INFORMACIJSKA DRUŽBA – IS´2009
16. oktober 2009
VZGOJA IN IZOBRAŽEVANJE V INFORMACIJSKI DRUŽBI

Poučevanje metode Scrum v sodelovanju s podjetjem za razvoj programske opreme

Teaching the Scrum method in cooperation with a software development company

Viljan Mahnič[1], Strahil Georgiev[2], Tomo Jarc[2]

[1]Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Tržaška 25, Ljubljana
viljan.mahnic@fri.uni-lj.si

[2]SRC Sistemske integracije, d.o.o., Tržaška 116, Ljubljana
strahil.georgiev@src.si, tomo.jarc@src.si

Povzetek

Vse večja uporaba agilnih metodologij za razvoj programske opreme zahteva, da učenje teh metodologij postane sestavni del izobraževanja bodočih inženirjev računalništva in informatike. Po drugi strani pa je možno skozi poučevanje teh metodologij preveriti tudi posamezne agilne koncepte in poiskati natančnejše odgovore na vprašanja o njihovi učinkovitosti. Zato se kot najprimernejša oblika poučevanja pogosto uporablja delo na projektih, ki omogočajo, da študenti v praksi spoznajo značilnosti agilnega pristopa, obenem pa služijo kot študije primera za ovrednotenje posameznih agilnih konceptov. V članku opisujemo, kako smo v sklopu predmeta Tehnologija programske opreme izpeljali učenje agilne metode Scrum v sodelovanju s podjetjem za razvoj programske opreme. Učenje je potekalo ob delu na realnem projektu, za katerega je seznam zahtev posredovalo podjetje, sodelavec tega podjetja pa je ves čas sodeloval s študenti kot predstavnik naročnika. Študenti so pri svojem delu dosledno uporabljali metodo Scrum in na koncu vsake iteracije s pomočjo ankete ocenili svoje izkušnje. V članku je najprej na kratko predstavljena metoda Scrum, nato sledi opis poteka dela na projektu, na koncu pa so predstavljeni rezultati ankete.

Ključne besede: agilne metodologije, Scrum, razvoj programske opreme, izobraževanje inženirjev računalništva, sodelovanje univerze z gospodarstvom

Abstract

The increasing use of agile methods for software development creates need for these methods to become part of education of future computer and information science engineers. On the other hand, teaching these methods gives us an opportunity to verify individual agile concepts and their effectiveness. For that reason, project work is appropriate and frequently used form of teaching that enables students to get acquainted with agile methods and, at the same time, provides case studies for evaluating individual agile concepts. We describe our approach to teaching the

Scrum agile method, within the software technology course, in cooperation with a software development company. Students were taught through work on a real project for which a list of requirements was submitted by the company. A co-worker of this company participated throughout the teaching period playing the role of customer's representative. During their work, students consistently used the Scrum method and at the end of each iteration they evaluated their experience by means of a questionnaire. In the article the Scrum method is presented first, then a description of work on the project is given and finally the results of the survey are described.

Keywords: agile methods, Scrum, software development, computer engineering education, university-industry co-operation

1 Uvod

Agilne metodologije (Abrahamsson et al., 2002) dobivajo vse večjo vlogo pri razvoju programske opreme. Rezultati ankete, ki jo je objavil Dr. Dobb's Journal leta 2008 (Ambler, 2008) kažejo, da se z uvedbo teh metodologij povečajo produktivnost, kakovost in zadovoljstvo vseh udeležencev v razvojnem procesu. Primerjava med podjetji, ki za razvoj programske opreme uporabljajo agilni pristop, in podjetji, ki se poslužujejo tradicionalnega discipliniranega pristopa (Ceschi et al., 2005), je pokazala, da uporaba agilnega pristopa izboljša vodenje projektov in odnose z uporabniki. Kljub številnim pozitivnim izkušnjam pa še vedno obstajajo dvomi o uspešnosti agilnih metodologij, češ da le-te z uporabo nekaterih tipičnih praks v ekstremni obliki vnašajo nestabilnost in povečujejo tveganje.

Za izobraževanje bodočih inženirjev računalništva in informatike je nedvomno pomembno, da jim v času študija posredujemo ustrezna znanja o uporabi agilnih metodologij. Zaradi pogosto nasprotujočih si mnenj o njihovi uspešnosti pa nudi poučevanje te tematike tudi veliko možnosti za preverjanje posameznih konceptov v praksi. Zato predmeti, ki obravnavajo razvoj programske opreme, pogosto vključujejo delo na projektih, ki so čim bolj podobni realnim, da bi študenti ob praktičnem delu spoznali prednosti in slabosti agilnega pristopa. Kot primere takega pristopa lahko omenimo poučevanje ekstremnega programiranja (Shukla & Williams, 2002; Dubinsky & Hazzan, 2003), preverjanje učinkovitosti testno vodenega razvoja in programiranja v parih (Xu & Rajlich, 2006) ter poučevanje razlik med agilnim in discipliniranim pristopom k razvoju programske opreme (Robillard & Dulipovici, 2008).

V študijskem letu 2008/09 smo se za tak pristop odločili tudi pri predmetu Tehnologija programske opreme na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Predmet poslušajo študenti univerzitetnega programa Računalništvo in informatika, smer Programska oprema, v zadnjem (osmem) semestru študija. Kot osnovo smo izbrali agilno metodo Scrum (Schwaber, 2004), ki je poleg ekstremnega programiranja najbolj razširjena med vsemi agilnimi metodami. Naš cilj je bil preizkusiti to metodo v čim bolj realnih okoliščinah, obenem pa empirično spremljati učinkovitost razvojnega procesa s pomočjo metrik, definiranih v (Mahnič & Vrana, 2007).

Da bi študentom omogočili delo na čim bolj realnem projektu smo se povezali s podjetjem SRC, ki je eno vodilnih slovenskih podjetij za razvoj programske opreme. SRC je za potrebe predmeta posredoval specifikacije zahtev za projekt "Informacijski sistem splošne bolnice" in zagotovil sodelovanje enega izmed svojih sodelavcev, ki je pri študentskem projektu sodeloval kot *Product Owner* – v terminologiji metode Scrum je to predstavnik naročnika oziroma poznavalec problemske domene. Naloga študentov je bila z dosledno uporabo

metode Scrum realizirati zahteve, ki so jih dobili, obenem pa beležiti rezultate meritev, s pomočjo katerih lahko računamo indikatorje učinkovitosti razvojnega procesa.

V nadaljevanju našega prispevka bomo najprej na kratko predstavili glavne značilnosti agilnih metodologij in opisali metodo Scrum. V tretjem delu bomo podrobno predstavili študentski projekt, ki je služil kot študija primera za učenje metode Scrum in vpeljavo metrik, s pomočjo katerih lahko spremljamo učinkovitost razvoja in zgradimo ustrezen repozitorij, skladno z zahtevami modela CMMI (Mahnič & Žabkar, 2007). Četrti del bo posvečen analizi rezultatov ankete, s katero smo analizirali zadovoljstvo študentov z metodo Scrum in načinom dela na projektu. Na koncu pa bomo navedli še najpomembnejše rezultate in izkušnje, ki smo jih pridobili z opisanim pristopom.

2 Agilne metodologije in metoda Scrum

2.1 Glavne značilnosti agilnih metodologij

Agilne metodologije razvoja programske opreme so se pojavile kot alternativa težko obvladljivim, natančno specificiranim ter postopkovno opredeljenim tradicionalnim metodologijam razvoja programske opreme. Agilne metodologije slonijo na tem, da so enostavnejše, ne zahtevajo veliko dokumentacije in se hitro prilagajajo spremembam, ki jih zahteva uporabnik. Obenem je tudi uporabnik bolj dejavno vključen v razvoj programske opreme.

Temelji agilnih metodologij so bili postavljeni leta 2001, ko se je sestala skupina 17 svetovalcev in praktikov in objavila štiri načela, ki veljajo za agilne metodologije (Manifesto, 2001) in sicer:

- 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 osnovi pogodb in
- odzivanje na spremembe je pomembnejše od sledenja planu.

Od takrat uporaba agilnih metodologij stalno narašča. Tako lahko iz že omenjene raziskave (Ambler, 2008), v kateri je sodelovalo 624 strokovnjakov s področja informacijske tehnologije (71 % iz Severne Amerike, 17% iz Evrope in 4,5% iz Azije), ugotovimo, da je 69% anketirancev že sodelovalo pri projektih, ki so bili vodeni po agilni metodologiji. Iz raziskave lahko tudi ugotovimo, da je bila stopnja uspešnosti projektov, vodenih po agilnih metodologijah, 77,5%, kar je bistveno več kot pri tradicionalnem discipliniranem pristopu.

V literaturi poznamo veliko agilnih metodologij, kot so na primer:

- XP-Extreme Programming (Beck, 2000),
- FDD-Feature-Driven Development,
- Crystal,
- Scrum itd.

Po podatkih, ki jih navajajo Schwaber, Leganza in D'Silva (2007), sta najbolj razširjeni agilni metodologiji ekstremno programiranje in Scrum.

2.2 Metoda Scrum

Metoda Scrum se je pojavila v prvi polovici devetdesetih let prejšnjega stoletja. Izvor imena izhaja iz ragbija in pomeni vrnitev žoge v igro. Gre za pristop k razvoju programske rešitve, ki predpisuje iterativen in inkrementalen način dela. Projekt je časovno razdeljen na več iteracij, imenovanih *Sprint***.** Vsaka iteracija traja 30 koledarskih dni in mora kot rezultat dati delujočo programsko kodo, ki predstavlja novo (dodatno) funkcionalnost programske rešitve. Programska koda mora biti v celoti preverjena, tako da jo lahko damo v uporabo naročniku. Na ta način naročnik postopno dobiva posamezne dele rešitve, ki jih lahko sproti preizkusi v praksi. Postopek razvoja po metodi Scrum je prikazan na sliki 1.

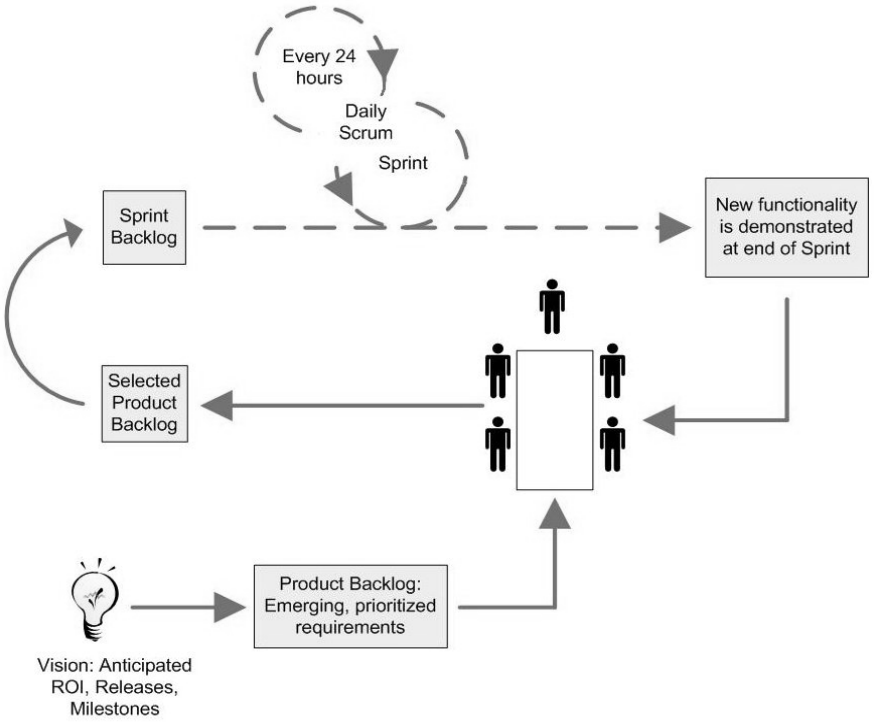
Osnovo za razvoj predstavlja seznam zahtev naročnika, imenovan *Product Backlog*. Seznam zahtev vzdržuje predstavnik naročnika (*Product Owner*), ki zastopa interese uporabnikov in skrbi za financiranje projekta. Seznam zahtev ni dokončen, ampak se ves čas izvajanja projekta dopolnjuje. Predstavnik naročnika po potrebi dodaja nove zahteve in jih razvršča po prioriteti skladno s trenutnimi potrebami uporabnikov.

Na začetku vsake iteracije se predstavnik naročnika sestane z razvojno skupino (v metodi Scrum jo imenujejo *Team*), da bi skupaj določili tisto podmnožico zahtev, ki bo realizirana v naslednji iteraciji. Sestanek (imenujejo ga *Sprint Planning Meeting*) traja 8 ur in je sestavljen iz dveh delov po 4 ure. V prvem delu se predstavnik naročnika in razvojna skupina dogovorita, katere zahteve iz seznama, ki imajo najvišjo prioriteto, bodo vključene v naslednjo iteracijo. V drugem delu pa razvojna skupina izdela seznam nalog (zanj se uporablja izraz *Sprint Backlog*), ki so potrebne za realizacijo dogovorjenih zahtev. Tudi seznam nalog se v vsaki iteraciji stalno dopolnjuje. Naloge, ki so na začetku definirane bolj grobo, je treba razdeliti na več manjših, tako da je za realizacijo vsake naloge potrebnih od 4 do 16 ur dela.

Znotraj vsake iteracije se člani razvojne skupine vsak dan dobijo na kratkem 15-minutnem sestanku, imenovanem *Daily Scrum Meeting*, na katerem mora vsak razvijalec odgovoriti na 3 vprašanja:

- Kaj si delal od prejšnjega sestanka?
- Kaj nameravaš delati do naslednjega sestanka?
- S katerimi težavami se srečuješ pri delu?

Na ta način je zagotovljen sproten vpogled v potek dela na projektu in takojšnje ukrepanje v primeru težav.



Slika 1: Postopek razvoja programske opreme po metodi Scrum (Schwaber, 2004)

Na koncu vsake iteracije razvojna skupina predstavi rezultate svojega dela predstavniku naročnika in vsem zainteresiranim uporabnikom. Predstavitev poteka v obliki posebnega sestanka, imenovanega *Sprint Review Meeting*, ki omogoča uporabnikom, da posredujejo svoje pripombe na opravljeno delo in dajejo predloge za vsebino naslednje iteracije.

Pred naslednjo iteracijo se razvojna skupina sestane tudi s skrbnikom metodologije (v metodi Scrum ga imenujejo *Scrum Master*) z namenom, da bi ocenili potek dela v prejšnji iteraciji in se dogovorili za izboljšave, ki bi omogočile, da bi postalo njihovo delo še bolj učinkovito, izdelana programska oprema pa še bolj kakovostna.

Skrbnik metodologije opravlja vlogo, ki je do neke mere podobna vlogi vodje projekta, vendar ne predpisuje in razporeja posameznih nalog, ampak predvsem skrbi, da delo poteka v skaldu z metodo Scrum na način, ki daje najboljše rezultate. Pomembna naloga skrbnika je, da zagotavlja razvojni skupini optimalne pogoje za delo in skrbi za sprotno odpravljanje morebitnih težav.

Razvojna skupina, ki je odgovorna za realizacijo zahtevane funkcionalnosti, je sestavljena interdisciplinarno in deluje po načelu samoorganizacije. Člani skupine samostojno razporejajo naloge in so kolektivno odgovorni za uspeh ali neuspeh projekta.

Na spletni strani Scrum Community Wiki (2009) navajajo, da Scrum uporabljajo pri svojem delu največja svetovna podjetja kot so IBM, Microsoft, Oracle, Yahoo, Google, Toyota, BMW..., prav tako pa številna manjša in srednje velika podjetja. Scrum se uporablja pri vseh možnih tipih projektov bodisi da gre za finančne projekte, spletne projekte ali projekte za zdravstvo.

3 Študentski projekt kot študija primera

Namen študentskega projekta pri predmetu Tehnologija programske opreme je bil naučiti študente uporabljati metodo Scrum na (skoraj) realnem projektu, ki bi potekal na podlagi dejanskih zahtev konkretnega naročnika. Da bi fakulteta pridobila tak projekt, je vzpostavila sodelovanje s podjetjem SRC, ki je za ta projekt pripravilo seznam zahtev in zagotovilo sodelovanje svojega sodelavca, ki je na projektu nastopal v vlogi predstavnika naročnika (kot *Product Owner*).

Podjetje SRC kot eno izmed vodilnih slovenskih podjetij na področju poslovnih tehnologij, že od samega začetka podpira nove ideje in metode vodenja projektov s ciljem izboljšati delovno okolje znotraj podjetja, dvigniti kvaliteto samega dela in povečati zadovoljstvo strank. Čeprav v podjetju agilne metodologije niso neznanka, saj so nekatere projekte že izpeljali na ta način, so želeli pridobiti še več izkušenj ter znanja s tega področja. Zato je podjetje izkoristilo priložnost sodelovanja s Fakulteto za računalništvo in informatiko s ciljem še bolj podrobno spoznati teoretično in praktično ozadje metode Scrum. Po drugi strani pa so študenti dobili priložnost, da metodo uporabijo na resničnem projektu, ki ga je posredovalo podjetje.

Ker SRC skupaj s svojim podjetjem Infonet Kranj, d.o.o. že vrsto let ponuja rešitve s področja zdravstva, je projekt obsegal razvoj informacijskega sistema splošne bolnice. Pri tem sodelovanju je podjetje SRC igralo vlogo naročnika, ki ga je kot *Product Owner* zastopal njegov sodelavec, sicer študent podiplomskega študija na Fakulteti za računalništvo in informatiko. Da bi razvoj potekal po vseh pravilih metode Scrum, smo natančno definirali tudi ostale vloge na projektu: predavatelj predmeta je igral vlogo skrbnika metodologije (*Scrum Master*), študenti pa so bili razdeljeni na tri štiričlanske skupine, ki so vsaka zase razvijale potrebno programsko opremo.

Na začetku je predstavnik naročnika (*Product Owner*) pripravil seznam zahtev (*Product Backlog*), ki je prikazan na sliki 2. Zahteve so bile grupirane v več sklopov, ki so predvidevali izdelavo in vzdrževanje elektronskega zdravstvenega kartona za vsakega pacienta, naročanje pacientov na preglede in vodenje postopka pregleda, povezovanje z zavarovalnico, od koder naj bi sistem dobival osebne podatke o pacientih in njihovem zavarovanju, ter beleženje podatkov o operativnih posegih. Poleg tega je predstavnik naročnika priskrbel razvojnim skupinam še grobi podatkovni model splošne bolnice in potrebne šifrante, kot je na primer šifrant zdravil.

Id	Kratek opis	Prioriteta	Začetna ocena časa	Faktor kompleksnosti	Prilagojena ocena z upoštevanjem faktorja
1	Kreiranje novega kartona	1	40	1,3	52
2	Prikaz kartona	1	32	1,2	38
3	Iskanje kartona	1	32	1	32
4	Spreminjanje kartonov	1	16	1	16
-	**Naročanje pacientov**				
5	Pogovorno okno za naročanje na pregled	2	40	1,3	52
6	Izbira zdravnika	2	24	1	24
7	Izpis prostih terminov zdravnika	2	24	1,2	29
8	Izpis tipa pregleda	2	24	1	24
9	Prijava na pregled	2	40	1,2	48
Sprint 1			**272**		**315**
10	Povezovanje z zavarovalnico	2	24	1,3	31
11	Vnos anamneze	2	8	1	8
12	Izpis seznama zdravil	2	16	1	16
13	Izpis diagnoz	2	16	1	16
14	Izpis kratkega življenjepisa zdravnika	2	8	1	8
15	Izpis seznama pacientov za pregled	2	16	1	16
16	Vnos pregleda	2	16	1,5	24
17	Kadrovska evidenca	2	32	1,2	39
18	Sprememba statusa za pregled in prikaz trenutnega pacienta pri zdravniku	2	16	1,3	21
19	Operativni poseg	2	40	1,3	52
20	Tiskanje receptov	3	24	1	24
21	Tiskanje bolniškega lista	3	24	1	24
22	Tiskanje napotnic	3	24	1	24
23	Projektna dokumentacija		20	1	20
Sprint 2			**284**		**323**
Release 1			**556**		**638**

Slika 2: Seznam zahtev (Product Backlog)

Celoten projekt je bil razdeljen na dve iteraciji. Vsaka iteracija se je tako, kot narekuje metodologija, začela s sestankom *Sprint Planning Meeting*, na katerem je *Product Owner* predstavil zahteve, in končala s sestankom *Sprint Review Meeting*, na katerem so razvojne skupine predstavile rezultate svojega dela. Na koncu vsake iteracije smo organizirali tudi *Sprint Retrospective Meeting*, na katerem smo analizirali dobre in slabe strani dela v prejšnjem *Sprintu* in se dogovorili, kaj bomo izboljšali v naslednjem.

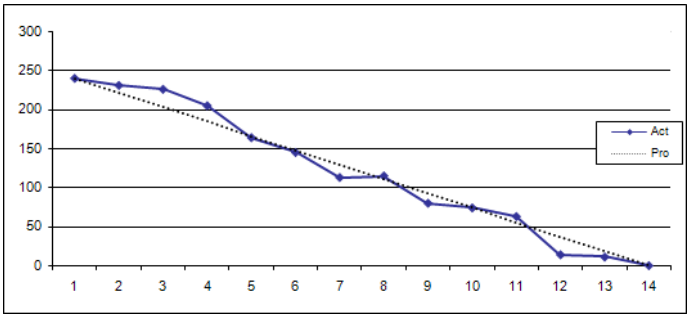
Zaradi študijskih obveznosti, ki jih imajo študenti pri drugih predmetih, je bilo seveda nemogoče pričakovati, da bodo 15-minutni sestanki *Daily Scrum Meeting* potekali vsak dan, tako kot je predpisano z metodo Scrum. Da pa bi vseeno čim bolj verno sledili zahtevam metode Scrum, smo od študentov zahtevali, da imajo tak sestanek dvakrat tedensko: ob ponedeljkih in četrtkih. Ob ponedeljkih smo sestanek izvedli v okviru vaj, prisostvovala pa sta tudi nosilec predmeta (kot *Scrum Master*) in sodelavec SRC (kot *Product Owner*). Ob četrtkih pa so študenti opravili sestanek samostojno. V prvi iteraciji, ki je trajala od 2.3.2009 do 6.4.2009, je bilo 11, v drugi iteraciji, ki je trajala od 9.4.2009 do 1.6.2009 pa 13 takih sestankov.

Vsaka razvojna skupina je morala za vsako iteracijo posebej vzdrževati svoj seznam nalog (*Sprint Backlog*). Za vsako nalogo je morala na začetku določiti izvajalca in oceniti, koliko ur dela je potrebnih za njeno realizacijo. Na vsakem *Daily Scrum Meetingu* pa so morali študenti zabeležiti, koliko ur dela je bilo opravljenih na vsaki nalogi in koliko ur dela je še ostalo, da bi bila naloga v celoti realizirana. Metoda Scrum sicer zahteva samo beleženje števila ur preostalega dela, vendar lahko s pomočjo podatkov o obsegu vloženega dela spremljamo indikatorje učinkovitosti razvojnega procesa v skladu z modelom, opisanim v (Mahnič & Vrana, 2007) in (Mahnič & Žabkar, 2007). Na ta način je projekt, ki so ga izvajali študenti, predstavljal tudi študijo primera uporabe opisanega modela.

Za vzdrževanje seznama nalog smo študentom vnaprej pripravili ustrezen obrazec. Primer izpolnjenega obrazca za eno izmed razvojnih skupin prikazuje slika 3. Študenti so morali izpolnjen obrazec poslati skrbniku metodologije po vsakem *Daily Scrum Meetingu*. S pomočjo podatkov o urah vloženega in preostalega dela je *Scrum Master* (pa tudi razvojna skupina sama) lahko sproti spremljal potek dela na projektu. Vsoto ur preostalega dela lahko za vsak *Daily Scrum Meeting* posebej prikažemo v obliki diagrama, imenovanega *Sprint Burndown Chart*, ki omogoča primerjavo dejanskega poteka projekta z idealno situacijo, ko se obseg preostalega dela linearno zmanjšuje sorazmerno s časom, ki je pretekel od začetka projekta. Primer takega diagrama za seznam nalog s slike 3 je prikazan na sliki 4.

Task ID	Task Description	Responsible	Task Status	Hours Spent													Hours Remaining												
1	2	3	4	1	2	3	4	5	6	7	8	9	10	11	12	13	14	1	2	3	4	5	6	7	8	9	10	11	12	13	14
T-2-01	Popravki podatkovnega modela	Patrik	Completed	0	5	2	0	0	0	0	0	0	0	0	1	0	0	10	5	1	1	1	1	1	1	1	1	1	1	0	0
T-2-02	Povezovanje z zavarovalnico	Patrik	Completed	0	0	0	10	0	0	0	0	0	0	0	0	0	2	10	10	10	0	0	0	0	0	0	0	0	0	0	0
T-2-03	Vnos zavarovanj pacientu	Patrik	Completed	0	0	0	5	0	0	0	0	0	0	0	0	0	2	5	5	5	0	0	0	0	0	0	0	0	0	0	0
T-2-04	Izpis seznama zdravil	Rok	In Progress	0	0	0	0	0	0	0	0	0	0	5	0	2	10	10	10	10	10	10	10	10	10	10	10	3	3	0	
T-2-05	Izpis kratkega življenjepisa zdravnika	Jure	Completed	0	0	0	0	0	0	0	0	0	0	2	0	0	5	5	5	5	5	5	5	5	5	5	5	0	0	0	
T-2-06	Vmesnik za izpis pregledov na določen dan	Matevž	Completed	0	0	0	0	0	13	0	0	0	0	0	0	0	15	15	15	15	15	15	0	0	0	0	0	0	0	0	
T-2-07	Spreminjanje pregleda	Rok	Completed	0	2	0	2	0	4	0	0	0	0	0	0	0	1	10	8	8	6	6	2	2	2	2	2	2	0	0	
T-2-08	Vmesnik za meritve	Rok	Completed	0	0	0	0	0	0	0	4	0	2	4	0	0	1	10	10	10	10	10	10	10	10	6	4	0	0	0	
T-2-09	Vmesnik za diagnoze	Rok	Completed	0	0	0	0	0	0	4	0	0	0	2	0	0	0	10	10	10	10	10	10	1	1	1	1	0	0	0	
T-2-10	Vnos operativnega posega	Patrik	In Progress	0	0	0	0	0	0	0	0	0	0	2	3	0	15	15	15	15	15	15	15	15	15	15	15	5	4	0	
T-2-11	Sprememba operativnega posega	Patrik	In Progress	0	0	0	0	0	0	0	0	0	0	2	3	0	15	15	15	15	15	15	15	15	15	15	15	5	4	0	
T-2-12	Tiskanje receptov	Matevž	Completed	0	0	0	0	0	0	0	2	0	3	0	0	2	10	10	10	10	10	10	10	10	4	3	0	0	0	0	
T-2-13	Tiskanje bolniških listov	Matevž	Completed	0	0	0	0	0	0	0	4	2	4	0	0	2	10	10	10	10	10	10	10	5	2	0	0	0	0	0	
T-2-14	Tiskanje napotnic	Matevž	Completed	0	0	0	0	0	0	0	2	0	2	0	0	0	10	10	10	10	10	10	10	10	4	3	0	0	0	0	
T-2-15	Administrativni vmesnik za dodajanje zdravnikov	Patrik	Completed	0	0	0	0	10	4	0	0	0	0	0	0	0	0	15	15	15	15	5	0	0	0	0	0	0	0	0	
T-2-16	Administrativni vmesnik za dodajanje medicinskih tehnikov	Patrik	Completed	0	0	0	0	15	0	0	0	0	0	0	0	0	0	15	15	15	15	0	0	0	0	0	0	0	0	0	
T-2-17	Administrativni vmesnik za urnike	Patrik	Completed	0	0	0	0	5	4	0	0	0	0	0	0	0	0	15	15	15	15	5	0	0	0	0	0	0	0	0	
T-2-18	Načrtovanje SQL razredov<->podatki	Matevž	Completed	0	1	0	2	0	4	2	0	0	0	0	0	0	0	10	9	9	7	7	2	0	0	0	0	0	0	0	
T-2-19	Avtentikacija in povezovanje z zavarovalniško bazo	Matevž	Completed	0	1	4	0	1	0	5	0	0	0	0	0	0	0	10	9	8	8	6	6	0	0	0	0	0	0	0	
T-2-20	Implementacija kontrolnih razredov	Matevž	Completed	0	0	0	0	2	0	0	0	2	0	0	0	0	0	10	10	10	10	6	6	6	6	0	0	0	0	0	
T-2-21	Dodatki k spletni strani	Jure	Completed	0	0	0	2	0	0	0	5	5	0	0	4	0	2	10	10	10	8	8	8	8	15	5	5	5	0	0	
T-2-22	Sprememba naročanja na pregled	Jure	Completed	0	0	0	0	0	0	0	0	0	0	0	6	0	0	10	10	10	10	10	10	10	10	10	10	10	0	0	
Total				0	9	6	21	33	16	24	13	13	6	11	12	6	12	240	231	226	205	164	145	113	115	80	74	63	14	11	0
																		240	222	203	185	166	148	129	111	92	74	55	37	18	0

Slika 3: Primer obrazca Sprint Backlog



Slika 4: Prikaz količine preostalega dela (v urah) – diagram Sprint Burndown Chart.

4 Analiza vprašalnika

Po vsaki iteraciji so študenti, ki so poslušali predmet Tehnologija programske opreme, izpolnili vprašalnik, s katerim smo želeli preveriti njihovo zadovoljstvo s potekom projekta in mnenje, ki ga imajo o metodi Scrum. V anketi je sodelovalo 30 študentov – poleg študentov, ki so delali na projektu informacijskega sistema splošne bolnice, tudi študenti, ki so razvijali orodje za vodenje projektov po metodi Scrum. Anketa je bila sestavljena iz 14 vprašanj, za vsako vprašanje pa so bili možni odgovori od 1 do 5. Ocena 1 je bila najslabša, ocena 5 pa najboljša. Poleg ocene ja lahko anketiranec pri vsakem vprašanju napisal tudi kratek komentar z utemeljitvijo svoje ocene.

4.1 Seznam zahtev

Prvi dve vprašanji sta se navezovali na seznam zahtev (*Product Backlog*).

Vprašanje 1: **Jasnost zastavljenega Product Backloga (Ali je bil Product Backlog za trenutni Sprint jasno zastavljen? Ali vam je bilo iz kratkega opisa za vsako zahtevo jasno, kaj Product Owner zahteva?)**

Splošna ocena je bila, da je bil opis posameznih zahtev prekratek oziroma premalo podrobno obrazložen. Vendar pa je bila večina nejasnosti pojasnjena na sestankih, kjer je sodeloval predstavnik naročnika (*Product Owner*). Kot je razvidno iz tabele 1, je bila povprečna ocena za to vprašanje v prvi iteraciji slabša kot v drugi. Vzrok je mogoče najti tudi v tem, da smo v drugi iteraciji za vsak projekt pripravili tudi primere uporabe, ki so študentom dali bolj jasno sliko o zahtevah.

Vprašanje 2: **Ocena časa za realizacijo posameznih zahtev iz Product Backloga (Ali so bile ocene potrebnega dela ustrezne?)**

Pri tem vprašanju je bila večina študentov mnenja, da so bile ocene časa za posamezne zahteve, ki jih je na začetku postavil *Product Owner*, razmeroma točne. Tudi tu se je povprečna ocena v drugi iteraciji precej izboljšala.

Vprašanje	**Sprint 1**	**Sprint 2**
Jasnost zastavljenega Product Backloga	3,2	3,9
Ocena časa za posamezne zahteve iz Product Backloga	3	3,8

Tabela 1: Povprečne ocene za vprašanji, povezani s seznamom zahtev

4.2 Vzdrževanje seznama nalog

Vprašanje 3: **Administracija pri metodi Scrum (Ali so bile preglednice, ki ste jih izpolnjevali, jasne in razumljive?)**

Vprašanje 4: **Obremenjenost z administracijo**

Vzdrževanje seznama nalog (*Sprint Backlog*) in beleženje števila opravljenih in preostalih ur je od članov razvojne skupine zahtevalo nekaj dodatnega administrativnega dela. Zato nas je zanimalo, kako študenti ocenjujejo to dodatno obremenitev. Odgovori so pokazali, da so na začetku študenti imeli težave, ker jim je bil postopek izpolnjevanja obrazca *Sprint Backlog* nejasen, kar zlasti velja za primere, ko je bilo treba neko bolj grobo opredeljeno nalogo razdeliti na več manjših in pri tem prvotno oceno za obseg preostalega dela nadomestiti z ocenami posameznih na novo definiranih nalog, ki so pri tem nastale. Vendar so se proti koncu navadili principa vnašanja podatkov, tako da ni bilo posebnih težav. To kaže tudi povprečna ocena v tabeli 2, ki se je v drugi iteraciji dvignila s 3,7 na 4,3. Glede četrtega vprašanja pa lahko iz povprečne ocene ugotovimo, da so bili študenti ves čas enako obremenjeni z administracijo, saj se povprečena ocena 3,3 ni spremenila.

Vprašanje	**Sprint 1**	**Sprint 2**
Administracija pri metodi Scrum	3,7	4,3
Obremenjenost z administracijo	3,3	3,3

Tabela 2: Povprečne ocene za vprašanji, povezani z administracijo pri metodi Scrum

4.3 Tehnične in vsebinske težave

Vprašanje 5: **Tehnične težave na začetku Sprinta**
Vprašanje 6: **Tehnične težave na koncu Sprinta**
Vsaka razvojna skupina se je sama odločila za tehnologijo, v kateri je razvijala svoj projekt. Tako se je nekaj skupin odločilo za tehnologijo, ki so jo posamezniki znotraj skupine že poznali. Nekatere skupine pa so se odločile za novo tehnologijo in so želele skozi razvoj projekta nabrati še dodatne izkušnje in znanja, zato so imele na začetku več težav. Tehnične težave so se pojavljale tudi zaradi združevanja kode, ki so jo napisali različni razvijalci.
Odgovori kažejo, da je bilo v prvi iteraciji več težav na začetku (povprečna ocena 3,3), manj pa na koncu (povprečna ocena 3,9). Nasprotno pa je bilo v drugi iteraciji manj tehničnih težav na začetku iteracije (povprečna ocena 4,1), več pa na koncu (povprečna ocena 3,7). To lahko razložimo z dejstvom, da so imeli študenti na začetku druge iteracije že vzpostavljeno potrebno tehnično infrastrukturo, na koncu pa so se srečevali s problemom integracije kode v delujočo rešitev. Podrobnosti so razvidne iz tabele 3.

Vprašanje 7: **Vsebinske težave (razumevanje zahtevane funkcionalnosti) na začetku Sprinta**
Vprašanje 8: **Vsebinske težave (razumevanje zahtevane funkcionalnosti) na koncu Sprinta** Pri vsebinskih težavah je prišlo do izraza dejstvo, da v razvojnih skupinah ni bilo predstavnika uporabnikov, ki bi lahko sproti odgovarjal na vprašanja razvijalcev. Čeprav metoda Scrum predvideva interdisciplinarno sestavo razvojne skupine (torej tudi predstavnike uporabnikov), tega pri našem projektu ni bilo moč zagotoviti, saj so bili vsi člani razvojne skupine razvijalci. Zato so študenti predlagali, da bi bilo bolje, če bi predstavnik naročnika (*Product Owner*) že med samo iteracijo pregledoval izdelane programe in bi tako sproti (ne pa šele na koncu) posredoval morebitne pripombe. V prvi iteraciji je bila povprečna ocena za sedmo vprašanje 3,5, za osmo vprašanje pa 4,1. Podobno kot pri vprašanjih o tehničnih težavah vidimo, da so se vsebinske težave povečale na koncu druge iteracije, ko je bilo treba posamezne programe povezati v delujočo rešitev. Povprečne ocene so prikazane v tabeli 3.

Vprašanje	**Sprint 1**	**Sprint 2**
Tehnične težave na začetku Sprinta	3,3	4,1
Tehnične težave na koncu Sprinta	3,9	3,7
Vsebinske težave na začetku Sprinta	3,5	3,8
Vsebinske težave na koncu Sprinta	4,1	3,7

Tabela 3: Povprečne ocene za vprašanja o tehničnih in vsebinskih težavah

4.4 Sodelovanje z ostalimi udeleženci v projektu

Vprašanje 9: **Sodelovanje s Scrum Master-jem**
Vprašanje 10: **Sodelovanje s Product Owner-jem**
Vprašanje 11: **Sodelovanje znotraj razvojne skupine**
Pri vprašanjih 9 in 10 so bili študenti dokaj zadovoljni s sodelovanjem s skrbnikom metodologije (*Scrum Master*) in predstavnikom naročnika (*Product Owner*). Pri vprašanju 11 pa so prišle do izraza lastnosti, ki veljajo tudi v razvojnih skupinah znotraj nekega podjetja. Tako so študenti v glavnem komentirali, da se med seboj že od prej dobro poznajo, kar jim je olajšalo koordinacijo dela znotraj skupine. Pri skupinah, ki so bile bolj heterogene, pa je bilo takih težav več. Skupna značilnost vseh ocen je, da so se v drugi iteraciji izboljšale, kar kaže na to, da metoda Scrum pozitivno vpliva na medsebojne odnose in skupinsko delo. Povrečne ocene za vsako vprašanje lahko vidimo v tabeli 4.

Vprašanje	Sprint 1	Sprint 2
Sodelovanje z Scrum Master-jem	4	4,3
Sodelovanje z Product Owner-jem	3,8	4
Sodelovanje znotraj razvojne skupine	4	4,1

Tabela 4: Poprečne ocene za vprašanja o sodelovanju z ostalimi udeleženci v projektu

4.5 Splošna vprašanja

Vprašanje 12: **Primernost obsega dela na projektu (Ali je bil obseg dela na projektu primerno izbran?)**

Vprašanje 13: **Splošna ocena vašega zadovoljstva s potekom dela na projektu**

Vprašanje 14: **Splošna ocena metodologije Scrum (Ali je ta metodologija koristna za delo razvojne skupine? Ali bi jo priporočili drugim razvijalcem?)**

Odgovori na dvanajsto vprašanje kažejo, da je bil obseg dela na projektu primerno izbran, tako da večina študentov ni bila preveč obremenjena, oziroma da zaradi dela na tem projektu niso trpele druge obveznosti, ki so jih imeli na fakulteti. Študenti so bili razmeroma zadovoljni s potekom dela in uporabljeno metodologijo. Iz komentarjev, ki so jih napisali ob svojih ocenah, je moč razbrati, da se jim metoda Scrum zdi primerna za delo večjih skupin na obsežnejših projektih. Bili so tudi mnenja, da metoda močno poveča vpogled v potek razvoja projekta, ne da bi zahtevala veliko administracije, ki se ji razvijalci težje privadijo. Povprečne ocene za to skupino vprašanj so prikazane v tabeli 5.

Vprašanje	Sprint 1	Sprint 2
Primernost obsega dela na projektu	3,8	3,7
Splošna ocena vašega zadovoljstva s potekom dela na projektu	3,7	3,8
Splošna ocena metodologije Scrum	3,8	3,9

Tabela 5: Poprečne ocene za splošna vprašanja

5 Zaključek

Opisana izvedba projekta pri predmetu Tehnologija programske opreme predstavlja nadaljevanje prizadevanj po tesnejšem sodelovanju z računalniškimi podjetji, ki so bila predstavljena že v enem naših prejšnjih prispevkov (Mahnič, 2008). Izkušnje so pokazale, da tovrstno sodelovanje lahko koristi vsem, ki so vključeni v pedagoški proces.

Študenti so ob delu na realnem projektu spoznali prednosti in slabosti metode Scrum, poleg tega pa so se srečali tudi s problemom kvantitativnega spremljanja učinkovitosti razvojnega procesa, ki pri agilnih metodah predstavlja pomemben raziskovalni izziv. Še posebej pomembno pa je, da so skozi delo dobili praktične izkušnje in prenosljive spretnosti, kot so skupinsko delo, medsebojno komuniciranje, načrtovanje in razporejanje nalog, priprava predstavitev izdelanih rešitev ipd. Teh znanj ni moč posredovati v obliki predavanj, ampak jih lahko pridobijo samo v profesionalnem delovnem okolju.

Podjetju SRC je sodelovanje pri predmetu omogočilo, da je na projektu, ki so ga izvajali študenti, brez tveganja in dodatne obremenitve za svoje sodelavce preizkusilo eno izmed potencialno zanimivih agilnih metodologij, ki bi jih lahko vključilo v svoje delo. Sodelavec SRC, ki je sodeloval pri študentskem projektu, je lahko na osnovi konkretnih izkušenj ocenil, katere so prednosti in slabosti metode Scrum in kako bi lahko to metodo najenostavneje vpeljali v že utečene postopke dela v podjetju. Na ta način je bil dosežen pretok znanja iz akademske sfere v prakso, ki ga pri nas še vse preveč pogrešamo. Na osnovi pridobljenih praktičnih izkušenj bodo v podjetju SRC izboljšali svojo interno metodologijo razvoja programske opreme.

Nosilec predmeta je skozi ta projekt na zanimiv način predstavil študentom eno izmed agilnih metodologij. Izkušnje namreč kažejo, da je motivacija študentov za učenje večja, če lahko pridobljeno znanje neposredno preizkusijo v praksi. Obenem pa je ta projekt imel tudi pomembno raziskovalno komponento: kot študija primera je služil za ovrednotenje modela metrik, ki ga razvijamo na fakulteti. Z njegovo pomočjo smo zbrali realne podatke, ki omogočajo računanje indikatorjev učinkovitosti razvoja programske opreme po metodi prislužene vrednosti.

Literatura:

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. (2002). *Agile software development methods,* VTT Electronic, Espoo.

Ambler, S. W. (2008). Has Agile Peaked? Let's look at the numbers, *Dr. Dobbs Journal*, maj 2008, dosegljivo na http://www.ddj.com/architect/207600615 (9.6.2009).

Beck, K. (2000). *Extreme Programming Explained*, Addison-Wesley, 2000.

M. Ceschi et al. (2005). Project Management in Plan-Based and Agile Companies, *IEEE Software*, 22(3): 21-27.

Dubinsky, Y. & Hazzan, O. (2003). eXtreme Programming as a Framework for Student-Project Coaching in Computer Science Capstone Courses, *Proceedings of the IEEE International Conference on Software – Science, Technology & Engineering (SwSTE'03)*.

Mahnič, V. (2008). Teaching Information System Technology in Partnership with IT Companies, Organizacija, 41(2): 71-78.

Mahnič, V. & Vrana, I. (2007). Using stakeholder driven process performance measurement for monitoring the performance of a Scrum based software development process, *Electrotechnical Review*, Ljubljana, 74(5): 241-247.

Mahnič, V. & Žabkar, N. (2007). Introducing CMMI Measurement and Analysis Practices into Scrum-based Software Development Process, *International Journal of Mathematics and Computers in Simulation*, 1(1): 65-72.

Manifesto for Agile Software Development, dosegljivo na http://www.agilemanifesto.org/ (9.6.2009).

Robillard, P. N. & Dulipovici, M. (2008). Teaching Agile versus Disciplined Processes, *International Journal of Engineering Education*, 24(4): 671-680.

Schwaber, C ., Leganza, G. & D'Silva, D. (2007), The Truth About Agile Processes, dosegljivo na http://www.forrester.com/Research/Document/0,7211,41836,00.html (05.06.2009)

Schwaber, K. (2004). *Agile Project Management with Scrum*, Microsoft Press, Redmond.

Scrum Community Wiki (2009). Firms Using Scrum, dosegljivo na http://scrumcommunity.pbworks.com/Firms-Using-Scrum (05.06.2009)

Shukla, A. & Williams, L. (2002). Adapting Extreme Programming For A Core Software Engineering Course, *Proceedings of the 15th Conference on Software Engineering Education and Training (CSEET'02)*.

Xu, S. & Rajlich, V. (2006). Empirical Validation of Test-Driven Pair Programming in Game Development, *Proceedings of the 5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse (ICIS-COMSAR'06)*.