Agilne metodologije razvoja informacijskih sistemov

Marko Bajec, Marjan Krisper
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
Tržaška 25, 1000 Ljubljana
marko.bajec@fri.uni-lj.si, marjan.krisper@fri.uni-lj.si

Povzetek

Pojem agilnost in trendi, ki jih s tem povezujemo, so se na področju metodologij razvoja informacijskih sistemov pojavili kot rezultat kritičnega pogleda na celovite, večinoma težko obvladljive metodologije, ki proces razvoja predpišejo do vsake najmanjše podrobnosti. Kljub kakovosti, ki jo takšne metodologije navadno dosegajo, je njihova uporaba v praksi omejena, saj jim z željo po čimprejšnjih rezultatih težko sledimo. Novi trendi priporočajo zasnovo in uporabo prilagodljivih metodologij, ki so »ravno dovolj« predpisljive ter hitro in enostavno prilagodljive. V prispevku opredelimo pojem agilnost in opišemo z njo povezane trende. V nadaljevanju podamo naše izkušnje pri uporabi novih trendov za zasnovo sodobnih metodologij razvoja informacijskih sistemov.

Abstract

Agile Information System Development Methodologies

The term "agile" and trends that are related to it have emerged in the IS community as a result to shortcomings of the complex methodologies which prescribe each development step in its most detail. In spite of reputation such methodologies gain in theory their practical use is only limited. Today's markets are extremely dynamic and call for rapid results which make the use of heavy methodologies difficult. New trends recommend the use of agile methodologies that lead into so called adaptable software development. In the paper we describe the concept of an agile methodology and discuss our experiences in developing a contemporary methodology based on agile-oriented values, principles and practices.

1 UVOD

Prve elaborirane metodologije razvoja informacijskih sistemov so nastale že v zgodnjih sedemdesetih letih, ko so na področju razvoja programske opreme kraljevali Codd, DeMarco, Yourdon, Cox, Booch, Orr, Parnas in drugi. S tem, ko so uspeli prepričati strokovno javnost, da je potrebno tudi za področje razvoja programske opreme urediti postopke, predpisati metode, tehnike in orodja, so ovrgli smiselnost uporabe ad hoc pristopov in postavili temelje za obdobje birokracije, kot ga danes nekateri imenujejo (7). Kasneje so bile namreč razvite in dokumentirane številne metodologije, bodisi na podlagi izkušenj ali teoretičnih izhodišč. Z željo po standardizaciji postopkov in izdelkov so nastali priročniki, ki so tudi na več tisoč straneh do potankosti predpisovali vsak najmanjši korak razvoja, za vsako aktivnost so bili predpisani številnimi dokumenti in formularji, za njihovo izvedbo pa so bila določena orodja in navodila za uporabo.

Breme, ki ga nosijo udeleženci, če morajo pri projektu slediti smernicam kompleksnih metodologij, je veliko, saj morajo poleg svojega primarnega dela skrbeti tudi za številne izdelke in naloge, ki se vsaj na videz zdijo sekundarnega pomena. Do pred kratkim je veljalo, da je takšen pristop pač potreben, če želimo razvijati sisteme, ki bodo učinkoviti in jih bo moč obvladovati tudi čez več let. Če ne bi bili izpostavljeni spremembam ter konkurenci pri razvoju programske opreme, bi morda tudi danes trdili, da je to edina možnost. Prav zaradi dinamike in konkurence, ki zahtevata hitre rezultate, se danes uveljavlja drugačen pogled na obravnavano problematiko – metodološke smernice se vračajo k ad hoc pristopom.

V prispevku opisujemo temeljna izhodišča nove šole mišljenja, ki jo danes radi povezujemo s pojmom *agilnost*. V slovenskem jeziku temu pravimo gibčnost ali prilagodljivost. Najprej se posvetimo metodologijam na splošno, s čimer želimo poudariti, da je metodologija več kot le skupek metod in tehnik. Sledi poglavje, kjer najprej opišemo ozadje nastanka trenda agilnih metodologij ter temeljna načela in priporočila, ki v zvezi s tem izhajajo iz literature ali temeljijo na neposredni izmenjavi mnenj največjih glasnikov novih pristopov. Naš pogled in razumevanje agilnih metodologij podamo v tretjem poglavju, kjer predstavimo

tudi izkušnje, ki smo jih pridobili z zasnovo in uvajanjem agilne metodologije na enem izmed slovenskih računalniških podjetij.

2 KAJ JE METODOLOGIJA IN ZAKAJ JO POTREBUJEMO?

2.1 Metodologija ima pomembno sociološko komponento

Ko govorimo o metodologijah, pogosto ne vemo dobro, kaj sploh je metodologija. Ni res, da številna podjetja pri svojem delu ne uporabljajo nobene metodologije, saj je ta prisotna pri vsakem organiziranem delu. Metodologija zajema vse, kar redno počnemo, da bi dosegli želen rezultat, torej izdelek ali storitev, ki je cilj našega dela. V primeru razvoja programske opreme to ne pomeni zgolj postopkov, ki so neposredno povezani z razvojem (npr. analiza, načrtovanje ipd.), temveč zajema tudi podporne postopke, načine komunikacije med sodelujočimi, pravila odločanja itn. Metodologijo lahko opredelimo tudi kot množico dogovorov (konvencij), s katerimi se projektna skupina/organizacija strinja (1).

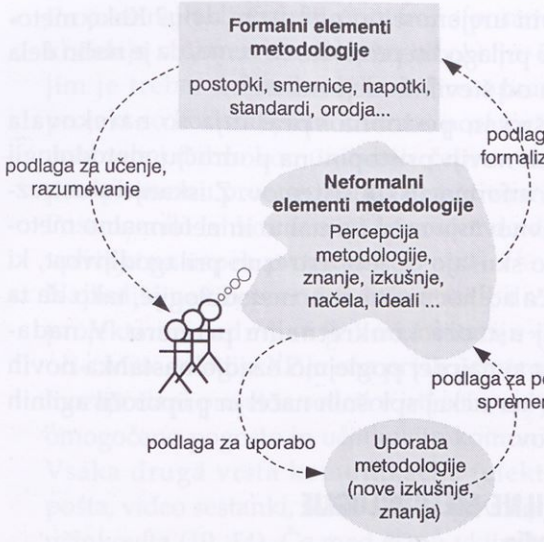
Metodologija je prežeta s filozofijo, načeli, idejami in pogledi organizacije in njenih članov, kar še posebej poudarja njeno sociološko komponento. Metodologija namreč ne nastane neodvisno od ljudi, katerim je namenjena. Čeprav obstajajo številne formalne metodologije, ki temeljijo na preizkušenih postopkih, ki so se v praksi izkazali kot dobri, si jih organizacije, ko jih privzamejo, vedno prilagodijo, tako da ustrezajo njihovemu načinu dela ter percepciji domene, za katero so vzpostavljene. Z uporabo metodologije se uporabniki učijo in pridobivajo nove izkušnje, s čimer se posledično bogati tudi metodologija sama.

2.2 Formalni in neformalni elementi metodologije

Metodologija zajema poleg formalno opredeljenih elementov, ki so navadno zapisani v obliki postopkov, pravil, napotkov, smernic, standardov itn. v navadnih ali elektronskih priročnikih, tudi številne nedokumentirane elemente. Posebno mesto med njimi nosi znanje, ki ga člani organizacije uporabljajo pri svojem delu. To je za organizacijo ključnega pomena, saj predstavlja tiste vrednote, ki so organizaciji lastne in so njena konkurenčno vrednost. Posledično velja to tudi za metodologijo. Temeljni elementi metodologije se skrivajo ravno v znanju in izkušnjah posameznikov.

V znanosti se z upravljanjem z znanjem ukvarja posebna veja (angl. *Knowledge Management*). Njen namen je iskanje tehnik in metod za zajem, formalizacijo in prenos znanja. Kot nas ugotovitve iz omenjenega področja učijo, se znanje deli na *eksplicitno* in *skrito* znanje (12). Eksplicitno znanje (angl. *Explicit Knowledge*) je moč izraziti v formalni obliki, skrito znanje (angl. *Tacit Knowledge*) pa je navadno subjektivne narave in prepleteno z izkušnjami, ideali, čustvi, intuicijo ipd., zaradi česar ga je težko izraziti. Vendar pa se lahko skrito znanje sčasoma rutinizira in tako postane izrazljivo. Omenjeni pojav ima tudi na področju metodologij svoje mesto. Dokumentirana metodologija je sprva podlaga, iz katere se uporabniki učijo in razvijejo svojo percepcijo o načinu dela, komunikaciji, odločanju ipd. Z uporabo pridobivajo nove izkušnje in bogatijo svoje znanje, kar vpliva tudi na metodologijo. Ko postane znanje, ki ga uporabljajo pri delu, dovolj rutinsko, pridobi obliko podatkov in tedaj lahko postane del formalne metodologije (9). Z uporabo se metodologija bogati s skritim znanjem posameznikov. Iz njega se lahko sčasoma izkristalizirajo formalne oblike (metode, postopki, smernice, napotki) ter postanejo del formalne metodologije. Pri tem je treba upoštevati, da je skrito znanje, ki je podlaga za nastanek eksplicitnega znanja, vedno bogatejše od stvaritve same (11, 13).

Slika 1 prikazuje cikel, ki poteka od uporabe formalne metodologije kot podlage za učenje ter



Slika 1: **Od neformalne do formalne metodologije**

razumevanje metodologije do formalizacije elementov neformalne metodologije, ki se izkažejo dovolj rutinski za standardizacijo.

Številna računalniška podjetja še vedno razvijajo programsko opremo na podlagi neformalno opredeljenih metodologij, ki niso dokumentirane. Čeprav so postopki znani in ustaljeni, jih podjetja redko eksplicitno zapišejo, še bolj poredko pa svoje metodologije osvežujejo v skladu s pridobljenimi izkušnjami in znanjem.

Meje med formalnim in neformalnim delom metodologije ni preprosto določiti. Če je formalni del metodologije obsežen, ga težko vzdržujemo, kar ima za posledico zastaranost metodologije in neustreznost dejanskemu stanju. Metodologija je namreč dinamična in se stalno spreminja. Obsežna neformalna metodologija pa je tvegana za organizacijo, saj je popolnoma odvisna od njenih uporabnikov. Potreba po formalizaciji metodologije izhaja že iz dejstva, da je razvoj programske opreme sistematičen proces, ki mora biti ustrezno zasnovan in dokumentiran, da lahko učinkovito usmerja posameznike in skupine k dobrim rezultatom. Če je metodologija opredeljena zgolj na neformalni ravni, je razvoj težko standardizirati, postopek pa postane nepregleden in neobvladljiv. Kako podrobno naj bo torej metodologija dokumentirana? Kateri elementi metodologije so najbolj stabilni in zato smiselni za formalizacijo? Kako zagotoviti, da bo metodologija enostavna in prilagodljiva, po drugi strani pa dovolj rigorozna, da bo od uporabnikov zahtevala sistematiko in urejenost pri njihovem delu? Kako metodologijo prilagoditi projektu, če vemo, da je način dela odvisen od številnih dejavnikov?

Takšna in podobna vprašanja so narekovala nastanek novih pristopov na področju metodologij razvoja informacijskih sistemov. Z iskanjem ustreznega ravnovesja med formalno in neformalno metodologijo skušajo doseči ustrezno prilagodljivost, ki omogoča ad hoc nastavitev metodologije, tako da ta čimbolj ustreza konkretnemu primeru. V nadaljevanju si najprej poglejmo ozadje nastanka novih trendov ter nekaj splošnih načel in priporočil agilnih pristopov.

3 AGILNE METODOLOGIJE

3.1 Ozadje

Temelji agilnih metodologij so bili postavljeni decembra 2001, ko se je sestala skupina sedemnajstih *gurujev*, tako ali drugače znanih s področja *lahkih* metodologij (*Adaptive Software Development, XP-Extreme Programming, Feature-Driven Development, Crystal, Scrum, Dynamic System Development Method* idr.). Namen sestanka je bil ugotoviti, kaj imajo »njihove« metodologije skupnega in na osnovi tega postaviti skupne metodološke osnove. Čeprav trdijo, da njihov namen ni enotna lahka metodologija – ULM *Unified Light Methodology* (spomnimo se, da je pred desetimi leti šlo za podobno situacijo, ko so se zaradi poplave objektno usmerjenih metod pomembni posamezniki združili in dali temelje jeziku UML *Unified Modeling Language*), pa gre za podoben primer. Tokrat so pod drobnogledom lahke metodologije, katerih število je v zadnjih letih močno naraslo. Njihova temeljna lastnost je učinkovitost in prilagodljivost, s čimer je skupina sedemnajstih, združena v zvezo *Agile Alliance (1)*, opisala tudi nov trend agilnih metodologij.

3.2 Temeljna načela agilnih metodologij

V skupni izjavi so člani zveze kot svoj cilj zapisali iskanje boljših pristopov razvoja programske opreme, tako da pri tem sami sodelujejo in pomagajo drugim. Iz tega so izpeljali štiri načela:

- Posamezniki in njihova komunikacija so pomembnejši kot sam proces in orodja,
- Delujoča programska oprema je pomembnejša kot popolna dokumentacija,
- Vključevanje (sodelovanje) uporabnika je pomembnejše kot pogajanje na podlagi pogodb
- Upoštevanje sprememb je pomembnejše od sledenja planu.

Proces, orodja, dokumentacija, pogodbe in plani so vsekakor pomembni elementi metodologije, vendar pa je v primeru, ko je treba stvari postaviti na tehtnico, pomembneje poskrbeti za posameznike, sodelovanje, delujočo programsko opremo, vključevanje uporabnika in reagiranje na spremembe.

Poglejmo vsako izmed načel podrobneje.

3.2.1 Posamezniki in komunikacija vs. proces in orodja

Več pozornosti je treba nameniti članom projekta, njihovemu znanju in tudi željam. Toga porazdelitev vlog, ki jih določa proces, med člane projekta, ni smiselna, če ni ustreznih ljudi. Načelo poudarja tudi pomen komunikacije, ki je ključna za uspešnost projekta. Boljše rezultate dosežemo, če sta komunikacija in sodelovanje med člani projekta dobra,

čeprav se ne držijo predpisanega procesa, kot pa v primeru, ko je proces predpisan, komunikacija med člani pa slaba.

3.2.2 Delujoča programska oprema vs. popolna dokumentacija

Delujoč program je največ, kar lahko dobi končni uporabnik. Dokumentirane zahteve, model statičnih elementov sistema, model interakcij in druga dokumentacija o problemski domeni in sistemu so koristni, vendar drugotnega pomena. Naročnik ne bo zadovoljen s kupom dokumentacije, če ne bo najprej videl delujočega sistema. Ena večjih kritik *slapovnega* procesa je, da naročnik do konca projekta čaka na sleherno komponento programske opreme. Vse, kar lahko medtem dobi, je dokumentacija. Pod drugi strani pa je dokumentacija temeljnega pomena, saj olajša vzdrževanje sistema, komunikacijo med izvajalcem in naročnikom, preučevanje kakovosti dela projektne skupine itn. Zato je pomembno, da pripravljamo ustrezno dokumentacijo. Vsak del dokumentacije mora biti utemeljen.

3.2.3 Sodelovanje uporabnika vs. pogajanje na podlagi pogodb

S tretjim načelom je pozornost posvečena razmerju med naročnikom in izvajalcem, ki je pogosto preveč formalen in strog. Za uspešnost projekta je ključnega pomena, da se naročnik in izvajalec dobro ujameta. Naročnik najbolje ve, kaj potrebuje, čeprav tega pogosto ne zna razložiti. V agilno usmerjenih metodologijah je zato vloga naročnika postavljena na pomembno mesto. Medtem ko lahko dober odnos med naročnikom in izvajalcem olajša tehnološko še tako zahteven projekt, lahko strog pogodbeni odnos in nerazumevanje med naročnikom in izvajalcem zamaje tudi preproste projekte.

3.2.4 Reagiranje na spremembe vs. sledenje planu

Spremembe so eden od razlogov, ki ga pogosto povezujemo z neuspešnimi projekti. Številni avtorji ugotavljajo pomen obvladovanja sprememb in temu sledijo tudi moderne metodologije. Dinamika poslovnih okolij botruje številnim spremembam, zato je naivno pričakovati, da bomo lahko v začetnih fazah projekta zajeli vse zahteve. Projektni plani so koristen element, vendar morajo omogočati spremembe, vsaj v smislu preureditve prioritet znotraj dogovorjenega okvirja. Planiranje in sledenje planu je koristno, vendar le dokler se to ne razlikuje preveč od dejanskega stanja. Najslabše je slediti planu, ki je zastarel.

3.3 Priporočila

Iz temeljnih načel agilnih metodologij so izpeljana številna priporočila v pomoč pri gradnji in vrednotenju metodologij. Vse metodologije, ki se upravičeno deklarirajo kot agilne, jih upoštevajo. Naj jih naštejemo le nekaj:

- Najvišja prioriteta je zadovoljstvo stranke, zato je priporočeno zgodnje in pogosto izdajanje komponent preizkušene in delujoče programske opreme. To je temelj prilagodljivega razvoja, saj na podlagi medprojektnih izdaj pridobimo koristne povratne informacije, tveganje, da bo izdelek neustrezen, pa zmanjšamo.
- Delujoče komponente programske opreme je treba izdajati pogosto, v ciklu, ki ni daljši od štirih mesecev. Če je naročnik zmožen sprejemati izdelke v krajših razmakih in je tudi razvojna ekipa sposobna upoštevati vse sprotne spremembe, so krajši cikli še boljši. Iterativni pristop je nujen.
- Spremembe v zahtevah naj bodo dobrodošle tudi v poznih fazah razvoja. Eno izmed načel agilnih pristopov je dober odnos med naročnikom in izvajalcem. Spremembe, ki nastanejo med projektom, in ki smo jih pripravljeni upoštevati, lahko prinesejo naročniku pomembno konkurenčno prednost, zato se jih ne smemo otepati. Iterativni pristop je tudi tu primeren.
- Projekti naj vključujejo motivirane posameznike, ki delajo v ustreznem delovnem okolju. Zaupati jim je treba, da bodo delo dobro opravili. Za projekt so koristnejši motivirani posamezniki, ki med seboj dobro komunicirajo, čeprav ne sledijo predpisanemu procesu, kot pa nemotivirani ljudje. Posamezniki lahko s svojim odnosom do dela zelo vplivajo na uspeh projekta.
- Najboljši način prenosa informacij do članov projekta in med njimi je komunikacija »iz oči v oči«. Metodologija XP je ena prvih priporočala, da morajo člani projekta sedeti v isti sobi, saj je s tem omogočeno pogosto in učinkovito komuniciranje. Vsaka druga vrsta komunikacije (elektronska pošta, video sestanki, telefonski pogovori) je manj učinkovita (10, 14). Če med člane vključimo tudi predstavnike naročnika, je za odgovor potrebno malo časa.

- Kakovost programske kode in arhitekture je potrebno stalno preverjati in izboljševati. S tem povečamo prilagodljivost. Pristop, kjer najprej izpeljemo poglobljeno analizo, izdelamo dober načrt in nato razvijemo programsko kodo, je priporočljiv, vendar si ga pogosto ne moremo privoščiti. Zaradi potrebe po hitrih rezultatih je kakovost zapostavljena, zato je pomembno, da si v vsakem ciklu prizadevamo za izboljšave.
- Enostavnost procesa je ključ do uspeha. Kako razviti kakovostno programsko opremo, pri tem pa opraviti čim manj stranskih nalog in izdelkov? Proces mora biti ravno dovolj zahteven. Raje malo manj kot pa preveč. To ne pomeni, da moramo aktivnosti izvajati površno ali pomanjkljivo. Nasprotno, delo mora biti opravljeno kvalitetno, paziti pa moramo, da ni po nepotrebnem kompliciranо. Pot do enostavnosti je težka, saj je včasih preprosteje pustiti stvari kompleksne, kot jih pa poenostaviti.
- Po vsakem intervalu (iteraciji) moramo preveriti proces, ki smo ga uporabljali. Glede na ugotovitve proces prilagodimo in izboljšamo za prihodnje iteracije.

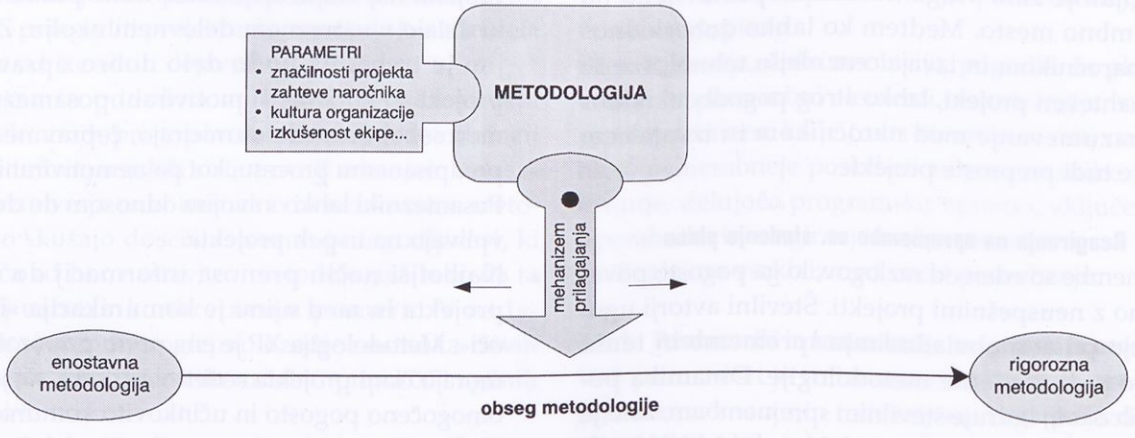
4 IZKUŠNJE PRI ZASNOVI IN UVEDBI AGILNE METODOLOGIJE

4.1 Razumevanje koncepta agilne metodologije

Načela in priporočila, opisana v prejšnjem razdelku, zajemajo le nekaj skupnih smernic, ki so se oblikovale v krogu zagovornikov agilnih pristopov. V precej širšem kontekstu obravnavata nove trende Cockburn in Highsmith v svojih knjigah (5, 7), kjer se ukvarjata z metodologijo razvoja kot celoto in podajata številna zanimiva razmišljanja, ki potrjujejo potrebo po prilagodljivem načinu razvoja programske opreme. Za razliko od njiju se Ambler in nekateri drugi avtorji v svojih delih posvečajo predvsem modeliranju kot ključnemu elementu za dosego učinkovitega razvoja programske opreme (npr. 2, 3, 4, 6). Po njihovem mnenju je modeliranje kritični element metodologije razvoja programske opreme, ki zahteva največjo prilagodljivost.

Iz raznih virov prihajajo tudi številne druge ideje, ki včasih odražajo subjektivna mnenja. To seveda povzroča določene nejasnosti, kar se kaže tudi v številu metodologij, ki jih danes avtorji deklarirajo kot agilne. Še posebej velja poudariti, da agilne metodologije niso nujno lahke metodologije in obratno, niso vse lahke metodologije tudi agilne. Prilagodljivost je pravzaprav popolnoma neodvisna od teže metodologije[1]. Tako lahka kot težka metodologija je lahko agilna, če omogoča ad hoc prilagajanje obsega metodologije dejanskim potrebam projekta.

Na spodnji sliki je koncept agilne metodologije predstavljen grafično. Agilna metodologija se s pomočjo mehanizma za prilagajanje v odvisnosti od parametrov, kot so značilnosti projekta, zahteve naročnika, kultura organizacije, izkušenost ekipe ipd., pozicionira na ustrezno mesto na skali, ki opredeljuje obseg in s tem tudi težo metodologije.



Slika 2: **Koncept agilne metodologije**

[1] *V (1) je teža metodologije opredeljena kot zmnožek obsega in gostote metodologije. Obseg metodologije pove, koliko različnih elementov, tj. aktivnosti, vlog, izdelkov, priporočil itd. zajema metodologija. Gostota pa se nanaša na raven podrobnosti, s katero so elementi opisani. Metodologija, kot je RUP, velja za težko metodologijo, saj zajema in podrobno opisuje številne elemente. Med tipične predstavnike lahkih metodologij sodijo Extreme Programming, Feature-Driven Development, Crystal Clear ipd.*

V nadaljevanju podajamo izkušnje, ki smo jih pridobili z izdelavo in uvajanjem agilne metodologije razvoja programske opreme v enem od slovenskih računalniških podjetij.

4.2 Prilagoditev referenčnega procesa

Podjetje, s katerim smo sodelovali, se je za formalizacijo procesa razvoja odločilo predvsem zaradi možnosti uvajanja novih članov, zaradi lažjega pridobivanja naročnikov ter zaradi potrebe po standardizaciji dokumentacije. Metodologija, ki se je uporabljala v podjetju, je bila precej stihijska in prepuščena posameznim izvajalcem, nova metodologija pa naj bi temeljila na dokumentiranem procesu, ki bi bil enostaven, učinkovit in prilagodljiv potrebam posameznih projektov.

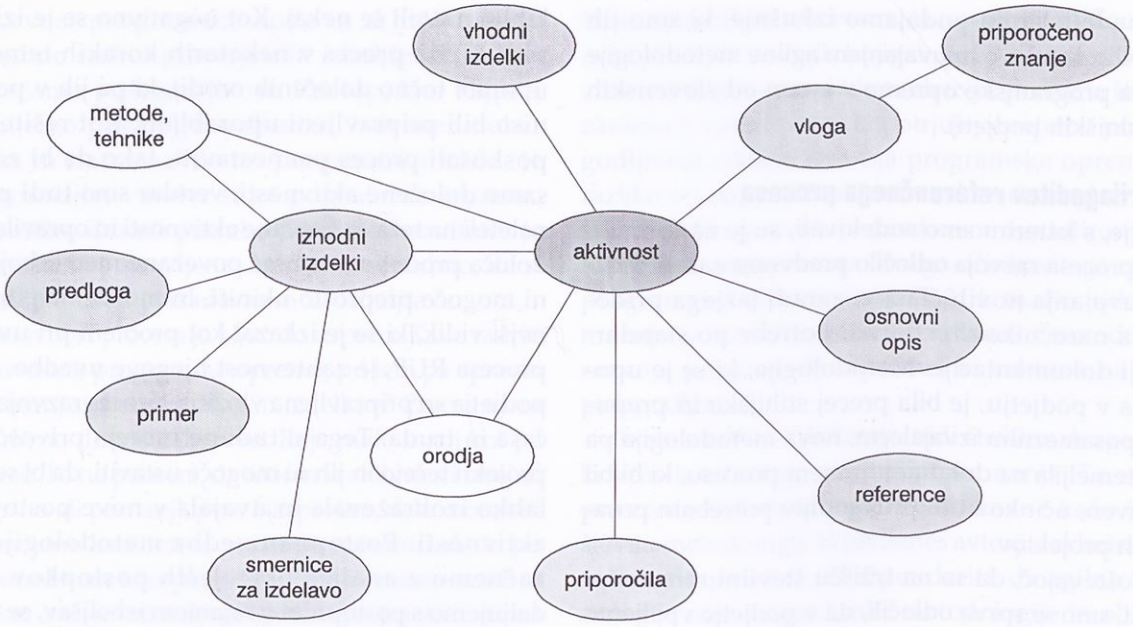
Upoštevajoč, da so na tržišču številni referenčni procesi, smo se sprva odločili, da v podjetje vpeljemo enega izmed preizkušenih modelov, nato pa ga prilagodimo, tako da bo ustrezal zahtevam podjetja. Pri izbiri ustreznih referenčnih procesov smo si pomagali z modelom, ki ga razvijamo v laboratoriju za informatiko v sklopu raziskovalne naloge (15, 16). Model temelji na predpostavkah o korelacijah med karakteristikami projektov in karakteristikami metodologij. Tako smo upoštevali karakteristike tipičnih projektov, ki jih podjetje izvaja, preferenčne želje podjetja (objektno usmerjen razvoj, iterativni življenjski cikel, lahka metodologija idr.) ter znanja in izkušnje posameznikov s področja razvoja programske opreme. Po izvedeni analizi rezultatov smo se odločili za vpeljavo procesa RUP (Rational Unified Process (8)), ki se je tudi v okviru uporabe modela za izbiro metodologije izkazal kot eden primernejših procesov s področja objektnega razvoja.

V prvem koraku smo izvedli izobraževanje. Izpeljali smo več tečajev, s katerimi smo uporabnikom predstavili podrobnosti procesa, obenem pa jih skušali naučiti, kako lahko proces uporabljajo pri svojem delu. Čeprav so uporabniki proces sprejeli kot dober, ga niso bili pripravljeni uporabljati v praksi, saj se jim je zdel preveč zahteven, pa tudi nepotreben. Eden takih korakov je na primer izdelava analize in načrta arhitekture, ki je po RUP precej zahtevna naloga. Arhitekturno ogrodje, ki so ga v podjetju razvili na osnovi večletnih izkušenj, namreč že samo po sebi določa postopek opredelitve arhitekture, kar izključuje številne podrobnosti, ki jih RUP predpisuje v sklopu aktivnosti za arhitekte. In takih korakov bi lahko našteli še nekaj. Kot negativno se je izkazalo tudi to, da proces v nekaterih korakih temelji na uporabi točno določenih orodij, ki pa jih v podjetju niso bili pripravljeni uporabljati. Kot rešitev smo poskušali proces poenostaviti, tako da bi zajemal samo določene aktivnosti, vendar smo tudi pri tem naleteli na težave. Številne aktivnosti in opravila, ki jih določa proces, so namreč povezane med seboj, in jih ni mogoče preprosto ukiniti. In morda najpomembnejši vidik, ki se je izkazal kot problem pri uvajanju procesa RUP, je zahtevnost njegove uvedbe. Redka podjetja so pripravljena vložiti v proces razvoja toliko časa in truda. Tega si tudi ne morejo privoščiti, saj projekti tečejo in jih ni mogoče ustaviti, da bi se ekipa lahko izobraževala in uvajala v nove postopke in aktivnosti. Postopna uvedba metodologije, kjer začnemo z analizo obstoječih postopkov in nadaljujemo s postopnim uvajanjem izboljšav, se nam je zdela primernejša.

4.3 Zajem osnovnega procesa z analizo obstoječe metodologije

Iz pogovorov smo kasneje ugotovili, da so zaposleni zadovoljni z mnogimi aktivnostmi, ki so jih izvajali pri razvoju programske opreme in da ne čutijo potrebe, da bi jih morali ukini ali spremeniti. Zato se je zdelo smiselno, da najprej preučimo obstoječo metodologijo in določimo njene osnovne elemente. Od uporabnikov smo skušali izvedeti, kaj radi počnejo in kaj se jim zdi odveč, katere dele dokumentacije izdelajo in zakaj, v katerih korakih imajo največ težav s komunikacijo, kje čutijo pomanjkljivosti v procesu, kaj bi spremenili, kaj obdržali itn. Pregledali smo dokumentacijo, ki nastaja pri projektih, preverili orodja ipd. Na podlagi ugotovitev smo nato dokumentirali osnovne korake metodologije. Pri opisu smo se držali priporočil agilnih pristopov in med osnovne elemente metodologije zajeli le tiste, ki so se zdeli dovolj stabilni in zato smiselni za formalizacijo. Izbrane elemente prikazuje slika 3.

Metodologijo smo predstavili grafično ter opisali njene gradnike. Ker je namen metodologije usmerjati sodelujoče na projektu, ki nastopajo v določenih vlogah, smo opise organizirali po vlogah. Osnovne vloge smo določili na podlagi lastnosti tipičnih projektov, ki jih v podjetju izvajajo, in upoštevali znanje in izkušnje zaposlenih. Za vsako vlogo smo zapisali priporočeno znanje. Aktivnosti vlog smo le kratko opisali, saj je treba upoštevati, da se dejanska metodologija pogosto



Slika 3: **Osnovni elementi opisa metodologije**

spreminja, kar otežuje vzdrževanje dolgih in podrobnih opisov. Za potrebe uvajanja novih članov smo dodali reference na podrobnejše opise (knjige, spletni naslovi).

Pomembna opisna elementa metodologije sta vhod in izhod aktivnosti. Vsak izdelek, ki nastane kot stranski ali končni izdelek aktivnosti, smo kratko opisali ter podali primer. Če je šlo za dokument, smo zaradi standardizacije dokumentacije pripravili tudi predlogo zanj, vendar smo pri tem upoštevali, da se uporabnik veliko več nauči iz primera kot iz predloge. Podrobnim opisom metod in tehnik ter orodij, ki naj se uporabljajo za izvedbo aktivnosti oz. za izdelavo izdelka, smo se izogibali, razen v primerih, ko je bilo to potrebno zaradi odvisnosti z drugimi aktivnostmi. Samo poznavanje tehnik in orodij namreč še ne zagotavlja, da bomo znali učinkovito izvesti aktivnost. Poleg tega v podjetju orodja in tehnike pogosto menjajo, odnos posameznikov do uporabe predpisanih tehnik in tudi orodij pa je negativen. Veliko večji pomen smo pripisali izkušnjam, ki so jih imeli posamezniki z izvajanjem določenih aktivnosti. Uporabnike smo izzvali, da smernice za izdelavo posameznih izdelkov ter priporočila za izvedbo aktivnosti zapišejo sami. S tem smo dosegli dvoje: (1) uporabniki se niso čutili ogrožene – uporabniki, še posebej tisti, ki so že izkušeni in prepričani, da svoje delo opravljajo dobro, ne vidijo radi, da se jih uči, (2) zajeli smo pomembno znanje, ki bi sicer ostalo le pri posameznikih.

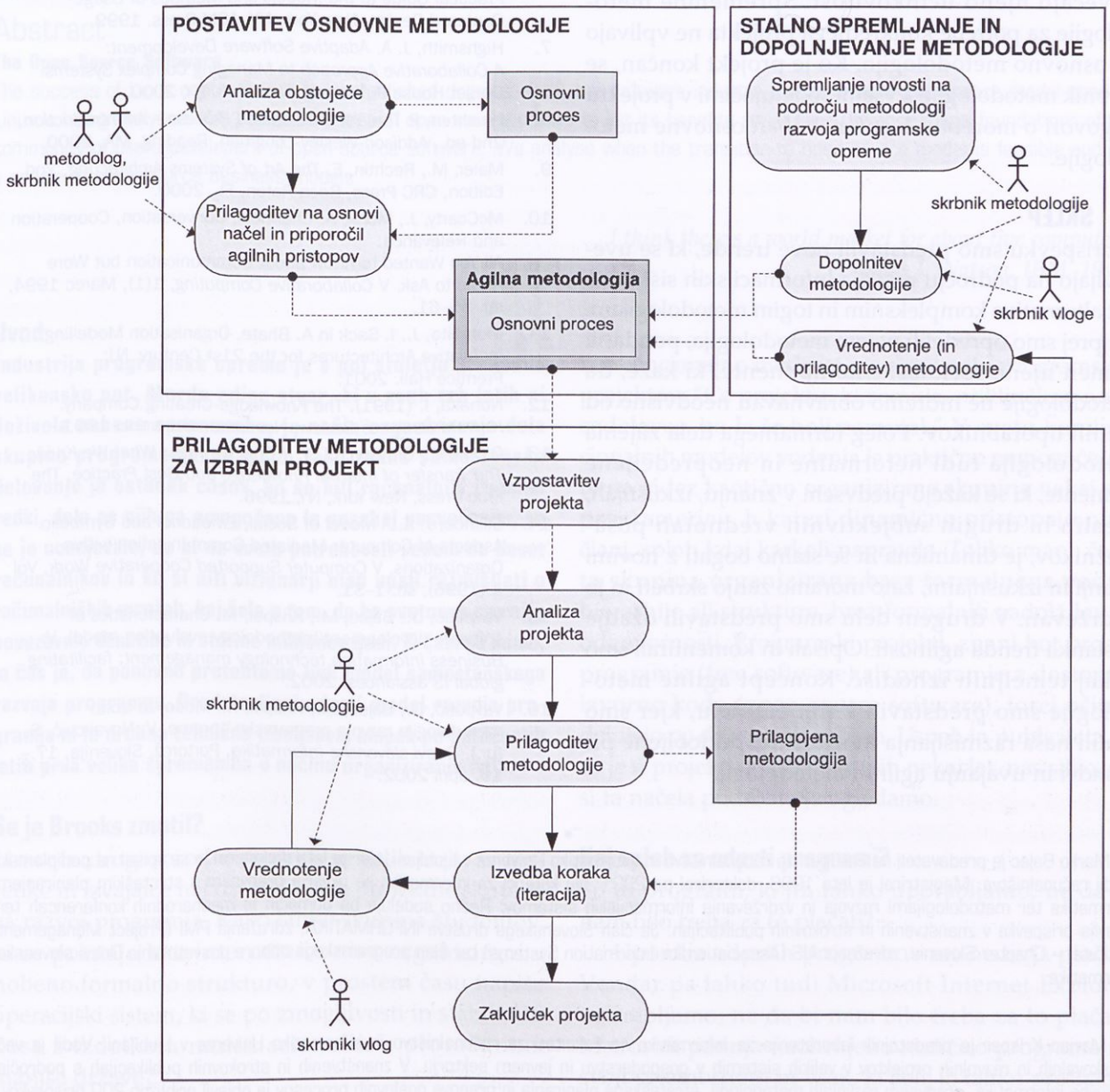
Grafično predstavitev ter opise osnovnih elementov metodologije smo zajeli s preprosto spletno aplikacijo, ki uporabnikom omogoča enostaven dostop (potrebujejo zgolj internetni brskalnik) ter dodajanje smernic in priporočil na podlagi izkušenj. Menimo, da je takšna predstavitev metodologije nujna, če želimo povečati njeno uporabnost. Večinoma gre namreč za kratke vpoglede in ne za učenje metodologije, za kar je priročnik v knjižni obliki boljši. To je še dodaten razlog, ki podpira uporabo referenc namesto dolgih opisov.

4.4 Uporaba, vrednotenje in prilagajanje metodologije

Že pri analizi in opredelitvi osnovnih korakov metodologije smo upoštevali temeljna načela in priporočila agilnih pristopov, s katerimi smo zagotovili preprostost in učinkovitost metodologije. Seveda je ključnega pomena za uporabnost metodologije njena zmožnost, da se prilagaja posameznim projektom. Vsak projekt je nekaj posebnega, zato je težko pričakovati, da bo zapisani proces (formalni del metodologije) dober za vse primere. Agilna metodologija mora biti prilagodljiva. Poleg tega je treba poskrbeti za njeno stalno obnavljanje, kar zajema in formalizira uporabne postopke in druge elemente. V

našem primeru smo predlagali določitev ustreznih vlog ter opredelitev procesa uporabe, vrednotenja in prilagajanja metodologije. Predlagani proces prikazuje slika 4.

Najprej je treba preučiti obstoječo metodologijo ter formalizirati njene osnovne elemente. Pri tem sodelujeta metodolog, ki je navadno zunanji strokovnjak, ter skrbnik metodologije. Z upoštevanjem načel in priporočil agilnih pristopov predlagata prvi osnutek agilne metodologije, ki jo morajo potrditi tudi drugi skrbniki vlog. Ti so posamezniki z največ izkušnjami za posamezno področje. V sodelovanju z drugimi uporabniki, ki nastopajo kot nosilci vloge, dopolnjujejo metodologijo s priporočili in smernicami, pridobljenih ob delu na posameznih projektih. Dopolnjevanje metodologije je dolžnost skrbnika metodologije, ki spremlja novosti na področju metodologij razvoja programske opreme.



Slika 4: **Proces uporabe, vrednotenja in prilagajanja metodologije**

Pri uporabi metodologije na konkretnem projektu skrbnik metodologije najprej preuči lastnosti projekta in določi osnovne korake metodologije. Sem sodi določanje opcijskih aktivnosti, določanje potrebnih vlog, izbira izdelkov, izbira življenjskega cikla (npr. iteracije znotraj faze konstrukcije) itn. Tako prilagojena metodologija je podlaga za izvedbo iteracije. Po zaključku iteracije se skrbnik metodologije in skrbniki vlog pogovorijo o uporabnosti metodologije. Navadno ne gre za korenite spremembe metodologije, temveč zgolj za podrobnosti, ki lahko še povečajo njeno učinkovitost. Spremembe metodologije za potrebe konkretnega projekta ne vplivajo na osnovno metodologijo. Ko je projekt končan, se skrbnik metodologije z vsemi sodelujočimi v projektu pogovori o morebitnih spremembah osnovne metodologije.

5 SKLEP

V prispevku smo predstavili nove trende, ki se uveljavljajo na področju razvoja informacijskih sistemov kot alternativa kompleksnim in togim metodologijam. Najprej smo opredelili pojem metodologija, poudarili pomen njene sociološke komponente, ki kaže, da metodologije ne moremo obravnavati neodvisno od njenih uporabnikov. Poleg formalnega dela zajema metodologija tudi neformalne in neopredeljene elemente, ki se kažejo predvsem v znanju, izkušnjah, idealih in drugih subjektivnih vrednotah posameznikov, je dinamična in se stalno bogati z novimi znanji in izkušnjami, zato moramo zanjo skrbeti in jo vzdrževati. V drugem delu smo predstavili ozadje nastanka trenda agilnosti. Opisali in komentirali smo nekaj temeljnih izhodišč. Koncept agilne metodologije smo predstavili v tretjem delu, kjer smo strnili naša razmišljanja in izkušnje, pridobljene pri zasnovi in uvajanju agilnih metodologij.

6 LITERATURA

1. Agile Alliance. *Manifesto for Agile Software Development:* www.agilealliance.org
2. Ambler W. S. *Agile Modeling: Effective Practices for extreme Programming and the Unified Process*. New York, John Wiley & Sons, Inc. 2002.
3. Ambler W. S. *The Object Primer, Second Edition: The Application Developer's Guide to Object Orientation*. New York, NY: Camridge University Press. 2001.
4. Ambler W. S. *Enterprise Unified Process White Paper*. www.ronin-intl.com/publications/unifiedProcess.htm
5. Cockburn, A. *Agile Software Development*. Pearson Education, Inc. Boston, MA, 2002.
6. Constantine, L. L. in Lockwood, L. A. D. *Software For Use: A Practical Guide to the Models and Methods of Usage-Centered Design*. New York, NY: ACM Press. 1999.
7. Highsmith, J. A. *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. Dorset House Publishing, New York, NY, 2000.
8. Krushten, P. *The Rational Unified Process – An Introduction*, 2nd ed., Addison-Wesley-Longman, Reading, MA, 2000.
9. Maier, M., Rechtin, E. *The Art of Systems Architecting*, 2nd Edition, CRC Press, Boca Raton, FL, 2000.
10. McCcarty, J., Monk, A. Channels, Conversation, Cooperation and Relevance: All You Wanted to Know about Communication but Were Afraid to Ask. V *Collaborative Computing*, 1(1), Marec 1994, str. 35–61.
11. Morabito, J., I. Sack in A. Bhate. Organisation Modelling, Innovative Architectures for the 21st Century. NJ: Prentice Hall. 2001.
12. Nonaka, I. (1991). The Knowledge-Creating Company. *Harvard Business Review*, November–december 1991.
13. O'Dell, C., Grayson, C. Jr. *If Only We Knew What We Know: The Transfer of Internal Knowledge and Best Practice*, The Free Press, New York, NY, 1996.
14. Sillince, J. A. A Model of Social, Emotional and Symbolic Aspects of Computer-Mediated Communication within Organizations. V *Computer Supported Cooperative Work*, Vol 4 (1996), str. 1–31.
15. Vavpotič, D., Bajec, M., Krisper, M. Characteristics of software development methodology evaluation model. V: *Business information technology management: facilitating global IS assiances*. 2002.
16. Vavpotič, D., Bajec, M., Krisper, M. Model za izbiro metodologije razvoja programske opreme. V: Novaković, S. (ur.). *Dnevi slovenske informatike*, Portorož, Slovenija, 17.–19. april 2002.

Dr. Marko Bajec je predavatelj na Fakulteti za računalništvo in informatiko Univerze v Ljubljani, kjer je leta diplomiral in se vpisal na podiplomski študij računalništva. Magistriral je leta 1998, doktoriral pa 2001. Na katedri za informatiko se ukvarja predvsem s strateškim planiranjem informatike ter metodologijami razvoja in vzdrževanja informacijskih sistemov. Redno sodeluje na domačih in mednarodnih konferencah ter objavlja prispevke v znanstvenih in strokovnih publikacijah. Je član Slovenskega društva INFORMATIKA, združenja PMI (Project Management Institute) – Chapter Slovenia, združenja AIS (Association for Information Systems) ter član programskega odbora posvetovanja Dnevi slovenske informatike.

Dr. Marjan Krisper je predstojnik laboratorija za informatiko na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Vodil je več raziskovalnih in razvojnih projektov v velikih sistemih v gospodarstvu in javnem sektorju. V znanstvenih in strokovnih publikacijah s področja poslovne informatike, predvsem razvojnih metodologij, strateškega planiranja in prenove poslovnih procesov je objavil približno 200 prispevkov. Je ustanovitveni član AIS (Association for Information Systems), član Slovenskega društva INFORMATIKA, PMI (Project Management Institute), Društva za umetno inteligenco, INFOSA idr.