Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Andrej Bačnar

Nadgradnja računalniškega orodja za vodenje agilnih projektov programske opreme

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Viljan Mahnič

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:
Nadgradnja računalniškega orodja za vodenje agilnih projektov programske opreme

Tematika naloge:

Vse večja razširjenost agilnih metod Scrum in Kanban zahteva razvoj primernih računalniških orodij za pomoč pri vodenju tovrstnih projektov. Eno izmed takih orodij je tudi ACScrum, ki se že nekaj let uporablja v pedagoškem procesu Fakultete za računalništvo in informatiko. Med njegovo uporabo so se pokazale potrebe po nadgradnjah, ki bi še bolj zmanjšale obseg administrativnega dela na strani pedagoškega osebja, omogočile vizualizacijo delovnega toka in izboljšale nadzor nad potekom študentskih projektov. V svojem diplomskem delu realizirajte omenjene nadgradnje s posebnim poudarkom na (1) avtomatizacji postopka za oblikovanje razvojnih skupin in pripravo projektov, (2) spremljanju razvojnega procesa preko Kanban table z vgrajenimi omejitvami obsega dela v izvajanju in (3) zagotavljanju boljšega nadzora nad izvajanjem predpisanih praks, vključno z avtomatskim opozarjanjem tistih razvojnih skupin, ki ne spoštujejo zahtevane metodologije.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Andrej Bačnar sem avtor diplomskega dela z naslovom:

Nadgradnja računalniškega orodja za vodenje agilnih projektov programske opreme

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom dr. Viljana Mahniča,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 16. marca 2015 Podpis avtorja:

Za pomoč pri izdelavi diplomske naloge se v prvi vrsti zahvaljujem mentorju prof. dr. Viljanu Mahniču, ki mi je bil v veliko podporo s svojim znanjem in praktičnimi nasveti.

Najlepša hvala tudi Katji in družini za podporo v času študija in tudi za potrpežljivost pri izdelavi te diplmske naloge.

Kazalo

Povzetek

Diplomska naloga opisuje razvoj nadgradnje programskega orodja za vodenje projektov po agilnih metodologijah. V prvem delu naloge so predstavljene osnovne značilnosti agilnih metodologij s poudarkom na metodologijah Scrum in Kanban ter njuni medsebojni primerjavi. Sledi kratek opis orodja, ki ga je bilo potrebno nadgraditi, in specifikacija zahtev za potrebne nadgradnje. Te zajemajo: vizualizacijo delovnega toka s pomočjo table, izdelavo dodatnih funkcionalnosti za spremljanje razvojnih skupin in izdelava dodatnih funkcionalnosti za lažjo administracijo uporabnikov. Sledi še predstavitev tehnične zasnove, ki zajema predstavitev uporabljenih tehnologij, podatkovnega modela in programske logike, ob koncu te diplomske naloge pa je predstavljena še uporaba nadgradenj v orodju.

Ključne besede: agilne metodologije, Scrum, Kanban, vodenje projektov, orodja za razvoj programske opreme.

Abstract

The thesis describes the development of an upgrade for an agile project management software tool. In the first part, thesis presents the basic characteristics of agile methodologies with the emphasis on Scrum and Kanban methodologies. The next chapter consists of a brief description of the existing tool and the upgrade requirements specification which includes: the workflow visualization by using the board, elaboration of additional functionality to monitor the development teams and creation of additional functionalities to facilitate the administration. Later the thesis focuses on the technical design including the presentation of the database model and the program logic. At the end of this thesis, the usage of the implemented upgrades is shown.

Keywords: agile metodologies, Scrum, Kanban, project management, software developement tools.

Poglavje 1

Uvod

Razvoj programske opreme se s časom spreminja. Vedno več podjetij ima cilj svoj produkt čim prej poslati na tržišče ter se hitro odzvati na spremembe v poslovnem okolju ali na spremenjene zahteve naročnika. Zaradi tega se vedno več podjetij za razvoj programske opreme odloča, da proces razvoja poteka po agilnih metodologijah. Med njimi je najbolj razširjena metodologija Scrum [9], v zadnjem času pa se vedno bolj uveljavlja metodologija Kanban in njene izpeljanke. Po zadnjih raziskavah [3] se izmed agilnih metodologij največ (55%) uporablja Scrum. Uporaba drugih metodologij, kot sta Kanban (5%) in Ekstremno programiranje(1%), je razmeroma nizka. Bolj pogosto se uporabljata v kombinaciji s Scrumom, kar velja predvsem za Ekstremno programiranje. A če pogledamo trend, lahko opazimo, da se je v zadnjih treh letih uporaba Kanbana skoraj podvojila [4] prav na račun povečanja uporabe v kombinaciji s Scrumom (Scrumban). Razlog za to je predvsem hitrejši način življenja, poslovno okolje se hitro spreminja in s tem tudi potrebe in želje naročnikov. Le-ti bi radi čimprej prišli do delujoče programske opreme.

Z naraščanjem uporabe agilnih metodologij se veča tudi potreba po orodjih za vodenje projektov po teh metodologijah. Samo v zadnjem letu se je uporaba teh orodij povečala za 6% [3]. Eno izmed orodij za vodenje projektov po metodologiji Scrum je bilo razvito tudi v Laboratoriju za tehnologijo

programske opreme na Fakulteti za računalništvo in informatiko in je bilo več let v uporabi v študijskem procesu. V tem času so se pojavile potrebe po nadgradnji orodja, predvsem potreba po vizualizaciji delovnega toka s pomočjo table. Poleg tega bi bile dobrodošle tudi nadgradnje za lažjo administracijo orodja, predvsem administracijo uporabnikov in projektov, pa tudi dodatne funkcionalnosti za spremljanje dela študentskih projektov. Cilj te diplomske naloge je torej nadgradnja omenjenega orodja, ki bi te potrebe zadovoljila in omogočila učinkovitejšo izvedbo projektov ter razbremenila pedagoško osebje pri nadzoru in administraciji.

V prvem delu diplomske naloge je opisan teoretični del, ki obsega predstavitev agilnih metodologij, kratko predstavitev metodologij Scrum in Kanban ter primerjavo med njima. V nadaljevanju je predstavljen razvoj orodja, ki obsega analizo obstoječega orodja, specifikacijo zahtev in predstavitev tehnične zasnove, na koncu pa je predstavljena še uporaba orodja in njegovih novih funkcionalnosti.

Poglavje 2

Agilne metodologije

2.1 Predstavitev agilnih metodologij

Agilne metode razvoja programske opreme so skupina metod, ki so v primerjavi z klasičnim slapovnim razvojem bolj prilagodljive do sprememb v okolju in posledično do sprememb v zahtevah naročnikov programske opreme. Leta 2001 je skupina 17-ih razvijalcev programske opreme izdala Manifest agilnega razvoja programske opreme, v katerem navajajo vrednote, ki jim sledijo agilne metodologije [2]. Prednost imajo:

- **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

V kratkem povzetku lahko opazimo, da je ključnega pomena dobra samoorganizacija in sodelovanje med člani razvojne skupine. Za stranko je bolj koristna delujoča programska oprema kot pa obsežna in natančna dokumentacija. Cilj je čim hitreje razviti čim več delujočih posameznih funkcionalnosti. Sodelovanje s stranko je ključnega pomena, saj lahko do neke mere še med samim razvojem spremeni prioritete in zahteve po neki funkcionalnosti.

Manifest agilnih metodologij temelji na dvanajstih principih [2]:

- Najvišja prioriteta je zadovoljiti stranko z zgodnjim in nepretrganim izdajanjem programske opreme.
- Sprejemamo spremembe zahtev, celo v poznih fazah razvoja. Agilni procesi vprežejo tovrstne spremembe v prid konkurenčnosti naše stranke.
- Delujočo programsko opremo izdajamo pogosto, znotraj obdobja nekaj tednov, do nekaj mesecev, s preferenco po krajšem časovnem okvirju.
- Poslovneži in razvijalci morajo skozi celoten projekt dnevno sodelovati.
- Projekte gradimo okrog motiviranih posameznikov. Omogočimo jim delovno okolje, nudimo podporo in jim zaupamo, da bodo svoje delo opravili.
- Najboljša in najučinkovitejša metoda posredovanja informacij razvojni ekipi in znotraj ekipe same, je pogovor iz oči v oči.
- Delujoča programska oprema je primarno merilo napredka.
- Agilni procesi promovirajo trajnostni razvoj. Sponzorji, razvijalci in uporabniki morajo biti zmožni konstantnega tempa za nedoločen čas.
- Nenehna težnja k tehnični odličnosti in k dobremu načrtovanju izboljša agilnost.
- Preprostost – umetnost zmanjševanja količine nepotrebnega dela – je bistvena.
- Najboljše arhitekture, zahteve in načrti izhajajo iz tistih ekip, ki so samoorganizirane.
- V rednih časovnih razdobjih ekipa išče načine, kako postati učinkovitejša ob rednem prilagajanju svojega delovanja.

Seveda pa imajo agilne metodologije tudi slabosti. Pri velikih in obsežnih projektih je težko predvideti, koliko časa in truda bo projekt zahteval, saj na začetku ne naredimo obsežnih analiz, kot v primeru klasičnega slapovnega razvoja. Dokumentacija je običajno slabša, kar lahko oteži vzdrževanje in morebitne nadgradnje. Predstavnik naročnika je tesno vpet v razvoj programske opreme, zato mora imeti jasno opredeljene cilje, vedeti mora, kakšno programsko opremo si želi imeti. V nasprotnem primeru lahko ekipa zaide in rezultat ni tak, kot si naročnik želi. Pogosto si naročnik ne vzame dovolj časa za potrebne pogovore. Poleg tega pa morajo tudi člani razvojne ekipe imeti določene izkušnje. Med samim razvojem je potrebno sprejemati pomembne odločitve, ki lahko ogrozijo projekt. Če so v ekipi novinci, je potrebno ekipo kombinirati z izkušenimi programerji. V razvojni ekipi je potrebno veliko komunikacije in sodelovanja, zato je pomembno, da so člani sposobni za delo v skupinah. Velika prednost agilnih metodologij je fleksibilnost razvoja. Če se pojavi nova zahteva, je ni težko realizirati. A hkrati je to lahko velika slabost, saj se lahko v primeru novih in novih zahtev projekt nikoli ne konča.

Glavni predstavniki agilnih metodologij so Scrum, Kanban, Ekstremno programiranje, vitek razvoj programske opreme (LSD - Lean software developement), prilagodljivi razvoj programske opreme (ASD - Adaptive software development) in drugi. Vsaka od naštetih ima določene prednosti in slabosti, zato se poleg osnovnih metod uporablja tudi več variant in hibridov, kot so denimo Scrumban - Scrum/Kanban hibrid, Scrum/XP hibrid in drugi.

2.2 Metodologija Scrum

Scrum je najpogostejša izmed agilnih metodologij, ki se uporablja za razvoj programske opreme. Scrum in njegove različice uporablja 73% razvijalcev, ki uporabljajo agilne metodologije [3].

V nadaljevanju so na kratko predstavljene aktivnosti, ki so značilne za Scrum. Slovenska terminologija in povzetek sta večinoma vzeta iz diplomske naloge Anžeta Časarja [7]. Orodje, ki je bilo izdelano v okviru njegove

diplomske naloge je bilo namreč izhodišče za to diplomsko nalogo.

Zahteve so podane v obliki uporabniških zgodb (ang. user stories) in testnih scenarijev, s katerimi ob koncu preverimo delovanje funkcionalnosti [11]. Vse zgodbe so zbrane v dokumentu, imenovanem seznam zahtev (ang. Product Backlog). Razvoj programske opreme poteka, podobno kot pri drugih agilnih metodologijah, v večih iteracijah, ki jih imenujemo sprinti. Dolžina in število sprintov se določi na začetku projekta. Prvotno naj bi bili dolgi 30 dni [9], a se je v praksi pokazalo, da so lahko tudi krajši. Dandanes se v praksi večinoma uporabljajo sprinti dolžine dveh do štirih tednov. Podmnožica zgodb, ki jih bo razvojna skupina realizirala tekom sprinta, se imenuje seznam nalog (ang Sprint Backog).

Scrum definira tri vloge v projektu:

- **Produktni vodja (ang. Product Owner)**: produktni vodja je predstavnik naročnika. Njegova naloga je specifikacija zahtev in določanje prioritet, saj ve, kaj naročnik potrebuje. Njegov cilj je v najkrajšem možnem času priti do delujoče programske opreme, ki podjetju oziroma organizaciji prinese največjo konkurenčno prednost. Na voljo mora biti za odgovore na dodatna vprašanja, ki se pojavijo tekom razvoja programske opreme. Odloča, kdaj je neka zgodba dokončana in primerna za uporabo.

- **Člani razvojne skupine (ang. Team Members)**: to je skupina, ki razvija neko funkcionalnost. Člani razvojne skupine imajo veliko odgovornost v primerjavi s klasičnimi oblikami vodenja, kjer je odgovornost prenešena na nadrejene. Člani sodelujejo pri določanju obsega dela za določen sprint, znotraj njega pa so popolnoma svobodni pri izbiri načina izvedbe.

- **Skrbnik metodologije (ang. Scrum Master)**: vodja skupine, ki opravlja funkciji vodenja in nadzora. Le-ta se pri Scrumu razlikuje od običajnega, saj razvojni skupini dela ne nalaga, ampak le skrbi za pravilen in nemoten potek procesa Scrum. Za dobro vodenje potrebuje

dobro poznavanje Scruma in izkušnje pri delu z njim. Razvojno skupino ščiti pred zunanjimi dejavniki, ki bi motile razvojni proces. Zagotavlja jim vse potrebne vire za čimbolj produktivno delo.

Pred samo izvedbo projekta je z razvojno skupino pomembno definirati, kaj pomeni, da je neka zgodba končana (ang. Definition of Done). Na vsak način morajo biti tiste zgodbe, ki so označene kot končane, ob koncu sprinta primerne za objavo.

Na osnovi seznama zahtev (Product Backlog) je potrebno definirati dolžino in število sprintov. Pred vsakim sprintom se izvede sestanek za načrtovanje sprinta (ang. Sprint Planning Meeting), ki traja približno en delovni dan. V prvi polovici sestanka produktni vodja predstavi uporabniške zgodbe, s poudarkom na tistih zgodbah, ki imajo najvišjo prioriteto. Razvojna skupina se skupaj s produktnim vodjem odloči, katere zgodbe bodo naredili v prihajajočem sprintu, in jih doda v seznam nalog za sprint (Sprint Backlog). Razvojna skupina mora oceniti zahtevnost posamezne zgodbe v točkah (ang. Story points). Ena točka običajno ustreza šestim uram efektivnega dela. Vsaka skupina ima svojo hitrost sprinta (ang. Velocity), ki nam pove, koliko točk lahko skupina realizira v enem sprintu. V sprint je potrebno dodati toliko zgodb, da je njihova vsota enaka predvideni hitrosti sprinta. V drugi polovici sestanka ima razvojna skupina čas, da posamezne zgodbe dodatno razčleni na naloge (ang. tasks) ter jih oceni. Produktni vodja mora biti na voljo za morebitna dodatna pojasnila.

Med sprintom je vsako jutro na sporedu 15 minutni pregledni sestanek (ang. Daily Scrum Meeting), na katerem mora vsak od članov razvojne skupine odgovoriti na tri ključna vprašanja:

1. Kaj je bilo narejeno z njegove strani v času od zadnjega sestanka?
2. Kaj bo predvidoma narejeno do naslednjega sestanka?
3. Ali je med razvojem naletel na težave in kakšne težave so to bile?

Namen tega sestanka je usklajevati delo celotne skupine in sproti reševati morebitne težave. Na koncu sprinta sta predvidena dva sestanka. Prvi se

imenuje Sestanek za pregled rezultatov sprinta (ang. Sprint Review Meeting), na katerem razvojna skupina produktnemu vodji predstavi opravljeno delo. Na sestanku so lahko prisotni vsi, ki so zainteresirani za projekt. Produktni vodja preveri sprejemne teste in oceni opravljeno delo. V kolikor je posamezna zgodba ustrezna, jo sprejme, v nasprotnem primeru pa jo zavrne. Drugi sestanek se imenuje Sestanek za oceno kakovosti razvojnega procesa (ang. Sprint Retrospective Meeting). Skliče ga skrbnik metodologije. Na sestanku skupina oceni sprint in poskuša najti tako dobre prakse kot tudi pomanjkljivosti v procesu razvoja. Na ta način lahko odkrijemo rezerve za izboljšanje produktivnosti in s tem dosežemo neprestano izboljševanje razvojnega procesa.

2.3 Metodologija Kanban

Kanban je ena izmed najhitreje rastočih agilnih metod. V letu 2012 se je njegova uporaba skoraj podvojila, predvsem na račun večje uporabe metode Scrumban [1]. V zadnjih dveh letih se je rast sicer upočasnila, a glede na ankete še vedno ostaja ena izmed najbolj pogosto uporabljenih metod [3].

Ime izvira iz japonskega jezika in pomeni signalna kartica (ang. signal card). S pomočjo vidnih signalov v procesu razvoja sporočamo, ali je v neki fazi prišlo do situacije, kjer je obseg dela pod dogovorjenim nivojem. Temu mehanizmu pravimo “pull” stategija; ko dela ni dovolj, si ga vzamemo sami, namesto da nam ga nekdo od zunaj potisne v sistem (“push” strategija) [6]. Kanban je razvil Taiichi Ohno pri Toyoti in je ena izmed metod, primerih za doseganje koncepta “just-in-time” proizvodnje [5]. Kanban uvaja prakse vitke proizvodnje v razvoj programske opreme z namenom zmanjševanja izgub, povečanja produktivnosti in skrajševanja dobavnih rokov [6].

Glavni cilj Kanbana je doseči čim manjši povprečni potrebni čas (ang. average lead time) za dokončanje neke funkcionalnosti. To dosežemo z omejitvijo števila oziroma obsega zahtev, ki jih lahko imamo sočasno v razvoju (ang. Work In Progress limit - WIP limit). Za vizualizacijo delovnega toka

uporabljamo tablo, ki je razdeljena na več stolpcev. Vsak stolpec predstavlja posamezno fazo v razvoju programske opreme [6]. Vsako funkcionalnost običajno predstavimo z uporabniškimi zgodbami (ang. user stories), ki so lahko tudi razčlenjene na naloge (ang. tasks). Kanban ne predpisuje, kako celoten projekt razčleniti na uporabniške zgodbe in naloge, je pa zaželeno, da imajo zgodbe čim bolj podoben obseg in strukturo. Uporabniške zgodbe oziroma naloge nato zapišemo na kartice. Kartice pomikamo po tabli glede na to, v kateri fazi razvoja se nahaja uporabniška zgodba. Vsak stolpec ima lahko omejitev števila kartic in sicer z namenom, da se skrajša potrebni čas za izdelavo neke funkcionalnosti. S tem tudi preprečimo, da bi v neki fazi razvoja člani skupine delali na več nalogah istočasno. Preko table lahko preprosto vidimo sliko celotnega projekta in prepoznamo ozka grla, tako da jih razvojna skupina lahko prednostno obravnava ter odpravi [8].

Omejitev WIP ima lahko tudi slabosti. V primeru prenizke omejitve WIP nekateri člani razvojne skupine lahko ostanejo brez dela, saj jim omejitev ne dovoli potegniti nove zgodbe v neko fazo in jih prisili v reševanje problematičnih zgodb ali celo čakanje, kar se lahko odraža v slabši produktivnosti. Nizka omejitev WIP sicer pomaga pri identifikaciji ozkih grl. Ob previsoki omejitvi WIP pa lahko pride do večjega števila zgodb ki mirujejo in s tem podaljšujejo povprečen čas razvoja zgodbe (lead time).

Nekatere table pri Kanbanu imajo tudi dodatno plavalno progo (ang. swimlane). Uporablja se za nujne primere za zgodbe z najvišjo prioriteto. Po tej plavalni progi lahko kartico vseeno pomaknemo naprej, čeprav s tem kršimo omejitev WIP. Slaba stran dodatne proge je možnost njene zlorabe za nenujne zgodbe.

Kot že rečeno, glavno merilo uspešnosti Kanban projektov je čas od vstopa do zaključka neke naloge v procesu razvoja (ang. lead time). Če cel projekt dobro razdelimo na uporabniške zgodbe z enakim obsegom dela, potem ocenjevanje zgodb ni več bistvenega pomena in postane nepotrebno, saj lahko preko potrebnega časa previdimo, kdaj bo določena zgodba realizirana in na voljo naročniku [6].

V nasprotju z metodo Scrum, Kanban ne predpisuje vlog, iteracij in ocenjevanja zgodb. Potreb po določevanju kakršnih koli časovnih okvirjev ni. Vse je odvisno od razvojne skupine in projekta, ki ga želi izvesti.

Čisti Kanban je bolj primeren za projekte, kjer ni potrebno posebno planiranje, na primer pri vzdrževanju ali pri podpori strankam, kjer je hiter odzivni čas najpomembnejši. Za razvoj nove programske opreme pa se uporabljajo kombinacije Kanbana z drugimi aglinimi metodologijami, predvsem s metodologijo Scrum, ki ponuja nekaj naprednejših tehnik planiranja [6].

2.4 Primerjava metod Kanban in Scrum

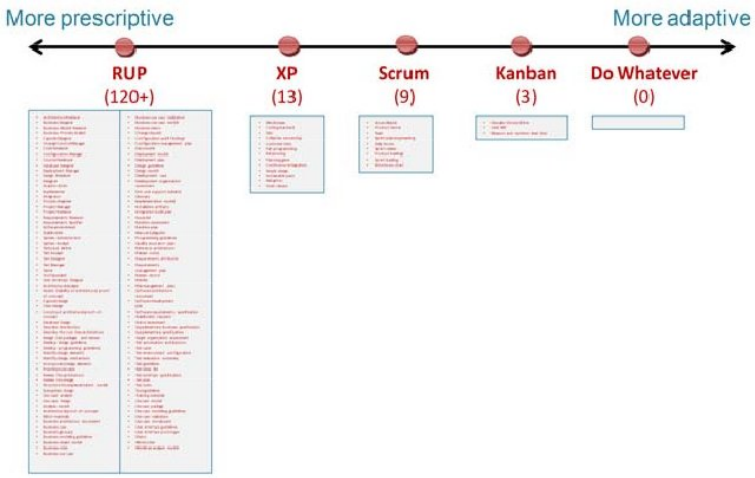
Tako Kanban kot tudi Scrum sta agilni metodologiji za razvoj programske opreme. Metodologiji nam predpisujeta le nekatere omejitve in navodila, kako organizirati proces razvoja, nikakor pa to ni dovolj za uspešno realizacijo projektov. Dobra metodologija ni nujno pogoj, da bo projekt uspel, niti ni nujno razlog za uspeh. Enako velja za slabe metodologije. Slaba metodologija ni nujno razlog za neuspeh projekta in tudi s slabo metodologijo je mogoče projekt uspešno zaključiti [10].

Kanban in Scrum se razlikujeta v številu pravil in omejitev, ki jih predpisujeta. Kot predstavnika agilnih metodologij sicer spadata v kategorijo, kjer je pravil najmanj, a jih ima Scrum vseeno dovolj, da za nekatere projekte ni najbolj primeren, kot na primer za vzdrževanja. Nove naloge lahko prihajajo dnevno. V tem primeru bi lahko skrajšali sprint na en dan, a za realizacijo nekaterih nalog potrebujemo več kot en dan, zato sprinti ne pridejo v poštev.

Primerjavo omejitev in predpisov med posameznimi metodologijami lahko vidimo na Sliki 2.1.

V nadaljevanju so predstavljene omejitve, predpisi in prakse, ki se tičejo metodologij Scrum in Kanban. V knjigi “Kanban and Scrum – making the most of both” sta jih predstavila H. Kniberg in M. Skarin [10].

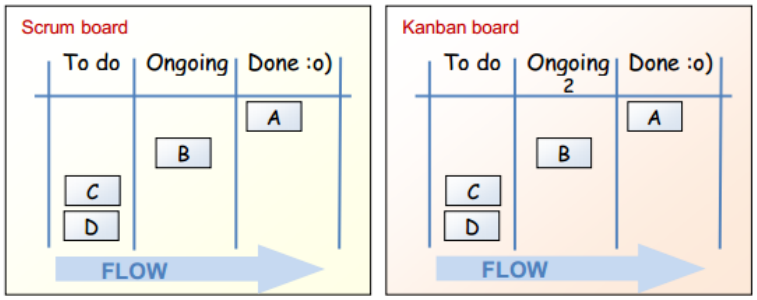
- **Vloge:** Scrum predpisuje tri vloge: produktni vodja (Product Owner), skrbnik metodologije (Scrum master) in člani razvojne skupine (Team



Slika 2.1: Slika prikazuje število pravil, ki jih posamezne metodologije predpisujejo

Members). Kanban vlog ne predpisuje, kar pa ne pomeni, da jih ne smemo imeti. Lahko jih imamo, lahko celo dodamo druge vloge, ampak je potrebno dobro premisliti, ali jih res potrebujemo. V Kanban projektih se sicer pogosto uporablja vloga produktnega vodje (Product Owner), ni pa nujna. V večjih projektih je verjetno smiselno imeti tudi vodjo projekta, ki skrbi za koordiniranje skupine, pri manjših projektih pa lahko to celo škodi. V primeru, da nismo prepričani, ali neko vlogo res potrebujemo, je bolje da je ne uporabimo. V tem primeru raje upoštevajmo princip "manj je več" [10].

- **Iteracije:** Scrum predpisuje več iteracij (sprintov), ki trajajo enako časa. Na začetku vsake iteracije ustvarimo plan. Uporabniške zgodbe dodajamo v sprint po prioriteti. Med samim sprintom spremembe niso mogoče, ampak skupina dela na zgodbah, za katere se je zavezala, da jih bo končala. Na koncu sprinta skupina predstavi opravljeno delo naročnikom. Sledi še sestanek skupine z namenom ovrednotenja in izboljšanja procesa. V eni iteraciji tako kombiniramo tri različne aktivnosti - planiranje, izboljšanje procesa in, če gre vse po načrtu, tudi



Slika 2.2: Razlika med tablo za Scrum in tablo za Kanban pri nekem preprostem projektu.

oddajo oziroma objavo nove funkcionalnosti.

Pri Kanbanu iteracije niso predpisane, tako da planiranje, izboljšanje procesa in oddaja produkta potekajo po želji. Novo programsko funkcionalnost lahko oddajamo na primer vsak teden ali pa preprosto takrat, ko je na voljo.

- **Omejitev količine dela, ki ga imamo sočasno v razvoju (ang. Work In Progress limit - WIP):** Scrum omejuje količino dela na način, da v neko iteracijo (sprint) lahko damo le toliko zgodb, da vsota njihovih točk ne presega hitrosti razvojne skupine. Kanban na drugi strani omejuje količino dela, ki ga imamo lahko v neki fazi razvoja. Če primerjamo tabli za Scrum in Kanban za nek preprost projekt, opazimo, da je tabla za Scrum, ki prikazuje en sprint, skoraj identična kot tabla za Kanban. Primer je prikazan na Sliki 2.2. Na njej je edina razlika številka 2 na tabli za Kanban, ki pomeni omejitev števila kartic v fazi "Ongoing".

  Skupine, ki razvijajo po metodologiji Scrum, običajno ugotovijo, da veliko število kartic v razvoju ni najbolj pametna ideja, zato najprej dokončajo začete zgodbe, preden so lotijo nove. Če na Scrum tablo dodajo to omejitev, Scrum tabla kar naenkrat postane Kanban tabla.

  V primeru kartic, ki imajo zelo različne časovne zahtevnosti, je na Kan-

ban tabli smiselno namesto števila kartic omejiti število točk v posamezni fazi razvoja.

- **Obe metodologiji sta empirični:** Empirični sta v tem smislu, da se od udeležencev pričakuje, da bodo eksperimentirali s procesom tako, da ga bodo prilagajali svojim potrebam. Nikjer namreč ni predpisano, kakšne morajo biti WIP omejitve, niti ni predpisano, koliko dela si je potrebno naložiti v neko iteracijo. Scrum predlaga, da člani razvojne skupine pokrivajo različna področja (ang. cross-functional teams), a nikjer ni predpisano katera.

  Za merjenje učinkovitosti metodologije lahko uporabimo kazalce, kot so na primer povprečni potrebni čas za izdelavo neke zgodbe (average lead time), kvaliteta produkta, produktivnost in zadovoljstvo strank. Nekaterih kazalcev ni tako preprosto izmeriti, ampak se moramo pogosto zanašati na lastno mnenje. Pri razvoju programske opreme so zato zelo pomembne povratne zanke (ang. feedback loops). Pri Scrumu so le-te predpisane in vključene v proces preko vsakodnevnih sestankov in predvsem preko sestanka za oceno kakovosti razvojnega procesa (ang. Sprint Retrospective Meeting), ki je na sporedu ob koncu vsakega sprinta. Pri Kanbanu taki sestanki niso nujni, so pa zaželeni. Kanban ima predost v tem, da so preko table lepo vidna ozka grla, prav tako je povprečen čas izdelave zgodbe (lead time) dober pokazatelj učinkovitosti. Ti kazalci so vidni takoj, zato lahko hitreje reagiramo v primerjavi s Scrumom. Kratka povratna zanka je lahko prednost. Prepogoste spremembe niso zaželene, saj se mora proces stabilizirati, preden ga lahko ponovno pravilno ovrednotimo.

- **Odzivnost na spremembe:** V primeru, da pride do nove zahteve z visoko prioriteto, je pri Scrumu potrebno počakati do naslednjega sprinta, preden lahko skupina obravnava zahtevo. Med sprintom namreč razvojne skupine ne smemo motiti z novimi nalogami. V povprečju je tako čas za sprejem nove zgodbe v sprint enak polovici dolžine sprinta.

Pri Kanbanu lahko takoj začnemo z obravnavo nove zgodbe, edina omejitev pri tem je, da v neki fazi ne sme biti preveč zgodb. Če je omejitev v neki fazi dosežena, velja pravilo, da lahko začnemo z novo zgodbo šele takrat, ko je neka zgodba končana.

- **Sestava razvojne skupine:** Pri Scrumu je predpisano, da je skupina sestavljena iz vseh področij, potrebnih za realizacijo zgodb (ang. cross-functional teams). Tabla tako pripada eni sami razvojni skupini. Pri Kanban tabli je obratno. Lahko si jo deli več razvojnih skupin sočasno. Tabla se namreč navezuje na nek pretok dela (workflow) in ni last razvojne skupine. Seveda pa je običajno potrebno definirati, katera skupina ima pristojnosti za določeno področje oziroma aktivnost.

- **Časovna zahtevnost zgodb:** Pri Scrumu je potrebno paziti, da nobena zgodba ni preobsežna za realizacijo v enem sprintu. Taka zgodba pri Kanbanu sicer ni prepovedana, vendar je smiselno razdeliti zahtevo na več enakovrednih manjših zgodb.

- **Ocenjevanje zgodb:** Pri Scrumu je potrebno vsako zgodbo oceniti. Še več, zgodbo je potrebno razčleniti na naloge in jih prav tako oceniti. Prav tako je potrebno oceniti, koliko dela lahko razvojna skupina opravi v enem sprintu. Kanban ocenjevanja ne predpisuje. Če pride do zahteve za oceno potrebnega dela, je odvisno od skupine, kako bo ocenjevanje izvedla.

- **Delo na več projektih hkrati:** Pri obeh metodologijah je možno delati na več projektih hkrati. Angleški izraz Product Backlog (seznam zahtev, potrebnih za realizacijo projekta) v tem primeru ni najboljši. V seznam zahtev damo zgodbe vseh projektov, zato bi bil boljši izraz "Team backlog", ki bi označeval zgodbe dodeljeni neki skupini. Razvoja se lahko lotimo tako, da v prvi sprint damo zgodbe enega projekta, v drug sprint zgodbe drugega projekta in tako dalje. Druga možnost pa je, da v sprinte dodamo zgodbe iz obeh projektov,

razvrščene po prioriteti. Pri Kanbanu lahko za vsak projekt na tabli uporabimo drugo barvo kartice in jih po tabli premikamo po običajnih pravilih. Druga možnost je, da za vsak projekt naredimo svojo plavalno progo (ang. swimlane), tako da imamo kartice različnih projektov med sabo ločene.

Ko se odločamo, po kateri metodologiji bomo projekt izpeljali, je najbolj pomembno, da se ne omejimo na principe iz ene metodologije, ampak jih med seboj kombiniramo po lastnih željah in potrebah. Veliko skupin, ki razvijajo po metodologiji Kanban, uporablja prakse iz drugih agilnih metodologij.

Ker se pri Kanbanu v veliki meri uporabljajo prakse iz metodologije Scrum, vedno bolj v veljavo prihaja metodologija **Scrumban**. Le ta označuje kombinacijo fleksibilnosti Kanbana z dodajanjem pravil in praks Scruma.

Poglavje 3

Razvoj orodja za vodenje projektov

3.1 Analiza obstoječega orodja

Obstoječe orodje je opisano v diplomski nalogi Anžeta Časarja [7].

V tem orodju so bile že implementirane naslednje funkcionalnosti:

- Prijava in odjava uporabnikov, dodelitev uporabniških vlog, urejanje uporabnikov.
- Kreiranje projektov, urejanje projektov, urejanje vlog na projektu.
- Kreiranje in urejanje uporabniških zgodb, razbitje le-teh na naloge (tasks) in dodelitev nalog članom razvojne skupine.
- Ocenjevanje zgodb na več načinov: "Planning poker", "Team estimation game", skrbnik metodologije lahko dodeli oceno tudi sam.
- Kreiranje in urejanje sprintov, dodajanje zgodb v sprint in vodenje evidence opravljenega dela na posameznih nalogah.
- Sprejemanje in zavračanje zgodb s strani produktnega vodje.

- Možnost izvajanja vsakodnevnih sestankov (Daily Scrum meetings) in podpora za pisanje dokumentacije projekta.

- Analize: Grafikona "Release burndown chart" in "Team involvement report", poročilo o realizaciji ("Realisation report"), poročilo o razbitju zgodb na naloge ("Decomposition report").

Orodje se že nekaj let uporablja v študijske namene pri nekaterih predmetih na Fakulteti za računalništvo in informatiko. Večje pomanjklivojsti niso bile odkrite, je pa to obdobje dalo nekaj idej, kako orodje še izboljšati. Zaradi večjega števila študentov je bila administracija uporabnikov dokaj zapletena. Težavo je predstavljal vnos uporabnikov v orodje. Ročni vnos uporabnikov preko obrazca v aplikaciji bi trajal predolgo, zato se je uvoz izvajal kar direktno v podatkovno bazo, kar pa pomeni povečanje možnosti za neskladje podatkov v bazi in posledično napak v delovanju orodja. Podobna težava je bila s kreiranjem projektov. Zaradi velikega števila študentov je bilo potrebno kreirati tudi veliko projektov. Vsi projekti so sicer imeli enake uporabniške zgodbe, vendar bi ročni vnos vseh zgodb preko aplikacije zahteval preveč časa. Tu se je pojavila potreba po multipliciranju projektov. V primeru naknadno dodane zgodbe je bila težava podobna, saj je bilo potrebno zgodbo dodati v vsak projekt posebej.

Poleg administrativnih ovir je bila slabost orodja tudi slaba vizualizacija delovnega toka. Za pregled opravljenega dela v primerjavi s preostalim delom je sicer bil na voljo grafikon "Release burndown chart", kar pa ni dovolj. Razvoj programske opreme običajno poteka po več fazah, njihovo število je lahko zelo različno. Orodje ni omogočalo spremljanja posamezne uporabniške zgodbe skozi te faze. Na voljo je bil le seznam zahtev (Product Backog), seznam izbranih zgodb za trenutni sprint (Sprint Backlog) in seznam dokončanih zgodb. Tako se je pojavila potreba po tabli, na kateri bi v vsakem trenutku na enem mestu lahko videli, v kateri fazi procesa razvoja se nahaja posamezna uporabniška zgodba. Na tabli bi lahko tudi dodatno omejili obseg dela v izvajanju.

Orodje se je uporabljalo v študijskem procesu, zato je bilo pomembno vedeti, kako dosledno so študenti izvajali prakse Scruma, torej, ali so se res preko orodja udeleževali vsakodnevnih sestankov. Zaradi večjega števila študentov se je pojavila potreba po avtomatiziranemu preverjanju udeležbe na teh sestankih. To ne velja samo za vsakodnevne sestanke, ampak tudi za druga pravila, kot so dekompozicija uporabniških zgodb na naloge in ocenjevanje nalog.

3.2 Specifikacija zahtev

V specifikaciji zahtev so navedene vse funkcionalnosti, ki jih je bilo treba izdelati v okviru te diplomske naloge. Zahteve so podane v obliki kartic. Na vsaki kartici je predstavljena uporabniška zgodba in njeni sprejemni testi. Uporabniška zgodba je predstavljena na način, da odgovori na vprašanje, kateri uporabniški vlogi je funkcionalnost namenjena in za kakšno funkcionalnost gre. Zajema lahko tudi dodatna pojasnila o zahtevani funkcionalnosti. Sprejemni testi pa določajo, kako preverimo pravilnost delovanja nove funkcionalnosti.

Predstavitev uporabniških zgodb in sprejemnih testov je razdeljena na več sklopov. Prvi in glavni sklop obsega zgodbe v povezavi z vizualizacijo delovnega toka oziroma table, sledijo zgodbe za spremljanje in nadzor razvojnih skupin, v zadnjem sklopu pa so predstavljene zgodbe, ki omogočajo lažjo administracijo.

3.2.1 Vizualizacija delovnega toka

Skrbnik metodologije ima možnost kreiranja table za vsak sprint. Tabla ima tri fiksne stolpce: “Product backlog”, “Sprint backlog” in “Done”. Med stolpca “Sprint backlog” in “Done” lahko skrbnik metodologije vrine poljubno število stolpcev in podstolpcev. Na začetku naslednjega sprinta ima tabla enako strukturo kot na koncu prejšnjega.

- Preveri kreiranje table za prvi sprint.
- Preveri kreiranje table za vsak naslednji sprint.
- Preveri dodajanje stolpca.
- Preveri dodajanje podstolpca.

Skrbnik metodologije lahko ureja in briše stolpce. Pred brisanjem stolpca mora biti le-ta prazen. Skrbnik metodologije lahko spreminja tudi vrstni red stolpcev.

- Preveri urejanje stolpca (sprememba imena, sprememba omejitve WIP).
- Preveri brisanje stolpca, v katerem so kartice.
- Preveri brisanje stolpca, v katerem ni kartic.
- Preveri brisanje nadstolpca.
- Preveri spreminjanje vrstnega reda stolpcev.
- Preveri spreminjanje vrstnega reda podstolpcev.

Skrbnik metodologije lahko dodaja ali odvzema dovoljenja za premik kartic iz enega stolpca v drugega za vse sodelujoče na projektu.

- Preveri dodajanje dovoljenj za premik za skrbnika metodologije.
- Preveri dodajanje dovoljenj za premik za produktnega vodjo.
- Preveri dodajanje dovoljenj za premik za člane razvojne skupine.
- Preveri odvzem dovoljenja za premik za vse vloge.
- Preveri, ali dovoljenja za premik ostanejo tudi na tabli, katere struktura je bila prenešena iz prejšnjega sprinta.

Skrbnik metodologije lahko omeji obseg dela v posameznem stolpcu. To lahko naredi na dva načina: z omejitvijo števila kartic in/ali z omejitvijo skupnega števila točk. Če je omejitev dosežena, se lahko kartica vseeno prestavi, vendar je treba ob tem zabeležiti razlog.

- Preveri dodajanje omejitev na stolpec pri njegovem urejanju.
- Preveri premik kartice v stolpec, kjer omejitev števila kartic še ni presežena.
- Preveri premik kartice v stolpec, kjer je omejitev števila kartic že presežena.
- Preveri premik kartice v stolpec, kjer omejitev skupnega števila točk še ni presežena.
- Preveri premik kartice v stolpec, kjer je omejitev skupnega števila točk že presežena.
- Preveri premik kartice, katera še nima ocene, v stolpec "Sprint backlog".

Vsi sodelujoči na projektu imajo možnost pregledovanja table s karticami, ki predstavljajo uporabniške zgodbe.

- Preveri prikaz table za vse uporabniške vloge.

Vsi sodelujoči na projektu lahko do podrobnosti in urejanja uporabniške zgodbe dostopajo tudi z izbiro kartice na tabli.

- Preveri dostop do urejanja zgodbe preko table.
- Preveri dostop do ocenjevanja "Planning Poker" preko table.
- Preveri dostop do opomb (notes) preko table.
- Preveri dostop do urejanja nalog (tasks) preko table.

Vsi sodelujoči na projektu lahko vidijo zgodovino premikanja kartice iz stolpca v stolpec.

- Preveri dostop do zgodovine premikanja kartice.
- Preveri zgodovino premikanja za vse kartice.
- Preveri zgodovino premikanja posebej za eno kartico.

Vsi sodelujoči na projektu lahko izpišejo število kršitev WIP za določeno podmnožico kartic v izbranem časovnem obdobju.

- Preveri prikaz vseh kršitev WIP.
- Preveri prikaz kršitev WIP za izbrano podmnožico kartic.
- Preveri kršitev WIP za nadstoplec/podstolpec.

3.2.2 Spremljanje in nadzor razvojnih skupin

Vsi sodelujoči na projektu lahko določijo podmnožico kartic in izdelajo kumulativni diagram delovnega toka (ang. Comulative Flow Diagram) za določeno časovno obdobje. Za izbrano časovno obdobje diagram prikazuje, koliko kartic je bilo v vsakem stolpcu po posameznih dnevih.

- Preverjaj pravilen prikaz diagrama skozi celoten projekt.

Vsi sodelujoči na projektu si lahko ogledujejo grafikon "Sprint Burndown chart". Na tem grafikonu je tabelarično in grafično prikazana primerjava vloženega dela s preostalim delom v sprintu.

- Preveri pravilen prikaz grafikona za prvi sprint.
- Preveri pravilen prikaz grafikona za naslednji sprint.
- Preveri spreminjanje grafikona ob vsakem opravljenem delu.

- Preveri, da prikaz pravilno upošteva zgodbe, ki niso bile dokončane v predhodnem sprintu.

Produktni vodja ima možnost preverjanja udeležbe na vsakodnevnih sestankih. Na začetku je treba izbrati časovni interval (datum od – do), v katerem bi moral biti sestanek izpeljan, in projekte, ki jih želimo preverjati. Za vsak projekt se preveri, ali je bil znotraj izbranega časovnega intervala sestanek pravilno izveden. Izpišejo se podatki o tistih članih razvojne skupine, ki niso odgovorili na vsa tri vprašanja. Poleg podatkov o vsakem članu naj bo okence, s katerim označimo, da je treba temu članu poslati e-mail z opozorilom. Na koncu naj bo gumb, s katerim se sproži pošiljanje opozoril po elektronski pošti.

- Preveri, da seznam vsebuje vse člane, ki v izbranem časovnem obdobju niso odgovorili na nobeno izmed treh vprašanj.

- Preveri, da seznam vsebuje vse člane, ki v izbranem časovnem obdobju niso odgovorili na vsaj eno izmed treh vprašanj.

- Preveri pošiljanje elektronskih sporočil enemu ali več članom hkrati.

Administrator ima možnost pregledovanja dekompozicije uporabniških zgodb na naloge (tasks). Na začetku je treba izbrati sprint, za katerega želimo poročilo, in projekte, ki jih želimo preverjati. Za vsak projekt se izpišejo samo podatki o uporabniških zgodbah, ki nimajo še določenih nalog ali pa za nekatere naloge še ni ocenjena časovna zahtevnost. Za vsak projekt naj bo na voljo okence, s katerim označimo, da je treba vsem članom skupine, ki delajo na tem projektu, poslati sporočilo (e-mail) z ustreznim opozorilom. Na koncu naj bo gumb, s katerim administrator sproži pošiljanje sporočil.

- Preveri pravilnost seznama projektov, v katerih člani niso opravili dekompozicije uporabniških zgodb na naloge (tasks).

- Preveri pravilnost seznama članov razvojne skupine za projekt, v katerem člani niso opravili dekompozicije uporabniških zgodb na naloge (tasks).
- Preveri pravilnost seznama projektov, v katerih člani niso opravili ocenjevanja časovne zahtevnosti ene ali več nalog.

3.2.3 Administracija uporabnikov in projektov

Administrator lahko iz študijskega informacijskega sistema uvozi seznam uporabnikov sistema, ki jih želi dodati v sistem. Seznam je v CSV obliki in med drugim vsebuje vpisno številko, priimek in ime in e-mail naslov. Privzeta vrednost za uporabniško ime naj bo naslov elektronske pošte, za geslo pa vpisna številka. Pri ponovnem uvozu se lahko seznam briše ali samo dopolni. Pri dopolnjevanju naj se dodajo samo novi uporabniki in brišejo tisti, ki jih ni več v novem seznamu. Uvoz naj se izvaja za vsak predmet posebej.

- Preveri za prvi uvoz.
- Preveri za ponovni uvoz študentov pri istem predmetu.
- Preveri za ponovni uvoz študentov pri drugem predmetu.

Administrator lahko izpiše seznam vseh uporabnikov sistema. Za vsakega uporabnika se izpišejo vsi njegovi podatki. Izpis naj bo urejen po predmetih.

- Preveri ujemanje s seznamom iz študijskega informacijskega sistema.

Administrator lahko kreira nov predmet. Za vsak predmet je potrebno definirati, kako velike skupine lahko študenti oblikujejo. Administrator določi rok za prijavo skupine. Vsak predmet ima lahko več terminov za laboratorijske vaje. V istem terminu lahko vaje obiskuje omejeno število skupin.

- Preveri dodajanje in urejanje predmetov.
- Preveri dodajanje in urejanje terminov za laboratorijske vaje.

Študent lahko prijavi skupino, ki bo delala na projektu. Ob prijavi navede toliko članov, kot jih mora vsebovati skupina. Za vsakega člana mora vpisati priimek in ime, ostali podatki pa se avtomatsko črpajo iz seznama. Enemu izmed članov dodeli vlogo skrbnika metodologije (Scrum Master). Ob prijavi si študent izbere projekt, na katerem želi skupina delati, ter termin vaj, na katere bo hodila.

- Preveri, da je zagotovljena konsistentnost podatkov (izbira je možna samo izmed študentov, ki so potencialni uporabniki in niso vključeni v nobeno skupino; skupina mora imeti predpisano število članov; določen mora biti en skrbnik metodologije - Scrum Master)
- Preveri, da študent, ki je že član ene skupine, ne more prijaviti še ene skupine.
- Preveri, da prijava na termin, ki je že polno zaseden, ni možna.
- Preveri možnost prijave skupine v času, ko so prijave odprte.
- Preveri možnost prijave skupine izven časa za prijavo skupine.

Katerikoli član skupine lahko spremeni njeno sestavo. Član skupine lahko enega ali več članov skupine nadomesti z drugimi člani, ki niso še vključeni v neko skupino.

- Preveri, da ima po urejanju skupina predpisano število članov.
- Preveri, da je možno samo dodajanje članov, ki niso v neki drugi skupini.
- Preveri, da je po urejanju vedno določen en skrbnik metodologije (Scrum Master).

Administrator lahko multiplicira nek projekt in ga dodeli drugim skupinam.

- Preveri, ali je dodeljen projekt res enak izvirnemu.
- Preveri, ali je bil projekt dodeljen vsem članom skupine.
- Preveri, da je po urejanju še vedno določen skrbnik metodologije (Scrum Master).

Administrator lahko naknadno doda zgodbo v projekt in zgodbo multiplicira vsem drugim skupinam.

- Preveri, ali je naknadno dodana zgodba res multiplicirana v vseh potrebnih projektih.

3.3 Predstavitev tehnične zasnove

3.3.1 Predstavitev uporabljenih tehnologij

Vse tehnologije in knjižnice, ki so bile uporabljene za izdelavo orodja, so odprtokodne, kar pomeni da ni posebnih omejitev pri njihovi uporabi. Večina tehnologij je bila uporabljena že v obstoječem orodju. Orodje je izdelano kot spletna aplikacija, zato lahko ločimo med tehnologijami, uporabljenimi na strani strežnika, in tehnologijami, uporabljenimi na strani odjemalca.

Glavna programska logika na strani strežnika je napisana v skriptnem jeziku PHP [13], ki je med spletnimi aplikacijami najbolj razširjen [12]. Za shranjevanje in posredovanje podatkov sem uporabil relacijsko podatkovno bazo MySQL [14]. PHP in MySQL sta sicer ena izmed najbolj pogostih uporabljenih kombinacij pri razvoju prostega programja. Kljub temu, da je MySQL odprtokodna podatkovna baza, ima dovolj funkcionalnosti za realizacijo večine projektov.

Na strani odjemalca se koda izvaja v brskalniku. Večina kode je tako napisane v programskem jeziku JavaScript [18]. Tehnologija AJAX [19] se

uporablja za zahtevke, kjer ni potrebno, da se osveži cela stran, ampak samo njen del. Omogoča tudi večjo odzivnost aplikacije, saj uporabniku ni potrebno vedno čakati na odgovor strežnika, ampak lahko že prej nadaljuje z delom. Za oblikovanje se uporablja tehnologija CSS [21]. Vsebina strani se prenaša v obliki HTML [22].

Poleg naštetih tehnologij se za lažjo izdelavo aplikacije uporabljajo različne knjižnice.

- **jQuery:** jQuery [15] je JavaScript knjižnica, ki poenostavi pisanje kode na strani odjemalca. Z njo lahko na enostaven način spreminjamo vsebino (HTML), stil (CSS) in programsko logiko. Poenostavi pisanje zank, dodajanje dogodkov na elemente, iskanje elementov in tudi pošiljanje in obdelavo AJAX zahtevkov. Do neke mere tudi uskladi delovanje aplikacije na različnih brskalnikih.

- **jQueryUI:** jQueryUI [16] je knjižnica za lažjo izdelavo uporabniškega vmesnika. Je kombinacija knjižnice JavaScript, CSS datotek in grafičnih elementov. Pogoj za njeno uporabo je uporaba knjižnice jQuery. S knjižnico jQueryUI je preprosto na stran dodati nekatere funkcije, na primer dialoge, gumbe, koledar, zavihke, predloge vnosov (autocomplete) in druge. Vse naštete sem tudi sam uporabil pri izdelavi orodja. Knjižnica omogoča tudi večjo interaktivnost, na primer sortiranje elementov grafičnega vmesnika in funkcionalnost povleci in spusti, ki sta v tem orodju nepogrešljivi pri delu s tablo.

- **pChart:** pChart [17] je PHP knjižnica za izdelavo grafikonov in diagramov. Z njo so v orodju realizirani grafikoni"Burndown", grafi za analizo dela članov razvojne skupine in drugi grafikoni.

- **PHPMailer:** PHP knjižnica PHPMailer [20] je namenjena pošiljanju elektronskih sporočil. V svoji aplikaciji sem jo uporabil za pošiljanje obvestil članom razvojnih skupin v primerih, ko se le-ti ne udeležujejo vsakodnevnih sestankov ali če svojih zgodb ne razdelijo na naloge, kot to zahteva Scrum.

3.3.2 Predstavitev podatkovnega modela

Izhodišče za podatkovni model je bil model, izdelan v okviru diplomske naloge [7]. Ta je vseboval 10 entitetnih tipov in dve povezovalni entiteti. Na podlagi tega modela, je podatkovna baza vsebovala 12 tabel: member, member_project, poker_estimate, poker_round, project, sprint, story, story_time, task, wall, wall_comment in work.

Orodje se je v naslednjih letih uporabljalo v študijskem procesu, medtem pa se je orodje dopolnjevalo. Dodana je bila podpora za vsakodnevne sestanke in podpora za nov način ocenjevanja zgodb (team estimation game). Podatkovni model je bil dopolnjen tako, da je bilo dodanih še šest tabel: daily_scrum, daily_scrum_item, team_estimation_log, team_estimation_member, team_estimation_round in settings.

Preden sem začel z delom, je torej podatkovna baza obsegala 18 tabel. Za potrebe realizacije table je bilo v podatkovni model potrebno dodati naslednje tabele:

- **Stolpec (columns):** na tabli predstavlja neko fazo oziramo stopnjo v procesu razvoja programske opreme. Poleg imena [column_name] ima tudi druge atribute. Za potrebe omejevanja sočasnega dela (WIP limit) ima atributa maksimalno število kartic v stolpcu [max_cards] in maksimalno število točk v stolpcu [max_velocity]. Za pravilen izris table je za vsak stolpec potrebno definirati, na katerem mestu v tabeli stoji [index]. Ker ima lahko stolpec tudi podstolpce, je potrebno zabeležiti, kateri stolpec je njegov oče [column_parent_id]. Za lažji izris tabele vodim tudi evidenco o nivoju stolpca [level], ki pomeni število njegovih "prednikov". Atribut [editable] pomaga najti stolpce, ki jih ni mogoče spreminjati v polnem obsegu. To so stolpci ki predstavljajo seznam nalog projekta(Product Backlog), seznam nalog sprinta (Sprint Backlog) in seznam dokončanih zgodb (Done). Ti stolpci imajo vrednost tega atributa enako 0, ostali pa 1. Vsaka tabla pripada nekemu sprintu in projektu. Podatka o tem se shranita v atributih [sprint_id] in [project_id].

- **Tabla (board):** V tej tabeli so shranjeni podatki o karticah na tabli. Kartica predstavlja zgodbo [story_id], ki je v nekem stolpcu [column_id]. Za lažje iskanje imam v tej tabeli tudi podatek o sprintu [sprint_id], ki sicer ne bi bil potreben, saj ga lahko pridobimo tudi posredno preko stolpca.

- **Zgodovina table (board_history):** Tu se za vsak projekt [project_id] beležijo dogodki [event] v povezavi s tablo, na primer vsi premiki, dodajanje zgodb v sprint, dodajanje novih zgodb v zbirko nalog (Product Bakclog) in podobni dogodki. Zapiše se tudi čas [date] in uporabnik, ki je dogodek sprožil [user_id]. Dogodek se beleži v obliki teksta, kar je največja pomanjkljivost te tabele. Iz nje je težko izdelati analize, saj dogodki niso dovolj strukturirani. Kot nadgradnja te tabele je bila izdelana tabela za beleženje zgodovine premikov.

- **Zgodovina premikov (move_history):** Namenjena je beleženju premikov kartic po tabli. V primerjavi z Zgodovino table je bolj strukturirana, saj posebej beleži atribute izvor in cilj kartice [from_column, to_column], beleži oznako zgodbe [story_id] in tudi vrsto akcije [action]. Ostali atributi so podobni [date, user_id, project_id, sprint_id]. Strukturiranost je tu večja, a beleži manj dogodkov kot tabela Zgodovina table, zato je le-ta ostala v modelu.

- **Dovoljenja za premik (move_permissions):** V tej tabeli se hranijo pravice za premik kartic iz enega stolpca v drug stolpec [column_id_from, column_id_to]. Pravice se lahko dodeljujejo glede na vlogo v projektu [role_id] (1=Scrum Master, 2=Product Owner, 0=Team member).

- **Kršitve omejitve WIP (wip_limit_violation):** Kot že večkrat omenjeno, WIP limit pomeni omejitev količine dela v neki fazi razvoja, torej v stolpcu. Vsak stolpec ima lahko definirano omejitev. V primeru kršitve se ta zabeleži v to tabelo. Atribut tip [type] beleži tip kršitve (kršitev števila kartic ali kršitev po številu točk). Beleži se

tudi datum, uporabnik, zgodba in razlog, zakaj je do kršitve prišlo [date, user, story_id, reason]. Beležijo se tudi podatki o dveh stolpcih. Eden od njih označuje stolpec kamor je bila kartica premaknjena [column_id], drugi pa beleži stolpec, kjer je bila omejitev kršena [column_id_violated]. Običajno sicer gre za isti stolpec, a v primeru podstolpcev to ne drži vedno.

Drugi del novega podatkovnega modela obsega tabele in entitetne tipe za lažjo administracijo uporabnikov. To se predvsem nanaša na lažje oblikovanje razvojnih skupin za potrebe študija.

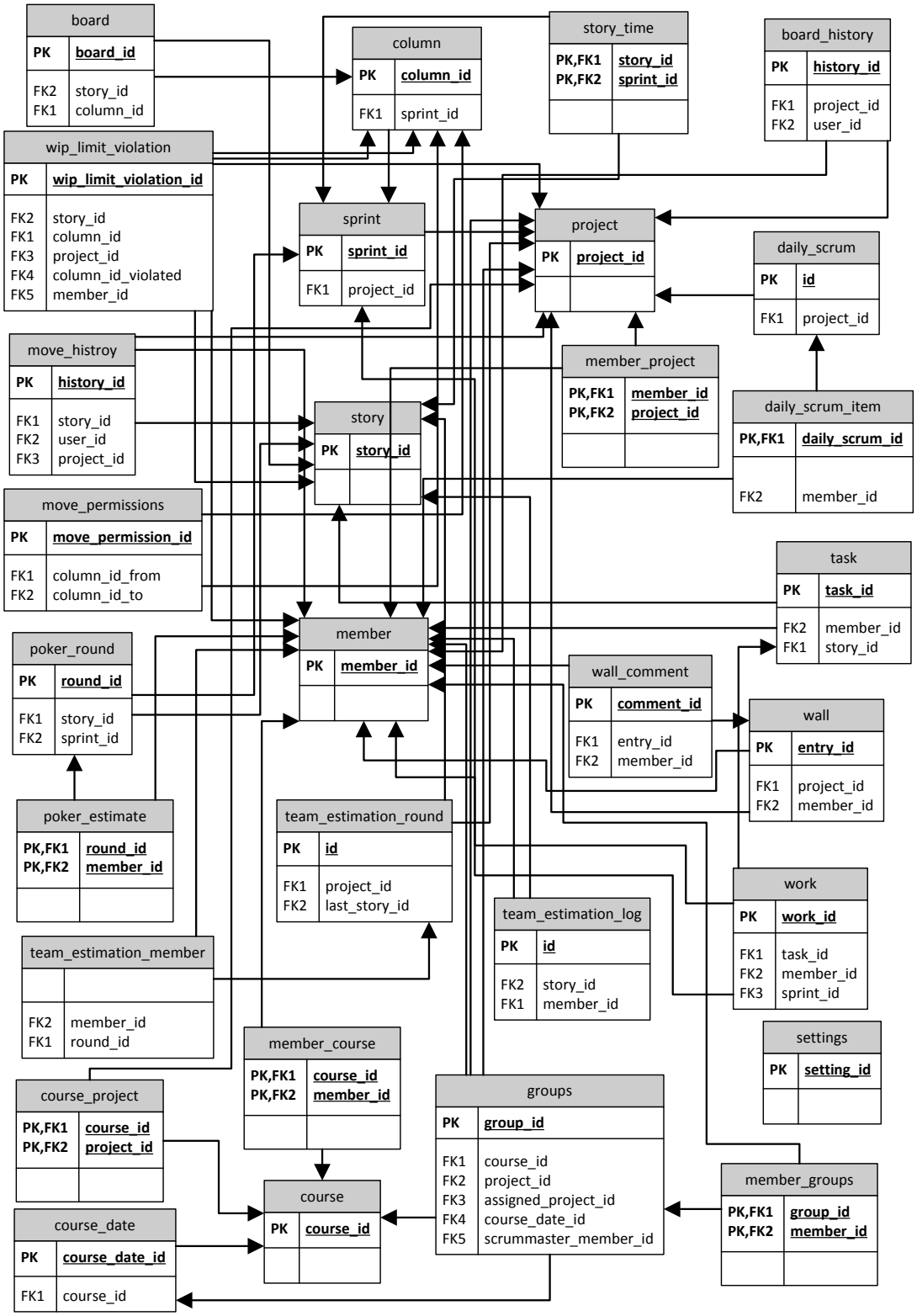
- **Predmet (course):** Orodje se lahko uporablja v študijskem procesu pri več predmetih. Za vsak predmet lahko določimo, kdaj lahko udeleženci prijavijo skupine in kako velike naj bodo [register_start, register_end, max_group_members].

- **Termini za laboratorijske vaje (course_date):** Običajno je pri vsakem predmetu [course_id] več terminov za vaje. Na vsak termin vaj [date] se lahko prijavi omejeno število skupin [max_groups].

- **Udeleženci predmeta (member_course):** Je povezovalna tabela, ki shranjuje podatke o udeležencih predmeta [member_id, course_id].

- **Projekti predmeta (course_project):** Še ena povezovalna tabela [course_id, project_id]. V tej tabeli so za vsak predmet našteti projekti, med katerimi lahko skupine izbirajo pri prijavi skupine. Na tem projektu delajo v nadaljevanju študijskega procesa.

- **Skupina (groups):** Tabela za predstavitev skupin. Za vsako skupino, poleg imena [group_name], vodimo evidenco izbranega projekta, na katerem bodo v nadaljevanju delali študenti [project_id]. Atribut [course_date_id] predstavlja izbran termin laboratorijskih vaj, [scrummaster_member_id] predstavlja člana skupine, ki bo na projektu dobil vlogo Scrum Master. Ko skupina dobi svojo instanco projekta, se podatek o tem hrani v atributu [assigned_project_id].



Slika 3.1: Predstavitev podatkovnega modela.

- **Člani skupine (member_groups):** Povezovalna tabela za shranjevanje podatkov o članih skupine [gorup_id, member_id].

Podatkovni model je prikazan na Sliki 3.1. Zaradi prostorske stiske na tem modelu ni navedenih nekaterih atributov. Poleg imen tabel so prikazani le še primarni in tuji ključi. Vsi atributi, skupaj s podatkovnimi tipi, so predstavljeni v prilogi A.

3.3.3 Predstavitev programske logike

Sama programska logika se ni bistveno spremenila v primerjavi z obstoječim orodjem [7], saj gre le za nadgradnjo orodja.

Glavna dostopna točka programa ostaja datoteka index.php. Program ima še dve dostopni točki, in sicer formsAjax.php in scriptsAjax.php. Kot že ime pove, sta namenjeni predvsem ajax zahtevkom. Glavne strani se nalagajo preko index.php, saj se preko nje nalaga večina JavaScript kode in večina uporabniškega vmesnika. Do ostalih datotek je možno dostopati le posredno preko teh treh dostopnih točk.

Jedro programske logike je bilo v izhodišču v treh razredih (forms, common in html) in dveh objektih (Mdb, Mform), sedaj pa sem dodal še četrti razred z imenom board. Razred forms je po novem razširitev razreda board. Nov je tudi objekt PHPMailer, ki je del istoimenske knjižnice.

Razred board je, podobno kot razred forms, glavno jedro orodja, saj preko njega poteka večina interakcije z uporabnikom. Njuna funkcija je enaka, ločena sta z namenom, da se prvotna izvorna koda in koda za nadgradnjo orodja ne mešata. Večina novih funkcionalnosti je torej napisanih v razredu board, v razredu forms pa so bile narejene le manjše spremembe. Z večino metod v razredu board so realizirani obrazci, nekaj pa jih je namenjenih za drugo programsko logiko. Sledi seznam metod in njihov kratek opis; seznam je razvrščen po abecedi.

- **addNewCourseDate:** obrazec za dodajanje in urejanje terminov laboratorijskih vaj.

- **boardActions:** metoda izvaja akcije na tabli, kot so sortiranje stolpcev, premikanje kartic in podobno.
- **courses:** obrazec za prikaz in brisanje predmetov.
- **createBoard:** metoda izvede kreiranje table za nek sprint, po potrebi kopira tablo iz prejšnjega sprinta.
- **dailyScrumMeetingReport:** obrazec za analizo opravljenih vsakodnevnih sestankov.
- **decompositionReport:** obrazec za analizo dekompozicije zgodb na naloge in preverjanje, ali so te naloge ocenjene.
- **deleteGroup:** metoda za brisanje skupine.
- **editCourse:** obrazec za urejanje predmetov.
- **editPermissions:** obrazec za urejanje dovoljenj za premikanje zgodb na tabli.
- **groupActions:** akcije za podporo pri kreiranju in urejanju skupin.
- **groups:** obrazec za pregled skupin in koda za multipliciranje projektov tem skupinam.
- **history:** obrazec za pregled zgodovine premikov na tabli.
- **movingChecks:** akcije, ki preverjajo dovoljenja in pravila za premike kartice na tabli.
- **newBoard:** obrazec za urejanje table.
- **newCourse:** obrazec za dodajanje predmetov.
- **newGroup:** obrazec za dodajanje skupin.
- **ruleViolation:** obrazec, kamor vpišemo razlog za kršitev WIP.

- **sendEmail:** koda za pošiljanje elektronskih sporočil.
- **storyDetails:** obrazec za prikaz uporabniške zgodbe.
- **updateDerivatives:** koda za multipliciranje naknadno dodane uporabniške zgodbe v druge projekte.

Poleg razreda board pa imamo še druge razrede. **Razred html** vsebuje funkcije, ki nam olajšajo izpis nekaterih HTML elementov, kot so na primer uporabniške zgodbe, kartice, tabla in drugo. V okviru te diplomske naloge so bile dodane naslednje funkcije:

- **course_dates:** funkcija izriše del obrazca pri urejanju predmeta. Ta del se navezuje na laboratorijske vaje.
- **groups:** funkcija izriše del obrazca na seznamu skupin. Klic te funkcije vrne seznam skupin za izbran predmet.
- **printCards:** funkcija izriše vse kartice za nek stolpec.
- **printTable:** funkcija izriše tablo.

Razred common vsebuje splošne funkcije, ki so potrebne na več mestih v aplikaciji. Dodane so bile naslednje funkcije:

- **getColumnData:** funkcija prebere iz baze podatke posameznega stolpca na tabli.
- **create_column_tree:** funkcija organizira stolpce v drevesno strukturo.
- **copyChildColumns:** s pomočjo te funkcije se izvede prenos strukture table v naslednji sprint.
- **getLastCustomColumn:** funkcija na tabli najde zadnji stolpec pred stolpcem “Done”.
- **in_multiarray:** funkcija za iskanje elementa v seznamih.

- **checkViolationRecursive:** funkcija za iskanje kršitev WIP v nadstolpcih.
- **getCardNumAndPtsOfColumn:** rekurzivna funkcija vrne vsoto vseh točk na karticah in število kartic v vseh podstolpcih.

Poglavje 4

Predstavitev uporabe orodja

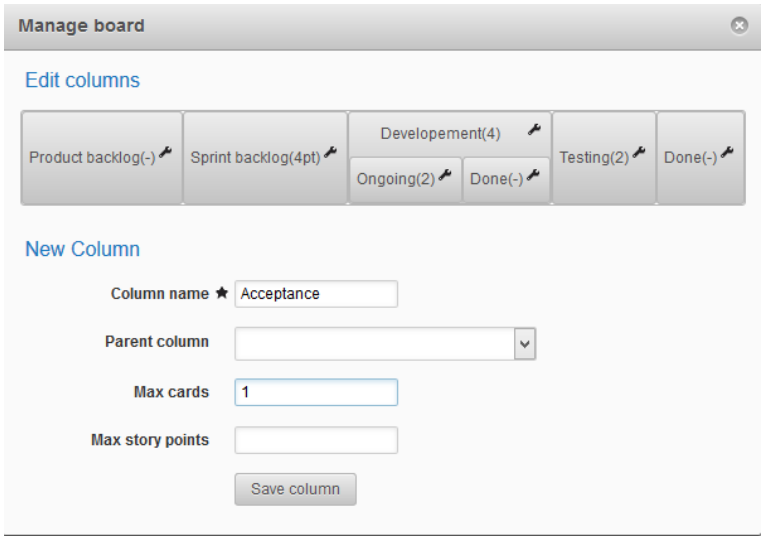
V tem poglavju so predstavljene nadgradnje orodja in njegova uporaba. Poglavje je razdeljeno na tri večje sklope nadgradenj.

4.1 Vizualizacija delovnega toka - tabla

4.1.1 Kreiranje table

Preden lahko kreiramo tablo, je potrebno vzpostaviti projekt, dodati uporabniške zgodbe in projekt razdeliti na več sprintov. Tablo je potrebno kreirati za vsak sprint posebej. To lahko naredi skrbnik metodologije (Scrum Master). V času sprinta se na strani za urejanje seznama zahtev (Product Backlog) pojavi gumb za kreiranje table (New board). S klikom na ta gumb se generira tabla, pojavi pa se okno za urejanje table (glej Sliko 4.1). Privzeto ima tabla tri fiksne stolpce (Product backlog, Sprint backlog, Done). Med zadnja dva stolpca je možno dodati poljubno število stolpcev in podstolpcev. Podstolpec dodamo tako, da si pri dodajanju iz seznama izberemo njegovega očeta (Parent column). Stolpcem na istem nivoju je možno spremeniti vrstni red.

Ob kreiranju stolpca lahko določimo tudi omejitev obsega dela v izvajanju, in sicer na dva načina. Omejimo lahko število kartic v stolpcu, lahko pa tudi omejimo skupno število točk (Story points). Ena točka običajno



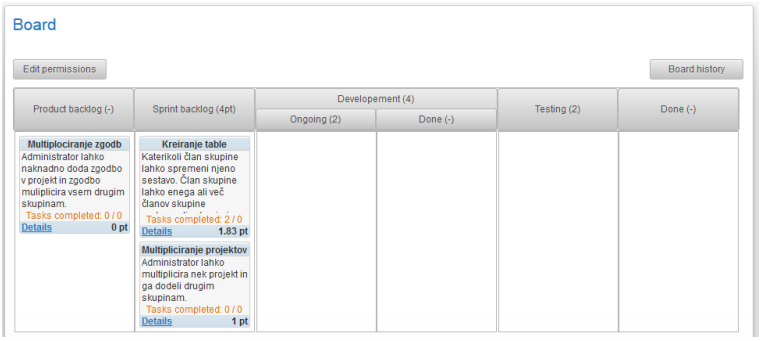
Slika 4.1: Okno za dodajanje in urejanje stolpcev na tabli.

predstavlja 6 ur efektivnega dela. Kot je razvidno iz Slike 4.1 je omejitev napisana v oklepajih ob imenu stolpca.

Urejanje table je možno tudi med samim sprintom, a ni zaželeno. Dostop do urejanja je prav tako možen preko strani za urejanje seznama zahtev (Product Backlog). Na istem mestu, kot gumb za kreiranje table, se bo nahajal gumb za urejanje table (Edit board). Dostop do table s karticami je možen s klikom na gumb Tabla (Board) v glavnem meniju. Na tabli so prisotne vse kartice z uporabniškimi zgodbami, ki jih je potrebno realizirati v okviru izbranega projekta. Tabla je prikazana na Sliki 4.2. V vsakem naslednjem sprintu struktura table ostane enaka, torej vsi stolpci in podstolpci se ohranijo. V primeru, da imamo v sprintu kartico, ki v prejšnjem sprintu ni bila potrjena ali zavrnjena, potem ta kartica dobi rdečo obrobo. Tako kartico je potrebno čimprej zavrniti ali sprejeti.

4.1.2 Premiki kartic

Premik kartice v drug stolpec realiziramo tako, da kartico primemo, jo prenesemo in spustimo v drug stolpec (Drag and drop). Vendar pa je za ta



Slika 4.2: Prikaz table s karticami.

korak potrebno imeti dovoljenje. Dovoljenja so realizirana na podlagi vlog. Sledi seznam privzetih dovoljenj, ki so enaka na vseh tablah in jih ni moč spreminjati.

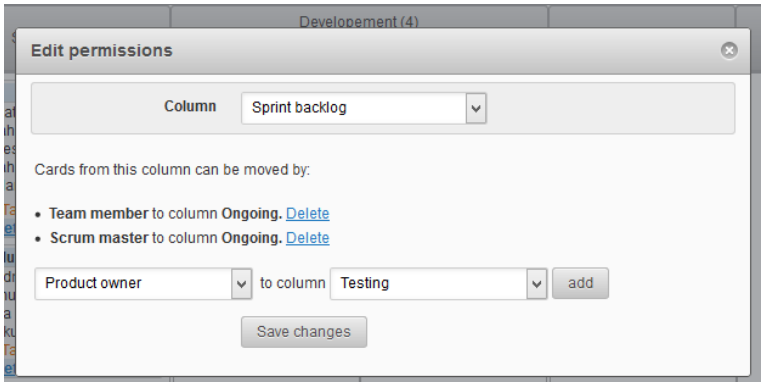
- Samo skrbnik metodologije lahko kartico iz stolpca Product backlog premakne v stolpec Sprint Backlog. Premik te vrste pomeni dodajanje zgodbe v sprint.

- Samo produktni vodja lahko premakne kartico iz predzadnjega stolpca v zadnji stolpec (Done), saj lahko le on sprejme neko zgodbo.

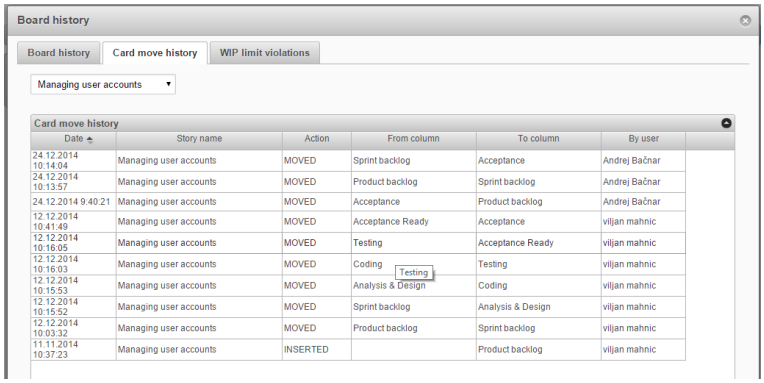
- Iz zadnjega stolpca (stolpca Done) ni možno premikati kartic.

- Samo produktni vodja lahko premika kartice v stolpec Product backlog iz vseh ostalih stolpcev (razen Done). Premik te vrste pomeni zavračanje zgodbe. Naslednje pravilo je izjema tega pravila.

- Skrbnik metodologije lahko premakne kartico iz Sprint backloga v Product Backlog. Ob tem mora veljati, da je bila zgodba v istem dnevu dodana v sprint, ter da na tej zgodbi še ni bilo opravljeno delo. Razlog za to dovoljenje je v tem, da lahko skrbnik metodologije med sestankom za planiranje sprinta zgodbo izloči iz sprinta, čeprav je zgodba že bila dodana v sprint.



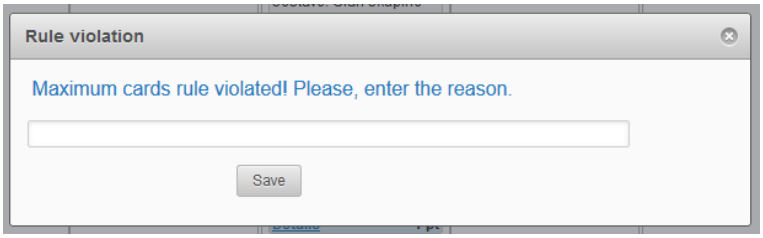
Slika 4.3: Dodajanje dovoljenj za premike.



Slika 4.4: Zgodovina premikov na tabli za izbrano zgodbo.

Ta pravila za premike izhajajo iz pravil metodologije Scrum. Za druge ročno dodane stolpce pa lahko pravila dodamo sami. Za dostop do obrazca za urejanje dovoljenj za premike je potrebno klikniti na gumb “Edit permissions” na tabli. V pojavnem oknu najprej iz seznama izberemo stolpec iz katerega bomo kartico premikali. Nato je potrebno izbrati vlogo, za katero želimo dodati pravilo, in stolpec, kamor želimo kartico premakniti. Ob koncu vsa dodana pravila še shranimo (glej Sliko 4.3).

Vsak premik kartice se beleži v arhivu, do katerega pridemo s klikom na gumb “Board history” na tabli. Premike je moč filtrirati za vsako zgodbo posebej. Seznam je prikazen na Sliki 4.4.



Slika 4.5: Opozorilo ob kršitvi omejitve dela v izvajanju.

4.1.3 Omejevanje obsega dela v izvajanju

Tabla omogoča omejevanje obsega dela v izvajanju (Work in Progress - WIP) za vsako fazo (stolpec) v razvoju programske opreme posebej. Dodajanje omejitve je opisano v razdelku 4.1.1.

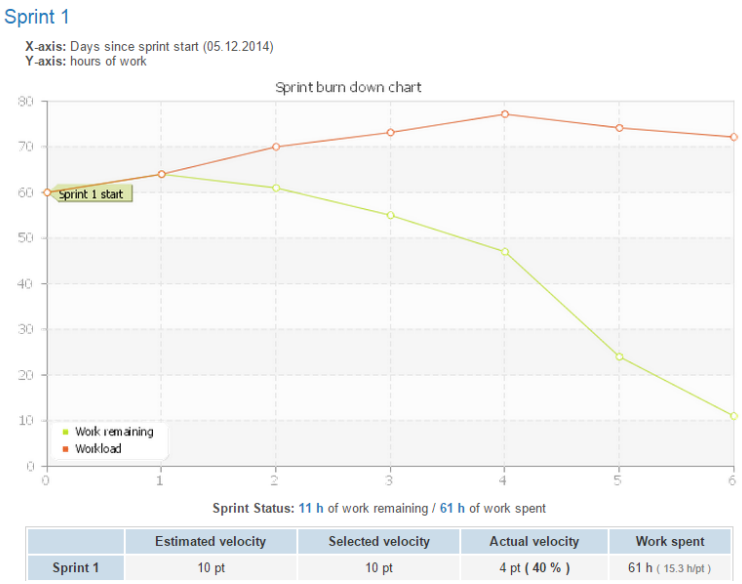
Ob premikih se vedno preveri, ali je bila v nekem stolpcu kršena omejitev. V primeru, da pride do kršitve, nas orodje opozori, za kakšno kršitev gre. Za nadaljevanje je potrebno vpisati razlog kršitve (Slika 4.5), v nasprotnem primeru orodje premika ne dovoli.

Seznam vseh kršitev se beleži v arhivu, do katerega pridemo s klikom na gumb "Board history" na tabli. Seznam je možno tudi filtrirati po več kriterijih.

4.2 Nadzor nad potekom projektov

4.2.1 Grafikon "Sprint burndown chart"

Za lažje spremljanje in analizo projektov je bil izdelan grafikon "Sprint Burndown chart", kjer je tabelarično in grafično prikazana primerjava vloženega dela s preostalim delom v sprintu. Grafikon je bil izdelan s pomočjo knjižnice pChart. Za prikaz grafikona je v glavnem meniju potrebno klikniti na gumb "Progress reports" in izbrati zavihek "Sprint Burndown". Os X prikazuje število dni od začetka sprinta, os Y pa število ur. Zelena črta predstavlja krivuljo preostalega dela v tem sprintu, medtem ko rdeča črta predstavlja vsoto



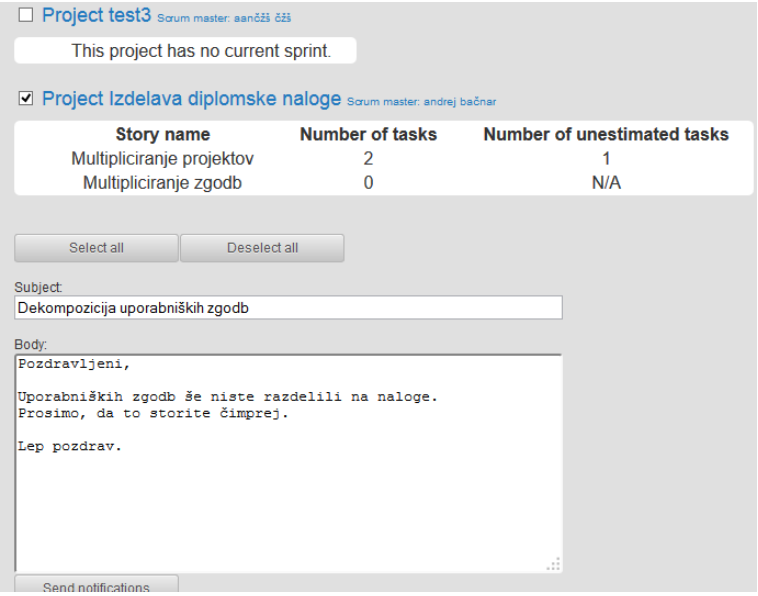
Slika 4.6: Grafikon "Sprint Burndown chart" prikazuje primerjavo vloženega dela s preostalim delom v sprintu.

že opravljenega dela in dela, ki ga je še potrebno opraviti. Razdalja med črtama v smeri Y torej predstavlja opravljeno delo v sprintu (glej Sliko 4.6).

Pod grafikonom se nahajajo tudi drugi statistični podatki, ki jih je dobro upoštevati predvsem pri planiranju naslednjih sprintov.

4.2.2 Poročilo o dekompoziciji uporabniških zgodb

Orodje se uporablja v študijskem procesu, zato je pomembno vedeti, kako dosledno študenti izvajajo prakse Scruma. Eno izmed pravil Scruma pravi, da je potrebno uporabniške zgodbe razdeliti na več nalog in za vsako nalogo oceniti obseg potrebnga dela. Dodana je bila funkcionalnost, ki izdela poročilo o doslednosti izvajanja tega pravila. Poročilo izdelamo tako, da gremo v nadzorno ploščo (Control panel) pod zavihek poročila (Reports). Tam si izberemo projekte, za katere naj se analiza izvede. Klik na "Decomposition report 2" odpre novo okno s poročilom, v katerem so navedene vse razvojne skupine, ki dekompozicije oziroma ocenjevanja nalog niso opravile .



Slika 4.7: Uporabniški vmesnik za pošiljanje sporočila skupinam, ki niso opravile dekompozicije uporabniških zgodb na naloge, oziroma nalog niso ocenile.

Administrator lahko preko elektronske pošte pošlje sporočilo tem skupinam in jih opozori. Vsebino elektronskega sporočila lahko prosto spreminja. Uporabniški vmesnik za pošiljanje sporočila skupinam je prikazan na Sliki 4.7.

4.2.3 Poročilo o udeležbi na vsakodnevnih sestankih

Poročilo o udeležbi na vsakodnevnih sestankih (Daily Scrum meeting report) nam prav tako služi za preverjanje doslednosti pravil Scruma. S pomočjo tega poročila preverimo, ali so se vsi člani razvojne skupine udeležili sestanka, ki poteka prek tega orodja. Vsak član mora zabeležiti odgovore na tri vprašanja opisana v poglavju 2.2. Podobno kot pri poročilu o dekompoziciji zgodb, tudi tu poročilo izdelamo tako, da gremo v nadzorno ploščo (Control panel) pod zavihek poročila (Reports). Tam si izberemo projekte, za katere naj se analiza izvede. Vpisati je potrebno še časovni interval, v katerem bi moral biti

sestanek izpeljan. S klikom na povezavo "GO" sprožimo analizo in v novem oknu se pojavi seznam skupin in njenih članov, ki sestanka niso opravili. Administrator lahko tudi tukaj pošlje elektronsko sporočilo članom, ki niso odgovorili na vsa vprašanja.

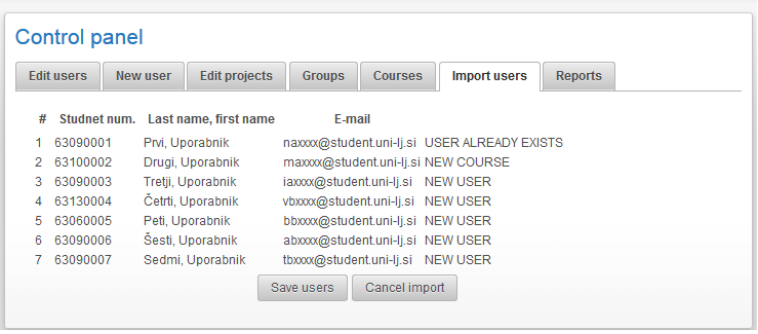
4.3 Orodja za administracijo

4.3.1 Uvoz uporabnikov

Ob začetku študijskega leta je potrebno v orodje vnesti vse študente, ki bodo orodje uporabljali. V orodje je bila dodana funkcionalnost, ki preko datoteke CSV uvozi seznam študentov v podatkovno bazo. Datoteko CSV lahko izvozimo iz študijskega informacijskega sistema. Datoteka mora biti kodirana v načinu UTF-8. Pred uvozom si moramo izbrati še predmet, ki ga študenti poslušajo. V primeru, da študenta ob uvozu ni v datoteki CSV, a je že vnesen v orodje, potem bo le-tega izbrisalo ali odjavilo od izbranega predmeta.

Uvoz poteka v naslednjem zaporedju. Najprej se seznam študentov iz datoteke prenese v sejo. Ob tem se za vsakega uporabnika določi akcija, ki se bo izvedla ob potrditvi uvoza. V nadaljevanju sledi predstavitev teh akcij.

- **NEW USER:** uporabnika še ni v sistemu, dodan bo nov uporabnik.
- **USER ALREADY EXISTS:** uporabnik je že v sistemu in prijavljen na izbran predmet. Izvedla se ne bo nobena akcija.
- **NEW COURSE:** uporabnik je že v sistemu, ampak ni prijavljen na izbran predmet. Uporabnik bo prijavljen na predmet.
- **DELETE USER:** uporabnik je v sistemu in prijavljen le na izbran predmet. V novem seznamu za uvoz ga ni. V tem primeru bo uporabnik izbrisan iz orodja.
- **REMOVE FROM COURSE:** uporabnik je v sistemu in prijavljen na izbran predmet in še na nek drug predmet. V novem seznamu



Slika 4.8: Seznam študentov za uvoz v orodje.

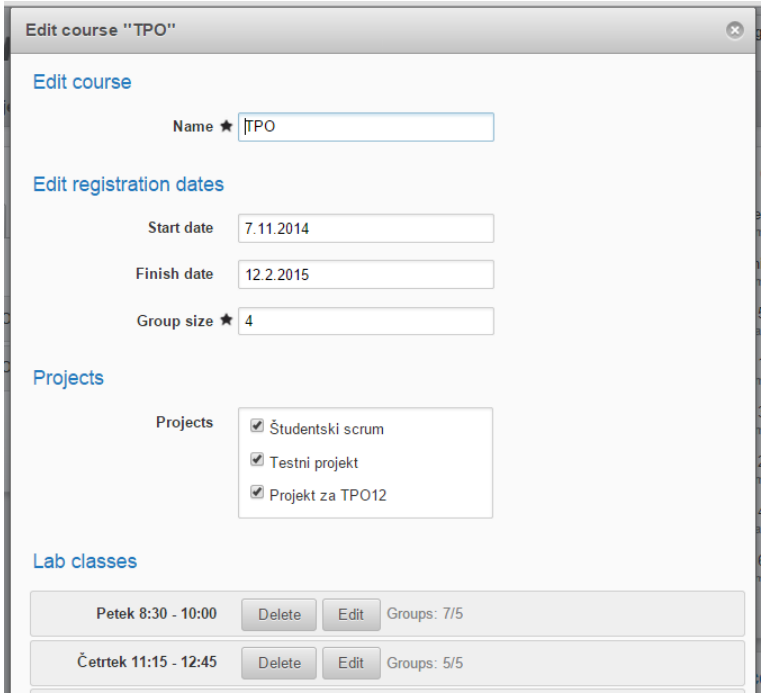
za uvoz ga ni. V tem primeru bo uporabnik odjavljen od izbranega predmeta.

Za vse udeležence predmeta se bo izpisala ena od zgornjih akcij, kot je prikazano na Sliki 4.8. Če smo s seznamom zadovoljni, uvoz potrdimo, v nasprotnem primeru pa ga lahko prekličemo. Ob potrditvi se dodeljene akcije izvedejo.

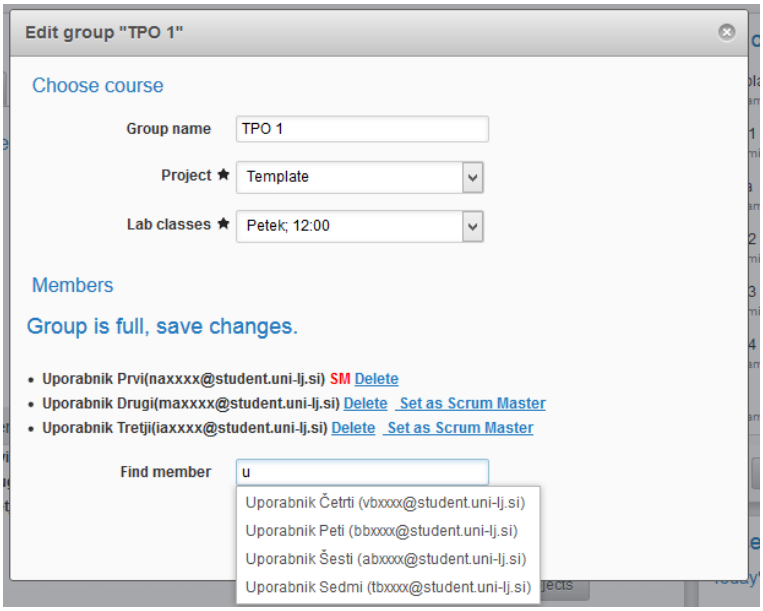
Uporabniško ime novih uporabnikov je enako njihovemu elektronskemu naslovu, privzeto geslo pa je vpisna številka.

4.3.2 Administracija predmetov

Ker se orodje uporablja v študijske namene pri več predmetih, je potrebno voditi evidenco le-teh. V nadzorni plošči (Control panel) pod zavihkom predmeti (Courses) se nahaja seznam predmetov, pri katerih se to orodje uporablja. Za vsak predmet lahko definiramo, kako velike skupine lahko študenti prijavijo in kdaj lahko to storijo. Poleg tega mora administrator določiti, med katerimi projekti lahko študenti izbirajo pri prijavi skupine. V nadaljevanju študija namreč vsaka skupina dela na izbranem projektu. Administrator vnese tudi termine laboratorijskih vaj in največje možno število skupin v vsakem terminu. Vse te podatke je možno spreminjati na obrazcu za urejanje predmeta (glej Sliko 4.9).



Slika 4.9: Obrazec za urejanje podatkov o predmetu.

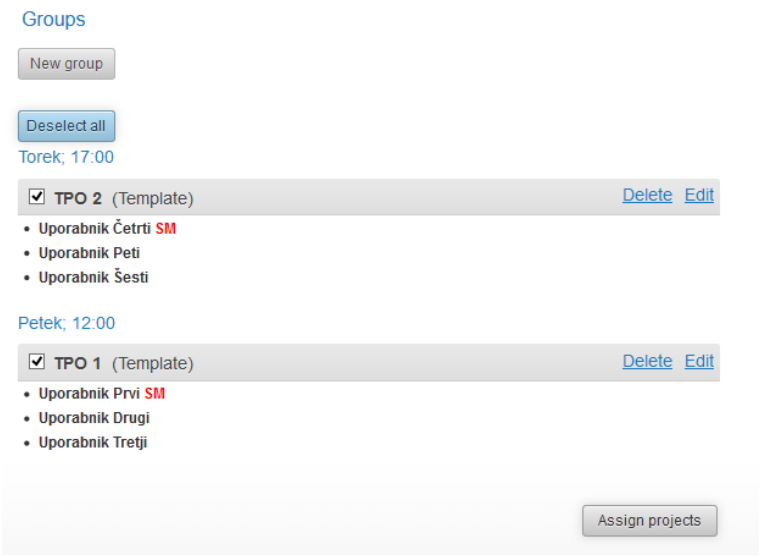


Slika 4.10: Obrazec za kreiranje in urejanje skupin.

4.3.3 Kreiranje skupin

Ko administrator vnese vse podatke o predmetih in terminih vaj, lahko odpre registracijo skupin. Študenti lahko preko nadzorne plošče (Control panel) dostopajo do seznama skupin. V primeru da še niso člani nobene skupine lahko začnejo s prijavo nove skupine s klikom na gumb "New group". Študent, ki skupino prijavlja, samodejno postane član skupine. Privzeto dobi tudi vlogo skrbnika metodologije (Scrum Master), ki jo lahko pozneje prenese na drugega člana. Študent lahko preko imena in priimka poišče druge potencialne člane njegove skupine. Izbira lahko samo med tistimi študenti, ki še niso člani nobene skupine. V skupino mora dodati toliko članov, da je skupina polna. Le administrator ima možnost kreiranja skupine z več ali manj člani od predpisanega števila. Ob kreiranju skupine si mora študent izbrati še termin vaj, ki jih bo skupina obiskovala, in projekt, na katerem želi delati.

Vsi člani skupine imajo možnost urejanja svoje skupine in brisanje le-te, v kolikor skupini še ni bil dodeljen projekt. Dodeljevanje projektov je opisano



Slika 4.11: Gumb "Assign projects" sproži multipliciranje projektov.

v naslednjem razdelku.

4.3.4 Multipliciranje projektov in uporabniških zgodb

Ko so skupine določene se lahko izvede multipliciranje projektov. Pri tem se projekt, ki ga je skupina izbrala ob prijavi, samodejno prekopira in dodeli skupini. Skupaj s projektom se prekopirajo tudi vse uporabniške zgodbe. Za multipliciranje najprej označimo skupine, katerim hočemo projekt dodeliti, in kliknemo na gumb "Assign projects" (glej Sliko).

Ko je projekt skupini dodeljen, skupin ni več mogoče urejati. Če želimo urediti člane skupine ali ji dodeliti drug projekt, jo je potrebno izbrisati in ponovno kreirati.

Poleg multipliciranja projektov je možno izvesti tudi multipliciranje uporabniških zgodb, ki se pojavijo med izvajanjem projekta. Najprej je zgodbo potrebno vnesti v izvoren projekt. Multipliciranje zgodbe sprožimo s klikom na povezavo na dnu obrazca seznama zahtev (Product Backlog).

Poglavje 5

Zaključek

Eden od glavnih namenov te diplomske naloge je bil izboljšati vizualizacijo delovnega toka s pomočjo table in preko nje omejiti delo v izvajanju. S tem smo orodju dodali nekatere elemente, značilne za razvoj po metodologiji Kanban. Prvotna verzija orodja je namreč bolj ali manj sledila le pravilom in principom metodologije Scrum. Za namen preverjanja doslednosti izvajanja pravil Scruma so bile dodane funkcionalnosti, ki kršitve teh pravil najdejo in omogočajo opozarjanje uporabnikov preko elektronske pošte. Orodje se uporablja v študijskem procesu, zato je, zaradi velikega števila študentov, bilo potrebno veliko administracije. Z nadgradnjami se je obseg administrativnih del močno zmanjšal. Del nalog se je prenesel na študente, ki lahko sedaj sami oblikujejo skupine, večino drugih operacij (multipliciranje projektov in uporabniških zgodb, dodajanje študentov v sistem) pa nam olajša orodje.

Z nadgradnjo orodja za vodenje projektov po agilnih metodologijah smo razvijalcem ponudili večji pregled nad projektom z namenom izboljšanja razvojnega procesa, pedagoško osebje pa smo razbremenili nepotrebnega administrativnega dela in mu omogočili večji nadzor nad študenti.

Nadgradnja orodja se je začela uporabljati v študijske namene šele pred nekaj tedni, zato večjih odzivov na nadgradnje še ni.

Prostora za dodatne nadgradnje je sicer še kar nekaj, predvsem v zvezi z vizualizacijo delovnega toka. Na tabli bi lahko uporabniško izkušnjo izboljšali

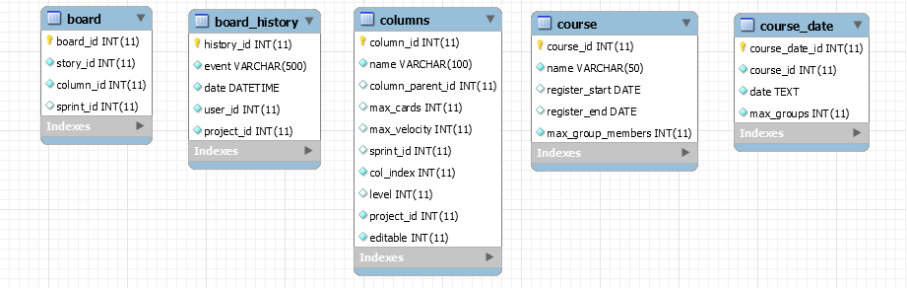
z možnostjo skrivanja stolpcev in/ali z možnostjo spremembe širine stolpca. Poleg tega bi lahko skrbnik metodologije prosto izbiral, kateri podatki naj se prikažejo na kartici. Smiselno bi bilo dodati še funkcionalnost za izračun povprečnega potrebnega časa za izdelavo neke zgodbe (ang. average lead time).

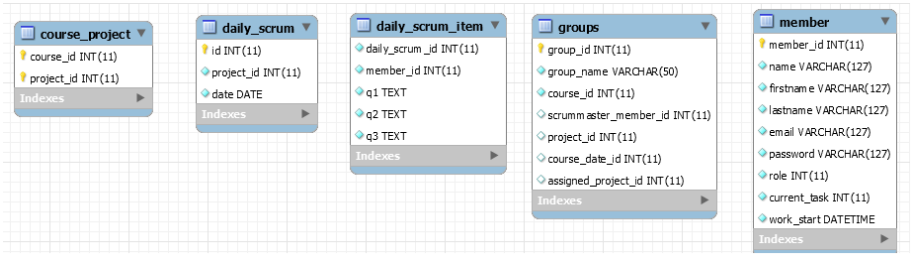
Pri vodenju projektov se je potrebno zavedati, da dobro orodje ni zadosten razlog za uspešen zaključek projekta, lahko pa precej pripomore k temu.

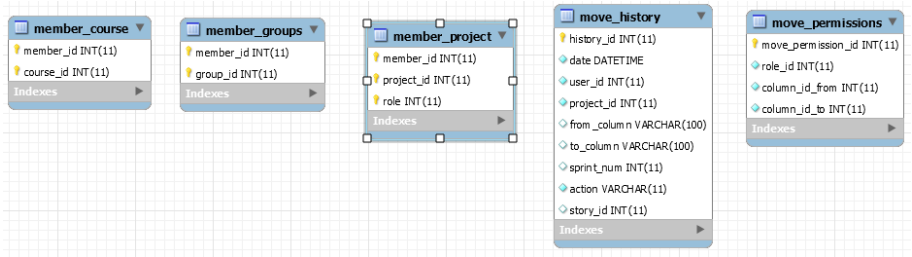
Poglavje 6

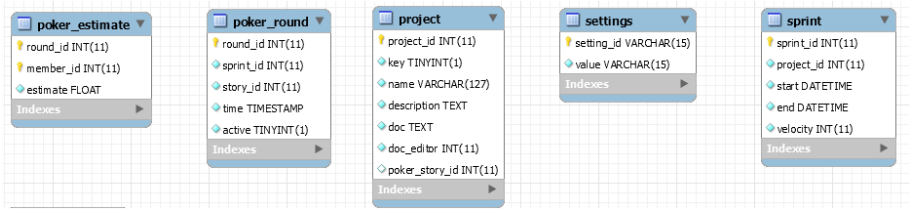
Priloge

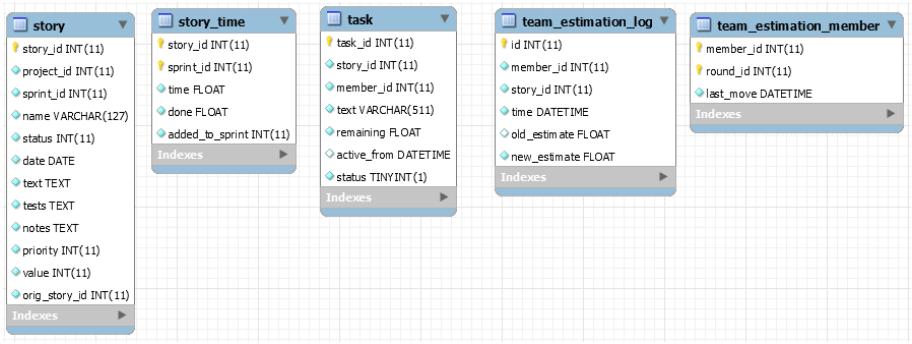
6.1 Priloga A: Prikaz vseh tabel in atributov

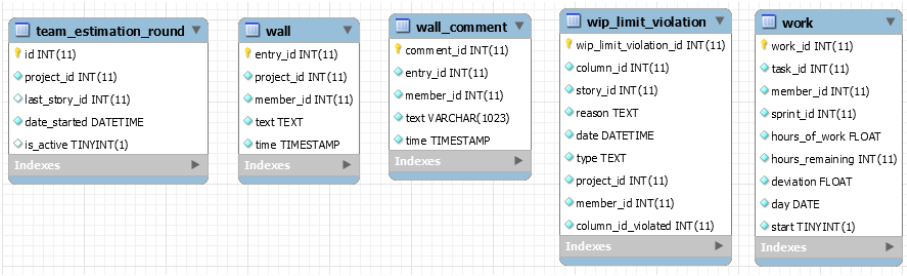



course_project
course_id INT(11)
project_id INT(11)
Indexes
daily_scrum
id INT(11)
project_id INT(11)
date DATE
Indexes
daily_scrum_item
daily_scrum_id INT(11)
member_id INT(11)
q1 TEXT
q2 TEXT
q3 TEXT
Indexes
groups
group_id INT(11)
group_name VARCHAR(50)
course_id INT(11)
scrummaster_member_id INT(11)
project_id INT(11)
course_date_id INT(11)
assigned_project_id INT(11)
Indexes
member
member_id INT(11)
name VARCHAR(127)
firstname VARCHAR(127)
lastname VARCHAR(127)
email VARCHAR(127)
password VARCHAR(127)
role INT(11)
current_task INT(11)
work_start DATETIME
Indexes


member_course
member_id INT(11)
course_id INT(11)
Indexes
member_groups
member_id INT(11)
group_id INT(11)
Indexes
member_project
member_id INT(11)
project_id INT(11)
role INT(11)
Indexes
move_history
history_id INT(11)
date DATETIME
user_id INT(11)
project_id INT(11)
from_column VARCHAR(100)
to_column VARCHAR(100)
sprint_num INT(11)
action VARCHAR(11)
story_id INT(11)
Indexes
move_permissions
move_permission_id INT(11)
role_id INT(11)
column_id_from INT(11)
column_id_to INT(11)
Indexes


poker_estimate
round_id INT(11)
member_id INT(11)
estimate FLOAT
Indexes
poker_round
round_id INT(11)
sprint_id INT(11)
story_id INT(11)
time TIMESTAMP
active TINYINT(1)
Indexes
project
project_id INT(11)
key TINYINT(1)
name VARCHAR(127)
description TEXT
doc TEXT
doc_editor INT(11)
poker_story_id INT(11)
Indexes
settings
setting_id VARCHAR(15)
value VARCHAR(15)
Indexes
sprint
sprint_id INT(11)
project_id INT(11)
start DATETIME
end DATETIME
velocity INT(11)
Indexes


story
story_id INT(11)
project_id INT(11)
sprint_id INT(11)
name VARCHAR(127)
status INT(11)
date DATE
text TEXT
tests TEXT
notes TEXT
priority INT(11)
value INT(11)
orig_story_id INT(11)
Indexes
story_time
story_id INT(11)
sprint_id INT(11)
time FLOAT
done FLOAT
added_to_sprint INT(11)
Indexes
task
task_id INT(11)
story_id INT(11)
member_id INT(11)
text VARCHAR(511)
remaining FLOAT
active_from DATETIME
status TINYINT(1)
Indexes
team_estimation_log
id INT(11)
member_id INT(11)
story_id INT(11)
time DATETIME
old_estimate FLOAT
new_estimate FLOAT
Indexes
team_estimation_member
member_id INT(11)
round_id INT(11)
last_move DATETIME
Indexes


team_estimation_round
id INT(11)
project_id INT(11)
last_story_id INT(11)
date_started DATETIME
is_active TINYINT(1)
Indexes
wall
entry_id INT(11)
project_id INT(11)
member_id INT(11)
text TEXT
time TIMESTAMP
Indexes
wall_comment
comment_id INT(11)
entry_id INT(11)
member_id INT(11)
text VARCHAR(1023)
time TIMESTAMP
Indexes
wip_limit_violation
wip_limit_violation_id INT(11)
column_id INT(11)
story_id INT(11)
reason TEXT
date DATETIME
type TEXT
project_id INT(11)
member_id INT(11)
column_id_violated INT(11)
Indexes
work
work_id INT(11)
task_id INT(11)
member_id INT(11)
sprint_id INT(11)
hours_of_work FLOAT
hours_remaining INT(11)
deviation FLOAT
day DATE
start TINYINT(1)
Indexes

Literatura

[1] C. Ladas, "Scrumban – Essays on Kanban Systems for Lean Software Development", Seattle, WA: Modus Cooperandi, 2008.

[2] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Martin Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, D. Thomas, "Manifesto for Agile Software Development", 2001, dostopno na:
http://agilemanifesto.org/

[3] 8th Annual State of Agile Development Survey, 2014. dostopno na:
http://www.versionone.com/pdf/2013-state-of-agile-survey.pdf

[4] 7th Annual State of Agile Development Survey, 2013. dostopno na:
http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf

[5] T. Ohno, "Toyota Production System - beyond large-scale production", Productivity Press, Junij 1988, stran 29.

[6] V. Mahnič, "Applying kanban principles to software development", Zbornik konference "IT for Practice 2013", Ostrava, 2013, stran 89-96.

[7] A. Časar ,"Programsko orodje za vodenje projektov po metodi scrum", Ljubljana, 2011 dostopno na:
http://eprints.fri.uni-lj.si/1586/1/%C4%8Casar1.pdf

[8] D. Radigan ,“A brief introduction to kanban“, prevzeto februar 2015, dostopno na:
https://www.atlassian.com/agile/kanban

[9] K. Schwaber, “Agile Project Management with Scrum“, Microsoft Press, 2004

[10] H. Kniberg, M. Skarin, “Kanban and Scrum – making the most of both“, C4Media Inc, 2010

[11] M. Cohn, “User stories applied for agile software development“, Addison-Wesley, 2004

[12] C. Díaz ,“Server-side programming language statistics“, prevzeto marec 2015, dostopno na:
http://blog.websitesframeworks.com/2013/03/programming-language-statistics-in-server-side-161/

[13] PHP, Dostopno na: “http://php.net/“

[14] MySql, Dostopno na: “http://www.mysql.com/“

[15] jQuery, Dostopno na: “http://jquery.com/“

[16] jQueryUI, Dostopno na: “http://jqueryui.com/“

[17] pChart, Dostopno na: “http://pchart.sourceforge.net/“

[18] JavaScript, Dostopno na: “http://www.w3schools.com/js/“

[19] AJAX, Dostopno na: “http://www.w3schools.com/ajax/“

[20] phpMailer, Dostopno na: “http://phpmailer.worxware.com/“

[21] CSS, Dostopno na: “http://www.w3schools.com/css/“

[22] HTML, Dostopno na: “http://www.w3schools.com/html/“