UDC: 004.41 INFO M: str. 20-27

NEKI ASPEKTI USPOSTAVLJANJA SLEDLJIVOSTI U AGILNOM RAZVOJU SOFTVERA
SOME ASPECTS OF ESTABLISHING TRACEABILITY IN AGILE SOFTWARE DEVELOPMENT

Sonja Dimitrijević, Snežana D. Pantelić

REZIME: Sledljivost u softverskom inženjerstvu je poznat koncept od velikog značaja za ovu oblast o čemu svedoče mnoga istraživanja. Međutim, inženjerske prakse koje su do sada razvijene za uspostavljanje sledljivosti, uglavnom ne odgovaraju sve više prisutnim agilnim metodologijama. Osim toga, u praksi agilnog razvoja softvera je prisutan negativan stav prema sledljivosti, dobrim delom kao posledica nedovoljnog poznavanja samog koncepta. Ovaj rad ima za cilj da pruži pogled na sledljivost iz perspektive agilnog razvoja softvera. U radu su prvo ukratko izloženi teorijski elementi sledljivosti u softverskom inženjerstvu. Na toj osnovi, analizirane su mogućnosti postojećih praksi i tehnoloških rešenja za uspostavljanje i poboljšanje sledljivosti u projektima agilnog razvoja softvera. Takođe, razmotreni su dometi aktuelnih pravaca istraživanja vezanih za realizaciju sledljivosti, ali u granicama utvrđenog agilnog konteksta. Na kraju su sumirani opšti izazovi i smernice za realizaciju efektivnih rešenja implementacije strategije sledljivosti u agilnom projektu.
KLJUČNE REČI: Sledljivost, Softversko inženjerstvo, Agilni razvoj softvera
ABSTRACT: Traceability in software engineering is a well known concept, of great importance for this field, addressed in numerous researches. However, the engineering practices in establishing traceability that have been developed so far, in most cases do not fit into advancing agile methodologies. Besides, negative attitude towards traceability is present in the practice of agile software development, which can be viewed as a consequence to insufficient understanding of this concept. The goal of this paper is to present a view on traceability from the point of agile software development. First, theoretical elements of traceability in software engineering are presented in brief. Based on that, existing practices and technological solutions with implementation in projects of agile software development are being analyzed to identify possibilities of establishing and improving traceability. Also, range of current directions in research related to the realization of traceability, within the boundaries of established agile context, has been elaborated. Finally, general challenges and guidelines on realization of effective solutions of the implementation of strategy of traceability in an agile project have been summarized and presented.
KEY WORDS: Traceability, Software Engineering, Agile Software Development

1. UVOD

Sledljivost, opšte poznata kao sposobnost da se opišu i prate veze između artifakata/izlaza procesa životnog ciklusa softvera, od velikog je značaja u tzv. klasičnim pristupima razvoja softvera. Različiti teorijski aspekti, prakse i rutine sledljivosti, podržani su značajnim radovima u literaturi, ali kao što je to često slučaj u softverskom inženjerstvu, bez jedinstvene polazne odrednice i sa različitim implikacijama. Uspostavljanje sledljivosti se zahteva ili preporučuje kao „dobra praksa“ u određenom broju međunarodnih standarda i inicijativa u oblasti softverskog inženjerstva, kao što su ISO 15504 i CMMI [1].

U mnogim „mission critical“ sistemima, od razvojnih inženjera se zahteva da pokažu, ne samo da je svaki zahtev u potpunosti realizovan, već i da nema dodatnog koda, niti koda kojem se ne može ući u trag. Prema Palmeru, adekvatna sledljivost može doneti bitne prednosti u oblasti projektnog menadžmenta, preglednosti procesa, verifikacije i validacije, održavanja [2]. U suprotnom, u velikoj meri utiče na neuspeh projekta i prekoračenje budžeta. U osnovi, cilj sledljivosti je da poboljša kvalitet softverskih sistema. Određenije, sledljivost može pružiti podršku analizi uticaja i integracije promena, praćenju napretka projekta uz smanjenje intervala razvoja, održavanju i poboljšanju softverskih sistema, ponovnoj upotrebi softverskih komponenti itd.

Iako su koncept i prakse sledljivosti davno uvedni, i njihov značaj i uloga su dobro poznati, postojeći širok opseg rešenja pokazuje značajna odstupanja u pogledu kvaliteta i efektivnosti. Modeli razvoja softvera, tehnologija i poslovne potrebe se neprestano menjaju, zahtevajući tako nove metode sledljivosti i alate podrške. Pristupi izučavanja sledljivosti su heterogeni. Odnose se na modele sledljivosti kojima se definišu značajni artifakti i tipovi veza, na uticajne faktore, različite implementacione modele, tehnologiju, troškovnu efikasnost i efektivnost rešenja, koristi.

Kada je reč o agilnim metodologijama, ne postoje eksplicitne prakse vezane za uspostavljanje sledljivosti. Karakteristično je nedovoljno poznavanje samog koncepta i prisutne predrasude u praksi o sledljivosti kao o nepotrebnom, neefikasnom naporu, pogotovo suvišnom u agilnom projektu. Sa postojećim tendencijama koje se karakterišu zahtevima brojnih organizacija, klijenata, regulative i standarda za omogućavanje sledljivosti tokom životnog ciklusa softvera, suočavaju se i agilne organizacije. Takođe, u standardizovanim okruženjima postoje nastojanja za povećenje agilnosti uvođenjem agilnih praksi bez narušavanja sistema kvaliteta, što znači i nenarušavanje uspostavljenog nivoa sledljivosti [3]. Sve to, tipičnim pitanjima koja se odnose na sledljivost, poput:

– koja je prava mera sledljivosti,
– kako doći do troškovno efektivnih rešenja sledljivosti,

sasvim prirodno pridodaje pitanja kao što su:

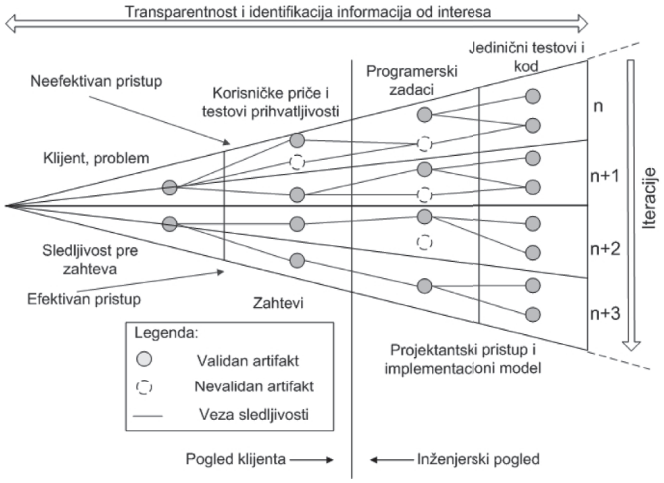
– da li postojeće agilne prakse obezbeđuju sledljivost i u kojoj meri,
– da li je moguće uspostaviti/poboljšati sledljivost na agilan način.

U ovom radu su prikazani rezultati istraživanja postojećeg mesta, pristupa i praksi uspostavljanja sledljivosti u agilnom razvoju softvera, uz osvrt na aktuelna istraživanja i očekivane buduće tendencije. Posle opštih razmatranja koncepta sledljivosti i predstavljanja agilnih metodologija, analiziran je problem sledljivosti u agilnom kontekstu i predočeni su osnovni pristupi i prakse. Iako se sledljivost može posmatrati sa mnogo aspekata, u fokusu ovog rada su metodološki i tehnološki aspekti zbog svoje aktuelnosti i značaja za dati kontekst. Na kraju su sumirani opšti izazovi i smernice za realizaciju efektivnih rešenja u agilnom projektu.

2. KONCEPT SLEDLJIVOSTI U SOFTVERSKOM INŽENJERSTVU

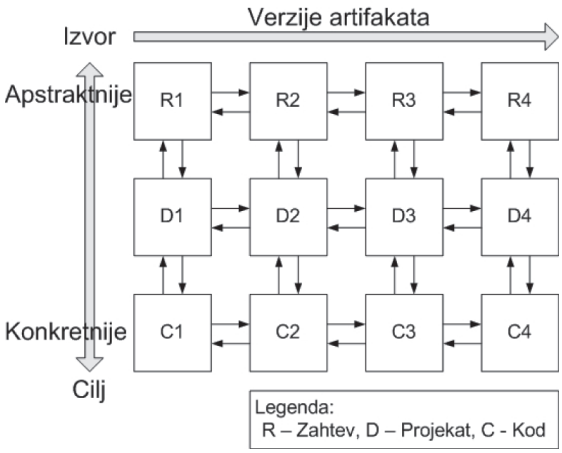
Sledljivost je opšte poznata kao sposobnost da se opišu i prate veze između artifakata/autputa procesa životnog ciklusa softvera kao što su zahtevi, elementi dizajna, izvorni kod, testovi. Koncept je dugo prisutan u teoriji i praksi i objavljeni su brojni značajni radovi na ovu temu. Međutim, iako su prezentovane brojne tehnike za rešavanje pitanja sledljivosti, sam problem ima različite definicije i fundamentalne konflikte u praksi [4].

Sledljivost zahteva je prisutna u praksi od sedamdesetih godina prošlog veka u cilju osiguranja kompletnosti i konzistentnosti informacija o proizvodu. Danas je nekoliko komplementarnih definicija široko prihvaćeno. Sledljivost zahteva se odnosi na „sposobnost da se opiše i prati život zahteva u oba smera“ u cilju razumevanja porekla zahteva i načina njegove realizacije u izlazima procesa kao što su projekat, kod, i slučajevi testiranja [4]. Isti autori uvode sledeće pojmove: sledljivost pre specifikacije zahteva koja se odnosi na aspekte života zahteva pre njihovog specifikovanja; i sledljivost posle specifikacije zahteva vezanu za apekte života zahteva koji rezultuju iz uključivanja zahteva u specifikacije (slika 1 [5]).



Slika 1. – *Sledljivost pre i posle specifikacije zahteva (SZ)*

Jedno od određenja definiše sledljivost u softverskom inženjerstvu kao „sposobnost da se opišu i prate veze između zahteva, dizajna i implementacionih artifakata“ [6]. Ističu se horizontalni i vertikalni aspekt sledljivosti, pri čemu je vertikalna sledljivost povezivanje zahteva, dizajna i koda, dok se horizontalna odnosi na veze između verzija zahteva (slika 2) [7].



Slika 2. – *Horizontalni i vertikalni aspekt sledljivosti*

Sledljivost, dakle, održava mrežu zavisnosti između projektnih artifakata. Neposredne prednosti od uspostavljene sledljivosti su poboljšan kvalitet proizvoda, efektivan odgovor promenama, kontrolisani zahtevi i proizvodi, i kao rezultat, smanjenje intervala razvoja [8].

Opšte prihvaćeno viđenje sledljivosti ograničava ovaj pojam na artifakte modela sistema. Navedeni artifakti opisuju sistem u razvoju na različitim nivoima apstrakcije, pa otuda generički termin model sistema. Artifakti projektnog menadžmenta (npr. artifakti planiranja) se obično ne prate do odgovarajućih delova modela sistema (npr. zahteva ili artifakata dizajna). To je iznenađujuće s obzirom na činjenicu da postoji mnogo značajnih veza između artifakata projektnog menadžmenta i sistema. Tek nekoliko pristupa uzima u obzir sledljivost između specifikacije sistema i artifakata projektnog menadžmenta, ipak poslednjih godina raste interes za ovakvo, šire viđenje sledljivosti. Tako uspostavljena sledljivost može doneti dodatne prednosti, između ostalog pojednostavljen pristup (direktna navigacija) relevantnim informacijama čiji su vlasnici i korisnici različiti učesnici u projektu, podršku samom procesu planiranja, omogućavanje primene validacionih tehnika na jedinstven model (za kriterijume kao što su konzistentnost i integritet) [6].

CMMI za drugi nivo zrelosti nalaže realizaciju sledljivosti zahteva, a koristi se definicijom IEEE po kojoj je sledljivost stepen do kojeg veza može biti uspostavljena između dva ili više proizvoda razvojnog procesa, naročito proizvoda koji imaju odnos prethodnik-sledbenik ili nadređeni-podređeni. Oblast procesa upravljanja zahtevima prema CMMI definiše specifične prakse koje se odnose na sledljivost. Preporučuje se održavanje dvosmerne sledljivosti između zahteva i drugih proizvoda razvojnog procesa, pri čemu je dvosmerna sledljivost sposobnost praćenja u oba smera, od zahteva do krajnjeg proizvoda i od krajnjeg proizvoda do zahteva [8].

ISO/IEC/IEEE 12207 koji uspostavlja najviši nivo arhitekture za životni ciklus softvera, u definicijama svojih procesa, takođe, sadrži specifične zahteve za uspostavljanje sledljivosti. Procesi su, osim aktivnostima i njima pripadajućim zadacima, definisani nazivom, iskazom o svrsi i očekivanim rezultatima. Sledljivost je eksplicitno referencirana u iskazima očekivanih rezultata devet procesa, datih u tabeli 1, dok su zahtevi za uspostavljanje sledljivosti konkretnije formulisani u okviru odgovarajućih zadataka. Takođe, zahtev za uspostavljanje sledljivosti se javlja i u okviru definicije jednog zadatka procesa upravljanja konfiguracijom gde se nalaže da „informacija o konfiguraciji dozvoli dvosmernu sledljivost do drugih specifikovanih stanja konfiguracije“ [9]. Izazovi usaglašavanja agilnog razvoja softvera i navedenog standarda koji se odnose na dokumentovanje većine izlaza aktivnosti, identifikovani u [9], dobrim delom se odnose upravo na realizaciju sledljivosti.

Tabela 1: ISO/IEC/IEEE 12207 procesi sa iskazom o sledljivosti u očekivanim rezultatima

Proces	**Očekivani rezultati – iskaz o sledljivosti**
Tehnički procesi	
Definisanje zahteva poverioca	Postiže se sledljivost zahteva poverilaca do poverilaca i njihovih potreba
Analiza sistemskih zahteva	Uspostavlja se konzistentnost i sledljivost između sistemskih zahteva i specifikacije zahteva klijenta/korisnika.
Dizajn arhitekture sistema	Održava se konzistentnost i sledljivost između sistemskih zahteva i dizajna arhitekture sistema.
Integracija sistema	Uspostavlja se konzistentnost i sledljivost između dizajna sistema i integrisanih elemenata sistema.
Procesi implementacije softvera	
Analiza softverskih zahteva	Uspostavlja se se konzistentnost i sledljivost između softverskih i sistemskih zahteva.
Dizajn softverske arhitekture	Uspostavlja se konzistentnost i sledljivost između softverskih zahteva i dizajna softverske arhitekure.
Detaljan dizajn softvera	Uspostavlja se konzistentnost i sledljivost između detaljnog dizajna, zahteva i dizajna arhitekture.
Konstrukcija softvera	Uspostavlja se konzistentnost i sledljivost između softverskih jedinica, zahteva i dizajna.
Integracija softvera	Uspostavlja se konzistentnost i sledljivost između dizajna softvera i (integrisanih) softverskih elemenata.

Iako mnogi klijenti, agencije, standardi i sistemi kvaliteta zahtevaju uspostavljanje sledljivosti, način njene implementacije se retko definiše.

Kompanije koje nastoje da uspostave sledljivost u svojim projektima, suočavaju se sa ozbiljnim izazovima. Uspostavljanje veza zahteva značajan napor, čak i za umereno kompleksne projekte. „Iako je zastupljena određena automatizacija u uspostavljanju veza, to je još uvek uglavnom manuelan proces nelinearne kompleksnosti“ [10]. Štaviše, jednom definisane, relacije degradiraju vremenom kako sistem evaluira. Zato je neophodno njihovo kontinuirano održavanje.

Dosadašnje prakse za uspostavljanje sledljivosti najčešće su obavezivale razvojne inženjere da održavaju matricu sledjivosti koristeći softver za obradu teksta, tabelarno prikazivanje, upravljanje bazom podataka ili upravljanje zahtevima. U takvim uslovima, analitičari su odgovorni za manuelnu realizaciju, održavanje i analizu veza, kao i za utvrđivanje skupa artifakata i nivoa granularnosti na kojem će sledljivost biti uspostavljena. Sledljivost na nižem nivou granularnosti može obezbediti znatno više informacija, međutim često rezultuje velikim brojem veza koje je teško održavati, a uz to i razumeti. U praksi, matrice sledljivosti koje su pažljivo konstruisane u ranim fazama životnog ciklusa softvera, obično se odlikuju sporom, ali stalnom degradacijom, tako da postaju sve neprecizniје i nepotpunije kako razvoj sistema odmiče.

Podstaknuti ovim problemima, značajna istraživanja su posvećena razvoju metoda automatizovanog uspostavljanja sledljivosti dinamičkim generisanjem veza/linkova uz pomoć metoda pretraživanja informacija (eng. information retrieval methods) kao što su model vektorskog prostora (eng. vector space model), semantičko indeksiranje (eng. semantic indexing) ili probabilistički mrežni modeli (eng. probabilistic network models). Međutim, za sada, metodi automatskog uspostavljanja sledljivosti zahtevaju učešće analitičara koji bi analizirao linkove-kandidate i izvršio filtraciju u cilju uklanjanja suvišnih/pogrešnih kandidata [11].

3. AGILAN RAZVOJ SOFTVERA

Agilan razvoj softvera se temelji na Manifestu agilne metodologije [12] (nadalje Manifest), dokumentu koji je prihvaćen kao kanonična definicija agilnog razvoja i pridruženih agilnih principa. Agilne metode su familija razvojnih procesa, baziranih na iterativnom razvoju, a ne jedinstven pristup, gde zahtevi i rešenja evoluiraju kroz kolaboraciju samoorganizujućih timova. Neke od agilnih metoda opisanih u literaturi su: Scrum, Ekstremno programiranje, Kristalne metodologije itd. Svaka agilna metoda ima svoje karakteristike i osobenosti , ali se u svim nastoje primeniti principi agilnog razvoja kao što su: iterativnost, rana isporuka funkcionalnog softvera, jednostavnost, stalna neformalna saradnja kako između članova projektnog tima, tako i između projektnog tima i korisnika, prilagodljivost promenama itd.

Agilni razvoj softvera je doneo neke značajne promene u odnosu na sistem inženjerski pristup. Verovatno je najuočljivija

razlika manja orijentisanost na dokumentaciju. On je, na više načina, orijentisan na kod i sledi stav da je ključni deo dokumentacije izvorni kod. To podrazumeva drugačije artifakte u odnosu na one koji se najčešće sugerišu u definicijama sledljivosti.

Aspekti uspostavljanja sledljivosti u kontekstu agilnog razvoja softvera, razmotreni u daljem tekstu, odnose se prevashodno na dve popularne i dobro dokumentovane metodologije, Ekstremno programiranje (nadalje XP) i Scrum.

Scrum je uglavnom projektno orijentisana agilna metodologija. Pruža širi pogled na životni ciklus softvera jer nije usko fokusirana na samu implementaciju. Za razliku od Scrum-a, XP ima fokus na tim i razvojne prakse. To ih čini odličnim parterima, zbog čega njihova zajednička primena nije retkost.

U XP-u se funkcionalnosti koje treba realizovati specifikuju u vidu korisničkih priča koje predstavljaju zahteve visokog nivoa. Posebna vrednost i značaj se pridaje bliskoj saradnji sa klijentom. XP zahteva predstavnika klijenta na licu mesta (eng. „on-site“ customer) koji igra značajnu ulogu u postavljanju i prioritetizaciji korisničkih priča. U osnovi XP-a je razvoj vođen testovima (eng. Test Driven Development, nadalje TDD). Naime, slučajevi testiranja se pišu pre kodiranja i kod se ne prihvata ukoliko nije verifikovan testiranjem. Korisničke priče i neuspeli testovi prihvatljivosti se u toku planiranja iteracije (jedna do tri nedelje) razlažu na programerske zadatke koji se, kao i korisničke priče, zapisuju na indeksiranim karticama (eng. index cards). Karakteristične prakse u XP-u su i: programiranje u paru, kolektivno vlasništvo koda, refaktorisanje, stalna integracija itd., a dominantni artifakti: korisničke priče, zadaci, kod, testovi/slučajevi testiranja (jedinični, testovi prihvatljivosti) [13].

Proces prikupljanja zahteva po Scrum metodologiji, može se uporediti sa odgovarajućim u XP-u. Projekat se odvija u iteracijama – sprintovima koji traju najviše trideset dana. U toku planiranja sprinta, vlasnik proizvoda (eng. product owner) koji je predstavnik klijenta, opisuje članovima tima funkcije najvišeg prioriteta. Tim na čelu sa Scrum masterom, odlučuje koje će funkcije biti isporučene u predstojećem sprintu. Zanimljivo je to da u Scrum procesu vlasniku proizvoda nije dozvoljeno da menja prioritete zahteva dok se sprint ne završi [14].

Karakteristični artifakti u Scrum metodologiji su [14]:

– „Product backlog“ – dokument visokog nivoa koji se odnosi na čitav projekat. Sadrži okviran opis svih zahtevanih funkcija, listu želja itd. prioritetizovanih prema poslovnim vrednostima.
– „Sprint backlog“ – dokument o načinu realizacije funkcija u predstojećem sprintu. Funkcije su razložene na zadatke/task-ove. Često je u upotrebi tabla (eng. Task Board) za praćenje statusa zadataka u aktuelnom sprintu (npr. „to do“, „in progress“, „done“), prisutna i u XP-u.
– „Burn Down Chart“ – javno prikazan grafikon koji prikazuje preostali rad u sprint backlog-u. Ažurira se svakodnevno tako da pruža jednostavan uvid u napredak sprinta. Mogu biti zastupljeni i drugi tipovi grafikona (npr. „Release Burn Down Chart”).

4. SLEDLJIVOST U AGILNOM PROJEKTU: PRISTUPI, PRAKSE, TEHNOLOŠKI ASPEKTI

Zbog karakterističnih softverskih artifakata i redosleda kojim se produkuju, potencijalna sledljivost u agilnom projektu donosi kako značajne mogućnosti, tako i specifične izazove.

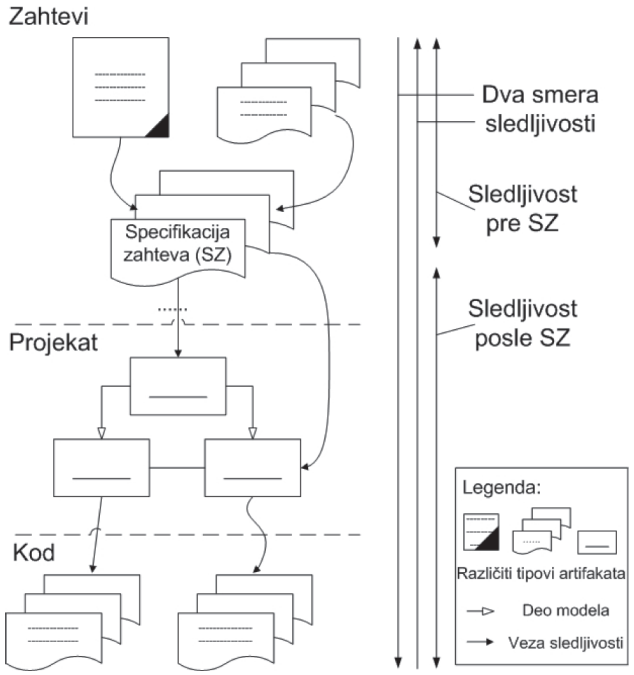
Različiti pristupi problemu sledljivosti su istraženi i publikovani u brojnim radovima. Zajedničko za većinu pristupa je to da implementiraju hijerarhijski model. Sintezom literature i iskustva, pristupi se mogu klasifikovati na sledeći način [8]: saradnja i zajedničko/deljeno znanje tima, alati opšte namene, matrice sledljivosti, pojedinačni artifakti sa „end-to-end“ vezama sledljivosti (kombinacija prethodna dva modela), rešenja zasnovana na alatima za upravljanje zahtevima, integrisana razvojna okruženja.

Saradnja i deljeno znanje tima je pristup karakterističan za manje timove, uglavnom zastupljen kod primene agilnih metodologija i rada u okruženju intezivnih promena. Prema [8], agilne metodologije realizuju „lean“ pristup sledljivosti zasnovan na znanju razvojnih inženjera o sistemu i upotrebi postojećih agilnih artifakata. Ova strategija sledljivosti može funkcionisati zadovoljavajuće u agilnom okruženju sa manjim timovima, bliskom saradnjom i deljenim vlasništvom nad softverom. Međutim, savremene tendencije i izazovi sa kojim se se suočavaju predstavnici agilnih metodologija su više nego podsticajni za traženje i primenu novih, efektivnijih rešenja kojim vrednosti i principi agilnog razvoja ne bi bili narušeni.

U [15] se zastupa stav da je poboljšanje sledljivosti u agilnom razvoju softvera neophodno, pri čemu ono ne mora biti nužno na „agilan način“, ali može i treba biti prilagođeno agilnom razvoju softvera i što manje nametljivo, odnosno preporučuje se tzv. „lean“ sledljivost. Zaključeno je da su sveobuhvatni ciljevi sledljivosti: transparentnost (sposobnost da se lako pristupi svim informacijama od interesa) i identifikacija (sposobnost da se identifikuju informacije od interesa tako da se mogu odvojiti nezavisni skupovi i kohezivno pridruže povezani). Takođe je naglašeno da sledljivost treba da bude u funkciji transparentnosti, preglednosti i upravljanja statusima, a ne sama sebi cilj.

U agilnom razvoju softvera jedinice rada se definišu kao korisničke priče ili zadaci. Indeksirane kartice (eng. Index cards) su značajan alat za upravljanje dijalogom i interakcijom sa klijentima, ali same indeksirane kartice nisu dovoljne za čuvanje, praćenje, sortiranje, pretraživanje i upravljanje promenom zahteva [15]. U okviru agilne paradigme, sledljivost između zahteva i dizajna/koda se realizuje preko jediničnih i testova prihvatljivosti. Testovi odražavaju sledljivost zahteva, jer se odnose na konkretan kod koji implementira zahteve [16]. Ukoliko se primenjuje TDD, razvojni inženjer počinje pisanjem slučajeva testiranja da bi onda realizovao tek onoliko programskog koda koliko je dovoljno da se prođe test. „Beleženjem relacija sledljivosti“, na primer, „između korisničkih priča, slučajeva testiranja i izvornog koda“, ali i drugih artifakata, „moguće je mapirati informacije iz koncepta visokog nivoa, poput funkcije ili zadatka, do izvornog koda“ [17], i obrnuto. Ipak, značajno pitanje koje se nameće, tiče

se načina „beleženja relacija“, koji bi bio u skladu sa agilnim razvojem. Na slici 3 (prilagođena prema [18]), prikazani su efektivan i neefektivan pristup sledljivosti u agilnom kontekstu. Neefektivan pristup podrazumeva nepostojanja potrebnih veza sledljivosti, kao i pogrešno uspostavljene veze između artifakata. Kvalitet uspostavljenjih veza ne sme da opada kako razvoj sistema odmiče.



Slika 3. – *Efektivan i neefektivan pristup sledljivosti u agilnom kontekstu*

Formalna matrica sledljivosti, bila je često predmet oštre kritike pristalica agilnog razvoja softvera. Prema [16], ključno je razumeti razliku između sledljivosti i matrice sledljivosti. Naime, sledjivost je željena karakteristika projekta u kojem se naglasak stavlja na transparentnost i preglednost značajnih informacija i njihovih odnosa. Matrica sledljivosti, sa druge strane, predstavlja jednu od mogućih implementacija sledljivosti. „Matrica sledljivosti je potrebna samo onda kada postoji opasnost od gubitka korporativne memorije koja je istinski vredna. Ako to nije slučaj, onda je sledjivost dostupna u mnogim oblicima kao što su konverzacije, korisničke priče, strateške teme, istorije, logovi, „jurnal“ baze, izvorni kod, automatski testovi, projektna dokumentacija, dnevni „scrum“-ovi, elektronska pošta itd [16].

Postoje održive alternative za postizanje transparentnosti i identifikacije, osim onog što mnogi smatraju formalnom sledljivošću. Neke od preporuka za uspostavljanje „lean“ sledljivosti su: prepoznavanje razlike između koncepta i konkretnih rešenja, primena alata za verzioniranje i praćenje promena, osnovna integracija između verzioniranja i praćenja promena, razvoj zasnovan na zadacima (eng. Task-Based Development - TBD), razvoj vođen testovima (eng. Test-Driven Development - TDD), primena jednostavnih alata (Wiki, Wiki-based Specification Framework), sledljivost zasnovana na događajima (eng. Event-Based Traceability - EBT), sledljivost zasnovana na pretrazi (eng. Search-Besad Traceability), sledljivost zasnovana na jednostavnoj pretrazi (eng. Simple Search-Based Traceability) [17].

Primena razvoja zasnovanog na zadacima pomaže organizovanje strukture rada u sistemu za verzioniranje tako da je moguća integracija sa sistemom za praćenje. Zadaci su zajednička nit koja povezuje zahteve za izlazima drugih procesa. Ukoliko se svi artifakti kreirani tokom životnog ciklusa softvera (zahtevi, kod, testovi itd.) nalaze u repozitorijumima sistema za verzioniranje povezani na ovaj način, onda je obezbeđena mogućnost za realizaciju jednog značajnog vida sledljivosti.

Razvoj vođen testovima u kombinaciji sa razvojem zasnovanim na zadacima može obezbediti još snažniju podršku sledljivosti. Razlog za to je orijentisanost TDD-a na „sitnije“ aktivnosti koje treba da rezultuju artifaktima strogo neophodnim za kompletnu i blagovremenu implementaciju zahteva.

Veliki broj alata danas obezbeđuje prilično moćna razvojna okruženja za realizaciju artifakata (razvoj softvera). Međutim podrška je obično ograničena na određen deo životnog ciklusa: upravljanje projektom, upravljanje zahtevima, upravljanje kodom, upravljanje testovima, modelovanje softverske arhitekture i generisanje koda. Ipak, kompleksnost koju oni donose može biti problematična, ukoliko svaki od alata koristi sopstvene formate za arhiviranje i mehanizme verzioniranja. Uspostavljanje sledljivosti između srodnih artifakata u jednom repozitorijumu, moglo bi biti jednostavno, međutim sledljivost između artifakata različitih faza životnog ciklusa kroz različite repozitorijume, verovatno bi donela probleme veće nego primitivniji načini realizacije u jednostavnim formatima zasnovanim na tekstu i istom alatu za verzioniranje. Iz ovog razloga, preporučuje se wiki-web zasnovan na tekstu sa mogućnostima verzioniranja. On može biti veoma pogodan alat za uspostavljanje sledljivosti između koda i drugih artifakata. Artifakti su jednostavno kreirani, na hijerarhijski način, sa hiperlinkovima koji vode do drugih relevantnih informacija. Na ovoj osnovi, moguće je nadograditi okvire za testiranje prihvatljivosti kao što su FIT i FitNesse [19].

FIT i FitNesse su pokušaji da se upotrebe izvršivi testovi kao mehanizam za specifikaciju softverskih zahteva. U [17] se naglašava da izvršivi testovi nisu adekvatna zamena dokumentovanim zahtevima, ukoliko je kod jedina dokumentacija. Međutim, ako su zahtevi specifikovani na način koji je ujedno jednostavan i SMART (akronim engleskih reči – **s**pecific (specifičan), **m**easurable (merljiv), **a**ttainable (dostižan), **r**elevant/**r**ealizable (relevantan/ostvariv), **t**ime-bound (vremenski ograničen)), to može učiniti njihovo povezivanje sa testovima trivijalnim.

Kodiranje i testiranje su dve blisko povezane aktivnosti u agilnom razvoju softvera koje zahtevaju da se razvojni inženjeri/programeri često „premeštaju“ između artifakata kodiranja i testiranja. Nažalost, veze između ovih artifakata su obično implicitno prisutne u izvornom kodu zbog čega su

razvojni inženjeri/programeri prinuđeni na vremenski zahtevne preglede.

Kada su u pitanju jedinični testovi, postojeći nivo sledljivosti između izvornog koda i jediničnih testova je na nezadovoljavajućem nivou [19]. Današnja integrisana razvojna okruženja (eng. Integrated Development Environment, nadalje IDE) nude malo podrške razvojnim inženjerima za pretragu jediničnih testova i koda. Čak i kada postoji podrška IDE-a za kreiranje testova, čvrsta, eksplicitna veza između ovih artifakata se retko održava. U ovakvim uslovima, praćenje promena se svodi na pretraživanje (uparivanje naziva) i manuelnu proveru. U više standardizovanim okruženjima (npr. xUnit), razvojni inženjeri, zahvaljujući vodičima i konvencijama koje karakterišu takva okruženja, na primer za fizičku lokaciju uskladištenja artifakata izvornog koda i testiranja, konvencije imenovanja, reference izvornog koda, mogu tražiti sopstvene načine za uspostavljanje odnosne sledjivosti. Stoga, ne čudi što je istraživački interes u velikoj meri usmeren na strategije automatizovane sledjivosti kao što su: konvencija imenovanja (eng. Naming Convention – NC), tipovi „fixture“ elemenata (eng. Fixture Element Types – FET), graf statičkih poziva (eng. Static Call Graf – SCG), poslednji poziv pre potvrde (eng. Last Call Before Assert – LCBA), leksička analiza (eng. Lexical Analysis – LA), ko-evolucija (eng. Co-Evolution – Co-EV). Neka od navedenih rešenja su inicirana u samom kodu, dok neka koriste informacije izvedene iz operativnog test paketa (eng. running test suite), kao i logova sistema za verzioniranje. Prema [20], strategija bazirana na konvencijama imenovanja rezultuje visokom efikasnošću i preciznošću, međutim zavisi od projektnih smernica i discipline razvojnih inženjera. Sa druge strane, strategije poput LCBA, LA ili Co-Ev imaju visoku primenjivost, ali slabiji rezultat u pogledu tačnosti. Kombinovanje navedenih strategija, omogućava najbolji ukupni rezultat, ali zahteva investiciju u alate, kao i investiciju u znanje o konvencijama kodiranja, razvojnoj metodologiji i dizajnu test paketa u cilju njihovog adekvatnog konfigurisanja i primene odgovarajućih tehnika na projektu.

Sledljivost zasnovana na događajima omogućava automatsko kreiranje veza zahvaljujući ne samo čvrstoj integraciji alata za verzioniranje i praćenje, nego i kontekstu rada koji može biti izveden iz IDE-a. Akcije i događaji iz IDE-a, mogu trigerovati akcije i događaje u drugim alatima i zajedno sa kontekstno specifičnim informacijama o kodu koji je izmenjen, kao i njegovoj strukturi, mogu biti zabeleženi u nekom strukturalnom formatu (npr. XML-u). Upiti i filteri, bili bi korišćeni ne samo za generisanje izveštaja o sledljivosti, već i za selekciju značajnih događaja i zahteva, te definisanje granularnosti.

Sledljivost zasnovana na pretrazi predstavlja jednu od aktuelnih tema vezanih za istraživanje sledljivosti. Ona se ne odnosi na manuelno, čak ni automatsko generisanje veza. Umesto toga, oslanja se na pametnoj (eng. smart), probabilističkoj i kontekstno senzitivnoj pretrazi projektnih informacija kako bi se dinamički otkrila povezanost i obezbedilo izveštavanje na osnovu određenog skupa ključnih reči.

Sledljivost bazirana na događajima i integraciji alata, evidentno, ostaje pogodnija i efektivnija za povezivanje informacija. Međutim, sledljivost zasnovana na pretrazi, mogla bi doneti jednu novu vrednost koja se ogleda u otkrivanju odgovora na pitanja „zašto“ i analizi uticaja i veza logičkih objekata i termina kroz projektnu bazu znanja (uključujući i slabije formalno strukturirane informacije kao što su blogovi, „mail“ liste, „wiki“-ji itd.). U svom najnaprednijem obliku, sledljivost zasnovana na pretrazi koristi ranije pomenute metode za pretraživanje informacija poput modela vektorskog prostora, semantičkog indeksiranja ili probabilističkih mrežnih modela za dinamičko generisanje veza. Međutim, dok razvoj ovakvih naprednih mehanizama pretrage ne dosegne adekvatan nivo, trenutno dostupne mašine za pretragu (eng. search-engine) i jednostavni alati, zajedno sa Wiki-Web aplikacijama, izvornim kodom i „dokumentacionim“ repozitorijumima, mogli bi biti uključeni u realizaciju sledljivosti u agilnom projektu.

Na tržištu je sve više alata namenjenih pružanju podrške upravljanju projektom agilnog razvoja softvera koji bi mogli obezbediti sledjivost između artifakata projektnog menadžmenta, i šire, zahvaljujući integraciji sa razvojnim okruženjima i drugim alatima. Jedno od sve popularnijih rešenja jeste Atlassian JIRA, alat za praćenje promena (eng. Issue and Project Tracking) zahvaljujući „plug-in“-u koji obezbeđuje GreenHopper. U glavnim crtama, nastojanja su usmerena ka omogućavanju podrške za upravljanje „backlog“-ovima, upravljanje indeksiranim karticama, njihovu prioritetizaciju i praćenje statusa jednostavnim pomeranjem kartica po virtuelnoj tabli. Takođe, savremeni alati nude funkcionalnosti za realizaciju dinamičkih „burndown“ grafikona, kao i različitih statističkih pregleda u cilju praćenja napretka projekta. Međutim, zamerke koje se obično upućuju na račun ovakvih alata, odnose se na nedovoljnu usklađenost podržanog toka procesa sa uobičajenim praksama i slabu podršku u vidu pojašnjenja mogućnosti postojećih funkcija i očekivane primene [7].

Tendencija integracije alata namenjenih različitim fazama životnog ciklusa softvera deluje obećavajuće po pitanju uspostavljanja sledljivosti između artifakata projektnog menadžmenta i sistema u razvoju. Tržište alata za upravljanje životnim ciklusom aplikacija (eng. Application Lifecycle Management - ALM) za agilni razvoj softvera je u velikoj ekspanziji (Rally Software, CollabNet itd.) [21]. ALM platforme su „end-to-end“ rešenja za upravljanje svim fazama životnog ciklusa softvera, od prikupljanja inicijalnih zahteva do krajnje isporuke, kako iz poslovne, tako i iz tehničke perspektive. Izgrađene na centralnom repozitorijumu koji se ponaša kao kičma čitave platforme, trebalo bi da ALM aplikacije omoguće sledljivost između artifakata sa različitih lokacija i tipova alata, kroz različite procese. Međutim, glavne teškoće, još jednom, vezane su za integraciju – alata, procesa, ljudi. Na primer, pokazalo se da se različiti alati u okviru takvih paketa često ne integrišu nimalo bolje od alata različitih vendora i da mere prilagođavanja, ako se zahtevaju od korisnika, obično nisu trivijalne [22].

5. OPŠTI IZAZOVI I SMERNICE

Uvođenje efektivnih praksi sledljivosti je praćeno brojnim izazovima, i to ne samo kod primene agilnih metodologija, i zahteva saradnju i disciplinu svih učesnika u razvojnom ciklusu. Neki od ključnih izazova su:

– izbor adekvatnog nivoa granularnosti;
– fragmentacija podrške životnom ciklusu softvera na veći broj alata, s tim u vezi, nedostatak uniformnosti repozitorijuma koji pohranjuju artifakte i njihova otežana integracija;
– nedovoljno poznavanje samog koncepta i ukorenjena predrasuda o sledljivosti kao o nepotrebnom, neefikasnom naporu, pogotovo suvišnom u agilnom projektu;
– upravljanje koristima – nepostojanje uvida u celu sliku (ljudi uglavnom vide svoj deo procesa koji nastoje da optimizuju).

Ne postoji jedinstveno rešenje za uspostavljanje sledljivosti. Različiti projekti zahtevaju različite tipove artifakata i različitu strukturu raspodele rada. Razvojna metodologija, veličina tima i kompleksnost projekta, kao i tehnologija, treba da opredele primenu strategije sledljivosti. Uprkos dokazanim prednostima, mnoge organizacije ne razmatraju dodatne mogućnosti realizacije sledljivosti zbog rasprostranjenog uverenja o njenoj neefektivnosti i troškovnoj neefikasnosti. Međutim, izborom odgovarajuće implementacione strategije odnos troškova i koristi efektivne sledljivosti može biti optimiziran, čak i u agilnom projektu. Ukoliko bi bilo dopune Manifesta u pogledu sledljivosti, ona bi verovatno glasila: „Pouzdana transparentnost, pre nego zamorna sledljivost“ [17]. To, međutim, ne znači da u agilnom projektu nisu potrebne efektivnije prakse za uspostavljanje sledljivosti, već da su potrebne prakse koje su u skladu sa vrednostima i principima agilnog razvoja, a pri tom mogu obezbediti adekvatan odnos troškovi-koristi.

Ključni tehnološki aspekti u kojima su sadržane preporuke za implementaciju efektivnog rešenja sledljivosti, primenjive i na agilni kontekst, uključuju povezivanje podataka, semantički sadržaj, verzioniranje artifakata i omogućavanje automatizacije [8]. Tome se može eksplicitno pridodati podrška alata za upravljanje agilnim projektom prilagođenih realnim procesima, kao i omogućavanje integracije heterogenih alata.

„Povezivanje podataka“ se odnosi na način međusobnog referenciranja artifakata. Na primer, skriptovi testova mogu nositi identifikatore zadataka. Omogućavanje automatizacije je možda najznačajniji tehnološki aspekt sledljivosti. Cilj je inkorporirati prakse sledljivosti u postojeće aktivnosti što je moguće manje nametljivo. Zbog toga je poslednjih godina automatizacija sledljivosti predmet brojnih istraživanja. Verzioniranje artifakata je neophodno kako bi se osiguralo da sistem sledljivosti u svakom trenutku može identifikovati i prikazati artifakat (npr. zadatak, kod, test) koji je predmet sledljivosti. Semantički sadržaj može biti najteži, ali i najdelotvorniji deo automatizovane sledljivosti jer omogućava izdvajanje značenja zajedno sa podacima, kao i vršenje analiza. Sledljivost se ne sme ograničiti na artifakte softverskog sistema u razvoju, već mora uključiti i artifakte projektnog menadžmenta. Iz tog razloga, neophodno je dalje usavršavanje alata za upravljanje projektom agilnog razvoja softvera, tako da se obezbedi jednostavno usklađivanje podržanih procesa sa realnim procesima u praksi. Međutim, i adekvatnom primenom postojećih alata, moguće je doći do značajnih prednosti. Problem integracije heterogenih alata i neuniformnih repozitorijuma koji pohranjuju artifakte je kompleksan, ali nudi se alternativa u vidu primitivnijih načina realizacije artifakata koji podrazumevaju jednostavne formate zasnovane na tekstu i istom alatu za verzioniranje. Uz sve navedeno, treba još pomenuti da poboljšanje upotrebljivosti sledljivosti preko odgovarajuće realizacije upita, navigacije i tehnika vizualizacije, može biti značajan aspekt kojem se posvećuje sve više pažnje.

Sledljivost je postala prepoznati atribut kvaliteta softvera. Posledično, postoje dva glavna motiva koja opredeljuju organizacije da realizuju prakse sledljivosti: stroga regulativa karakteristična za određene domene poslovanja, ili obaveza implementacije određenog standarda, uključujući sertifikaciju ili ne (najčešće zbog zahteva klijenata). Mnogo ređe, motiv je usvajanje realnih prednosti koje sledljivost može da donese kao što su, poboljšan kvalitet softverskog proizvoda i viša efikasnost u njegovoj realizaciji. Međutim, motiv za usvajanje praksi sledljivosti, često je od suštinskog značaja. Ukoliko se na sledljivost gleda kao na nekoristan teret nametnut regulativom, zahtevima sponzora ili klijenata, pozitivni rezultati mogu izostati kao posledica nedovoljnog ulaganja ili neodgovarajućeg kanalisanja resursa za uspostavljanje sledljivosti. Sa druge strane, organizacije koje su uverene u korisnost sledljivosti, ulažu u strategije, tehnologiju i alate za njeno efektivno i efikasno uspostavljanje. Ipak, treba imati u vidu da se prakse sledljivosti ne realizuju na slepo, već se ulaže u evaluaciju alternativa, razvoj scenarija korisnosti, optimizaciju koristi i prilagođavanje zahteva sledljivosti konkretnim potrebama. To je ujedno preporuka agilnim organizacijama koje se i same sve više suočavaju sa navedenim izazovima.

6. ZAKLJUČAK

Sledljivost je željena karakteristika softverskih projekata, generički pojam koji treba razlikovati od konkretnih rešenja, naročito „formalnih“, neprimenjivih na agilni kontekst. Poboljšanje sledljivosti u agilnom projektu je neophodno, pri čemu ono može i treba da bude u skladu sa vrednostima i principima agilnog razvoja, i pri tome, što manje nametljivo. Traganje za efektivnim rešenjima sledljivosti, zbog karakteristika agilnih procesa, artifakata i redosleda kojim se produkuju, praćeno je specifičnim izazovima, ali može rezultovati višestrukim pozitivnim implikacijama. Razvoj tehnologije donosi unapređenje postojećih i nove obećavajuće pristupe, no i u aktuelnim uslovima, izbor implementacione strategije koja odgovara konkretnom projektu je ključan za efektivnu i troškovno efikasnu sledljivost.

LITERATURA

[1] http://www.sei.cmu.edu/cmmi/, avgust 2010.

[2] Kannenberg, A., Saiedian, H.: *Why Software Requirements Traceability Remains a Challenge*, CrossTalk – The Journal of Defense Software Engineering, http://www.stsc.hill.af.mil/CrossTalk/, Jul/Aug 2009.

[3] Dimitrijević, S., Pantelić, D. S.: *Pogled na procese životnog ciklusa softvera kod primene agilne metodologije*, XXV ICT Konferencija i izložba INFOTECH 2010, str. 091 od 1 do 6, Srbija, Vrnjačka Banja, 2010.

[4] Gotel, O. C. Z., Finkelstein, C. W.: *An Analysis of the Requirements Traceability Problem*, the First International Conference on Requirements Engineering, pp. 94-101, Colorado Springs, USA, 1994.

[5] Winkler, S., Von Pilgrim, J.: *A survey of traceability in requirements engineering and model-driven development* (theme section in book Software and Systems Modeling), Springer, 2009.

[6] Helming, J., Koegel, M., Naughton, H.: *Towards Traceability from Project Management to System Models*, 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering, TEFSE '09, pp. 11-15, Vancouver, Canada, 2009.

[7] Delgadillo, L.: *Story-Wall Lightwight Requirements Management for Agile SW Development*, Student/Faculty Research Day, CSIS, Pace University, New York, 2007.

[8] Kirova, V., Kirby, N., Kothari, D., Childress, G.: *Effective requirements traceability: Models, tools, and practices*, Bell Labs Technical Journal, vol. 12 , Issue 4, pp. 143-157, 2008.

[9] ISO/IEC 12207:2008 (IEEE Std 12207-2008) – Systems and software engineering – Software life cycle processes, ISO/IEC JTC1 and IEEE Computer Society, 2008.

[10] Egeyda, A., Grunbacher, P., Heindl, M., Biffl, S.: *Value-Based Requirements Traceability: Lessons Learned*, 15th IEEE International Requirements Engineering Conference (RE 2007), pp. 115-118, Delhi, India, 2007.

[11] Cleland-Huang, J.: *Just Enough Requirements Traceability*, Computer Software and Applications Conference, 2006. COMPSAC '06. 30th Annual International, pp. 41-42, Chicago, USA, 2006.

[12] http://agilemanifesto.org/, avgust 2010.

[13] http://www.extremeprogramming.org/, avgust 2010.

[14] http://www.scrumalliance.org/, avgust 2010.

[15] Appleton, B.: *The Trouble with Tracing: Traceability Dissected*, CM Crossroads, 2005, http://www.cmcrossroads.com/agile-scm/6685-the-trouble-with-tracing-traceability-dissected, avgust 2010.

[16] Ratanotayanon, S., Elliott Sim, S., Gallardo-Valencia, R.: *Supporting Program Comprehension in Agile with Links to User Stories*, 2009 Agile Conference, AGILE '09, pp. 26-32, Chicago, USA, 2009.

[17] Appleton B., CowHam R., Berczuk S.: *Lean Traceability: a smattering of strategies and solutions*, CM Crossroads, 2007, http://www.cmcrossroads.com/agile-scm/9089-lean-traceability-a-smattering-of-strategies-and-solutions, avgust 2010.

[18] Lee, C., Guadaqno, L., Jia, X.: *An Agile Approach to Capturing Requirements and Traceability*, 2003 ICSE Workshop on Traceability in Emerging Forms of Software Engineering, TEFSE '03, pp. 17-23, Montreal, Canada, 2003.

[19] Ghanam, Y., Maurer, F.: *Extreme Product Line Engineering: Managing Variability and Traceability via Executable Specifications*, Agile Conference, AGILE '09, pp. 41-48, Chicago, USA, 2009.

[20] Van Rompaey, B., Demeyer, S., *Establishing Traceability Links between Unit Test Cases and Units under Test*, 2009 European Conference on Software Maintenance and Reengineering, CSMR, pp.209-218, 2009.

[21] Goth, G., *Agile Tool Market Growing with the Philosophy*, IEEE Software, vol. 26, no. 2, pp. 88-91, Mar./Apr. 2009.

[22] Weatherall, B.: *The "Best of Breed" Agile ALM Solution*, http://www.accurev.com/product-reviews/ALMSolution-Weatherall.pdf, avgust 2010.

Istraživanja prikazana u ovom radu su realizovana u okviru Projekata TR 14021 i TR 13004, podržanih programom Tehnološkog razvoja Ministarstva nauke i tehnološkog razvoja Republike Srbije.

Sonja Dimitrijević, Institut „Mihajlo Pupin", Beograd
Kontakt: sonja.dimitrijevic@pupin.rs
Oblasti interesovanja: softversko inženjerstvo, upravljanje poslovnim procesima, upravljanje procesom razvoja softvera, standardizacija

Snežana D. Pantelić, Institut „Mihajlo Pupin", Beograd
Kontakt: snezana.pantelic@pupin.rs
Oblasti interesovanja: IT menadžment, upravljanje poslovnim procesima, upravljanje procesom razvoja softvera, projektovanje IS