Preslikavo razširjenega diagrama aktivnosti v pripadajoč mešan graf, prikazuje Slika 17.



Slika 17: Preslikava razširjenega diagrama aktivnosti v mešan graf

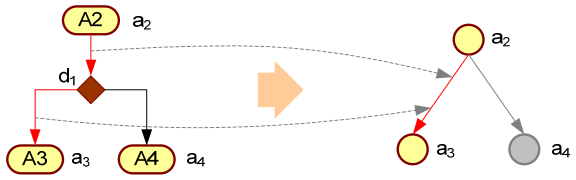
Rezultat preslikave predstavlja *graf osnovnega procesa*[20]. Pri preslikavi razširjenega diagrama aktivnosti se elementi procesa iz omenjenega diagrama preslikajo v vozlišča grafa s pomočjo funkcije *Transf*. Vsako vozlišče iz množice V predstavlja določen element procesa. Ker se v vozlišča grafa preslikajo tako aktivnosti kot preostali elementi procesa, imamo v grafu *različne tipe vozlišč*. Teh je toliko, kolikor različnih elementov procesa opredeljuje metamodel osnovne metodologije, ki se uporablja v okviru organizacijskega sistema. Zaradi večje preglednosti samega grafa na sliki (glej Slika

[20] *Mešan graf, ki predstavlja elemente osnovnega procesa in povezave med njimi.*

17), ta vsebuje samo en primer preslikave elementov, ki pripadajo pojasnjevalnemu diagramu. Funkcija *Transf* je opredeljena kot preslikava iz unije množic *A* in *ME* v množico vozlišč *V*, grafa osnovnega procesa *G* (glej razdelek 3.2.3), kar zapišemo:

(1) $Transf : A \bigcup ME \rightarrow V$;

Usmerjene povezave iz diagrama aktivnosti, ki povezujejo aktivnosti neposredno ali preko kontrolnih elementov, se preslikajo v usmerjene povezave med vozlišči grafa s pomočjo surjektivne funkcije *TransfA*. Zaradi surjektivnosti funkcije *TransfA*, se lahko ena ali več usmerjenih povezav iz razširjenega grafa aktivnosti (eden ali več elementov relacije *Trig*) preslikajo v eno (isto) usmerjeno povezavo med dvema vozliščema grafa *G*. Primer take preslikave prikazuje Slika 18.



Slika 18: Primer preslikave iz relacije Trig v množico usmerjenih povezav A

Funkcija *TransfA* je opredeljena kot preslikava iz relacije *Trig* v množico usmerjenih povezav *A* grafa *G* (glej razdelek 3.2.3.1), kar zapišemo:

(2) $TransfA : Trig \rightarrow A$;

Tako bi preslikavo usmerjenih povezav iz diagrama aktivnosti, v usmerjeno povezavo grafa *G* iz slike (glej Slika 18), z uporabo funkcije (2) opisali na naslednji način:

$TransfA(a_2\ Trig\ d_1) = (a_2, a_3);$
$TransfA(d_1\ Trig\ a_3) = (a_2, a_3);$

Povezave med aktivnostmi in ostalimi elementi procesa, ki so prisotne v pojasnjevalnih diagramih, pa se preslikajo v neusmerjene povezave med ustreznimi vozlišči grafa s pomočjo unkcije *TransfE*. Ta je opredeljena kot bijektivna preslikava iz relacije *Con* v množico *E* (glej razdelek 3.2.3.2), kar zapišemo:

(3) $TransfE : Con \rightarrow E$;

V formalni obliki se graf osnovnega procesa v organizacijskem sistemu opiše na naslednji način:

$V = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, r_1, ia_1, oa_1, oa_2, t_1, t_2, tl_1, tl_2, ...\};$

$E = \{(r_1, a_2), (ia_1, a_2), (oa_1, a_2), (oa_2, a_2), (t_1, a_2), (t_2, a_2), (tl_1, t_2), (tl_2, t_2), ...\};$

$A = \{(a_1, a_2), (a_2, a_3), (a_2, a_4), (a_3, a_5), (a_4, a_5), (a_4, a_6), (a_4, a_7), (a_5, a_8), (a_5, a_9), (a_6, a_9), (a_7, a_{10}), (a_{10}, a_7), (a_8, a_{11}), (a_9, a_{11}), (a_{10}, a_{11})\};$

Ker nismo našteli vseh elementov procesa, tudi v množici vozlišč *V* in množici neusmerjenih povezav *E* nismo navedli vseh elementov.

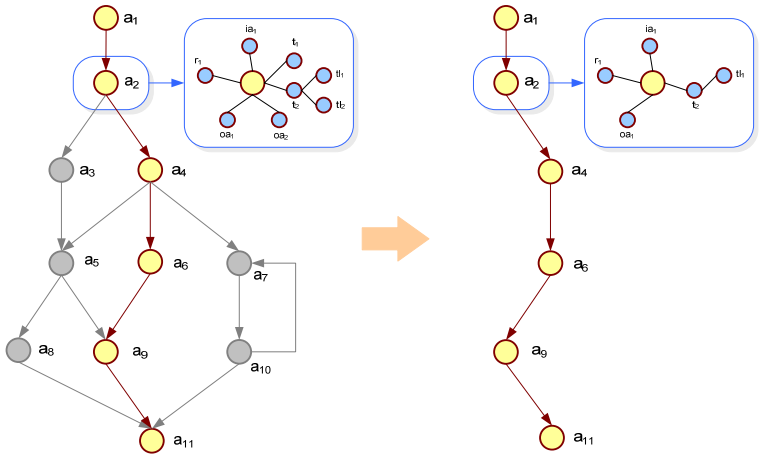
3.4.2.2 Predstavitev pristopa za prilagajanje procesa

Predlagan pristop za prilagajanje procesa predvideva, da v pridobljenem grafu *G* poiščemo pot od izbranega začetnega vozlišča do končnega vozlišča, ki v grafu *G* predstavljata vozlišči tipa aktivnost. Pot predstavlja prilagojen proces za določen projekt. Pristop temelji na strategiji izbire različnih poti znotraj metodologije, o kateri smo govorili v razdelku o stopnjah prilagajanja metodologij (glej razdelek 2.3.1.2). Pri tem nadgrajuje omenjeno strategijo tako, da odpravi omejitev glede števila možnih prehodov skozi proces. Naš pristop ne predpiše vseh možnih poti skozi proces vnaprej. Prilagajanje procesa poteka tako, da se v odvisnosti od lastnosti projekta v prilagojeno različico procesa sproti dodajajo ustrezne - izbrane aktivnosti osnovnega procesa (vozlišča grafa) in tako sproti gradi pot skozi graf *G*. Prilagojene različice procesa torej *ne dobimo vnaprej, že pred začetkom izvajanja projekta*, ampak se ta gradi *med samim izvajanjem* projekta, saj so nekatere odločitve o izbiri ustreznih aktivnosti odvisne od trenutnih razmer, ki nastopijo med izvajanjem. Pri tem gre predvsem za odločitve glede izbire ustreznega delovnega toka znotraj osnovnega procesa. Vsaka tako opredeljena pot v grafu oziroma prehod skozi osnovni proces predstavlja različico procesa, ki je prilagojena potrebam individualnega projekta. Na ta način je možno število prehodov skozi diagram aktivnosti lahko bistveno večje, kot pri strategiji izbire poti znotraj metodologije, hkrati pa je dobljena različica procesa natančneje prilagojena (se bolj prilega) potrebam trenutnega projekta.

Pri iskanju poti skozi graf, se poiščejo le aktivnosti, ki ustrezajo potrebam projekta. Za vsako od njih pa je potrebno izbrati še ustrezno podmnožico preostalih elementov procesa, ki se z njo povezujejo. To se izvede na podlagi zajetih karakteristik projekta, za katerega se izvaja prilagajanje osnovnega procesa.

3.4.2.2.1 Formalen opis grafa prilagojenega procesa

Prilagojeno različico procesa predstavimo z novim grafom *G'*, ki vsebuje podmnožico vozlišč in povezav grafa osnovnega procesa *G*. Vozlišča v novem grafu so lahko različnih tipov in predstavljajo aktivnosti in pripadajoče elemente procesa, ki tvorijo prilagojeno različico procesa, kar prikazuje Slika 19.



Slika 19: Graf osnovnega procesa in graf prilagojene različice procesa

Dobljeni mešani graf prilagojene različice procesa *G'* se v formalni obliki opiše kot urejena trojka:

G' = (V', E', A'), kjer velja, da je:

$V' \subseteq V$, in sicer $V' = \{ a_1, a_2, a_3, a_4, a_6, a_9, a_{11}, r_1, ia_1, oa_1, t_2, tl_1, ..\}$;

$E' \subseteq E$, in sicer $E' = \{(r_1, a_2), (ia_1, a_2), (oa_1, a_2), (t_2, a_2), (tl_1, t_2), ...\}$;

$A' \subseteq A$, in sicer $A' = \{(a_1, a_2), (a_2, a_4), (a_4, a_5), (a_5, a_9), (a_9, a_{11})\}$;

3.4.2.3 Odločitvena pravila

Potek prilagajanja oziroma gradnje različice procesa usmerjajo odločitvena pravila. Odločitvenih pravil je običajno že v zasnovi postopka zelo veliko, skozi proces pridobivanja novih izkušenj, v okviru uporabe predlaganega pristopa (glej razdelek 3.4.3), pa se nabor še povečuje. Zaradi tega je odločitvena pravila smiselno urediti glede na namen uporabe, kar bomo podrobno predstavili v razdelku o odločitvenih pravilih (glej razdelek 3.5). Predlagan pristop v grobem predvideva uporabo treh pomembnejših

skupin odločitvenih pravil. V prvo skupino (množica *FR*) se uvrščajo pravila, ki se nanašajo na izbor ustreznega delovnega toka znotraj osnovnega procesa. V drugi skupini (množica *SR*) se nahajajo pravila za izbor elementov procesa, ki so povezani z aktivnostmi (vloga, izdelek, tehnika, orodje itd.). V tretjo skupino (množica *CR*) pa se uvrščajo ti. posebna pravila za zagotavljanje celovitosti prilagojene različice procesa.

Pri umestitvi odločitvenih pravil v kontekst predlaganega pristopa ima ključni pomen graf osnovnega procesa *G*. Tega smo opredelili kot skupek vozlišč (aktivnosti) ter usmerjenih in neusmerjenih povezav, ki predstavijo osnovni proces organizacijskega sistema na višjem nivoju abstrakcije. Predlagan pristop tako predvideva vzpostavitev povezave med prvo skupino pravil (množica *FR*) in množico usmerjenih povezav v grafu ter med drugo skupino pravil (množica *SR*) in množico neusmerjenih povezav v grafu osnovnega procesa. Na ta način je mogoče:

- usmerjeni povezavi v grafu pripisati odločitveno pravilo iz prve množice pravil *FR*, ki pove, pod kakšnimi pogoji se bo naslednica določene aktivnosti (prikazana z vozliščem na koncu usmerjene povezave v grafu osnovnega procesa *G*) vključila v prilagojeno različico procesa.
- vsaki neusmerjeni povezavi v grafu pripisati odločitveno pravilo iz druge množice pravil *SR*, ki pove, ali se glede na zajete *karakteristike projekta*, določen element procesa vključi v različico procesa ali ne.

Povezavo med skupino pravil *FR* in množico usmerjenih povezav *A*, grafa *G*, formalno opišemo s funkcijo *AssignRuleA*, ki vsako odločitveno pravilo iz množice pravil *FR* pripiše natanko eni povezavi iz množice usmerjenih povezav *A*, grafa osnovnega procesa *G*:

(4) $AssignRuleA : FR \rightarrow A$;

Povezavo med skupino pravil *SR* in množico neusmerjenih povezav *E*, grafa *G*, formalno opišemo s funkcijo *AssignRuleE*, ki vsako odločitveno pravilo iz množice pravil *SR* pripiše natanko eni povezavi iz množice neusmerjenih povezav *E*, grafa osnovnega procesa *G*:

(5) $AssignRuleE : SR \rightarrow E$;

Rečemo lahko tudi, da je obstoj povezave opredeljen z odločitvenim pravilom, ki je tej povezani pripisano. Če je pogoj v pogojnem delu pravila resničen, potem povezava obstaja, v nasprotnem primeru pa ne.

Omenimo naj, da ni nujno, da se vsaki povezavi v grafu pripiše odločitveno pravilo. To v primeru usmerjenih povezav pomeni, da vozlišča v grafu (tipa aktivnost), ki se nahaja na koncu povezave, ni možno izločiti, zaradi česar je to prisotno v vsakem grafu *G'* različice procesa - govorimo o ti. *fiksni* aktivnosti oziroma o aktivnosti, ki se je vključena v vse različice procesa. Podobno velja za preostale elemente procesa (vozlišča grafa), ki so z neusmerjenimi povezavami povezani z aktivnostmi ali drugimi elementi.

Tretja skupina predstavlja posebno obliko pravil, s pomočjo katerih se glede na podatke v metamodelu osnovnega procesa preverja celovitost prilagojene različice procesa, kar bo podrobno opisano v okviru razdelka o odločitvenih pravilih (glej razdelek 3.5). Poleg predlaganih treh skupin pravil, pristop predvideva tudi opredelitev dodatnih odločitvenih pravil, ki se uporabljajo v okviru mehanizma sklepanja in tvorijo ti. množico pravil sklepanja. Ta za predstavitev osnovne ideje pristopa ni tako pomembna, zato jo na tem mestu samo omenjamo.

Na koncu razdelka o odločitvenih pravilih naj omenimo še pojem *situacije*. V grafu osnovnega procesa se lahko pojavijo različne situacije v smislu števila vhodnih in izhodnih povezav nekega vozlišča. Glede na to je potrebno pravilno opredeliti pogoje v odločitvenih pravilih, ki so pripisani izhodnim povezavam aktivnosti in izhodnim povezavam drugih elementov v obravnavani situaciji. Opis situacij in omejitve, ki jih je potrebno upoštevati pri opredeljevanju pravil v okviru situacij, bomo predstavili v razdelku o situacijah (glej razdelek 3.5.4).

3.4.3 Predvidena uvedba predlaganega pristopa v organizacijski sistem

3.4.3.1 Uvod

Uvedba predlaganega pristopa za prilagajanje oziroma konstruiranje različic procesa glede na individualne potrebe projektov in njegova kasnejša uporaba v okviru organizacijskega sistema poteka postopoma. V okviru uvedbe smo predvideli naslednje tri faze:

- fazo konstruiranja prilagodljivega (osnovnega) procesa organizacijskega sistema,
- fazo učenja in
- fazo uporabe pristopa.

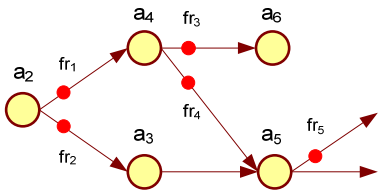
Namen podpoglavja je predstaviti cilje predvidenih faz in potek uvedbe ter uporabe predlaganega pristopa v obravnavanem organizacijskem sistemu.

3.4.3.2 Faza konstruiranja prilagodljivega procesa

V fazi konstruiranja procesa se v okviru izdelave prilagodljive (agilne) metodologije za potrebe obravnavanega organizacijskega sistema (glej razdelek 2.4.5.2.1 in podpoglavje 3.2) izdela prilagodljiv proces razvoja programske opreme, ki ga v disertaciji poimenujemo osnovni proces. Ključnega pomena v tej fazi je izdelava grafa osnovnega procesa *G* in opredelitev množice odločitvenih pravil *FR*, ki se navezujejo na usmerjene povezave v grafu *G* in predstavljajo ti. *pravila za izbor ustreznega procesnega toka* znotraj osnovnega procesa. Izdelan graf osnovnega procesa in odločitvena pravila iz množice *FR* predstavljajo osnovo za nadaljnje prilagajanje procesa. Usmerjene povezave v grafu, katerim je možno pripisati odločitvena pravila iz množice *FR*, so označene z rdečim simbolom in jih prikazuje Slika 20. Odločitvena pravila iz množice FR je možno pripisati:

- usmerjenim povezavam, ki vodijo v vozlišča, katera predstavljajo alternativne aktivnosti - govorimo o modeliranju kontrolnega elementa "odločitev" (glej razdelek 3.5.4.2) ali
- usmerjenim povezavam, ki vodijo v vozlišča, katera predstavljajo množico aktivnosti, ki se lahko izvajajo sočasno - govorimo o modeliranju kontrolnega elementa "sinhronizacija" (glej razdelek 3.5.4.3).

Vsa v omenjeni fazi že opredeljena in nova pravila, ki se opredelijo v naslednjih fazah uvedbe pristopa, tvorijo ti. *bazo pravil*[21] (glej razdelek 3.6.4).



Slika 20: Usmerjene povezave s pripisanimi odločitvenimi pravili

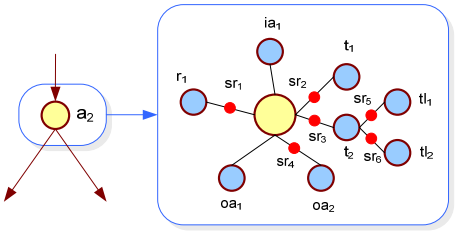
Preslikavo odločitvenih pravil iz množice *FR* v ustrezne usmerjene povezave množice *A*, grafa *G*, bi z uporabo funkcije (4) opisali na naslednji način:

[21] *Zbirka odločitvenih pravil.*

$FR = \{fr_1, fr_2, fr_3, fr_4, fr_5\}$
$AssignRuleA(fr_1) = (a_2, a_4)$,
$AssignRuleA(fr_2) = (a_2, a_3)$,
$AssignRuleA(fr_3) = (a_4, a_6)$ itd.

3.4.3.3 Faza učenja

V fazi učenja se v bazo znanja dodaja nova pravila. Ta pravila tvorijo množico pravil *SR* oziroma množico ti. *strukturnih pravil*. Če se omejimo na množico elementov procesa, ki pripadajo določeni aktivnosti (vloga, izdelek, tehnika, orodje itd.) oziroma na vozlišča v grafu osnovnega procesa, ki predstavljajo aktivnost in njej pripadajoče elemente procesa (vozlišča grafa *G* drugih tipov), potem lahko rečemo, da posamezno pravilo iz množice *SR* opredeljuje obstoj neusmerjene povezave med obravnavano aktivnostjo (vozliščem tipa aktivnost) in elementom procesa (vozlišče kakega drugega tipa), ki se z njo povezujejo neposredno ali posredno, preko drugih elementov (vozlišč). Slika 21 prikazuje neusmerjene povezave s pripisanimi odločitvenimi pravili iz množice *SR*.



Slika 21: Neusmerjene povezave s pripisanimi odločitvenimi pravili

Preslikavo odločitvenih pravil iz množice *SR* v ustrezne neusmerjene povezave množice *E*, grafa *G*, bi z uporabo funkcije (5) opisali na naslednji način:

$FR = \{sr_1, sr_2, sr_3, sr_4, sr_5\}$
$AssignRuleE(sr_1) = (a_2, r_1)$,
$AssignRuleE(sr_2) = (a_2, t_1)$,
$AssignRuleE(sr_3) = (a_2, t_2)$ itd.

V fazi učenja se posamezni neusmerjeni povezavi v grafu osnovnega procesa pripiše strukturno pravilo, ki opredeljuje obstoj povezave med vozliščema, ki sta z njo povezana. Obstoj neke povezave je odvisen od lastnosti projektov, zato pravimo, da odločitveno pravilo iz množice *SR* določi obstoj povezave na podlagi določene podmnožice projektnih karakteristik. Vzemimo primer: če strukturno pravilo potrdi

obstoj neusmerjene povezave med vozliščem tipa aktivnost in vozliščem tipa vloga, potem se bo vozlišču pripadajoča vloga iz razširjenega diagrama aktivnosti nahajala v prilagojeni različici procesa. V nasprotnem primeru (obstoj povezave glede na karakteristike projekt ni mogoč) se vozlišče izloči.

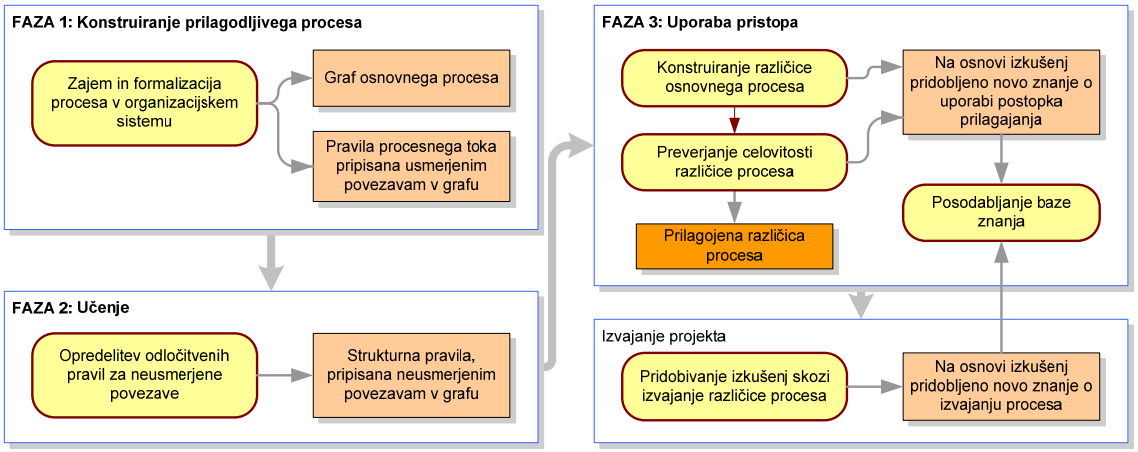
V fazo učenja se je možno kasneje tudi vrniti. Do tega pride v primeru, ko je kateri izmed povezav v grafu *G* potrebno dodati novo strukturno pravilo, ali ažurirati že obstoječe. Ažuriranje množice strukturnih pravil *SR* se izvede na temelju izkušenj, ki se pridobivajo skozi iskanje izpeljanih dejstev (glej razdelek 3.6.6.3). Tista izpeljana dejstva, ki se skozi uporabo pristopa (faza 3) pojavljajo najpogosteje je smiselno opredeliti kot nova strukturna pravila in jih vključiti v graf, tako da se jih pripiše določenim neusmerjenim povezavam v njem.

3.4.3.4 Faza uporabe pristopa

V okviru faze uporabe predlaganega pristopa se lahko prične dejansko izvajanje postopka prilagajanja osnovnega procesa potrebam projektov, ki se izvajajo v organizacijskem sistemu. Za vsako različico procesa, ki se izdela, je potrebno na koncu postopka preveriti še njeno celovitost oziroma popolnost. Celovitost različice se preveri z uporabo ti. pravil celovitosti, ki tvorijo množico pravil *CR*. V to skupino spadajo pravila, ki jih opredeli metodolog na temelju metamodela procesa in zagotavljajo popolnost izdelane različice procesa in s tem tudi njeno minimalnost. Brez uporabe teh pravil, bi lahko v postopku prilagajanja izključno na osnovi pravil iz množice *SR*, prišli do zaključka, da določena vloga ni potrebna za izvajanje določene aktivnosti. V tem primeru bi ta aktivnost ostala brez opredeljene vloge. Metodolog z opredelitvijo pravila npr. "vsako aktivnost izvaja vsaj ena vloga" zagotovi, da bo za vsako aktivnost sigurno določena tudi ustrezna vloga.

Poleg same uporabe pristopa k prilagajanju, se v tej fazi poskrbi še za posodabljanje baze znanja, za kar je odgovoren metodolog. Ta na podlagi izkušenj, pridobljenih skozi uporabo prilagojene različice procesa na konkretnem projektu in na podlagi izkušenj z uporabo postopka prilagajanja, v bazo pravil po potrebi dodaja nova pravila ter spreminja ali izloča obstoječa pravila.

Predvideno uvedbo predlaganega pristopa za prilagajanje procesa konkretnim potrebam projektov v določen organizacijski sistem prikazuje Slika 22.



Slika 22: Uvedba pristopa za prilagajanje procesa v organizacijski sistem

3.5 Odločitvena pravila

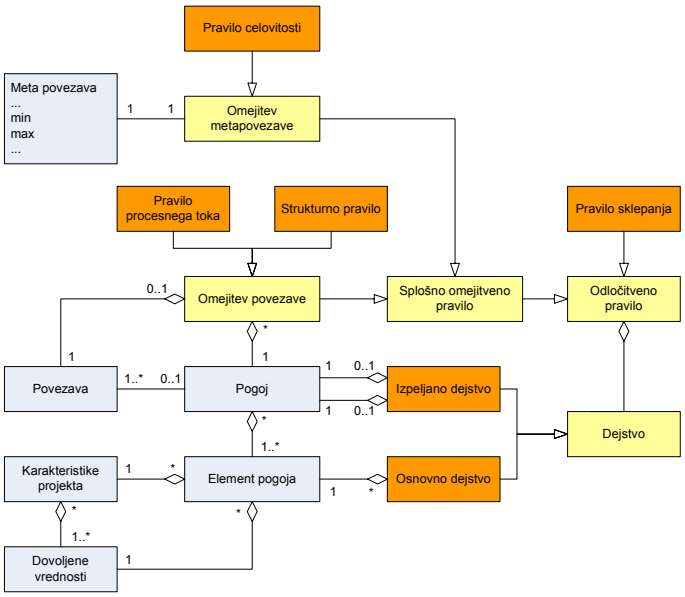
3.5.1 Uvod

V tem podpoglavju se posvetimo opredelitvi odločitvenih pravil. Na začetku predstavimo metamodel, ki opredeljuje tipe odločitvenih pravil in njihove medsebojne povezave. Poleg pravil vsebuje metamodel še dodatne koncepte, ki predstavljajo ali sestavne elemente odločitvenih pravil ali pa ostale bistvene elemente, s katerimi se odločitvena pravila povezujejo. V nadaljevanju podamo zgradbo posameznih tipov odločitvenih pravil, kjer natančno opišemo njihovo strukturo in poudarimo namen posameznih elementov. Pri tem izhajamo iz prispevka o izdelavi metodologij, prilagojenih specifičnim potrebam konkretnih projektov avtorjev Bajca, Vavpotiča in Krisperja [Bajec 2005a]. V zaključku podpoglavja sledi še razdelek o situacijah, v okviru katerega opredelimo načine za modeliranje kontrolnih elementov za usmerjanje procesnega toka in podamo omejitve glede uporabe posameznih tipov odločitvenih pravil.

3.5.2 Metamodel odločitvenih pravil

Odločitvena pravila, njihove komponente in povezave med njimi ter ostale elemente, na katere se odločitvena pravila navezujejo, prikažemo z uporabo metamodela odločitvenih pravil. Posamezni elementi metamodela so prikazani s pomočjo pravokotnikov. Povezave med njimi pa so lahko navadne asociacije, agregacije (črta z rombom) in

generalizacije / specializacije (črta zaključena s puščico). Metamodel odločitvenih pravil prikazuje Slika 23.



Slika 23: Metamodel odločitvenih pravil [Bajec 2005b]

Osrednji element metamodela je *odločitveno pravilo*, ki je predstavljen kot generalizacija *dejstev*, *splošnih omejitvenih pravil* in *pravil sklepanja*.

Splošno omejitveno pravilo predstavlja generalizacijo pravil, ki so opredeljena na temelju *omejitev metapovezav* v metamodelu splošnega procesa in pravil, ki opredeljujejo *omejitve na povezavah* v grafu splošnega procesa. Pravila za opredelitev omejitev na povezavah v grafu, predstavljajo generalizacijo *pravil procesnega toka* in *strukturnih pravil*. Iz omejitev posameznih metapovezav pa izvirajo opredelitve *pravil celovitosti*. Vsaki *metapovezavi* pripada določena omejitev metapovezave, ki je opredeljena na podlagi minimalne in maksimalne številnosti te povezave.

Dejstvo je predstavljeno kot generalizacija *osnovnega dejstva* in *izpeljanega dejstva*. Osnovno dejstvo je sestavljeno iz enega *elementa pogoja*, ki ga sestavlja natančno ena *karakteristika projekta*. Za vsako karakteristiko projekta je opredeljena vsaj ena *dovoljena vrednost* (te vrednosti sestavljajo zalogo vrednosti karakteristike D_i) in vsaka dovoljena vrednost se lahko nahaja v več elementih pogoja. Element pogoja lahko pripada več *pogojem*.

Isti pogoj se lahko uporabi v okviru več pravil za opredeljevanje omejitev povezave. Vsako pravilo za opredeljevanje omejitev povezav je pripisano natanko eni povezavi,

medtem ko ni nujno, da je vsaki povezavi pripisano odločitveno pravilo. Velja tudi, da se isti pogoj nanaša na vsaj eno povezavo, medtem ko ni nujno, da se na določeno povezavo nanaša kakšen od pogojev.

Izpeljana dejstva dobimo iz osnovnih dejstev. Pri tem ima bistveno vlogo mehanizem sklepanja, katerega delovanje temelji na pravilih sklepanja in bo predstavljeno v nadaljevanju disertacije.

V okviru obravnave zgradbe odločitvenih pravil (glej razdelek 3.5.3) bomo podrobneje opredelili bistvene koncepte predstavljene v metamodelu ter natančno opredelili njihovo zgradbo. Bistveni koncepti metamodela, predstavljeni v nadaljevanju so naslednji:

- pravilo procesnega toka,
- strukturno pravilo,
- osnovno dejstvo,
- izpeljano dejstvo,
- pravilo sklepanja in
- pravilo celovitosti.

3.5.3 Zgradba odločitvenih pravil

3.5.3.1 Uvod

Ker je natančno opredelitev ključnih konceptov metamodela (glej razdelek 3.5.2) lažje podati skozi opis njihove strukture, smo se odločili, da jih bomo podali v okviru zgradbe odločitvenih pravil. V nadaljevanju se bomo zato najprej posvetili zgradbi pravil procesnega toka in zgradbi strukturnih pravil. Za ta pravila je značilno, da se razlikujejo samo v tipu povezav katerim se prirejajo, zaradi česar imajo identično zgradbo. V nadaljevanju sledi podrobna opredelitev osnovnih in izpeljanih dejstev, kot elementov, na temelju katerih deluje več tipov odločitvenih pravil, med drugim tudi pravila sklepanja, ki jih predstavimo za tem. V zaključku razdelka podamo še natančno opredelitev pravil celovitosti, ki se uporabljajo za preverjanje celovitosti prilagojene različice procesa.

3.5.3.2 Zgradba strukturnih pravil in pravil procesnega toka

Pravila procesnega toka in strukturna pravila imajo naslednjo skupno lastnost: navezujejo se na določeno povezavo v grafu splošnega procesa *G*. Zaradi tega imajo pravila iz obeh skupin enako zgradbo, ki ima naslednjo obliko:

IF *element procesa x* **AND** *pogoj* **THEN** *element procesa y*.

Odločitveno pravilo ima obliko *if-then* stavka. Premisa[22] opredeljuje pogoj, ki mora biti izpolnjen, če hočemo, da se bo akcija, opredeljena v sklepu[23], izvedla. V našem primeru predstavlja pogojni del stavka ti. *omejitev*, ki opredeljuje obstoj povezave med dvema vozliščema grafa. Prvo od omenjenih vozlišč predstavlja element procesa x, drugo pa element procesa y. Pogojni del stavka (premisa) je opredeljen s pomočjo dveh komponent, in sicer: *element procesa x* in *pogoj*, ki ju povezuje logični operator konjunkcije. Posledični del stavka pa opredeljuje komponenta: *element procesa y*. Pomen posameznih komponent, ki sestavljajo odločitveno pravilo, je naslednji:

Element procesa x: Komponenta pravila predstavlja začetno vozlišče povezave v grafu *G*, na katero se odločitveno pravilo nanaša. V primeru pravil procesnega toka, to vozlišče vedno predstavlja neko aktivnost osnovnega procesa. V primeru strukturnih pravil, pa to vozlišče lahko predstavlja aktivnost ali kakšen drug tip elementa procesa (katerikoli tip vozlišča).

Element procesa y: Komponenta pravila predstavlja končno vozlišče povezave v grafu *G*, na katero se odločitveno pravilo nanaša. V primeru pravil procesnega toka, to vozlišče vedno predstavlja neko aktivnost osnovnega procesa. V primeru strukturnih pravil, pa to vozlišče predstavlja lahko samo nek drug tip elementa procesa. Če je pogojni del odločitvenega pravila (premisa) resničen, se vozlišče grafa, ki predstavlja *element procesa y*, doda v graf, ki predstavlja prilagojeno različico procesa.

Pogoj: Komponenta pravila je sestavljena iz več elementov pogoja, ki so med seboj povezani z logičnimi operatorji konjunkcije, disjunkcije in negacije. Število elementov pogoja je odvisno od števila karakteristik projekta, ki vplivajo na vključitev elementa procesa v prilagojeno različico procesa, ki je predstavljen s končnim vozliščem povezave. Struktura komponente *pogoj* ima naslednjo obliko:

EP 1 **LO** *EP 2* **LO**...**LO** *EP n*,

[22] *Pogojni del if-then stavka.*
[23] *Posledični del if-then stavka.*

kjer so *EP1* do *EPn* okrajšave za elemente pogoja, **LO** pa oznake logičnih operatorjev konjunkcije, disjunkcije ali negacije.

Element pogoja, ki predstavlja sestavni del pogoja v odločitvenem pravilu, ima pri strukturnih pravilih naslednjo strukturo:

KP x **PO** *vrednost x*,

kjer je *KPx* okrajšava za določeno *karakteristiko projekta*, **PO** pa eden od naslednjih primerjalnih operatorjev: =, <>, >, <. V okviru pravil procesnega toka pa ima lahko element pogoja, poleg prej navedene, še naslednjo strukturo:

ATR x **PO** *vrednost x*,

kjer je *ATR x* okrajšava za določen *atribut*, ki ima vpliv na izbiro procesnega toka, **PO** pa eden od naslednjih primerjalnih operatorjev: =, <>, >, <.

Komponenta *vrednost x* predstavlja vrednost, ki jo mora zavzeti *karakteristika projekta x* (*KP x*) oziroma *atribut x* (*ATR x*), da je element pogoja resničen. Dovoljene vrednosti, ki jih lahko zavzame *vrednost x* so opredeljene z zalogo vrednosti karakteristike *KP x*. Vrednosti atributov, ki vplivajo na izbiro procesnega toka, pa se opredelijo glede na potrebe organizacijskega sistema.

3.5.3.2.1 Pravilo procesnega toka

Pravilo procesnega toka predstavlja specializacijo pravila za opredelitev omejitve, ki je pripisana usmerjeni povezavi v grafu osnovnega procesa. Ta pravila tvorijo množico pravil *FR* (glej razdelek 3.4.2.3), na podlagi katerih se v postopku prilagajanja procesa izbere ustrezen delovni tok znotraj osnovnega procesa.

Vsako pravilo procesnega toka se v fazi konstruiranja prilagodljivega procesa (glej razdelek 3.4.3.2) pripiše določeni usmerjeni povezavi s pomočjo funkcije *AssignRuleA*. Uvedemo naslednjo omejitev:

Pravila procesnega toka je možno pripisati samo tistim izhodnim usmerjenim povezavam, ki tvorijo množico vsaj dveh izhodnih povezav iz določenega vozlišča, tipa aktivnost. Pri tem omenjena množica simbolizira odločitev med možnimi tokovi procesa ali pa vejitev procesnega toka v več tokov, ki se lahko odvijajo vzporedno. Če izhodna povezava vozlišča vodi v neko drugo vozlišče z več vhodnimi povezavami, se ji lahko pripiše pravilo procesnega toka (glej razdelek 3.5.4.5).

Z navedeno omejitvijo smo preprečili možnost dodelitve pravila procesnega toka eni sami povezavi, ki poteka med dvema vozliščema tipa aktivnost. Na ta način smo odločanje o vključevanju aktivnosti v prilagojeno različico procesa izločili iz domene pravil procesnega toka in ga obravnavamo izključno v okviru pravil celovitosti (glej razdelek 3.5.3.6). Natančna opredelitev možne uporabe pravil procesnega toka in omejitve, ki jih je pri tem potrebno upoštevati, bodo opredeljene v razdelku o situacijah (glej razdelek 3.5.4).

3.5.3.2.2 Strukturno pravilo

Strukturno pravilo predstavlja specializacijo pravila za opredelitev omejitve, ki je pripisana neusmerjeni povezavi v grafu osnovnega procesa *G*. Razlika je torej v tipu povezave, kateri se pravilo lahko priredi. Ta pravila tvorijo množico pravil *SR* (glej razdelek 3.4.2.3), na podlagi katerih se v postopku prilagajanja procesa izbere preostale elemente procesa (razen aktivnosti), ki naj bi se glede na karakteristike projekta nahajali v prilagojeni različici procesa.

Vsako strukturno pravilo se v okviru faze učenja (glej razdelek 3.4.3.3) pripiše določeni neusmerjeni povezavi s pomočjo funkcije *AssignRuleE*. Strukturno pravilo opredeljuje obstoj tiste povezave, kateri je pripisano in velja, da jih je možno opredeliti za vse neusmerjene povezave v grafu osnovnega procesa. Tiste neusmerjene povezave, na katere se ne nanaša nobeno strukturno pravilo se v postopku prilagajanja ne more izločiti, kar ima za posledico dvoje:

- če taka povezava povezuje vozlišče tipa aktivnost z vozliščem nekega druga tipa, potem slednje vozlišče ne more biti izločeno iz grafa *G*, kar pomeni, da se mora temu vozlišču ustrezen element procesa nahajati v vseh prilagojenih različicah procesa.
- če povezava povezuje dva vozlišča, katerih tip ni aktivnost, potem vozlišče, ki je bolj oddaljeno[24] od vozlišča tipa aktivnost, ne more biti izločeno iz grafa *G*. To pa pomeni, da se mora temu vozlišču ustrezen element procesa nahajati v vseh različicah procesa.

Natančna opredelitev možne uporabe strukturnih pravil in možne omejitve, ki jih je pri tem potrebno upoštevati, bodo opredeljene v razdelku o situacijah (glej razdelek 3.5.4).

[24] *Število skokov od vozlišča tipa aktivnost do obravnavanega vozlišča v grafu osnovnega procesa.*

3.5.3.3 Osnovno dejstvo

Zgradba osnovnega dejstva je opredeljena z obliko komponente pravila *element pogoja*. Osnovno dejstvo je opredeljeno s konkretno vrednostjo določene karakteristike projekta. Ker se vrednosti karakteristik za posamezne projekte med seboj razlikujejo, velja, da vsak projekt opredeljuje svojo množico osnovnih dejstev *MOD*. Množica osnovnih dejstev predstavlja vhodne podatke o projektu, za katerega se izvaja prilagajanje procesa. Na podlagi ujemanja (ang. pattern matching) osnovnih dejstev z elementi pogoja v premisi odločitvenega pravila, se za vsako pravilo ugotovi, ali je vrednost pogoja resnična ali neresnična. Vrednost pogoja v pravilu je torej opredeljena, ko se izvede preverjanje ujemanja med elementi pogoja in osnovnimi dejstvi iz množice *MOD*.

Osnovno dejstvo ima naslednjo strukturo:

KP x **PO** *vrednost x*,

kjer je *KPx* okrajšava za določeno *karakteristiko projekta x*, **PO** pa eden od naslednjih primerjalnih operatorjev: =, <>, >, <. *Vrednost x* predstavlja dejansko vrednost karakteristike za obravnavani projekt.

Ujemanje z osnovnimi dejstvi poteka v okviru treh skupin pravil. Rekli bi tudi lahko, da osnovna dejstva predstavljajo vhodne podatke za tri tipe odločitvenih pravil, in sicer:

- pravila procesnega toka,
- strukturna pravila in
- pravila sklepanja.

3.5.3.4 Izpeljano dejstvo

Glavna značilnost izpeljanih dejstev je, da jih pridobimo iz množice osnovnih dejstev *MOD* (glej razdelek 3.5.3.3), s pomočjo pravil sklepanja. Izpeljano dejstvo je po strukturi podobno komponenti pravila *pogoj*, ki smo jo obravnavali v okviru strukturnih pravil (glej razdelek 3.5.3.2):

EID 1 **LO** *EID 2* **LO**...**LO** *EID n*,

EID1 do *EIDn* predstavljajo okrajšave za elemente izpeljanega dejstva, **LO** pa logični operator konjunkcije, disjunkcije ali negacije. Struktura elementa izpeljanega dejstva je enaka strukturi elementa pogoja, ki smo ga opredelili v razdelku o strukturnih pravilih

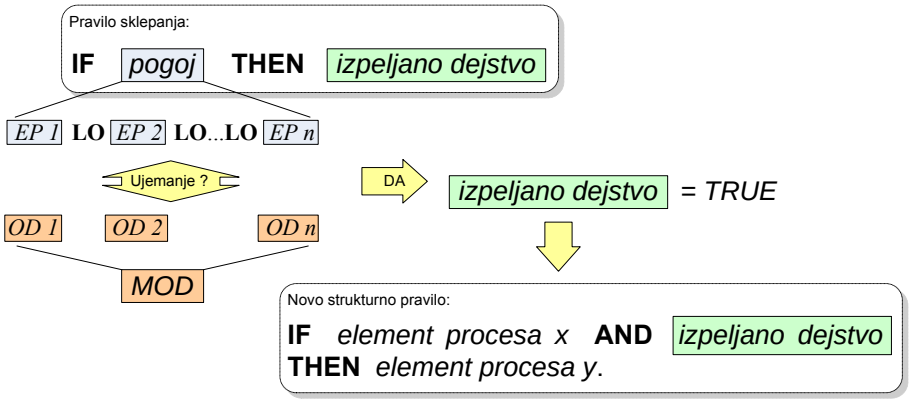
(glej razdelek 3.5.3.2). Izpeljano dejstvo lahko torej vsebuje enega ali več elementov. Več o pomenu in vlogi izpeljanih dejstev bomo obravnavali v okviru opredelitve pravil sklepanja, ki sledi v nadaljevanju disertacije.

3.5.3.5 Pravilo sklepanja

V okviru predlaganega pristopa k prilagajanju procesa predvidevamo tudi uporabo sistema za izvajanje pravil. Delovanje sistema za izvajanje pravil temelji na uporabi mehanizma sklepanja. Omenjeni mehanizem na podlagi premis v pravilih sklepanja izpelje določene zaključke, ki so predstavljeni z novimi (izpeljanimi) dejstvi, ki se dodajo v delovni pomnilnik sistema za izvajanje pravil (glej razdelek 3.6.7).

V našem primeru predstavlja prej omenjeni zaključek *izpeljano dejstvo*, ki ga dobimo na podlagi množice osnovnih dejstev *MOD* in *pravil sklepanja*. Pravila sklepanja lahko uporabimo za opredelitev povezav med posameznimi karakteristikami projekta oziroma njihovih medsebojnih odvisnosti, kar bomo prikazali v okviru razdelka o opisu medsebojnih odvisnosti karakteristik projekta s pravili sklepanja (glej razdelek 3.6.7.6).

Na osnovi izpeljanih dejstev je možno opredeliti tudi dodatne pogoje, ki se uporabijo za opredelitev premis dodatnih strukturnih pravil. Lahko tudi rečemo, da nam pravila sklepanja, v sklopu faze učenja (glej razdelek 3.4.3.3), omogočajo opredeliti dodatna odločitvena pravila, ki se lahko pripišejo posameznim neusmerjenim povezavam v grafu splošnega procesa. To prikazuje Slika 24:



Slika 24: Opredelitev dodatnega odločitvenega pravila

Pravilo sklepanja ima naslednjo strukturo:

IF *pogoj* **THEN** *izpeljano dejstvo*.

Pogojni del stavka (premiso) opredeljuje komponenta *pogoj*, ki mora biti resničen, če hočemo da bo *izpeljano dejstvo* resnično. Struktura komponente *pogoj* je enaka, kot smo jo opredelili že v okviru opredelitve strukturnih pravil (glej razdelek 3.5.3.2). Vsak pogoj je lahko sestavljen iz enega ali več elementov pogoja. Podobno velja za komponento *izpeljano dejstvo*, ki pa smo jo podrobno opredelili že v predhodnem razdelku disertacije.

Izpeljano dejstvo bo resnično takrat, ko bo pogoj, naveden v premisi pravila sklepanja, resničen. To se bo zgodilo takrat, ko se bodo vsi elementi pogoja v pravilu sklepanja ujemali z dejanskimi vrednostmi karakteristik, ki so zajete v obliki osnovnih dejstev množice *MOD*.

3.5.3.6 Pravilo celovitosti

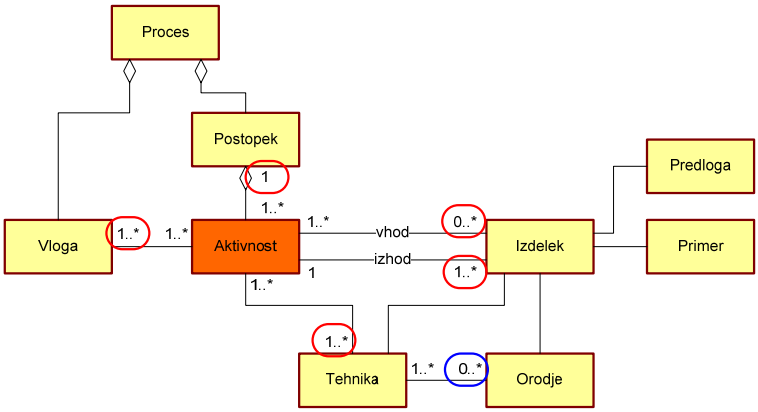
V okviru metamodela prilagodljive metodologije so predstavljeni tudi metaelementi splošnega procesa organizacijskega sistema in metapovezave med njimi. Metapovezavi se lahko pripiše številnost, ki opredeljuje najmanjše in največje število primerkov določenega metaelementa, s katerimi se primerek opazovanega metaelementa lahko povezuje. Vidimo, da je možno že na nivoju samega metamodela opredeliti določene omejitve, ki zagotavljajo, da bo na metamodelu temelječ primerek splošnega procesa v organizacijskem sistemu popoln oziroma celovit.

Če hočemo zagotoviti, da bodo tudi prilagojene različice osnovnega procesa zadoščale pogojem celovitosti, moramo v postopku prilagajanja upoštevati enake omejitve kot pri konstruiranju splošnega procesa. Zaradi tega lahko pravila celovitosti opredelimo na temelju prej omenjene številnosti metapovezav, s čimer se zagotovi, da bo posamična različica procesa popolna in hkrati minimalna. Pravila celovitosti tvorijo množico pravil *CR* (glej razdelek 3.4.3.4).

Slika 25 prikazuje možni izsek metamodela metodologije, ki prikazuje nekatere metaelemente procesa in metapovezave med njimi. Vsaki metapovezavi sta pripisani dve številnosti. V okviru predlaganega pristopa, nas za opredelitev pravil celovitosti zanimajo izključno naslednje številnosti:

- številnosti, ki opredeljujejo, s koliko primerki nekega metaelementa drugega tipa se lahko povezuje metalement tipa aktivnost (glej Slika 25, številnosti obkrožene z rdečo) in
- številnosti, ki opredeljujejo, s koliko primerki nekega metaelementa Y se lahko obravnavan primerek metaelementa X povezuje. Pri tem je metaelement X od

metaelementa aktivnost oddaljen manj, kot metalement Y (potrebno število povezav do metaelementa). Hkrati pa je potrebno upoštevati, da se oddaljenost metaelementa Y od metalementa aktivnost določa preko metaelementa X (glej Slika 25, številnosti obkrožene z modro).



Slika 25: Primer metaelementov procesa in metapovezav med njimi

Za pravila celovitosti smo predvideli naslednjo strukturo:

IF *metaelement procesa x* **AND** *metaelement procesa y* **AND** *pogoj* **THEN** *akcija*.

Pogoj v premisi je sestavljen iz treh komponent. Prvi dve komponenti predstavljata metaelementa, ki natančno opredeljujeta metapovezavo in s tem tudi številnost, ki se bo upoštevala pri preverjanju celovitosti prilagojene različice procesa. Tretja komponenta predstavlja logični *pogoj*. Če je pogoj v premisi pravila izpolnjen, potem se izvede *akcija*, ki jo opredeljuje posledični del pravila.

Ker so pravila celovitosti opredeljena na podlagi metapovezav in njihovih številnosti, je za zagotavljanje celovitosti prilagojene različice procesa potrebno različico procesa preveriti glede na vsa tista pravila, ki imajo v komponenti *metaelement procesa x* navedene enake tipe elementov kot se pojavljajo v grafu prilagojene različice. Preverjanje celovitosti poteka na naslednji način: Za vsak tip vozlišča v grafu prilagojene različice procesa je potrebno ugotoviti, katera pravila celovitosti se navezujejo nanj, oziroma katera pravila celovitosti imajo v komponenti *metaelement procesa x* naveden element enakega tipa. Npr.: za vsa vozlišča tipa aktivnost se poišče pravila celovitosti, ki imajo v komponenti *metaelement procesa x* navedeno vrednost aktivnost. Omenjena pravila tvorijo množico pravil P. Vsako pravilo iz množice P opredeljuje, s katerim tipom vozlišč se obravnavan tip vozlišč lahko povezuje (komponenta *metaelement procesa y*). V grafu prilagojene različice procesa G' se za

vsako vozlišče obravnavanega tipa ugotovi, kolikšno je število povezav z vozlišči posameznega tipa (možni tipi vozlišč so opredeljeni s pravili iz množice *P*). Če se to število nahaja okviru, kot ga opredeljuje komponenta *pogoj*, je celovitost zagotovljena, v nasprotnem primeru pa se izvrši akcija, ki jo opredeljuje posledični del odločitvenega pravila.

Pomen posameznih komponent, ki sestavljajo pravilo celovitosti, je naslednji:

Metaelement procesa x: Komponenta opredeljuje tip elementov procesa oziroma vozlišč v grafu prilagojene različice procesa (aktivnost, vloga, izdelek, tehnika itd.), na katere se pravilo celovitosti nanaša. Komponenta običajno določa metaelement tipa aktivnost. V primeru, da komponenta določa metaelement drugega tipa, mora biti metaelement, naveden v komponenti *metaelement procesa y*, za eno povezavo bolj oddaljen od metaelementa aktivnost kot pa obravnavan metaelement.

Metaelement procesa y: Komponenta opredeljuje tip elementov procesa oziroma vozlišč v grafu prilagojene različice procesa, ki se povezujejo s tipom vozlišča opredeljenem v komponenti pravila *metaelement procesa x*.

Pogoj: Komponenta pravila opredeljuje najmanjše in največje število elementov tipa *metaelement procesa y*, s katerimi se lahko povezuje tip elementa, ki je opredeljen v okviru komponente *metaelement procesa x*. Struktura komponente pogoj ima tako naslednjo obliko:

NOT (**min** =< x =< **max**),

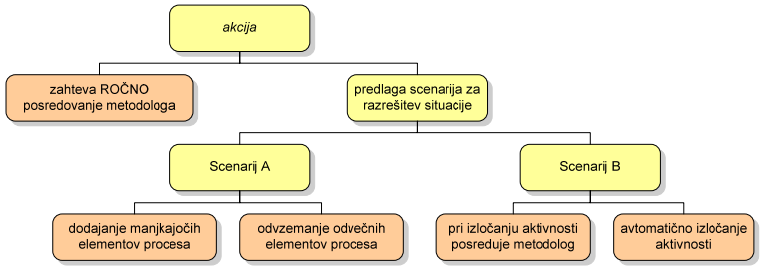
kjer *x* predstavlja dejansko število povezav, ki potekajo med vozliščem tipa *metaelement procesa x* in vozlišči tipa *metaelement procesa y*. Vrednosti x je potrebno pridobiti za vsako vozlišče grafa prilagojene različice procesa, in sicer za vsako odločitveno pravilo iz množice *P*, katera se nanaša na tip obravnavanega vozlišča. NOT predstavlja logični operator negacije.

Akcija: Komponenta opredeljuje, kaj se mora zgoditi, če pogoj v premisi pravila ni resničen. V tem primeru govorimo o kršenju celovitosti različice procesa. V okviru predlaganega pristopa smo predvideli več možnih načinov za razreševanje problema celovitosti. Način za razreševanje konfliktne situacije se navede v okviru kompenente *akcija*. Tako lahko akcija:

- *zahteva posredovanje metodologa*: V tem primeru *akcija* prekine izvajanje postopka prilagajanja in zahteva posredovanje metodologa. Ta razreši problem celovitosti tako, da doda ali odvzame ustrezen element procesa iz prilagojene različice procesa.

- *predlaga scenarij za možno razrešitev konfliktne situacije*: V tem primeru smo opredlili dva možna scenarija:
  - *scenarij A*: Predvideva dodajanje manjkajočih ali odvzemanje odvečnih elementov procesa iz prilagojene različice, glede na najmanjšo ali največjo vrednost, ki je opredeljena v komponenti *pogoj*.
  - *scenarij B*: Scenarij se nanaša na problem izhodnih izdelkov določene aktivnosti. V splošnem namreč velja, da je rezultat vsake aktivnosti nek izdelek. Če postopek prilagajanja za določeno aktivnost ne opredeli nobenega izhodnega izdelka, se je potrebno vprašati, ali je vključitev aktivnosti v različico procesa sploh smiselna. Scenarij B predlaga dve možni rešitvi, in sicer:
    - posredovanje metodologa, ki se mora odločiti o izločitvi omenjene aktivnosti iz različice procesa in
    - avtomatično izločanje aktivnosti.

Možne akcije, ki jih pravilo celovitosti lahko predlaga v okviru komponente *akcija*, so prikazane na sliki (glej Slika 26):



Slika 26: Načini razreševanja problema celovitosti

3.5.3.7 Sklep

V razdelku o zgradbi pravil smo opredelili vse vrste odločitvenih pravil, ki se uporabljajo v okviru predlaganega pristopa k prilagajanju procesa in podrobno opredelili strukturo in pomen njihovih sestavnih delov. Zaradi večje preglednosti nad njimi, jih na tem mestu povzemamo v strjeni obliki. Zaradi celovitosti dodajamo tudi povzetek osnovnih in izpeljanih dejstev.

pravilo procesnega toka	**IF** element procesa x **AND** pogoj **THEN** element procesa y
strukturno pravilo	**IF** element procesa x **AND** pogoj **THEN** element procesa y
pravilo sklepanja	**IF** pogoj **THEN** izpeljano dejstvo

pravilo celovitosti	**IF** metaelement procesa x **AND** metaelement procesa y **AND** pogoj **THEN** akcija
osnovno dejstvo	KP x **PO** vrednost x
izpeljano dejstvo	*EID 1* **LO** *EID 2* **LO**...**LO** *EID n*

3.5.4 Odločitvena pravila v okviru posameznih situacij

3.5.4.1 Uvod

Mešan graf *G* predstavlja formalen opis splošnega procesa v organizacijskem sistemu na višjem nivoju abstrakcije. Opredelili smo ga kot skupek vozlišč različnih tipov (elementov procesa) ter usmerjenih in neusmerjenih povezav, v katere se preslika razširjen graf aktivnosti (glej razdelek 3.4.2.1.1), s katerim zajamemo osnovni proces v organizacijskem sistemu. Ker graf *G* ne vsebuje posebnih gradnikov za prikaz odločitev, sinhronizacijo in združitev, je njihov pomen potrebno modelirati (predstaviti) na drugačen način. Glede na to bomo v nadaljevanju opredelili več ti. *situacij*. Situacija modelira določen kontrolni element (odločitev, sinhronizacija, združitev) ali kombinacijo njih, iz razširjenega diagrama aktivnosti, z opredelitvijo omejitve glede pogojev v odločitvenih pravilih, katera so pripisana izhodnim povezavam obravnavanega vozlišča v situaciji.

V nadaljevanju bomo predstavili kontrolne elemente in njihove možne osnovne kombinacije, ki se lahko pojavijo v okviru razširjenega diagrama aktivnosti ter prikazali, kako jih modeliramo s pomočjo situacij na nivoju predstavitve z grafom. Zanima nas torej, katera odločitvena pravila je potrebno v okviru situacije pripisati izhodnim povezavam vozlišča in kakšne omejitve veljajo za pogoje v teh pravilih. Na koncu razdelka bo prikazan tudi zahtevnejši primer kombinacije več kontrolnih elementov in ustrezna situacija v grafu.

3.5.4.2 Situacija vzajemnega izključevanja

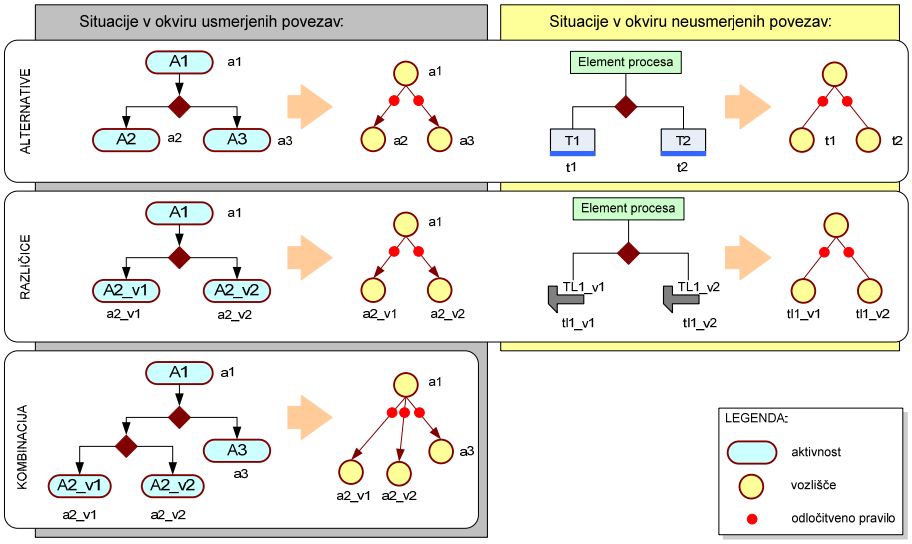
Situacija vzajemnega izključevanja se na nivoju prikaza z grafom uporablja za modeliranje odločitvenega elementa ali za kombinacijo le teh. Nastopita lahko dve skupini primerov, in sicer:

- V okviru usmerjenih povezav grafa:
  1. odločitveni element predstavlja možnost izbire med alternativnimi aktivnostmi procesa,
  2. odločitveni element predstavlja možnost izbire med več različicami iste aktivnosti in
  3. kombinacija prvih dveh primerov.
- V okviru neusmerjenih povezav grafa:
  1. odločitveni element predstavlja možnost izbire med več alternativnimi elementi procesa in
  2. odločitveni element predstavlja možnost izbire med več različicami elementa procesa.

V okviru prve skupine primerov je situacija v grafu opredeljena z več izhodnimi povezavami iz vozlišča tipa aktivnost, katerim se lahko pripiše le pravila procesnega toka. Za ta pravila mora veljati naslednja omejitev: *pogoji v premisah v omenjenih pravilih se morajo medsebojno izključevati*.

V okviru druge skupine primerov je situacija v grafu opredeljena z več izhodnimi povezavami iz vozlišča kateregakoli tipa, katerim se lahko pripiše le strukturna pravila. Za ta pravila mora veljati enaka omejitev, kot v primeru prej omenjenih pravil procesnega toka, torej: *pogoji v premisah v omenjenih pravilih se morajo medsebojno izključevati*.

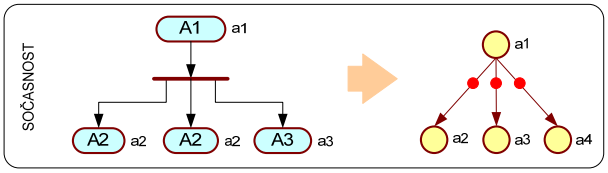
Možne situacije vzajemnega izključevanja so prikazane na sliki (glej Slika 27):



Slika 27: Situacije vzajemnega izključevanja

3.5.4.3 Situacija sočasnega izvajanja

Situacija sočasnega izvajanja se na nivoju prikaza z grafom uporablja za modeliranje sinhronizacijskega elementa - vejitve. Element sinhronizacije - vejitev modeliramo s pomočjo situacije v grafu, ki je opredeljena z več izhodnimi povezavami iz vozlišča tipa aktivnost, katerim se lahko pripiše le pravila procesnega toka. Za ta pravila mora veljati naslednja omejitev: *pogoji v premisah omenjenih pravilih se ne smejo medsebojno izključevati, če hočemo, da se bodo vzporedne aktivnosti izvajale sočasno*. Iz množice danih aktivnosti, ki se lahko izvajajo vzporedno, se v postopku prilagajanj izbere podmnožica aktivnosti, ki ustreza potrebam projekta in se bodo izvajale sočasno. Situacijo sočasnega izvajanja prikazuje Slika 28:



Slika 28: Situacija sočasnega izvajanja

3.5.4.4 Situacija kombinacije

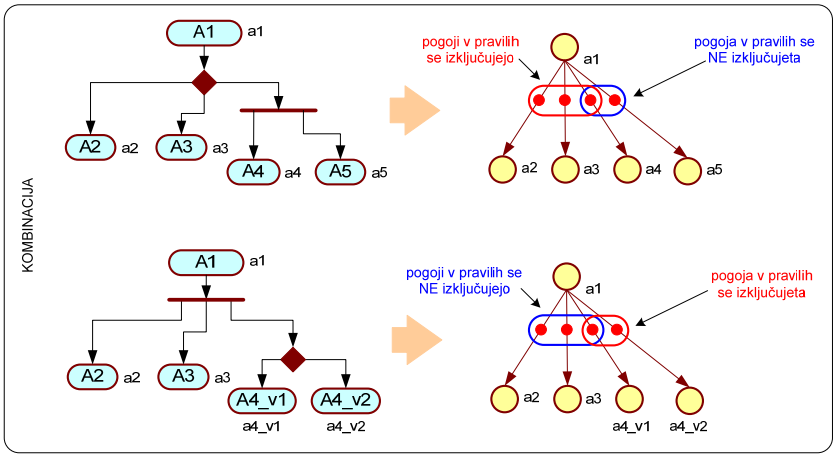
Situacija kombinacije se na nivoju prikaza z grafom uporablja za modeliranje odločitvenega elementa v kombinaciji z elementom sinhronizacije - vejitve. Nastopita lahko dva primera:

- odločitveni element predstavlja možnost izbire med alternativnimi aktivnostmi med katerimi se določene aktivnosti lahko izvajajo tudi sočasno in
- element vejitve predstavlja možnost izbire podmnožice aktivnosti, ki se izvajajo sočasno. Med aktivnostmi so lahko tudi aktivnosti z več različicami.

V okviru prvega primera je situacija v grafu opredeljena z več izhodnimi povezavami iz vozlišča tipa aktivnost, katerim se lahko pripiše samo pravila procesnega toka. Za ta pravila morajo veljati naslednje omejitve:

- *pogoji v premisah pravil, ki pripadajo alternativnim povezavam se medsebojno izključujejo,*
- *pogoji v premisah pravil, ki pripadajo povezavam v sočasne aktivnosti, se med seboj ne smejo izključevati, če hočemo, da se te povezave lahko izvajajo sočasno*.

Obe situaciji kombinacije sta prikazani na sliki (glej Slika 29).



Slika 29: Situaciji kombinacije

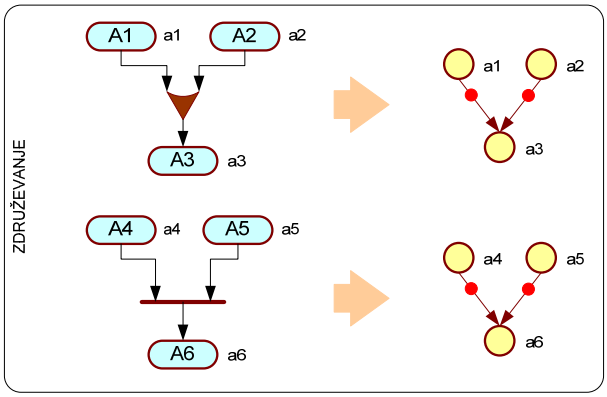
3.5.4.5 Situacija združevanja

Situacija združevanja se na nivoju prikaza z grafom uporablja za modeliranje sinhronizacijskega elementa - "prave" sinhronizacije ali modeliranje kontrolnega elementa združevanja. Nastopita lahko dva primera:

- element združevanja predstavlja združitev dveh tokov, kjer se izhodni tok procesa lahko nadaljuje takoj, ko je ena izmed vhodnih aktivnosti dokončana in
- sinhronizacijski element predstavlja združitev dveh tokov, kjer se izhodni tok procesa nadaljuje šele takrat, ko so vse vhodne aktivnosti v element dokončane.

V okviru prvega in drugega primera je situacija v grafu opredeljena z več vhodnimi povezavami v vozlišče tipa aktivnost, katerim se lahko pripiše samo pravila procesnega toka. Za ta pravila mora veljati naslednja omejitev: *pogoji v premisah pravil morajo biti enaki, če gre za združevanje tokov, ki izvirata v istem vozlišču grafa (pogoj je mišljen v smislu pogoja opredeljenega v razdelku 3.5.3.2). V nasprotnem primeru omejitve glede pogoja ni.*

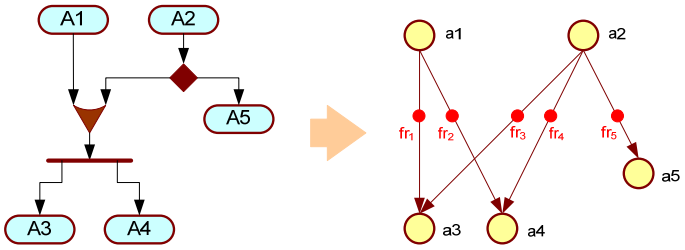
Obe situaciji združevanja sta prikazani na sliki (glej Slika 30).



Slika 30: Situaciji združevanja

3.5.4.6 Zahtevnejši primer in ustrezna situacija v grafu

Primer zahtevnejše kombinacije kontrolnih elementov, ki se lahko pojavi v realnem razširjenem diagramu aktivnosti splošnega procesa, prikazuje Slika 31. Podali bomo omejitve, ki veljajo glede pogojev v pravilih, ki so pripisani povezavam, ki modelirajo ustrezno situacijo v grafu.



Slika 31: Primer situacije v grafu

Situacija v našem primeru je sestavljena iz dveh osnovnih situacij. Prvo situacijo tvorijo vozlišča tipa aktivnost a1, a3, a4 in predstavlja ti. situacijo sočasnega izvajanja (glej razdelek 3.5.4.3), drugo pa vozlišča tipa aktivnost a2, a3, a4, a5 in predstavlja ti. situacijo kombinacije (glej razdelek 3.5.4.4). Skladno s tem veljajo v obeh osnovnih situacijah tudi omejitve glede pogojev v pravilih, kot smo jih opredelili v opisih ustreznih osnovnih situacij.

Velja, da se pogoja v pravilih fr_1 in fr_2 medsebojno ne smeta izključevati (kar seveda tudi ne pomeni, da morata biti enaka). Enako velja tudi za pogoja v pravilih fr_3 in fr_4, dočim mora za pogoj, ki pripada pravilu fr_5 veljati naslednje:

$$p_3 \in fr_3 \wedge p_4 \in fr_4 \wedge p_5 \in fr_5 \rightarrow (p_3 = \neg p_5) \wedge (p_4 = \neg p_5)$$

3.6 Odločitveni model za prilagajanje procesa

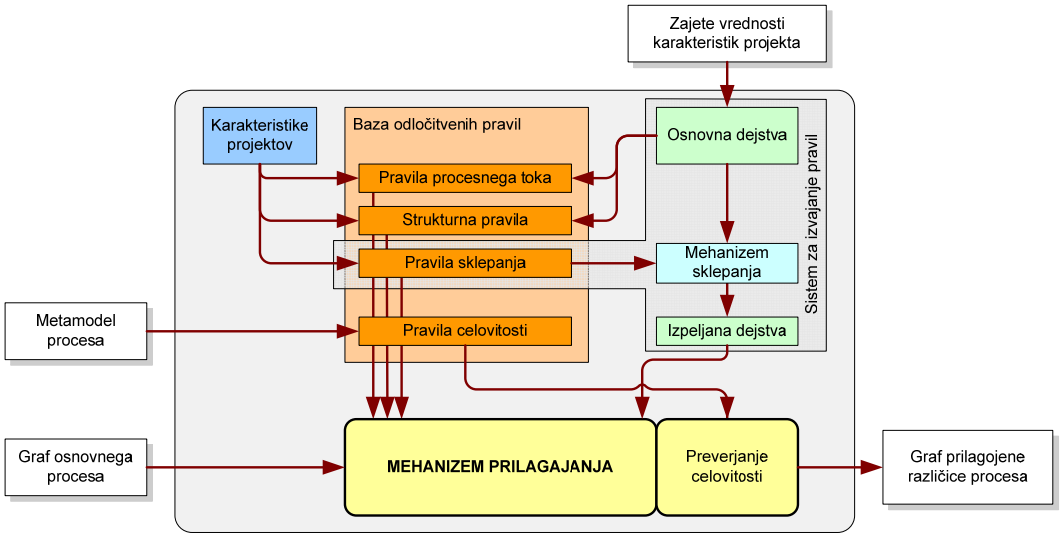
3.6.1 Uvod

V nadaljevanju disertacije je predstavljen koncept odločitvenega modela za prilagajanje procesa potrebam konkretnih projektov. Opredelitev njegove zgradbe in delovanja predstavlja enega ključnih ciljev doktorske disertacije. Na začetku bomo najprej podali shemo konceptualne zgradbe odločitvenega modela, s pomočjo katere bomo opredelili najpomembnejše elemente modela. Ključni element odločitvenega modela predstavlja mehanizem prilagajanja, ki ga bomo opredelili kot rekurzivni postopek za gradnjo prilagojene različice procesa, temelječ na uporabi predhodno opredeljenih odločitvenih pravilih. Nato bomo predstavili še ostale elemente konceptualnega modela in njihove vloge v okviru le tega. Na koncu podpoglavja bomo v okviru predstavitve mehanizma za pojasnjevanje rezultatov odločanja, opredelili še način za zagotavljanje transparentnosti odločitev, ki se sprejmejo v okviru izgradnje prilagojene različice procesa.

3.6.2 Konceptualna zgradba odločitvenega modela

Namen odločitvenega modela za prilagajanje procesa individualnim potrebam projektov je za vsak element procesa, ki je opredeljen z metamodelom osnovne metodologije, vpeljane v organizacijski sistem, podati odgovor, ali bo ta vključen v prilagojeno različico procesa ali ne. Da bi lahko odgovorili na vprašanje, ali je določen element procesa primeren za vključitev v prilagojeno različico procesa, je najprej potrebno odgovoriti na več vprašanj. Kakšne so karakteristike projekta? Kako posamezna karakteristika projekta vpliva na vključevanje posameznih elementov v različice proces? Ali se karakteristika navezuje samo na določene elemente procesa, oziroma ali obstajajo med njimi morebitne odvisnosti? Katere karakteristike vplivajo samo na vključevanje *aktivnosti* v različico procesa? Katere karakteristike projektov vplivajo na vključevanje ostalih elementov v različico procesa?

Našteta vprašanja so nam služila kot izhodišče za zasnovo odločitvenega modela, katerega konceptualno zgradbo prikazuje Slika 32.



Slika 32: Konceptualna zgradba odločitvenega modela

Osrednji element odločitvenega modela predstavlja mehanizem za prilagajanje osnovnega procesa, ki deluje na temelju odločitvenih pravil, opredeljenih v razdelku 3.5. Ostali elementi, ki tvorijo konceptualni model, so še:

- baza odločitvenih pravil,
- množica karakteristik projektov,
- osnovna in izpeljana dejstva ter
- mehanizem sklepanja v sklopu sistema za izvajanje pravil.

Ker ostali elementi odločitvenega modela predstavljajo koncepte, ki so bili predstavljeni že v okviru predhodnih razdelkov, se bomo v nadaljevanju posvetili predvsem opredelitvi mehanizma prilagajanja, ostale elemente pa bomo opredelili le do nivoja, do katerega je to v tem poglavju še smiselno.

3.6.3 Mehanizem prilagajanja

3.6.3.1 Uvod

Mehanizem prilagajanja tvori osrednji element odločitvenega modela. V doktorski disertaciji smo mehanizem prilagajanja opredelili kot rekurzivni postopek, ki v grafu splošnega procesa G poišče podgraf prilagojene različice procesa. Podgraf predstavlja mešan graf in je določen s trojko $G' = (V', E', A')$, katere pomen posamičnih komponent

smo podali že v razdelku 3.4.2.2.1. Gradnjo podgrafa usmerjajo odločitvena pravila. Mehanizem prilagajanja pri tem uporablja tri skupine odločitvenih pravil (glej Slika 32):

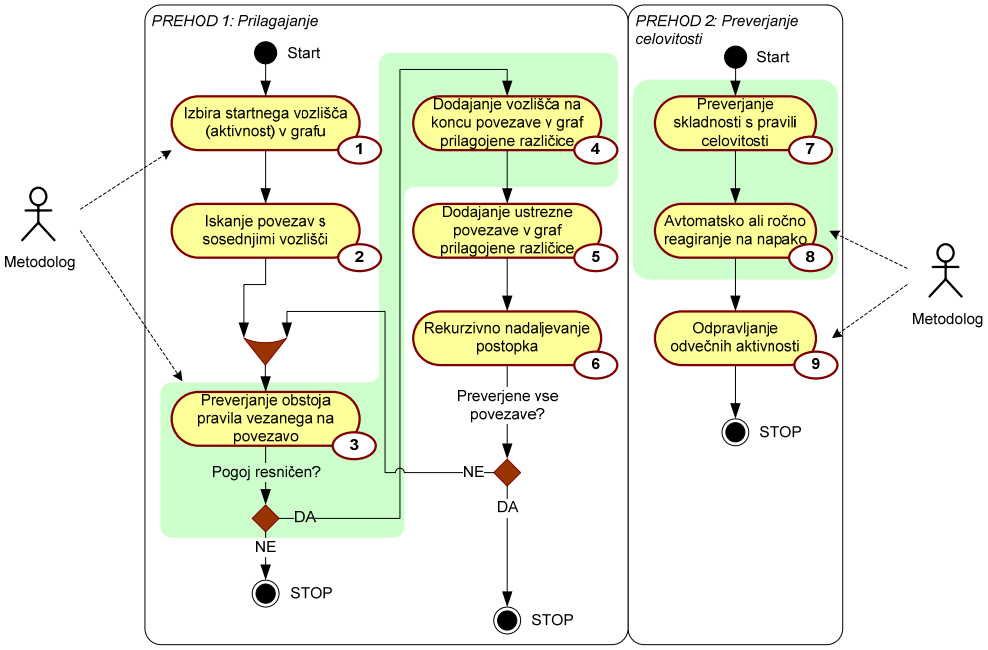
- pravila procesnega toka,
- strukturna pravila in
- pravila sklepanja,

ter izpeljana dejstva, na podlagi katerih se za posamezni projekt lahko opredeli še dodatna skupina strukturnih pravil (glej razdelek 3.5.3.5). Opis naštetih skupin odločitvenih pravil in njihov vpliv na vključevanje vozlišč v podgraf prilagojene različice procesa G', smo podali že podpoglavju 3.5, zato bomo tukaj podali le način njihove uporabe v okviru postopka prilagajanja.

V okviru mehanizma prilagajanja se nahaja tudi element za preverjanje celovitosti. Ta opredeljuje način za preverjanje celovitosti prilagojene različice procesa, ki temelji na ugotavljanju ujemanja števila vozlišč določenega tipa v grafu različice procesa G', z omejitvami, ki jih predpisuje metamodel metodologije. Element v sklopu mehanizma prilagajanja, za preverjanje celovitosti, smo opredelili kot postopek, ki temelji na uporabi odločitvenih pravil celovitosti in tvori sestavni del postopka za gradnjo prilagojenih različic procesa.

3.6.3.2 Postopek izgradnje prilagojene različice procesa

Element odločitvenega modela, mehanizem prilagajanja je opredeljen s postopkom, ki ga prikazuje diagram aktivnosti na sliki (glej Slika 33). Kot bomo predpostavili v nadaljevanju, se lahko postopek prilagajanja v nekaterih primerih tudi ustavi. V takih primerih je potrebno posredovanje ustrezne vloge, kar smo v diagramu prikazali z vlogo metodologa, ki posreduje v aktivnostih, v katerih se lahko postopek ustavi.



Slika 33: Postopek za prilagajanje osnovnega procesa

Postopek predvideva izgradnjo različice procesa v dveh prehodih skozi graf. Pri tem je potrebno opozoriti, da je s prvim prehodom mišljen prehod skozi graf osnovnega procesa *G*, z drugim pa prehod skozi graf prilagojene različice procesa *G'*. Vhod v postopek predstavlja graf osnovnega procesa, ki je opredeljen s trojko *G = (V, E, A)*. V okviru prvega prehoda je potrebno v grafu *G* najprej opredeliti startno vozlišče (tipa aktivnost), ki predstavlja točko v grafu osnovnega procesa, v kateri se bo postopek prilagajanja pričel izvajati (aktivnost 1). Startno vozlišče poda metodolog, tako da opredeli aktivnost iz osnovnega procesa, s katero se mora prilagojena različica procesa pričeti. V naslednji aktivnosti (aktivnost 2) se poišče povezave obravnavanega vozlišča s sosednjimi vozlišči. Te povezave tvorijo množico povezav *L*, v kateri se nahajajo tako usmerjene kot neusmerjene povezave. Sledi zanka, v kateri se za vsako povezavo iz množice *L* ugotovi, ali je nanjo vezano kakšno odločitveno pravilo (aktivnost 3). Kako se odločitvena pravila pripiše posameznim tipom povezav, opredeljuje razdelek 3.4.2.3. Če je obravnavani povezavi pripisano odločitveno pravilo (strukturno pravilo ali pravilo procesnega toka) in je pogoj v njegovi premisi resničen, se vozlišče iz grafa osnovnega procesa, ki se nahaja na koncu omenjene povezave in je hkrati opredeljeno tudi v *then* delu stavka (posledičnem delu) odločitvenega pravila, doda v graf prilagojene različice (aktivnost 4). V okviru četrte aktivnosti lahko nastopita tudi naslednja dva dogodka:

- logične vrednosti pogoja v premisi pravila ni možno izračunati. V tem primeru se postopek ustavi in zahteva posredovanje metodologa. Ta se mora odločiti o vključitvi problematičnega vozlišča v graf (glej tudi razdelek 3.6.3.3),
- če povezavi (usmerjeni ali neusmerjeni) ni pripisano nobeno pravilo, pomeni, da se mora vozlišče na njenem koncu obvezno nahajati v grafu prilagojene različice G'. Govorimo o ti. fiksnem vozlišču. Prav tako se mora v graf G' prenesti obravnavana povezava.

V graf prilagojene različice G' se zatem doda tudi obravnavana povezava (aktivnost 5), kateri je lahko pripisano odločitveno pravilo ali pa ne. V nadaljevanju postopka sledi rekurzivni klic taistega postopka, katerega parameter predstavlja vozlišče, ki je bilo dodano v graf prilagojenega procesa (aktivnost 6). Ko postopek preveri vse povezave, ki vodijo iz trenutnega vozlišča, se konča.

Izhod postopka predstavlja graf prilagojene različice G'. Vozlišča tega grafa predstavljajo elemente procesa, ki tvorijo različico procesa, katera je prilagojena potrebam konkretnega projekta. Vendar pa zanjo velja, da ta še ni nujno celovita. Zaradi nepopolnosti zajetih vrednosti karakteristik se lahko zgodi, da postopek prilagajanja ne najde vseh vozlišč (elementov procesa), ki bi sicer morali biti vsebovani v prilagojeni različici procesa, glede na omejitve opredeljene v metamodelu metodologije organizacijskega sistema. Problem smo rešili z opredelitvijo pravil celovitosti.

V okviru drugega prehoda skozi graf prilagojene različice procesa G', postopek prilagajanja zagotovi celovitost prilagojene različice. Najprej se preveri skladnost prilagojene različice procesa s pravili celovitosti (aktivnost 7). Kako se preverja skladnost, je podrobno opredeljeno v razdelku 3.5.3.6, kjer smo pravila za zagotavljanje celovitosti opredelili. V primeru neskladja z omejitvami iz metamodela, se lahko z uporabo pravila celovitosti sproži akcijo, ki jo pravilo predpisuje (aktivnost 8). Akcije v posledičnem delu pravil celovitosti lahko predlagajo različne načine za reševanje problema celovitosti. Te smo predstavili že v okviru obravnave pravila celovitosti (glej razdelek 3.5.3.6).

Vse aktivnosti v sklopu predstavljenega postopka, v katerih pride do izvajanja pravil, so na sliki (glej Slika 33) osenčene zeleno.

Na koncu postopka je potrebno v grafu G' obravnavati še vozlišča tipa aktivnost, katera nimajo nobenega sosednjega vozlišča, tipa izhodni izdelek (aktivnost 9). V takem primeru govorimo o aktivnostih brez izhodnih izdelkov, ki so bile vključene v prilagojeno različico procesa. Če aktivnost nima opredeljenega nobenega izhodnega izdelka, se moramo vprašati, ali je izvajanje te aktivnosti sploh smiselno. Odločitev o

tem prepustimo metodologu, ki opredeli, ali se obravnavano aktivnost izključi iz prilagojene različice. *Na ta način zagotovimo, da je v okviru postopka prilagajanja možno izločiti katerokoli aktivnost, kar ob predpostavki iz razdelka 3.5.3.2.1, o tem, katerim usmerjenim povezavam je možno pripisati pravilo procesnega toka, ni bilo mogoče*.

Izhod predstavljenega postopka predstavlja graf prilagojene različice procesa, ki je opredeljen s trojko $G' = (V', E', A')$. Iz izhodnega grafa je možno dobiti razširjen diagram aktivnosti prilagojene različice tako, da vozlišča tipa aktivnost iz množice V' preslikamo v aktivnosti, med katerimi potekajo usmerjene povezave iz množice A', ostala vozlišča iz množice V' pa se preslikajo v ustrezne elemente procesa, ki se z aktivnostmi povezujejo preko neusmerjenih povezav množice E'.

3.6.3.3 Obravnavanje neopredeljenih vrednosti karakteristik projekta

Pri zajemu karakteristik se lahko zgodi, da vrednosti določene karakteristike ne moremo opredeliti. Vzrok za to je lahko nepoznavanje pomena karakteristike, nedokončno izražene zahteve naročnika, nesprejete odločitve o uporabljeni arhitekturi programske opreme itd. V tem primeru ima določena karakteristika projekta neopredeljeno vrednost, ki lahko povzroča težave pri izračunu logične vrednosti elementa pogoja, ki to karakteristiko vsebuje, oziroma onemogoča izračun logične vrednosti celotnega pogoja v pravilu (odvisno od uporabljenih logičnih operatorjev v pogoju).

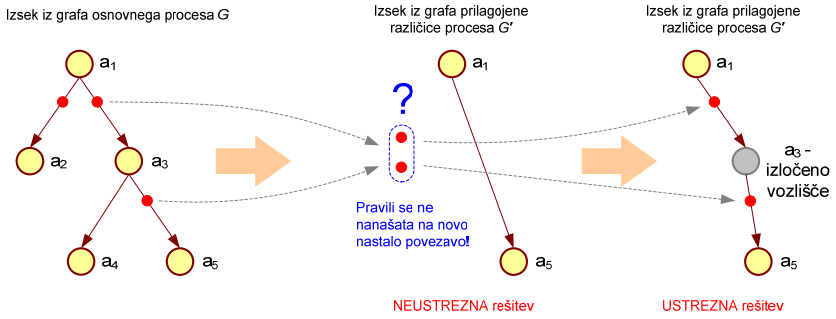
Ker predlagani pristop za prilagajanje procesa predvideva delovanje odločitvenega modela tudi ob uporabi nepopolnih vhodnih podatkih (neopredeljenih vrednostih karakteristik), je potrebno opredeliti ustrezno strategijo za reševanje tega problema. Najenostavnejši pristop k reševanju problema zahteva ustavitev postopka prilagajanja in posredovanje metodologa. Metodolog ima dve možnosti:

- ali opredeli vrednost karakteristike, če postane ta na podlagi izpeljanih dejstev do trenutka prekinitve že znana,
- ali pa mora sam sprejeti odločitev o vključitvi elementa, na katerega se nanaša odločitveno pravilo s problematično karakteristiko, v različico procesa.

Podoben problem se pojavi tudi ob zajemu atributov (glej razdelek 3.5.3.2), ki se uporabljajo v okviru elementov pogoja v pravilih procesnega toka. Tudi v tem primeru je najprimerneje postopek prilagajanja prekiniti in zahtevati posredovanje metodologa.

3.6.3.4 Obravnava izločanja vozlišč tipa aktivnost

Pri izločanju vozlišča tipa aktivnost iz grafa prilagojene različice procesa G' je potrebno biti pozoren na dejstvo, da se na novo nastalo povezavo med sosednjima vozliščema izločene aktivnosti ne nanaša nobeno odločitveno pravilo, kar onemogoči izvedbo postopka kreiranja pojasnila, ki ga bomo predstavili v okviru razdelka o zagotavljanju transparentnosti odločitev (glej razdelek 3.6.8). Problem in naknadno rešitev prikazuje Slika 34:



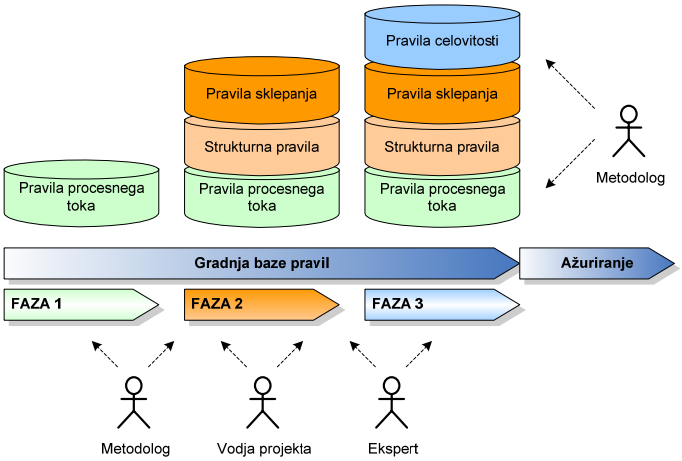
Slika 34: Prikaz problema izločanja vozlišča tipa aktivnost

Težavo je mogoče rešiti na naslednji način: ker se morebitno izločanje vozlišča, tipa aktivnost, ki nima sosednjih vozlišč tipa izhodni izdelek, izvede šele na koncu postopka prilagajanja, se tega vozlišča iz grafa dejansko ne izloči, ampak samo označi, kot izločeno. Na ta način je možnost izvedbe postopka pojasnjevanja še vedno zagotovljena, saj ostanejo povezave iz grafa G prisotne tudi v grafu G'.

3.6.4 Baza odločitvenih pravil

3.6.4.1 Uvod

Baza odločitvenih pravil predstavlja element konceptualnega modela, ki je namenjen shranjevanju odločitvenih pravil, katera smo predvideli za uporabo v okviru mehanizma za prilagajanje procesa. Odločitvena pravila smo v okviru podpoglavja 3.5 že razdelili v skupine, glede na namen njihove uporabe, zato si bomo v nadaljevanju ogledali, kako poteka gradnja baze odločitvenih pravil in njeno ažuriranje. Slika 35 prikazuje življenjski cikel baze pravil, ki obsega njeno konstruiranje in ažuriranje. Skozi predvidene faze uvedbe pristopa v organizacijski sistem, je prikazana gradnja baze, sodelujoče vloge ter odločitvena pravila, ki se v posamezni fazi dodajo v bazo.



Slika 35: Življenjski cikel baze pravil

3.6.4.2 Gradnja baze odločitvenih pravil

Gradnja baze pravil poteka po korakih, in sicer skladno s fazami uvedbe predlaganega pristopa v organizacijski sistem. V razdelku 3.4.3 smo v okviru uvedbe pristopa predvideli tri faze, in sicer:

- fazo konstruiranja prilagodljivega procesa,
- fazo učenja in
- fazo uporabe pristopa.

V fazi konstruiranja prilagodljivega procesa se bazo pravil napolni s pravili za izbiro ustreznega procesnega toka. Ta pravila se navezujejo na usmerjene povezave v grafu osnovnega procesa *G*, zato imajo v svojih premisah opredeljena začetna vozlišča (tipa aktivnost) povezav na katere se nanašajo. Pogoji v premisah so odvisni tako od parametrov, ki imajo vpliv na izbiranje procesnega toka, kot od karakteristik projekta. Pri opredelitvi pravil procesnega toka sodelujejo vloge, ki imajo izkušnje na področju modeliranja procesa ter ustrezna teoretična znanja. To so lahko metodolog, vodje projektov in eksperti na področju prilagajanja procesa.

V fazi učenja se v bazo odločitvenih pravil doda nova pravila, ki opredeljujejo obstoj neusmerjenih povezav v grafu splošnega procesa, oziroma ti. strukturna pravila. Pogoji v premisah teh pravil so sestavljenih iz več elementov pogoja. Vsakemu elementu pogoja se v okviru faze učenja določi vrednost karakteristike, pri kateri bo njegova logična vrednost resnična.

V okviru faze učenja se opredeli tudi ti. pravila sklepanja. Ta omogočajo preko izpeljanih dejstev opredeliti dodatna odločitvena pravila, ki se pripišejo posameznim neusmerjenim povezavam v grafu splošnega procesa. Pravila sklepanja uporabimo za predstavitev povezav med karakteristikami projektov oziroma njihovih medsebojnih odvisnosti. Strukturna pravila temeljijo na vrednostih karakteristik projektov, zato morajo vloge, ki jih opredelijo, dobro poznati vplive vseh vrst karakteristik na vključevanje elementov procesa v različice procesa, ki so namenjene potrebam različnih projektov. Vloge, ki so odgovorne za opredelite strukturnih pravil so podobne kot tiste, o katerih smo govorili v okviru pravil procesnega toka. Če pa hočemo doseči čim višji nivo prilagojenosti procesa tudi s sociološkega vidika, je priporočljivo, da ima vsaj ena od omenjenih vlog tudi ustrezna znanja s področij:

- kulturoloških vplivov na delo v skupinah [Hofstede 1997b in 2001; Olson 2003/2004, DeMarco 1999],
- organizacijskih znanosti in ravnateljevanja [Mihelčič 1999] ter
- sociooloških ved.

V fazi uporabe pristopa se prične dejansko izvajanje postopka prilagajanja osnovnega procesa. Ker je vsako prilagojeno različico procesa potrebno preveriti z vidika celovitosti, se v okviru faze uporabe opredeli še četrta množica pravil, in sicer pravila celovitosti. Ta temeljijo na omejitvah, ki jih predpisuje metamodel procesa v organizacijskem sistemu. Metamodel procesa zato tudi predstavlja vhod predlagan odločitveni model.

3.6.4.3 Ažuriranje baze pravil

Ažuriranje baze odločitvenih pravil obsega tri operacije nad pravili, in sicer dodajanje novih odločitvenih pravil, odstranjevanje ter posodabljanje obstoječih odločitvenih pravil. Postopek ažuriranja poteka v okviru:

- *faze uporabe pristopa*: Faza uporabe predstavlja operativno fazo predlaganega pristopa, v kateri se izvaja prilagajanje osnovnega procesa potrebam konkretnih projektov. Pri uporabi pristopa pridobivamo nove izkušnje o poteku prilagajanja, na podlagi katerih nastaja novo znanje. Novo znanje lahko predstavimo v obliki novih odločitvenih pravil kateregakoli tipa, ki jih metodolog doda v bazo, ali pa na njihovi osnovi posodobi ali celo izloči nekatera že obstoječa pravila v bazi pravil.
- *izvajanja projekta, ki uporablja prilagojeno različico procesa*: Med izvajanjem konkretnega projekta lahko ugotavljamo, ali poteka izvajanje aktivnosti na projektu v skladu s prilagojeno različico procesa. Če je različica procesa ustrezna, tako s

tehničnega kot sociološkega vidika, potem je med razvijalci dobro sprejeta [Vavpotič 2005]. V nasprotnem primeru mora metodolog ugotoviti, zakaj se določenih elementov različice procesa ne uporablja in na temelju teh ugotovitev ustrezno ažurirati odločitvena pravila. V okviru izvajanja projektov se lahko ažurira vse tipe odločitvenih pravil.

3.6.5 Množica karakteristik projektov

3.6.5.1 Uvod

Karakteristike projektov smo podrobno opredelili že v podpoglavju 3.3, zato se bomo v nadaljevanju disertacije osredotočili le na predstavitev pomena karakteristik z vidika elementa, ki predstavlja sestavni del konceptualne zgradbe odločitvenega modela.

3.6.5.2 Vloga elementa karakteristike projekta

V okviru konceptualne zgradbe odločitvenega modela so projektne karakteristike predstavljene s posebnim elementom, ki tako kot ostali gradniki, predstavlja enega od konceptov odločitvenega modela. Element predstavlja mehanizem za shranjevanje opredelitev karakteristik, ki se trenutno uporabljajo za opisovanje lastnosti projektov in za shranjevanje njihovih dopustnih vrednosti (zalog vrednosti D_i).

Omenjeni mehanizem je potreben, ker predstavljajo opredelitve karakteristik in njihovih zalog vrednosti temelj za nadaljnjo opredelitev pogojev v premisah naslednjih odločitvenih pravil:

- strukturna pravila,
- pravila procesnega toka in
- pravila sklepanja (glej razdelek 3.5.3).

Nabor karakteristik in njihove zaloge vrednosti se opredelijo v fazi konstruiranja prilagodljive metodologije, in sicer pred konstruiranjem osnovnega procesa. Pri tem je potrebno upoštevati tipične lastnosti projektov, ki se izvajajo v organizacijskem sistemu in na podlagi teh opredeliti posamične skupine karakteristik. Iz tega sledi, da naj se nabor opredeljenih karakteristik v prihodnosti ne bi bistveno spreminjal, oziroma naj bi ostal stalen.

Kljub temu pa odločitveni model dopušča tudi razširitve, tako glede obsega kot tudi zalog vrednosti karakteristik. Če se nabor karakteristik spremeni, ali če se spremeni zaloga vrednosti določene karakteristike, je potrebno izvesti ažuriranje ustreznih odločitvenih pravil, ki imajo v svojih premisah omenjeno karakteristiko navedeno kot enega od elementov pogoja. To je v konceptualnem modelu prikazano s povezavami, ki potekajo od elementa karakteristike projekta k elementom odločitvenega modela, ki predstavljajo odločitvena pravila.

3.6.6 Osnovna in izpeljana dejstva

3.6.6.1 Uvod

Tako osnovna kot izpeljana dejstva smo predstavili že v razdelku, v katerem smo podali opredelitev zgradbe odločitvenih pravil (glej razdelek 3.5.3). Namen tega razdelka je pojasniti njun pomen v okviru konceptualne zgradbe odločitvenega modela, kjer posamezni tip dejstev opredelimo z vidika vpliva na odločanje o vključevanju elementov procesa v prilagojene različice osnovnega procesa.

3.6.6.2 Osnovna dejstva

Osnovna dejstva predstavljajo v okviru konceptualne zgradbe odločitvenega modela element, ki omogoča predstavitev zajetih karakteristik konkretnega projekta v obliki, ki je primerna za uporabo v okviru odločitvenih pravil. Lahko bi tudi rekli, da osnovna dejstva predstavljajo prehod, skozi katerega v odločitveni model vstopajo vrednosti karakteristik za posamezni projekt. Kot smo predpostavili že v okviru razdelka 3.5.3.3, vsak projekt opredeljuje svojo množico osnovnih dejstev *MOD*. Množica osnovnih dejstev *MOD* se uporabi pri prilagajanju osnovnega procesa za določen projekt, v okviru faze uporabe pristopa. Vsako osnovno dejstvo se primerja z ustreznimi elementi pogojev v premisah odločitvenih pravil, ki so prikazani na sliki (glej Slika 32). Pravila celovitosti so pri omenjeni primerjavi izvzeta. V primeru, da se osnovno dejstvo v vrednosti karakteristike ujema z elementom pogoja, potem je logična vrednost tega elementa pogoja resnična. Logična vrednost pogoja v odločitvenem pravilu se tako izračuna iz logičnih vrednosti posameznih elementov, ki tvorijo obravnavani pogoj.

3.6.6.3 Izpeljana dejstva

Izpeljana dejstva nimajo enake vloge kot osnovna dejstva, saj ne predstavljajo vrednosti karakteristik projekta. Njihov namen je predstaviti dejstva, ki jih je možno izpeljati na temelju osnovnih dejstev s pomočjo pravil sklepanja (glej razdelek 3.5.3.5). Na temelju izpeljanih dejstev je v fazi učenja (glej razdelek 3.6.4.2) mogoče opredeliti dodatna strukturna pravila, ki opisujejo odvisnosti med karakteristikami projekta in imajo vpliv na vključevanje določenih elementov procesa v prilagojeno različico procesa. Kako se na osnovi izpeljanih dejstev opredeli novo strukturno pravilo je prikazano v razdelku o zgradbi pravil sklepanja (glej razdelek 3.5.3.5). Izpeljana dejstva, posredno preko dodatno opredeljenih strukturnih pravil vplivajo na izbiro ustreznih elementov procesa, ki se izbirajo v okviru mehanizma prilagajanja (glej Slika 32).

3.6.7 Sistem za izvajanje pravil

3.6.7.1 Uvod

Kot smo omenili že v razdelku o osnovnih in izpeljanih dejstvih (glej razdelek 3.6.6), se slednja generirajo na temelju osnovnih dejstev in pravil sklepanja. Za to poskrbi mehanizem sklepanja, ki predstavlja enega od sestavnih delov sistema za izvajanje pravil. Opis sistema za izvajanje pravil bomo podali v nadaljevanju razdelka, saj predstavlja enega od konceptov odločitvenega modela. V nadaljevanju bomo poleg opisa sistema za izvajanje pravil prikazali še njegovo arhitekturo in ga umestili v okvir konceptualne zgradbe odločitvenega modela za prilagajanje procesa. V zaključku razdelka bomo prikazali opis medsebojnih odvisnosti med karakteristikami projekta z uporabo pravil sklepanja.

3.6.7.2 Opredelitev pojma sistem za izvajanje pravil

Sistem za izvajanje pravil je na konceptualni ravni opredeljen z množico trditev in množico odločitvenih pravil, ki določajo način delovanja nad omenjeno množico trditev. Z vidika zgradbe so sistemi za izvajanje pravil razmeroma preprosti. Njihov osrednji element predstavlja množica odločitvenih pravil, ki sama po sebi že predstavlja osnovo ti. ekspertnih sistemov, kateri se množično uporabljajo na številnih področjih človekovega delovanja [Merritt 2000]. Celotno filozofijo ekspertnega sistema je možno strniti v naslednjo opredelitev: Ekspertni sistem predstavlja znanje eksperta, ki je zakodirano v obliki množice odločitvenih pravil. Ko je ekspertni sistem izpostavljen

določeni situaciji, bo na temelju uporabe mehanizmov umetne inteligence reagiral na podoben način kot pravi ekspert (človek).

Na večino modernih sistemov za izvajanje pravil je mogoče gledati z vidika bolj ali manj specializiranih lupin ekspertnih sistemov, ki podpirajo različne operacije, odvisne od okolja uporabe ali programiranje v okviru specifične domene.

Danes so sistemi za izvajanje pravil, tako tisti, ki naj bi nadomestili človeškega eksperta, kot tisti namenjeni avtomatizaciji poslovanja, del vsakega organizacijskega sistema. Njihova uporaba je postala povsem rutinska na področjih kot so: naročanje zalog, nadzor industrijskih procesov, usmerjanje telefonskih klicev in procesiranje spletnih obrazcev. Veliko komercialnih aplikacijskih strežnikov že vključuje določen stroj za izvajanje pravil, preostali pa ponujajo integracijo z njimi na eksplicitni ali implicitni način.

Sistemi za izvajanje pravil predstavljajo posebno skupino računalniških programov, za katere se uporablja tudi izraz stroji za izvajanje pravil. Stroj za izvajanje pravil sam po sebi ne vsebuje nobenih odločitvenih pravil, vse dokler jih vanj ne vnese programer. Iz tega sledi, da sam stroj ne vsebuje nobenega specifičnega znanja. Edino kar zna, je izvajanje pravil, ki so vanj vnesena. Stroj za izvajanje pravil običajno predstavlja del *okolja za razvoj in implementacijo pravil*. Lastnosti takih okolij se lahko med seboj zelo razlikujejo, kar je odvisno od namena uporabe samega stroja za izvajanje pravil.

3.6.7.2.1 Pravilo

Pravilo predstavlja obliko napotka oziroma ukaza, ki se uporabi v določeni situaciji. Na podlagi podane, zelo široke opredelitve, bi lahko sklepali, da je možno celotno znanje o svetu zakodirati v obliki odločitvenih pravil. Izkušnje so pokazale, da to pogosto drži. V splošnem pa lahko rečemo, da je možno vsako informacijo, ki jo je mogoče predstaviti v obliki logičnih izrazov, podati v obliki pravila.

Običajno se pravilo nahaja v obliki *if-then* stavka, v katerem *if* del (vzročni del) predstavlja ti. *premiso* ali *predikat*. Posledični del stavka (kar sledi za *then*) opredeljuje *akcijo* ali *zaključek*. *Domeno* pravila predstavlja množica vseh kombinacij podatkov, ki jih pravilo lahko uporabi. Sistem za izvajanje pravil uporablja pravila, na osnovi katerih iz premis izpeljuje zaključke. Na podlagi opisa iz zadnjega odstavka, je tudi lepo vidna sorodnost s pravili, ki smo jih opredelili za potrebe prilagajanja procesa.

3.6.7.3 Arhitektura sistema za izvajanje pravil

V prvotnih ekspertnih sistemih so bila pravila prepletena z ostalo programsko kodo, ki je tvorila ekspertni sistem, kar je pomenilo, da je bilo potrebno za nov ekspertni sistem (nova problemska domena) razvoj pričeti od začetka. Kasneje so ugotovili, da je znanje o določeni domeni smiselno ločiti od preostalega dela ekspertnega sistema, kar je pripeljalo do razvoja lupin ekspertnih sistemov. Kot smo že omenili, je na večino modernih sistemov za izvajanje pravil mogoče gledati z vidika bolj ali manj specializiranih lupin ekspertnih sistemov. Tipičen sistem za izvajanje pravil tako vsebuje naslednje elemente:

- mehanizem sklepanja,
- bazo pravil in
- delovni pomnilnik.

Mehanizem sklepanja tvorijo naslednje enote:

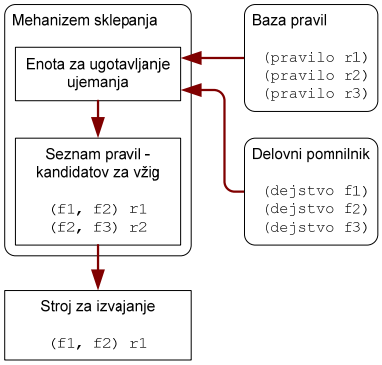
- enota za ugotavljanje ujemanja (ang. pattern matcher),
- seznam pravil - kandidatov za vžig (ang. agenda) in
- stroj za izvajanje.

Arhitekturo tipičnega sistema za izvajanje pravil prikazuje Slika 36. Sistem za izvajanje pravil uporablja zelo enostavno tehniko. V okviru enote za ugotavljanje ujemanja uporabi vsebino baze pravil, ki vsebuje znanje, zakodirano v obliki *if-then* stavkov in dejstva, shranjena v delovnem pomnilniku. Sistem pregleda pogoje v vseh pravilih in kreira konfliktno množico, ki jo sestavljajo pravila, katerih premise se ujemajo z dejstvi iz delovnega pomnilnika. Ta pravila tvorijo seznam pravil - kandidatov za vžig. Izmed pravil v tej podmnožici, se izbere eno, ki se sproži (vžge). Izbor pravila iz konfliktne množice je odvisen od uporabljene *strategije za razreševanje konfliktne situacije*[25] [AIdepot]. Ko izbrano pravilo "vžge", se izvede *akcija*, ki jo opredeljuje posledični del pravila (*then* del stavka). Akcija lahko opredeljuje izvedbo ažuriranja delovnega pomnilnika, spremembo vsebine baze pravil, oziroma karkoli drugega kar opredeli programer. Pravila se prožijo v zanki, dokler ni izpolnjen eden od naslednjih dveh pogojev:

- na voljo ni več nobenega pravila, ki bi se v premisi ujemal z dejstvi iz delovnega pomnilnika,

[25] *Strategija na temelju katere se iz konfliktne množice pravil izbere pravilo za vžig. Poznamo več srategij: strategija prvega pravila, strategija naključnega pravila, strategija najbolj specifičnega pravila, strategija najmanj uporabljanega pravila in strategija "najboljšega" pravila.*

- vžge pravilo, ki ima v akciji opredeljen zaključek izvajanja.



Slika 36: Arhitekura sistema za izvajanje pravil

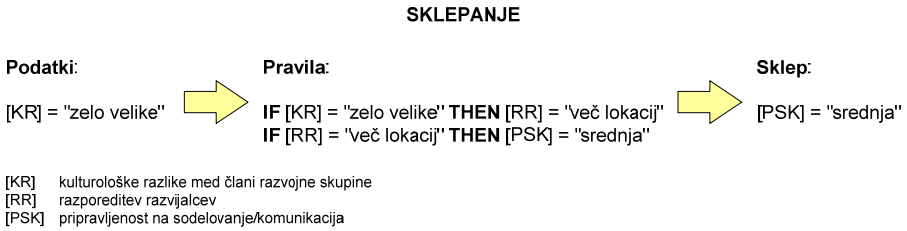
3.6.7.4 Metode sklepanja

Za reševanje problemov uporabljajo sistemi za izvajanje pravil dva temeljna pristopa, in sicer podatkovno usmerjen pristop (ang. forward chaining) in ciljno usmerjen pristop (ang. backward chaining) [Merritt 2000]. V nadaljevanju bomo predstavili podatkovno usmerjen pristop, ker se ta izkaže kot uporaben za iskanje novih (izpeljanih) dejstev, ki se uporabljajo v okviru odločitvenega modela za prilagajanje procesa.

3.6.7.4.1 Podatkovno usmerjen pristop

Za veliko problemov ni mogoče našteti vseh možnih odgovorov vnaprej in izmed njih izbrati pravilnega. V to kategorijo problemov se uvršča tudi problem prilagajanja procesa, saj imamo na voljo skoraj neskončno množico kombinacij rešitev (poti skozi graf). Za reševanje omenjenih problemov je primerna uporaba podatkovno usmerjenega pristopa, v okviru katerega se sklepanje izvaja na naslednji način: sistem spremlja trenutno stanje rešitve problema in išče pravila, ki bodo omenjeno stanje premaknila bližje h končni rešitvi.

Pri podatkovno usmerjenem pristopu je že pred izvedbo sklepanja potrebno na vhodu sistema zagotoviti vse potrebne podatke, v čemer se tudi razlikuje od ciljno usmerjenega pristopa, kjer se podatke v sistem vnaša po potrebi. Delovanje mehanizma sklepanja, ki uporablja podatkovno usmerjen pristop, je prikazano na sliki (glej Slika 37). Pristop primerja pogoje v premisah pravil z dejstvi v delovnem pomnilniku in izbira pravila za vžig.



Slika 37: Delovanje mehanizma sklepanja, ki temelji na podatkovno usmerjenem pristopu

3.6.7.5 Umestitev sistema za izvajanje pravil v okvir odločitvenega modela

Sistem za izvajanje pravil tvorijo v okviru konceptualne zgradbe odločitvenega modela za prilagajanje procesa (glej Slika 32) naslednji elementi, ki so hkrati tudi del odločitvenega modela:

- pravila sklepanja, ki predstavljajo bazo pravil sistema za izvajane pravil,
- osnovna in izpeljana dejstva, ki predstavljajo delovni pomnilnik sistema in
- mehanizem sklepanja.

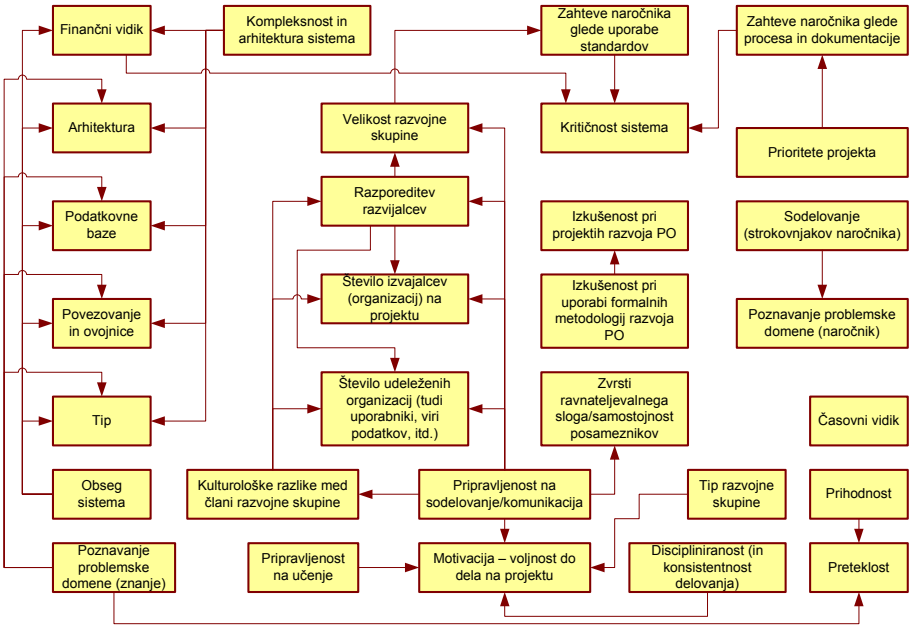
3.6.7.6 Opis odvisnosti med karakteristikami projekta s pravili sklepanja

Med karakteristikami projektov obstajajo medsebojne odvisnosti. Nekatere od njih se pojavljajo pri vseh projektih, zato jih opredelimo kot "splošne odvisnosti", druge pa se pojavijo redkeje, samo v okviru nekaterih projektov in jih opredelimo kot "posebne odvisnosti".

Ker se vrednost posamične karakteristike projekta predstavi v obliki osnovnega dejstva, je s pomočjo sistema za izvajanje pravil možno priti do dodatnih spoznanj, ki veljajo za vrednosti karakteristik obravnavanega projekta. Novo spoznanje se predstavi v obliki izpeljanega dejstva, ki se izpelje na temelju osnovnih dejstev, s pomočjo pravil sklepanja. Ker je prav pravilo sklepanja tisto, ki med seboj povezuje osnovna in izpeljana dejstva (glej razdelek 3.5.3.5), lahko od tod sklepamo, da je pravilo sklepanja možno uporabiti za opis odvisnosti med dvema karakteristikama projekta.

Medsebojno odvisnost med karakteristikama lahko prikažemo z usmerjeno povezavo, ki poteka od odvisne karakteristike proti karakteristiki, od katere je prva odvisna. Na ta način dobimo ti. diagram odvisnosti. S preiskovanjem diagrama odvisnosti, je možno na osnovi usmerjenih povezav med karakteristikami, opredeliti ti. dodatno plast pravil sklepanja, na osnovi katerih pridemo do izpeljanih dejstev.

Za izdelavo diagrama odvisnosti je odgovoren metodolog, ki dobro razume medsebojni vpliv karakteristik projekta. Diagram se izdela v fazi uporabe pristopa za prilagajanje procesa (glej razdelek 3.4.3.4), ponavadi za opis posebnih odvisnosti, ki so značilne za posamezni projekt. Primer diagrama odvisnosti, ki vsebuje tako splošne kot posebne odvisnosti podaja Slika 38:



Slika 38: Diagram odvisnosti med karakteristikami projekta

Omenimo naj še, da pri izdelavi diagrama odvisnosti ni potrebno identificirati vseh odvisnosti, ki veljajo med karakteristikami. Pri opredelitvi splošnih odvisnosti si metodolog lahko pomagamo z dognanji iz strokovne in znanstvene literature, dočim se pri opredelitvi posebnih odvisnosti izkaže predvsem njegova izkušenost. Vsekakor je bolje identificirati vsaj nekaj odvisnosti, kot pa nobene.

3.6.7.6.1 Prednosti uporabe sistema za izvajanje pravil

Vključitev sistema za izvajanje pravil v odločitveni model za prilagajanje procesa ni potreben pogoj za njegovo pravilno delovanje. Glavni namen uporabe tega sistema je izboljšati sam postopek prilagajanja osnovnega procesa. To se doseže z uporabo pravil sklepanja, na temelju katerih se izpelje nova (izpeljana) dejstva, ki posredujejo dodatne informacije o relacijah med karakteristikami projektov. Na temelju teh informacij je možno izvesti ukrepe, ki vodijo k izboljšavam odločitvenega modela, npr. posodabljanje odločitvenih pravil ali posodabljanje vrednosti karakteristik obravnavanega projekta.

Nadaljnje prednosti, ki jih lahko identificiramo pri uporabi sistema za izvajanje pravil, so še:

- višja kakovost sprejetih odločitev: Gre za odločitve, ki se nanašajo na vključevanje elementov procesa v prilagojene različice procesa. Sistem za izvajanje pravil zagotavlja konsistentne odgovore, zato je kakovost sprejetih odločitev višja.
- zajem strokovnega znanja, ki je zelo redko: Sistem za izvajanje pravil omogoča zajem posebnih odvisnosti med karakteristikami projektov.
- višja raven kakovosti pojasnjevanja rezultatov: Vključitev posameznega elementa procesa v prilagojeno različico lahko pojasnimo tudi na temelju odločitvenih pravil, ki so bila opredeljena na podlagi izpeljanih dejstev (glej razdelek 3.5.3.5).

3.6.8 Zagotavljanje transparentnosti odločitev

3.6.8.1 Uvod

V okviru odločitvenega modela se na podlagi odločitvenih pravil sprejemajo odločitve o vključevanju posamičnih vozlišč grafa *G* (elementov procesa) v graf *G'*, ki predstavlja prilagojeno različico procesa. S tega vidika predstavlja vsaka različica osnovnega procesa oziroma rezultat prilagajanja, neko končno množico sprejetih odločitev, za katere velja, da mora vsaka od njih biti pojasnljiva. Na ta način se zagotovi transparentnost samega odločitvenega modela, hkrati pa lahko vedno izvemo, na temelju katerih odločitvenih pravil je bil nek element procesa vključen v različico. V nadaljevanju razdelka sledi podrobna opredelitev mehanizma za pojasnjevanje rezultatov, s katerim zagotovimo zahtevano transparentnost predlaganega odločitvenega modela.

3.6.8.2 Mehanizem za pojasnjevanje rezultatov

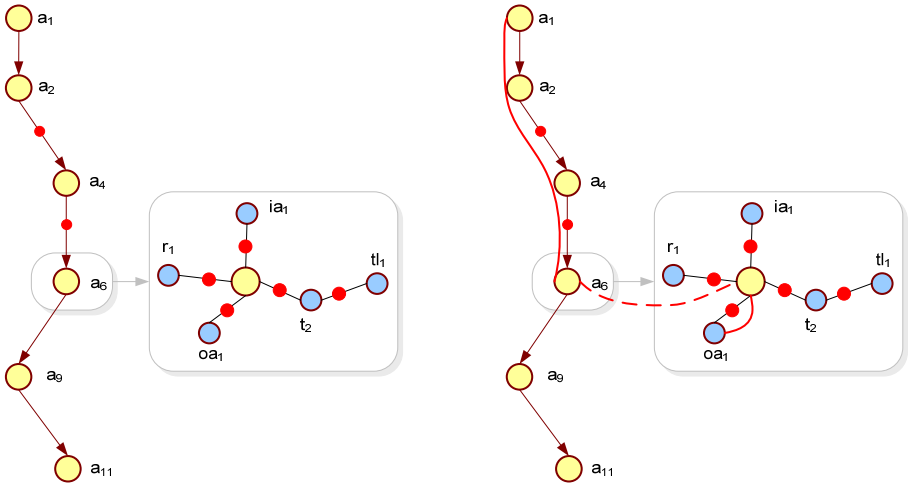
Odločitveni model, v taki obliki, kot smo ga predstavili v dosedanjih razdelkih, sicer že omogoča prilagoditi osnovni proces organizacijskega sistema projektom, ki se v okviru le tega izvajajo. Kljub temu pa nanj še vedno lahko gledamo le z vidika ti. črne škatle, saj nimamo neposrednega vpogleda v to, na podlagi katerih pravil je bil določen element procesa vključen v prilagojeno različico.

Omenjeni problem smo rešili z opredelitvijo mehanizma za pojasnjevanje rezultatov. Njegov namen je omogočiti pojasnjevanje odločitev, ki se sprejemajo v okviru gradnje

različice procesa na osnovi odločitvenih pravil in na ta način zagotoviti transparentnost celotnega odločitvenega modela, ki ga predlagamo v okviru pristopa.

3.6.8.2.1 Izhodišča za opredelitev mehanizma pojasnjevanja rezultatov

Mehanizem za pojasnjevanje rezultatov smo zasnovali kot postopek, katerega vhod predstavlja graf prilagojene različice osnovnega procesa $G' = (V', E', A')$. Povezave s pripisanimi pravili v omenjenem grafu predstavljajo temelj za pojasnjevanje odločitev o vključevanju elementov v prilagojen proces. V okviru aktivnosti 5, postopka gradnje prilagojene različice procesa, se v graf različice preslikajo povezave iz grafa osnovnega procesa (glej razdelek 3.6.3.2). Iz tega lahko sklepamo, da so tudi povezavam v grafu G' pripisana pravila iz različnih skupin odločitvenih pravil (strukturna pravila, pravila procesnega toka in strukturna pravila, pridobljena na temelju izpeljanih dejstev), kar je prikazano na primeru grafa prilagojene različice, na sliki (glej Slika 39, levo). Povezavam prirejena odločitvena pravila so prikazana z rdečo piko:



Slika 39: Graf prilagojene različice procesa in sprehod v grafu

Na temelju povezav iz množic E' in A', grafa G', lahko za vsako vozlišče, ki pripada množici V', pojasnimo, na temelju katerih odločitvenih pravil je bilo to vključeno v graf G', oziroma zakaj je bil element procesa, ki temu vozlišču ustreza, vključen v prilagojeno različico. Vzemimo za primer vozlišče oa_1, ki predstavlja izhodni izdelek aktivnosti a_6. Če želimo pojasniti, njegovo vključitev v graf, se moramo sprehoditi od ciljnega vozlišča, ki ga v podanem primeru predstavlja vozlišče oa_1, do začetnega vozlišča grafa (začetna aktivnost - a_1). Odločitvena pravila pripisana povezavam, ki tvorijo sprehod v grafu (glej Slika 39, desno), predstavljajo množico pravil *EXP1*, s pomočjo katere je možno pojasniti vključenost danega vozlišča v graf.

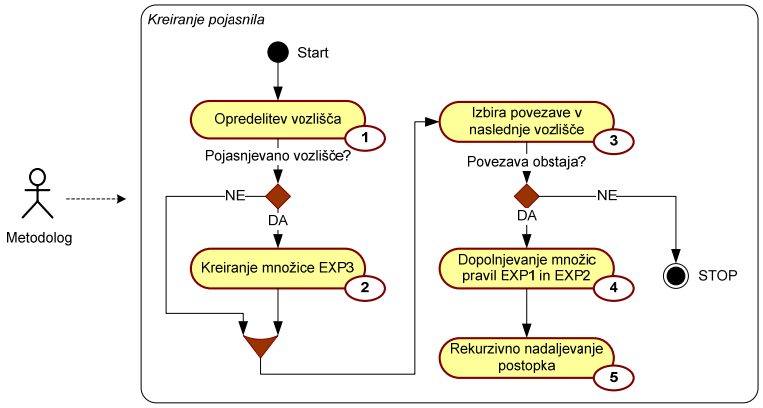
Vendar pa množica *EXP1* še ne predstavlja vsega, na podlagi česar je možno pojasniti vključenost posameznih vozlišč v graf prilagojene različice, saj so v njej zajeta samo strukturna pravila in pravila procesnega toka. Ker se nekatera strukturna pravila lahko opredelijo na podlagi izpeljanega dejstva (glej razdelek 3.5.3.5), je za podrobnejše pojasnjevanje vključenosti pripadajočega vozlišča v graf prilagojene različice *G′*, možno uporabiti tudi mehanizem za izvajanje pravil. Pojasnilo, ki ga generira omenjeni mehanizem, tvori množico pravil sklepanja *EXP2*, na temelju katerih je mehanizem prišel do določenega izpeljanega dejstva. Obe do sedaj opredeljeni pojasnjevalni množici, *EXP1* in *EXP2*, je možno tvoriti na sorazmerno enostaven način.

Poleg omenjenih množic, pa je možno opredeliti še dodatno pojasnjevalno množico *EXP3*, ki jo tvorijo pravila celovitosti, uporabljena v okviru preverjanja celovitosti za obravnavano vozlišče v grafu različice procesa *G′*. Množica *EXP3* ima še posebej velik pomen v primeru pojasnjevanja vključenosti vozlišča, ki ni bilo izbrano na podlagi odločitvenih pravil preostalih tipov. Takrat predstavlja množica *EXP3* edino pojasnilo, zakaj je bilo omenjeno vozlišče vključeno v graf prilagojene različice *G′*.

Obravnavo izhodišč za opredelitev pojasnjevanja lahko torej sklenemo z naslednjo ugotovitvijo: Pojasnilo o vključenosti določenega vozlišča v graf prilagojene različice procesa *G′* je predstavljeno z unijo množic: $EXP1 \cup EXP2 \cup EXP3$, katero označimo z oznako *EXP*. V nadaljevanju sledi opredelitev postopka za kreiranje pojasnila in umestitev pojasnjevalnih množic v okvir njegovih aktivnosti.

3.6.8.2.2 Postopek kreiranja pojasnila

Postopek kreiranja pojasnila mora na svojem izhodu zagotoviti pojasnilo o tem, zakaj je določeno vozlišče vključeno v graf prilagojene različice. Pojasnilo je podano v obliki seznama odločitvenih pravil, ki tvorijo množico *EXP*. Vrstni red pravil mora biti podan v takem zaporedju, kot si sledijo vozlišča v sprehodu grafa, in sicer od obravnavanega vozlišča, proti začetnemu vozlišču grafa *G′*. S tem se zagotovi, da se vključenost obravnavanega vozlišča pojasnjuje v smeri vzrokov. Postopek pojasnjevanja prikazuje diagram aktivnosti na sliki (glej Slika 40):



Slika 40: Postopek kreiranja pojasnila

V okviru prve aktivnosti (aktivnost 1) postopka se opredeli obravnavano vozlišče postopka, ki predstavlja element sprehoda v grafu *G'*. Če je to vozlišče, za katerega je potrebno pojasniti vključenost v graf, se v naslednjem koraku postopka kreira množica *EXP3* (aktivnost 2), ki vključenost začetnega vozlišča pojasni z vidika pravil celovitosti.

V okviru tretje aktivnosti (aktivnost 3) se izbere povezava v tisto vozlišče, ki je en korak bližje začetnemu vozlišču grafa, kot obravnavano vozlišče. Če ta povezava obstaja, se v množico *EXP1* in/ali množico *EXP2* doda pravila, ki so tej povezavi pripisana (aktivnost 4), drugače se postopek zaključi.

Sledi rekurzivno nadaljevanje postopka (aktivnost 5) v vozlišču, ki se nahaja na začetku povezave, ki smo jo obravnavali v četrti aktivnosti. Ker to vozlišče ni vozlišče, za katerega se pojasnjuje vključenost v graf *G'*, se zanj in vse nadaljnje predhodnike do vključno startnega vozlišča grafa, ne bo v množico *EXP3* dodalo nobenega pravila celovitosti več.

Postopek uporablja ustrezna vloga v organizacijskem sistemu, v našem primeru metodolog, kar je prikazano na sliki (glej Slika 40). Namen uporabe postopka je pridobivanje izkušenj o delovanju samega odločitvenega modela in s tem posledično izboljševanje delovanja le tega, skozi ažuriranje odločitvenih pravil v bazi pravil.

3.6.9 Sklep

Rezultat prilagajanja predstavlja graf različice procesa, ki je opredeljen s trojko $G' = (V', E', A')$. Graf predstavlja formalen prikaz prilagojene različice procesa na višjem nivoju abstrakcije, ki se ga pred uporabo v realnem okolju preslika v diagram, ki

uporablja eno od notacij za prikaz procesa. V našem primeru bi to bil razširjen diagram aktivnosti. Z uporabo notacije, ki jo razumejo razvijalci programske opreme, približamo proces njegovim bodočim uporabnikom - razvijalcem, ki sodelujejo v projektni skupini.

4 Programsko orodje

4.1 Uvod

V okviru navedbe ciljev pričujoče doktorske disertacije (glej razdelek 2.4.4) smo navedli tudi opredelitev izhodišč za izdelavo prototipa programskega orodja za podporo prilagajanja procesa. Programsko orodje, ki ga predstavljamo v nadaljevanju, temelji na opredelitvah formalnega pristopa za prilagajanje procesa, ki smo jih podali v okviru tretjega poglavja doktorske naloge.

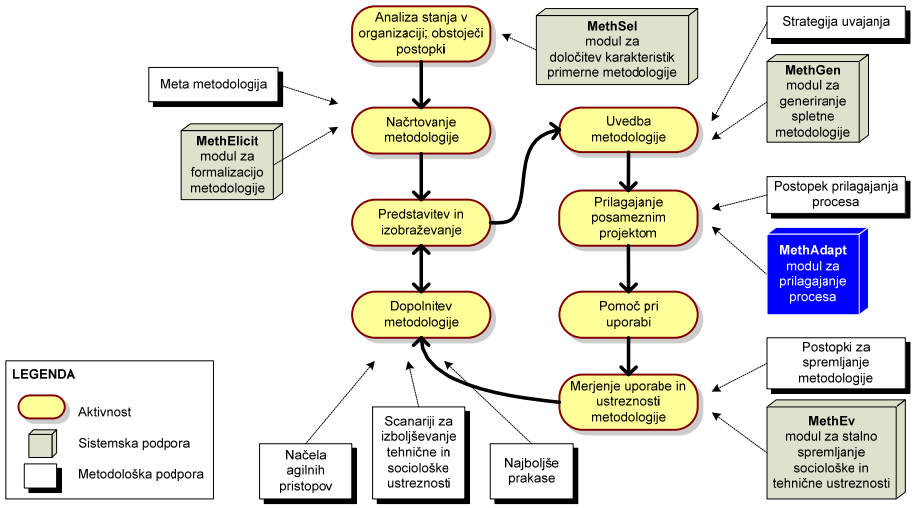
Na začetku poglavja podamo kratko predstavitev raziskovalnega projekta, v okviru katerega poteka izdelava doktorske disertacije in razvoj programskega modula za prilagajanje procesa razvoja IS. V nadaljevanju sledi predstavitev orodja AMT, kot skupka modulov za podporo dela metodologov pri načrtovanju in razvoju metodologij v organizacijskih sistemih za razvoj programske opreme. Sledi predstavitev modula za prilagajanje osnovnega procesa MethAdapt, ki predstavlja jedro poglavja. V okviru tega se osredotočimo na predstavitev zahtevane funkcionalnosti programskega modula in na predstavitev ključnih primerov uporabe. Pri tem temeljimo na prispevku o prilagajanju metodologij potrebam konkretnih projektov avtorjev Bajca, Vavpotiča in Krisperja [Bajec 2005b]. Opis modula sklenemo s predstavitvijo lupine ekspertnega sistema Jess in njeno vlogo v okviru programskega modula.

4.2 Projekt centra odličnosti

Center odličnosti za informacijske in komunikacijske tehnologije in storitve (CO ICT) predstavlja partnersko povezavo tehničnih fakultet, med njimi tudi Fakultete za računalništvo in informatiko v Ljubljani, inštituta IJS ter ustanov in podjetij, ki želijo oblikovati in združiti tehnično, aplikativno, inovativno, razvojno ter raziskovalno odličnost na širšem multidisciplinarnem področju informacijskih in telekomunikacijskih tehnologij ter storitev.

V okrilju projekta centra odličnosti - Informacijske in komunikacijske tehnologije in storitve (CO ICT) poteka v Laboratoriju za informatiko, Fakultete za računalništvo in informatiko, Univerze v Ljubljani, raziskovalno delo v okviru projekta OBVLADOVANJE PROCESA RAZVOJA PRI RAZVOJU REŠITEV ZA ELEKTRONSKO POSLOVANJE, katerega cilj je izdelati *pristop in računalniško podporo* za načrtovanje in razvoj metodologij, ki so tako tehnično kot tudi socialno

ustrezne posameznim organizacijskim sistemom oziroma njihovim projektom. Slika 41 prikazuje scenarij za načrtovanje in razvoj prilagodljive metodologije razvoja IS v organizacijskem sistemu. V okviru scenarija so prikazane izbrane aktivnosti, za katere je potrebno v okviru projekta CO ICT izboljšati metodološko podporo in izdelati programske module za podporo njihove avtomatizacije.



Slika 41: Metodološka podpora in programski moduli za podporo izvajanja aktivnosti scenarija za načrtovanje in razvoj prilagodljive metodologije razvoja IS

Prikazani programski moduli predstavljajo sistemsko podporo za delo z metodologijo. Vseh pet modulov skupaj tvori obširnejšo aplikacijo, ki nosi ime AMT (ang. Agile Methodology Toolset) in bo predmet obravnave naslednjega podpoglavja.

4.3 Aplikacija AMT

Aplikacija AMT je programsko orodje, ki je namenjeno zajemu, prilagajanju in spremljanju tehnične in socialne ustreznosti metodologij razvoja IS. Sestavljen je iz naslednjih petih programskih modulov:

- **MethElicit**: Nudi podporo formalizaciji metodologije.
- **MethSel**: Omogoča opredelitev karakteristik metodologije, ki bi bila primerna za uporabo z vidika celotnega organizacijskega sistema.
- **MethGen**: Omogoča avtomatično generiranje prikaza metodologije, primernega za splet.

- **MethAdapt**: Podpira prilagajanje osnovnega procesa potrebam projektov, ki se izvajajo v okviru organizacijskega sistema. Osnovni proces se določi z uporabo modula MethSel.
- **MethEv**: Nudi podporo za stalno spremljanje sociološke in tehnične ustreznosti.

Orodje AMT smo zasnovali kot spletno aplikacijo, ki teče na spletnem strežniku. Podatki, ki jih aplikacija uporablja, se hranijo na podatkovnem strežniku, do same aplikacije pa se dostopa s katerikoli spletnim odjemalcem, ki podpira skriptni jezik Java in aktivne strežniške strani (ang. Active Server Pages). Gre torej za uporabo klasične trinivojske arhitekture, za katero smo se odločili zaradi potrebe zagotavljanja enostavne uporabe in enostavnosti dostopanja do same aplikacije (ne potrebujemo namenskega odjemalca). Za izdelavo orodja AMT smo izbrali Microsoftovo tehnologijo .NET, za podatkovni strežnik pa smo predvideli uporabo produkta SQL Server 2000.

Namen izdelave programskega orodja je nuditi metodologu pomoč pri njegovem delu z metodologijami. Postopki izbire, prilagajanja in ugotavljanja sprejetosti metodologij so namreč zelo kompleksni in brez ustrezne računalniške podpore v praksi zelo težko izvedljivi. Z ustrezno programsko podporo je tako mogoče zagotoviti hitrejše in učinkovitejše izvajanje aktivnosti, za katere je v okviru svojih nalog odgovoren metodolog, hkrati pa ga razbremenimo tudi zahtev glede posedovanja dodatnih znanj s področij organizacijskih in sociooloških ved, ki so potrebna za učinkovito izvedbo postopkov izbire in prilagajanja metodologij v organizacijskem sistemu.

V nadaljevanju disertacije se bomo osredotočili na opredelitev izhodišč za izdelavo prototipa modula MethAdapt, ki omogoča prilagajanje osnovnega procesa potrebam projektov, ki se izvajajo v okviru danega organizacijskega sistema. Omenjeni modul je nastal na temelju opredelitve inovativnega pristopa za prilagajanje procesa, ki smo ga formalno opredelili v predhodnem poglavju (glej poglavje 3) in je na sliki (glej Slika 41) obarvan modro.

4.4 Modul MethAdapt

4.4.1 Uvod

Programski modul MethAdapt predstavlja sestavni del obsežnejše aplikacije AMT in je namenjen podpori postopka prilagajanja razvojnega procesa. Modul predstavlja interaktivno orodje, ki ga pri svojem delu uporablja metodolog oziroma katera druga vloga, ki je v organizacijskem sistemu zadolžena za prilagajanje osnovnega procesa

lastnostim individualnih projektov. Modul MethAdapt temelji na opredelitvi odločitvenega modela, ki smo ga predlagali v okviru uvedbe pristopa za prilagajanje procesa v organizacijski sistem (glej poglavje 3) in bo v nadaljevanju disertacije podrobno opisan.

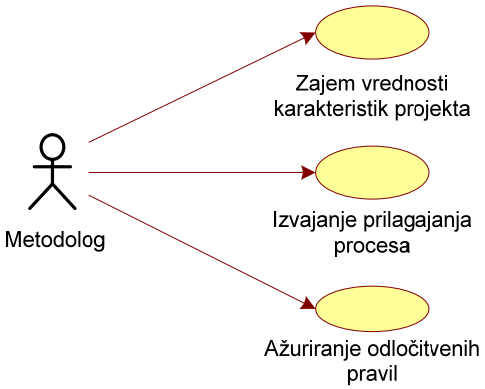
V razdelku najprej skozi ključne primere uporabe predstavimo predvideno funkcionalnost modula MethAdapt in hierarhično strukturo njegovih zaslonskih mask. Predvidena funkcionalnost modula temelji na zahtevah, ki jih podajajo cilji doktorske disertacije, glede izdelave programskega orodja (glej razdelek 2.4.4.3). Sledi opis logičnega podatkovnega modela, ki opredeljuje podatke, potrebne za delovanje modula. V okviru opisa programske logike, v nadaljevanju predstavimo dva algoritma, ki predstavljata temelj za delovanje modula. Prvi je namenjen konstruiranju prilagojene različice procesa, drugi pa zagotavljanju njene celovitost. Kot tretjega predstavimo algoritem za pojasnjevanje rezultatov prilagajanja. Razdelek zaključimo s predstavitvijo načina uporabe mehanizma sklepanja, ki ga uporabimo kot dodatno možnost za iskanje izpeljanih dejstev, na temelju katerih lahko izboljšamo celotni proces prilagajanja.

4.4.2 Model primerov uporabe modula MethAdapt

Predvideno funkcionalnost modula MethAdapt prikažemo s pomočjo modela primerov uporabe. Opisi primerov uporabe opredeljujejo zahtevane funkcionalnosti, ki jih mora modul MethAdapt nuditi uporabniku (metodologu oziroma vodji projekta). V nadaljevanju razdelka bodo predstavljene naslednje ključne funkcionalnosti modula:

- zajem vrednosti karakteristik projekta,
- izvajanje prilagajanja procesa in
- ažuriranje odločitvenih pravil

Funkcionalnosti modula MethAdapt so prikazane na diagramu primerov uporabe na sliki (glej Slika 42):



Slika 42: Diagram primera uporabe

4.4.2.1 Zajem vrednosti karakteristik projekta

Primer uporabe zajema vrednosti karakteristik projekta prikažemo z opisom neslednjih tokov dogodkov:

- Tok dogodkov "Zajem tehnoloških karakteristik":
  1. Sistem izpiše zaslonsko masko 2.2.1 s seznamom tehnoloških karakteristik.
  2. Uporabnik vnese vrednosti karakteristik in potrdi vnos.
  3. Sistem zapiše vrednosti tehnoloških karakteristik v podatkovno bazo.
  4. Sistem obvesti uporabnika o uspešnosti izvedene akcije.
- Tok dogodkov "Zajem sociooloških karakteristik":
  1. Sistem izpiše zaslonsko masko 2.2.2 s seznamom sociooloških karakteristik.
  2. Uporabnik vnese vrednosti karakteristik.
  3. Sistem zapiše vrednosti sociooloških karakteristik v podatkovno bazo.
  4. Sistem obvesti uporabnika o uspešnosti izvedene akcije.
- Tok dogodkov "Zajem zahtev naročnika":
  1. Sistem izpiše zaslonsko masko 2.2.3 s seznamom možnih zahtev naročnika.
  2. Uporabnik vnese vrednosti zahtev.
  3. Sistem zapiše vrednosti zahtev v podatkovno bazo.
  4. Sistem obvesti uporabnika o uspešnosti izvedene akcije.
- Tok dogodkov "Izpis izpeljanih dejstev":
  1. Sistem omogoči izpis zaslonske maske z izpeljanimi dejstvi (v okviru zaslonske maske 2.2).
  2. Uporabnik lahko zahteva izpis izpeljanih dejstev. Izpeljana dejstva se pridobijo na temelju pravil sklepanja s pomočjo zunanjega mehanizma sklepanja, ki se vključi v aplikacijo.
  3. Sistem izpiše zaslonsko masko 2.2.4 z izpeljanimi dejstvi. Na podlagi izpeljanih dejstev lahko uporabnik pride do novih spoznanj o povezavah med karakteristikami projekta.
  4. Sistem omogoči uporabniku vpogled, na podlagi katerih osnovnih dejstev je mehanizem sklepanja, s pomočjo pravil sklepanja, prišel do izpeljanega dejstva.

5. Uporabnik lahko popravi vrednosti predhodno vnesenih karakteristik obravnavanega projekta.

4.4.2.2 Izvajanje prilagajanja procesa

Primer uporabe izvajanja prilagajanja procesa prikažemo z opisom naslednjega toka dogodkov:

- Tok dogodkov "Prilagodi proces":
  1. Sistem izpiše zaslonsko masko 2.3 z možnostjo sprožanja postopka prilagajanja.
  2. Uporabnik sproži prilagajanje procesa.
  3. Sistem na isti zaslonski maski 2.3 izpiše seznam elementov procesa, ki sestavljajo prilagojeno različico procesa.
  4. Uporabnik lahko za vsak element procesa preko povezave na zaslonsko masko za izpis 2.3.1, dobi odgovor, na podlagi katerih odločitvenih pravil je bil ta vključen v prilagojeno različico. Izpiše se zaslonska maska s pravili, ki pojasnjujejo vključenost določenega elementa v različico procesa (uporaba algoritma za pojasnjevanje rezultatov prilagajanja).

4.4.2.3 Ažuriranje odločitvenih pravil

Primer uporabe ažuriranja odločitvenih pravil prikažemo z opisom naslednjih tokov dogodkov:

- Tok dogodkov "Pregled in posodabljanje pravil":
  1. Sistem izpiše zaslonsko masko 3.1, na kateri uporabnik izbere katera pravila naj se izpišejo (filter pravil glede na tip ali element procesa).
  2. Uporabnik potrdi izpis pravil želenega tipa.
  3. Sistem izpiše pravila želenega tipa.
  4. Uporabnik preko povezave "posodobi" izvede klic ustrezne (glede na tip odločitvenega pravila) zaslonske maske za posodabljanje (klic zaslonske maske 3.1.1 ali 3.1.2 ali 3.1.3).
- Tok dogodkov "Posodabljanje pravila procesnega toka ali strukturnega pravila":
  1. Sistem izpiše zaslonsko masko 3.1.1 za posodabljanje odločitvenega pravila in tip pravila, ki se posodablja.
  2. Glede na tip pravila, ki se posodablja, sistem izpiše njegove ustrezne komponente.
  3. Sistem izpiše obstoječi pogoj pravila, ki se posodablja.
  4. Uporabnik spremeni pogoj, ki je sestavljen iz več elementov pogoja in sproži posodabljanje pravila.
  5. Sistem zapiše posodobljeno pravilo v podatkovno bazo in obvesti uporabnika o uspešnosti izvedene akcije.
- Tok dogodkov "Posodabljanje pravila celovitosti":
  1. Sistem izpiše zaslonsko masko 3.1.2 za posodabljanje pravila celovitosti.

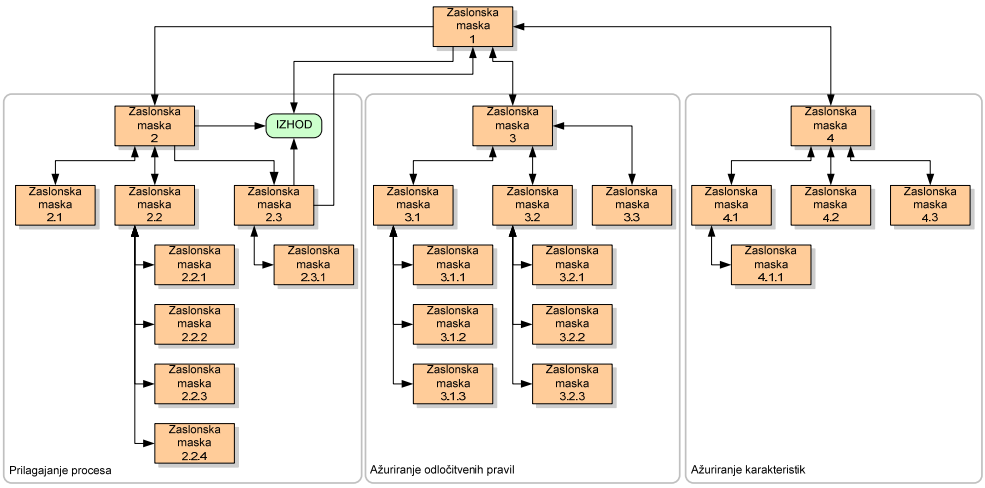
2. Sistem izpiše obstoječi pogoj pravila, ki se posodablja.
3. Uporabnik spremeni pogoj, ki je sestavljen iz več elementov pogoja in sproži posodabljanje pravila.
4. Sistem zapiše posodobljeno pravilo v podatkovno bazo in obvesti uporabnika o uspešnosti izvedene akcije.

- Tok dogodkov "Posodabljanje pravila sklepanja":
  1. Sistem izpiše zaslonsko masko 3.1.3 za posodabljanje pravila sklepanja.
  2. Sistem izpiše obstoječi pogoj pravila in izpeljano dejstvo.
  3. Uporabnik lahko posodobi pogoj, ki je lahko sestavljen iz več elementov pogoja.
  4. Uporabnik lahko posodobi izpeljano dejstvo, ki je lahko podobno strukturo kot komponenta pogoj ter sproži posodabljanje pravila.
  5. Sistem zapiše posodobljeno pravilo v podatkovno bazo in obvesti uporabnika o uspešnosti izvedene akcije.
- Tok dogodkov "Dodajanje pravil":
  1. Sistem izpiše zaslonsko masko 3.2, na kateri uporabnik izbere, kateri tip odločitvenega pravila bo dodal.
  2. Uporabnik izbere tip pravila za dodajanje.
- Tok dogodkov "Dodajanje pravila procesnega toka ali strukturnega pravila":
  1. Sistem izpiše zaslonsko masko 3.2.1 za dodajanje odločitvenega pravila.
  2. Sistem ponudi uporabniku na izbiro, ali bo dodal pravilo procesnega toka ali strukturno pravilo.
  3. Uporabnik vnese oba elementa procesa, med katerima velja odločitveno pravilo.
  4. Uporabnik doda pogoj, ki je sestavljen iz več elementov pogoja in sproži dodajanje pravila.
  5. Sistem zapiše dodano pravilo v podatkovno bazo in obvesti uporabnika o uspešnosti izvedene akcije.
- Tok dogodkov "Dodajanje pravila celovitosti":
  1. Sistem izpiše zaslonsko masko 3.2.2 za dodajanje pravila celovitosti.
  2. Uporabnik vnese oba metaelementa med katerima velja pravilo celovitosti.
  3. Uporabnik doda pogoj, ki je sestavljen iz več elementov pogoja in sproži dodajanje pravila.
  4. Sistem zapiše dodano pravilo v podatkovno bazo in obvesti uporabnika o uspešnosti izvedene akcije.
- Tok dogodkov "Dodajanje pravila sklepanja":
  1. Sistem izpiše zaslonsko masko 3.2.3 za dodajanje pravila sklepanja.
  2. Uporabnik doda pogoj, ki je lahko sestavljen iz več elementov pogoja.
  3. Uporabnik doda izpeljano dejstvo, ki ima podobno strukturo kot komponenta pogoj ter sproži posodabljanje pravila.
  4. Sistem zapiše dodano pravilo v podatkovno bazo in obvesti uporabnika o uspešnosti izvedene akcije.
- Tok dogodkov "Brisanje pravil":
  1. Sistem izpiše zaslonsko masko 3.3, na kateri uporabnik izbere katera pravila naj se izpišejo (filter pravil glede na tip ali element procesa).

2. Uporabnik potrdi izpis pravil želenega tipa.
3. Uporabnik med izpisanimi pravili označi pravila za brisanje in sproži postopek brisanja.
4. Sistem obvesti uporabnika o uspešnosti izvedene akcije.

4.4.3 Struktura zaslonskih mask

Modul MethAdapt tvori množica zaslonskih mask, preko katerih poteka interaktivna komunikacija med uporabnikom - metodologom in sistemom. Zaslonske maske so med seboj povezane preko spletnih povezav v hierarhično strukturo, ki jo prikazuje Slika 43:



Slika 43: Hierarhija zaslonskih mask

Podroben načrt zaslonskih mask, v okviru katerega smo za vsako zaslonsko masko opredelili izgled, njene komponente in povezave z drugimi zaslonskimi maskami, se nahaja v poglavju Priloga.

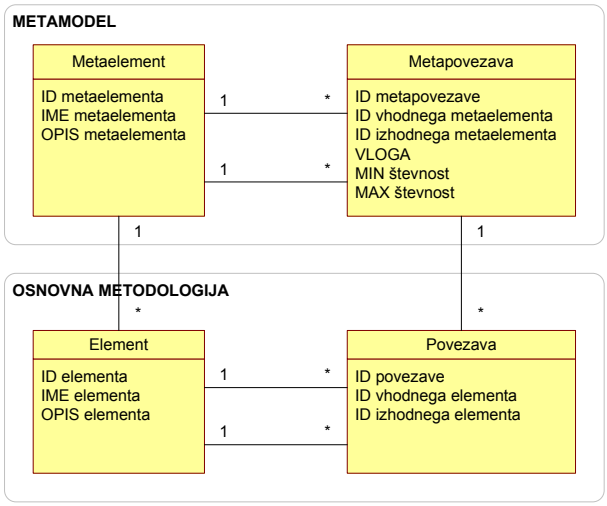
4.4.4 Podatkovna struktura za shranjevanje elementov procesa

Že v okviru razdelka o orodjih za podporo konstruiranja metodologij (glej razdelek 2.3.1.6), smo omenili, da je repozitorij eno bistvenih orodij za uspešno izvedbo konstruiranja metodologije, ko gre za najvišjo stopnjo prilagodljivosti konstruiranja - modularno konstruiranje metodologij (glej razdelek 2.3.1.2). Repozitorij metodologij, ki ga označimo tudi s pojmom baza metodologij, hrani fragmente metodologij, skupaj z njihovimi opisi. Največji izziv pri razvoju repozitorija tako predstavlja opredelitev mehanizma za visokonivojsko klasifikacijo fragmentov metodologij, ki mora zagotoviti

pravilen opis posameznega fragmenta metodologije, tako da vemo, čemu je ta namenjen, ne da bi morali pregledovati njegovo specifikacijo. Raziskave na področju opredelitve ustreznega mehanizma za iskanje fragmentov so še vedno zelo intenzivne.

Pristop k prilagajanju procesa, ki ga predlagamo v okviru doktorske disertacije, temelji na iskanju poti skozi graf, ki predstavlja osnovni proces organizacijskega sistema (glej razdelek 3.4.2.1). Iz tega sledi, da so povezave med elementi, ki sestavljajo proces, vnaprej znane. Za vsak element procesa, ki tvori splošni proces, vemo, s katerimi sosednjimi elementi je povezan, zaradi česar je množica elementov - kandidatov za nadaljevanje gradnje procesa, v vsakem trenutku poznana. Repozitorija fragmentov, kot ga opredeljuje področje konstruiranja metodologij, v okviru disertacije predlaganega pristopa, tako sploh ne potrebujemo. Na podlagi te ugotovitve bomo v nadaljevanju opredelili podatkovno strukturo za shranjevanje elementov procesa.

Ker se podatkovna struktura za shranjevanje elementov procesa uporablja v okviru vseh modulov orodja AMT, mora posledično ta podpirati shranjevanje vseh elementov metodologije, ki so opredeljeni v okviru metamodela zajete metodologije organizacijskega sistema. Elemente metodologije, in s tem tudi elemente procesa, opredeljuje metamodel metodologije, iz česar sledi, da je v okviru podatkovne strukture potrebno hraniti tudi podatke o metamodelu zajete metodologije. Na podlagi podanih zahtev predlagamo podatkovno strukturo, ki jo prikazuje konceptualni podatkovni model na sliki (glej Slika 44):



Slika 44: Entitetni tipi za shranjevanje elementov procesa

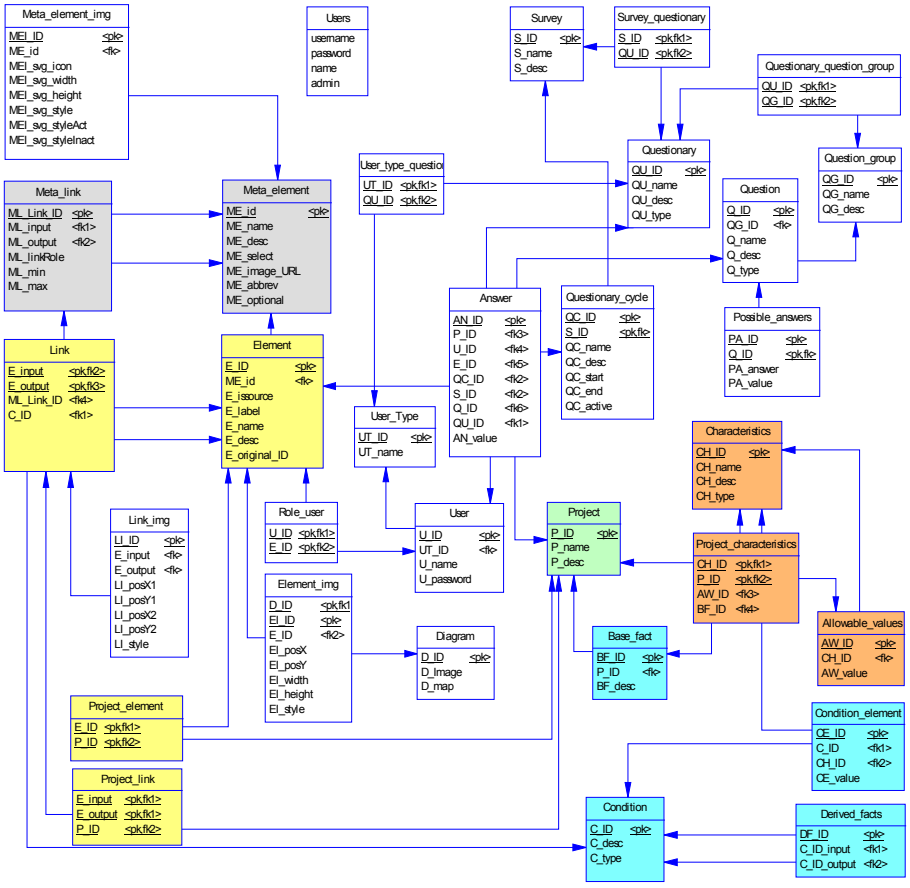
Entitetna tipa na zgornji polovici slike (glej Slika 44), sta namenjena shranjevanju podatkov o elementih in povezavah, ki tvorijo metamodel prilagodljive metodologije v organizacijskem sistemu. Vsaka metapovezava povezuje med seboj dva metaelementa. V okviru entitetnega tipa Metapovezava je potrebno poudariti tudi pomen atributov MIN številnost in MAX številnost. Ta dva atributa opredeljujeta omejitev, ki je pripisana metapovezavi in označuje število primerkov metaelementa s katerim se določen metaelement lahko povezuje. Podrobno smo pomen te omejitve predstavili že v okviru razdelka o pravilih celovitosti (glej razdelek 3.5.3.6).

Spodnja polovica slike (glej Slika 44) prikazuje entitetna tipa namenjena shranjevanju podatkov o elementih dejanske metodologije, ki je bila uvedena v nek organizacijski sistem - metodologija, ki se opredeli v prvi fazi scenarija za konstruiranje prilagodljivih metodologij za razvoj IS (glej razdelek 2.4.5).

Logični podatkovni model za shranjevanje podatkov o elementih metodologije vsebuje še nekatere dodatne atribute (glej Slika 45), ki so potrebni z vidika delovanja preostalih modulov orodja AMT.

4.4.5 Podatkovni model

Aplikacija AMT shranjuje podatke v enotni podatkovni bazi, do katere lahko dostopajo vsi njeni moduli. Vsak od programskih modulov uporablja svojo množico tabel, v kateri se lahko nahajajo tudi take, ki hranijo podatke, ki so relevantni tudi za druge module. V nadaljevanju je predstavljen logični podatkovni model celotne podatkovne strukture AMT, v okviru katerega so tabele, ki pripadajo modulu MethAdapt, obarvane (glej Slika 45). Omenjene tabele tvorijo logični podatkovni podmodel obravnavanega modula. Nebarvane tabele v modelu pripadajo ostalim modulom v okviru AMT.



Slika 45: Logični podatkovni model

4.4.5.1 Opis tabel

V nadaljevanju podajamo opis tabel, ki jih uporablja modul MethAdapt (glej Tabela 3). Za vsako tabelo smo navedli ime in opis podatkov, ki se v njej hranijo.

Tabela	Opis
Element	Podatki o elementih, ki tvorijo osnovno metodologijo v organizacijskem sistemu. V okviru teh elementov se nahajajo tudi elementi osnovnega procesa organizacijskega sistema. Elemente, ki lahko nastopajo v okviru splošnega procesa, opredeljujejo metaelementi metodologije, ki se hranijo v tabeli Meta_element.
Link	Podatki o povezavah med elementi procesa. Vsaka povezava lahko obstaja med dvema elementoma procesa.
Project_element	Povezovalna tabela z referencami na tiste elemente procesa, ki se nahajajo v prilagojeni različici procesa.
Project_link	Povezovalna tabela, ki vsebuje reference na tiste povezave, ki se nahajajo v prilagojeni različici procesa.

Project	Podatki o projektih, za katere se izvede prilagajanje osnovnega procesa.
Characteristics	Tabela hrani podatke o karakteristikah, ki se uporabljajo za opisovanje lastnosti projektov.
Project_characteristics	Tabela s podatki o vrednostih karakteristik za posamični projekt.
Allowable_values	Zaloge vrednosti (dovoljene vrednosti) karakteristik.
Condition	Podatki o pogojih, ki predstavljajo elemente odločitvenih pravil. Tabela vsebuje reference na povezave, katerim so pogoji pripisani.
Condition_element	Tabela elementov pogoja. Element pogoja predstavlja sestavni del pogoja.
Base_fact	Tabela osnovnih dejstev. Osnovno dejstvo predstavlja temeljno resnico, ki velja o nečem. V našem primeru predstavljajo osnovna dejstva vrednosti karakteristik za posamezni projekt, za katerega se izvaja prilagajanje osnovnega procesa.
Derived_fact	Tabela izpeljanih dejstev, ki jih pridobimo na temelju osnovnih dejstev.

Tabela 3: Logični podatkovni podmodel modula MethAdapt

4.4.6 Programska logika

Temelj programske logike modula MethAdapt predstavljajo trije algoritmi, ki so podrobneje opredeljeni v nadaljevanju razdelka. Opredelitev vsakega od njih smo podali v obliki psevdokode, za katero je značilno, da se osredotoča na delovanje algoritma in ne na jezikovne posebnosti določenega programskega jezika. Poimenovanja, ki smo jih uporabili v prikazanih procedurah, se ujemajo s poimenovanji tabel in atributov v logičnem podatkovnem modelu (glej predhodni razdelek), s čimer skušamo zagotoviti čim večjo razumljivost delovanja algoritma. V razdelku bomo obravnavali:

- algoritem za konstruiranje prilagojene različice procesa,
- algoritem za preverjanje celovitosti prilagojene različice procesa in
- algoritem za pojasnjevanje rezultatov prilagajanja.

4.4.6.1 Algoritem za konstruiranje prilagojene različice procesa

Algoritem za konstruiranje prilagojene različice procesa je v okviru modula MethAdapt implementiran z rekurzivno proceduro KreirajInstancoProcesa. Delovanje procedure temelji na povezovanju elementov procesa, shranjenih v tabeli Element in njihovih medsebojnih povezav, shranjenih v tabeli Link. Procedura zapisuje podatke o prilagojeni različici procesa v dve tabeli, in sicer Project_element in Project_link (glej

razdelek 4.4.5.1). Ob klicu, je proceduri KreirajInstancoProcesa potrebno posredovati dva parametra:

- `StartnaAktivnost`: vrednost atributa E_ID aktivnosti, ki predstavlja začetno aktivnost prilagojene različice procesa.
- `Projekt`: vrednost atributa P_ID projekta, za katerega se bo izdelala prilagojena različica procesa.

Delovanje algoritma temelji na postopku izgradnje prilagojene različice procesa, ki smo ga obravnavali v okviru razdelka o postopku za izgradnjo prilagojene različice procesa (glej razdelek 3.6.3.2). V nadaljevanju podajamo psevdokodo procedure KreirajInstancoProcesa:

```
PROCEDURE KreirajInstancoProcesa (StartnaAktivnost: integer; Projekt: integer);
BEGIN
   //vstavi začetno aktivnost v tabelo Project_element
   INSERT INTO Project_element VALUES (Projekt, StartnaAktivnost);
   //poišči povezave iz trenutne aktivnosti
   SELECT E_output, C_ID INTO Links FROM Link WHERE E_input = StartnaAktivnost;
   FOR l IN Links DO
   BEGIN
      //preveri, ali je povezavi pripisan pogoj. Ali je C_ID<>NULL?
      IF Povezava_ima_pripisan_pogoj THEN
      BEGIN
         //preveri, ali je pogoj, pripisan povezavi izračunljiv
         IF Pogoj_ni_izračunljiv THEN
         PosredovanjeMetodologa (l);
         IF PogojiIzpolnjeni (StartnaAktivnost, l.E_output) THEN
         BEGIN
            //vstavi element na koncu povezave v tabelo Project_element
            INSERT INTO Project_element VALUES (Projekt, l.E.output);
            //vstavi povezavo med obrvnavano aktivnostjo in izhodnim elementom
            //v tabelo Project_link
            INSERT INTO Project_link VALUES (Projekt, StartnaAktivnost,
            l.E_output);
            //rekurzivni klic procedure
            KreirajInstancoProcesa (l.E_output, Projekt);
         END;
      END
      ELSE
      BEGIN
         //vstavi element na koncu povezave v tabelo Project_element
         INSERT INTO Project_element VALUES (Projekt, l.E.output);
         //vstavi povezavo med obrvnavano aktivnostjo in izhodnim elementom
         //v tabelo Project_link
         INSERT INTO Project_link VALUES (Projekt, StartnaAktivnost,
         l.E_output);
         //rekurzivni klic procedure
         KreirajInstancoProcesa (l.E_output, Projekt);
      END;
   END;
END;
```

4.4.6.2 Algoritem za preverjanje celovitosti prilagojene različice procesa

Algoritem za preverjanje celovitosti prilagojene različice procesa je v okviru modula MethAdapt implementiran s proceduro PreveriInstancoProcesa. Delovanje procedure

temelji na uporabi pravil celovitosti (glej razdelek 3.5.3.6), s pomočjo katerih se preveri celovitost prilagojene različice procesa. Ob klicu, je proceduri PreveriInstancoProcesa potrebno posredovati naslednji parameter:

- `Projekt`: vrednost atributa P_ID projekta, za katerega je potrebno preveriti celovitost prilagojene različice procesa.

Delovanje algoritma temelji na predpostavkah in opisu pravil celovitosti, ki smo jih predstavili v okviru opredelitve pravil celovitosti (glej razdelek 3.5.3.6). V nadaljevanju podajamo psevdokodo procedure PreveriInstancoProcesa:

```
PROCEDURE PreveriInstancoProcesa (Projekt: integer);
BEGIN
   //izberi vse povezave, ki se nahajajo v različici procesa
   SELECT E_input, E_output INTO Links FROM Project_link WHERE P_ID = Projekt;
   //za vsako povezavo preveri, ali se podreja omejitvam pravila celovitosti
   FOR l IN Links DO
   BEGIN
      //ugotovi kakšna je številnost za obravnavano povezavo (min, max)
      SELECT min, max INTO Min, Max FROM Meta_link WHERE M_LinkID =
        (SELECT M_LinkID FROM Link WHERE E_input = l.E_input AND E_output =
         l.E_output);
      //izberi tip izhodnega elementa procesa
      SELECT ME_ID INTO ElType FROM Element WHERE E_ID = l.E_output;
      //preveri število povezav iz vhodnega elementa do elementov istega tipa,
      //kot je izhodni element
      SELECT COUNT(*) INTO NumOfLinks
      FROM Project_link PL,Element E
      WHERE PL.E_output = E.E_ID AND E.ME_ID = ElType;
      //če se število povezav nahaja izven opredeljenih mej, sporoči napako
      IF NumOfLinks NOT BETWEEN (Min, Max) THEN JaviNapako;
   END;
END;
```

4.4.6.3 Algoritem za pojasnjevanje rezultatov prilagajanja

Algoritem za pojasnjevanje rezultatov je v okviru modula MethAdapt implementiran s proceduro PojasniVključitevElementa, ki izpiše vsa odločitvena pravila, ki so pripeljala do vključitve obravnavanega elementa v prilagojeno različico procesa. Pri tem se odločitvena pravila izpišejo v smeri od obravnavanega elementa proti korenu drevesa (glej razdelek 3.6.8). Ob klicu je proceduri PojasniVključitevElementa potrebno posredovati naslednje parametre:

- `Projekt`: vrednost atributa P_ID projekta, v okviru katerega se nahaja pojasnjevano vozlišče oziroma element procesa.
- `Id_vozlišče`: vrednost atributa E_ID vozlišča, za katerega je potrebno preveriti celovitost prilagojene različice procesa.
- `Projekt`: vrednost atributa P_ID projekta, za katerega je potrebno preveriti celovitost prilagojene različice procesa.

Delovanje algoritma temelji na mehanizmu za pojasnjevanje rezultatov, ki smo ga obravnavali v okviru razdelka 3.6.8.2. V nadaljevanju podajamo psevdokodo procedure PojasniVključitevElementa:

```
PROCEDURE PojasniVključitevElementa (Projekt: integer; Id_vozlišče: integer;
Tekoče_vozlišče: integer);
BEGIN
   //če gre za pojasnjevano vozlišče, potem se kreira množica pravil
   //celovitosti
   IF Id_vozlišče = Tekoče_vozlišče THEN Kreiraj EXP3;
   //izberi povezave, ki se končujejo v obravnavanem vozlišču
   SELECT E_input, E_output INTO Links FROM Project_link WHERE E_output =
   Tekoče_vozlišče;
   //preštej število povezav, ki vodijo v obravnavano vozlišče
   SELECT COUNT(*) INTO NumOfLinks FROM Links;
   //če povezava obstaja, se procedura rekurzivno nadaljuje, drugače pa se
   //konča
   IF (NumOfLinks>0) THEN
   BEGIN
      Dopolni(EXP2);
      Dopolni(EXP3);
      PojasniVključitevElementa (Projekt; Id_vozlišče; E_input);
   END;
END;
```

4.4.7 Uporaba sistema za izvajanje pravil

4.4.7.1 Opredelitev izhodišč za uporabo sistema za izvajanje pravil

Opredelitev ciljev doktorske disertacije, v okviru določitve izhodišč za izdelavo prototipa programskega orodja za prilagajanje procesa, ne predvideva vključitve sistema za izvajanje pravil v okvir prototipa modula MethAdapt.

Kljub temu je vključitev sistema za izvajanje pravil v modul MethAdapt nujna, če hočemo zagotoviti celovito programsko podporo predlaganemu pristopu za prilagajanje procesa. Prav zaradi kasnejšega zagotavljanja celovitosti programskega orodja, smo v okviru načrta modula MethAdpat, prej omenjeno vključitev, vseeno predvideli. V nadaljevanju razdelka zato podajamo samo opis predvidene izbire sistema za izvajanje pravil in opišemo namen njegove vključitve v modul.

4.4.7.2 Jess

V okviru modula MethAdapt smo predvideli tudi uporabo sistema za izvajanje pravil. Odločili smo se za uporabo programskega okolja za izvajanje pravil Jess [Jess]. Jess je napisan v programskem jeziku Java, zato predstavlja idealno orodje za vstavljanje tehnologije za izvajanje pravil v programske produkte, napisane v tem jeziku, hkrati pa

ga je možno enostavno povezati tudi z modulom MethAdapt, katerega izhodišča smo opredelili v okviru doktorske disertacije.

Programsko okolje za izvajanje pravil Jess je nastalo na osnovi lupine ekspertnega sistema CLIPS [Clips]. Za razliko od slednjega, je Jess dinamično okolje, ki temelji na jeziku Java, zaradi česar nudi tudi neposreden dostop do vseh javanskih aplikacijskih vmesnikov (ang. API) za dostopanje do podatkovnih baz, manipuliranje z datotekami, rokovaje z mrežnimi povezavami itd.

Za razliko od nekaterih komercialnih programskih produktov, npr. ILOG [ILOG], ki nudijo namensko podporo izvajanju pravil, je Jess brezplačen za uporabo v akademske namene. V primeru vključitve v komercialne produkte pa je zanj potrebno plačati relativno zelo majhen znesek, ki je v primerjavi s komercialnimi produkti skoraj zanemarljiv.

4.4.7.3 Namen uporabe sistema za izvajanje pravil

Sistem za izvajanje pravil je v okviru modula MethAdapt namenjen pridobivanju informacij o obstoju odvisnosti med dvojicami karakteristik določenega projekta. Odvisnost med karakteristikama projekta se lahko opredeli v obliki pravila sklepanja, in sicer lahko že v fazi učenja - druga faza predvidene uvedbe predlaganega pristopa v organizacijski sistem (glej razdelek 3.4.3.3). Če v okviru danega projekta, za katerega se izvaja postopek prilagajanja procesa, obstaja odvisnost med karakteristikama projekta, potem bo pogojni del določenega pravila sklepanja lahko resničen, njegov sklepni del pa bo predstavljal veljavno izpeljano dejstvo. Množica izpeljanih dejstev predstavlja novo znanje, ki ga je kasneje možno uporabiti kot temelj za ažuriranje strukturnih pravil iz množice *SR* (glej razdelek 3.5.3.5).

5 Sklep

5.1 Uvod

V okviru poglavja najprej podamo končne ugotovitve, do katerih smo prišli v okviru izdelave doktorske disertacije. Sledi predstavitev prispevkov k znanosti. V zaključku poglavja podamo še ideje, ki so se porodile med raziskovalnim delom in samo izdelavo doktorske disertacije ter možnosti za nadaljnje delo.

5.2 Končne ugotovitve

Doktorska disertacija predstavlja nov pristop k prilagajanju procesa razvoja IS za potrebe individualnih projektov, ki se izvajajo v okviru danega organizacijskega sistema. Predlagani pristop je formaliziran z opredelitvijo odločitvenega modela, ki postopek prilagajanja procesa opredeli na temelju uporabe odločitvenih pravil.

Delo v okviru disertacije temelji na izsledkih raziskav s področja konstruiranja metodologij. To področje opredeljuje več oblik situacijskega konstruiranja metodologij, katere se med seboj ločijo po stopnji prilagodljivosti, ki jo nudijo uporabniku v okviru konstruiranja. Disertacija razširja obstoječ pristop k prilagajanju procesa, ki ga obravnava ena od oblik situacijskega konstruiranja metodologij. Obstoječ pristop temelji na predpostavki, da je prilagajanje splošno namenske metodologije možno izvesti z izbiro ustrezne, vnaprej določene poti skozi metodologijo. Tega disertacija razširi tako, da vnaprej določene poti skozi metodologijo odpravi. Pot skozi metodologijo se opredeli skozi postopek prilagajanja procesa in ta (skonstruirana pot) na koncu predstavlja prilagojeno različico procesa. Na ta način se lahko sam proces, ki ga predpisuje metodologija natančneje prilagodi, tako da se bolj prilega samim potrebam projekta, za katerega se prilagajanje procesa izvaja.

Disertacija v okviru poglavja o prilagajanju procesa razvoja IS opredeli pojem elementa procesa. Ta se v nadaljevanju naloge uporabi v okviru samega pristopa za prilagajanje, kot opredelitev najmanjše enote, ki se lahko uporabi kot sestavni del za konstruiranje procesa.

Na temelju opredelitve predlaganega pristopa, disertacija zasnuje konceptualno zgradbo odločitvenega modela, ki na podlagi odločitvenih pravil usmerja postopek prilagajanja. Teoretična podlaga, ki je nastala v okviru snovanja odločitvenega modela, se pri

snovanju prototipa programskega orodja uporabi za opredelitev izhodišč za njegovo izdelavo. V okviru tega disertacija opredeli ključne algoritme za izdelavo prilagojenih instanc osnovnega procesa in predstavi podatkovno strukturo za shranjevanje elementov procesa.

5.3 Pregled prispevkov k znanosti

Disertacija je prispevala naslednje prispevke k znanosti:

- **Formalno opredelitev odločitvenega modela za prilagajanje procesa**. V disertaciji predstavljen odločitveni model za prilagajanje procesa razvoja IS temelji na uporabi odločitvenih pravil in karakteristik za opis lastnosti projektov. Odločitvena pravila se delijo v več tipov. Pravila procesnega toka in strukturna pravila predstavljajo temelj za delovanje mehanizma prilagajanja, kateri usmerja konstruiranje prilagojene različice procesa. Tretja skupina pravil - pravila celovitosti se uporabi v okviru postopka za preverjanje celovitosti izdelane različice procesa. Opredelijo se na podlagi omejitev, ki jih predpisuje metamodel metodologije. Četrta skupina odločitvenih pravil se uporabi v okviru mehanizma sklepanja. Ta pravila opredeljujejo odvisnosti med karakteristikami projektov in omogočajo pridobivanje novih dejstev, s katerimi se lahko natančneje opredeli lastnosti projekta in s tem posredno izboljša učinek postopka prilagajanja. Lastnosti projekta disertacija obravnava tako s tehnološkega kot sociološkega vidika.

- **Opredelitev elementa procesa**. V okviru formalne opredelitve odločitvenega modela disertacija poda tudi opredelitev pojma elementa procesa, z uporabo metamodela prilagodljive metodologije, s pomočjo katerega se formalno predstavi znanje o zajeti metodologiji v obravnavanem organizacijskem sistemu.

- **Opredelitev sociooloških karakteristik**. V okviru formalne opredelitve odločitvenega modela poda disertacija opredelitev sociooloških karakteristik, s katerimi se v okviru odločitvenega modela zajame sociološko komponento razvijalcev, ki sestavljajo projektno skupino.

- **Opredelitev izhodišč za izdelavo prototipa programskega orodja za prilagajanje procesa**. V disertaciji predlagano orodje za podporo postopka prilagajanja procesa temelji na teoretičnih opredelitvah, podanih v okviru predlaganega pristopa, ki temelji na uporabi odločitvenega modela za

prilagajanje procesa. Predlagano orodje predstavlja zelo močan pripomoček za pomoč pri delu metodologa. Še posebej, če je podprt z modulom za risanje procesa, ki preko grafične predstavitve osnovnega procesa organizacijskega sistema omogoča vnašati odločitvena pravila za izbor elementov in izris končne - prilagojene različice procesa. Osnovni proces se prilagodi lastnostim projekta, ki se zajamejo s pomočjo karakteristik.

- **Opredelitev podatkovne strukture za shranjevanje elementov procesa**. V okviru opredelitve izhodišč za izdelavo programskega orodja za prilagajanje procesa je v disertaciji predlagana podatkovna struktura za shranjevanje podatkov o elementih procesa.

5.4 Predlogi za nadaljnje delo

V disertaciji smo podali formalno opredelitev odločitvenega modela za prilagajanje procesa razvoja IS in postavili temelje za izdelavo prototipa programskega orodja za podporo prilagajanju procesa. Možnosti, ki jih vidimo za nadaljevanje dela, je zato več. Med pomembnejšimi so:

- razširitev predlaganega pristopa z repozitorijem,
- izdelava repozitorija,
- raziskava možnosti prilagajanja vsebine opisov elementov procesa posameznim članom razvojne skupine,
- raziskava možnosti prenosa predlaganega pristopa na druge problemske domene.

5.4.1 Razširitev predlaganega pristopa z repozitorijem

Najvišja stopnja prilagodljivosti situacijskega konstruiranja metodologij predvideva izgradnjo metodologij iz posameznih sestavnih delov (fragmentov), ki se hranijo v repozitoriju fragmentov (glej razdelek 2.3.1). Na podlagi uvedbe repozitorija fragmentov je možno tudi pristop za prilagajanje procesa, ki ga predlagamo v okviru doktorske disertacije, funkcionalno razširiti tako, da bi omogočal dosegati višjo raven prilagodljivosti.

Ideja o razširitvi pristopa temelji na uporabi dveh tipov elementov procesa:

- obstoječih elementih procesa, ki se nahajajo v množici elementov osnovne metodologije, ki je bila predhodno že uvedena v organizacijski sistem in
- elementov procesa, ki jih opredeljujejo procesni fragmenti v okviru repozitorija fragmentov.

V disertaciji predlagan pristop za prilagajanje procesa bi bilo potrebno razširiti tako, da bi za obravnavani projekt omogočal dopolniti izbrano podmnožico elementov procesa iz osnovne metodologije z manjkajočimi elementi procesa, ki bi se pridobili iz repozitorija fragmentov.

Izbrano podmnožico elementov osnovne metodologije se opredeli glede na kriterije sprejetosti, ki ocenjujejo, kako je določen element procesa pri uporabnikih metodologije sprejet s tehničnega in sociološkega vidika [Vavpotič 2005]. Tako bi omenjeno podmnožico tvorili elementi, ki so s strani razvijalcev v projektni skupini zelo dobro sprejeti. Izbrani elementi bi predstavljali ključne elemente procesa v prilagojeni različici osnovnega procesa, med katere bi se v nadaljevanju umestilo manjkajoče elemente procesa, ki bi jih pridobili iz repozitorija fragmentov. Pri tem bi bilo potrebno v okviru razširitve pristopa opredeliti ustrezen mehanizem, ki bi na podlagi porajajočih potreb med gradnjo različice procesa, lahko opredelil lastnosti iskanih elementov procesa. Predvidevamo, da bi za opredelitev takega mehanizma morali globlje poseči v teorijo situacijskega konstruiranja metodologij in upoštevati relacije alternativnosti, komplementarnosti in predhodnosti fragmentov metodologij, ki jih v svojih delih opredeljuje Mirbel [Mirbell 2003 in 2004].

Pogoj za predstavljeno razširitev je povezava obstoječega odločitvenega modela za prilagajanje procesa z repozitorijem fragmentov, zato je uporaba le tega predpogoj za nadaljevanje dela.

5.4.2 Izdelava repozitorija

Repozitorij predstavlja eno bistvenih orodij za uspešno izvedbo konstruiranja metodologij, ko gre za najvišjo stopnjo prilagodljivosti postopka konstruiranja. Iz tega lahko sklepamo, da o razširitvi v disertaciji predlaganega pristopa, z možnostjo uporabe fragmentov metodologij, ki se hranijo v repozitoriju, lahko razmišljamo šele takrat, ko imamo repozitorij na razpolago.

Bistveni del repozitorija ne predstavlja podatkovna struktura za shranjevanje podatkov o fragmentih metodologij z njihovimi opisi, ampak mehanizem za učinkovito iskanje fragmentov v repozitoriju. Kot smo omenili že v razdelku o orodjih za podporo

konstruiranja metodologij (glej razdelek 2.3.1.6), predstavlja v okviru razvoja repozitorija še danes največji izziv za znanstvenike s področja situacijskega konstruiranja metodologij prav razvoj mehanizma za visokonivojsko klasifikacijo fragmentov. Omenjeni mehanizem mora zagotoviti pravilen opis posameznega fragmenta metodologije, ki se nahaja v repozitoriju, tako da vemo čemu je ta namenjen, ne da bi nam bilo treba pregledati njegovo specifikacijo. Na podlagi pravilne klasifikacije fragmentov metodologij je možno izvajati učinkovite poizvedbe nad repozitorijem.

Nadaljnje delo v smeri izdelave repozitorija vidimo predvsem v opredelitvi učinkovite klasifikacijske sheme za porazdelitev fragmentov metodologij v skupine glede na kontekst njihove uporabe. Pri tem bi seveda poleg tehničnega vidika uporabe fragmenta morali upoštevati tudi sociološki vidik konteksta, v katerem naj bi se fragment uporabil.

5.4.3 Raziskava možnosti prilagajanja vsebine opisov elementov procesa posameznim članom razvojne skupine

Pristop za prilagajanje procesa, ki ga predlagamo v okviru doktorske disertacije, bi lahko razširili tudi tako, da bi omogočal dodatno prilagajanje samih elementov procesa, ki so pred tem že bili vključeni v prilagojeno različico procesa. Vključenim elementom bi bilo možno prilagoditi obseg opisa, glede na lastnosti samega uporabnika, ki jih uporablja. V okviru nadaljnje raziskave bi bilo potrebno identificirati dodatne karakteristike posameznika, ki vplivajo na zahtevano raven obsega opisa posameznega elementa procesa. Pri tem bi se osredotočili predvsem na sociološke karakteristike.

Razlika z obstoječim pristopom, ki je predlagan v okviru disertacije, se kaže v objektu opazovanja, za katerega se opredeljujejo sociološke karakteristike projektov. V okviru predlaganega pristopa opazujemo sociološke karakteristike razvijalcev z vidika celotne razvojne skupine. Pri nadgradnji obstoječega pristopa pa se točka opazovanja sociooloških karakteristik prestavi na posameznika v razvojni skupini.

Identifikacija ustreznih karakteristik, ki imajo vpliv na obseg opisa elementa, bi omogočala opredeliti dodatno plast odločitvenih pravil - ti. pravila formalnosti opisov. Na podlagi teh bi se izvedla prilagoditev opisov elementov procesa glede na razvijalce, ki te elemente dejansko uporabljajo.

Na ta način bi lahko dosegli, da bi pristop za prilagajanje procesa postal še bolj prilagodljiv in bi posledično zagotavljal višjo raven prilagojenosti skonstruiranih različic osnovnega procesa.

5.4.4 Raziskava možnosti prenosa predlaganega pristopa na druge problemske domene

Pristop, ki ga predlaga doktorska disertacija, bi teoretično lahko uporabili pri obravnavanju širše skupine procesov, ki niso neposredno povezani s področjem razvoja aplikacij.

V okviru predlagane raziskave bi bilo potrebno ugotoviti, kako hitro je možno obstoječi odločitveni model za prilagajanje procesa razvoja IS prilagoditi novi domeni. Z raziskavo bi bilo potrebno:

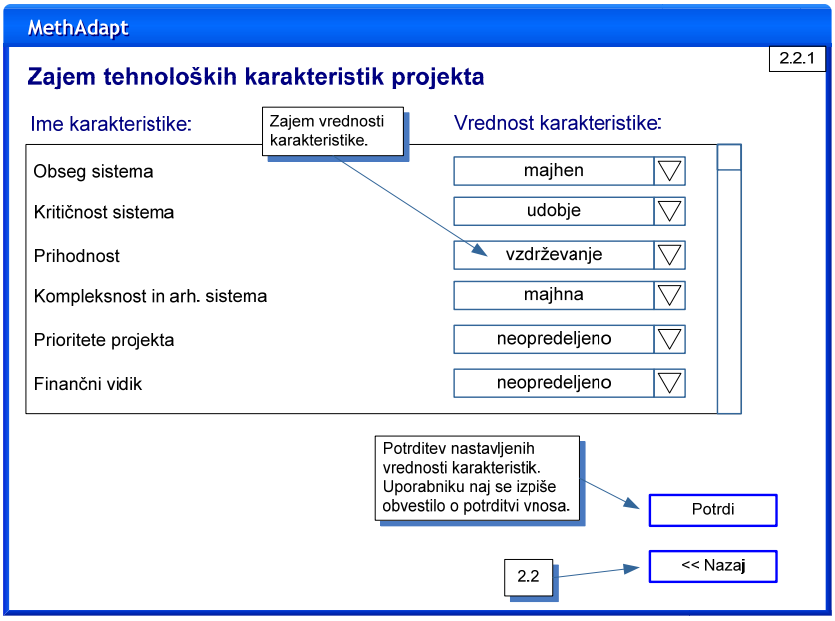
- opredeliti pristop za učinkovito identifikacijo novih oziroma dodatnih karakteristik, ki vplivajo na prilagajanje procesa v okviru nove domene,
- opredeliti pristop za čim hitrejši zajem znanja o novem procesu in
- identificirati postopke za konstruiranje novih odločitvenih pravil.

Z izsledki raziskave bi lahko opredelili izhodišča za izdelavo parametriziranega ogrodja odločitvenega modela, ki bi se lahko prilagajal zahtevam določene domene, na podlagi njenih karakteristik.

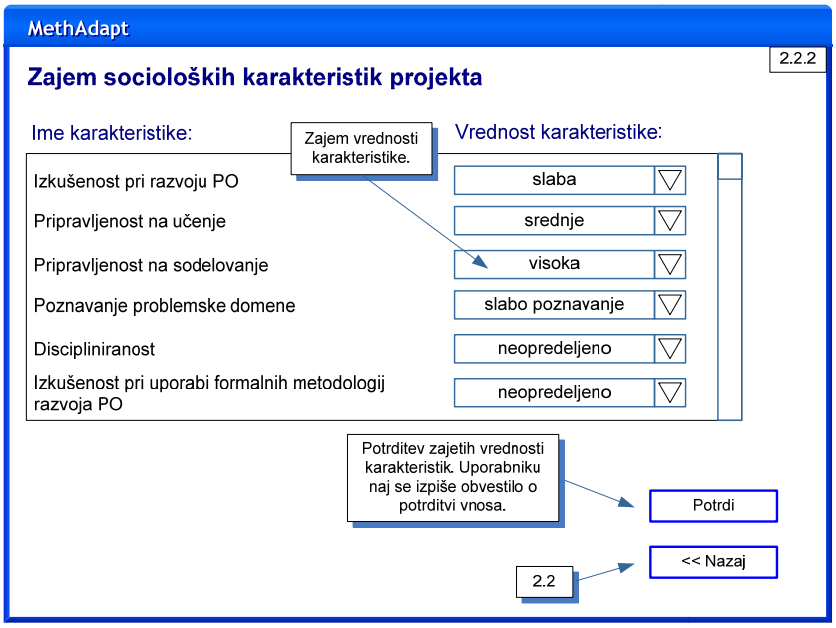
Priloga: Zaslonske maske modula MethAdapt

V prilogi podajamo specifikacije ključnih zaslonskih mask, ki omogočajo komunikacijo med uporabnikom - metodologom in programsko logiko modula MethAdapt in ki smo jih navedli tudi v okviru opisov primerov uporabe modula. Vsaka predstavljena zaslonska maska nosi v desnem zgornjem kotu identifikacijsko številko, ki opredeljuje njeno mesto v hierarhiji zaslonskih mask (glej razdelek 4.4.3).Elementi, ki se nahajajo na posameznih zaslonskih maskah so tudi dodatno obrazloženi.

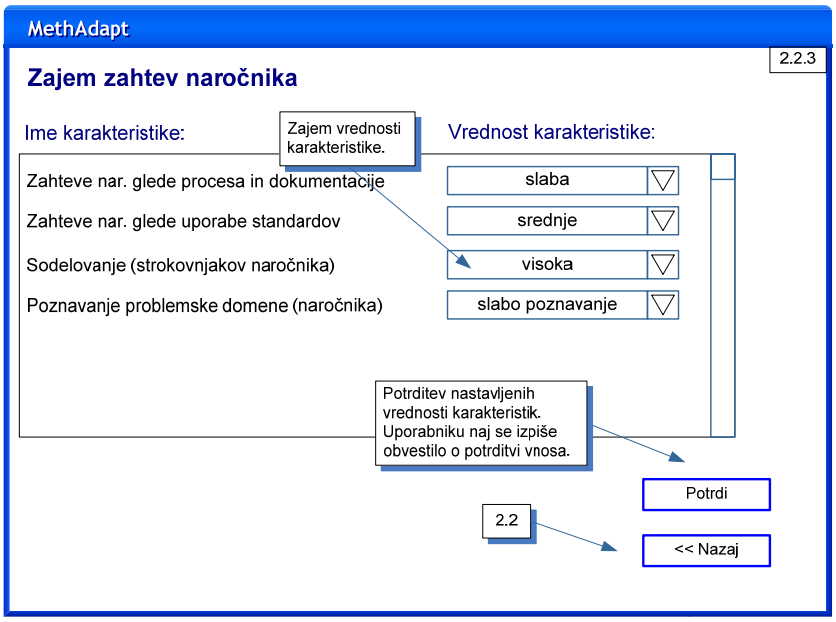
Oznaka zaslonske maske:2.2.1
Namen: Zaslonska maska za zajem tehnoloških karakteristik projekta.
Sklop: Prilagajanje procesa



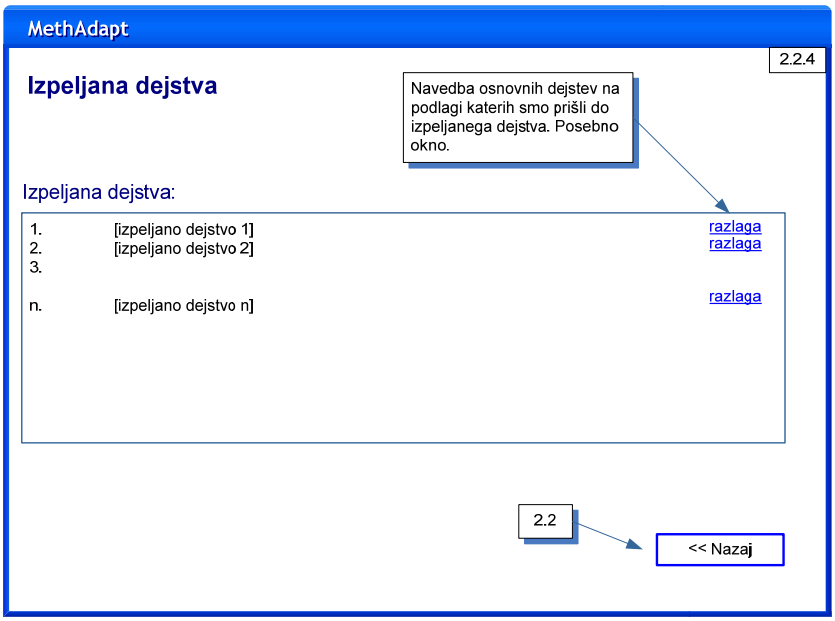
Oznaka zaslonske maske:2.2.2
Namen: Zaslonska maska za zajem sociooloških karakteristik projekta.
Sklop: Prilagajanje procesa



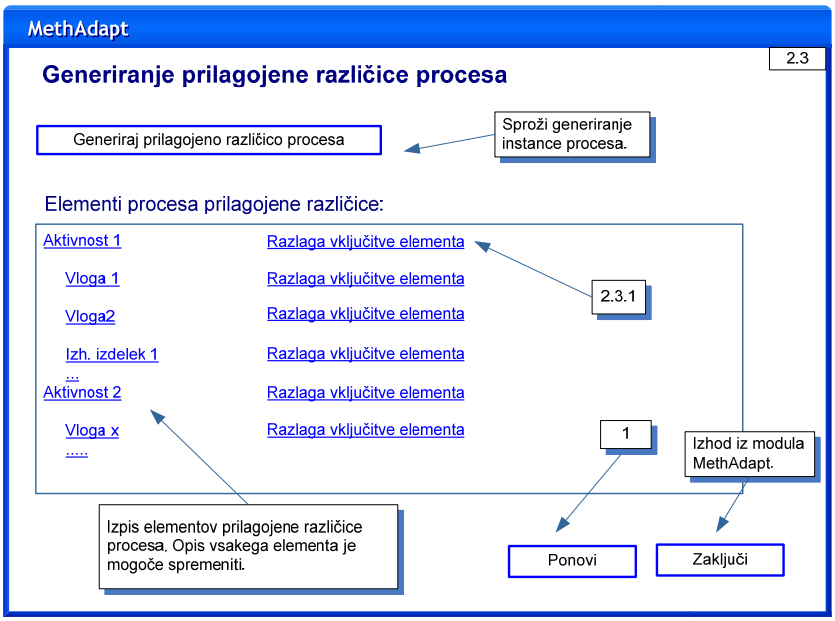
Oznaka zaslonske maske:2.2.3
Namen: Zaslonska maska za zajem zahtev naročnika projekta.
Sklop: Prilagajanje procesa



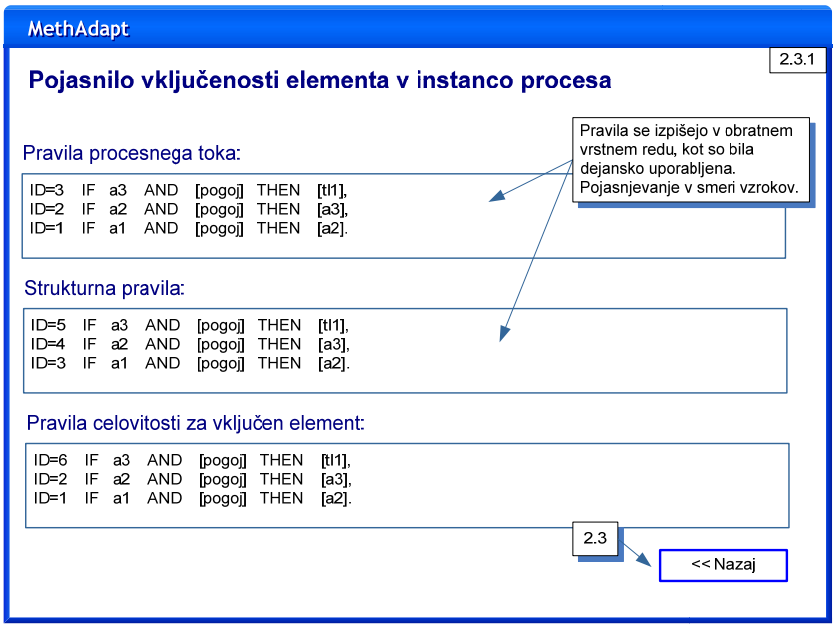
Oznaka zaslonske maske:2.2.4
Namen: Zaslonska maska za prikaz izpeljanih dejstev. Za vsako izpeljano dejstvo je možno ugotoviti, iz katerih osnovnih dejstev in pravil sklepanja je bilo pridobljeno.
Sklop: Prilagajanje procesa



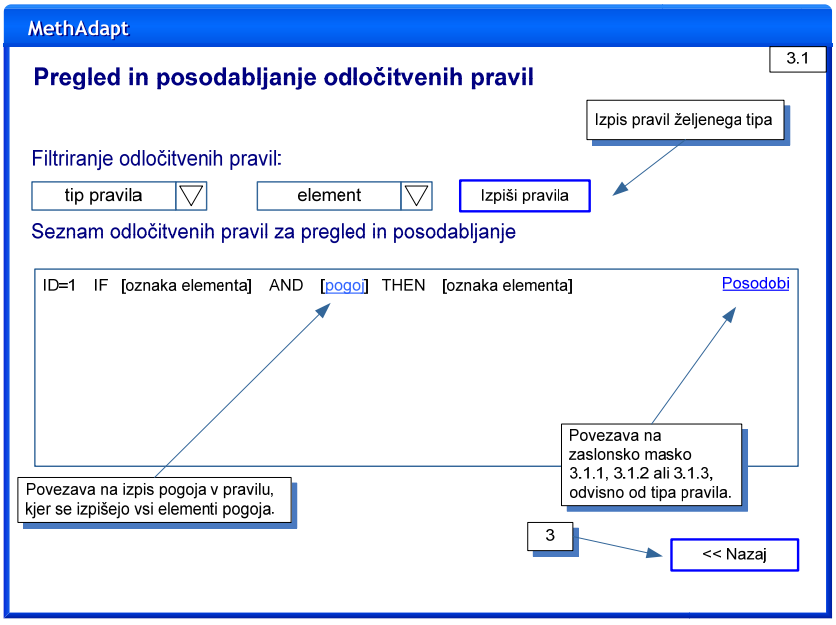
Oznaka zaslonske maske:2.3
Namen: Zaslonska maska za izvajanje prilagajanja procesa in izpis rezultata izvajanja. Za vsak element je možno pridobiti pojasnilo o vključitvi.
Sklop: Prilagajanje procesa



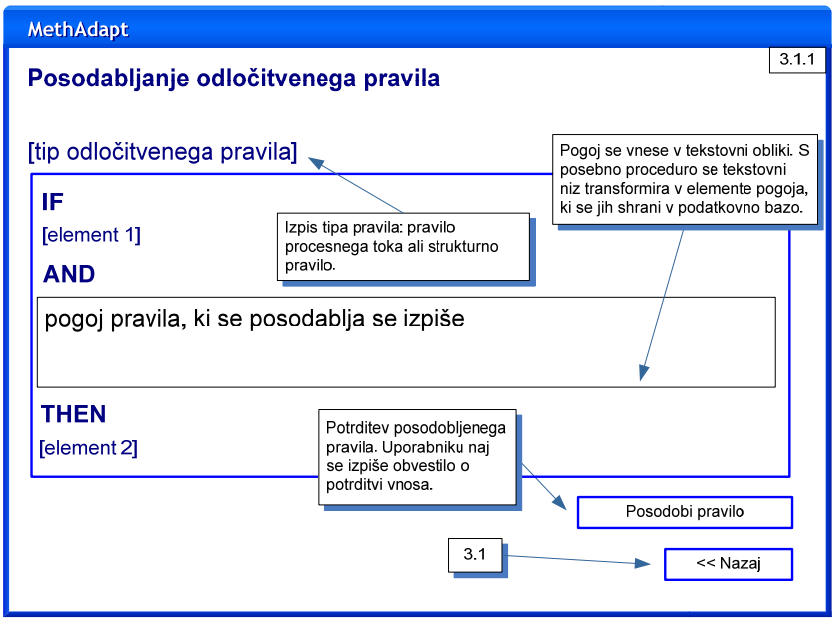
Oznaka zaslonske maske:2.3.1
Namen: Zaslonska maska za izpis odločitvenih pravil, ki pojasnjujejo kako je prišlo do vključitve elementa procesa v instanco procesa.
Sklop: Prilagajanje procesa



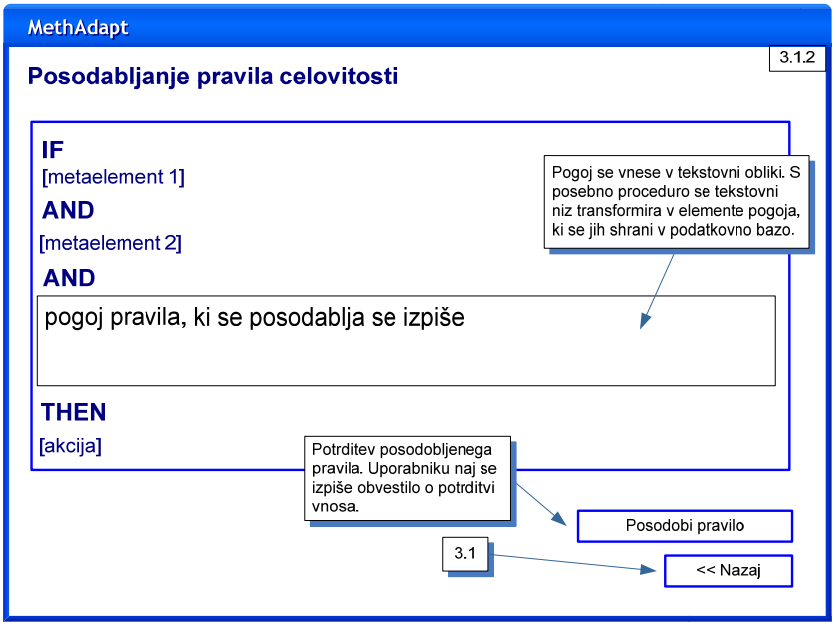
Oznaka zaslonske maske:3.1
Namen: Zaslonska maska omogoča pregledovanje in posodabljanje odločitvenih pravil.
Sklop: Ažuriranje odločitvenih pravil



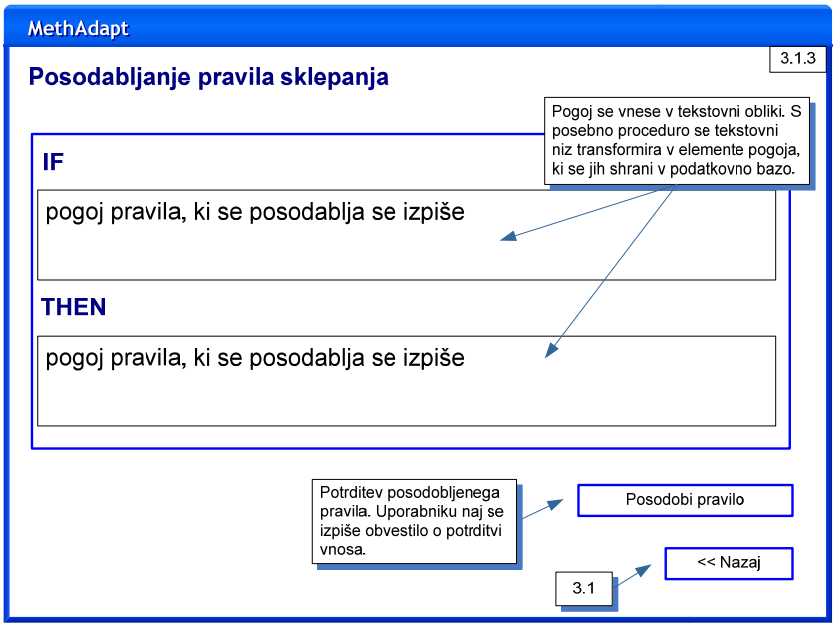
Oznaka zaslonske maske:3.1.1
Namen: Zaslonska maska za posodabljanje strukturnih pravil ali pravil procesnega toka.
Sklop: Ažuriranje odločitvenih pravil



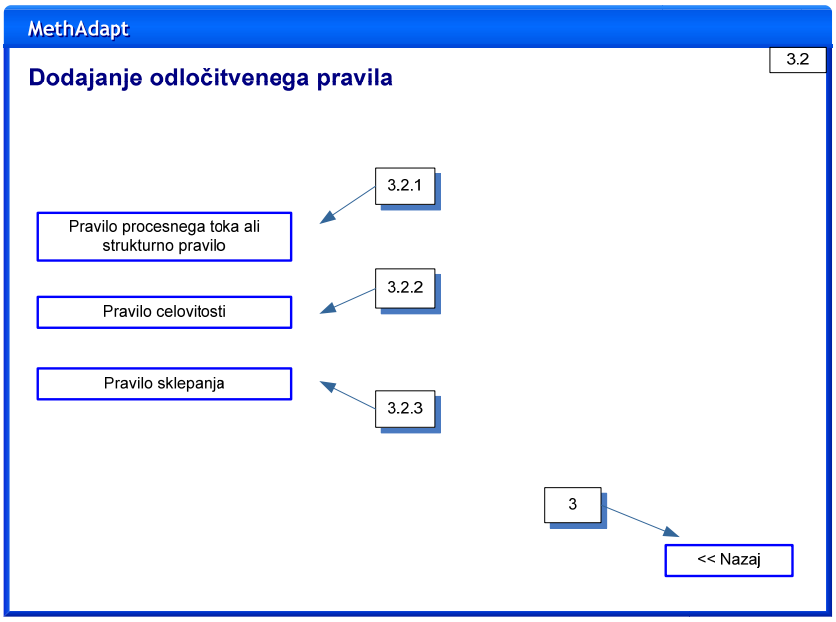
Oznaka zaslonske maske:3.1.2
Namen: Zaslonska maska za posodabljanje pravil celovitosti.
Sklop: Ažuriranje odločitvenih pravil



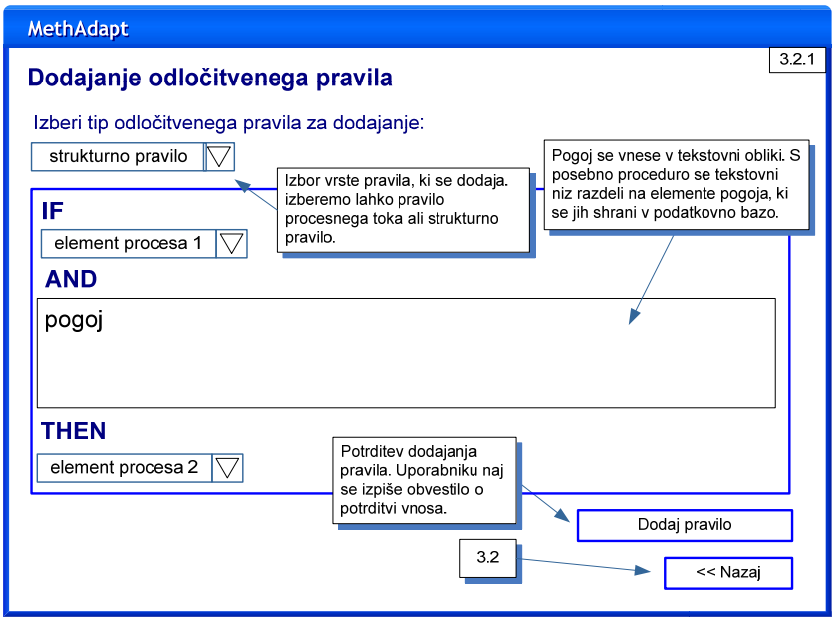
Oznaka zaslonske maske:3.1.3
Namen: Zaslonska maska za posodabljanje pravil sklepanja.
Sklop: Ažuriranje odločitvenih pravil



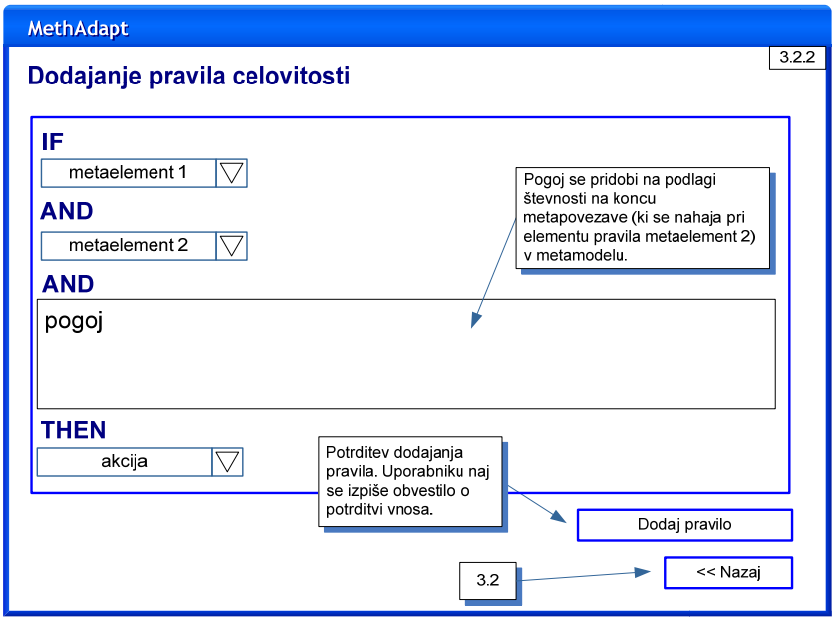
Oznaka zaslonske maske:3.2
Namen: Zaslonska maska za dodajanje odločitvenih pravil različnih tipov.
Sklop: Ažuriranje odločitvenih pravil



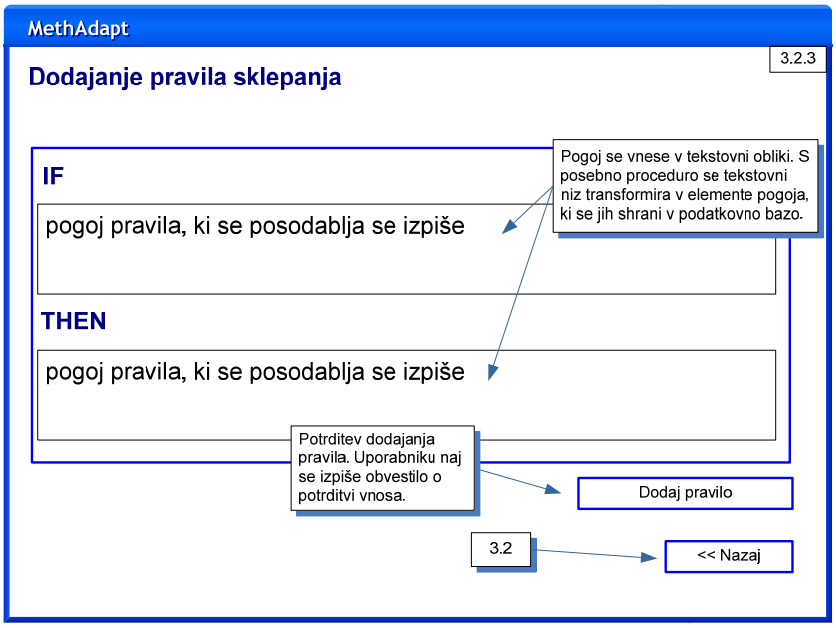
Oznaka zaslonske maske:3.2.1
Namen: Zaslonska maska omogoča dodajati v bazo pravil nova strukturna pravila in pravila procesnega toka.
Sklop: Ažuriranje odločitvenih pravil



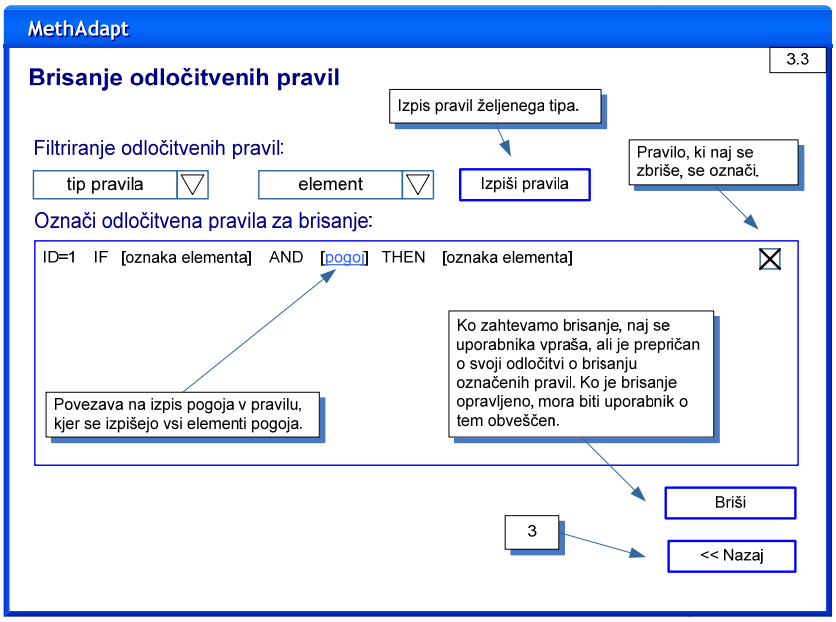
Oznaka zaslonske maske:3.2.2
Namen: Zaslonska maska omogoča dodajati v bazo pravil nova pravila celovitosti.
Sklop: Ažuriranje odločitvenih pravil



Oznaka zaslonske maske:3.2.3
Namen: Zaslonska maska omogoča dodajati v bazo pravil nova pravila sklepanja.
Sklop: Ažuriranje odločitvenih pravil



Oznaka zaslonske maske:3.3
Namen: Zaslonska maska omogoča brisanje obstoječih odločitvenih pravil v bazi pravil.
Sklop: Ažuriranje odločitvenih pravil



Pojmovnik

AMT	Agile Methodology Toolset. Programsko orodje, ki je namenjeno zajemu, prilagajanju in spremljanju tehnične in socialne ustreznosti metodologij razvoja IS.
BAZA ODLOČITVENIH PRAVIL (v okviru disertacije predlaganega pristopa)	Mehanizem za hranjenje odločitvenih pravil.
BAZA PRAVIL (v okviru sistema za izvajanje pravil)	Element sistema za izvajanje pravil, ki hrani znanje o neki domeni v obliki *if-then* pravil.
CAME	Computer Aided Method Engineering. Programsko orodje za podporo konstruiranja metodologij.
CASE	Computer Aided Software Engineering. Programsko orodja za podporo razvoja IS.
CLIPS	C Language Integrated Production System. Lupina ekspertnega sistema.
CMM	Capability Maturity Model. Model, ki opisuje nivoje zrelosti procesa razvoja programske opreme.
GRAF OSNOVNEGA PROCESA	Graf osnovnega procesa je mešan graf, ki predstavlja elemente osnovnega procesa in povezave med njimi. Predstavljen je s trojko $G = (V, E, A)$.
INSTANCA	Primerek.
IS	Informacijski sistem.

IT	Informacijska tehnologija.
JESS	Programsko okolje za izvajanje pravil napisano v programskem jeziku Java.
KONFLIKTNA MNOŽICA	Predstavlja podmnožico pravil - kandidatov, ki jih lahko sistem za izvajanje pravil v danem trenutku sproži. Govorimo o odločitvenih pravili, ki lahko ti."vžgejo".
KONFLIKTNA SITUACIJA	Konfliktna situacija je opredeljena s konfliktno množico pravil.
MDA	Model Driven Architecture. Modelno usmerjena arhitektura.
MDD	Model Driven Development. Modelno usmerjen razvoj aplikacij.
MEHANIZEM SKLEPANJA	Element sistema za izvajanje pravil, v okviru katerega se ugotavlja ujemanje med pogojnim delom pravila in dejstvi ter kreiranje konfliktne množice pravil.
METODA	Poseben način na katerega nekaj naredimo.
METODOLOGIJA	Skupek metod in principov za izvrševanje določene aktivnosti.
ODDALJENOST VOZLIŠČA	Število skokov od vozlišča tipa aktivnost do obravnavanega vozlišča v grafu osnovnega procesa.
PO	Programska oprema.
PREMISA	Pogojni del if-then stavka. Predstavlja pogojni del oziroma predpostavko v odločitvenem pravilu. Če je predpostavka resnična, potem lahko sklepamo na resničnost posledičnega dela odločitvenega pravila.

RAD	Rapid Application Development. Pospešen razvoj aplikacij.
SDLC	System Development Life Cycle. Življenski cikel razvoja programske opreme.
SKLEP	Posledični del if-then stavka oziroma posledični del odločitvenega pravila.
SME	Situational Method Engineering. Situacijsko konstruiranje metodologij.
SMSDM	Standard Metamodel for Software Development Methodologies. Standardni metamodel za metodologije razvoja programske opreme.
STRATEGIJA RAZREŠEVANJA KONFLIKTNE SITUACIJE	Strategija na temelju katere se iz konfliktne množice pravil izbere pravilo za vžig. Poznamo več strategij: strategija prvega pravila, strategija naključnega pravila, strategija najbolj specifičnega pravila, strategija najmanj uporabljanega pravila in strategija "najboljšega" pravila.
TACITNO ZNANJE	Skrito znanje, ki se nahaja v glavah zaposlenih.
UML	Unified Modelling Language. Enotni modelirni jezik.
ŽIVLJENSKI CIKEL	Določa zaporedje izvajanja aktivnosti in postopkov. Poznamo več življenjskih ciklov: slapovni, iterativni, inkrementalni, prototipiranje itd.

Literatura

[Acuna 1999] ACUNA, S., T., M. LOPEZ, N. JURISTO, A. MORENO (1999). A process model applicable to software engineering and knowledge engineering. International Journal of Software Engineering and Knowledge Engineering 9, 5 663-687

[Ambler 1998] AMBLER, S., W. (1998). Process patterns: Building Large-Scale Systems Using Object Technology, Cambridge University Press

[Ambler 2002] AMBLER, S. W. (2002). Agile modelling - Effective Practices for eXtreme Programming and the Unified Process. John Wiley & Sons, Inc., New York, ISBN: 0-471-20282-7

[Armour 2001] ARMOUR, P. G. (2001). Matching Process to Types of Teams. Communications of the ACM, Vol. 44, No. 7, pp. 21-23

[Arni-Bloch 2005] ARNI-BLOCH, N. (2005). Towards a CAME Tools for Situational Method Engineering. Interop-ESA 2005, Geneva 21-25 February 2005

[Avison 1999] AVISON, D., F. LAU, M. MYERS and P. A. NIELSEN (1999). Action Research. Communications of the ACM, january 1999, Vol. 42, No. 1

[Avison 2003a] AVISON, D. E. and G. FITZGERALD (2003). Where Now for Development Methodologies? Communications of the ACM, Vol. 46, No. 1, pp. 79-82

[Avison 2003b] AVISON, D. E., G. FITZGERALD (2003). Information Systems Development: Methodologies, Techniques and Tools. Third edition, McGraw-Hill Education, ISBN: 0-07-709626-6

[Bajec 2004a] BAJEC, M., M. KRISPER (2004). Izbrana načela agilnega načrtovanja. Zbornik posvetovanja, Dnevi slovenske informatike 2004, Portorož-slovenija, 14.-16. april

[Bajec 2004b] BAJEC, M., M. KRISPER (2004). Scenarij za načrtovanje, uvedbo in uporabo agilnih metodologij razvoja informacijskih sistemov. Zbornik posvetovanja, Dnevi slovenske informatike 2004, Portorož-slovenija, 14.-16. april

[Bajec 2004c] BAJEC, M., M. KRISPER (2004). The scenario for constructing flexible, people-focused systems development methodologies. The 12th European Conference on Information Systems, June 14-16. 2004, Turku Finland

[Bajec 2004d] BAJEC M., D. VAVPOTIČ, M. KRISPER (2004). The scenario and tool-support for constructing flexible, people-focused systems development methodologies. V: Thirteenth international conference on information systems development: Advances in Theory, Practice and Education. Vilna, Litva, 9. september 2004

[Bajec 2005a] BAJEC, M., D. VAVPOTIČ, M. KRISPER (2005). Uporaba ter socio-tehnična ustreznost metodologij razvoja informacijskih sistemov v slovenskih podjetjih. Zbornik oposvetovanja Dnevi slovenske informatike, Portorož 2005, ISBN 961-6165-18-6

[Bajec 2005b] BAJEC, M., D. VAVPOTIČ, M. KRISPER (2005). An approach for creating project-specific software development methodologies. IBIMA 2005

[Baskerville 1999] BASKERVILLE, R., L. (1999). Investigating Information Systems with Action Research. Communications of the Association for Information Systems, Vol. 2, october 1999

[Beck 2001] BECK, K., M. BEEDLE, A. VAN BENNEKUM, A. COCKBURN, W. CUNNINGHAM, M. FOWLER, J. GRENNING, J. HIGHSMITH, A. HUNT, R. JEFFRIES, J. KERN, B. MARICK, R. C. MARTIN, S. MELLOR, K. SCHWABER, J. SUTHERLAND, D. THOMAS (2001). Agile Manifesto, http://www.agilemanifesto.org/

[Becker 2002] BECKER-KORNSTAEDT, U. and R. REINERT (2002). A Concept to Support Process Model Maintenance through Systematic Experience Capture. Copyright 2002 ACM ISBN 1-58113-556-4/02/0700, pp. 465-468

[Bohanec 1995] BOHANEC, M., V. RAJKOVIČ (1995). Večparametrski odločitveni modeli, Organizacija 28, 1995, 427-438

[Brinkkemper 1996a] BRINKKEMPER, S. (1996). Method engineering: Engineering of information systems development methods and tools. Information and Software Technology, No. 38, Elsevier Science B.V., pp. 275-280

[Brinkkemper 1996b] BRINKKEMPER, S., K. LYYTINEN and R. J. WELKE (1996). Method Engineering - Principles of method construction and tool support. Proceedings of the IFIP TC8, WG 8.1/8.2 Working Conference of Method Engineering, 26-28 August 1996, Atlanta, USA, Chapman & Hall, ISBN: 041279750 X

[Brinkkemper 1999] BRINKKEMPER, S., M. SAEKI and F. HARMSEN (1999). Meta-modeling based assembly techniques for situational method engineering. Information Systems, Vol. 24, No. 3, pp. 209-228

[Budlong 1996] BUDLONG, F., P. SZULEWSKI, R. GANSKA (1996). Process Tailoring for Software Project Plans. The Software Technology Support Center, OO-ALC/TISE, Hill AFB, Utah

[Charvat 2003] CHARVAT., J. (2003). Project Management Methodologies-Selecting, implementing, and Supporting Methodologies and Process for Projects. John Wiley & Sons, Inc., Hoboken, New Jersey, ISBN: 0-471-22178-3

[Cockburn 1996] COCKBURN, A. (1996). The interaction of Social Issues and Software Architecture. Communications of the ACM, Vol. 39, No. 10, pp. 40-46

[Cockburn 1999] COCKBURN, A. (1999). A Methodology Per Project, http://alistair.cockburn.us

[Cockburn 2000a] COCKBURN, A. (2000). Selecting a Project's Methodology. IEEE Software, Vol. 17, No. 4, pp. 64-71

[Cockburn 2000b] COCKBURN, A. (2000). Just-In-Time Methodology Construction. Extreme Programming and Flexible Process conference, Sardinia, June 2000

[Cockburn 2000c] COCKBURN, A. (2000). Characterizing People as Non-Linear, First-Order Components in Software Development, 4th International Multi-Conference on Systems, Cybernetics, and Informatics, Orlando, FL, June 2000

[Cockburn 2002] COCKBURN, A. (2002). Agile Software Development. Addison-Wesley, Pearson Education 2002, 115, ISBN: 0201699699

[Curtis 1992] CURTIS, B., M. KELLNER, J. OVER (1992). Process modelling. Communication of the ACM 35, 9, 75-90

[DeMarco 1999] DEMARCO, T. and T. LISTER (1999). Peopleware: Productive Projects and Teams, 2nd Ed. Dorset House Publishing Company, Incorporated, 2nd edition (February 1, 1999), ISBN: 0932633439

[DeMarco 1999] DEMARCO, T. and T. LISTER (1999). Peopleware: Productive Projects and Teams, 2nd Ed. Dorset House Publishing Company, Incorporated, 2nd edition (February 1, 1999), ISBN 0932633439

[Deneckere 2001] DENECKERE., R. (2001). Approche d'extension de me´thodes fonde´e sur l'utilisation de composants ge´ne´riques, PhDthesis,University of Paris 1

[Domino 2002] DOMINO, M., A. HEVNER, R. W. COLLINS (2002). Applying Agile Software Development Processes to Global Virtual Teams: A Study of Communication Modalities. Copyright 2002 ACM ISBN 1-58113-466-5-02/05, pp. 76-78

[Dowson1988] DOWSON, M. (1988). Iteration in the Software Process. Proc 9th Int. Conf. on Software Engineering

[Eriksen 2000] ERIKSEN, L. (2000). Limitations and opportunities of system development methods in Web information system design. In Organizational and Social Perspectives on Information Technology. R. Baskerville, J. Stage, and J. DeGross, Eds. Kluwer, Boston

[Feiler 1993] FEILER, P., H., W. S. Humphrey (1993). Software Process Development and Enactment: Concepts and Definitions. Proc. 2nd Int. Conf. on "Software Process"

[Finkelstein 1994] FINKELSTEIN, A., J. KRAMER, B. NUSEIBEH (1994). Software Process Modelling and Technology. Research studies.

[Firesmith 2004] FIRESMITH, D. (2004). Creating a Project-Specific Requirements Engineering Process. Journal of Object Technology, Vol. 3, No. 5, May-June 2004

[Fitzgerald 1997] FITZGERALD, G., A. PHILIPPIDIS and P. PROBERT (1999). Information systems development, maintenance, and enhancement: Findings from a UK study. Int. J. Info. Mgmt. 20, 2 (Apr. 1999)

[Fitzgerald 1998] FITZGERALD, B. (1998). An empirical investigation into the adoption of system development methodologies. Information & Management 34, pp. 317-328

[Fitzgerald 1999] FITZGERALD, B. (1999). System Development Methodologies: The Problem of Tenses, Information Technology & People

[Fitzgerald 2003] FITZGERALD, B., N. L. RUSSO and T. O'KANE (2003). Software development Method Tailoring at Motorola. Communications of the ACM, Vol. 46, No. 4, pp. 65-70

[Garmus 1996] GARMUS, D. and D. HERRON (1996). Measuring the software process. Prentice-Hall, Inc., ISBN: 0-13-349002-5

[Gladden 1982] GLADDEN, G. R. (1982): Stop the life cycle, I want to get off. Software Engineering Notes, 7, 2, pp. 35-39.

[Glass 2002] GLASS, R.L., I. VESSLEY and V. RAMESH (2002). Research in software engineering: an analysis of the literature. Information and Software Technology, No. 44, pp. 491-506

[Hardy 1995] HARDY, C., J. THOMPSON, H. EDWARDS (1995). The use, limitations and customization of structured systems development methods in the United Kingdom. Information and Software Technology, 37 (9), pp. 467-477

[Harmsen 1994] HARMSEN, F., S. BRINKKEMPER, H. OEI (1994). Situational Method Engineering for information system project approaches, Methods and associated tools for the information systems life cycle, Amsterdam, 169-194

[Harmsen 1996] HARMSEN, F., M. SAEKI (1996). Comparison of four Method Engineering languages. Proceedings of the IFIP TC8, WG 8.1/8.2 Working Conference of Method Engineering, 26-28 August 1996, Atlanta, USA, Chapman & Hall, ISBN: 041279750 X

[Henderson 2003] HENDERSON-SELLERS, B. (2003). Method engineering for OO systems development. Communications of the ACM, Vol. 46, No. 10, pp. 73-78

[Henderson 2005] HENDERSON-SELLERS, B. (2005). Agent-Oriented Methodologies. Idea Group Publishing, ISBN 1-59140-581-5

[Henderson 2005] HENDERSON-SELLERS, B., C. GONZALEZ-PEREZ (2005). A comparison of four metamodels and the creation of a new generic standard. Information and software technology 47(1), 49-65

[Highsmith 2002a] HIGHSMITH, J. and Cutter Consortium (2002). What is Agile Software Development? CrossTalk: The journal of Defence Software Engineering, October 2002

[Highsmith 2002b] HIGHSMITH, J. A. (2000). Adaptive Software Development - A collaborative approach to managing complex systems. Dorset House Publishing Co., Inc, ISBN: 0-932633-40-4

[Highsmith 2002c] HIGHSMITH, J. (2002). Agile Software Development Ecosystems. Addison Wesley, May 26, 2002, ISBN: 0-201-76043-6

[Hirch 2002] HIRSCH, M. (2002). Making RUP Agile. Conference on Object Oriented Programming Systems Languages and Applications, OOPSLA 2002 Practitioners Reports, pp. 1-8

[Hofstede 1997a] HOFSTEDE, A. and T.F. VERHOEF (1997). On the feasibility of situational method engineering. Information Systems, Vol. 22, No. 6/7, pp. 401-422

[Hofstede 1997b] HOFSTEDE, G. (1997). Cultures and organizations-Software of the mind. McGraw-Hill, ISBN: 0-07-029307-4

[Hofstede 1998] HOFSTEDE, A.H.M. and H.A. PROPER (1998). How to formalize it? Formalization principles for information system development methods. Information and Software Technology, No. 40, pp. 519-540

[Hofstede 2001] HOFSTEDE, G. (2001). Culture's Consequences - Comparing values, behaviours, institutions, and organizations across nations - 2nd ed. Sage Publications, Inc., ISBN: 0-8039-7323-3

[Huisman 2003] HUISMAN, M., J. IVARI (2003). Systems Development Methodology Use in South Africa. Ninth Americas Conference on Information Systems

[Huismann 2002] HUISMAN, M., J. LIVARI (2002). The individual deployment of systems development methodologies, Information Systems Development Conference, Riga

[Karlson 2004] KARLSON, F., P. J. AGERFALK (2004). Method configuration: adapting to situational characteristics while creating reusable assets. Information and Software technology, Vol. x, No. x, Article in press

[Kelly 1996] KELLY, S., K. LYYTINEN, M. ROSSI (1996). MetaEdit+: A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment. In: Proceedings of the 8th International Conference on Advanced Information Systems Engineering (CAiSE'96), Heraklion, Crete, Greece, Constantopoulos et al. (eds.), , LNCS 1080, Springer-Verlag (1996), 1-21

[Krisper 2004a] KRISPER, M., RUPNIK, R., ROŽANEC, A., BAJEC, M., OSOJNIK, R., TOMAŽIČ, R., SILIČ, M. (ur.). (2003). Enotna metodologija razvoja informacijskih sistemov. [Zv. 2], Strateško planiranje. 2. izd. Ljubljana: Vlada Republike Slovenije, Center Vlade RS za informatiko, 446 str., ilustr. ISBN 961-6389-08-4. [COBISS.SI-ID 128167168]

[Krisper 2004b] KRISPER, M., RUPNIK, R., ZRNEC, A., BAJEC, M., OSOJNIK, R., TOMAŽIČ, R., SILIČ, M. (ur.). (2003). Enotna metodologija razvoja informacijskih sistemov. [Zv. 3], Strukturni razvoj. 2. izd. Ljubljana: Vlada Republike Slovenije, Center Vlade RS za informatiko, 446 str., ilustr.

[Kumar 1992] KUMAR, K., R. WELKE (1992). Methodology engineering: A proposal for situation-specific methodology construction, Challenges and Strategies for Research in Systems Development, John Wiley & Sons, 257-268

[Maddison 1983] MADDISON, R.N. (1983). Information System Methodologies. Wiley Heyden, Chichester, UK

[Maier 2000] MAIER, M., E. RECHTIN (2000). The Art of Systems Architecting. 2nd Edition, CRC Press, Boca Raton, FL

[McChesney 1991] McCHESNEY, I., R. (1991). Toward a classification scheme for software process modelling approaches. Information and Software Technology 37, 7, 363-374

[McConnell 1996] McCONNELL, S. (1996). Rapid development: Timing Wild Software Schedules, Microsoft Press

[Merritt 2000] MERRITT, D. (2000). Building Expert Systems in Prolog. Amzi! inc. 5861 Greentree Road Lebanon, OH 45036 U.S.A.

[Mihelčič 1999] MIHELČIČ, M. (1999). Organizacija in ravnateljevanje. Založba FE in FRI, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani

[Mihelčič 2000] MIHELČIČ, M. (2000). Poslovne funkcije. Založba FE in FRI, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani

[Miller 2001] MILLER, G. (2001). Sizing Up Today's Lightweight Software Processes. IT Professional, Institute of Electrical and Electronics Engineers

[Mirbel 2003] MIRBEL, I. (2003). A polymorphic context frame to support scalability and evolvability of information system development process. Project MECOSI, Rapport de recherche, ISRN 13S/RR-2003-23-FR, October 2003

[Mirbel 2004] MIRBEL, I. (2004). Rethinking ISD Methods: Fitting project team members profiles. Project EXeCO, Rapport de recherche, ISRN 13S/RR-2004-13-FR, April 2004

[Mohamed 2004] MOHAMED, B., A., J. RALYTE, C. ROLLAND (2001). Constructing the Lyee method with a method engineering approach, Knowledge-Based Systems 17, 239–248

[Morabito 2001] MORABITO, J., I. SACK and A. BHATE (2001). Organisation Modelling, Innovative Architectures for the 21st Century. NJ: Prentice Hall

[Nuseibeh 1996] KARLSON, F., P. J. AGERFALK (2004). Method configuration: adapting to situational characteristics while creating reusable assets. Information and Software technology, Vol. x, No. x, Article in press

[O. Agency 1995] The Object Agency (1995). Comparison of Object-Oriented Development Methodologies, Object Agency inc., 1995

[Odell 1996] ODELL, J., J. (1996). A primer to method engineering, Method engineering : Principles of method construction and tool support, Chapman&Hall, 1-7

[O'Dell 1996] O'DELL, C., C. Jr. GRAYSON (1996). If Only We Knew What We Know: The Transfer of Internal Knowledge and Best Practice. The Free Press, New York, NY, 1996.

[Olson 2003/2004] OLSON, S. J., G. M. OLSON (2003/2004). Culture Surprises in Remote Software Development Teams. Queue, Vol. 1, No. 9, pp. 52-59

[Oxford 2000] Oxford Advanced Learner's Dictionary of Current English (2000). Sixth edition, Electronic edition on CD-ROM, Oxford University Press, 2000

[Paige 1997] PAIGE, F. R. (1997). A Meta-Method for Formal Method Integration. Proc. Formal Methods Europe '97, LNCS 1313, Springer-Verlag

[Paige 1999] PAIGE, F. R. (1999). When are methods complementary? Information and Software Technology, No. 41, pp. 57-162

[Pfleeger 1998] PFLEEGER, S-L. (1998). Software Engineering: Theory and Practice. Prentice-hall

[Plihon 1996] PLIHON, V. (1996). Un environnement pour l'inge´nierie des me´thodes, PhD thesis, University of Paris 1—Sorbonne

[Prakash 1997] PRAKASH, N. (1997). Towards a Formal Definition of Methods. Requirements Eng, Vol. 2. Springer-Verlag, 23-50

[Prakash 1999] PRAKASH, N. (1999). On Method Statics and Dynamics. Information Systems. Vol.34, No.8

[Ralyte 2001] RALYTE, J., C. ROLLAND (2001). An Assembly Process Model for Method Engineering. Proceedings of the 13th CAISE01, Interlaken, Switzerland

[Ralyte 2003] RALYTE, J., R. DENECKERE and C. ROLLAND (2003). Towards a Generic Model for Situational Method Engineering. Proceedings of CAISE03, 15th International Conference on Advanced Information Systems Engineering, Klagenfurt/Velden, Autriche

[Rolland 1993] ROLLAND, C. (1993). Modeling the Requirements Engineering Process, 3rd European-Japanese Seminar on Information Modelling and Knowledge Bases, Budapest, Hungary

[Rolland 1998a] ROLLAND, C., V. PLIHON, J. RALYTE (1998). Specifying the Reuse Context of Scenario Method Chunks, Proceedings of the 10th International Conference on Advanced Information System Engineering (CAISE98), Pisa, Italy

[Rolland 1998b] ROLLAND, C. (1998). A Comprehensive View of Process Engineering. Proceedings of the 10th International Conference CAiSE'98, B. Lecture Notes in Computer Science 1413, Pernici, C. Thanos (Eds), Springer. Pisa, Italy

[Rossi 2000] ROSSI, M., J. TOLVANEN, B. RAMESH, K. LYYTINEN and J KAIPALA (2000). Method Rationale in Method Engineering. Proceedings of the 33rd Hawaii International Conference on System Sciences - 2000, pp. 1-10

[Schwartz 2004] SCHWARTZ, R. B. and M. C. RUSSO (2004). How to Quickly Find Articles in the Top IS Journals. Communications of the ACM, Vol. 47, No. 2, pp. 98-101

[Singh 2003] SINGH, S. and P. KOTZE (2003). An Overview of Systems Design and Development Methodologies with Regard to the Involvement of Users and Other Stakeholders. Proceedings of SAICSIT 2003, University of South Africa, pp. 37-47

[Smith 1993] SMITH, H. A., J. D. McKEEN (1993). Re-engineering the Corporation: Where Does I.S. Fit In: Proceedings of the 26 th Hawaii International Conference on Systems Science, (eds. J.F. Nunamaker, R.H. Sprague), Vol. 3., IEEE Computer Society Press, USA

[Smolander 1992] SMOLANDER, K. (1992). A Model for Modeling Systems Development Methods. In: Next Generation CASE Tools (eds. K. Lyytinen, V.-P. Tahvanainen) IOS Press, Amsterdam, Netherlands, pp. 224-239

[Song 1995] SONG, X. (1995). A Framework for Understanding the Integration of Design Methodologies. Software Engineering Notes, Vol. 20, No. 1, pp. 46-54

[SPC 1992] SPC (1992). Process Definition and Modeling Guidebook. Software Productivity Consortium, SPC-92041-CMC

[Theunissen 2003] THEUNISSEN, W., H., M., G. K. DERRICK, B. W. WATSON (2003). Standards and Agile Software Development. ESPRESSO Research Group. Department of Computer Science, University of Pretoria

[Tolvanen 1996] TOLVANEN, J. P., M. ROSSI, H. LIU (1996). Method engineering: Current research directions and implications for future research. In: IFIP TC8 Conference on Method Engineering, Chapman & Hall

[Tolvanen 1998] TOLVANEN, J. (1998). Incremental Method Engineering with Modeling Tools - Theoretical Principles and Empirical Evidence. PhD thesis. University of Jyvaskyla, ISBN: 951-39-0303-6

[Vasconcelos 1997] VASCONCELOS, F., M., C. M. L. WERNER (1997). Software development process reuse based on patterns. Proceedings of the 9th International Conference on Software Engineering and Knowledge Engineering, 97-104

[Vavpotič 2002] VAVPOTIČ, D., M. BAJEC, M. KRISPER (2002). Characteristics of software development methodology evaluation model. Business information technology management: facilitating global IS alliances, Bitworld'2002, Guayaquil, Ecuador (2002)

[Vavpotič 2003] VAVPOTIČ, D. (2003). Metodologije razvoja programske opreme in njihova izbira. Magistrsko delo. Fakulteta za računalništvo in informatiko, Univerza v Ljubljani

[Vavpotič 2005] VAVPOTIČ, D. (2005). Opredelitev izhodišč za merjenje in izboljševanje tehnične učinkovitosti in socialne sprejetosti metodologij razvoja programske opreme. Doktorska disertacija v nastajanju. Fakulteta za računalništvo in informatiko, Univerza v Ljubljani

[Weinberg 1998] WEINBERG, G. (1998). The Psychology of Computer Programming. Dorset House Publishing Company, Incorporated, Silver Ann edition (September 1998), ISBN 0932633420

[West 2000] WEST, D. B. (2000). Introduction to Graph Theory. Prentice Hall; 2 edition (August 23, 2000), ISBN: 0130144002

[Yourdon 1992] YOURDON, E. (1992). The Decline and Fall of the American Programmer. Prentice-Hall Englewood Cliffs, NJ.

[Zrnec 2004a] ZRNEC, A., M. KRISPER (2004). Definition of the formal decision model and tool support for adapting software development process to particular project circumstances. Doctoral consortium, Thirteenth International Conference On Information Systems Development, Advances in Theory, Practice and Education, September 9-11, 2004, Vilna, Litva, članek je sprejet na posvetovanje

[Zrnec 2004b] ZRNEC, A., D. VAVPOTIČ, R. RUPNIK, M. BAJEC, M. KRISPER (2004). Določitev metodologije za razvojne projekte informacijskih sistemov v državni upravi na podlagi metodologije EMRIS. INDO 2004, Portorož 2004, 13. -15. september

Pomembnejši spletni naslovi

Na spodnjem seznamu se nahaja nekaj pomembnejših spletnih naslovov, na katerih se nahajajo gradiva, relevantna za temo doktorske disertacije.

AIdepot	Introduction to Rule-Based Systems, [http://ai-depot.com]
CLIPS	A Tool for Building Expert Systems, [http://www.ghg.net/clips/CLIPS.html]
ILOG	ILOG Inc., [http://www.ilog.com]
Jess	The Rule Engine for the JavaTM Platform, [http://herzberg.ca.sandia.gov/jess/]
Mejabi 1997	MEJABI, O. O. and J. J. BLACK (1997). Process Management: The New Frontier For Continuous Improvement And Total Quality, [http://www.simplexsystems.com/ProcessMgt.htm]

Izjava

Podpisani Aljaž Zrnec izjavljam, da sem doktorsko nalogo izdelal samostojno, pod vodstvom mentorja doc.dr.Marjana Krisperja.

Ljubljana, 2.11.2005 Aljaž Zrnec

Zahvala

Zahvala gre doc. dr. Marjanu Krisperju, ki mi je kot mentor nudil strokovno pomoč, nasvete in ideje v zvezi s širšo tematiko doktorske disertacije. Zahvaljujem se mu tudi za vso izkazano spodbudo in pripravljenost za pomoč v kritičnih trenutkih.

Zahvaljujem se tudi izr. prof. dr. Viljanu Mahniču za izkazano pomoč pri izdelavi doktorske disertacije.

Prav tako se zahvaljujem doc. dr. Marku Bajcu za izkazano pomoč in številne koristne nasvete pri izdelavi doktorske disertacije ter konstruktivne pripombe glede vsebine.

Zahvaljujem se svoji ženi Mariji, ki mi je ves čas doktorskega študija stala ob strani in me bodrila ter sinu Tilnu za vso potrpežljivost. Še enkrat, hvala!

Zahvaljujem se tudi staršem, ki so mi v času študija pomagali najti še zadnje ure časa, ki sem ga lahko posvetil delu v okviru disertacije.

Hkrati se zahvaljujem še vsem ostalim, ki so kakorkoli prispevali k nastanku tega dela.