Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matevž Krajnik

Agilna metoda Scrum - Sistematični pregled literature

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2016

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matevž Krajnik

Agilna metoda Scrum - Sistematični pregled literature

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Mentor: prof. dr. Viljan Mahnič

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:
Agilna metoda Scrum - Sistematični pregled literature

Tematika naloge:

Po vzoru preglednih znanstvenih člankov izdelajte sistematični pregled literature, ki obravnava uporabo agilne metode Scrum na področju razvoja programske opreme. Pregled naj skuša odgovoriti na tri raziskovalna vprašanja: katere vrste raziskav metode Scrum obravnavajo članki v znanstveni literaturi, kakšne so koristi in izzivi ob uporabi metode Scrum in kaj predlagajo dosedanje raziskave za lažjo vpeljavo te metode v prakso. Ugotovitve primerjajte z izkušnjami, ki ste jih pridobili sami pri delu na projektu pri predmetu Tehnologija programske opreme.

Za strokovno podporo in napotke se zahvaljujem svojemu mentorju, prof. dr. Viljanu Mahniču, ki me je usmerjal s pravimi nasveti. Za moralno podporo pa gre velika zahvala moji punci Ajši, sestri Nini z družino ter mojim staršem.

Kazalo

Povzetek

Naslov: Agilna metoda Scrum - Sistematični pregled literature

Scrum je agilna metoda razvoja programske opreme, s katero lahko podjetja razvijajo izdelke hitreje ter bolj učinkovito. Zaradi inkrementalnega in iterativnega razvoja ter velike angažiranosti stranke lahko projekt napreduje bolje, lažje pa je tudi implementirati spremembe, ki bi si jih stranka morda želela. V tem diplomskem delu je bil opravljen pregled obstoječe znanstvene literature na temo metode Scrum v razvoju programske opreme, pregled pa je prinesel odgovore na tri zastavljena raziskovalna vprašanja: kakšne vrste raziskav metode Scrum obravnavajo članki v znanstveni literaturi, kakšne so pridobitve in izzivi ob uporabi metode Scrum ter kaj predlagajo dosedanje raziskave metode Scrum za najlažjo uveljavitev metode pri delu. Opravljena je bila tudi primerjava ugotovitev pregleda literature z lastnimi izkušnjami dela na projektu pri predmetu Tehnologija programske opreme.

Ključne besede: Scrum, agilne metode, programska oprema, pregled literature.

Abstract

Title: Agile method Scrum - A Systematic literature review

Scrum is an agile method for software engineering, used by companies to develop products faster and more efficiently. Because the customer is more engaged in the process and the development is incremental and iterative projects progress better, it is also easier to implement any changes in functionality the customer might want. In this thesis a review of existing scientific literature regarding the method Scrum in software engineering has been made. The review brought forth answers to three previously set research questions: which types of research of the method Scrum are present in the scientific literature, what are the advantages and challenges in working with Scrum, and what does previous research suggest for the easiest implementation of the method at work. A comparison of results of this review and our own experiences with working on a project in Software engineering class has been made.

Keywords: Scrum, agile methods, software, literature review.

Poglavje 1

Uvod

Agilna metoda Scrum [2] je v razvoju programske opreme v uporabi že vrsto let. Razlogi za njeno uporabo so učinkovita izvedba projektov, hitrost razvoja, odzivnost na spremembe strankinih zahtev, nezahtevno oziroma malo načrtovanja. Tradicionalni razvoj programske opreme je sicer logičen, saj se vnaprej naredi načrt celotnega razvoja, ki se mu nato sledi. Ampak v današnjem svetu, kjer se poslovno okolje tako hitro spreminja, načrtovanje za daljše časovno obdobje ni smiselno. Prav tako pa stranke lažje specificirajo zahteve, ko že imajo na voljo del delujoče rešitve.

S popularnostjo metode je raslo tudi število izvedenih raziskav ter objavljenih člankov na to temo. Z diplomsko nalogo želim narediti pregled nad obstoječo znanstveno literaturo o metodi Scrum. Ugotoviti želim, kakšne vrste raziskav na to temo obstajajo. Prav tako pa želim iz pregleda literature najti pozitivne in negativne posledice uporabe metode Scrum ter nasvete za najlažjo uvedbo metode v razvoj v podjetju.

1.1 Metodologija

V diplomskem delu sem uporabil metodološki pristop *sistematičnega pregleda literature.*

Sistematični pregled literature je proces zbiranja in kritičnega analiziranja obstoječih raziskovalnih študij ali člankov [3]. Pred iskanjem same literature je treba postaviti raziskovalno vprašanje (oziroma vprašanja), na katera želimo s pregledom literature odgovoriti.

Ko vprašanja poznamo, lahko pričnemo z iskanjem primerne literature. Najprej izberemo iskalne parametre (iskalni niz, npr. "*parameter1 AND parameter2 OR parameter3*") ter primerne knjižnice (npr. digitalne knjižnice *IEEE*, *ScienceDirect*, ...). V izbranih knjižnicah z iskalnim nizom najdemo zadetke (članke) ter zabeležimo njihovo število. Ker vsi zadetki niso primerni, se lotimo filtriranja. To poteka v več korakih:

1. Iz zadetkov odstranimo tiste, za katere že po naslovu vemo, da ne bodo prispevali k odgovorom na naša raziskovalna vprašanja. Zabeležimo število preostalih.

2. Iz preostalih zadetkov še enkrat preberemo naslove ter povzetke in ključne besede. Izločimo neprimerne ter ponovno zabeležimo število preostalih.

3. Preostale članke pregledamo v celoti in še zadnjič izločimo tiste, ki ne vsebujejo odgovorov na naša vprašanja.

4. Zabeležimo končno število preostalih člankov. Ti predstavljajo *primarne študije*, ki jih bomo uporabili za pridobitev odgovorov na naša raziskovalna vprašanja.

1.2 Kaj sledi

V naslednjih poglavjih bom na kratko predstavil agilno metodo Scrum, kako se tradicionalni razvoj programske opreme od nje razlikuje, opisal zbiranje primerne literature ter obdelavo izbranih člankov (primarne študije). Zaključil bom s predstavitvijo rezultatov ter odgovoril na zadana vprašanja.

Poglavje 2

Opis agilne metode Scrum

Agilne metode ter prakse razvoja programske opreme so se pojavile zaradi potrebe po učinkovitejšem sprejemanju hitrih sprememb v zahtevah programske opreme ter pričakovanju strank [6]. Scrum je najpogosteje uporabljena agilna metoda pri razvoju programske opreme. Temelji na principu iterativnega in inkrementalnega razvoja.

Iterativnost: osnovna enota razvoja je sprint, ki traja od enega do štirih tednov. V teoriji je na koncu vsakega sprinta razvit del opreme že na voljo za stranko (lastnika - Product Owner), saj naj bi bili posamezni zahtevani deli opreme med seboj čim bolj neodvisni.

Inkrementalnost: lastnik poda zahtevane funkcionalnosti opreme (Product Backlog) - razdeljene na posamezne uporabniške zgodbe (angl. *user stories*). Ekipe na vsakem Sprint Planning sestanku za vsak sprint vzamejo nekaj zgodb, ki jih bodo realizirale (Sprint Backlog), ter jih razdelijo na naloge (angl. *tasks*). Ekipe izberejo zgodbe za sprint z malo načrtovanja v prihodnost.

Scrum je uspešna metoda ravno zato, ker ni načrtovanja razvoja opreme od začetka do konca, ampak zgolj minimalno načrtovanje za vsak sprint. Je odprt za spremembe in potrebuje veliko angažiranosti s strani stranke (lastnika), da vsi sodelujoči točno razumejo kaj stranka želi. Scrum je hiter, rezultati pa so vidni na koncu vsakega sprinta.

2.1 Delovanje metode Scrum

2.1.1 Pomembne vloge

Metoda Scrum definira tri glavne vloge, ki sodelujejo na projektu:

Product Owner - Stranka / Lastnik projekta

Je glasnik stranke, predstavlja naročniško podjetje. Product Owner napiše uporabniške zgodbe (funkcionalnosti, ki jih mora končni izdelek vsebovati), jim določi poslovno vrednost in posledično prioritete (*Must Have* - kar izdelek mora vsebovati, *Should Have* - kar naj bi izdelek vseboval, *Could Have* - kar izdelek še lahko vsebuje) ter jih doda v Product Backlog. Product Owner nima stika s tehnično izvedbo projekta, čeprav je pogosto v stiku z razvojno ekipo, saj ji nudi dodatno razlago uporabniških zgodb.

Development Team - Razvojna ekipa

Je odgovorna za razvoj in dostavo realiziranih uporabniških zgodb na koncu vsakega kratkega časovnega intervala - sprinta. Posebnost metode Scrum je, da razvojna ekipa ni odvisna od vodje projekta, da ji diktira razvojni proces, temveč je samoodvisna ekipa, ki sama izbira katere uporabniške zgodbe bo lahko realizirala v sprintu, te zgodbe razdeli na naloge (tasks), jih dodeli posameznim razvijalcem v ekipi, ti pa jih nato razvijejo.

Rezultat, dober ali slab, ni nikoli pripisan zgolj enemu posamezniku, temveč celotni razvojni ekipi.

Razvojne ekipe so majhne, od 5 do 9 posameznikov, saj je tako komunikacija znotraj ekipe najbolj učinkovita [4].

Scrum Master

Ni običajni vodja projekta, temveč nek vmesni člen med razvojno ekipo in motečimi dejavniki. Skrbi za ekipo, jih spodbuja, nadzoruje, da se ekipa drži procesov, ki jih narekuje metoda Scrum. Zato vodi ekipne dogodke

(sestanke) ter drži ekipo na pravi poti konstantnega napredka. Sodeluje s Product Owner-jem, da ekipa najbolje razume zahtevane funkcionalnosti.

Na začetku razvoja Scrum Master ne more aktivno sodelovati pri razvoju, saj ima zadosti dela s tem, da ekipa deluje po pravilih in praksah, ki jih določa metoda Scrum. Kasneje, po nekaj sprintih, pa lahko začne sodelovati pri razvoju, saj se potrebni delovni procesi utečejo [4].

Pomembno delo Scrum Master-ja je tudi določanje, kako se lotiti sprememb v zahtevah stranke. Product Owner namreč lahko sporoči kakšno dodatno zahtevo ali pa spremembo obstoječe zahteve. Scrum Master pa se mora odločiti, kako to spremembo vpeljati v razvoj. S Product Owner-jem se lahko dogovori za preložitev realizacije zahteve do naslednjega sprinta, izjemoma pa lahko prekliče trenutni sprint in ga začne znova [4]. Scrum Master torej ščiti razvojno ekipo pred takimi ovirami, ki bi lahko upočasnile ali zapletle razvojni proces.

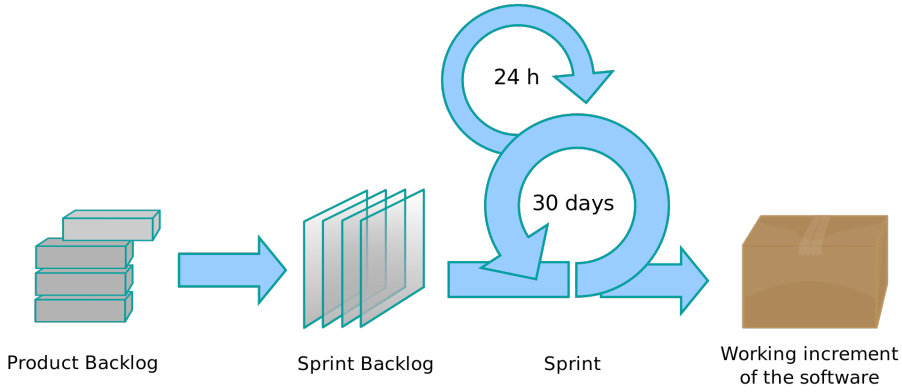
2.1.2 Delovni tok

Glavna razvojna enota je *sprint*. To je vnaprej časovno določen interval, ki ponavadi traja od enega tedna do enega meseca. Najbolj pogosto trajanje je dva tedna.

Pred vsakim sprintom je faza načrtovanja (*sprint planning*), kjer ekipa določi uporabniške zgodbe, ki jih bodo realizirali v tem sprintu (Sprint Backlog), ter jih razdeli na posamezne naloge, Na koncu vsakega sprinta se ekipa zbere na pregledu in retrospektivi (*sprint review meeting* in *sprint retrospective meeting*).

Metoda Scrum določa, da so po končanem sprintu izbrane uporabniške zgodbe realizirane, stestirane, dokumentirane in pripravljene za odpremo.

Razvojna ekipa se vsak dan sreča na *Daily Scrum* sestanku. To je kratek sestanek (traja običajno 15 minut), kjer vsak član razvojne ekipe pove, kaj je delal prejšnji dan, kaj bo delal ta dan ter ali predvideva kakršnekoli ovire, ki bi njega ali razvojno ekipo zaustavljale pri doseganju cilja sprinta. Če ekipa najde oviro, se določi osebo, ki jo bo poskušala odpraviti.



Slika 2.1: Preprost prikaz delovanja metode Scrum. Ko so uporabniške zgodbe definirane, jih razvojna ekipa iz Product Backlog-a izbere nekaj, ki jih bo razvila v naslednjem sprintu, ki traja do enega meseca. Po tem sprintu so izbrane uporabniške zgodbe realizirane in delujoče.

Na sestanku *Sprint Review* ekipa pregleda opravljeno in neopravljeno delo. Opravljeno delo predstavi stranki (Product Owner-ju).

Na sestanku *Sprint Retrospective* pa ekipa prepozna stvari, ki so v sprintu delovale brez problema ter možnosti za izboljšave, ki jih lahko vpeljejo v naslednje sprinte.

Razvojna ekipa sodeluje tudi v procesu izpopolnjevanja zahtev (*Backlog Refinement*), kjer pregledujejo zahteve - uporabniške zgodbe iz Product Backlog-a, ter preverjajo, da so le-te primerno prioritizirane in pripravljene na realizacijo v naslednjih sprintih. Posamezne zgodbe lahko razpadejo na več manjših.

Če je razvojnih ekip več, potem se uporabi scrum več ekip (*Scrum of scrums*). V tej situaciji se ponavadi določi predstavnika vsake razvojne ekipe (ne nujno Scrum Master), ki sodeluje na Daily Scrum sestankih.

Potek dela po metodi Scrum je prikazan na sliki 2.1.

2.2 Kaj pa tradicionalni razvoj?

Tradicionalni razvoj programske opreme pogosto imenujemo kar Slapovni model. Tak razvoj zahteva ogromno načrtovanja in dokumentiranja na začetku razvoja. Določijo se naloge, ki jih je potrebno izvesti, ekipa pa oceni časovno zahtevnost projekta. Ko je celoten načrt odobren, lahko ekipa začne z razvojem. Prva ekipa naredi svoj del, nato razvoj preda naslednji ekipi, itd. Po končanem razvoju se produkt pregleda in preda stranki.

Tak način dela je izredno logičen. Pred začetkom razvoja razmisli, kako boš stvar realiziral, vse zapiši in sledi načrtu. Problem je, da ljudje ne delujemo najbolje na ta način. Tradicionalni razvoj od nas terja, da dobimo vse dobre ideje že na začetku razvoja, ko vse še načrtujemo. Seveda pa ideje dobivamo med procesom, pogosto tudi na koncu. Dobra ideja pozno v razvoju namesto, da bi predstavljala priložnost, da se izdelek izboljša oziroma prilagodi dejanskim potrebam naročnika, tako predstavlja motnjo v razvoju, saj te ideje v prvotnem načrtu ni, posledično pa je potrebnih veliko sprememb, da se to idejo vpelje v končno rešitev [5].

Pri tradicionalnem razvoju programske opreme stranke večinoma dobijo, kar želijo, ampak končni izdelek ni nujno najboljši, kakršen bi lahko bil, saj razvoj poteka strogo po načrtu, ki je bil oblikovan pred začetkom razvoja. To pa pomeni, da sproti ne moremo dosti spreminjati željenega končnega izdelka [5].

Za razliko od tradicionalnega razvoja programske opreme so agilne metode (vključujoč metodo Scrum) fleksibilne, sprotne spremembe so popolnoma izvedljive, hitro prinesejo rezultate (delujoče dele programske opreme).

Poglavje 3

Pridobivanje primerne literature

3.1 Cilji sistematičnega pregleda literature

Agilne metode razvoja programske opreme, še posebej pa metoda Scrum so v zadnjih letih pridobile ogromno na popularnosti v industriji programske opreme. Nova podjetja za razvoj v glavnem uporabljajo agilne metode, vedno več pa je tudi že obstoječih podjetij, ki želijo svoj razvojni proces spremeniti iz tradicionalnega v agilnega.

To pomeni, da obstaja potreba po izkušnjah ter nasvetih ljudi, ki so se z metodo Scrum že srečali in jo uporabljali. Iz tega razloga smo si zastavili sledeča vprašanja, da preverimo, kaj obstoječa literatura na to temo že ponuja:

- RV1. Kakšne so pridobitve in izzivi ob uporabi metode Scrum?
- RV2. Kaj predlagajo dosedanje raziskave metode Scrum za najlažjo uveljavitev metode pri delu?
- RV3. Kakšne vrste raziskav metode Scrum obravnavajo članki v znanstveni literaturi?

3.2 Potek pregleda

3.2.1 Viri podatkov ter iskalna strategija

Za iskanje literature smo uporabili sledeče digitalne knjižnice:

- IEEE Xplore (`http://ieeexplore.ieee.org/Xplore/home.jsp`)
- ACM Digital Library (`http://dl.acm.org/`)
- ScienceDirect - Elsevier (`http://www.sciencedirect.com/`)
- Scopus (`https://www.scopus.com/`)

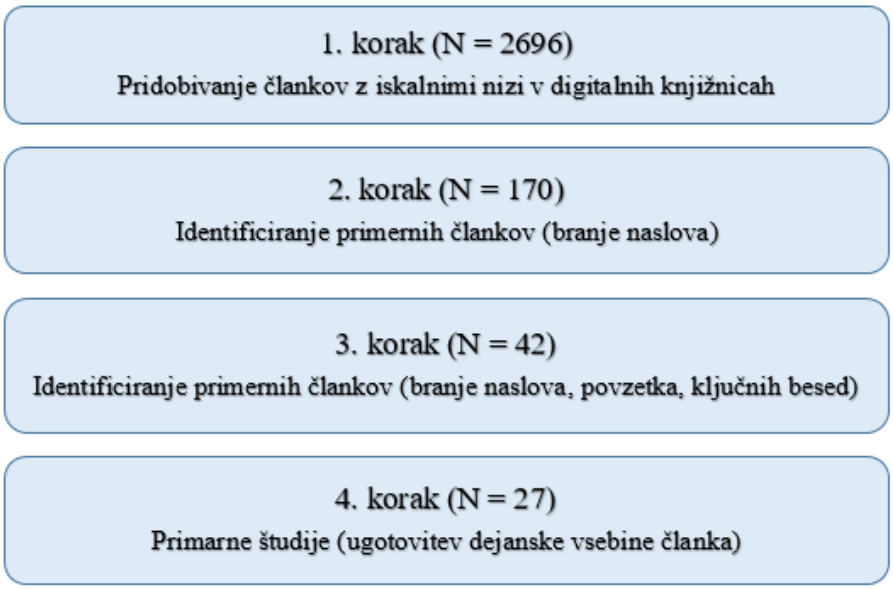
Iz navedenih virov podatkov smo želeli pridobiti članke, ki bodo prispevali k odgovorom na zastavljena raziskovalna vprašanja. Ker nas zanima metoda Scrum v industriji razvoja programske opreme, smo oblikovali obširna iskalna niza:

1. Scrum AND software engineering
2. Scrum AND software development

Niza smo združili z operatorjem OR Boolove algebre, kar pomeni, da je moral članek vsebovati le enega izmed nizov, da je bil vrnjen kot zadetek iskanja.

3.2.2 Proces izbiranja

V izbor so prišli članki, ki se osredotočajo na metodo Scrum v razvoju programske opreme. Prav tako pa smo v izbor vključili članke, ki širše obravnavajo agilne metode, a se vseeno večinoma posvečajo metodi Scrum. Željeni jezik člankov je bil angleščina, člankov v slovenščini iskanje ni vrnilo, v izbor smo vključili en članek v srbohrvaškem jeziku, zaradi nepoznavanja jezika pa smo morali izločiti članke v vseh ostalih jezikih.



Slika 3.1: Koraki izbiranja primarnih študij

Slika 3.1 predstavlja proces sistematičnega pregleda literature skozi vse korake.

Na začetku, v prvem koraku, je iskanje v digitalnih knjižnicah vrnilo 2696 člankov.

V drugem koraku smo prebrali naslove vseh člankov ter izločili tiste, pri katerih je že sam naslov povedal, da k našemu pregledu in raziskovalnim vprašanjem ne bi prispevali. Na primer, med zadetki so se pojavili članki, ki so govorili o metodi Scrum v drugih panogah ali pa so se osredotočali na drugo agilno metodo, Scrum pa je bil v članku zgolj omenjen. Prav tako smo izločili članke, pri katerih smo že po naslovu videli, da ne bodo pisali o prednostih oziroma slabostih metode Scrum oziroma o implementaciji metode pri delu. Po drugem koraku nam je ostalo 170 člankov.

V tretjem koraku smo prebrali povzetke ter ključne besede preostalih člankov. S pomočjo tega koraka smo lahko še bolje videli, ali bo članek prispeval k odgovorom na zastavljena raziskovalna vprašanja, in izločili tiste,

ki ne bi prispevali. Po tretjem koraku nam jih je ostalo še 42.

V zadnjem koraku smo še enkrat prebrali povzetke člankov, prav tako pa smo na hitro pregledali vsebino člankov, saj se lahko zgodi, da povzetek ne opisuje vsebine članka dovolj natančno. Po tem zadnjem koraku nam je ostalo 27 člankov. Ti predstavljajo *primarne študije*, ki so predmet nadaljne podrobne analize.

Primarne študije so navedene v prilogi Priloga A. Primarne študije. Nanje se v besedilu sklicujemo s črko Š in zaporedno številko študije, npr. "[Š1]".

Poglavje 4

Rezultati

Že samo pridobivanje primerne literature je prineslo potrditev o vsej večji popularnosti metode Scrum v svetu razvoja programske opreme. Število objavljenih člankov ter druge literature na to temo je vsako leto več. Velika večina teh člankov piše o prednostih metode Scrum ter drugih agilnih metod v primerjavi s tradicionalnim razvojem programske opreme.

V tabeli 4.1 vidimo razdelitev primarnih študij glede na leto objave.

Leta	2000 - 2007	2008 - 2010	2011 - 2013	2014 - 2016	Skupaj
Št. člankov	4	8	11	4	27
Odstotek	15 %	30 %	40 %	15 %	100 %

Tabela 4.1: Primarne študije glede na izdano leto

4.1 *RV1: Kakšne so pridobitve in izzivi ob uporabi metode Scrum?*

Med branjem primarnih študij so nas najprej zanimale prednosti metode Scrum, seveda pa tudi slabosti. Po pričakovanjih je analiza vrnila predvsem pozitivne rezultate. Z metodo Scrum so zadovoljni tako razvijalci kot stranke, kljub temu, da za slednje Scrum prinaša več dela kot tradicionalen razvoj

programske opreme.

4.1.1 Komunikacija in sodelovanje

Zagotovo je ena največjih prednosti Scruma izboljšana komunikacija ter sodelovanje znotraj razvijalske ekipe ter med razvijalci in strankami [Š7, Š9]:

Ker je ena glavnih vrednot Scruma "delujoča programska oprema pred obsežno dokumentacijo", obstaja skozi sestanke Daily Scrum, Sprint Review, Sprint Retrospective, ..., ter povečano komunikacijo večji prenos znanja znotraj ekipe (npr. na Daily Scrum sestankih lahko razvijalec pove, da ima težavo, ostali razvijalci pa mu kasneje priskočijo na pomoč. S tem pride do odlične izmenjave "know-howa"[Š1, Š9, Š11, Š19]), prav tako pa zmanjšan napor pisanja obsežne dokumentacije [Š26]. Slaba stran tega pa je, da se lahko znanje enostavno izgubi, če člani sestankov ne jemljejo resno oziroma ekipo zapustijo. V tem primeru je tradicionalni razvoj programske opreme boljši, saj je vse znanje, pridobljeno skozi razvoj, zapisano v dokumentaciji, kjer se ne more izgubiti [Š1, Š21].

Stranke postanejo dejansko vključene v to, kar se za njih razvija [Š5, Š7, Š20], razvijalcem pa ta konstantna komunikacija s stranko odgovarja, saj se jim vedno pojavljajo vprašanja glede implementacije, stranka pa ima zanje vedno najnovejše odgovore ter informacije [Š5, Š13]. Na žalost pa stranke vedno ne znajo najbolje razložiti, kaj točno želijo od rešitve, kar za razvijalce predstavlja veliko težavo [Š5].

Čeprav ima tesno sodelovanje med razvijalci ter stranko veliko prednosti, pa to pomeni večje breme za obe strani, saj si morata obe vzeti čas od razvoja oziroma poslovnih procesov, da se sestaneta [Š1, Š14, Š20, Š21].

Poseben izziv predstavlja globalno distribuiran razvoj, saj postane komunikacija največji problem, vpliv na ekipno sodelovanje pa imajo lahko tudi kulturne razlike [Š15].

4.1.2 Odzivnost, učinkovitost, fleksibilnost

Ko se metoda Scrum v podjetju uspešno uteče, je pričakovati izboljšanje v produktivnosti [Š2, Š3, Š7, Š18], odzivnosti na spremembe [Š3, Š7, Š8, Š20] ter nižji ceni razvoja [Š17, Š18].

Razvijalci pravijo, da lahko fleksibilno razvijajo tisto, kar je zares potrebno, saj se osredotočijo na to, kar je za stranko pomembno in izpustijo stvari, ki so manj relavantne [Š1, Š18, Š24].

Še ena prednost je krajši čas, ki ga podjetje potrebuje, da programsko rešitev spravi do trga [Š1, Š23].

Razvijalci bolj zgodaj zaznajo težave [Š8], prav tako pa se strinjajo, da je boljša hitra, a napačna odločitev, ki traja eno dolžino sprinta (največ en mesec, odvisno od implementacije metode Scrum), kot pa pravilna odločitev, za katero se porabi veliko časa [Š7].

Pogoste izdaje programske opreme ter demonstracije implementiranih rešitev zagotovijo, da se razvijalski čas učinkovito porablja na zahteva, ki so za stranko najbolj pomembne [Š8].

Kljub vsem prednostim pa ne gre pričakovati, da bo metoda Scrum že takoj po vpeljavi v razvoj začela krajšati čas razvoja ter nižati stroške. Na začetku je namreč dosti časa potrebnega za pravilno usposabljanje zaposlenih, posledično pa je možno tudi povečanje stroškov [Š18, Š21, Š23].

4.1.3 Kakovost

Velika prednost metode Scrum je gotovo v izboljšani kakovosti produktov [Š3, Š17, Š18], nekateri razvijalci pa pravijo, da jim je Scrum pomagal, da so pri svojih rešitvah postali bolj inovativni [Š3].

Eden razlogov za izboljšanje kakovosti je ta, da se večje zahteve strank razbijejo na manjše uporabniške zgodbe, kar pomeni, da lahko ekipe osnovne funkcionalnosti zahteve izdajo v enem sprintu, v kasnejših pa implementacijo izboljšujejo glede na pridobljene dejanske uporabniške podatke [Š4].

4.1.4 Transparentnost

Pri metodi Scrum je celoten proces razvoja transparenten, saj se uporabljajo različna orodja (npr. tabla z nalogami, angl. *shared task-board*) ter Daily Scrum sestanki, ki prinesejo vidno stanje razvoja. To pa omogoči razvijalcem, da se nemudoma odzovejo na težave [Š1, Š9, Š13, Š24, Š27].

Nekateri ljudje niso najboljši pri načrtovanju svoje delovne obremenitve. Tudi pri tem pomaga metoda Scrum, saj s transparentnostjo ter Sprint Planning sestanki drži ljudi na pravi poti [Š7, Š9], prav tako pa se bolj zavedajo, kaj je od njih pričakovano [Š9].

Ena izmed posledic pa je, da razvijalci menijo, da je za implementacijo Scruma potrebna večja disciplina kot pri tradicionalnem razvoju, kar vidijo kot slabost, ki negativno vpliva na sprejetje metode Scrum [Š1].

Ker Scrum zahteva veliko transparentnosti ter sodelovanja, lahko določenim posameznikom povzroči stres, zaradi osebnosti, pomanjkanja sposobnosti ali pa, ker jim ni všeč delovanje v ekipi [Š2, Š17, Š19].

4.1.5 Delovni tok

V [Š5] so opravili analizo delovnih ur pred ter po implementaciji metode Scrum v njihov razvoj. Ugotovili so, da so z metodo Scrum znižali število nadur ter skorajda izničili nihanja njihove razporeditve. Z drugimi besedami, vpeljali so stalen in enakomeren tempo razvoja, kar je v nasprotju s tradicionalnim razvojem programske opreme, kjer je moč opaziti veliko povečanje obremenitve ter nadur pred posameznimi mejniki (angl. *milestone*) [Š1, Š2, Š22].

Programska oprema se izdaja v kratkih intervalih - sprintih - zato ni pritiska na razvijalce, da jim vse uspe že v prvem poskusu, s tem pa se pojavi tudi kultura "testiranja in učenja"[Š4].

Scrum že med samim razvojem zahteva veliko angažiranja s strani strank, saj morajo stranke rešitev pregledati, preden se jo izda na trg, s tem pa se izniči možnost, da bi bile stranke nezadovoljne z izdano rešitvijo [Š4].

Prednost Scruma je tudi v tem, da celoten končni izdelek razpade na zaporedje obvladljivih delov, ki pa se jih je mnogo lažje lotiti [Š9].

Čeprav so posamezne naloge manjše ter enostavnejše, se lahko kompleksnost projekta poveča zaradi drugih faktorjev. Razvijalci poročajo predvsem o problemu na začetku projekta, saj Scrum ne določa začetnega dolgoročnega načrtovanja, nekdo pa mora vseeno razmišljati o prihodnosti ter načrtovati portfelj izdelkov in strategijo razvoja na višji ravni. Prav tako pa se je vedno težje osredotočiti na razvoj novih delov programske opreme, medtem ko je sočasno potrebno izboljšati že izdano programsko opremo (npr. zaradi obstoječih napak v implementaciji ali sprememb strankinih zahtev) [Š1, Š21].

Še dve zaznani slabosti Scruma pa sta, da pri razvoju ni predvidljivosti, zgodi pa se tudi, da lahko nekatere zgodbe opravi le določen razvijalec (zaradi edinstvenega znanja), kar pa prepreči samo-organizacijo ter povzroči zastoje v razvoju (angl. *bottleneck*) [Š19].

V [Š10] so ugotovili, da manjša podjetja lažje sledijo principom metode Scrum, saj imajo večja podjetja večinoma še stare, podedovane (angl. *legacy*) prakse, ki preprečujejo nadaljno posvojitev novih.

4.1.6 Scrum sestanki

Metoda Scrum vpelje uporabne sestanke, ki jih je treba opravljati vsakodnevno (Daily Scrum) ali pa pred oziroma po sprintu (Sprint Planning, Sprint Review, Sprint Retrospective).

Ti so koristni tako za razvijalce (kot je omenjeno zgoraj) kot za stranke.

Ker je v Sprint Planning sestanek vključena celotna ekipa, vsak ve, kaj se od njega zahteva [Š5].

Stranke pa imajo rade Sprint Review sestanke, saj si včasih težko predstavljajo končni rezultat, ti sestanki pa omogočijo, da se pomisleki glede izdelka odkrijejo sproti [Š5].

Slabost sestankov, ki jih nalaga metoda Scrum je ta, da če ljudje na sestanke ne pridejo pripravljeni, se lahko ti časovno zavlečejo [Š5].

Prav tako pa večja kot je ekipa, daljši so sestanki, težje je zagotoviti korist za vse udeležence [Š17, Š19].

4.1.7 Zadovoljstvo

Podjetja pravijo, da Scrum vpliva na boljše počutje razvijalcev, kar vpliva na zmanjšanje stresa [Š2, Š19] ter energičnost in entuziazem ekipe [Š7, Š13, Š17, Š24].

Ker Scrum zahteva veliko sodelovanja prav tako vpliva na grajenje zaupanja znotraj ekipe [Š13].

Razvijalci pravijo, da so bolj zadovoljni [Š20], saj jim metoda omogoča, da se nalog lotijo po svoje, zato pa so lahko ponosni na svoje rešitve [Š1, Š11, Š13], prav tako pa že zgodaj v razvoju v programski rešitvi dosežejo dobro pokritost zahtev [Š1].

Stranke so bolj zadovoljne s programsko opremo, imajo večje spoštovanje do razvijalcev, razumejo, da se lahko njihova pričakovanja ter dejanski rezultati razlikujejo, če ni prisotnih jasnih navodil ter konstantne komunikacije [Š5].

4.2 *RV2: Kaj predlagajo dosedanje raziskave metode Scrum za najlažjo uveljavitev metode pri delu?*

Vsaka sprememba delovnega procesa zahteva trud ter čas. Ampak kot smo ravnokar videli, ima metoda Scrum veliko prednosti pred tradicionalnim razvojem programske opreme. Številna podjetja se ravno zaradi teh prednosti odločajo za vpeljavo te metode v svoj razvoj. Katere predloge, ki jim bodo pomagali pri vpeljavi Scruma, pa lahko podjetja pridobijo iz obstoječe literature?

4.2.1 Komunikacija in sodelovanje

Bližnje sodelovanje s strankami ter ocenjevanje potencialno odpremljive programske opreme pripomoreta k temu, da so potrebe strank zadovoljene [Š1, Š18, Š27].

Izboljšana komunikacija je ena glavnih prednosti metode Scrum, vendar pa je stil komunikacije pomembnejši od frekvence, neformalna komunikacija pa lahko izboljša uspeh projekta [Š17].

Globalno distribuiran razvoj zahteva, da ima ekipa sinhroniziran delovni čas, v katerem lahko izvrši sestanke, ki jih narekuje metoda Scrum. Pogosto se namreč zgodi, da člani ekipe delajo v različnih časovnih pasovih. Če ekipa nima te možnosti, pa lahko uporabi druge prakse, kot so strogo komunikacijsko politiko, zmanjšano število sestankov, ... Da se izboljša sodelovanje je možna uporaba izmenjave obiskov, neformalnih sestankov distribuiranih članov ekipe, postopno distribuiranje ekipe, itd. Ekipa lahko uporabi številne načine komunikacije, npr. elektronsko pošto, konferenčne klice, takojšnje sporočanje (angl. *Instant Messaging - IM*), ipd. [Š15].

4.2.2 Ljudje

Za uspešno implementacijo metode Scrum pri delu je potrebno imeti ljudi, ki so o metodi poučeni, prav tako pa morajo biti člani ekipe brez predsodkov ter dovzetni za spremembe [Š11]. To ne velja le za razvijalce, temveč tudi za stranke, saj morajo razumeti, kaj se od njih v času razvoja pričakuje [Š5, Š17].

Razvojna ekipa mora imeti občutek, da jim stranka (Product Owner) sledi, da v razvoju niso sami. Product Owner ima pooblastila ter odgovornost za končni izid [Š6, Š25]. Prav tako pa je Product Owner odgovoren za oblikovanje vizije ter prioritet Product Backloga, razvojna ekipa pa se mora osredotočiti na učinkovito dostavo teh prioritet [Š6, Š25].

Predlagana velikost ekipe je od 5 do 9 članov, vključujoč Scrum Masterja ter Product Ownerja [Š16].

Vsi ljudje preprosto niso naklonjeni metodi Scrum, nekateri zelo sposobni ljudje so izredno individualistični, iz tega razloga jih Scrum ovira, posledično pa oni ovirajo produktivnost ekipe. Take ljudi je najbolje odstraniti iz Scrum ekipe [Š19].

Razvoj pri metodi Scrum je ekipno delo, zato ni dobro, če posamezni člani ekipe opravljajo prekomerno količino dela ali nadure, da bi postali junaki v ekipi, saj to neposredno vpliva na ostale člane ter načrt dela [Š25].

4.2.3 Delovni tok

Pravilno definiranje uporabniških zgodb je izrednega pomena [Š4], da se izognemo razlikam v interpretaciji. Uporabniške zgodbe morajo biti nedvoumne ter nuditi zadostno količino informacij.

Stalen in enakomeren tempo razvoja je prednost Scruma, ki se jo je treba držati. Ekipe morajo za naslednji sprint načrtovati toliko dela, kot so ga opravile v prejšnjem. S tem se ohranja razlika do tradicionalnega razvoja programske opreme, kjer se postavijo roki in ekipe ponavadi zmanjšajo kakovost in opravljajo nadure, da izdajo rešitev [Š2].

Splača se hitro doseči dobro pokritost zgodb iz Product Backloga [Š24], da v kasnejših sprintih ostane čas za izboljševanje in odzivanje na spremembe zahtev.

Ker za testiranje pri Scrumu pogosto zmanjka časa, je pametno za vsako uporabniško zgodbo, ki se jo razvija v sprintu, dodeliti nalogo (angl. *task*), ki je namenjena testiranju [Š7, Š10]. Tako se zagotovi, da se testiranju dodeli dovolj časa.

4.2.4 Scrum sestanki

Predlog za krajše in jedrnate sestanke je, da pred sestanki ekipe razjasnijo čimveč vprašanj prek elektronske pošte ali drugega komunikacijskega kanala [Š3].

Daily Scrum ni čas za iskanje rešitev izraženih problemov. Člani, ki jih problem zadeva oziroma ga lahko pomagajo rešiti se morajo dobiti posebej [Š25].

4.2.5 Prilagoditev Scruma

Scrum preprosto ni za vsako podjetje, saj mora imeti organizacija, ki želi vpeljati Scrum v svoj razvoj, za to pravo kulturo - podporo za pogajanja, zmožnost spreminjanja, sodelovanje ter nenehno izmenjavo izkušenj ter znanja [Š17, Š18].

Priporočljivo je postopno vpeljevanje metode Scrum. Primer: pri *BabyCenter.com* so Scrum vpeljali z metodo "plazi se, hodi, teci". Začeli so z majhnim pilotnim projektom in dnevnimi Daily stand-up sestanki. Nato so nadaljevali z večjimi projekti, vpeljali izdelovanje Product Backloga ter Sprint Review sestanke. Kasneje pa so dodali še ostale procese metode Scrum [Š4]. Postopno vpeljevanje Scruma predlaga tudi [Š26], kjer so prav tako prepoznali pomembnost pilotnega projekta.

Ker so si podjetja različna, enaka dolžina sprinta ne ustreza vsem. Podjetje mora samo ugotoviti, ali jim bolj ustrezajo krajši ali daljši sprinti. Vseeno pa članki poročajo, da so sprinti, dolgi 4 tedne, predolgi, saj se ekipe ne morejo odzvati na zahteve strank, zanka povratne informacije pa postane predolga. Sprinti, krajši od dveh tednov, pa povzročijo, da upravljanje s projektom (sestanki Sprint Planning ter Sprint Review) vzame preveč časa [Š10, Š12].

Nekaj študij predlaga, da podjetja metodo Scrum z manjšimi spremembami prilagodijo tako, da jim bolje ustreza:

- Sprint Preview, razvijalci nekaj dni pred Sprint Planning sestankom pregledajo sprint ter možnosti implementacije [Š4].
- Sprint Break, po nekaj sprintih razvijalcem predstavlja odmor od dela na sprintu, Product managerjem pa nekaj časa za strateški razmislek

ter ocenjevanje dosedanjega dela [Š4].

- Prioritiziranje nalog omogoča razvijalcem, da naloge z nižjo prioriteto znotraj sprinta izpustijo, če jim zmanjka časa [Š11].
- Le začetne naloge se dodelijo članom ekipe, saj se drugače naloge pogosto prestavljajo znotraj sprinta [Š11].
- Če uporabniška zgodba po sprintu ni dokončana v celoti, se ustvari novo uporabniško zgodbo, ki zajema manjkajoče funkcionalnosti [Š11].

4.3 *RV3: Kakšne vrste raziskav metode Scrum obravnavajo članki v znanstveni literaturi?*

Poleg zgornjih vprašanj nas je zanimalo tudi, kakšne vrste člankov lahko identificiramo med primarnimi študijami, kje so bili objavljeni, na kakšen način so avtorji pridobivali podatke za pisanje člankov, kater tip raziskave je bil uporabljen, ...

4.3.1 Kje so bile primarne študije objavljene

Med identificiranimi primarnimi študijami je večina člankov (81 %), kot lahko vidimo v tabeli 4.2, izdanih v zbornikih znanstvenih konferenc (angl. *conference proceedings*), npr. *Hawaii International Conference on System Science (HICSS), Agile Conference (AGILE), itd.*

Ostali članki (19 %) so objavljeni v znanstvenih revijah, npr. *IEEE Software, Indian Journal of Science and Technology, itd.*

Vir študije	Članki na konferencah	Članki v revijah	Skupaj
Študije	Š1-Š8, Š10-Š19, Š22-Š24, Š27	Š9, Š20, Š21, Š25, Š26	Š1-Š27
Število	22	5	27
Odstotek	81 %	19 %	100 %

Tabela 4.2: Primarne študije glede na to, kje so bile objavljene

V tabelah 4.4 in 4.5 smo uporabili okrajšave oziroma kratice. Te so razložene v tabeli 4.3.

Okrajšava/kratica	Razlaga
ER	empirična raziskava
ŠP	študija primera
PL	pregled literature
PI	poročilo izkušenj
RŠ	raziskovalna študija
AR	akcijska raziskava
Spr. proj.	spremljanje projekta
Lst. izk.	lastne izkušnje
Obst. lit.	obstoječa literatura
GOV org.	vladna organizacija

Tabela 4.3: Razlaga okrajšav oziroma kratic za tabeli 4.4 in 4.5

4.3.2 Kateri tipi raziskav so bili uporabljeni v primarnih študijah

Primarne študije so bile opravljene z enim (ali več) od 6-ih identificiranih tipov raziskav. Razporeditev primarnih študij glede na tip raziskave lahko vidimo v tabeli 4.4.

Večino (82 %) predstavljajo študije primera (30 %), empirične študije (26 %) ter pregledi literature (26 %).

4 primarne študije (15 %) so vsebovale poročila izkušenj, 2 (7 %) sta

bili odprti raziskovalni študiji (angl. *exploratory study*), 1 (4 %) pa je bila akcijska raziskava (angl. *action research*).

Študija Š8 je predstavitev okrogle mize. V njej so predstavljeni kratki povzetki govorov gostov okrogle mize, iz tega razloga ne moremo identificirati tipa raziskave oziroma načina pridobivanja podatkov pri tej študiji.

Raziskava	ER	ŠP	PL	PI	RŠ	AR	Skupaj
Študije	Š1, Š4, Š7, Š10, Š18, Š19, Š27	Š3, Š5, Š11-Š13, Š22, Š23, Š26	Š6, Š15, Š17, Š20, Š21, Š25, Š27	Š6, Š9, Š14, Š16	Š2, Š21	Š24	Š1-Š27 brez Š8
Število	7	8	7	4	2	1	26
Odstotek	26 %	30 %	26 %	15 %	7 %	4 %	96 %

Tabela 4.4: Primarne študije glede na tip raziskave/članka

4.3.3 Kateri način pridobivanja podatkov je bil uporabljen v primarnih študijah

Določili smo 5 načinov pridobivanja podatkov, ki so bili uporabljeni v primarnih študijah. Razporeditev primarnih študij glede na to kategorijo lahko vidimo v tabeli 4.5.

V 11-ih (41 %) primarnih študijah so podatke pridobivali tako, da so projekt oziroma projekte pozorno spremljali med njihovim izvajanjem. Večinoma to ni bil edin način zbiranja podatkov, temveč so uporabljali še druge načine, npr. ankete [Š5], lastne izkušnje [Š6], intervju oziroma pogovor z udeleženci projekta [Š13].

Enak delež primarnih študij je uporabljalo ankete (26 %) ter intervjuje oziroma pogovore z udeleženci projektov (26 %).

Seveda so pri večini identificiranih primarnih študij avtorji uporabili obstoječo literaturo, vendar pa je bilo 7 (26 %) študij takih, pri katerih je

Podatki	Ankete	Spr. proj.	Intervju	Lst. izk.	Obst. lit.	Skupaj
Študije	Š2, Š5, Š10, Š18, Š21, Š26, Š27	Š3-Š7, Š9, Š11-Š13, Š19, Š22	Š1, Š3, Š12, Š13, Š23, Š24, Š26	Š6, Š14, Š16	Š6, Š15, Š17, Š20, Š21, Š25, Š27	Š1-Š27 brez Š8
Število	7	11	7	3	7	26
Odstotek	26 %	41 %	26 %	11 %	26 %	96 %

Tabela 4.5: Primarne študije glede na način pridobivanja podatkov

(sistematični) pregled literature predstavljal večji del študije. Literaturo so iskali v elektronsko dostopnih knjižnicah (npr. *IEEE Xplore, ACM Digital Library, ScienceDirect - Elsevier* ter druge).

Pri treh (11 %) primarnih študijah so bile kot vir podatkov identificirane lastne izkušnje udeležencev v projektih.

4.3.4 V kakšnem okolju so bile primarne študije izvedene

Velika večina (85 %) primarnih študij, kot lahko vidimo v tabeli 4.6, je potekala v industrijskem okolju. To nas ne preseneča, saj se v tem okolju tudi največ dogaja v smislu razvoja programske opreme ter agilnih metod. Za industrijsko okolje je na voljo veliko več informacii ter možnosti raziskav.

Okolje	Industrija	Akademsko	GOV org.	Skupaj
Študije	Š1, Š2, Š4-Š10, Š12-Š21, Š23-Š25, Š27	Š3, Š8, Š11, Š22	Š26	Š1-Š27
Število	23	4	1	27
Odstotek	85 %	15 %	4 %	100 %

Tabela 4.6: Primarne študije glede na obravnavano okolje

Tematiko akademskega oziroma univerzitetnega okolja so obravnavale 4 (15 %) primarne študije. Tovrstnih raziskav je vse več, saj se univerzitete

zavedajo, da gre razvoj programske opreme v smeri agilnih metod ter da industrija potrebuje razvijalce, ki so o teh metodah podučeni.

Obravnavano okolje ene (4 %) primarne študije pa je bila vladna organizacija. Ta študija je skušala dokazati, da lahko tudi vladne oziroma druge bolj toge organizacije z dovolj truda vpeljejo agilne metode v svoj razvoj.

Poglavje 5

Primerjava ugotovitev z lastnimi izkušnjami

V tretjem letniku študija računalništva in informatike smo se pri predmetu Tehnologija programske opreme prvič srečali z agilno metodo Scrum.

Pri predavanjih smo spoznali kako Scrum sploh poteka in kakšne so naloge razvijalcev, Scrum Masterja ter Product Ownerja. Spoznali smo podjetje Parsek (`http://parsek.com/`), ki je predstavljalo naročnika projekta. Na vajah pa smo spoznali orodje ACscrum, s katerim smo skozi celoten razvoj beležili zahtevane podatke. S tem orodjem smo si pomagali tudi pri Sprint Planning sestankih, saj nam je olajšalo ocenjevanje zahtevnosti uporabniških zgodb, izbiranje zgodb za sprinte ter razdelitev zgodb na manjše naloge. Delo je potekalo v skupinah, s po 4-imi člani. En izmed članov je predstavljal tako Scrum Masterja kot razvijalca.

Z metodo Scrum smo se vsi v naši ekipi srečali prvič, zato nas je zanimalo, kako bo vplivala na razvoj našega izdelka.

Najprej smo ugotovili, da je metoda od nas zahtevala več discipline in malo več dela.

Skozi semester smo opravili tri sprinte, pred vsakim pa smo opravili Sprint Planning sestanek. Prvi Sprint Planning sestanek je trajal dve uri, saj smo

morali oceniti časovno zahtevnost vseh uporabniških zgodb, izbrati zgodbe za prvi sprint ter jih razdeliti na naloge. Vsak član ekipe si je izbral naloge, za katere je mislil, da jih bo znal opraviti. Razdelili smo si vse naloge, a smo kmalu ugotovili, da si moramo nekaj nalog med seboj zamenjati (npr. en član je imel več predznanja o razvoju naloge). Do enake ugotovitve so prišli v [Š11]. Zato smo si pri naslednjih dveh sprintih na Sprint Planning sestanku razdelili le nekaj osnovnih nalog.

Čeprav nismo opravljali Daily Scrum sestankov vsak dan, temveč dvakrat na teden, smo morali biti vseeno bolj dosledni pri beleženju našega napredka.

Med člani ekipe je stalno potekala neformalna komunikacija, saj smo se srečevali na fakulteti in pogovarjali prek spletne klepetalnice. Ko je član povedal, da mu neka naloga ne gre, je to takoj sporočil ostalim, kmalu zatem pa je prejel predloge za rešitev. Člani smo si med seboj predlagali tudi, katerih nalog se je pametneje najprej lotiti, katere pa si pustiti za konec. Ugotovili smo, da je Scrum pripomogel k boljšemu sodelovanju in komunikaciji med člani.

Z orodjem ACscrum smo dosegli veliko transparentnost statusa razvoja, saj je vsak član lahko videl koliko časa so člani ekipe delali na posameznih nalogah, katere naloge so dokončane, katerih se je še treba lotiti. Prav tako smo lahko spremljali naš Burndown Chart, graf, ki je prikazoval naš napredek. Na voljo smo imeli tudi diagram Team Involvement, ki je prikazoval, za kolikšen delež izdelka je odgovoren posamezen član.

O kakovosti ter hitrosti razvoja ne moremo komentirati, saj nimamo izkušenj z delom na podobnih projektih.

Na koncu vsakega sprinta smo ugotovili, da nismo izrabili prednosti stalnega in enakomernega tempa razvoja, ki jo prinaša Scrum. V smislu razdelitve količine dela je bil naš razvoj bolj podoben tradicionalnemu, saj smo na

začetku sprinta delali bolj počasi, pred koncem pa občutno povečali količino dela.

Všeč nam je bil Sprint Review sestanek, saj smo ob demonstraciji izdelka stranki dobili njihove konkretne pomisleke glede določenih funkcionalnosti. S tem smo si pomagali, da smo izdelek razvili tako, kot so si zamislili v naročniškem podjetju. Pred tem sestankom smo si namreč nekatere funkcionalnosti predstavljali malo drugače.

Sprint Planning sestanki so potekali hitro, saj je ekipa znašala le 4 člane. Za ocenjevanje zahtevnosti smo uporabljali metodo Team Estimation Game [1], ki nam je bila vsem všeč, saj smo lahko zahtevnost določene uporabniške zgodbe primerjali z zahtevnostjo ostalih, ki smo jih že ocenili, in tako lažje, hitreje ter bolj natančno ocenili vse zgodbe.

Pri tem projektu smo imeli proste roke pri izbiri tehnologij za razvoj izdelka. S to možnostjo smo bili zelo zadovoljni. Naša ekipa je delala z ogrodjem Django (`https://www.djangoproject.com/`), saj sva dva člana s tem ogrodjem že delala in se nama je zdel idealen za ta projekt.

Ko smo se po koncu projekta z ostalimi skupinami pogovarjali, smo ugotovili, da so nam bile pri metodi Scrum najbolj všeč tri stvari:

- Sami smo si lahko izbrali tehnologijo ter orodja za razvoj izdelka.
- Celoten projekt je razpadel na manjše uporabniške zgodbe, te pa na še manjše naloge, ki so bile brez večjih težav obvladljive.
- Sodelovanje in medsebojna pomoč sta bila ključnega pomena za uspeh.

Poglavje 6

Zaključek

Agilni metodi Scrum popularnost raste iz leta v leto. Zanimalo nas je, ali je Scrum resnično boljši od tradicionalnega razvoja programske opreme.

V tem diplomskem delu smo opravili sistematični pregled literature na temo metode Scrum, da smo ugotovili njegove prednosti ter slabosti.

Ugotovili smo, da ima obstoječa literatura na Scrum izredno pozitiven pogled. Najpomembnejše prednosti so izboljšana komunikacija ter sodelovanje, izboljšana kakovost izdelkov, odzivnost ter učinkovitost razvojnih ekip, transparentnost razvojnega procesa ter zadovoljstvo tako razvijalcev kot strank.

Seveda pa literatura nudi vpogled tudi v slabosti metode Scrum, vendar je teh občutno manj.

V delu smo ugotovili tudi nekaj predlogov za najlažjo uveljavitev metode Scrum pri delu, ki jih literatura ponuja.

Da lahko sodelovanje ter komunikacija predstavljata veliko prednost metode Scrum, se je treba potruditi, tudi stranke se morajo popolnoma angažirati ter biti prisotne pri razvojnem procesu.

Sestanke je treba vzeti resno, da se od njih največ odnese.

Seveda pa je potrebno imeti za uspešno uporabo Scruma ekipo pravih ljudi.

Podjetja pa so lahko agilna pri implementaciji metode Scrum, kar pomeni, da si lahko metodo prilagodijo glede na svoje potrebe ter ugotovitve.

Nazadnje smo ugotovili tudi nekaj informacij o samih primarnih študijah. Večinoma so bile objavljene v zbornikih konferenc ter vsebovale študije primera, empirične študije ali pa preglede literature. Avtorji so za pridobivanje podatkov za študije v večini spremljali projekte, izvajali intervjuje ter ankete. Na koncu smo ugotovili, da le 5 izmed 27-ih primarnih študij ni obravnavalo tematike iz industrijskega okolja.

To delo priporočamo v branje vsem, ki želijo izvedeti več informacij o metodi Scrum ter kaj o njej, njenih prednostih in slabostih že piše v znanstveni literaturi.

Priloga A. Primarne študije

[Š1] S. Overhage and S. Schlauderer. Investigating the long-term acceptance of agile methodologies: An empirical study of developer perceptions in scrum projects. V *System Science (HICSS), 2012 45th Hawaii International Conference on*, strani 5452–5461, Januar 2012.

[Š2] M. Laanti. Agile and wellbeing – stress, empowerment, and performance in scrum and kanban teams. V *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, strani 4761–4770, Januar 2013.

[Š3] X. Zhang and B. Dorn. Agile practices in a small-scale, time-intensive web development project. V *Information Technology: New Generations (ITNG), 2011 Eighth International Conference on*, strani 1060–1061, April 2011.

[Š4] K. Nottonson and K. DeLong. Crawl, walk, run: 4 years of agile adoption at babycenter.com. V *Agile, 2008. AGILE '08. Conference*, strani 116–120, Avgust 2008.

[Š5] C. Mann and F. Maurer. A case study on the impact of scrum on overtime and customer satisfaction. V *Agile Development Conference (ADC'05)*, strani 70–79, Julij 2005.

[Š6] K. H. Judy and I. Krumins-Beens. Great scrums need great product owners: Unbounded collaboration and collective product ownership. V

Hawaii International Conference on System Sciences, Proceedings of the 41st Annual, strani 462–462, Januar 2008

[Š7] R. Moore, K. Reff, J. Graham, and B. Hackerson. Scrum at a fortune 500 manufacturing company. V *Agile Conference (AGILE), 2007*, pages 175–180, Avgust 2007

[Š8] M. Kajko-Mattsson, A. Aguiar, K. Boness, H. Kaindl, R. Pooley, and A. Tael. Long-term perspective of agile methods. V *Software Engineering Advances, 2009. ICSEA '09. Fourth International Conference on*, strani 1–2, September 2009

[Š9] L. Rising and N. S. Janoff. The scrum software development process for small teams. *IEEE Software*, 17(4):26–32, Julij 2000.

[Š10] V. P. Eloranta, K. Koskimies, T. Mikkonen, and J. Vuorinen. Scrum anti-patterns – an empirical study. V *2013 20th Asia-Pacific Software Engineering Conference (APSEC)*, volume 1, strani 503–510, December 2013.

[Š11] M. Gannon. An agile implementation of scrum. V *Aerospace Conference, 2013 IEEE*, strani 1–7, Marec 2013.

[Š12] L. Williams, G. Brown, A. Meltzer, and N. Nagappan. Scrum + engineering practices: Experiences of three microsoft teams. V *2011 International Symposium on Empirical Software Engineering and Measurement*, strani 463–471, September 2011.

[Š13] L. Pries-Heje and J. Pries-Heje. Why scrum works: A case study from an agile distributed project in denmark and india. V *Agile Conference (AGILE), 2011*, strani 20–28, Avgust 2011.

[Š14] A. Njeguš and G. Milanov. Qualitative comparison of agile and iterative software development methodologies. V *Telecommunications Forum (TELFOR)*, 2011 19th, strani 1269–1272, November 2011.

[Š15] E. Hossain, M. A. Babar, and H. y. Paik. Using scrum in global software development: A systematic literature review. V *2009 Fourth IEEE International Conference on Global Software Engineering*, strani 175– 184, Julij 2009.

[Š16] A. Mundra, S. Misra, and C. A. Dhawale. Practical scrum-scrum team: Way to produce successful and quality software. V *Computational Science and Its Applications (ICCSA), 2013 13th International Conference on*, strani 119–123, Junij 2013.

[Š17] J. López-Martínez, R. Juárez-Ramírez, C. Huertas, S. Jiménez, and C. Guerra-García. Problems in the adoption of agile-scrum methodologies: A systematic literature review. V *2016 4th International Conference in Software Engineering Research and Innovation (CONISOFT)*, strani 141–148, April 2016.

[Š18] G. M. Kapitsaki and M. Christou. Where is scrum in the current agile world? V *Evaluation of Novel Approaches to Software Engineering (ENASE), 2014 International Conference on*, strani 1–8, April 2014.

[Š19] J. Sutherland and I. Altman. Organizational transformation with scrum: How a venture capital group gets twice as much done with half the work. V *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, strani 1–9, Januar 2010.

[Š20] Tore Dybå and Torgeir Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9–10):833 – 859, 2008.

[Š21] Jibi Abraham, Vishal Bhatnagar, Ashish Agrawal, Mohd. Aurangzeb Atiq, and L.S. Maurya. A current study on the limitations of agile methods in industry using secure google forms. *Procedia Computer Science*, 78:291 – 297, 2016.

[Š22] Iva Krasteva and Sylvia Ilieva. Adopting an agile methodology: Why it did not work. V *Proceedings of the 2008 International Workshop on Scrutinizing Agile Practices or Shoot-out at the Agile Corral*, APOS '08, strani 33–36, New York, NY, USA, 2008. ACM.

[Š23] Kevin Vlaanderen, Peter van Stijn, Sjaak Brinkkemper, and Inge van de Weerd. Growing into agility: Process implementation paths for scrum. V *Proceedings of the 13th International Conference on Product-Focused Software Process Improvement*, PROFES'12, strani 116–130, Berlin, Heidelberg, 2012. Springer-Verlag.

[Š24] Torgeir Dingsøyr, Geir Kjetil Hanssen, Tore Dybå, Geir Anker, and Jens Olav Nygaard. Developing software with scrum in a small cross-organizational project. V *Proceedings of the 13th European Conference on Software Process Improvement*, EuroSPI'06, strani 5–15, Berlin, Heidelberg, 2006. Springer-Verlag.

[Š25] R. Vijay Anand and M. Dinakaran. Issues in scrum agile development principles and practices in software development. *Indian Journal of Science and Technology*, 8(35), 2015.

[Š26] H. Hajjdiab, A.S. Taleb, and J. Ali. An industrial case study for scrum adoption. *Journal of Software*, 7(1):237–242, 2012.

[Š27] H.F. Landim, A.B. Albuquerque, and T.C. Macedo. Procedures and conditions that influence on the efficiency of some agile practices. V *7th International Conference on the Quality of Information and Communications Technology, QUATIC 2010*, strani 385–390, 2010.

Literatura

[1] How to play the team estimation game. http://www.agilelearninglabs.com/2012/05/how-to-play-the-team-estimation-game/. (Dostop 4. avgust 2016).

[2] Scrum (software development) - wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Scrum_(software_development). (Dostop 27. marec 2016).

[3] Systematic review - wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Systematic_review. (Dostop 7. junij 2016).

[4] International Scrum Institute. Scrum institute. http://www.scrum-institute.org/, 2016. (Dostop 27. marec 2016).

[5] Jeff Sutherland, Ken Schwaber, Co-creators Of Scrum, and Contact Jeff Sutherl. The scrum papers: Nuts, bolts, and origins of an agile process. 2007.

[6] Laurie Williams. Agile software development methodologies and practices. *Advances in Computers*, 80:1–44, 2010.