Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Domen Perenič

Analiza točnosti ocen uporabniških zgodb

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:

Analiza točnosti ocen uporabniških zgodb

Tematika naloge:

Pri agilnem razvoju programske opreme se za ocenjevanje zahtevnosti uporabniških zgodb največkrat uporablja metoda planning poker. V zadnjem času pa postaja vse bolj popularen način ocenjevanja po metodi team estimation game. Predstavite obe metodi in analizirajte točnost ocen na podlagi podatkov, pridobljenih v okviru študije, ki je potekala pri predmetu Tehnologija programske opreme v študijskem letu 2014/15. Opišite zasnovo študije in postopek zbiranja podatkov ter z ustreznimi statističnimi metodami ugotovite, ali obstaja statistično pomembna razlika med ocenami, pridobljenimi po eni in drugi metodi. Pri tem izhajajte iz že obstoječe znanstvene literature, ki obravnava problematiko skupinskega ocenjevanja.

Izjava o avtorstvu zaključnega dela

Spodaj podpisani Domen Perenič, vpisna številka 63060262, avtor zaključnega dela z naslovom:

Analiza točnosti ocen uporabniških zgodb (angl. *Accuracy analysis of user story estimates*)

IZJAVLJAM

1. da sem pisno zaključno delo študija izdelal samostojno, pod mentorstvom prof. dr. Viljana Mahniča;
2. da je tiskana oblika pisnega zaključnega dela študija istovetna elektronski obliki pisnega zaključnega dela študija;
3. da sem pridobil/-a vsa potrebna dovoljenja za uporabo podatkov in avtorskih del v pisnem zaključnem delu študija in jih v pisnem zaključnem delu študija jasno označil/-a;
4. da sem pri pripravi pisnega zaključnega dela študija ravnal/-a v skladu z etičnimi načeli in, kjer je to potrebno, za raziskavo pridobil/-a soglasje etične komisije;
5. soglašam, da se elektronska oblika pisnega zaključnega dela študija uporabi za preverjanje podobnosti vsebine z drugimi deli s programsko opremo za preverjanje podobnosti vsebine, ki je povezana s študijskim informacijskim sistemom članice;
6. da na UL neodplačno, neizključno, prostorsko in časovno neomejeno prenašam pravico shranitve avtorskega dela v elektronski obliki, pravico reproduciranja ter pravico dajanja pisnega zaključnega dela študija na voljo javnosti na svetovnem spletu preko Repozitorija UL;
7. dovoljujem objavo svojih osebnih podatkov, ki so navedeni v pisnem zaključnem delu študija in tej izjavi, skupaj z objavo pisnega zaključnega dela študija.

V Ljubljani, dne 8. decembra 2016 Podpis študenta/-ke:

Zahvaljujem se mentorju prof. dr. Viljanu Mahniču in doc. dr. Tomažu Hovelji za nasvete ter pomoč pri izdelavi diplomske naloge. Zahvaljujem se tudi vsem bližnjim za podporo med študijem.

Kazalo

Povzetek

Abstract

1 Uvod **1**

2 Agilne metodologije **5**

2.1 Uvod 5

2.2 Manifest agilnega razvoja programske opreme 6

2.3 Scrum 7

3 Metode ocenjevanja **13**

3.1 Skupinske metode ocenjevanja 14

3.2 Vrste skupinskih metod ocenjevanja 15

3.3 Raziskovalna vprašanja 24

4 Potek študije **27**

4.1 Sprint 0 28

4.2 Vloge 29

4.3 AC Scrum 30

4.4 Seznam zahtev 30

4.5 Potek sprintov 31

4.6 Ocenjevanje zahtev 35

5 Analiza točnosti ocen **37**

5.1 Statistika 38

5.2 Kako *Planning poker* vpliva na spremembo začetnih ocen posameznikov? 39

5.3 Kako se *Team estimation game* primerja s *Planning pokrom*? . 42

5.4 Veljavnost študije 44

6 Zaključek **47**

Seznam uporabljenih kratic

kratica	angleško	slovensko
BRE	Balanced measure of relative error	Uravnotežena relativna napaka
BREbias	Balanced measure of relative error with bias	Uravnotežena relativna napaka z usmerjenostjo
TEG	Team estimation game	Team estimation game
TPO	Software Engineering	Tehnologija programske opreme

Povzetek

Naslov: Analiza točnosti ocen uporabniških zgodb

Agilne metode so trenutno prevladujoč način razvoja programske opreme. V diplomskem delu so predstavljeni glavni principi agilnega razvoja, s poudarkom na metodi *Scrum*, in nekatere skupinske metode ocenjevanja. Za ocenjevanje zahtevnosti uporabniških zgodb se največkrat uporablja metoda *Planning poker*, v zadnjem času pa postaja vse bolj popularna tudi metoda *Team estimation game*, tako da je bil poudarek na tema dvema metodama. Opravljena je bila analiza točnosti ocen na podlagi podatkov, pridobljenih v okviru študije, ki je potekala pri predmetu Tehnologija programske opreme v študijskem letu 2014/15. Opisana je zasnova študije in potek zbiranja podatkov. Slednji so bili z ustreznimi statističnimi metodami analizirani, rezultati pa podrobno razloženi. Cilj diplomske naloge je bil ugotoviti kako *Planning poker* vpliva na spremembo statistične kombinacije začetnih ocen posameznikov in ali obstaja statistično pomembna razlika med ocenami, pridobljenimi po tej metodi in po metodi *Team estimation game*. Statistična analiza je pokazala, da je, čeprav z zanemarljivo velikostjo učinka, uporaba metode *Planning poker* zmanjšala stopnjo optimizma, prisotnega pri statistični kombinaciji začetnih ocen. *Team estimation game* pa je privedel do še manj optimističnih ocen kot *Planning poker*. Primerjava točnosti ocen ni pokazala statistično značilnih razlik.

Ključne besede: agilni razvoj programske opreme, analiza, ocene, planning poker, skram, točnost, team estimation game, uporabniška zgodba.

Abstract

Title: Accuracy analysis of user story estimates

Agile methods are currently the dominant way of software development. This thesis presents the main principles of agile software development with emphasis on the method Scrum and some of the group techniques for software estimation. Planning poker is the most often used technique for estimating the size of user stories, but lately Team estimation game has been gaining popularity too, so emphasis was put on these two. An accuracy analysis of user story estimates based on data, collected during a study that was carried out at Software Engineering course in year 2014/15, was performed. Design of the study and the process of collecting the data is described. The data was analysed using appropriate statistical methods and the results were explained in detail. The goal of the thesis was to find out how final estimates obtained through Planning poker differ from the statistical combination of the estimates obtained during the first round of the process and whether there is a statistically important difference between the estimates obtained through Planning poker and those obtained through TEG. Our analysis showed that Planning poker decreased the over-optimism present in the statistical combination of initial estimates. However, the size of effect was very small. Estimates obtained through TEG were even less optimistic than those obtained through Planning poker. Accuracy analysis of estimates showed no statistically significant difference.

Keywords: accuracy, agile software development, analysis, estimates, planning poker, scrum, team estimation game, user story.

Poglavje 1

Uvod

Za razvoj programske opreme se je razvilo že kar nekaj metod, ki jih v glavnem uvrščamo v tradicionalne in agilne.

Čeprav se je tradicionalni pristop razvijal in uporabljal od samega začetka razvoja programske opreme, ni prinašal želenih rezultatov. Le majhen delež projektov pod tem pristopom je bil namreč uspešno končan. Največja pomankljivost tradicionalnih metod je bila ta, da so naročniki in uporabniki prišli do delujoče programske opreme šele na koncu projekta.

Želja po izboljšanju postopka razvoja je bila glavni razlog za pojavitev agilnih metodologij. Izboljšanje planiranja in razvoja so si začetniki tega gibanja zamislili tako, da bi omogočalo redno dostavo delujoče programske opreme. Ta ideja se je med razvijalci hitro prijela in agilne metode so postale prevladujoč način razvoja programske opreme.

Pri agilnih projektih so programske zahteve predstavljene v obliki uporabniških zgodb (angl. *User stories*). Slednje razvijalcem služijo kot osnova za planiranje projekta, uporabljajo se kot vodilo pri implementaciji želenih funkcionalnosti, hkrati pa so ravno uporabniške zgodbe tisti element razvoja, ki se ga ocenjuje. Ocenjevanje je pomemben del planiranja, saj lahko le preko ocen vseh uporabniških zgodb pridemo tudi do časovne ocene projekta, ki nam pove, kdaj naj bi bilo delo zaključeno in izdelek pripravljen na izdajo ter uporabo. Točnost ocen tako neposredno vpliva na kakovost celotnega plana.

Ocenjuje se zahtevnost vsake zgodbe posebej, enota, ki se pri tem uporablja pa je točka (angl. *Story point*).

Metod za ocenjevanje je veliko, a v naši diplomski nalogi smo se osredotočili na dve. To sta *Planning poker* in *Team estimation game* (TEG). Oboje sta metodi skupinskega ocenjevanja, ki sta med najbolj razširjenimi v agilnih metodologijah.

Skupinske metode po eni strani nudijo veliko koristi, a zraven pridejo tudi določene nevarnosti skupinskega dela, ki bi lahko slabo vplivale na končni rezultat ocenjevanja. Ker je agilnost še vedno razmeroma nov pristop pri razvoju programske opreme, je bilo do sedaj na tem področju izvedeno le majhno število raziskav, njihovi rezultati pa niso ravno enotni. Tako še vedno ni jasno, kako učinkovite so skupinske metode ocenjevanja, kar je tudi glavni vir naše motivacije.

Objavljenih je bilo le malo raziskav povezanih s *Planning pokrom*. Ker se poleg pomanjkanja raziskav v njih pojavljajo še nasprotovanja glede vpliva metode na končno oceno, smo to preverili. Natančneje, preverili smo, če se z uporabo te metode točnost in optimizem začetnih ocen posameznikov poveča ali zmanjša. Podobnih raziskav o TEG, kot smo si jo zamislili v tej nalogi, sploh nismo našli. Tako smo izvedli tudi primerjavo te metode s *Planning pokrom* in preverili, če je ena izrazito boljša od druge. Primerjali smo njuno točnost in nagnjenost k optimizmu.

Za osnovo te naloge smo uporabili podatke študije, kjer je večje število skupin študentov, med katerimi sem bil tudi sam, razvijalo enak projekt. Razvoj je potekal po metodi *Scrum*, ki je vodilen način implementacije agilnosti v razvoj programske opreme. Programske zahteve so bile tudi pri nas predstavljene v obliki uporabniških zgodb, kot je standard pri agilnem razvoju. Ocenjevanje zgodb je potekalo po dveh metodah. S tem namenom so bile razvojne skupine (angl. *Teams*) razdeljene na dva dela. Prvi so izvajali ocenjevanje po metodi *Planning poker*, drugi pa po TEG.

V diplomski nalogi smo analizirali in interpretirali podatke, ki smo jih zbrali tekom opisane študije. Z rezultati našega dela bi radi prispevali k

empiričnim študijam, ki se nanašajo na skupinsko ocenjevanje pri razvoju programske opreme z agilno metodologijo.

V poglavju 2, kjer je povzeta teorija s Cohnovega vidika [2, 3], bomo najprej več izvedeli o sami agilnosti, Scrumu in uporabniških zgodbah. V nadaljevanju (Poglavje 3) bosta podrobneje opisani izbrani metodi ocenjevanja, *Planning poker* in TEG. Zraven bomo za primerjavo spoznali še nekaj drugih metod in tudi splošnih značilnosti skupinskega ocenjevanja. Podrobnosti o naši študiji so opisane v poglavju 4. Sledijo še rezultati analize (Poglavje 5) in zaključne misli (Poglavje 6).

Poglavje 2

Agilne metodologije

2.1 Uvod

Čeprav se je začelo pojavljati že prej, se je agilno gibanje uradno začelo leta 2001, z objavo Manifesta agilnega razvoja programske opreme [1]. Napisala ga je skupina 17 strokovnjakov, ki so se poimenovali Agilno združenje (angl. *Agile Alliance*). Povod za izdelavo manifesta je bilo dejstvo, da je razvoj programske opreme po tradicionalni metodologiji dosegal zelo nizek delež uspešno končanih projektov. Skupina *Standish group* je leta 2015 izdala poročilo o analizi zadnjih let [5], ki kaže le 11% uspešno zaključenih in kar 29% spodletelih projektov. V istem obdobju imajo agilne metode kar 39% uspešno zaključenih in le 9% spodletelih projektov.

Tradicionalna metodologija temelji na obsežni dokumentaciji in zaporednem razvoju. Razdeljena je na več faz, to so ponavadi analiza zahtev, planiranje, implementacija, testiranje in vzdrževanje. Razvoj ne napreduje v naslednjo fazo, dokler trenutna ni končana.

Tukaj pa se že pokažejo prve slabosti. Uporabniki tako namreč pridejo do uporabne programske opreme šele, ko je cel projekt končan. Pri tradicionalnem procesu razvoja se predvideva, da pravilna izvedba zgodnjih aktivnosti s sledenjem vnaprej točno določenih postopkov in zahtev zmanjša tako število kasnejših popravkov kot tudi stroškov razvoja. S tem pa se onemogoči

zmožnost prilagajanja nepričakovanim spremembam in težavam, kar je tista glavna pomankljivost, ki jo agilne metodologije skušajo odpraviti. Ker se morajo razvijalci pri tradicionalih metodah strogo držati plana, to povzroča pri njih še dodaten stres in namesto da bi se osredotočili na kvaliteto programske opreme, se osredotočijo na določene ocene in postavljene časovne roke, kar spet negativno vpliva na razvoj. Med razvojno skupino, uporabniki in naročniki, je veliko manj komunikacije kot pri agilnem procesu, kar je še ena pomankljivost tradicionalne metodologije.

Leta 2009 je Forrester opravil raziskavo [13], kjer je že takrat 35% anketirancev izjavilo, da uporabljajo agilne metode. Vse kaže na hitro naraščujočo popularnost agilnosti.

2.2 Manifest agilnega razvoja programske opreme

V manifestu so zapisane najpomembnejše vrednote in principi agilnosti, ki naj bi omogočali hitrejši in boljši razvoj ter prilagajanje nenadnim spremembam. Glavne štiri vrednote, ki zagotavljajo, da se razvoj programske opreme odvija po agilni metodologiji, so naslednje:

- `posamezniki in interakcije` pred procesi in orodji,
- `delujoča programska oprema` pred vseobsežno dokumentacijo,
- `sodelovanje s stranko` pred pogodbenimi pogajanji,
- `odziv na spremembe` pred togim sledenjem načrtom.

"Četudi cenimo dejavnike na desni, vseeno bolj cenimo tiste na levi." [1]

To pomeni, da agilni razvoj bolj ceni posameznike in interakcijo med njimi kot proces in orodja, ker vedo, da bo dobro delujoča razvojna skupina prinesla boljše rezultate. Agilni proces sprejema prednosti in slabosti vsakega posameznika. Prve se preko skupinskega sodelovanja izkoristijo, slabosti pa odpravljajo.

Agilnost je iterativen proces, kar dobro ponazarja načelo, ki pravi, da je delujoča programska oprema več vredna kot vseobsežna dokumentacija. S takim pristopom imamo na koncu vsake iteracije izboljšano verzijo in delujočo programsko opremo. To omogoča tudi zgodnjo in boljšo komunikacijo z naročnikom. Zagotovi se, da razvojna skupina v vsaki iteraciji dela na najbolj pomembnih funkcijah, s čimer se hkrati zadovolji naročnika in uporabnike.

S tem smo že načeli tretjo točko. Pogajanja okoli pogodbe niso tako pomembna kot sodelovanje s stranko. Pogodbe so pomembne pri projektih ampak podrobnosti in zastavljeni pogoji včasih povzočijo slabe odnose med razvojno skupino in naročnikom. Bolj pomembno je, da vsi vpleteni v projekt, sodelujejo k skupnemu cilju. To se nanaša na razvojno skupino, naročnika in potencialne stranke ter uporabnike.

Zmožnost prilagajanja spremembam se bolj ceni kot slepo sledenje načrtu. Cilj vsake iteracije je dostaviti čimveč delujoče programske opreme z najvišjo prioriteto. Pri večini projektov je nemogoče vnaprej določiti vse podrobnosti in zahteve naročnika. Med samim razvojem pa tudi razvijalci pridobivajo na izkušnjah in znanju. Zato se na začetni plan gleda le kot na grobo vodilo. Agilna skupina upošteva vse te faktorje in se jim sproti prilagaja po svojih najboljših močeh [3].

2.3 Scrum

V sklopu agilnosti je nastalo več metod za razvoj programske opreme. Nekatere med njimi so Scrum, ekstremno programiranje [14], *Scrumban* [16] in *Kanban* [15]. Pri naši študiji smo projekt razvijali po Scrumu, ki je tudi daleč najbolj razširjen med agilnimi metodami.

Scrum daje poudarek na medsebojno komunikacijo in način, kako razvojna skupina dela, bolj kot na samo delo, ki se ga opravlja. Pomemben je torej postopek razvoja. Sledi principom agilnega manifesta in tako dostavlja visoko kakovostno programsko opremo. Uporablja ga vedno več razvijalcev,

ker je preprost, praktičen in popularen [13].

Je iterativen in inkrementalen proces. Napredek se kaže skozi zaporedno izpopolnitev. Scrum gradi na procesu učenja in razvojni skupini daje orodja za samoorganizacijo in sprotno izboljšanje kakovosti ter hitrosti njihovega dela [12]. Planira delo za eno iteracijo, kateremu sledijo izboljšave v naslednjih. Programska oprema se tako razvija in objavlja po delih. Vsak del, objavljen na koncu iteracije, je zaključen sklop nekih funkcionalnosti (angl. *Potentially Shippable Product Functionality*) [2].

2.3.1 Potek Scruma

Povedali smo že, da se projekti po tej metodi razvijajo v zaporednih iteracijah. Ena iteracija se drugače imenuje tudi sprint (angl. *Sprint*).

Dolžina sprintov se od projekta do projekta lahko precej razlikuje, saj nanjo vpliva veliko faktorjev, kot so velikost in zahtevnost projekta, izkušenost razvojne skupine itd. Cohn [3] priporoča dvotedensko dolžino posamezne iteracije.

Scrum ima točno določene vloge za vse, ki sodelujejo pri projektu. Na vrhu je skrbnik izdelka (angl. *Product owner*), ki predstavlja stranko. Skrbnik izdelka je oseba z vizijo o končnem cilju in skrbi za vsebino izdelka [12]. Sledi mu skrbnik procesa (angl. *Scrum master*), katerega glavna naloga je, da razvojni skupini omogoča nemoteno delo in poskrbi za sledenje osnovnim pravilom Scruma. Skupino ponavadi sestavlja od 4 do 7 razvijalcev in je samoorganizirajoča. To pomeni, da je vodstvo odgovorno za oblikovanje strateških ciljev, razvojna skupina pa za način, na katerega jih bo dosegla [12].

Pri Scrumu so vse želene funkcionalnosti izdelka zapisane na seznamu zahtev (angl. *Product backlog*). Skozi razvoj se ta ves čas osvežuje. Lahko se mu doda nove zahteve ali odstrani tiste, ki izgubijo na svojem pomenu. Še nedokončane zahteve se lahko med iteracijami spreminjajo in popravljajo. Vedno torej vsebuje seznam vseh funkcionalnosti, ki so trenutno zaželene pri končnem izdelku. Zahteve so ponavadi predstavljene kot uporabniške zgodbe

(Poglavje 2.3.2).

Tekom razvoja Scrum predpisuje več sestankov, ki poskrbijo za komunkacijo med vsemi soudeleženci projekta in tako tudi za boljši razvoj izdelka.

Na začetku vsake iteracije se izvede sestanek za planiranje sprinta (angl. *Sprint planning meeting*). Ta ponavadi traja en delovni dan in udeležijo se ga lahko vsi, ki sodelujejo pri projektu. Skrbnik izdelka pred začetkom dela na posamezni izdaji predstavi vse ali vsaj najbolj pomembne uporabniške zgodbe. Vedno se najprej predstavi tiste z najvišjo prioriteto, saj te že nakazujejo cilj sprinta.

Za predstavitvijo zgodb pride na vrsto razvojna skupina s svojimi vprašanji glede le-teh. Takšna komunikacija med njimi in skrbnikom izdelka jim omogoči boljši vpogled v zahtevnost posameznih zgodb, kar je v veliko pomoč pri ocenjevanju, ki tej razpravi sledi. Tako lahko namreč pridemo do bolj točnih ocen zgodb. Slednje pa vodijo tudi do bolj točne ocene časa, potrebnega za razvoj celotnega izdelka, kar ima pomembno vlogo pri planiranju projekta. Več o ocenjevanju uporabniških zgodb sledi v poglavju 4.6.

Razvojna skupina določi hitrost (angl. *Velocity*) oziroma količino dela, za katerega verjame, da ga bo lahko opravila v enem sprintu. Pri tem se kot enota uporabljajo prej določene ocene zgodb.

Skupina nato izbere zgodbe, katere namerava razviti v tekoči iteraciji, pri čemer upošteva njihovo oceno in prioriteto. Prednost imajo tiste z najvišjo prioriteto. Izbrane zgodbe se nato premaknejo na drug seznam, ki določa vsebino sprinta (angl. *Sprint backlog*). Te zgodbe se nato razdelijo še na več manjših nalog (angl. *Tasks)*, kar še dodatno pripomore k boljši definiciji in razumevanju zgodb.

Sledi začetek dela. Če to poteka hitreje, kot je bilo pričakovano, in pred koncem sprinta zmanjka izbranih zgodb, se seznamu z vsebino sprinta lahko doda še katero s seznama vseh zahtev. Skrbnik izdelka v tem primeru določi dodatne zgodbe, ki jih razvojna skupina lahko doda v tekoči sprint.

V iskanju najučinkovitejšega načina, kako zbirati in deliti podatke o razvoju projekta, so bili uvedeni dnevni Scrum sestanki (angl. *Daily scrum*).

To so kratki vsakodnevni sestanki, kjer udeleženci odgovorijo na tri vprašanja.

- `Kaj si naredil včeraj?`
- `Kaj boš naredil danes?`
- `Imaš kakšne težave pri delu?`

Ti sestanki tako ponujajo dnevni vpogled v napredek dela in razvojni skupini omogočajo boljše sprotno prilagajanje nepričakovanim težavam. Vidi se, kaj je kdo že naredil in na čemu namerava nadaljevati delo, kar olajša koordinacijo procesa. Pokaže se tudi, če ima kateri član skupine težave pri svojem delu, v katerem primeru mu lahko ostali pomagajo priti do rešitve.

Vsak sprint mora dostaviti nadgradnjo izdelka, ki je že pripravljena za potencialno izdajo [2]. To pomeni, da mora razvojna skupina tekom vsakega sprinta proizvesti sprogramirano, stestirano in uporabno programsko opremo. Svoje dosežke pokažejo na koncu sprinta, na posebnem sestanku (angl. *Sprint review meeting*), kjer se demonstrira in preveri realizirane zgodbe. Sodelujejo vsi udeleženci, to so razvojna skupina, skrbnik procesa in skrbnik izdelka ter tudi potencialni uporabniki.

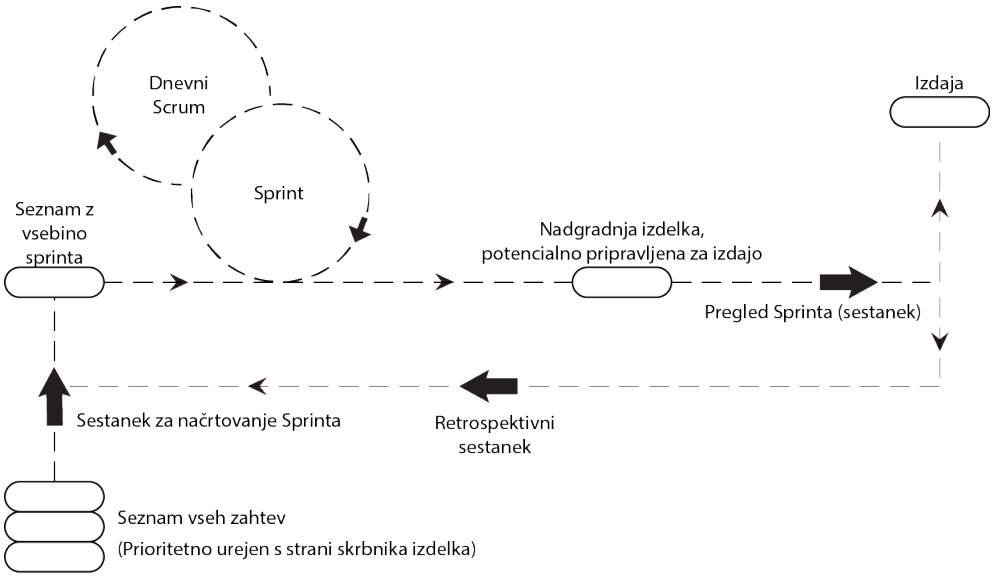
Zahteve, ki so bile popolnoma realizirane in jih skrbnik izdelka potrdi, predstavljajo naslednjo nadgradnjo izdelka in se izdajo za takojšnjo uporabo.

Pred začetkom naslednjega sprinta sledi še zadnji sestanek v tem krogu, ki se mu reče retrospektivni sestanek ali kar retrospektiva sprinta (angl. *Sprint retrospective meeting*). Preveri se, če so bili izpolnjeni cilji, zadani na sestanku za planiranje sprinta oziroma v kolikšni meri so bili. Analizira se delo in težave, ki so nastale med sprintom, kar pripomore k boljšemu planiranju naslednjega.

Splošno uveljavljen potek Scruma je za lažjo predstavo prikazan še na sliki 2.1.

2.3.2 Uporabniške zgodbe

Uporabniške zgodbe opisujejo funkcionalnosti, ki se naročniku zdijo vredne razvoja v sklopu njihovega izdelka. Poudarka vredno dejstvo je ravno to,



Slika 2.1: Grafični prikaz poteka Scruma od seznama zahtev, ki ga pripravi skrbnik izdelka, do objave. Dolžine iteracije se razlikujejo, a kot smo že povedali, Cohn priporoča 14 dni.

da so napisane s strani naročnika oziroma tako, da jih ta razume tudi, če se ne spozna na razvoj programske opreme in z njim povezano terminologijo. Razvojni skupini tako lažje razloži, kaj pričakuje od realizirane zgodbe. Naročnik mora biti vpleten v proces razvoja od začetka do konca. To je tudi ena večjih sprememb, ki jih agilnost z uporabo uporabniških zgodb prinaša v primerjavi s tradicionalnim načinom razvoja [2].

Uporabniška zgodba je sestavljena iz treh delov. Zapisan opis in pogovori glede zgodbe, ki pomagajo definirati podrobnosti, sta dva. Tretji del so sprejemni testi.

Opis in pogovori glede zgodbe omogočajo razvojni skupini boljše razumevanje, kaj se zahteva od njih. Boljše razumevanje pa vodi tudi do točnejših ocen zgodb. Merska enota, s katero razvijalci preko izbrane metode ocenjevanja ocenijo zahtevnost in velikost zgodbe, se preprosto imenuje točka. Je relativna ocena, ker se določa na podlagi primerjave med različnimi zgodbami. To je tudi razlog za učinkovitost točk, saj popolnoma točne ocene za

posamezno zgodbo ni mogoče določiti.

Sprejemni testi se uporabljajo kot kriterij, ki na koncu pokaže, če je bila zgodba realizirana tako, kot si je skrbnik izdelka zamislil. Le če je temu res tako, lahko zgodbo označimo kot zaključeno.

Tradicionalno se te tri vidike zapiše na papirnato kartico. Čeprav so na njej predstavljene zahteve naročnika, sta bolj pomembna ostala dva vidika uporabniške zgodbe. Pogovor med razvojno skupino in naročnikom, ki določi podrobnosti zgodbe in sprejemni testi, kateri jih na tudi nek način zabeležijo ter služijo za potrditev njihove izvedbe.

Poglavje 3

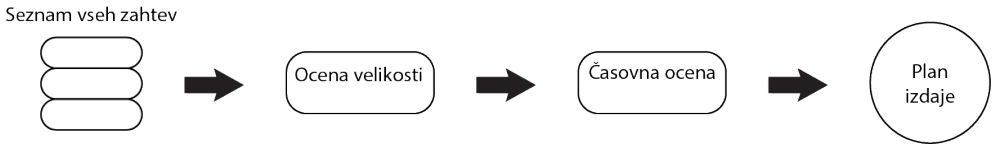
Metode ocenjevanja

Ocenjevanje ima kot sestavni del planiranja velik vpliv na razvoj programske opreme. Netočnost ocen zahtev in uporaba neučinkovitih metod ocenjevanja sta namreč med glavnimi vzroki za časovni in cenovni presežek projektov. Slednja se pri razvoju programske opreme kažeta v kar 30 % - 40 % projektov [4]. To lahko pomeni neuspešnost projekta ali vsaj nižjo kakovost razvite programske opreme.

Točne ocene v nasprotju omogočajo točno predvideti čas, v katerem naj bi bil izdelek dokončno razvit in pripravljen na izdajo. To pomeni tudi boljši plan in projekt se tako lahko izogne prekoračitvi predvidenega časa ter stroškov, kar močno izboljša verjetnost uspešnega zaključka projekta.

Da pridemo do časovne ocene, ko naj bi bil izdelek končan, moramo torej najprej oceniti njegovo velikost [3]. Postopek je prikazan na sliki 3.1. Problem pri ocenjevanju velikosti projekta pa je, da tako kot pri vseh ostalih aktivnostih, kjer se ugiba, kaj se bo zgodilo v prihodnosti, tudi tu ni mogoče zagotovo vedeti [4].

Vidno je, da ima ocenjevanje pri planiranju projektov res veliko vlogo. Raziskave kažejo, da je najpogostejše uporabljena metoda ocenjevanja, uporaba mnenja izkušenih strokovnjakov [4, 8]. Za dosego ocene velikosti projekta oziroma posamezne funkcionalnosti je potreben posvet z enim ali več strokovnjaki. Predvideva se, da lahko slednji s svojimi izkušnjami, znanjem



Slika 3.1: Plan izdaje. Ocenjevalci dobijo seznam vseh zahtev, ki morajo biti v projektu uresničene. Ocenijo velikost vsake od teh, ki se nato pretvori v čas potreben za njihovo realizacijo. Šele zatem se lahko naredi celoten plan razvoja projekta.

in razumevanjem danega projekta pridejo do najbolj točne ocene.

3.1 Skupinske metode ocenjevanja

Skupinske metode temeljijo na kolektivnem znanju več ljudi. Na področju razvoja programske opreme so takšne metode še vedno v fazi uvajanja, ker še niso bile dovolj raziskane in tako ni močnih dokazov o njihovi učinkovitosti. Se pa pojavljajo študije, ki kažejo, da takšen pristop lahko prinese precej koristi k procesu ocenjevanja programske opreme.

Sodelovanje ljudi z različnimi ozadji in znanjem naj bi omogočilo skupini v primerjavi s posameznikom najti več potencialnih težav, učinkovitejše rešitve zanje in delati boljše odločitve tekom ocenjevanja, planiranja ter samega vodenja projekta. Računa se na to, da se v skupinskem postopku zmanjša vpliv posameznika in bolj enakovredno upošteva vse udeležene. S tem naj bi se med drugim zmanjšal optimizem ocen, kateri se pripisuje ocenjevanju glede na mnenje strokovnjakov. To trditev bomo preverili tudi v naši nalogi. Za najboljšo izkoriščenost potenciala skupine so torej raznolikost, neodvisnost in enakopravnost članov zelo pomembni faktorji [4, 8].

Raziskave na področju razvoja programske opreme kažejo, da so tako dobljene ocene bolj točne kot ocene posameznika [4]. Kljub temu se porajajo tudi vprašanja glede negativnih posledic skupinskih metod in ali je skupinsko reševanje težav res bolj učinkovito. Njihova produktivnost naj ne bi

izkoriščala celotnega potenciala skupine zaradi težav koordinacije in socialne dinamike v skupini.

Težave s koordinacijo se pojavljajo pri uporabi zapletenih metod in lahko pri projektu povzročijo visoko režijo. S socialno dinamiko je povezanih več potencialnih nevarnosti. Ena takih je neenakopravnost članov skupine. V takem primeru prevladuje mnenje bolj vplivnega člana v skupini, kateremu se ostali prilagodijo.

Še dve nevarnosti skupinskega dela, ki spadata k socialni dinamiki skupine, sta skupinsko mišljenje in polarizacija. O prvem govorimo, ko se skupine preveč osredotočijo na dosego soglasja in zato upoštevajo le informacije, ki potrjujejo stališče, za katero mislijo, da je pravilno. S tem pa zanemarijo pravilno analizo težave ter možnih rešitev. O polarizaciji pri ocenjevanju govorimo, ko je končna skupinska ocena bolj ekstremna, kot je bila najbolj ekstremna začetna ocena med posamezniki. To pa direktno nasprotuje tudi zmanjševanju optimizma, ki ga pričakujemo od skupinskih metod ocenjevanja programske opreme [4].

V nadaljevanju bomo spoznali nekaj takšnih metod in naslovili ta vprašanja.

3.2 Vrste skupinskih metod ocenjevanja

Obstaja več vrst skupinskih metod. Statistična skupina je na primer skupina ljudi, med katerimi pri ocenjevanju ni nobene interakcije. Posamezne ocene so statistično združene v eno, končno [4].

Ko govorimo o nestrukturirani skupini, govorimo o skupini ljudi, ki med seboj delijo svoje vidike in skupaj pridejo do končne odločitve, ne da bi se pri tem morali držati nekih specifičnih navodil in strukture [4].

Podrobneje si bomo pogledali strukturirane skupine, saj k tem prištevamo tudi *Planning poker* in TEG. Proces posameznih strukturiranih skupin se precej razlikuje glede na to, koliko strukture metoda določa in koliko medsebojne interakcije v skupini dovoljuje [4]. Metode v naslednjih poglavjih (3.2.1 - 3.2.4) vse spadajo v to skupino.

3.2.1 *Delphi*

Metoda *Delphi* sledi naslednjim korakom:

- Koordinator najprej vsakemu strokovnjaku posebej predstavi specifikacijo zahtev.
- Predloži jim obrazec za ocenjevanje.
- Člani skupine anonimno izpolnijo obrazec, pri čemer lahko morebitna vprašanja postavljajo le koordinatorju.
- Koordinator po ocenjevanju sestavi povzetek odzivov skupine in razloge za podane ocene. Dopišejo se na obrazec, ki zahteva ponovno ocenjevanje. Postopek se ponavlja dokler je odstopanje ocen preveliko.

Delphi torej omejuje interakcijo med člani skupine, s čimer poskuša dobiti najbolj zanesljive ocene. V isti namen se za končno oceno vzame kar povprečje vseh ocen, saj se s tem poskuša zmanjšati vpliv posamezne. Poleg tega se z anonimnostjo poskuša zmanjšati vpliv posameznika v skupini, tako kot tudi vpliv večine na skupno oceno. Iteracija skupini omogoča, da se med ocenjevanjem učijo en od drugega in posledično mogoče spremenijo svoje ocene. Poudariti je potrebno, da vse to poteka preko koordinatorja in obrazcev za ocenjevanje. Dejanske razprave med člani skupine pri tej metodi ni [4].

3.2.2 *Wideband Delphi*

Gre se v bistvu za nadgrajeno različico metode *Delphi*. V primerjavi z njo je pri *Wideband Delphi* razširjena komunikacija v skupini, kar naj bi bil glavni primankljaj *Delphija*. Postopek metode je sledeč:

- Koordinator predstavi vsem udeležencem specifikacijo zahtev.
- Predloži jim obrazec za ocenjevanje.

- Skliče se skupinski sestanek, kjer vsi strokovnjaki skupaj s koordinatorjem predebatirajo o zahtevi in problemih njene realizacije.
- Strokovnjaki anonimno izpolnijo obrazce in jih predajo koordinatorju.
- Dokler se ne dobi ocen s sprejemljivo stopnjo odstopanja, se naslednji koraki ponavljajo:
  - Koordinator za skupino pripravi povzetek ocen na obrazcu za ponovno ocenjevanje.
  - Izvede se še en sestanek, katerega glavni namen je razprava strokovnjakov o točkah, kjer se njihove ocene najbolj razlikujejo.
  - Ponovno se anonimno izpolni obrazce.

Vidimo, da je postopek res precej podoben kot pri *Delphiju*. Največja novost pri tej metodi je torej ravno to, da strokovnjaki v živo razpravljajo o svojih ocenah [4].

3.2.3 *Planning poker*

Skupinska metoda *Planning poker* je ena novejših, kar se tiče ocenjevanja glede na mnenje več strokovnjakov. Je nadgrajena različica *Wideband Delphija*, prilagojena za agilni razvoj [4]. Najbolj se uporablja pri Scrumu in ekstremnem programiranju [8].

Potek sledi naslednjim korakom:

- Skrbnik izdelka predstavi uporabniško zgodbo in njene zahteve.
- Sledi razprava, ki pojasni podrobnosti predstavljene zgodbe. Razvijalci analizirajo zgodbo in delo, potrebno za implementacijo, pri čemer se lahko z dodatnimi vprašanji obrnejo na skrbnika projekta.
- Vsak član skupine anonimno napiše svojo oceno na papirnato kartico.
- Vse kartice se istočasno razkrijejo.

- Za razkritjem kartic sledijo dve možnosti:
  - Če je med ocenami preveliko odstopanje sledi ponovna razprava, s katero se poskuša priti do soglasja. Ocenjevanje se ponovi.
  - Če je ocena enotna, se jo shrani in gre naprej, na naslednjo zgodbo.

Metoda poskuša združiti pozitivne učinke strukturiranih metod, tako kot tudi koristi nestrukturiranih skupin [4]. Z medsebojno interakcijo članov se upa na to, da se bo zagotovilo dobro splošno razumevanje zahtev in ciljev ter tudi samih ocen, ki jih podajo ocenjevalci. K temu pripomore še istočasno razkritje vseh kartic, s čimer se zmanjša vpliv posameznega člana na skupino in s tem tudi na končno oceno.

Predvidoma naj bi *Planning poker* privedel do bolj točnih ocen, kot so tiste od posameznega strokovnjaka. To naj bi med drugim omogočal ravno s tem, ker vsi razvijalci sodelujejo pri procesu in enakovredno prispevajo k oceni [8].

3.2.4 *Team estimation game (TEG)*

TEG je še ena izmed novejših tehnik ocenjevanja in glede na našo poizvedbo relativno slabo raziskana in dokumentirana. Izhaja iz predpostavke, da je ocenjevanje velikega števila zahtev z metodami, kot je *Planning poker*, lahko zelo potratno. Cilj te metode je poskusiti uravnotežiti delo in čas, vložena v ocenjevanje zahtev, z dostavljeno vrednostjo ocen [10] .

TEG uporablja princip igre, s katero naj bi se postopek ocenjevanja olajšal in pohitril. Izboljšala pa naj bi se tudi točnost ocen. Ponavadi se igra na primerno veliki površini, na primer mizi, s fizičnimi karticami, na katerih so zapisane zahteve.

Podobno kot pri *Planning pokru* se ocenjuje relativna zahtevnost uporabniških zgodb in ne čas, potreben za njihovo realizacijo, le da TEG daje še večji poudarek na relativnost ocen. Enota, ki se jo pri tem uporablja, je tudi tu točka.

Potek metode se v bistvu deli na dve stopnji. Prva je urejanje kartic po stolpcih. Razvršča se jih relativno glede na pričakovano količino dela, vloženega v realizacijo zahtev, zapisanih na njih. V drugi stopnji se vsakemu stolpcu dodeli primerna ocena. Ponavadi vsebujejo stolpci na levi strani manj zapletene zahteve in s tem nižje ocene, stolpci na desni pa višje.

Ta način razvrščanja temelji na misli, da je ocenjevalcem lažje relativno primerjati težavnosti zahtev med seboj, kot da se preveč poglabljajo v podrobnosti in poskušajo vsako posebej točno oceniti.

Podrobneje gledano tehnika sledi naslednjim korakom:

- Prvi član skupine iz seznama zahtev izbere eno in jo postavi na stolpec po svoji izbiri.
- Naslednji član ima tri možnosti. Lahko:
  - Vzame naslednjo kartico in jo postavi v stolpec, relativno glede na prejšnjo. To lahko pomeni levo ali desno od tega stolpca ali pa kar na istega. Po potrebi se ustvari nove, vmesne stolpce.
  - Premakne eno od kartic, že postavljenih na igralno površino, na po svojem mnenju bolj primeren stolpec.
  - Preskoči potezo, če se strinja s porazdelitvijo vseh kartic.
- Ti koraki se ponavljajo dokler niso vse kartice razvrščene po stoplcih in vsi člani skupine preskočijo potezo. To pomeni, da se skupina strinja z relativno razporeditvijo zahtev.
- Stolpcem se določi ocena, ki predstavlja zapletenost vsebovanih zahtev.

3.2.5 Primerjava splošnih značilnosti metod

Moløkken-Østvold in drugi [9] ter Furulund [4] primerjajo različne skupinske metode glede na strukturo, anonimnost, interakcijo med člani in stopnjo potrebne režije, kot je to vidno v tabeli 3.1. V tabelo smo dodali še TEG, ki smo ga ocenili po svoji presoji. Statistične skupine so vključene bolj kot

Metoda	Struktura	Anonimnost	Interakcija	Režija
Statistične skupine	Lahka	Da	Brez	Omejena
Nestrukturirane skupine	Brez	Brez	Da	Omejena
Delphi	Težka	Da	Brez	Velika
Wideband delphi	Zmerna	Omejena	Omejena	Zmerna
Planning poker	Lahka	Brez	Da	Omejena
Team estimation game	Lahka	Brez	Da	Omejena

Tabela 3.1: Primerjava različnih metod.

zanimivost, nestrukturirane pa bomo še uporabili za primerjavo s strukturiranimi.

Anonimnost ima tu (Tabela 3.1) največji vpliv na ostale značilnosti in potek skupinskih metod. Z anonimnostjo se v skupini skuša ustvariti takšno okolje, kjer bi vsak član lahko podal svojo oceno, neodvisno od mnenja ostalih članov. Zmanjšati se skuša politični (npr. vpliv nadrejenega na ostale člane skupine) in družbeni vpliv na ocenjevalce, tako kot tudi polarizacijo. A istočasno se v procesu poveča režija. Z anonimnostjo se namreč pojavi potreba po koordinaciji procesa in s tem po strukturi metod. Odpadejo sicer razprave o zahtevah, a to lahko pomeni slabšo analizo le-teh in s tem več kasnejših popravkov [4].

Nestrukturirane skupine imajo pričakovano zelo nizko režijo. S pomanjkanjem strukture in predpisov, ki bi se jih morali držati pri postopku, se ocenjevanje izvede hitreje. Poleg tega vključujejo še nekaj pozitivnih značilnosti skupinskega dela, kot so delitev dela, odpravljanje pristranskosti med člani in boljša motivacija zaradi dela v skupini. Ravno s pomanjkanjem strukture pa so veliko bolj odprte za vse že omenjene negativne učinke skupine.

Tehnika *Delphi* ima očitno najbolj strogo določeno strukturo, s čimer poskuša izboljšati skupinsko produktivnost. Ima pa to tudi negativno posledico,

to je velika režija. Veliko časa in truda se namreč vloži v sam proces ocenjevanja. Odsotnost interakcije in popolna anonimnost med člani tekom ocenjevanja preprečujeta potencialne nezaželene učinke skupinskih metod. Se pa z razkritjem povprečja ocen vseeno poraja vprašanje, če je končna ocena dejansko rezultat kolektivnega znanja ali posledica medsebojnega vpliva. Možno je namreč, da člani popravijo svojo začetno oceno tako, da se približajo ostalim [4, 8].

Wideband Delphi v primerjavi z *Delphijem* dopušča medsebojno interakcijo med ocenjevanjem in zmanjša zapletenost ter režijo metode. Z uvedbo interakcije v proces deluje ravno nasprotno kot svoj predhodnik, katerega cilj je zmanjšati vpliv posameznika v skupini na ostale z omejevanjem sodelovanja. *Wideband Delphi* računa na to, da bo ravno to sodelovanje in deljenje znanja privedlo do boljšega razumevanja zahteve, katera se ocenjuje. To omogoča boljšo definicijo in razdelitev zahteve na manjše naloge ter s tem boljšo oceno. Spregledane naloge so drugače lahko eden od razlogov za manj točno oceno [4].

Takoj vidimo, da *Planning poker* v primerjavi z *Wideband Delphijem* vpelje v svoj proces še več medsebojne interakcije, s čimer se izgubi anonimnost med člani. Hkrati se s tem lahko zmanjša kompleksnost strukture, kar omogoča še manjšo režijo. Z več interakcije se med ocenjevalci deli več znanja in izkušenj, kar lahko zelo pripomore k večji točnosti ocen. Kaže se, da to še posebej velja pri zahtevah, podobnih tistim, s katerimi je skupina že imela opravka [4]. Vseeno pa se zaradi podobnosti omenjenih strukturiranih metod poraja isto vprašanje okoli razlogov, ki so privedli do končne ocene. Z več razprave med člani je namreč pri *Planning pokru* tudi več možnosti za neželene posledice skupinskega dela, kot so polarizacija in prilagajanje posameznega člana skupini.

TEG je po značilnostih zelo podobna *Planning pokru*. Obe metodi v proces vključujeta medsebojno interakcijo ocenjevalcev, čeprav TEG v nekoliko manjši meri. Anonimnosti tudi tukaj ni. Z manj zapleteno strukturo pa TEG omogoča manjšo režijo, kar je tudi cilj metode.

3.2.6 Zmožnost metod, da preprečijo negativne učinke skupinskega dela

Vsaka metoda predstavlja kompromis med stopnjo vključenih ukrepov proti negativnim učinkom skupinskega dela in stopnjo zmožnosti izkoriščanja pozitivnih. Furulund [4] je tako v svoji raziskavi razčlenil možnost vsake skupinske metode, da prepreči oziroma omeji nezaželene stranske učinke skupinskega dela (Tabela 3.2).

Metoda	Problemi s koordinacijo	Družbeni in politični konflikti	Skupinsko mišljenje in polarizacija
Nestrukturirane skupine	Ne	Ne	Ne
Delphi	Da	Da	Delno
Wideband delphi	Da	Delno	Delno
Planning poker	Da	Delno	Delno
Team estimation game	Delno	Delno	Delno

Tabela 3.2: Primerjava različnih metod glede na zmožnost preprečevanja neželenih stranskih učinkov skupinskih metod.

Nestrukturirane skupine ne vključujejo v svoj proces nobene anonimnosti, tako da ne izvajajo nobenih ukrepov za preprečevanje nezaželenih stranskih učinkov [4].

Omenili smo že, da ima anonimnost velik vpliv na potek in značilnosti skupinskih metod. Naslednje metode so jo v svoj proces tudi vključile, kar pomeni, da so morale definirati določeno strukturo.

Delphi je dober primer tega, saj veliko da na anonimnost. Neodvisnost članov skupine namreč igra pomembno vlogo pri procesu. Omeji se zmožnost prilagajanja ocen skupini in prepreči politični ter družbeni vpliv posameznika v skupini. S končno oceno, kot povprečjem posameznih, pa se nekoliko odpre

možnost za polarizacijo.

Wideband Delphi in *Planning poker* predstavljata kombinacijo med zaščito neodvisnosti posameznega člana in stopnjo dovoljene medsebojne komunikacije [4]. Obe metodi članom skupine omogočata neodvisno ocenjevanje. Pri prvi se to izvede anonimno po razlagi in razpravi o aktualni zahtevi. *Planning poker* pa ga integrira tako, da vsi člani istočasno pokažejo svojo oceno, čemur nato lahko sledi razprava. Na ta način se poskuša izkoristiti potencial pozitivnega učinka medsebojne komunikacije v takšni meri, da bi bile posledice možnih nezaželenih učinkov zanemarljive [4].

Pri TEG lahko ocenjevalci med igro razpravljajo o zapletenosti zahtev in razporeditvi kartic v stolpce ter o razlogih za postavitev na izbrano mesto. To nekoliko odpira možnost za nastanek nezaželenih učinkov skupinskih metod, kot so družbeni in politični vpliv posameznika na ostale, prilagajanje skupini ter polarizacija. Se pa z lažjo strukturo izgubi potreba po tolikšni koordinaciji, kot je prisotna pri *Planning pokru* in *Wideband Delphiju*.

3.2.7 Zaključek

Opisane skupinske metode so vse zasnovane na podlagi mnenja, da je uporaba skupine za ocenjevanje programske opreme dobra ideja [4]. Poskušajo najti recept, ki bo najbolje izkoristil pozitivne učinke skupinskega dela in zmanjšal vpliv negativnih. Vsaka metoda poskrbi po svoje, da se znanje in izkušnje v skupini delijo. Posledično ocenjevalci zahtevo bolje analizirajo, kar vodi do bolj točne ocene in to je tisto, kar nas tudi najbolj zanima.

Na tem področju je bilo izvedenih malo raziskav, a jasno je, da lahko s pravilno uporabo te metode precej prispevajo k ocenjevanju programske opreme in sami točnosti ocen. Furulund [4] meni, da bi z empiričnimi raziskavami lahko določili kdaj, kako in kje je najbolje uporabiti katero metodo. Z analizo *Planning pokra* in TEG v tej diplomski nalogi upamo, da bomo prispevali tudi k temu cilju.

3.3 Raziskovalna vprašanja

Okoli metod, ki za ocenjevanje uporabljajo mnenje več strokovnjakov in njihove učinkovitosti, se porajajo številna vprašanja. Opisali smo že nekaj dobrih in slabih značilnosti takšnega ocenjevanja. Tako ni presenetljivo, da je mnenje strokovnjakov na tem področju razdvojeno.

Za primer vzemimo optimizem. Raziskave v splošnem kažejo, da so ocene zahtev pri razvoju programske opreme preveč optimistične. Uporaba skupinskih metod ocenjevanja naj bi to popravila. Res obstajajo raziskave, ki kažejo na prevladovanje pozitivnih učinkov skupinskega ocenjevanja. Začetni optimizem ocen naj bi se zmanjšal, zraven pa naj bi se povečala še točnost. Ampak na drugi strani se najdejo tudi raziskave, ki kažejo ravno nasprotno in opozarjajo na nevarnosti takšnih skupinskih procesov. Pravijo, da takšne metode lahko celo povečajo začetni optimizem posameznika in s tem zmanjšajo točnost ocene.

Na podlagi naše študije bomo preverili to nasprotovanje pri izbranih metodah, to sta *Planning poker* in TEG, ter poskusili odgovoriti še na nekatera druga vprašanja, ki se vežejo na našo temo.

3.3.1 Kako *Planning poker* vpliva na spremembo začetnih ocen posameznikov?

Metoda ima s svojo zasnovo velik potencial, da izkoristi prednosti skupinskega dela. S tem, ko so dejanski razvijalci projekta tudi ocenjevalci, naj bi se povečala njihova zaveza in motivacija pri procesu. Ker se ocene razkrijejo istočasno, se zmanjša vpliv posameznika na skupino in zagotovi, da vsak član poda svoj glas. Z razpravo o podanih ocenah, predvsem o najvišji in najnižji, se vnaprej najde več potencialnih problemov pri kasnejši implementaciji zahteve [8].

Študija izvedena s strani Mahniča in Hovelje [8] kaže, da uporaba skupinskega ocenjevanja, kot je *Planning poker*, pri študentih poveča optimizem ocen in celo zmanjša njihovo točnost. Po drugi strani sta v sklopu iste študije

prišla do zaključka, da se to spremeni, če so ocenjevalci bolj izkušeni. Analiza ocen strokovnjakov namreč kaže, da se nagnjenost k optimizmu oziroma pesimizmu s *Planning pokrom* zmanjša in točnost ocen poveča. To pa je tudi v skladu z rezultati raziskave Moløkken-Østvolda [9], ki ravno tako kaže, da uporaba te metode res zmanjša optimizem in poveča točnost ocen.

Z analizo ocen, pridobljenih v naši študiji, bomo poskusili prispevati k eni od zgornjih trditev. Preverili bomo torej vpliv *Planning pokra* na spremembo optimizma začetnih ocen posameznikov in če so končne ocene dobljene po tej skupinski metodi bolj točne kot začetne. Vprašanji, ki smo si ju tu zastavili, sta naslednji:

- `Ali so ocene, dobljene po metodi` ***`Planning poker`***`, manj optimistične kot statistična kombinacija začetnih ocen posameznikov?`
- `Ali so ocene, dobljene po metodi` ***`Planning poker`***`, bolj točne kot statistična kombinacija začetnih ocen posameznikov?`

Zaradi podobnosti naše študije s prvim delom študije [8], ki analizira ocene študentov, vseeno pričakujemo, da bodo tudi naši rezultati bližje tistim. V drugem delu so bile namreč analizirane ocene strokovnjakov, Moløkken-Østvoldova raziskava [9] pa je bila tudi postavljena malo drugače. V njej je sodelovala le ena razvojna skupina, tako da je bil vzorec za analizo veliko manjši kot pri nas. Omeniti je potrebno tudi, da je bila ta skupina sestavljena iz strokovnjakov, ki so delali na resničnem projektu. Mogoče najpomembnejša razlika pa je ta, da so ocenjevali naloge in ne celotnih uporabniških zgodb.

3.3.2 Kako se *Team estimation game* primerja s *Planning pokrom*?

V literaturi se bolj pogosto pojavljajo študije povezane s *Planning pokrom* kot s TEG. Mislimo, da je tudi slednjo metodo vredno bolje spoznati. S

svojim načinom igre lahko naredi ocenjevanje bolj sproščeno in morda celo zabavno. Poleg tega z relativnim ocenjevanjem zahtev cilja na hitrejšo izvedbo in manjšo režijo. Vprašanje pa je, kako točne so tako dobljene ocene.

Kot del naše diplomske naloge bomo torej analizirali ocene, ki smo jih dobili z metodo TEG. Najprej bomo preverili nagnjenost metode k optimizmu in pokazali, če je ta bolj ali manj optimistična kot *Planning poker*. Primerjali bomo tudi točnost ocen, dobljenih po obeh metodah. Zanima nas, katera je boljša. Druge takšne študije na področju nismo našli. Naši vprašanji sta tako sledeči:

- Ali so ocene, dobljene po metodi *Team estimation game*, manj optimistične kot ocene, dobljene po metodi *Planning poker*?
- Ali so ocene, dobljene po metodi *Team estimation game*, bolj točne kot ocene, dobljene po metodi *Planning poker*?

Rezultat analize upamo, da bo pokazal, če je *Planning poker* upravičeno bolj razširjen, ali pa si tudi TEG zasluži več pozornosti. Tako bomo poskusili prispevati še k študijam in razpoznavnosti te metode.

Poglavje 4

Potek študije

Študija je potekala v sklopu predmeta Tehnologija programske opreme (TPO), na Fakulteti za računalništvo in informatiko, ki je del univerze v Ljubljani. Odvijal se je v zadnjem semestru tretjega in hkrati zadnjega letnika dodiplomskega študija.

Na tej točki izobraževanja so študenti že osvojili tradicionalne metode razvoja programske opreme, osnove podatkovnih baz in informacijskih sistemov ter vodenja podobnih projektov, kar je bilo ključnega pomena, da smo študijo sploh lahko izpeljali [10].

Vsebina predmeta se je delila na dva dela, predavanja in vaje. Na začetku semestra so se študenti spoznali s teorijo, potrebno za izvedbo študije, kjer je bil poudarek na agilnem razvoju programske opreme, posebno s Scrumom, uporabniških zgodbah, *Planning pokru* in TEG. Ko so to znanje osvojili, so tekom preostanka semestra tudi izpeljali določen projekt.

Projekt je bil zasnovan s strani nosilca predmeta. Omislil si je razvoj študijskega infromacijskega sistema, ker je tudi sam že imel izkušnje na tem področju. Izdelek je moral biti narejen v obliki spletne aplikacije. Celotna zasnova in zahteve projekta so bile podane zelo detajlno, da bi ta izpadel čimbolj resnično. V aplikacijo je bilo med drugim potrebno implementirati vpis na fakulteto ter evidenco vseh vpisanih po vseh letnikih, sistem za izvajanje izpitov ter hranjenje njihovih rezultatov in sistem za hranjenje podatkov

o vseh predmetih, profesorjih in podobno.

Študenti so se razdelili v dvajset razvojnih skupin. Vsaka je v študiji delala na svojem projektu. Da bi dobili čim večji vzorec za statistično analizo ocen, kar je tudi cilj te diplomske naloge, so vse skupine dobile enak projekt. Tako so vsi razvijali že omenjeni študijski informacijski sistem.

Proste roke so imeli tudi, ko je prišlo do tehnologije, s katero se je projekt razvijal. Vsaka skupina si je namreč lahko izbrala poljubno tehnologijo. Študentom je bilo tako omogočeno, da so razvijali v sebi najbolj ustreznem in poznanem okolju.

Proces razvoja je temeljil na principih metode Scrum. Določene so bile vloge in izvajali so se vsi sestanki, ki jih tehnika predvideva. Čas, ki je bil na razpolago tekom semestra, je bil razdeljen na štiri sprinte. Imeli smo uvodni sprint 0 (Poglavje 4.1) in tri klasične sprinte enake dolžine, v katerih se je dejansko razvijal izdelek. Nekatere stvari so bile prilagojene potrebam in omejitvam študentov ter izvajanju študije v sklopu študija. Natančneje je potek opisan v nadaljevanju tega poglavja.

4.1 Sprint 0

Sprint 0 se pri razvoju s Scrumom pogosto pojavlja, čeprav v pravilih slednjega ni točno določen. Mnenja o njegovi uporabi niso enotna, a nekaj napotkov je vredno upoštevati.

Eden od teh je zasnova seznama zahtev. V tem začetnem sprintu se lahko določi in zapiše vsaj tiste zgodbe, ki bi služile kot temelj razvoju izdelka.

Postavilo naj bi se osnove za nadaljnji razvoj projekta, tako da bi naslednji sprinti res lahko tekli po načelih Scruma. S tem še posebej mislimo načelo, ki pravi, da mora vsaka iteracija proizvesti delujočo nadgradnjo izdelka. To bi bilo lahko v sprintu 0 izvedljivo z izbiro in razvojem samo nekaj kritičnih zgodb, katerih realizacija bi pomenila tudi postavitev osnovnega ogrodja izdelka [11].

Pri profesionalni razvojni skupini, ki se na svoje delo dobro spozna in

je že razvijala po Scrumu, je to seveda mogoče. V našem primeru pa ni bilo. Skupine so bile namreč sestavljene iz študentov, ki so bili šele pri koncu svojega študija. Na naš sprint 0 je tako imelo največji vpliv dejstvo, da veliko študentov ni še nikoli razvijalo programske opreme v profesionalnem okolju, po točno določeni metodologiji.

Sprint 0 je zato pri nas temeljil na predavanjih. Na njih smo se spoznali s teorijo o agilni metodologiji, Scrumu, *Planning pokru*, TEG in vsem ostalem, kar smo potrebovali za nadaljnje delo. Z obema metodama ocenjevanja smo se spoznali tudi praktično, saj je bilo proti koncu sprinta izvedeno poskusno ocenjevanje, da bi naslednje iteracije tekle bolj gladko. Na koncu tega sprinta pa smo tudi ocenili začetni seznam zahtev.

4.2 Vloge

Profesor in nosilec predmeta TPO je prevzel vlogo skrbnika izdelka. Od slednjega se med drugim zahteva, da dobro pozna področje dela. S svojim znanjem in izkušnjami na področju razvoja programske opreme je bil za to nalogo več kot primerna izbira. Razvojnim skupinam je bil vedno na voljo, s čimer je uspešno opravljal še eno pomembno nalogo svoje vloge.

Skupine so bile sestavljene iz študentov. Pri študiji je sodelovalo dvajset skupin, ki so bile večinoma sestavljene iz štirih članov. Tekom študija so študentje že dobro spoznali principe programiranja in z malo uvoda v izbrane metode (Poglavje 4.1) so bili pripravljeni na razvoj podanega izdelka.

Delo skrbnika procesa je v študiji opravljalo pedagoško osebje. Povedali smo že, da je glavna naloga skrbnika procesa ta, da razvojno skupino usmerja in skrbi, da celoten postopek sledi načelom Scruma.

Ker je ocenjevanje potekalo po dveh metodah, so se tudi skupine razdelile na dva dela. Trinajst jih je ocenjevalo s *Planning pokrom*, ostalih 7 pa je uporabljalo TEG.

4.3 AC Scrum

Pri študiji smo za lažje izvajanje in sledenje postopku razvoja uporabljali posebno programsko orodje, imenovano AC Scrum. Razvito je bilo prav na naši fakulteti z namenom, da omogoča digitalno vodenje celotnega projekta.

Profesor in njegovi asistenti so vnaprej pripravili začetno stanje, kar je med drugim vključevalo prvotni seznam zahtev, ki se je med sprinti lahko še spreminjal. Vsak študent je imel svoj uporabniški račun, preko katerega se je prijavil v sistem. Vsaka razvojna skupina je dobila svoj digitalni projekt. Orodje je bilo dostopno preko spleta, tako da se ga je dalo uporabljati od kjerkoli.

Uporabljalo se ga je za planiranje in dokumentiranje sprintov. Sem je spadala določitev posameznega sprinta, izbira zgodb za vsakega, ocenjevanje z metodama *Planning poker* in TEG, integriran je bil digitalni dnevni Scrum sestanek in še več. Da so študenti lahko sami začeli ocenjevanje, so morali izbrati enega člana iz svoje skupine, ki je poleg razvijalca v sistemu AC Scrum prevzel še vlogo skrbnika procesa, s čimer je dobil potrebne pravice. Poudariti je potrebno, da je bila to edina dodatna naloga izbranega člana, pravo delo skrbnika procesa je opravljalo pedagoško osebje.

Med razvojem so vse skupine preko tega orodja tudi dokumentirale svoj napredek dela skupaj s porabljenim časom, kar je omogočalo sprotni in kasnejši pregled sprintov ter celotnega projekta. Nam pa je to omogočilo analizo podatkov, vključno z ocenami uporabniških zgodb, katere so nam še posebej zanimive.

4.4 Seznam zahtev

Profesor je seznam zahtev sestavil še preden smo končali s sprintom 0 in nam ga na koncu le-tega tudi predstavil. To je bila prva verzija seznama, ki je vsebovala 30 uporabniških zgodb. Vsem je tudi že določil svojo prioriteto. Imeli smo 14 *must have* zgodb, 9 *should have*, 4 *could have* in 3 *won't have this time*. Pomembnost pada od *must have* zgodb, ki morajo biti obvezno

implementirane v rešitvi, do *won't have this time*, katere služijo bolj kot vpogled v kar je zaželjeno za razvoj v poznejših sprintih.

Na sestanku za planiranje tretjega sprinta smo dobili drugo, posodobljeno izdajo seznama. Ta je vsebovala 20 novih uporabniških zgodb, spet urejenih po prioriteti s strani skrbnika izdelka. Dodatno so se nekaterim zgodbam iz prve izdaje spremenile prioritete.

Na koncu je torej projekt imel na celotnem seznamu 50 zahtev. Od tega je bilo 31 *must have*, 10 *should have* in 9 *could have* zgodb. Da bi se razvilo vse zahteve pri nas ni bilo izvedljivo, ker so bile razvojne skupine sestavljene iz študentov, ki jim je študija predstavljala le del študijskega programa in so poleg tega imeli še druge obveznosti. Za uspešno opravljen predmet je bilo potrebno realizirati vsaj 20 *must have* zgodb. *Should have* in *could have* so bile na razpolago za tiste, ki so želeli boljšo oceno.

4.5 Potek sprintov

Študija se je odvijala čez štiri sprinte, pri čemer se je izdelek razvijal v zadnjih treh. Njen potek je podrobno prikazan tudi v tabeli 4.1, ki temelji na podobni študiji [8].

Prvi, sprint 0, je trajal tri tedne in je bil bolj namenjen uvajanju študentov v teorijo (Tabela 4.1; Korak A) kot samemu razvoju. Po predstavitvi uporabniških zgodb (Tabela 4.1; Korak B) smo v sklopu tega sprinta ocenili vse zgodbe iz seznama zahtev (Tabela 4.1; Korak C) z namenom, da bi lažje definirali ostale sprinte in naredili plan izdaje.

Normalno se s pomočjo ocen vseh zgodb oceni tudi približen datum, ko naj bi bil izdelek zaključen in v celoti pripravljen na izdajo. Skupaj s hitrostjo razvoja (Tabela 4.1; Korak D), kar razvojna skupina oceni med planiranjem sprinta, je takšno časovno oceno precej lahko izračunati. Če namreč vemo koliko točk lahko skupina realizira v enem sprintu, vemo tudi koliko sprintov bo potrebnih za vse zgodbe. Tudi v naši študiji smo najprej ocenili vse zgodbe in skupine so določile svojo hitrost. A imeli smo omejen čas, tako da

A	Sprint 0	Spoznavanje s Scrumom in uporabniškimi zgodbami
B		Predstavljen začetni seznam zahtev
C		Ocenjevanje uporabniških zgodb s *Planning pokrom* in TEG
C.1		Skrbnik izdelka predstavi zahteve uporabniških zgodb
C.2		Razprava razvojne skupine o delu, potrebnem za realizacijo uporabniške zgodbe
C.3.1		Pri *Planning pokru* shranjene posamezne začetne ocene
C.3.2		Pri TEG postavitev uporabniške zgodbe na stolpec
C.4		Shranjena končna ocena razvojne skupine
D		Ocenjena hitrost vsake razvojne skupine
E		Definiran plan izdaje vsake razvojne skupine
F	Sprinti 1, 2 in 3	Sestanek za planiranje sprinta
F.1		Nove (in preostale) zgodbe ocenjene
F.2		Določena vsebina sprinta
F.3		Uporabniške zgodbe razdeljene na naloge
F.4		Naloge dodeljene članom razvojnih skupin
F.5		Naloge ocenjene
G		Dnevni Scrum sestanki
G.1		Zabeležen čas, porabljen na vsaki nalogi
G.2		Zabeležena količina dela, potrebna za dokončanje vsake naloge
H		Sestanek za pregled sprinta
H.1		Skrbnik izdelka oceni implementacijo uporabniških zgodb
H.2		Za vsako nedokončano/zavrnjeno zgodbo ustvarjena nova zgodba
I	Konec študije	Izračunan dejanski čas na osnovi podatkov zabeležnih na dnevnih Scrum sestankih

Tabela 4.1: Potek študije.

je bilo število sprintov in realiziranih zgodb v končni izdaji prilagojeno temu časovnemu okvirju.

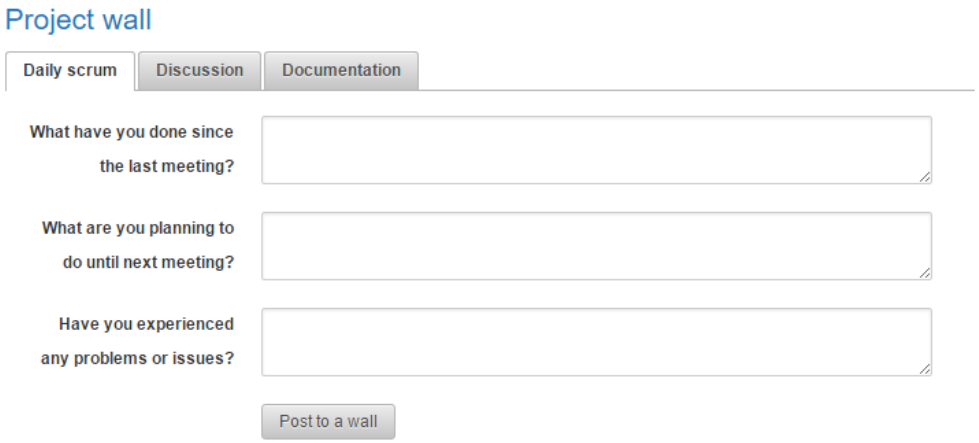
Čas, ki nam je po uvodnem sprintu ostal do izteka semestra, je bil tako razdeljen na troje in dobili smo dolžino ostalih iteracij. Vsaka je trajala štiri tedne, kar je dlje kot predlaga Cohn [3], a je glede na naše okoliščine bolj ustrezno. Planiralo se je, da bo po zadnjem sprintu prva izdaja izdelka pripravljena (Tabela 4.1; Korak E).

Na začetku vsake iteracije smo imeli sestanek za planiranje sprinta (Tabela 4.1; Korak F). Razvojne skupine so se sestale z skrbnikom izdelka. Ta je predhodno že vpisal oziroma posodobil seznam zahtev v AC Scrumu in novosti na sestanku vsem skupaj predstavil. Zatem se je izvedlo ocenjevanje uporabniških zgodb, ki je podrobneje opisano v nadaljevanju (Poglavje 4.6). Ocenilo se je vse nove zgodbe in tiste, ki so skupini ostale še iz prejšnjega sprinta (Tabela 4.1; Korak F.1).

Razvojne skupine so si nato v aplikaciji sestavile seznam uporabniških zgodb, katere so nameravale realizirati v tekočem sprintu (Tabela 4.1; Korak F.2). Te so študenti sami razbili še na podrobnejše, manjše naloge (Tabela 4.1; Korak F.3). AC Scrum je omogočal, da si je vsak posameznik lahko izbral svojo nalogo (Tabela 4.1; Korak F.4), katero se je odločil realizirati. Tako se je v skupini točno vedelo, na katerih zgodbah kdo dela, kar je zmanjšalo število konfliktov med razvojem.

Število in zahtevnost zgodb v seznamu z vsebino sprinta sta bila seveda odvisna od ocene hitrosti vsake skupine, tj. koliko dela so ocenili, da bodo lahko opravili v iteraciji. Med sprintom je bilo razvojni skupini ta seznam dovoljeno tudi nekoliko spreminjati. V primeru, da bi bile vse izbrane zgodbe predčasno zaključene, so imeli možnost dodati nove in tako opraviti več dela, kot so sprva planirali.

Redno so se izvajali dnevni Scrum sestanki (Tabela 4.1; Korak G). Ker se je študija izvajala tekom študija in od študentov ni bilo mogoče pričakovati, da se bodo sestali vsak dan, kot je pri originalnih pravilih Scruma, so se pri nas izvajali dvakrat tedensko. Tudi izvedba sestanka je bila prilagojena tem



Slika 4.1: Beleženje dnevnih Scrum sestankov v AC Scrumu.

omejitvam. Od študentov namreč ni bilo zahtevano, da se fizično sestanejo, ampak so lahko od kjerkoli izpolnili določen obrazec preko AC Scruma, kjer so morali odgovoriti na tista tri vprašanja, značilna za dnevni Scrum sestanek (Poglavje 2.3.1). Tako so razvijalci poročali o svojem napredku in težavah pri delu ter o svojih nadaljnjih planih, kar je pripomoglo k boljši koordinaciji razvoja. Sproti so morali beležiti ure, porabljene na vsaki nalogi in čas, ki je predvidoma še ostal do konca njene realizacije (Tabela 4.1; Koraka G.1 in G.2). V orodju je bilo omogočeno tudi sprotno pisanje dokumentacije in razprava med člani skupine, kot je vidno na sliki 4.1.

Na koncu vsake iteracije sta sledila še dva sestanka. Prvi je bil namenjen pregledu sprinta (Tabela 4.1; Korak H). Razvojne skupine so skrbniku izdelka predstavile narejene uporabniške zgodbe. Ta si je njihovo delo temeljito ogledal in šel skozi vse sprejemne teste. Šele ko je bil res prepričan, da je bila zgodba realizirana tako, kot si je on zamislil, se je ta lahko označila kot zaključena (Tabela 4.1; Korak H.1). V nasprotnem primeru se je definirala nova uporabniška zgodba, katera je bila sestavljena iz zahtev, ki v originalni zgodbi še niso bile uspešno narejene (Tabela 4.1; Korak H.2). Novo zgodbo se je lahko ponovno izbralo za delo v naslednjem sprintu.

Zadnji je bil retrospektivni sestanek. Na tega so se za hitrejšo izvedbo,

zaradi omejenega časa in velikega števila skupin, morali vsi vnaprej pripraviti. Študenti so v skupinah analizirali svoje delo in splošen potek sprinta, primerjali dejansko hitrost razvoja s planirano. Definirali so težave, ki so se pojavile in skupaj našli potencialne rešitve oziroma možnosti izboljšanja za naslednji sprint. Na samem sestanku je sledila še razprava o tem s profesorjem, ki je s svojimi nasveti razvojnim skupinam pomagal še dodatno izboljšati celoten proces.

AC Scrum je shranjeval podatke, kot je predvidena hitrost skupine, velikost planiranega seznama z vsebino sprinta in ocene zgodb. V podatkovno bazo se je vpisoval tudi čas, porabljen na posamezni nalogi. Tako je bilo možno spremljati in analizirati napredek vseh skupin (Tabela 4.1; Korak I). V ta namen je naše programsko orodje imelo vgrajen še graf s prikazom poteka dela med sprinti (angl. *Burn down chart*), prikaz realiziranih zgodb skozi celoten razvoj in podobno.

4.6 Ocenjevanje zahtev

Zahteve so bile, kot je to značilno za Scrum, predstavljene v obliki uporabniških zgodb. Ocenjevalo se jih je po dveh skupinskih metodah, to sta *Planning poker* in TEG.

Enota ocenjevanja je bila tudi pri nas točka. Omenili smo že, da je to relativna ocena. Pomembno je torej le, ali se nam zdi ena zgodba večja, manjša ali enakovredna ostalim. Zgodba, ki je ocenjena z dvema točkama, bi morala biti dvakrat večja kot tista, ocenjena z eno [3]. Na tak način naj bi bilo ocenjevanje lažje. Pri agilnem razvoju se med ocenjevanjem pogosto še ne ve vseh podrobnosti zahtev, a vseeno jih je potrebno oceniti. Relativne točke so tudi v takih primerih učinkovite, ker lahko zgodbo primerjamo z drugimi in jo približno ocenimo.

Točke lahko pretvorimo v idealne dni, s čimer lažje pridemo do časovne ocene razvoja. Idealen dan je namreč čas, ki ga dejansko porabimo na razvoju neke funkcionalnosti ali zgodbe. Ne upoštevamo časa porabljenega za

koordinacijo dela, odmorov razvijalca in podobnega. Mi smo kot idealen dan vzeli 6 ur. Glavni razlog za to je bila večja skladnost z ostalimi raziskavami.

V naši študiji se je ocenjevanje uporabniških zgodb izvedlo s pomočjo AC Scruma. Ta je namreč imel vgrajeni orodji za digitalno ocenjevanje z obema metodama.

Skrbnik procesa je pri *Planning pokru* izbral zgodbo, ki se bo ocenjevala in sledila je razprava razvojne skupine o količini potrebnega dela in časa za realizacijo ter testiranje. Zatem je začel igro. Ker je bilo orodje v obliki spletne aplikacije, je lahko vsak član sodeloval pri procesu preko svojega računalnika. Na razpolago so bile vnaprej določene vrednosti (0.5, 1, 2, 3, 5, 8, 13 in 20), med katerimi so ocenjevalci izbrali eno, možno pa je bilo tudi vpisati vrednost po svoji meri. Rezultati aktualnega kroga glasovanja so se skupini pokazali šele, ko so vsi ocenjevalci oddali svojo oceno. Kljub digitalni izvedbi ocenjevanja se je torej ohranila neodvisnost z istočasnim razkritjem vseh ocen, kar je pomemben del *Planning pokra*. Če je bilo v ocenah preveliko odstopanje, je v skupini sledila razprava. Še posebej je bilo zaželjeno, da se pogovorita člana z najvišjo in najnižjo oceno. Temu je sledil nov krog. Ko so prišli do enotne ocene, se je ocenjevanje te zgodbe zaključilo in začelo z naslednjo.

Podobno je v AC Scrumu implementiran tudi TEG. Zgodbe za ocenjevanje so na razpolago preko spustnega seznama. Zraven je igralna površina, na katero se relativno razvršča igralne kartice. Vsaka poteza se takoj pokaže vsem članom, tako da lažje sledijo nastanku in razvoju stolpcev različnih zapletenosti [10].

Pri ocenjevanju smo se v primeru kakšnih nejasnosti glede zgodb študenti lahko obrnili na skrbnika izdelka in jih s svojimi vprašanji razčistili.

Poglavje 5

Analiza točnosti ocen

Naš prvi cilj je ugotoviti, kako *Planning poker* vpliva na spremembo začetnih ocen posameznikov. Pri tem nas zanima, če so končne ocene, dobljene po tej metodi, manj ali bolj optimistične. Poleg tega smo se vprašali še, če se z uporabo te metode natančnost začetnih ocen poveča ali zmanjša.

V drugem delu analize smo primerjali točnost ocen, dobljenih po obeh metodah, torej po *Planning pokru* in TEG. Zanimalo nas je, katera je bolj točna. Preverili smo še, katera metoda je bolj optimistična.

V ta namen smo izkoristili našo študijo, kjer so skupine študentov razvijale določen, enak projekt po metodologiji Scrum. Sodelovalo je 20 razvojnih skupin. Da bi za analizo podatkov dobili čimbolj homogen vzorec, smo takoj na začetku izločili 2 skupini, ki sta realizirali premalo uporabniških zgodb. Za namen naše analize smo izmed vseh izbrali 24 zgodb, katere je večina preostalih 18 razvojnih skupin uspešno končala. Pričakovano vseh 24 spada pod *must have* prioriteto, ki so bile pogoj za pozitivno oceno pri predmetu. Dodatnih zgodb se je namreč lotilo le majhno število skupin.

Preden so lahko začeli delati na razvoju danih zgodb, so te morali oceniti. Spomnimo se, da je 11 razvojnih skupin uporabljalo *Planning poker*, 7 pa TEG. Ocene uporabniških zgodb in čas, porabljen za njihovo realizacijo, so se preko AC Scruma shranjevale v podatkovno bazo. Iz baze smo izluščili za nas pomembne podatke in jih analizirali s pomočjo programa SPSS.

5.1 Statistika

Za izračun točnosti ocen, dobljenih po obeh metodah ocenjevanja, je bila uporabljena uravnotežena relativna napaka BRE (angl. *Balanced measure of relative error*). Prvi razlog za to izbiro je bil ta, da je bil BRE uporabljen v velikem številu že opravljenih študij [4, 8, 9], kar nam je olajšalo primerjavo naših rezultatov. Drugi razlog je pa ravno ta, da je BRE uravnotežena meritev, kar pomeni, da enakomerno upošteva podcenjene in precenjene ocene [8].

$$BRE = \frac{|\text{dejansko porabljeni čas - ocenjeni čas}|}{\min(\text{dejansko porabljeni čas, ocenjeni čas})} \tag{5.1}$$

Da bi ugotovili, ali se končne ocene, dobljene po *Planning pokru* in TEG, nagibajo k optimizmu ali pesimizmu, smo izračunali BREbias. Ta namreč meri tako velikost napake kot tudi njeno usmeritev.

$$BREbias = \frac{(\text{dejansko porabljeni čas - ocenjeni čas})}{\min(\text{dejansko porabljeni čas, ocenjeni čas})} \tag{5.2}$$

Naši podatki niso bili normalno porazdeljeni, kar je vidno v tabeli 5.1 iz asimetičnosti in sploščenosti, ki so v normalni porazdelitvi blizu ničle. Sploščenost še posebej kaže na nenormalno porazdelitev, saj njene visoke vrednosti kažejo na velika odstopanja v podatkih oziroma v našem primeru redke ekstremne vrednosti ocen.

Zaradi visoke podobnosti podatkov z raziskavo [8] je bilo to tudi pričakovano. Tako sta bila tudi tukaj uporabljena *Wilcoxon signed-rank* test in *Mann–Whitney–Wilcoxon* test. Oba sta neparametrična statistična testa, primerna za uporabo nad takimi podatki, ki niso normalno porazdeljeni. *Wilcoxon signed-rank* test je primeren za iskanje razlik med dvema povezanima vzorcema, s primerjanjem povprečnih vrednosti [17]. Uporabljen je bil torej pri prvih dveh raziskovalnih vprašanjih (Poglavje 5.2). *Mann–Whitney–Wilcoxon* test se je uporabil pri drugih dveh (Poglavje 5.3), saj je namenjen za primerjavo dveh neodvisnih vzorcev [8]. Za prikaz velikosti učinka rezultatov smo pri testih izračunali še Cliffovo delto (angl. *Cliff's delta*), ki ravno tako ne zahteva normalne porazdelitve podatkov. Za to smo uporabili zastonjsko programsko opremo *Cliff's Delta Calculator*, ki je predstavljena v članku [7].

	BRE statistične kombinacije	BRE *Planing poker*	BREbias statistične kombinacije	BREbias *Planing poker*	BRE TEG	BREbias TEG
Št. veljavnih zgodb	263	263	263	263	164	164
Št. neveljavnih zgodb	1	1	1	1	4	4
Povprečna vrednost	,9313	,9219	,2866	,1188	,7886	-,2484
Mediana	,5625	,5000	,2381	,1515	,4065	,0000
Std. odklon	1,19201	1,45721	1,48629	1,72120	1,35304	1,54736
Asimetričnost	4,095	6,210	-2,130	-4,016	5,901	-4,466
Std. napaka asimetričnosti	,150	,150	,150	,150	,190	,190
Sploščenost	27,856	58,445	18,269	39,194	48,481	33,523
Std. napaka sploščenosti	,299	,299	,299	,299	,377	,377
Razpon ocen	11,75	17,00	17,76	23,33	13,46	16,24

Tabela 5.1: Statistika študije.

V tabeli 5.1 vidimo, da imamo pri *Planning pokru* eno neveljavno uporabniško zgodbo, pri TEG pa še štiri. Povedali smo že, da smo za analizo izbrali zgodbe, ki jih je razvila večina razvojnih skupin. Te neveljavne so torej tiste izjeme, ki s strani kakšne skupine niso bile dokončane. Na končni rezultat niso bistveno vplivale, saj jih je očitno zelo malo.

5.2 Kako *Planning poker* vpliva na spremembo začetnih ocen posameznikov?

Da bi odgovorili na zastavljeno vprašanje, smo izvedli analizo in primerjavo začetnih ocen posameznikov s končnimi ocenami, dobljenimi po tej metodi.

Že v referenčnih raziskavah [8, 9] je bilo upoštevano Jørgensenovo [6] priporočilo, ki pravi, da je aritmetično povprečje pogosto učinkovit način združevanja ocen. Tako smo za prvi del primerjave tudi mi preprosto vzeli povprečje začetnih ocen vseh članov v skupini, torej povprečje ocen, ki so jih ocenjevalci podali v prvem krogu igre.

Končna ocena je potrebovala malce razmisleka, ker so se uporabniške

zgodbe pri študiji večkrat ocenjevale. Vsaka je bila prvič ocenjena v sprintu, v katerem je bila predstavljena. Če se je zgodba v istem sprintu v celoti razvila, smo tako že imeli določeno končno oceno. V nasprotnem primeru se je ocenjevanje zgodbe ponovilo.

Če se razvoj ocenjene zgodbe v tekočem sprintu ni niti začel, se jo je v naslednjem sprintu ponovno ocenilo in tako smo dobili novo končno oceno.

Malo bolj se zaplete v primerih, ko se je razvoj zgodbe tekom aktualnega sprinta začel, a ta na koncu sprinta še ni bila v celoti končana. Takrat se je še enkrat ocenilo samo preostalo delo, potrebno za dokončno realiziranje. Za našo analizo smo zato vzeli tisto prejšnjo oceno, ki je bila sprejeta, preden se je začel razvoj zgodbe.

V vsakem primeru smo torej kot končno oceno, dobljeno po *Planning pokru*, vzeli zadnjo oceno, ki se je nanašala na celotno zgodbo.

5.2.1 Ali so ocene, dobljene po metodi *Planning poker*, manj optimistične kot statistična kombinacija začetnih ocen posameznikov?

Da bi lahko odgovorili na zastavljeno vprašanje, smo za primerjavo uporabili *Wilcoxon signed-rank* test, dodatno smo pa izračunali še Cliffovo delto.

Primerjali smo BREbias statistične kombinacije začetnih ocen proti hipotezi nič. Slednja pravi, da je BREbias enak nič oziroma da v našem primeru ni prisotnega optimizma. Na enak način smo z dano hipotezo primerjali še BREbias *Planning pokra*.

Iz tabele 5.1 lahko razberemo povprečno vrednost BREbias, ki je pri statistični kombinaciji enaka 0,2866, pri *Planning pokru* pa 0,1188. Očitno je, da v obeh primerih govorimo o optimizmu. Rezultati *Wilcoxon signed-rank* testa so predstavljeni v prvem in drugem delu tabele 5.2. Pri obeh testih nam vrednost P, ki je manjša od 0,05 (0,000 in 0,001) pove, da je ta optimizem tudi statistično značilen in ga je zato potebno upoštevati. Hipotezo nič torej zavržemo. Večjo statistično pomembnost pa kaže optimizem statistične

	N	Povprečen rang	Vsota rangov	Z	P	Cliffova delta
0 - BREbias statistične kombinacije						
Negativni rangi	163	140,66	22927,00	-5,048	0,000	-0,2548
Pozitivni rangi	96	111,91	10743,00			
Vezani rangi	4					
Skupaj	263					
0 - BREbias *Planning poker*						
Negativni rangi	149	133,00	19816,50	-3,477	0,001	-0,1863
Pozitivni rangi	102	115,78	11809,50			
Vezani rangi	12					
Skupaj	263					

Tabela 5.2: Analiza začetnega optimizma.

kombinacije začetnih ocen s srednje velikim učinkom, saj je Cliffova delta = -0,2548. V primerjavi z njo smo pri *Planning pokru* dobili manjši učinek (Cliffova delta = -0,1863).

Optimizem začetnih ocen posameznikov je bil pričakovan in je v skladu z referenčno raziskavo [9] in analizo ocen študentov pri [8]. Podobno je tudi pri ocenah strokovnjakov v slednji, le da se tiste nagibajo k pesimizmu.

V naslednjem delu analize smo izvedli primerjavo med obema BREbiasoma. Preverili smo torej spremembo optimizma statistične kombinacije začetnih ocen z uporabo metode *Planning poker*. Rezultati so v prvem delu tabele 5.3. Kaže, da je uporaba te skupinske metode ocenjevanja dejansko zmanjšala začetni optimizem ocenjevalcev. Vrednost P = 0,002 nam tudi tukaj pove, da je ta ugotovitev statistično značilna. Temu pa ne moremo pripisati statistične pomembnosti, ker je Cliffova delta = -0,0644.

Prišli smo torej do zaključka, ki ni v skladu z našim pričakovanjem. Kljub podobnosti študije s prvim delom [8], kjer so analizirane ocene študentov, naši rezultati kažejo nasprotno. Čeprav je učinek zanemarljiv, metoda *Planning poker* preko skupinskega dela zmanjša začetni optimizem posameznikov. Je pa to v skladu z raziskavo [9] in drugim delom [8], ki se nanaša na ocene strokovnjakov.

	N	Povprečen rang	Vsota rangov	Z	P	Cliffova delta
BREbias *Planning poker* - BREbias statistične kombinacije						
Negativni rangi	99	100,26	9925,50	-3,158	0,002	-0,0644
Pozitivni rangi	77	73,38	5650,50			
Vezani rangi	87					
Skupaj	263					
BRE *Planning poker* - BRE statistične kombinacije						
Negativni rangi	91	91,55	8331,00	-0,940	0,347	-0,0428
Pozitivni rangi	84	84,51	7069,00			
Vezani rangi	88					
Skupaj	263					

Tabela 5.3: Primerjava *Planning pokra* s statistično kombinacijo začetnih ocen posameznikov.

5.2.2 Ali so ocene, dobljene po metodi *Planning poker*, bolj točne kot statistična kombinacija začetnih ocen posameznikov?

Da bi ugotovili kakšen vpliv ima *Planning poker* na točnost ocen, je bil izveden še en *Wilcoxon signed-rank* test. Primerjali smo BRE končnih ocen, dobljenih kot rezultat metode in BRE statistične kombinacije začetnih ocen študentov. Rezultati testa so prikazani v drugem delu tabele 5.3.

Če pogledamo obe vrednosti BRE iz tabele 5.1, vidimo, da je pri *Planning pokru* malo manjša velikost napake (povprečna vrednost = 0,9219 in mediana = 0,5 v primerjavi s povrečno vrednostjo = 0,9313 in mediano = 0,5625). Ampak test, ki smo ga tu izvedli, kaže, da temu ne moremo pripisati statistične značilnosti. P je namreč enak 0,347 in to je precej čez mejo zavrnitve hipoteze, da sta vzorca enaka. Tudi Cliffova delta s svojo majhno vrednostjo (-0,0428) potrjuje, da je učinek zanemarljiv.

5.3 Kako se *Team estimation game* primerja s *Planning pokrom*?

TEG je zasnovan tako, da se ocenjevanje izvede hitreje kot pri metodah, kakor je *Planning poker*. Kljub temu ta metoda cilja na ocene, konkurenčne

ali celo boljše od tistih, dobljenih po podobnih metodah. Zato smo TEG in *Planning poker* primerjali med seboj.

5.3.1 Ali so ocene, dobljene po metodi *Team estimation game*, manj optimistične kot ocene, dobljene po metodi *Planning poker*?

Pri analizi *Planning pokra* smo ugotovili, da so bile končne ocene optimistične. Vprašali smo se, kako je z optimizmom pri TEG. Povprečna vrednost BREbiasa = -0,2484 kaže celo na možnost pesimizma ocen.

Rezultati testa, ki so prikazani v tabeli 5.4 pa pravijo, da pri ocenjevanju s TEG ni prišlo do statistično pomembne značilnosti med BREbiasom in testno vrednostjo 0, ker je P = 0,127. To potrjuje tudi Cliffova delta, ki je enaka 0,0854.

	N	Povprečen rang	Vsota rangov	Z	P	Cliffova delta
0 - BREbias TEG						
Negativni rangi	68	70,32	4781,50	-1,528	0,127	0,0854
Pozitivni rangi	81	78,93	6393,50			
Vezani rangi	15					
Skupaj	164					

Tabela 5.4: Analiza optimizma pri TEG.

Vseeno smo primerjali še obe metodi ocenjevanja po njunem BREbiasu. Rezultat *Mann–Whitney–Wilcoxonovega* testa je prikazan v prvem delu tabele 5.5.

Odgovor na naše trenutno raziskovalno vprašanje je pritrdilen. V naši študiji je TEG privedel do manj optimističnih ocen kot *Planning poker*, kar nakazuje že njun BREbias. To trditev nam dokazuje statistično značilna razlika med metodama, saj je P = 0,001. Spet pa je velikost učinka majhna, saj je Cliffova delta = 0,1951.

	N	Povprečen rang	Vsota rangov	Mann-Whitney U	Wilcoxon W	Z	P	Cliffova delta
BREbias *Planning pokra* - BREbias TEG								
TEG	164	188,44	30904,00	17374,000	30904,00	-3,381	0,001	0,1951
Planning poker	263	229,94	60474,00					
Skupaj	427							
BRE *Planning poker* - BRE TEG								
TEG	164	200,16	32826,50	19296,500	32826,500	-1,830	0,067	0,1042
Planning poker	263	222,63	58551,50					
Skupaj	427							

Tabela 5.5: Primerjava *Planning pokra* s TEG.

5.3.2 Ali so ocene, dobljene po metodi *Team estimation game*, bolj točne kot ocene, dobljene po metodi *Planning poker*?

Z odgovorom na prejšnje raziskovalno vprašanje smo ugotovili, da je optimizem pri TEG značilno manjši kot pri *Planning pokru*. Zanima nas še, kako se to pozna na točnosti metode. Analizirali smo ocene uporabniških zgodb, dobljene po TEG in izvedli primerjavo s končnimi ocenami *Planning pokra*.

S tem namenom je bil opravljen še en *Mann–Whitney–Wilcoxon* test, tokrat med BRE *Planning pokra* in BRE TEG. Rezultati so prikazani v drugem delu tabele 5.5.

Kljub medianama BRE *Planning pokra* = 0,5 in BRE TEG = 0,4065 je naš test pokazal, da razliki v točnosti med metodama ne moremo pripisati statistične značilnosti (vrednost P = 0,067). Tudi Cliffova delta kaže na zelo majhen učinek, njena vrednost je namreč enaka 0,1042.

5.4 Veljavnost študije

Celotna študija je bila zasnovana tako, da je zagotavljala kar največjo stopnjo veljavnosti. Simuliralo se je pravo industrijsko okolje, kolikor je bilo to mogoče glede na to, da je razvoj potekal v sklopu študija. Izdelek, ki se je razvijal, je bil zasnovan na osnovi študijskega informacijskega sistema, ki je bil dejansko razvit za Fakulteto za računalništvo in informatiko. Držali smo se principov Scruma in pravil uporabljenih metod ocenjevanja, študenti pa

so morali razviti testirano in delujočo programsko kodo.

Potencialna nevarnost veljavnosti naših rezultatov bi lahko bilo vprašanje, ali so študenti vestno in točno beležili čas, porabljen na razvoju uporabniških zgodb. A upamo si trditi, da je odgovor na to vprašanje pritrdilen. Študenti so se držali navodil in redno beležili svoj trud.

Z namenom, da bi še bolj utrdili veljavnost analize, je v študiji sodelovalo 20 skupin. Vse so razvijale enak izdelek in ocenjevale enak sklop uporabniških zgodb. S tem se ocene lahko direktno primerjajo [8].

Poglavje 6

Zaključek

Izvedena je bila študija, kjer je več skupin študentov razvijalo svoj projekt. To je bil študijski informacijski sistem. Razvoj je sledil principom metode Scrum, ki je danes najbolj razširjena agilna metoda. Programske zahteve so pri Scrumu tipično predstavljene v obliki uporabniških zgodb. Za ocenjevanje slednjih sta bili uporabljeni dve skupinski metodi ocenjevanja: *Planning poker* in TEG. Ocene in dejanski čas, porabljen na zgodbah, so se tekom študije shranjevali v AC Scrumu. Po koncu študije smo iz podatkovne baze AC Scruma izluščili vse podatke, potrebne za naše delo, in si tako pripravili temelje za analizo. Izvedena je bila analiza optimizma in točnosti ocen brez in z uporabo *Planning pokra* ter TEG. Med seboj smo primerjali tudi učinkovitost obeh metod.

V prvem delu analize smo pokazali, da je po pričakovanjih pri statistični kombinaciji začetnih ocen posameznikov v precejšnji meri prisoten optimizem. Slednji je prisoten tudi pri končnih ocenah, dobljenih s *Planning pokrom*. *Wilcoxon signed-rank* test, ki smo ga tu izvedli, kaže, da se je z uporabo te skupinske metode stopnja optimizma zmanjšala. Čeprav je velikost učinka zanemarljiva, smo se približali stališču, ki pravi, da *Planning poker* zmanjša začetni optimizem ocen. Dobljeni rezultati upamo, da bodo še dodatno spodbudili nadaljnje raziskave omenjene metode. Analiza točnosti pa je pokazala, da med točnostjo statistične kombinacije začetnih ocen posame-

znikov in končnih ocen, dobljenih po metodi *Planning poker*, ni statistično značilne razlike. Z upoštevanjem rezultatov študije [8], kjer so študenti z uporabo *Planning pokra* prišli celo do manj točnih ocen in [8, 9], kjer so, nasprotno, izkušeni strokovnjaki prišli do bolj točnih ocen, smo prišli do naslednjega zaključka. Menimo, da je vpeljevanje agilnih metodologij v študij bodočih razvijalcev programske opreme dobra stvar. Zgodnje seznanjenje študentov z agilnim razvojem in *Planning pokrom* (ter TEG) bi se lahko v veliki meri obrestovalo kasneje, ko si naberejo več izkušenj, pri njihovi profesionalni karieri na področju razvoja programske opreme.

Rezultati primerjave obeh metod so pokazali, da med točnostjo ocen, dobljenih po *Planning pokru* in TEG, ni statistično značilne razlike. To je bilo zaradi podobnosti metod deloma tudi pričakovano. A kaže, da je TEG privedel do manj optimističnih ocen. Velikost učinka je sicer tudi tu majhna. Nam pa vse to pove, da se TEG lahko kosa s *Planning pokrom*. Če upoštevamo še dejstvo, da je prvi zasnovan tako, da ocenjevanje poteka hitreje kot pri slednjem, lahko rečemo, da si TEG zasluži več pozornosti. Menimo, da bi lahko precej prispeval k razvoju programske opreme. Z našo študijo tako poskušamo prispevati k razpoznavnosti te metode in upamo, da se v bodoče opravi več podobnih raziskav ter da se posledično poveča uporaba TEG med razvijalci programske opreme.

Literatura

[1] K. Beck. Manifesto for agile software development. `http://agilemanifesto.org`, 2001. Dostopano: 19.3.2016.

[2] Mike Cohn. *User stories applied: For agile software development.* Addison-Wesley Professional, 2004.

[3] Mike Cohn. *Agile estimating and planning.* Pearson Education, 2005.

[4] Kristian Marius Furulund. Empirical research on software effort estimation accuracy. 2007.

[5] Shane Hastie in Stéphane Wojewoda. Standish group 2015 chaos report - Q&A with Jennifer Lynch. `http://www.infoq.com/articles/standish-chaos-2015`, 2015. Dostopano: 2.4.2016.

[6] Magne Jorgensen. Practical guidelines for expert-judgment-based software effort estimation. *IEEE software*, 22(3):57–63, 2005.

[7] Guillermo Macbeth, Eugenia Razumiejczyk, in R. D. Ledesma. Cliff's Delta Calculator: A non-parametric effect size program for two groups of observations. *Universitas Psychologica*, 10:545 – 555, 05 2011.

[8] Viljan Mahnič in Tomaž Hovelja. On using planning poker for estimating user stories. *Journal of Systems and Software*, 85(9):2086–2095, 2012.

[9] Kjetil Moløkken-Østvold, Nils Christian Haugen, in Hans Christian Benestad. Using planning poker for combining expert estimates in software projects. *Journal of Systems and Software*, 81(12):2106–2117, 2008.

[10] Marko Poženel in Viljan Mahnič. Studying agile software estimation techniques: the design of an empirical study with students. *Global Journal of Engineering Education*, 18(2), 2016.

[11] Anurag Prakash. What is sprint zero? `https://www.scrumalliance.org/community/articles/2013/september/what-is-sprint-zero`, 2013. Dostopano: 25.9.2016.

[12] Jeff Sutherland. *V gruču do uspeha*. Založba Pasadena d.o.o., 2016.

[13] Dave West, Tom Grant, M. Gerush, in D. D'silva. Agile development: Mainstream adoption has changed agility. *Forrester Research*, 2:41, 2010.

[14] Extreme programming. `https://en.wikipedia.org/wiki/Extreme_programming`. Dostopano: 25.11.2016.

[15] Kanban (development). `https://en.wikipedia.org/wiki/Kanban_(development)`. Dostopano: 25.11.2016.

[16] Scrumban. `https://en.wikipedia.org/wiki/Scrumban`. Dostopano: 25.11.2016.

[17] Wilcoxon signed-rank test. `https://en.wikipedia.org/wiki/Wilcoxon_signed-rank_test`. Dostopano: 9.4.2016.