Pristop in podporno orodje za delno avtomatski zajem metode razvoja programske opreme

Marko Janković, Slavko Žitnik, Lovro Šubelj, Neli Blagus, Aljaž Zrnec, Marko Bajec
Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Tržaška c. 25, 1000 Ljubljana
{ime.priimek}@fri.uni-lj.si

Izvleček

Študije kažejo, da uporaba in sledenje metodam za razvoj programske opreme ter njihova sposobnost prilagajanja zahtevam posameznega projekta in načinu dela razvojne ekipe pomembno vplivajo na uspešnost IT-projektov in prispeva k večji kakovosti razvite programske opreme. Kljub temu je njihova uporaba v praksi redka. Eden izmed razlogov za to je, da obstoječi pristopi zahtevajo znatno sodelovanje razvijalcev. V prispevku opišemo pristop in podporna orodja, ki omogočajo popis osnovne metode dela podjetja in njeno vzdrževanje na podlagi spremljanja dejanskih metod razvoja programske opreme ob minimalnem sodelovanju razvijalcev. Tako bodo predpisane metode odsevale dejanski način dela na projektih, zato pričakujemo, da jih bodo razvijalci sprejeli kot svoje in jim sledili bolj dosledno. Poleg tega bi s tem dosegli višjo stopnjo zrelosti procesa razvoja programske opreme.

Ključne besede: proces razvoja programske opreme, razvoj metod za potrebe specifične situacije, metode za razvoj programske opreme, izboljšave procesa razvoja programske opreme, sistemi za nadzor verzij.

Abstract

Approach and supporting toolset for semi-automatic documentation of software development methods

Studies show that the usage and acceptance of software development methods as well as their ability to adapt to the specific needs of a project and development team have a significant impact on the performance of IT projects, which in turn contributes to a higher quality of the software developed. However, their usage in practice remains low. One of the reasons is the fact that existing approaches require a significant involvement of developers. To overcome this problem we outline an approach and supporting tools which monitor in-action methods during project performance and capture the knowledge needed to supplement the current base method, with only a negligible involvement of developers. This way the prescribed methods will reflect the reality of working on projects, so we expect that the developers will perceive them as a valuable asset and will follow them more rigorously. Furthermore, this will lead to a higher maturity of software development processes.

Key words: software process, situational method engineering, software development methods, software process improvement, revision control systems.

1 UVOD

Številne študije (Bajec, Vavpotič & Krisper, 2007; Bekkers s sod., 2008; Fitzgerald & Hartnett, 2005; Karlsson & Agerfalk, 2004; Van de Weerd s sod., 2006) jasno kažejo, da ima uporaba metod za razvoj programske opreme (angl. software development method – SDM) pozitiven vpliv na proces razvoja programske opreme in prispeva k višji kakovosti produkta, tj. razvite programske opreme. Kljub vsem prednostim pa podjetja v veliki meri še vedno nimajo popisanih in dokumentiranih metod razvoja programske opreme. Tudi podjetja, ki imajo dokumentirane metode, teh ne vzdržujejo ali jim ne sledijo dosledno, kot bi bilo to pričakovano v teoriji. Znanje in izkušnje, pridobljene na posameznih projektih, torej ostajajo v glavah posameznikov. Razlogov, ki pojasnijo nastali položaj, je več (Mohan & Ahlemann, 2011; Mirbel & Ralyte, 2006; Riemenschneider s sod., 2002). Med vsemi izpostavimo predvsem (1) strogost – večina metod za razvoj programske opreme ne omogoča učinkovitega prilagajanja metod zahtevam posameznega projekta, in (2) socialno-tehnično neprimernost – metode pogosto vsili podjetje in niso prilagojene potrebam in znanju razvojne ekipe, zato jih razvijalci ne sprejmejo in jim ne sledijo v želeni meri (Vavpotič & Bajec, 2009).

Predstavljeni položaj splošno priznavajo raziskovalna skupnost in do neke mere tudi posamezniki iz prakse. Vsekakor pomeni velik in resen problem. Glede na raziskavo, ki jo je opravila družba McKinsey v sodelovanju z Univerzo v Oxfordu (Bloch, Blumberg & Laartz, 2012), kar polovica vseh velikih projektov (projekti z začetno ceno, ki presega 15 milijonov dolarjev) preseže načrtovane stroške, se ne dokonča v dogovorjenem času ali pa ne izpolni vseh zahtev, ki so bile dogovorjene. Podobno so ugotovili tudi pri Standish Group,[1] kjer so ugotovili, da se je samo 16 odstotkov opazovanih projektov končalo v skladu s planom, 32 odstotkov jih je bilo prekinjenih, preden so bili končani, in 52 odstotkov jih je bilo končanih kasneje ter z večjimi stroški, kot je bilo predvideno.

V prispevku predstavimo inovativen pristop in podporno orodje za konstruiranje metod, ki bosta vodila k večji uporabi metod razvoja programske opreme v praksi. Za razliko od obstoječih pristopov in rešitev predlagani pristop gradi bazo znanja o razvoju programske opreme znotraj podjetja na podlagi opazovanja dela na projektih. Predpostavljamo, da lahko o tem, kaj razvijalci dejansko delajo na projektu, katere dokumente ustvarjajo ter kakšna sta vrstni red ustvarjanja dokumentov in vrstni red izvajanja aktivnosti, sklepamo na podlagi opazovanja komunikacije med razvijalci in sistemi za nadzor verzij. S tem bomo omogočili, da bodo predpisano osnovno metodo podjetja stalno posodabljali in prilagajali zahtevam projektov in razvojnih skupin z minimalnim sodelovanjem razvijalcev.

V nadaljevanju najprej na kratko predstavimo različne obstoječe rešitve in pristope, ki so se razvili zaradi slabe uporabe predpisanih metod v praksi, ter predstavimo njihove pomanjkljivosti (razdelek 2). V razdelku 3 predstavimo inovativen pristop za konstruiranje metod. V razdelku 4 opišemo orodje, ki je potrebno za podporo predlaganega pristopa, v razdelku 5 pa opišemo postopek vrednotenja. Za konec sledi sklep in predstavitev nadaljnjega dela (razdelek 6).

2 PREGLED OBSTOJEČIH REŠITEV IN PRISTOPOV

V preteklosti je bilo predlaganih več pristopov in rešitev z namenom povečanja uporabe metod razvoja programske opreme v praksi. Pojavile so se različne metode, orodja in procesna ogrodja, ki omogočajo prilagajanje metode potrebam in zahtevam posameznega projekta in razvojne ekipe, npr. IBM Rational Method Composer, Microsoft Solutions Framework (Garcia, Vizcaino & Ebert, 2011). Kljub temu da ta orodja pogosto vključujejo primere dobrih praks, ki temeljijo na dolgoletnih izkušnjah in opazovanju procesa razvoja programske opreme, za učinkovito uporabo zahtevajo specifično znanje. V primeru prilagajanja metode z uporabo orodja IBM Rational Method Composer je treba podrobno poznati elemente procesnega ogrodja RUP (Rational Unified Process).

Priča smo tudi uveljavljanju številnih lahkih in agilnih metod (npr. Scrum, Extreme programming), ki so se pojavile kot odgovor na zapletene in neprilagodljive tradicionalne metode ter v ospredje postavljajo bolj človeško naravnan pristop (Laanti, Salo & Abrahamsson, 2011; Dyba & Dingsøyr, 2008). S tega vidika se zdi uporaba lahkih in agilnih metod dobra alternativa, saj si te prizadevajo približati razvijalcem in ne prinašajo tolikšne obremenitve kot tradicionalne metode. Vseeno pa zaradi poenostavitev, ki jih prinašajo, podjetja redko uporabljajo formalno zapisane metode in raje sledijo viziji posameznikov. To vodi do tega, da načini dela v podjetju niso formalno zapisani in s tem dragoceno znanje in izkušnje, pridobljene na projektih, ostajajo v glavah razvijalcev.

Obsežen sklop znanja se je oblikoval tudi na področju razvoja metod za potrebe specifične situacije (angl. situational method engineering – SME). SME neposredno obravnava togost metod razvoja programske opreme in omogoča prilagajanje in ustvarjanje metod glede na potrebe in lastnosti projekta. Predlagani in razviti so bili številni pristopi (Ralyte & Rolland, 2001; Brinkkemper, Saeki & Harmsen, 1998; Deneckere s sod., 2008), ki so bili načrtovani in izdelani s ciljem, da bi omogočali ustvarjanje novih metod iz osnovnih delov (angl. fragments) že obstoječih metod in prilagajanje že obstoječih metod zahtevam posameznega projekta, vendar empirične študije kažejo, da je njihova uporaba v praksi redka (Mirbel & Rivieres, 2002; Ralyte & Rolland, 2001; Bajec, Vavpotič & Krisper, 2007). Ustvarjanje in prilagajanje metod je časovno zelo potratno, zahteva veliko predanost in vključenost razvijalcev. Poleg tega je treba za učinkovito konstruiranje novih metod poznati vse vrste že obstoječih metod in v večini podjetij nimajo takšnega znanja. Številni pristopi SME prav tako ne upoštevajo tehničnih in socialnih vidikov, kar pogosto vodi do metod, ki so tehnično ali socialno neskladne z delom

[1] http://www.standishgroup.com

razvijalcev ali s potrebami organizacije. Posledično razvijalci predpisane metode zato pogosto dojemajo kot nekoristno in nepotrebno breme in jim ne sledijo v tolikšni meri, kot bi bilo pričakovano v teoriji. Podrobnejši pregled področja SME je na voljo v delu Henderson-Sellers in Ralyte (2010).

Kljub vsem obstoječim rešitvam in pristopom številne študije (Bajec, Vavpotič & Krisper, 2007; Bajec s sod., 2007; Vavpotič & Bajec, 2009; Vlaanderen s sod., 2011) kažejo, da ostaja uporaba metod razvoja programske opreme v praksi nizka. Številna podjetja še vedno ne uporabljajo lastnih, formalno zapisanih metod, zato delujejo v skladu z neformalnimi pravili, ki temeljijo predvsem na znanju in izkušnjah posameznih članov razvojne skupine. Tudi podjetja, ki imajo formalno zapisane metode, tem ne sledijo dosledno. Razlogi za to so predvsem (1) stroški režije, ki pridejo z uporabo teh pristopov, in (2) specifična znanja, ki so potrebna za njihovo učinkovito uporabo.

3 INOVATIVNI PRISTOP ZA KONSTRUIRANJE METOD

Glavna motivacija za naše delo je nizka uporaba in nedosledno sledenje metodam razvoja programske opreme v praksi, kar vodi do številnih neuspešnih IT-projektov in nizke kakovosti razvite programske opreme (Bekkers s sod., 2008; Fitzgerald & Hartnett, 2005). V ta namen predlagamo inovativen pristop za konstruiranje metod in podporno orodje, katerega glavni cilji so a) delno avtomatski zajem znanja, ki ga imajo podjetja in njihovi posamezniki o praksi in pristopih pri procesu razvoja programske opreme, in na podlagi tega oblikovanje in popis dejanskih načinov dela, b) spremljanje in usmerjanje dela razvijalcev na projektih razvoja programske opreme in c) kontinuirano posodabljanje in optimizacija osnovnih metod razvoja programske opreme. Za razliko od drugih pristopov, ki si prizadevajo za dosego enakega cilja, je prednost našega, da zahteva le minimalno sodelovanje razvijalcev. To je znano kot eden izmed glavnih razlogov, zakaj se podobne pobude in pristopi niso uspeli uveljaviti v praksi (Mirbel & Ralyte, 2006; Karlsson & Agerfalk, 2012).

Za doseg ciljev nameravamo opazovati delo razvijalcev in drugih, vključenih v proces razvoja programske opreme. Pri tem nas bo zanimalo, kaj dejansko delajo na projektu, katere dokumente ustvarjajo, kakšne odločitve sprejemajo v določenih okoliščinah ipd. Predpostavljamo, da lahko o slednjem sklepamo na podlagi opazovanja komunikacije med razvijalcem in sistemom za nadzor verzij (angl. revision control system). Za vsak dokument, ki ga bo uporabnik dodal ali spremenil, bomo s pomočjo različnih tehnik, npr. z rudarjenjem po besedilih (angl. text mining), rudarjenjem po procesih (angl. process mining), poskušali ugotoviti, v kateri aktivnosti se nahaja uporabnik in ali je njegovo delo v skladu s predpisano metodo. V primeru odstopanja od predpisane metode bomo s pomočjo uporabnika ugotovili, ali je prišlo do napake (uporabnik je preskočil aktivnost, ki bi jo moral predhodno končati) ali gre za novo znanje, ki ga bomo uporabili za posodobitev osnovne metode podjetja. S tem bi zajeli in dokumentirali znanje podjetja in posameznikov o razvoju programske opreme in tako preprečili, da se znanje o razvoju programske opreme nahaja samo v glavah posameznikov. S popisom dejanskega načina dela na projektih prav tako omogočimo pomoč in vodenje razvijalcev pri drugih projektih s podobnimi lastnostmi. Kot rezultat pričakujemo, da bodo podjetja dosegla višjo raven zrelosti procesa razvoja programske opreme po modelu CMMI (angl. Capability Maturity Model Integration).

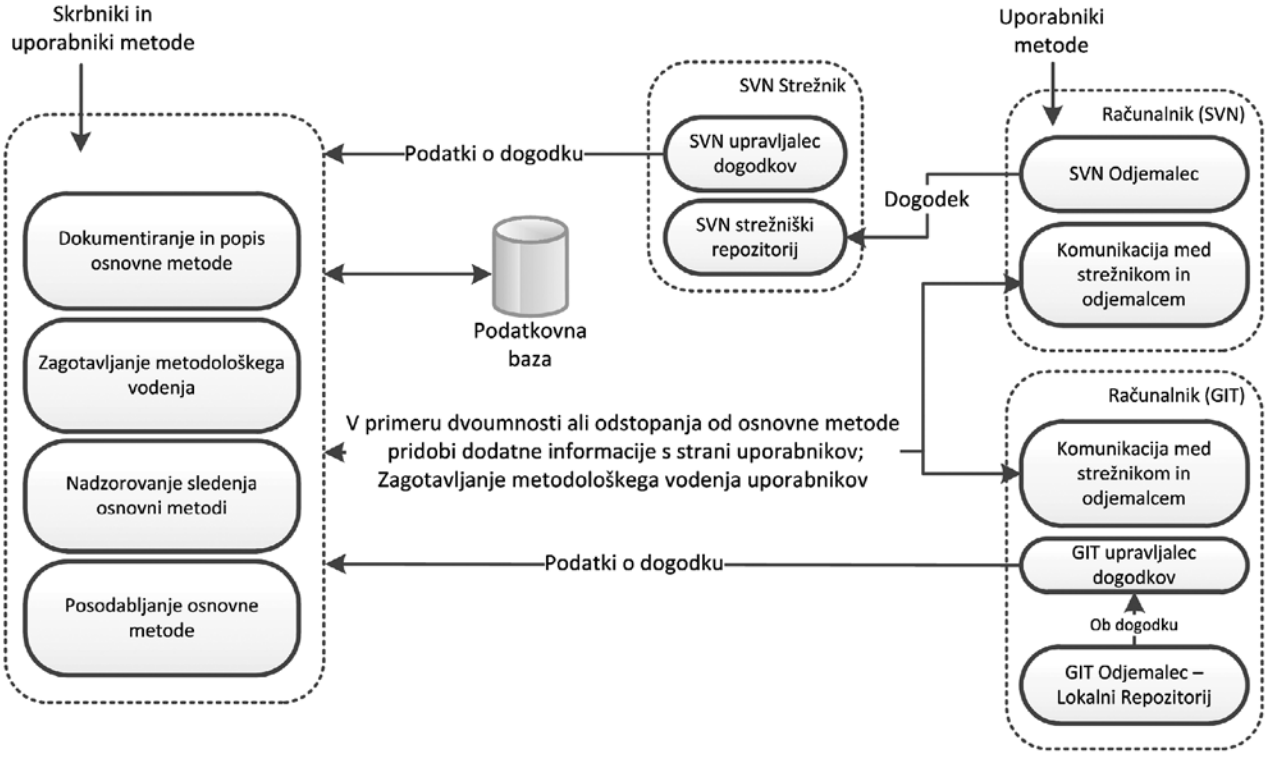
4 ORODJE ZA PODPORO PREDLAGANEGA PRISTOPA

V tem razdelku je predstavljena visokonivojska arhitektura podpornega orodja (slika 1), katerega glavni namen je olajšanje vpeljave predlaganega pristopa v prakso in testiranje postavljenih hipotez. Orodje bo samostojno in ne bo kot komponenta vključeno v razvojno okolje. V nadaljevanju na kratko predstavimo najpomembnejše komponente, njihov namen in vlogo.

4.1 Dokumentiranje in popis osnovne metode

Za podporo, uveljavitev in sledenje predpisanim metodam razvoja programske opreme je treba najprej zajeti trenutni proces razvoja in ga ustrezno dokumentirati. V podjetju, odvisno od vrst projektov, ki jih opravlja, je lahko predpisanih več različnih procesov razvoja – sestavi vseh procesov pravimo osnovna metoda (angl. base method) podjetja.

Podjetja se v splošnem strinjajo, da bi morale biti metode razvoja dokumentirane, vendar se med seboj razlikujejo v ravni podrobnosti, ki se jim zdijo uporabne in bi jih bilo smiselno popisati (Garcia s sod., 2011). Zajem, popis in ažuriranje dejanskega



Slika 1: **Visokonivojska arhitektura predlaganega orodja s podporo za Subversion (SVN) in GIT**

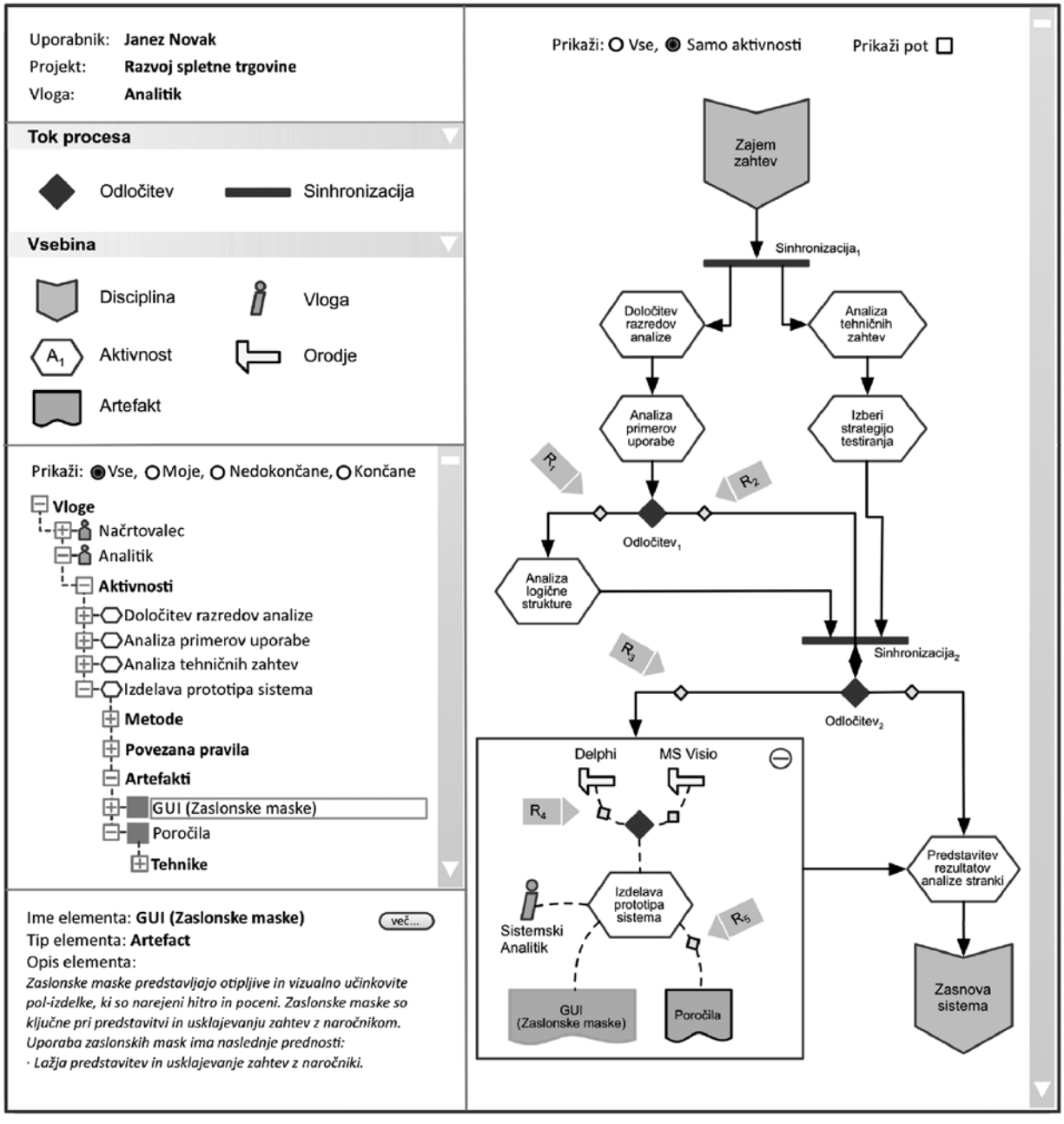
načina dela je običajno zamudno in drago opravilo, zato se podjetja za le redko odločajo za to. Podporno orodje mora tako omogočati zajem in popis osnovne metode podjetja na interaktiven in preprost način ter med drugim omogočiti, da podjetja sama določijo, v kolikšnem obsegu in kateri elementi (npr. aktivnosti, tehnike, orodja) naj se zajemajo. Orodje mora poleg osnovnih metod omogočati tudi zajem podpornih metamodelov (Bajec & Vavpotič, 2008). V prvem koraku vodilni razvijalci, ki so najbolj seznanjeni s trenutnim delom v podjetju in so določeni za skrbnike metode, ustvarijo metamodel, v katerem zajamejo ključna načela osnovne metode (npr. določijo metaelemente, ki lahko nastopajo v osnovni metodi, dovoljene povezave med njimi in druga pravila, kot npr. katere podelemente mora obvezno vsebovati določen element ipd.). Metamodel tako skrbi za konsistentnost osnovne metode in zagotavlja urejenost in celovitost. Skrbniki metode prav tako popišejo začetno osnovno metodo, ki naj bi ji razvijalci sledili pri svojem delu in ki bo uporabljena kot začetni vhod v komponento za nadzorovanje sledenja osnovni metodi.

Orodje mora omogočati vizualizacijo osnove metode. Odločili smo se za prikaz v obliki diagrama in drevesne strukture. Uporabniki metode imajo možnost urejanja (dodajanje, spreminjanje elementov in povezav) osnovne metode podjetja. Primer zaslonske maske uporabniškega vmesnika je prikazan na sliki 2. Posodabljanje metode mora biti omogočeno na interaktiven in uporabniku prijazen način (npr. uporabnik mora imeti možnost izbire elementa iz menija na levi in ga s tehniko povleci in spusti postaviti na želeno mesto v diagramu). Za posamezno aktivnost, orodje in druge elemente metode lahko uporabnik doda kratek opis in predloge za primere dobre prakse.

4.2 Zagotavljanje metodološkega vodenja

Dokumentirana osnovna metoda podjetja vključuje aktivnosti, ki jih je treba izvesti med razvojem različne tehnike in orodja, ki ju lahko uporabijo razvijalci, ter predloge in primere dobre prakse, ki služijo kot opora in vodilo razvijalcem pri njihovem delu. Prav to je še posebno pomembno za nove razvijalce, ki se še spoznavajo s procesom razvoja programske opreme znotraj podjetja.

Vsak razvijalec mora imeti dostop do pregleda osnovne metode podjetja. Razvijalec lahko za posamezni projekt preveri pot skozi predpisano metodo. Interaktivno, s klikanjem na elemente osnovne metode se uporabniku prikaže podrobnejši opis in predstavitev izbranega elementa. Poleg tega so za vsako aktivnost in razvojno orodje na voljo tudi predloge in primeri dobre prakse, ki uporabniku služijo kot vodilo pri delu. Primer zaslonske maske uporabniškega vmesnika je enak zaslonu na sliki 2, pri čemer imajo samo uporabniki z ustreznimi pravicami možnost spreminjanja osnovne metode podjetja (dodajanja novih elementov in povezav). Pri tem želimo predvsem izključiti nove razvijalce, ki se še spoznavajo s procesom razvoja v podjetju in še nimajo dovolj znanja, da bi bili primerni za popis novih načinov dela.



Slika 2: **Primer zaslonske maske komponente za zajem osnovne metode podjetja**

4.3 Nadzorovanje sledenja osnovni metodi

Najpomembnejša funkcionalnost orodja je sposobnost sledenja metodi v praksi (angl. in-action method) in zaznavanje morebitnih odstopanj od predpisane osnovne metode podjetja. Orodje se pri zaznavanju odstopanj lahko zanaša na (1) osnovno metodo podjetja, ki predpisuje, kako je treba izvajati različne vrste projektov, in (2) spremljanje metode v praksi, ki ni dokumentirana, vendar lahko o njej sklepamo na podlagi opazovanja dela razvijalcev in njihove komunikacije s sistemi oz. aplikacijami, ki jih uporabljajo pri izvajanju različnih opravil med razvojem.

Predpostavljamo, da lahko o metodi, ki se uporablja v praksi, sklepamo na podlagi rudarjenja po dnevniku sistema za nadzor verzij (angl. revision control system – RCS) in s skrbnim upoštevanjem osnovne metode, ki določa povezave med elementi metode, kot so aktivnosti, artefakti in vloge. Osnovna metoda nam tako omogoča, da vemo, katere artefakte lahko pričakujemo v posamezni aktivnosti od razvijalcev. Na podlagi artefaktov, ki so že dodani v RCS, lahko sklepamo o poti skozi osnovno metodo in katere aktivnosti so bile že izvedene za posamezni projekt. V primeru, ko je dodani artefakt v skladu z osnovno metodo, lahko nadaljujemo, sicer je treba zajeti razloge in okoliščine za odstopanje. Za Subversion in GIT, ki veljata za dva izmed najbolj uporabljenih RCS, smo za ta namen razvili ustrezne razširitve, ki spremljajo in zajemajo aktivnosti uporabnika. Predlagano orodje predvideva splošni vmesnik za poročanje o delu razvijalcev z namenom naknadnega dodajanja podpore za različne vrste aplikacij, kar bi omogočalo bolj natančno spremljanje dela razvijalcev.

Tipično ločimo dve vrsti razlogov, zaradi katerih metoda v praksi odstopa od predpisane.

- Uporabnik je naredil napako ter npr. dodal artefakt B pred artefaktom A, kar ni v skladu z osnovno metodo. V tem primeru je uporabnik verjetno pozabil izvesti aktivnost, ki je odgovorna za izdelavo artefakta A.
- Posebne okoliščine so uporabnika vodile, da se je odločil drugače, kot je predpisano z osnovno metodo.

V prvem primeru od uporabnika pričakujemo, da bo odpravil oz. kompenziral posledice napake (npr. izdelava manjkajočih artefaktov). V drugem primeru je treba ustrezno posodobiti osnovno metodo (dodajanje novih elementov in povezav), tako da bo zajemala novo znanje, ki ga lahko uporabimo v prihodnjih primerih.

Orodje mora omogočati, da na podlagi podatkov, pridobljenih iz RCS, in z uporabo različnih tehnik, kot so rudarjenje besedil, rudarjenje po procesih ipd., določimo aktivnost, v kateri se nahaja uporabnik, in kateri aktivnosti pripadajo dodani artefakti.

4.4 Upravljavec dogodkov RCS

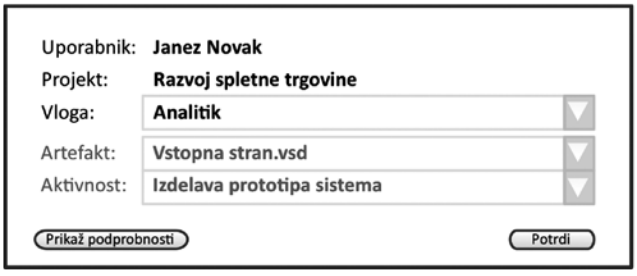
Večina sistemov za nadzor verzij podpira pred- in podogodkovne prožilce (angl. pre/post event hooks), ki omogočajo izvajanje določenih aktivnosti pred določenim dogodkom ali po njem, kot je npr. potrditev (angl. commit). To je pomembno, saj nam omogoča, da prestrezamo vse akcije, ki jih uporabnik izvaja z RCS, in tako dobimo informacije o aktivnostih na projektu. S tem namenom smo razvili razširitve za izbrane RCS, kar nam omogoča zbiranje podatkov, kot so komentar, uporabniško ime, časovni žig, spremenjeni dokumenti, tipi dokumentov ipd. Vsi zbrani podatki so nato poslani na strežnik, na katerem jih obdelamo in uporabimo za ugotavljanje, ali so uporabnikove aktivnosti v skladu s predpisano metodo podjetja.

4.5 Komunikacija med strežnikom in odjemalcem

V določenih primerih informacije, pridobljene na podlagi rudarjenja po dnevniku sistemov za nadzor verzij in na podlagi rudarjenja po vsebini artefaktov, ne bodo omogočile avtomatične določitve, kateri aktivnosti pripada določeni artefakt (tj. aktivnost, v kateri se nahaja razvijalec). Za take primere orodje vsebuje komponento, ki omogoča komunikacijo med strežnikom in uporabnikom. Glavni namen te komponente je, da od uporabnika zajame dodatne informacije, ki so potrebne za umestitev posameznega artefakta v določeno aktivnost. Za potrditev pravilnosti npr. se uporabniku prikaže pogovorno okno (slika 3) z vnaprej izpolnjenimi polji. V primeru, da je sistem pravilno določil aktivnost, kateri pripada artefakt, uporabnik pritisne gumb »Potrdi«, sicer pred tem izbere ustrezne vrednosti.

Komponenta ima pomembno vlogo tudi v primeru odstopanja od predpisane metode. V tem primeru uporabniku posreduje vprašanja, ki so dinamično generirana na strežniški strani in katerih glavni namen je zajem posebnih okoliščin in razlogov, ki so botrovali delovanju, ki ni v skladu s predpisano metodo. Zajete informacije pomenijo znanje o novem

načinu dela na projektih, ki ga uporabimo za posodobitev osnovne metode podjetja. V primerih, ko odstopanje ni namerno, uporabniku lahko zagotovimo metodološko vodenje in mu predstavimo primere dobre prakse za aktivnost, v kateri se nahaja.



Slika 3: **Primer pogovornega okna za zajem informacij o artefaktu, ki je bil dodan v RCS**

4.6 Posodabljanje osnovne metode

Ob ugotovitvi, da se metoda v praksi razlikuje od osnovne, predpisane metode (npr. sistem ugotovi, da je razvijalec dodal dokument, ki ni bil pričakovan v skladu z osnovno metodo), je treba o tem obvestiti razvijalca in ugotoviti, ali je prišlo do odstopanja namerno. V primeru, da odstopanje ni bilo namerno, razvijalcu zagotovimo metodološko vodenje in ga seznanimo s primeri dobre prakse za trenutno aktivnost. Poleg tega od njega zahtevamo, da popravi storjeno napako in vse morebitne posledice. V nasprotnem primeru, ko je odstopanje namerno, je treba zajeti razloge in okoliščine, v katerih je novi način razvoja primeren, ter jih skupaj z novimi povezavami in elementi metode uporabiti za dopolnitev osnovne metode podjetja. Za zajem vseh potrebnih podatkov uporabniku posredujemo obrazec, ki je dinamično ustvarjen na strežniški strani. Obrazec je sestavljen iz različnih tipov vprašanj, ki omogočajo, da od uporabnika na preprost način zajamemo znanje o tem, kateri so novo dodani elementi, kakšne so povezave med novimi in že obstoječimi elementi ter kakšne so okoliščine in razlogi (zapisano v obliki pravil), ki so botrovali k delovanju, ki ni v skladu s predpisano metodo. V primeru odstopanja lahko uporabniki osnovno metodo posodobijo tudi prek grafičnega vmesnika za zajem osnovne metode podjetja. Osnovna metoda podjetja bi se tako stalno prilagajala zahtevam projektov in odražala dejansko delo razvijalcev, ki bi aktivno sodelovali pri njenem oblikovanju. Zaradi tega pričakujemo, da bodo razvijalci predpisano metodo dojemali kot nekaj uporabnega in koristnega in ne le kot dodatno, nepotrebno breme.

5 VREDNOTENJE PRISTOPA

Vrednotenje našega pristopa bo potekalo v dveh korakih. Najprej bomo vrednotenje izvedli v akademskem okolju. Predlagani pristop in podporno orodje bomo uporabili pri predmetu, pri katerem bodo morali študentje razviti informacijski sistem in pri tem slediti točno določeni vnaprej predpisani metodi razvoja. To nam bo omogočilo preveriti, ali se orodje obnaša tako, kot je bilo pričakovano, in izpolnjuje vse predpisane zahteve. Poleg tega bomo lahko ocenili, ali so zajeti podatki zadostni za nadzor in spremljanje sledenja osnovne metode podjetja. Po končanem razvoju informacijskega sistema bomo opravili pogovore z vsemi sodelujočimi z namenom, da pridobimo povratne informacije o mogočih izboljšavah ter percepciji uporabnosti pristopa in podpornega orodja. Na podlagi rezultatov vrednotenja in zbranih informacij bomo optimizirali in izboljšali tako pristop kot tudi podporno orodje.

V drugem koraku bo vrednotenje pristopa in podpornega orodja izvedeno v sodelovanju z industrijo. Glavni namen tega je, da preverimo uspešnost predlaganega pristopa v praksi ter s tem tudi testiramo postavljene hipoteze. Trenutno smo že dogovorjeni za sodelovanje z lokalno industrijo, prav tako pa sodelujemo z raziskovalno skupino univerze v Parizu 1, ki nam bo pomagala vpeljati naš pristop v francoska podjetja. Idealen vzorec podjetij bi vključeval podjetja različnih velikosti (mikro, mala, srednja in velika), ki opravljajo različne projekte in so iz različnih kulturnih okolij. Ko bo predlagani pristop vpeljan v posamezno podjetje in bodo razvijalci seznanjeni z vsemi možnostmi uporabe, bomo na podlagi intervjujev z razvijalci oblikovali trenutno osnovno metodo podjetja, ki bo služila kot izhodiščna točka za nadaljnje opazovanje. Nato bomo s pomočjo podpornega orodja opazovali delovanje razvijalcev in shranjevali vse aktivnosti, ki jih ti opravljajo v povezavi s predlaganim pristopom (npr. potreba po metodološkem vodenju, spreminjanje osnovne metode, spreminjanje opisov posameznih elementov osnovne metode). Prav tako bomo zapisovali, koliko časa razvijalec porabi za uporabo novega pristopa v splošnem in koliko časa za posamezno funkcionalnost. Na podlagi števila odstopanj od predpisane metode, števila zahtev za metodološko vodenje in števila posodobitev

osnovne metode bomo lahko sklepali, ali so razvijalci seznanjeni s predpisano metodo podjetja in ali je ta skladna z metodo, ki jo razvijalci uporabljajo na projektih. Vsi zbrani podatki bodo prav tako uporabljeni za analizo uporabnosti in integracije pristopa v prakso, obenem pa nam bodo omogočili vpogled v sam proces razvoja programske opreme. Za potrditev zbranih podatkov in ugotovitev ter zajem podrobnosti o sami uporabnosti pristopa bomo na mesečni ravni izvajali intervjuje z razvijalci, pri čemer bomo uporabljali inovativne tehnike, kot je npr. »think-aloud experiment«, kar nam bo omogočilo zajem pomembnih in relevantnih informacij v povezavi s predlaganim pristopom.

6 SKLEP

V prispevku smo predstavili inovativni pristop in podporno orodje za konstruiranje metod. Glavna motivacija za to je bila nizka uporaba in nedosledno sledenje metodam razvoja programske opreme v praksi, kar vodi do številnih neuspešnih IT-projektov in nizke kakovosti razvite programske opreme. S pristopom želimo omogočiti zajem in popis znanja o razvoju programske opreme z minimalnim sodelovanjem razvijalcev. Prav tako želimo omogočiti spremljanje uveljavljanja predpisane metode v praksi in zajem znanja ter posodobitev osnovne metode v primerih odstopanja. Pričakujemo, da se bodo z uporabo predlaganega pristopa osnovne metode podjetja bolje prilagajale dejanskemu načinu dela na projektih ter znanju in zahtevam razvojne skupine. S pristopom tako omogočimo, da se znanje o razvoju programske opreme ne nahaja samo v glavah posameznikov, temveč je sistematsko dokumentirano. To omogoča hitrejše uvajanje novih in neizkušenih razvijalcev.

Trenutno se ukvarjamo z razvojem podpornega orodja, ki nam bo omogočilo testiranje hipotez v praksi. Končano orodje in sam pristop nameravamo vrednotiti v okolju fakultete, izboljšano in optimizirano pa kasneje tudi v različnih domačih in tujih podjetjih. Nadaljnje delo vključuje tudi razvoj razširitev za druge sisteme, ki jih razvijalci uporabljajo pri svojem delu (npr. JIRA, Bugzilla, Sharepoint), kar bi omogočilo lažje in bolj celovito spremljanje dela razvijalcev in posledično omogočilo zajem dodatnega znanja o razvoju programske opreme.

7 VIRI IN LITERATURA

[1] Bajec, M., Vavpotič, D., Krisper, M. (2007). Practice-driven approach for creating project-specific software development methods. *Information and Software Technology*, št. 49, zv. 4, str. 345–365.

[2] Bajec, M., Vavpotič, D. (2008). A framework and tool-support for reengineering software development methods. *Informatica, Lith. Acad. Sci.*, št. 19, zv. 3, str. 321–344.

[3] Bajec, M., Vavpotič, D., Furlan, Š., Krisper, M. (2007). Software process improvement based on the method engineering principles. *Situational Method Engineering: Fundamentals and Experiences, ser. IFIP International Federation for Information Processing*, J. Ralyté, S. Brinkkemper, and B. Henderson-Sellers (ur.). Springer Boston, 2007, št. 244, str. 283–297.

[4] Bekkers, W., van de Weerd, I., Brinkkemper, S., Mahieu, A. (2008). The influence of situational factors in software product management: An empirical study. Second International Workshop on Software Product Management. *IWSPM '08*, str. 41–48.

[5] Bloch, M., Blumberg, S., Laartz, J. (2012). *Delivering large-scale it projects on time, on budget, and on value*. http://www.mckinsey.com/client service/business technology.

[6] Brinkkemper, S., Saeki, M., Harmsen, F. (1998). Assembly Techniques for Method Engineering. *Proceedings of the 10th International Conference on Advanced Information Systems Engineering*, London, UK, str. 381–400.

[7] Deneckere, R., Iacovelli, A., Kornyshova, E., Souveyet, C. (2008). From Method Fragments to Method Services. *Proc. of EMMSAD'08*, Montpellier, France, str. 80–96.

[8] Dybå, T., Dingøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, št. 50, zv. 9–10, str. 833–859.

[9] Fitzgerald, B., Hartnett, G. (2005). A study of the use of agile methods within intel. *Business Agility and Information Technology Diffusion, ser. IFIP International Federation for Information Processing*, R. Baskerville, L. Mathiassen, J. Pries-Heje, and J. DeGross (ur.). Springer US, št. 180, str. 187–202.

[10] García, F., Vizcaion, A., Ebert, C. (2011). Process management tools. *IEEE Software*, št. 28, zv. 2, str. 15–18.

[11] García, J., Amescua, A., Sánchez, M., Bermón, L. (2011). Design guidelines for software processes knowledge repository development. *Information and Software Technology*, št. 53, zv. 8, str. 834–850.

[12] Henderson-Sellers, B., Ralyté, J. (2010). Situational method engineering: State-of-the-art review. *Journal of Universal Computer Science*, št. 16, zv. 3, str. 424–478.

[13] Karlsson, F. Ågerfalk, P. J. (2004). Method configuration: adapting to situational characteristics while creating reusable assets. *Information and Software Technology*, št. 46, zv. 9, str. 619–633.

[14] Karlsson, F. Ågerfalk, P. J. (2012). MC Sandbox: Devising a tool for method-user-centered method configuration. *Information and Software Technology*, št. 54, zv. 5, str. 501–516.

[15] Laanti, M., Salo, O., Abrahamsson, P. (2011). Agile methods rapidly replacing traditional methods at nokia: A survey of opinions on agile transformation. *Information and Software Technology*, št. 53, zv. 3, str. 276–290.

[16] Mohan, K., Ahlemann, F. (2011). What methodology attributes are critical for potential users? Understanding the effect of human needs. *Advanced Information Systems Engineering, ser. Lecture Notes in Computer Science*, H. Mouratidis in C. Rolland (ur.). Springer Berlin / Heidelberg, št. 6741, str. 314–328.

[17] Mirbel, I., Rivieres, V. de (2002). Adapting analysis and design to software context: The JECKO approach. *Object-Oriented Information Systems*, Z. Bellahsène, D. Patel in C. Rolland (ur.). Springer Berlin Heidelberg, str. 223–228.
[18] Mirbel, I., Ralyte, J. (2006). Situational method engineering: combining assembly-based and roadmap-driven approaches. *Requirements Engineering*, št. 11, zv. 1, str. 58–78.
[19] Ralyte, J., Rolland, C. (2001). An assembly process model for method engineering. *Advanced Information Systems Engineering*, K. R. Dittrich, A. Geppert in M. C. Norrie (ur.). Springer Berlin Heidelberg, str. 267–283.
[20] Riemenschneider, C., Hardgrave, B., Davis, F. (2002). Explaining software developer acceptance of methodologies: a comparison of five theoretical models. *IEEE Transactions on Software Engineering*, št. 28, zv. 12, str. 1135–1145.
[21] van de Weerd, I., Brinkkemper, S., Souer, J., Versendaal, J. (2006). A situational implementation method for web-based content management system-applications: method engineering and validation in practice. *Software Process: Improvement and Practice*, št. 11, zv. 5, str. 521–538.
[22] Vavpotič, D., Bajec, M. (2009). An approach for concurrent evaluation of technical and social aspects of software development methodologies. *Information and Software Technology*, št. 51, zv. 2, str. 528–545.
[23] Vlaanderen, K., van de Weerd, I., Brinkkemper, S. (2011). The online method engine: From process assessment to method execution. *Engineering Methods in the Service-Oriented Context, ser. IFIP Advances in Information and Communication Technology*, J. Ralyté, I. Mirbel in R. Deneckère (ur.). Springer Boston, št. 351, str. 108–122.

▪

Marko Janković je mladi raziskovalec na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Njegovo glavno raziskovalno področje je razvoj novih pristopov za izboljšanje metod razvoja programske opreme in njihove uporabe v praksi.

▪

Slavko Žitnik je doktorski študent na Fakulteti za računalništvo in informatiko Univerze v Ljubljani in hkrati zaposlen kot mladi raziskovalec iz gospodarstva v podjetju Optilab, d. o. o. Raziskovalno se ukvarja predvsem s procesiranjem besedil, bolj natančno z razpoznavanjem entitet in povezav med njimi z uporabo metod strojnega učenja in semantičnih tehnologij.

▪

Lovro Šubelj je asistent na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Poučuje predvsem predmete s področja podatkovnih baz. Raziskovalno se ukvarja z analizo realnih omrežij, natančneje z odkrivanjem značilnih skupin vozlišč v velikih kompleksnih omrežjih. Je avtor ali soavtor številnih prispevkov v strokovnih in znanstvenih publikacijah.

▪

Neli Blagus je mlada raziskovalka v Laboratoriju za podatkovne tehnologije na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Raziskovalno se ukvarja z analizo omrežij.

▪

Aljaž Zrnec je magistriral leta 2002 na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Leta 2006 je doktoriral s področja konstruiranja metodologij. Zaposlen je v Laboratoriju za podatkovne tehnologije kot asistent za področje podatkovnih baz. Na raziskovalnem področju se ukvarja s konstruiranjem metodologij, podatkovnimi bazami NoSQL in z računalništvom v oblaku. Je avtor ali soavtor številnih prispevkov v strokovnih in znanstvenih publikacijah.

▪

Marko Bajec je izredni profesor na Fakulteti za računalništvo in informatiko Univerze v Ljubljani, kjer poučuje dodiplomske in podiplomske predmete s področja razvoja informacijskih sistemov in podatkovnih baz. Raziskovalno se ukvarja z metodami in pristopi k snovanju in razvoju informacijskih sistemov, obvladovanjem informatike ter v zadnjih letih predvsem s podatkovnimi tehnologijami za predstavitev, analizo in vizualizacijo podatkov. Leta 2009 je ustanovil Laboratorij za podatkovne tehnologije ter prevzel njegovo vodenje. Je član številnih domačih in tujih združenj, komisij in odborov. V okviru fakultete je vodil več aplikativnih in raziskovalnih projektov. Svoje raziskovalne rezultate in dosežke iz prakse redno objavlja v domačih in mednarodnih znanstvenih in strokovnih krogih.