Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Peter Zemljak

Analiza uvedbe metode Scrum v manjšem podjetju

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:

Tematika naloge:

Predstavite agilno metodo za razvoj programske opreme Scrum in opišite poskus njene uvedbe v podjetju, kjer delate. Podrobneje predstavite projekt, na katerem je bila metoda uvedena, in način, kako ste jo vpeljali. Iz opisa naj bo razvidno, kako ste izvajali tipične prakse, ki jih predpisuje Scrum, in v kolikšni meri ste bili pri tem uspešni. Nalogo zaključite s kritično analizo sprejetja posameznih praks in predlogi, kako uspešneje uporabljati Scrum v prihodnje.

Zahvaljujem se mentorju prof. dr. Viljanu Mahniču za odzivnost, vodenje in pomoč pri izdelavi diplomskega dela. Zahvaljujem se tudi sodelavcem: Luki, Vitu in Dejanu. Na koncu se zahvaljujem še družini za podporo v času študija in Nataši, ki mi je bila v veliko pomoč pri izdelavi diplomskega dela.

Kazalo

Seznam uporabljenih kratic

kratica	opis
GIJ	gospodarska javna infrastruktura
DEM	Daljinski energetski manager

Povzetek

Naslov: Analiza uvedbe metode Scrum v manjšem podjetju

Avtor: Peter Zemljak

V diplomskem delu je opisana uvedba metode Scrum v manjšem podjetju. Scrum je bil uporabljen za razvoj spletne aplikacije Javna razsvetljava. Opisali smo razloge za uvedbo, predstavili projekt, pri katerem smo uporabili Scrum, ter opisali postopek implementacije Scruma v podjetju. Opravili smo analizo uporabljene metode ter s pomočjo intervjujev ocenili prakse Scruma. Poiskali smo tudi predloge za boljše izvajanje Scruma.

Ključne besede: Agilni razvoj programske opreme, Scrum, energetsko upravljanje, E2 Manager, Javna razsvetljava.

Abstract

Title: Analysis of Scrum introduction in a small company

Author: Peter Zemljak

In this bachelor thesis we have described implementation of Scrum in a small company. Scrum was used for the development of web application Javna razvetljava. We have described the reasons for Scrum introduction, presented the project for which Scrum was used and described implementation process of Scrum in our company. We have analysed the used method and evaluated Scrum practices with interviews. We have also proposed improvements for Scrum usage.

Keywords: Agile software development, Scrum, energy management, E2 Manager, Javna razsvetljava.

Poglavje 1

Uvod

V prejšnjih desetletjih je bil v večini projektov za razvoj programske opreme uporabljen slapovni model, pri katerem se projekt po fazah postopno pomika do končnega izdelka. Razvoj po slapovnem modelu je bil počasen in nepredvidljiv, rezultat pa pogosto ni bil končni izdelek, ki ga je želela stranka ali končni uporabnik. Takratni načini vodenja projektov, ki so temeljili na podrobnih vnaprej izdelanih grafih in tabelah, so vodstvu omogočali popoln nadzor nad razvojnim procesom, vendar so projekti skoraj vedno zamujali in močno presegali proračune [19].

Da bi odpravili omenjene težave, je bila pred dvajsetimi leti oblikovana metoda Scrum kot hitrejši, zanesljivejši in učinkovitejši način razvijanja programske opreme. Predstavlja radikalno spremembo načina dela, ki je do takrat temeljil na odločitvah vodilnih. Scrum je podoben evolucijskim, prilagodljivim in avtonomnim sistemom. Metoda je takoj po nastanku postala najbolj priljubljen način razvijanja programske opreme v Silicijevi dolini v Ameriki [19].

Podjetje, v katerem sem zaposlen, se ukvarja z energetskim upravljanjem objektov. Izdelujemo energetske preglede, energetske izkaznice in energetske koncepte za objekte ter občine. Prav tako se ukvarjamo z energetskim upravljanjem in načrtovanjem javne razsvetljave. Programerji v podjetju izdelujemo in vzdržujemo programsko opremo, ki je uporabljena kot pripomoček

pri izvajanju prej naštetih dejavnosti podjetja. Ker na trgu še ni bilo dobrega sistema za vodenje in upravljanje javne razsvetljave, smo se v podjetju odločili izdelati aplikacijo, ki nam bo to omogočala. Tako je nastala spletna aplikacija Javna razsvetljava.

Do takrat smo za razvoj aplikacij v podjetju uporabljali metodo dela, ki je temeljila na slapovnem modelu, s katero smo imeli veliko težav. Zaradi pogostih sprememb pri specifikacijah aplikacij so projekti trajali tudi dvakrat dlje, kot je bilo sprva načrtovano. Ob začetku projekta Javna razsvetljava smo se odločili uvesti spremembe, ki bi nam omogočile lažje prilagajanje nenehnim spremembam in zmanjšale zamude. Ker je bila to zelo dobra priložnost za spremembo načina dela, smo se z direktorjem dogovorili za uporabo metode Scrum.

Namen te diplomske naloge je opisati vpeljavo in izvajanje metode Scrum v našem podjetju ter analizirati uspešnost uporabljene metode. Zanimalo nas je, kako se bo Scrum obnesel v primeru, ko je udeležencev v razvoju malo in je komunikacija s končnimi uporabniki takojšnja.

V nadaljevanju bomo podrobneje opisali metodo Scrum. Nato bomo predstavili projekt, pri katerem smo uporabili Scrum. Opisali bomo potek uvedbe Scruma v našem podjetju in ovire, na katere smo naleteli pri uvedbi metode. V četrtem poglavju bomo analizirali uspešnost uporabe metode Scrum pri razvoju projekta ter podali predloge za izboljšanje.

Poglavje 2

Opis metode Scrum

2.1 Agilne metode

Izraz „agilen“ sega v leto 2001, ko se je 17 vodilnih strokovnjakov na področju razvoja programske opreme zbralo na sestanku, da bi primerjali svoje ideje. Na tem sestanku so ustvarili dokument, ki ga danes imenujemo Manifest agilnega razvoja programske opreme *(angl. Agile Manifesto)*[16]. Ta dokument postavi v ospredje naslednje vrednote:

- 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.

Nekateri od njih so kasneje ustanovili neprofitno organizacijo, imenovano „The Agile Alliance“, katere cilj je promocija agilnega razvoja. Manifest agilnega razvoja programske opreme predpisuje principe in postopke, ki bi jih morale upoštevati vse agilne metode [21]. Več o agilnih metodah si lahko preberemo v članku [15].

2.2 Metoda Scrum

Scrum je zasnovan na teoriji empiričnega nadzora procesov oziroma empirizmu. Empirizem trdi, da se znanje pridobi z izkušnjami in odločitvami na podlagi znanega. Scrum uporablja iterativni, inkrementalni pristop za optimizacijo predvidljivosti in nadzor tveganja. Vsako izvedbo empiričnega nadzora procesov podpirajo trije stebri: transparentnost, pregled in prilagoditev [18].

Ker v metodi Scrum obstaja nekaj izrazov, ki še niso ustrezno prevedeni v slovenščino, bomo v tem diplomskem delu uporabljali izraze, ki se pojavljajo v [19].

2.2.1 Pojmi v metodi Scrum

V metodi Scrum se pojavljajo pojmi, ki lahko imajo zunaj konteksta Scruma drugačen pomen.

Uporabniška zgodba *(angl. user story)* predstavlja ohlapno opisane funkcionalnosti izdelka. Je kratek zapis, podan v enem ali dveh stavkih, s perspektive končnega uporabnika.

Seznam zahtev *(angl. product backlog)* je seznam uporabniških zgodb, ki jih produkt potrebuje in so osnova za črpanje podatkov o delovanju produkta. Seznam zahtev ni nikoli zaključen, saj se zahteve nenehno dodajajo in posodabljajo glede na odzive strank.

Naloga *(angl. task)* je del uporabniške zgodbe, ki definira, na kakšen način naj bo delo opravljeno. Praviloma so naloge krajše in vsebujejo le eno vrsto dela (samo programiranje ...).

Točka *(angl. story point)* predstavlja oceno zahtevnosti uporabniške zgodbe.

Koncept „done“ določa, kdaj je naloga končana. Predstavlja pravila, ki si jih udeleženci v Scrumu postavijo, da lahko nalogo označijo kot končano.

Sprint predstavlja časovno obdobje, v katerem se ustvari uporaben produkt. Sprint lahko traja največ en mesec, v praksi pa so najbolj pogosti eno- ali dvotedenski sprinti. Priporočeno je, da je trajanje vseh sprintov v

projektu enako dolgo. Sprintu določimo tudi končni datum, ki ga med trajanjem sprinta ni več mogoče spremeniti. Nov sprint se začne takoj, ko se prejšnji sprint konča. Med sprintom se vsebina sprinta ne sme spreminjati, kar pomeni, da uporabniških zgodb sprintu ne smemo dodajati ali odvzemati. Skrbnik izdelka lahko sprint tudi prekine, v tem primeru se že izdelane uporabniške zgodbe preverijo, neizdelane pa vrnejo nazaj na seznam zahtev. Prekinitve sprintov se le redko dogajajo.

Hitrost *(angl. velocity)* predstavlja seštevek števila točk uporabniških zgodb, ki jih je razvojna skupina realizirala v enem sprintu.

2.2.2 Vloge v metodi Scrum

V metodi Scrum nastopajo tri vloge: skrbnik izdelka *(angl. product owner)*, skrbnik procesa *(angl. scrum master)* in razvojna skupina *(angl. development team)*.

Skrbnik izdelka je predstavnik naročnika v metodi Scrum. Odgovoren je za tveganja in uspeh izdelka. Njegova osnovna naloga je urejanje in posodabljanje seznama zahtev. Skrbnik izdelka mora uporabniškim zgodbam na seznamu dodeliti prioriteto. Na koncu mora opravljene uporabniške zgodbe tudi potrditi. Vedno mora biti na voljo ostalim udeležencem za dodatna pojasnila o uporabniških zgodbah.

Razvojno skupino sestavljajo ljudje, ki so zadolženi za realizacijo uporabniških zgodb. Skupina mora sama določiti način, s katerim realizira uporabniško zgodbo. To naredi s pomočjo razčlenjevanja uporabniških zgodb na več manjših nalog. Člani razvojne skupine morajo posedovati vsa znanja za realizacijo nalog. Scrum ima razvojno skupino za celoto in tako spodbuja sodelovanje med člani skupine. Člani svoja znanja delijo med sabo in tako dosežejo večjo učinkovitost. Priporočena velikost skupine v Scrumu je od 3 do 9 oseb. Pri manjših skupinah lahko pride do pomanjkanja znanj za dokončanje zgodbe, pri večjih pa postane upravljanje skupine preveč zapleteno.

Skrbnik procesa skrbi za uporabo in razumevanje metode Scrum med vsemi udeleženci Scruma. Njegova naloga je, da se metoda Scrum dosledno

in pravilno izvaja na vseh ravneh organizacije. Prav tako je njegova naloga, da najde in odstrani vse ovire, ki ovirajo razvojno skupino pri razvoju, in s tem pripomore k večji produktivnosti in hitrosti skupine.

2.2.3 Dogodki v metodi Scrum

Predpisani dogodki se v Scrumu uporabljajo za ustvarjanje rutine in optimizacijo dela z najmanjšo izgubo časa. Vsi dogodki v Scrumu so časovno omejeni in omogočajo analizo ter izboljšanje procesa dela. Če dogodek izpustimo, izgubimo tudi priložnost za izboljšavo.

Sestanek za načrtovanje sprinta *(angl. sprint planning meeting)* se izvede ob začetku sprinta. Na tem sestanku se vsi udeleženci Scruma dogovorijo o vsebini naslednjega sprinta. Najprej se dogovorijo, katere uporabniške zgodbe bodo realizirali v sprintu. Če so uporabniške zgodbe nejasne, jih mora skrbnik izdelka podrobneje razložiti. Uporabniškim zgodbam se nato oceni zahtevnost s številom točk in se jih razbije na naloge. Na podlagi že opravljenih sprintov si lahko razvojna skupina pri načrtovanju pomaga s hitrostjo, ki jo je dosegla v prejšnjih sprintih. Nalogam se oceni še čas izvajanja v urah ali minutah ter določi način opravljanja nalog. Na koncu sestanka mora imeti ekipa izbrane uporabniške zgodbe, ki jih bo realizirala v tem sprintu, ter način opravljanja nalog.

Na dnevnih sestankih *(angl. daily scrum)* razvojna skupina uskladi svoje aktivnosti in ustvari dnevni načrt dela. Ti sestanki trajajo največ 15 minut in potekajo vsak dan ob istem času. Na sestankih vsak član skupine odgovori na naslednja vprašanja: „Kako si včeraj ekipi pomagal dokončati sprint?“, „Kako boš danes ekipi pomagal dokončati sprint?“ in „Kaj ekipo ovira pri doseganju cilja sprinta?“. Ta sestanek je namenjen le članom razvojne skupine in ne vključuje skrbnika izdelka.

Pregled sprinta *(angl. sprint review)* je sestanek, ki se izvede na koncu vsakega sprinta. Ekipa pokaže, katere uporabniške zgodbe je realizirala v preteklem sprintu. Na seznamu zahtev se realizirane uporabniške zgodbe označijo kot končane. Na tem sestanku lahko poleg udeležencev v Scrumu

sodelujejo tudi drugi, ki jih zanima potek projekta.

Retrospektiva sprinta *(angl. sprint retrospective)* je sestanek, na katerem razvojna skupina oceni samo sebe in poišče možnosti za izboljšanje produktivnosti. Preveri delo v preteklem sprintu in poišče morebitne ovire, ki so se pojavljale med sprintom in upočasnjujejo razvojno skupino pri delu. Skupaj s skrbnikom procesa izdela načrt za odpravo teh ovir, ki bo uporabljen v naslednjem sprintu.

Poglavje 3

Opis projekta

Podjetje, v katerem sem zaposlen, se ukvarja z energetskim upravljanjem in ima 12 zaposlenih. V podjetju smo zaposleni trije programerji, ki smo zadolženi za razvijanje in vzdrževanje aplikacij, ki so uporabljene kot pripomoček pri energetskem upravljanju stavb in javne razsvetljave. V podjetju smo do zdaj razvili tri večje aplikacije. DEM *(Daljinski energetski manager)* je že upokojena namizna aplikacija, ki smo jo uporabljali v podjetju do leta 2014 kot pripomoček za energetsko upravljanje stavb. Nadomestil jo je E2 Manager, ki je posodobljena spletna verzija aplikacije DEM in jo bomo podrobneje opisali v nadaljevanju, ter spletna aplikacija Javna razsvetljava.

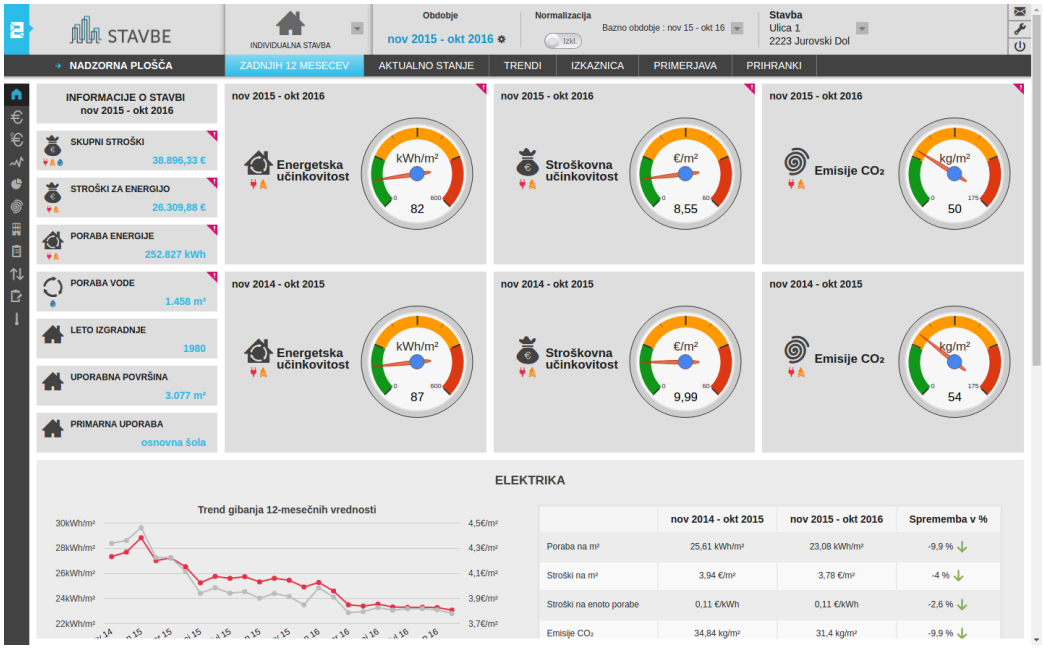
Metodo Scrum smo uporabili za projekt, ki je zajemal razvoj in izdelavo spletne aplikacije Javna razsvetljava. Aplikacija bi omogočala vodenje katastra gospodarske javne infrastrukture (GIJ) javne razsvetljave, energetsko upravljanje javne razsvetljave ter vodenje vzdrževanja omrežja javne razsvetljave. Strankam bi jo ponudili v paketu z že obstoječo aplikacijo E2 Manager, zato je naročnik zahteval možnost prijave v obe aplikaciji z enakim uporabniškim imenom in geslom (*single sign-on*).

3.1 E2 Manager

E2 Manager je spletna aplikacija za energetsko knjigovodstvo in upravljanje stavb. Na podlagi vnesenih računov za elektriko, vodo in ogrevanje aplikacija omogoča prikaz stroškov in porabe za energente v različnih časovnih obdobjih. Spletna aplikacija je namenjena predvsem poslovnim objektom, lahko pa se uporablja tudi za manjše stanovanjske objekte, saj pokaže potencialne možnosti za prihranke.

Za uspešno upravljanje stavbe z aplikacijo E2 Manager moramo najprej v sistem vnesti lastnosti stavbe. Te lastnosti so ime stavbe, naslov, klimatski kraj (najbližja meteorološka postaja), katastrska občina, dejavnost stavbe, parcelna številka, zemljepisna dolžina in širina, leto izgradnje, neto površina, uporabna površina in druge. Nato vnesemo še podatke o stroškovnih in odjemnih mestih ter nastavimo predlogo računa, ki nam olajša kasnejši vnos računov. Po vnosu računov nam aplikacija omogoča tudi urejanje teh računov, pregled arhiva računov, pregled aktivnosti posameznih odjemnih mest in pregled manjkajočih računov. V aplikacijo lahko uvozimo tudi e-račune.

Po vnosu računov lahko zbrane podatke opazujemo na več načinov. Na začetni strani aplikacije so prikazani podatki o porabi, stroških in emisijah ogljikovega dioksida za obdobje zadnjih 12 mesecev, za obdobje aktualnega leta, trendi porabe in stroškov skozi več let, poenostavljena energetska izkaznica stavbe, primerjava stavbe z drugimi stavbami, ki že obstajajo v aplikaciji, ter prihranki, doseženi pri posamezni stavbi (slika 3.1). Podatke o porabi in stroških lahko v aplikaciji opazujemo v različnih časovnih obdobjih, kar vključuje prikaz v aktualnem letu, prikaz podatkov v zadnjih 12 mesecih, mesečni prikaz, prikaz po letih in dinamični prikaz, pri katerem obdobje prikaza izbere uporabnik (slika 3.2). Na voljo so tudi podatki o ogljičnem odtisu stavbe, s katerimi lahko opazujemo ekvivalent in izpuste ogljikovega dioksida ter drugih toplogrednih plinov (slika 3.3). Vse zgoraj naštete podatke lahko združujemo po vrstah storitve, energentih in stroškovnih mestih, na voljo pa so tudi v po površini normirani obliki (slika 3.4). V aplikaciji je na voljo

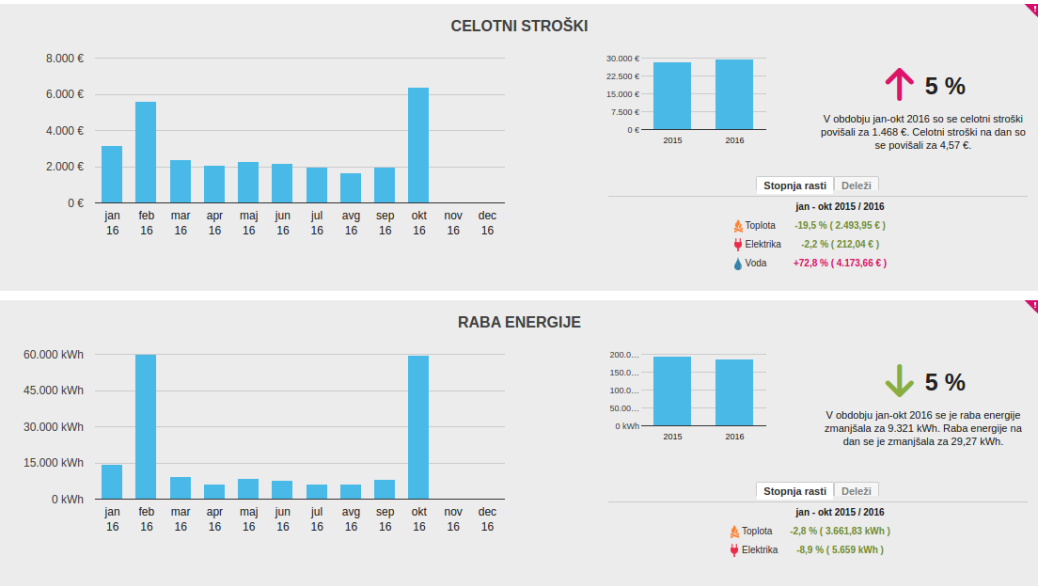


Slika 3.1: Začetna stran programa E2 Manager. Prikaz podatkov o stroških, porabi in emisijah ogljikovega dioksida za zadnjih 12 mesecev od trenutno aktivnega meseca.

normalizacija podatkov po temperaturnem primanjkljaju v izbranem klimatskem kraju ali po poljubnem parametru, ki ga vnese uporabnik. Vsi podatki so na voljo v grafični in tabelarični obliki. Iz vseh podatkov je mogoče ustvariti tudi poročila, ki lahko vsebujejo podatke posamezne stavbe ali skupine stavb in so na voljo v formatu PDF.

3.2 Javna razsvetljava

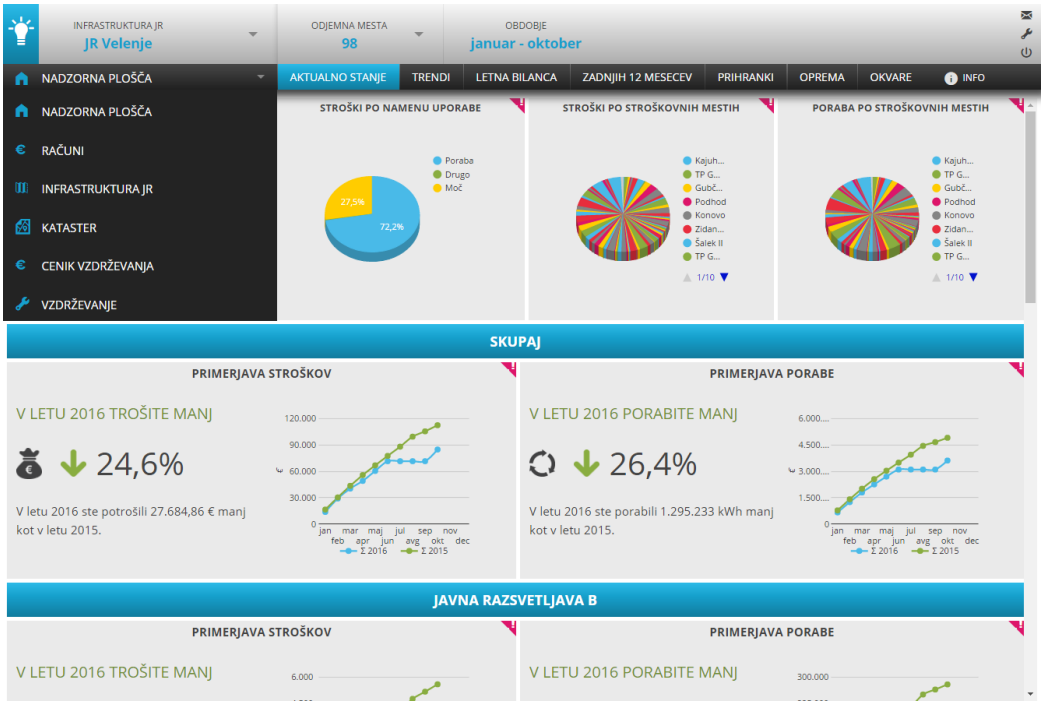
Spletna aplikacija Javna razsvetljava je namenjena energetskemu upravljanju omrežja javne razsvetljave. Omogoča grafični prikaz stroškov in porabe električne energije po odjemnih mestih na podlagi vnesenih računov za elektriko. Dodatno nam omogoča tudi javljanje napak na omrežju, evidentiranje izvedenih vzdrževalnih del in vzdrževanje katastra gospodarske javne infra-



Slika 3.2: Pregled podatkov o porabi in stroških v različnih časovnih intervalih v aplikaciji E2 Manager.

Slika 3.3: Pregled emisij ogljikovega dioksida in drugih toplogrednih plinov v aplikaciji E2 Manager.

Slika 3.4: Prikaz podatkov o porabi vode, normiranih po površini, in meni z možnostmi združevanja podatkov v aplikaciji E2 Manager.

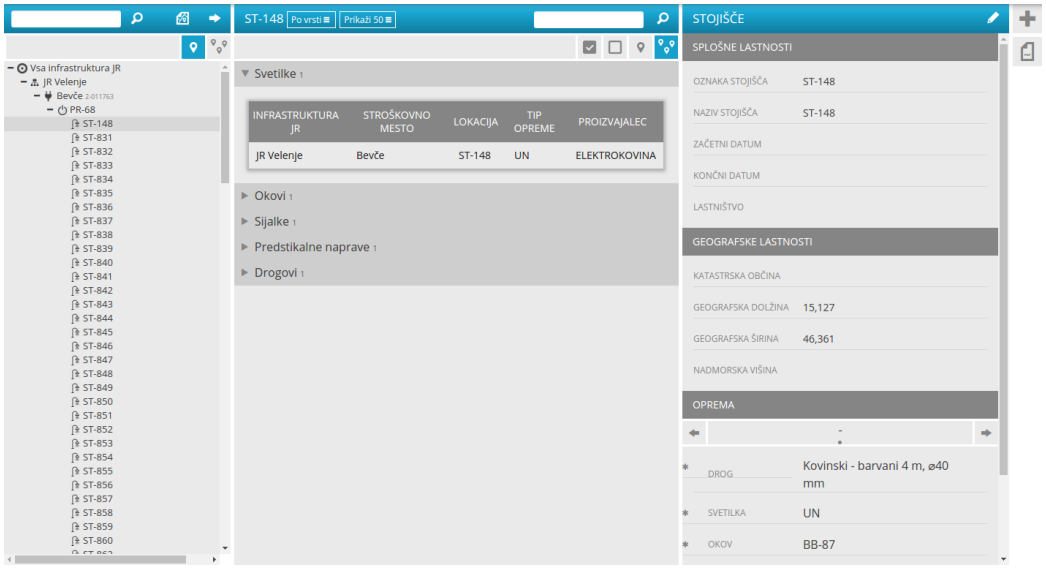


Slika 3.5: Začetna stran spletne aplikacije Javna razsvetljava. Prikazuje podatke o porabi in stroških za elektriko v aktualnem letu v občini Velenje.

strukture (GIJ) javne razsvetljave. Spletna aplikacija olajša delo skrbniku javne razsvetljave in omogoča hitrejše odpravljanje okvar na omrežju.

Vnos računov za elektriko in prikaz podatkov stroškov in porabe za električno energijo po odjemnih mestih deluje po enakem principu kot pri aplikaciji E2 Manager in je podrobneje opisan v poglavju 3.1. Začetna stran aplikacije je prikazana na sliki 3.5.

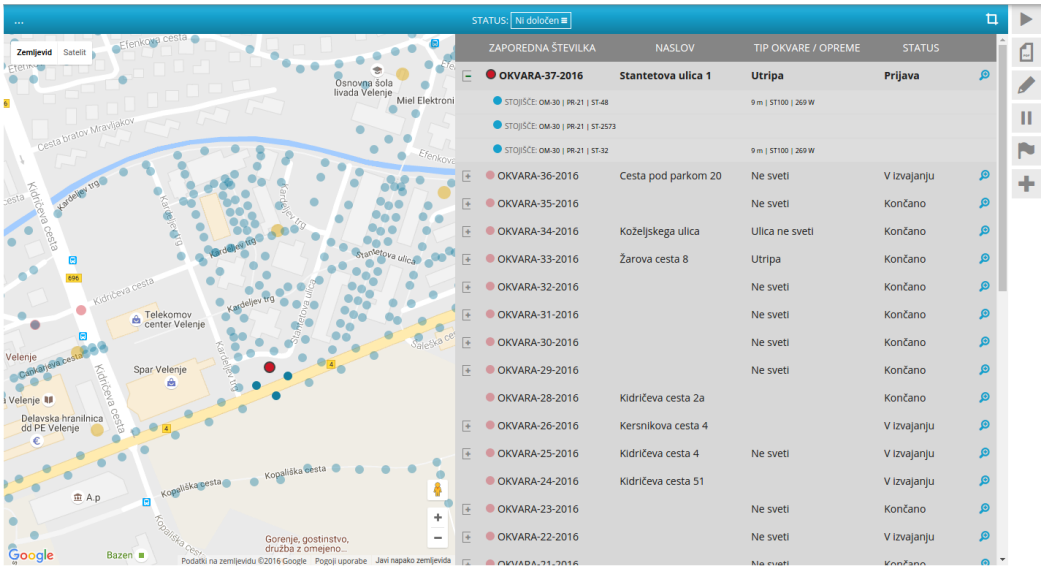
Vodenje katastra omogoča pregled sestavnih delov in zgodovine sprememb posameznega elementa omrežja (slika 3.6). Vsebuje tudi pregled obratovalnih shem in moči za poljuben del omrežja. Omogoča nam pripravo poročil o elementih na omrežju, ki lahko zajemajo tudi zemljevid lokacij ali slike delov na posameznem elementu. Podatke lahko uporabniki uvozijo v aplikacijo iz arhiva podatkov o omrežju javne razsvetljave, ki je voden po posameznih občinah.



Slika 3.6: Prikaz podatkov za posamezen element v katastru GIJ javne razsvetljave v občini Velenje.

Vodenje vzdrževanja omogoča pregled okvar (slika 3.7), izdelavo delovnih nalogov za odpravljanje napak ter poročila o izvršenih vzdrževalnih delih na omrežju. Vzdrževalci imajo tako boljši nadzor nad okvarami na omrežju, kar jim omogoča hitrejše odpravljanje okvar.

Občani lahko prijavijo napako na omrežju prek spletnega vmesnika (slika 3.8), ki je povezan z aplikacijo. Namen tega vmesnika je kar najhitreje in čim bolj natančno opozoriti vzdrževalca na okvaro v omrežju.



Slika 3.7: Prikaz okvar, ki so bile prijavljene na omrežju javne razsvetljave v aplikaciji Javna razsvetljava v občini Velenje.

Slika 3.8: Stran za prijavo napak na omrežju javne razsvetljave, namenjena občanom Velenja.

Poglavje 4

Izvedba projekta

Izvedbo projekta Javna razsvetljava smo začeli konec maja 2015. Prva izdaja je bila načrtovana v začetku leta 2016. Scrum smo na projektu uporabljali do novembra 2015 in v tem času izvedli 11 dvotedenskih sprintov. Projekt se je nato nadaljeval po slapovnem modelu in bil izdan v začetku novembra 2016, kar je 10 mesecev po načrtovanem roku.

4.1 Uporabljene tehnologije

Za aplikacijo Javna razsvetljava smo uporabili najnovejše tehnologije, ki so bile na voljo ob začetku projekta. Programska koda je napisana v Javi, nadgrajeni z ogrodjem Spring, za pomoč pri izdelavi uporabniškega vmesnika pa smo uporabili ogrodje Wicket.

Projekt smo izdelali s programskim jezikom Java verzije 8 [6]. S prehodom na Javo 8 smo največ pridobili z lambda izrazi *(angl. lambda expression)*, ki omogočajo obravnavanje funkcije kot argumenta metode ali dela kode kot podatek. Z njimi smo lahko izdelali krajšo, preglednejšo in razumljivejšo programsko kodo [7].

Dobrodošla novost so bili tudi tokovi *(angl. stream)*, ki olajšajo delo s seznami podatkov, in nova zbirka za delo s časovnimi elementi *Date-Time Package*. Več o novostih v Javi 8 je mogoče prebrati na spletni strani [7].

Za osnovo aplikacije smo uporabili ogrodje Spring [14]. Spring je odprtokodno ogrodje, ki ga je ustvaril Rod Johnson. Bilo je ustvarjeno z namenom poenostavitve razvoja profesionalnih aplikacij in omogočanja uporabe osnovnih gradnikov Java za dosego ciljev, ki so bili prej dosegljivi samo s pomočjo *enterprise Java beans*. Spring ni uporaben samo za razvoj na strežniški strani. Vsaka aplikacija, ki uporablja Javo, lahko z uporabo Springa pridobi na področjih enostavnosti, testiranja in šibke sklopljenosti *(angl. loose coupling)* [20]. Spring Boot, Spring Data in Spring Security so deli ogrodja Spring, s katerimi smo si najbolj pomagali. Spring Boot nam je olajšal osnovno konfiguracijo programa, saj že vsebuje privzete nastavitve za spletne aplikacije, ki jih v veliki meri ni bilo treba spreminjati. Spring Data nam je olajšal interakcijo s podatkovno bazo. S Spring Security smo si pomagali pri implementaciji varnostnih mehanizmov v aplikaciji.

Uporabniški vmesnik smo izdelali s pomočjo ogrodja Wicket 7 [1]. Wicket je ogrodje, ki poenostavi izdelavo spletnih aplikacij. Uporablja objektni programski model, ki spodbuja uporabnika k ustvarjanju kode, ki jo je lažje vzdrževati. Prav tako pomaga pri rasti aplikacij s komponentami, ki so primerne za večkratno uporabo [17].

V aplikaciji smo uporabili podatkovno bazo Postgresql [11], za dodatne elemente v uporabniškem vmesniku, kot so gumbi, meniji in pojavna okna, smo uporabili ogrodje Foundation 5 [4], za bolje strukturirano programsko kodo pa smo pri programiranju uporabljali orodje Lombok [12].

4.2 Izbor metodologije

Pred začetkom projekta Javna razsvetljava smo v podjetju uporabljali metodo razvoja, ki je temeljila na slapovnem modelu. Najprej smo dobili zahteve, nato smo izdelali načrt izdelave ter se lotili kodiranja. Zaradi pogostega spreminjanja zahtev smo morali fazo načrtovanja večkrat ponoviti, v skrajnih primerih smo morali znova kodirati, saj so se zahteve preveč spremenile in jim že izdelane rešitve niso več ustrezale. Prav tako so bile zahteve razpršene na

več mestih, kot so elektronska pošta, dokumenti z zahtevki, listki na mizah. Ta sistem nam je pogosto povzročal težave, zato smo programerji v podjetju dali pobudo za spremembo metode dela.

Odločili smo se za Scrum, ker omogoča hitre izdaje in predvsem agilnost, ki smo jo potrebovali zaradi hitro spreminjajočih se zahtev. Prav tako smo hoteli voditi enoten arhiv zahtev. Želeli smo, da bi bile zahteve, ki smo jih morali realizirati, zbrane na enem mestu. V Scrumu se za ta namen uporablja seznam zahtev, ki je opisan v poglavju 2.2.1. Pri uvedbi smo si pomagali z vodnikom The Scrum Guide [18], ki nas je vodil pri začetnih korakih.

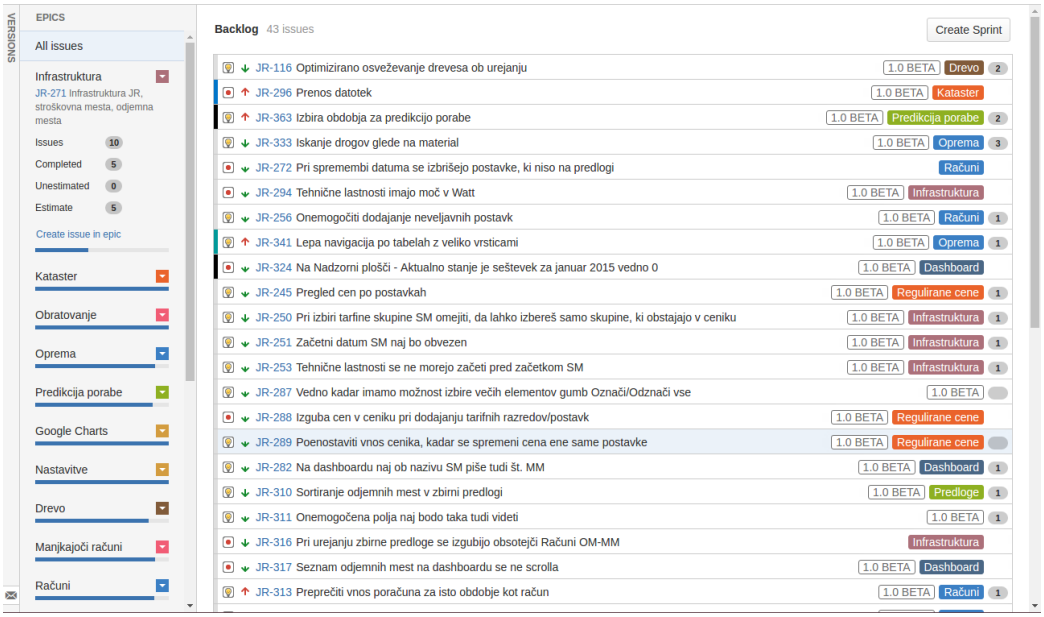
4.3 Vloge

Vloge v Scrumu smo si razdelili v našem podjetju.

Skrbnik izdelka je postal direktor našega podjetja. To vlogo smo mu dodelili, ker je že pred uvedbo Scruma skrbel za načrtovanje funkcionalnosti aplikacij, ki smo jih izdelali v podjetju. Vedno nam je bil na voljo za podrobne razlage uporabniških zgodb. Na začetku je kazalo, da smo vlogo skrbnika izdelka dodelili pravilno, a se je kasneje pokazalo, da direktor vloge ni pravilno izvajal, kar je opisano v naslednjih poglavjih.

Skrbnik procesa je postal vodja informacijske službe. Ta položaj smo mu dodelili, ker je imel največ izkušenj z metodami dela. Bil je tudi pobudnik za uvedbo metode Scrum v našem podjetju. Za uspešno izvajanje vloge skrbnika procesa je natančno preučil pravila metode Scrum. Zaradi majhnega števila programerjev, zaposlenih v našem podjetju, je bil skrbnik procesa hkrati tudi član razvojne skupine.

Razvojno skupino smo sestavljali vodja informacijske službe in dva programerja, eden od teh sem bil jaz. Ker sem poleg razvoja nove aplikacije bil zadolžen še za vzdrževanje aplikacije E2 Manager, sem pri projektu v povprečju sodeloval le tri dni na teden. Vsi udeleženci v skupini smo imeli delovno mesto v istem prostoru, kar nam je olajšalo izvajanje metode Scrum.



Slika 4.1: Seznam zahtev v programski opremi Jira. V levem stolpcu so naštete velike uporabniške zgodbe *(epic)*, desni stolpec pa vsebuje vse uporabniške zgodbe *(story)* in hrošče *(bug)*, ki še niso realizirani.

4.4 Orodje za vodenje projekta

Orodje za vodenje projekta je namenjeno upravljanju uporabniških zgodb, njihovih delov in spremljanju napredka pri izdelavi produkta. Za vodenje projekta smo izbrali programsko opremo Jira [8], saj smo jo v podjetju uporabljali že prej.

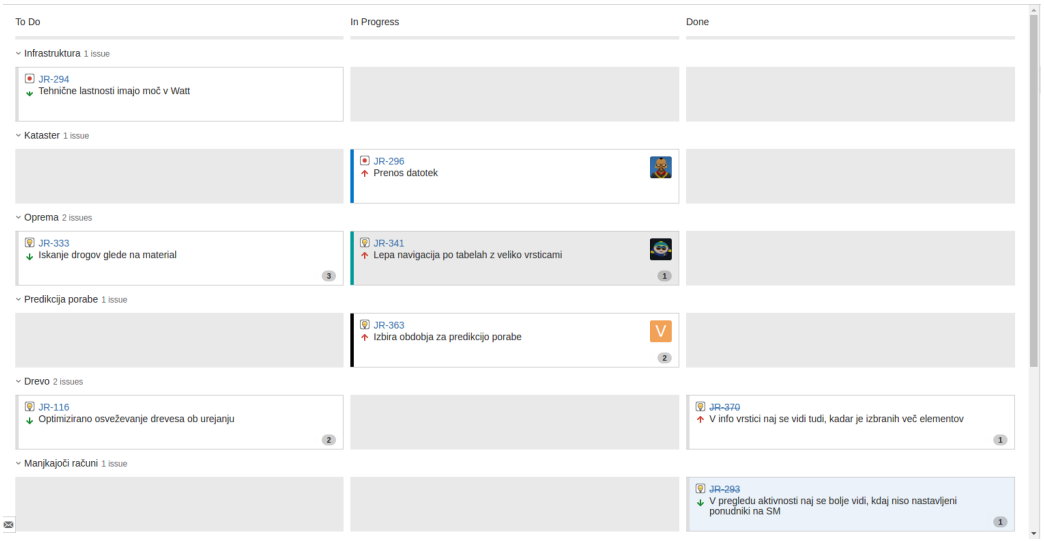
Jira omogoča upravljanje več metod dela, med njimi tudi Scruma. V Jiri je voden seznam zahtev, v katerega vnašamo uporabniške zgodbe, ki jih lahko označimo na več načinov (slika 4.1). Obsežne uporabniške zgodbe označimo z oznako *epic* in jih lahko po potrebi razbijemo na več manjših uporabniških zgodb z oznako *story*. Na seznam zahtev lahko dodajamo tudi hrošče, ki imajo ozanko *bug*. Naloge, na katere razdelimo uporabniške zgodbe, imajo oznako *sub-task*. V Jiri obstaja oznaka *task*, a se za agilno vodenje projektov ne uporablja [10].

Ocenjeno zahtevnost uporabniške zgodbe v številu točk vnesemo v Jiro kot *story point*. Število točk lahko vnesemo le za uporabniške zgodbe *(epic, story)*, nalogam *(sub-task)* pa vnesemo ocenjeno trajanje izdelave v dnevih, urah ali minutah. Sprejemne teste za uporabniško zgodbo vnesemo v Jiro v polje za opis zgodbe *description*. Vsaki uporabniški zgodbi določimo tudi prioriteto.

Ob začetku sprinta moramo v orodju Jira izbrati uporabniške zgodbe, ki naj jih sprint vsebuje. Nato Jira izračuna načrtovano hitrost, ki bi jo morali doseči, da bi dokončali vse uporabniške zgodbe v sprintu. Med sprintom lahko spremljamo status izdelave uporabniških zgodb, ki so lahko na čakanju *(to do)*, v izdelavi *(in progress)* ali dokončane *(done)* (slika 4.2). Jira omogoča spremljanje napredka z grafičnim prikazom realiziranih zgodb na celotnem projektu s *cumulative flow diagram* kot tudi v posameznih sprintih z *burn-down chart*. Zelo dobro se integrira z orodjem Bitbucket [2] in programskim paketom Eclipse [3]. Oba smo že pred uvedbo Scruma uporabljali za razvoj programske opreme. Več o uporabi Jire za Scrum je mogoče prebrati na spletni strani [9].

4.5 Dolžina sprinta

Pri določanju dolžine sprinta se uporabniki metode Scrum nismo strinjali. Skrbnik izdelka je hotel imeti sprinte dolge en teden, da bi imel hitro na voljo funkcionalen izdelek z vsebovanimi izboljšavami in bi jih lahko hitro preizkusil v praksi. V razvojni skupini smo predvidevali, da bi v tako kratkem obdobju težko opravili dovolj nalog, ki bi se povezovale med seboj. Hoteli smo imeti vsaj tri tedne dolge sprinte, kar bi olajšalo programiranje in odpravilo pogosto menjavanje konteksta ter nepotrebne sestanke. Na koncu smo sprejeli kompromis in dolžino sprinta določili na dva tedna.



Slika 4.2: Pregled statusa izdelave uporabniških zgodb v sprintu. V levem stolpcu so naloge na čakanju *(to do)*, v srednjem v izdelavi *(in progress)* in v desnem dokončane *(done)*.

4.6 Ocenjevanje zgodb

Za ocenjevanje uporabniških zgodb smo uporabili metodo *planning poker*. Pri metodi *planning poker* smo uporabili karte, na katerih so napisana števila v Fibbonacijevem zaporedju od 1 do 20 (slika 4.3). Karte so tako oštevilčene zato, da so med števili večje razlike in uporabniške zgodbe lažje ocenimo s primernim številom. Za potrebe ocenjevanja je vsak programer dobil svoj komplet kart. Pri ocenjevanju uporabniških zgodb je vsak programer ovrednotil uporabniško zgodbo s številom točk, ki ga ponazarjajo števila na kartah. Vsi udeleženci smo istočasno pokazali karto, ki smo jo izbrali. Če so bila izbrana števila sosednja v zaporedju, se je kot število točk uporabniške zgodbe upoštevalo povprečje števil na kartah. Če pa števila niso bila sosednja, sta udeleženca z najnižjim in najvišjim številom na karti utemeljila svojo odločitev, nato pa smo igro ponavljali, dokler nismo vsi izbrali enakih ali sosednjih števil.

Slika 4.3: Karte, ki smo jih uporabljali za igranje *planning poker*, s števili v Fibbonacijevem zaporedju od 1 do 20. Komplet vsebuje še karto za neskončnost, ki ponazarja zgodbo, ki jo je treba razbiti na manjše zgodbe, in karto *latte*, ki smo jo uporabili, ko je kdo od udeležencev potreboval odmor.

4.7 Uporabniške zgodbe

Uporabniške zgodbe je skrbnik izdelka vpisoval v seznam zahtev, ki smo ga vodili z orodjem za vodenje projekta Jira. Vsebovale so kratek opis funkcionalnosti in zahteve, ki smo jih morali doseči. Uporabniške zgodbe so včasih pokrivale večji nabor funkcionalnosti in jih je bilo treba razdeliti na več manjših uporabniških zgodb, pri čemer je vsaka opisovala posamezno funkcionalnost. Vsaki uporabniški zgodbi je skrbnik izdelka določil tudi prioriteto. Projekt Javna razsvetljava je sestavljalo 19 velikih uporabniških zgodb *(epic)*. Uporabniške zgodbe je dodajala na seznam zahtev tudi razvojna skupina, ko je med delom opazila potrebo po funkcionalnosti v aplikaciji.

Primer uporabniške zgodbe je uporaba zemljevida za navigacijo po elementih javne razsvetljave. Za ta namen je bila v aplikaciji pred izvedbo zgodbe uporabljena drevesna struktura, ki omogoča izbor posameznih elementov omrežja javne razsvetljave, označenih s šifro. Elementi v drevesu

so hierarhično razporejeni v razmerju „oče“-„sin“ in so med seboj povezani. Zgodba se je glasila: „Za navigacijo po programu mora biti na voljo zemljevid, ki temelji na Google Maps API[5]“. Vsebovala je tri sprejemne teste.

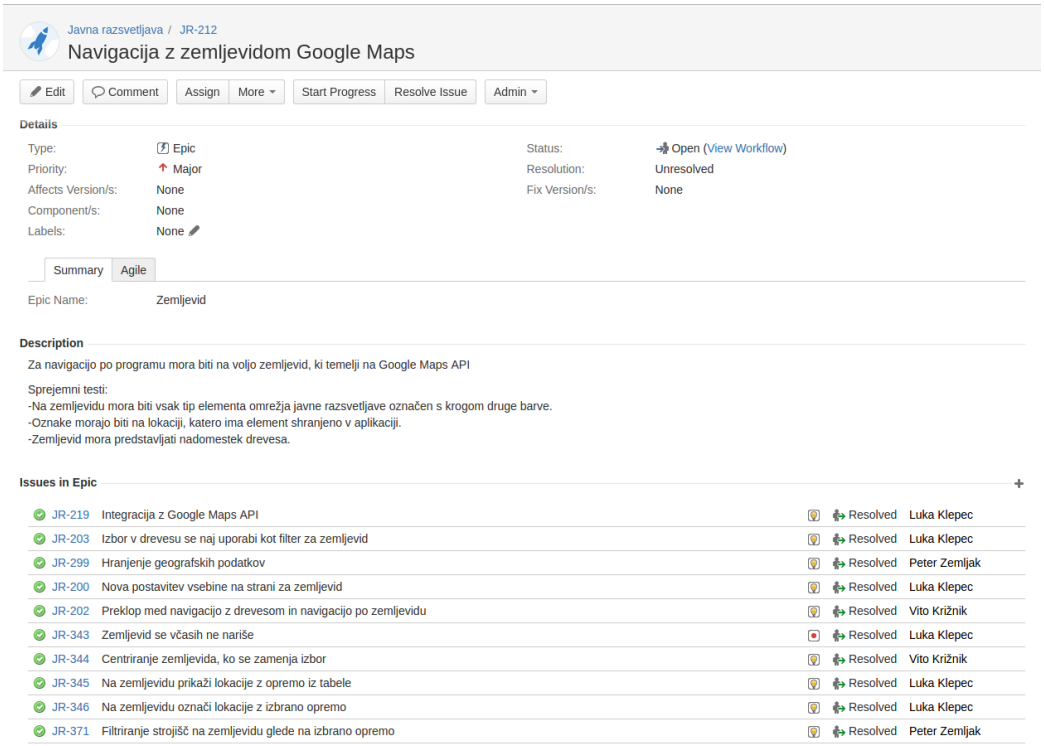
- Na zemljevidu mora biti vsak tip elementa omrežja javne razsvetljave označen s krogom druge barve.
- Oznake morajo biti na lokaciji, katero ima element shranjeno v aplikaciji.
- Zemljevid mora predstavljati nadomestek drevesa.

Uporabniško zgodbo smo razbili na devet manjših uporabniških zgodb in jih vnesli v orodje Jira (slika 4.4). Nato smo te uporabniške zgodbe ocenili na način, opisan v poglavju 4.6, njihovo število točk vnesli v orodje Jira in jih razdelili na naloge. Nalogam smo ocenili čas izvajanja v urah ali minutah ter ga vnesli v Jiro. Naloge smo si razdeljevali na dnevnih sestankih. Ko smo krovno uporabniško zgodbo prvič zaključili, jo je skrbnik izdelka zavrnil, ker ni bil zadovoljen z izborom barv posameznih elementov. To je pomenilo, da se je uporabniška zgodba vrnila na seznam zahtev. Ob začetku naslednjega sprinta smo ponovno ustvarili naloge, ki so reševale težavo z barvami elementov. Tokrat je skrbnik izdelka uporabniško zgodbo potrdil in smo jo lahko zaključili. Rezultat zaključene zgodbe je prikazan na sliki 4.5.

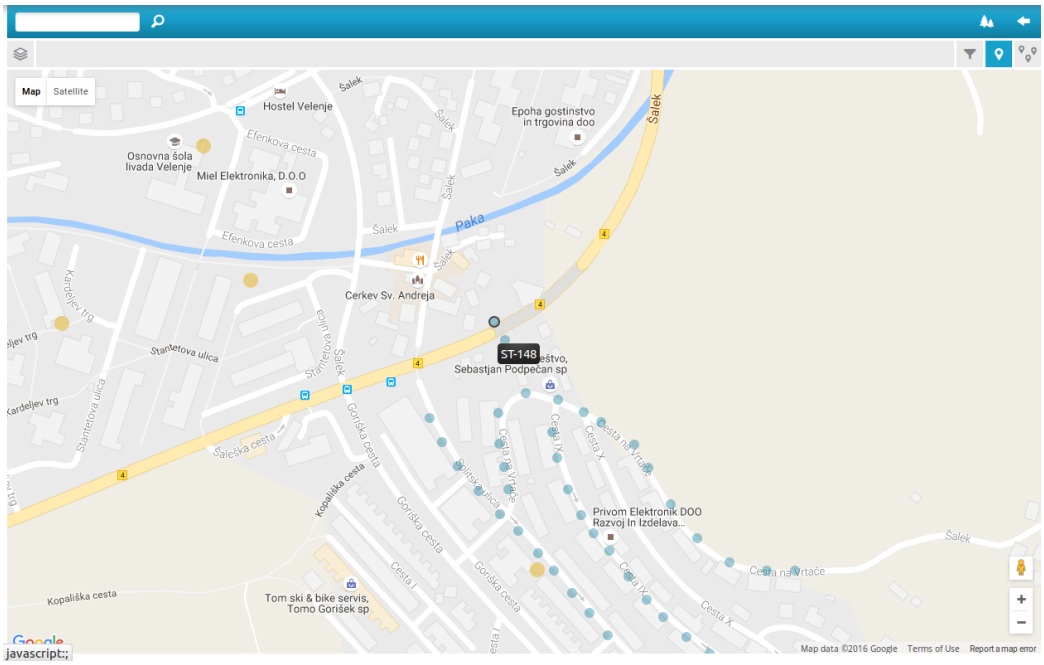
4.8 Koncept „done“

Koncept „done“ smo postavili tako, da je morala napisana koda ustrezati petim pogojem.

- Uporabniški vmesnik mora izgledati pravilno v vseh brskalnikih.
- Vse operacije, ki vračajo vnaprej znane rezultate, morajo biti pokrite s testi enot *(angl. unit tests)*.
- Vsa koda mora vsebovati dokumentacijo *javadoc*.



Slika 4.4: Prikaz uporabniške zgodbe „Za navigacijo po programu mora biti na voljo zemljevid, ki temelji na Google Maps API“, razdeljene na 9 manjših uporabniških zgodb in z enim hroščem v programu Jira.



Slika 4.5: Integracija programa Javna razsvetljava z zemljevidom Google Maps. Označen je element s šifro „ST-148“.

- Vso kodo pregleda nekdo od kolegov, ki ni avtor kode *(peer review)*.
- Končni izdelek mora biti ročno testiran v produkcijskem okolju.

Včasih je razvojna skupina za potrditev uporabniške zgodbe potrebovala mnenje skrbnika izdelka, ki je preveril delovanje rešitve. Ker je skrbnik izdelka pri nekaterih uporabniških zgodbah sprejemne teste podal zelo ohlapno, smo se pri potrjevanju večkrat obrnili nanj, saj včasih kljub več sestankom nismo bili popolnoma prepričani o ustreznosti rešitve.

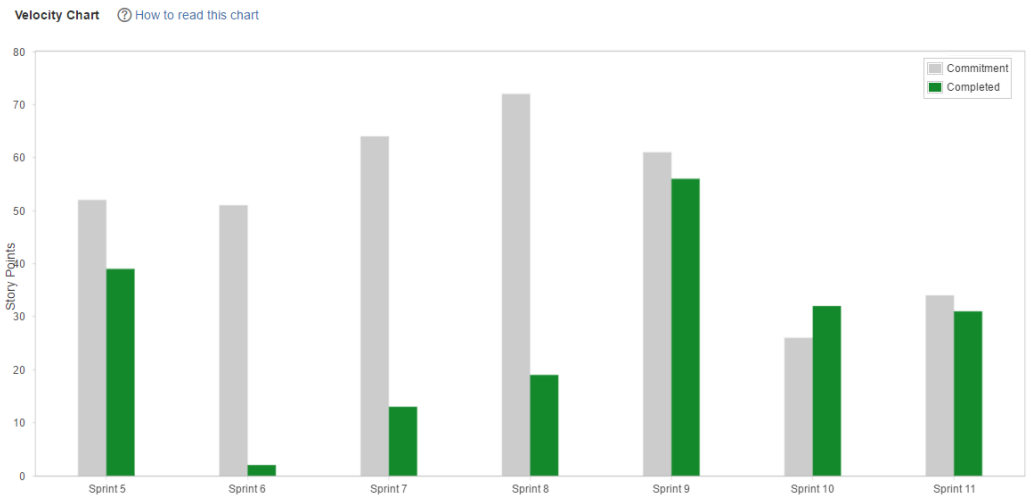
4.9 Sestanki Scrum

Pri izvedbi sestankov Scrum nam je bilo v veliko pomoč majhno število udeležencev v Scrumu, saj smo bili na vseh sestankih vsi udeleženci na voljo.

Pred vsakim sprintom smo izvedli sestanek za načrtovanje sprinta. Ti sestanki so v začetnih sprintih včasih trajali tudi do šest ur, saj še nismo imeli postavljene osnove programa in smo se odločali med različnimi programskimi rešitvami, ki bi jih uporabili pri delu. Pri kasnejših sprintih so bili ti sestanki dolgi največ tri ure, saj smo se lahko osredotočili le na zahteve skrbnika izdelka.

Dnevne sestanke smo na začetku izvajali v času pred malico. Ker smo bili vsi programerji skupaj v istem prostoru, nam dnevni sestanki na začetku niso predstavljali večjega problema. Teh sestankov v četrtem sprintu nismo več izvajali vsakodnevno, ampak smo izvedli le dva ali tri na teden, saj smo se o nalogah dogovarjali med delom.

Ob koncu vsakega posameznega sprinta smo izvedli pregled sprinta in retrospektivo sprinta. Sestanka smo združili v enega, saj smo ob pregledu uporabniških zgodb hkrati analizirali tudi težave, ki so se pojavljale pri izdelavi. Pri prvih sprintih so ti sestanki trajali tudi tri ure, saj smo opazili veliko pomanjkljivosti pri izdelavi projekta kot tudi pri izvajanju metode Scrum. Po treh sprintih, ko smo metodo Scrum dobro spoznali in odpravili napake pri postopku izdelave, so bili ti sestanki krajši.



Slika 4.6: Graf *velocity chart*, ki prikazuje število načrtovanih (siva barva) in realiziranih (zelena barva) točk za zadnjih 7 sprintov. Velike razlike med načrtovanim in realiziranim številom točk so pojasnjene v poglavju 5.2.10.

4.10 Spremljanje napredka

Pri spremljanju napredka projekta nam je bilo v veliko pomoč orodje Jira. Uporabljali smo ga vsi udeleženci v Scrumu. Skrbnika izdelka je zanimal predvsem seznam zahtev s katerim je lahko spremljal, katere uporabniške zgodbe so že izdelane ter koliko jih še ostaja, in je lahko s temi podatki približno ocenil čas izdaje. Skrbnika procesa in razvojno skupino so bolj zanimale analize posameznih sprintov. Spremljali so prikaz hitrosti posameznega sprinta na grafih *burndown chart*, ki v Jiri prikažejo načrtovano in dejansko hitrost razvojne skupine. Spremljali so tudi graf *velocity chart*, ki pokaže razliko med načrtovanim in realiziranim številom točk za zadnjih 7 sprintov in tako omogoča primerjavo med posameznimi sprinti (slika 4.6). Graf omogoča prikaz le zadnjih 7 sprintov v projektu.

Poglavje 5

Analiza izvedbe metode Scrum

Analiza izvedbe metode Scrum je bila izvedena po koncu uporabe metode v podjetju. Zaradi majhnega števila udeležencev pri projektu, s skrbnikom izdelka skupaj 4 člani, so bili z udeleženci opravljeni intervjuji, dodal sem še svoje mnenje.

Od vseh udeležencev, ki so sodelovali v analizi metode Scrum, sta dva že imela izkušnje z drugačnimi metodami dela. Oba sta bila prej zaposlena v drugih podjetjih, v katerih so uporabljali slapovni model. Zame je bilo to prvo srečanje s kakršnokoli metodo dela. S Scrumom nihče od udeležencev ni imel izkušenj.

5.1 Analizirane prakse Scrum

Pri intervjujih sem raziskal naslednje prakse Scrum:

- uporaba uporabniških zgodb,
- ocena uporabniških zgodb z metodo *planning poker*,
- načrtovanje sprinta,
- seznam zahtev produkta,
- dnevni sestanki,

- spremljanje napredka,
- koncept „done“,
- pregled sprinta,
- retrospektiva sprinta in
- vloge Scrum.

Prakse smo ocenjevali s štirimi stopnjami: zelo dobro, dobro, slabo ali zelo slabo. Prakse, ki so bile ocenjene kot dobre ali zelo dobre smo izvajali redno in po pravilih metode Scrum, medtem ko smo prakse, ocenjene kot slabe ali zelo slabe, izvajali redko, nedosledno ali sploh ne.

5.2 Rezultati analize

Rezultati analize (tabela 5.1) kažejo, da izvedba Scruma v našem podjetju ni bila najboljša. Nekatere prakse so sicer bile ocenjene z oceno zelo dobro, kar pomeni, da smo jih redno in pravilno izvajali, a niso bile dovolj, da bi se Scrum obdržal.

5.2.1 Uporaba uporabniških zgodb

Uporabniške zgodbe so bile med udeleženci Scruma dobro sprejete. Najbolje jih je sprejela razvojna skupina, saj je lahko na seznam zahtev dodajala kratke opise funkcionalnosti, po katerih so se pojavile potrebe med delom in jih bo treba realizirati v prihodnosti. Okoliščine teh nalog so se lahko med delom tudi spremenile, a to zaradi načina opisovanja ni bistveno vplivalo na vnesene uporabniške zgodbe. Pri realizaciji uporabniških zgodb je razvojna skupina lahko upoštevala trenutno stanje v aplikaciji in ne okoliščin, ki so veljale ob ustvarjanju zgodbe. Uporabniške zgodbe je na seznam zahtev dodajal tudi skrbnik izdelka, ki pa je imel v prvih treh sprintih težave z določanjem prioritete, saj je vsem uporabniškim zgodbam, ki jih je dodal, določil najvišjo

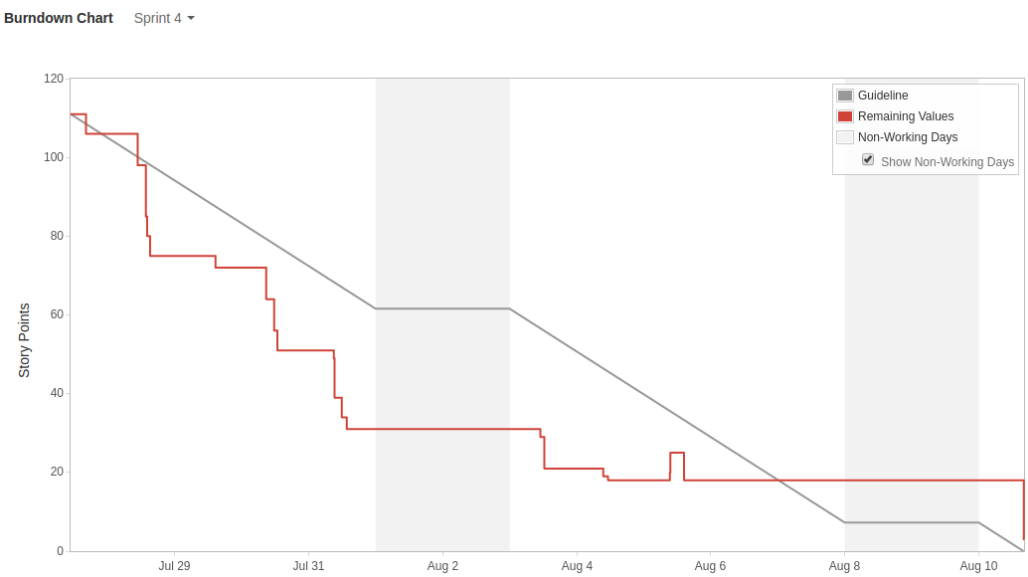
prakse Scrum	ocena
uporaba uporabniških zgodb	dobro
ocena uporabniških zgodb z metodo *planning poker*	zelo dobro
načrtovanje sprinta	zelo dobro
seznam zahtev produkta	slabo
dnevni sestanki	slabo
spremljanje napredka	dobro
koncept „done“	slabo
pregled sprinta	slabo
retrospektiva sprinta	zelo dobro
vloge Scrum	zelo slabo

Tabela 5.1: Tabela ocen praks Scrum, ki smo jih dobili z intervjuji sodelujočih pri projektu Javna razsvetljava.

prioriteto. Na napako ga je skrbnik procesa opozoril in v kasnejših sprintih je uporabniškim zgodbam določil ustreznejše stopnje prioritete.

5.2.2 Ocena uporabniških zgodb z metodo *planning poker*

Praksa ocenjevanja uporabniških zgodb se je pokazala za zelo dobro. Vsi v razvojni skupini smo jo dobro sprejeli in jo redno izvajali. Pomagala nam je najti prevelike uporabniške zgodbe, ki smo jih morali razčleniti na manjše, obvladljivejše uporabniške zgodbe. Po dveh izvedenih sprintih smo se strinjali tudi v ocenah in smo morali le redko izvesti *planning poker* več kot v eni iteraciji. Udeleženec razvojne skupine je o tej metodi povedal: „Že na začetku vidiš, kaj te čaka in za kaj boš potreboval največ časa.“.



Slika 5.1: Potek dobro izvedenega sprinta 4, prikazanega z *burndown chart*. Prikazani sta načrtovana (siva črta) in dejanska (rdeča črta) hitrost.

5.2.3 Načrtovanje sprinta

Sprinte smo redno in z vsakim sprintom uspešneje načrtovali. Pri tem delu nam je bilo v veliko pomoč orodje Jira. Ko smo določili hitrost razvojne skupine, smo lahko z orodjem Jira hitro sestavili seznam uporabniških zgodb, ki smo jih morali realizirati v sprintu. Na sliki 5.1 je prikazan potek sprinta, ki smo ga dobro načrtovali, saj se načrtovan seštevek točk uporabniških zgodb v sprintu le malenkost razlikuje od števila točk dokončanih uporabniških zgodb. To prakso smo uporabljali tudi po opustitvi Scruma.

5.2.4 Seznam zahtev produkta

Seznam zahtev je bila slabše ocenjena praksa, saj je večina udeležencev menila, da je ta seznam nepregleden. Iz ohlapno opisanih uporabniških zgodb ni bilo mogoče takoj razbrati specifikacij produkta, kot smo jih bili vajeni pri slapovnem modelu. Prav tako so se pojavile težave zaradi različnih tipov

elementov na seznamu zahtev, saj je poleg uporabniških zgodb vseboval tudi poročila o hroščih, ki so se pojavili v aplikaciji. Pri razvrščanju elementov na seznam zahtev smo v prvih treh sprintih imeli težave zaradi dodajanja uporabniških zgodb na seznam s strani skrbnika izdelka, opisanega v poglavju 5.2.1. Mnenje enega izmed članov razvojne skupine je bilo: „Seznam zahtev je slab nadomestek za specifikacijo produkta.“.

5.2.5 Dnevni sestanki

Dnevni sestanki so bili slabše izvedena praksa. V prvih treh sprintih smo jih še redno izvajali, v kasnejših pa smo jih zmanjšali na dva ali tri na teden. V razvojni skupini smo menili, da so nepotrebni, saj smo se o napredku pri nalogah in medsebojni pomoči usklajevali sproti. Tako so se nam zdeli vsakodnevni 15-minutni sestanki le potrata časa, ki bi ga lahko porabili za izdelavo nalog.

5.2.6 Spremljanje napredka

Spremljanje napredka sprinta smo zelo dobro sprejeli in je zaradi doslednega vpisovanja trajanja dela delovalo zelo dobro. Omogočilo nam je dobro analizo naše sposobnosti ocenjevanja. O vsakem sprintu nas je zanimalo, koliko dela nas še čaka do konca ter ali nam bo v tem času uspelo opraviti vse izbrane naloge. Včasih smo zaradi dosege cilja sprinta v zadnjih dneh v projekt vložili več dela, kot je bilo sprva načrtovano. Spremljanje napredka produkta je zaradi neprestanega dodajanja novih uporabniških zgodb in pojavljanja hroščev v aplikaciji bilo slabše sprejeto. Projektu ni bilo videti konca, zato se spremljanje napredka celotnega projekta ni veliko uporabljalo. Spremljanje napredka bi v celoti lahko ocenili kot dobro sprejeto prakso.

5.2.7 Koncept „done“

Koncept „done“ smo na začetku zelo dobro definirali, pri izvajanju pa se je zataknilo. Večkrat se je namreč zgodilo, da smo kakšno nalogo ali upo-

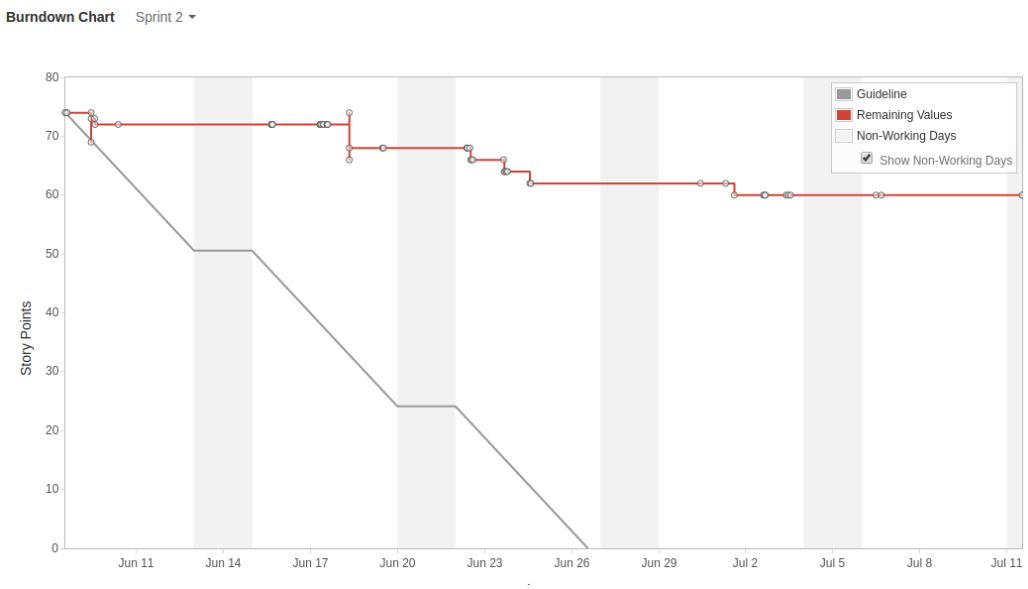
rabniško zgodbo morali nujno dokončati do konca sprinta, zato smo koncept „done“ izvedli pomanjkljivo. Izvedli smo lahko le del preverjanja delovanja, za ostalo je zmanjkalo časa. Zaradi slabega upoštevanja koncepta „done“ se je pojavilo veliko hroščev in tudi koda je bila slabše oblikovana. Mnenje udeleženca je bilo, da je bil koncept „zelo dobro definiran, a se ga nihče ni v celoti držal“.

5.2.8 Pregled sprinta

Sestanke pregleda sprinta smo združevali s sestanki retrospektive sprinta, vendar smo se na teh sestankih bolj osredotočali na izračunavanje hitrosti razvojne skupine ter na odpravljanje napak. Na sestankih sta sodelovala razvojna skupina in skrbnik procesa. Skrbnik izdelka se je udeležil le dveh sestankov. Razlog za slabše izvajanje je bil vpogled skrbnika izdelka v orodje Jira, zaradi česar je lahko že med sprintom videl, katere naloge so dokončane. Ker drugih interesnih skupin ni bilo, smo te sestanke preskočili ali pa so bili le kratek del sestanka retrospektive sprinta.

5.2.9 Retrospektiva sprinta

Retrospektive sprinta smo redno izvajali po vsakem sprintu. Osrednji del teh sestankov je bil določitev dejanske hitrosti razvojne skupine ter primerjava dejanske hitrosti z načrtovano hitrostjo. Načrtovano hitrost smo izračunali tako, da smo sešteli število točk vseh uporabniških zgodb, ki so bile predvidene za izdelavo v tem sprintu. Dejanska hitrost pa je bila izračunana med sprintom, tako da se je zabeležilo število točk izdelanih uporabniških zgodb. Pri spremljanju hitrosti nam je bilo v veliko pomoč orodje Jira, s katerim so bile hitrosti prikazane v grafični obliki (več v poglavju 4.10). Če je bila dejanska hitrost nižja od načrtovane, smo poiskali ovire, ki so povzročile zmanjšanje hitrosti, in načine za odpravo teh ovir.



Slika 5.2: Potek drugega sprinta, prikazanega z *burndown chart*, ki se je izjalovil zaradi dodajanja dodatnih nalog. Prikazani sta načrtovana (siva črta) in dejanska (rdeča črta) hitrost.

5.2.10 Vloge Scrum

Vloge Scrum so bile po mnenju udeležencev v projektu Javna razsvetljava najslabše ocenjena praksa Scruma. Čeprav smo si vloge razdelili in se seznanili s pravili posamezne vloge, smo ta pravila večkrat prekršili. Največ težav sta s svojimi vlogami imela skrbnik izdelka in skrbnik procesa. Skrbnik izdelka je bil direktor našega podjetja, zato skrbnik procesa ni mogel uveljaviti vseh pravil, ki so bila predpisana za njegovo vlogo. Skrbnik izdelka je večkrat presegel svoja pooblastila in razvojni skupini postavljal naloge, ki niso bile del sprinta, ter tako onemogočal uspešno izvajanje Scruma. Ta problem se je ponavljal skozi celotno trajanje projekta, zato so bili drugi, šesti, sedmi in osmi sprint zelo slabo izvedeni (sliki 5.2 in 4.6).

5.3 Skupna ocena

Na podlagi celotne analize bi bilo mogoče ugotoviti, da smo v našem podjetju Scrum sprejeli le polovično. Nekatere prakse, ki bi morale biti redno izvajane, se pri nas niso obnesle zaradi zgoraj naštetih argumentov. Lahko bi sklenili, da smo Scrum nevede izvajali v obliki Scrumbut [13]. Ta oblika predpisuje Scrum, pri katerem se določen del Scruma ne izvaja in vsebuje pojasnilo, zakaj se ne izvaja v celoti ter kako so udeleženci to popravili.

5.4 Razlog za opustitev

Po 11 sprintih smo opustili metodo Scrum. Čeprav je bilo razlogov za opustitev več, je bil največja ovira pri izvedbi Scruma skrbnik izdelka. Pojasnili smo mu vse koncepte Scruma, z njimi se je strinjal, a se jih ni držal. Večkrat je od razvojne skupine zahteval izdelek že pred koncem sprinta ali pa je med sprintom spremenil zahteve uporabniške zgodbe tako, da jih v času sprinta ni bilo več mogoče dokončati. S skrbnikom izdelka se nismo mogli uskladiti, zato smo predlagali, da se Scrum ukine, ker nam je administracija ob nenehno spreminjajočih se zahtevah in vmesnih izdajah vzela preveč časa.

Ob opustitvi metode Scrum smo opustili tudi večino uporabljenih praks. Nismo več izvajali sprintov in sestankov, prav tako smo prenehali z vodenjem seznama zahtev in se vrnili k zahtevam v papirnati obliki. Kljub temu se je nekaj delov obdržalo. Scrum je dober vtis naredil predvsem na razvojno skupino, ki nekatere dele uporablja še danes, čeprav programiranje sedaj poteka po slapovnem modelu. Še vedno smo skupaj v eni pisarni in si pri nalogah pomagamo. Naloge razbijamo na manjše dele in ocenjujemo čas izvajanja nalog, kar nam pomaga pri načrtovanju dela.

5.5 Predlogi za izboljšanje

Če bi v našem podjetju hoteli obdržati metodo Scrum, bi moralo celotno podjetje začeti delati bolj „agilno“, kar bi pomenilo velike spremembe v načinu

dela. Da bi se izognili večjim spremembam, bi lahko uporabili Scrum v obliki Scrumbut.

Scrumbut je metoda, ki izhaja iz metode Scrum, a ne uporablja vseh njenih delov. Namenjena je za primere, ko uporaba metode Scrum razkrije pomanjkljivost, ki ji preprečuje uspešno izvajanje, a je odprava pomanjkljivosti prezahtevna. Scrumbut ohrani te pomanjkljivosti in prilagodi Scrum tako, da se lahko ta kljub temu uspešno izvaja [13].

Dele Scruma, ki se v metodi Scrumbut spremenijo, opišemo v stavkih. Vsak stavek se začne z „Uporabljam Scrum, ampak“, sledita razlog za spremembo prakse ter pojasnilo o spremembi. V našem podjetju so bile nekatere prakse Scruma dobro sprejete, zato bi bilo najbolj smiselno te prakse obdržati. Slabo sprejete prakse bi lahko prilagodili in jih vključili v metodo Scrumbut.

Metoda Scrumbut, ki bi jo lahko izvajali v našem podjetju, bi se glasila:

- Uporabljamo Scrum, ampak ne izvajamo dnevnih sestankov, saj se nam zdijo nepotrebni, zato jih izvajamo dvakrat na teden.
- Uporabljamo Scrum, ampak ne izvajamo sestankov pregleda sprinta, ker jih združujemo s sestanki retrospektive sprinta.
- Uporabljamo Scrum, ampak nam nadrejeni včasih naložijo posebne naloge, zaradi česar nam zmanjka časa za popolno upoštevanje koncepta „done“.
- Uporabljamo Scrum, ampak lahko sprintu dodamo naloge, ki so nujne za izdelavo, kar rešimo s podaljšanjem sprinta za največ dva dni.

Zgoraj opisana metoda rešuje največje težave, na katere smo naleteli pri uporabi Scruma, in bi nam omogočila, da bi Scrum v obliki Scrumbut uporabljali še naprej. Morali pa bi spremljati, ali je ta metoda učinkovita ali nas le upočasnjuje pri delu.

Poglavje 6

Sklepne ugotovitve

Sodelovanje pri uvedbi in izvajanju metode Scrum pri projektu Javna razsvetljava je bilo zelo zanimiva in poučna izkušnja. Videli smo, da ni zagotovila, da bo Scrum vsepovsod uspel in se obdržal. V primeru, ki je opisan v nalogi, se žal ni. Projekt smo kasneje dokončali brez Scruma in ga danes že uporabljajo v nekaterih slovenskih občinah, a sem prepričan, da bi z vztrajanjem pri Scrumu projekt dokončali mesec ali dva prej. Ta projekt je lahko primer, na kaj moramo biti pri uvajanju Scruma pozorni, ter opozorilo, čemu se moramo izogibati, da bi Scrum uporabljali uspešno na dolgi rok. Iz projekta lahko izluščimo splošna priporočila, ki bodo prišla prav vsakomur, ki si želi uporabljati Scrum.

Prvo in najpomembnejše priporočilo je, da je treba Scrum uporabljati dosledno na vseh ravneh organizacije, ne glede na trenutni položaj ali plačni razred. Če le eden udeleženec ne sprejme Scruma, ga lahko tako zaustavi, da postane neuporaben. Zato je pomembno, da se Scruma lotimo celostno in na predpisane položaje postavimo ljudi, o katerih smo prepričani, da bodo svoje naloge opravljali pravilno in vestno.

Drugo pomembno načelo je natančno sledenje navodilom metode Scrum. Tudi če se nam zdi kakšno pravilo nepotrebno ali celo neumno, se pokaže, da bi z doslednim upoštevanjem teh pravil na dolgi rok več pridobili. Bolje je, da pravila upoštevamo že od začetka, kot da kasneje odpravljamo napake, ki

so nastale zato, ker se ne držimo pravil.

Upam, da bodo ta priporočila in moje izkušnje z izvajanjem metode Scrum komu pomagali pri uvedbi metode Scrum in preprečili ponavljanje naših napak.

Sam sem veliko pridobil s sodelovanjem pri Scrumu, saj sem lahko neposredno opazoval kako Scrum deluje v podjetju, in spoznal njegove dobre in slabe lastnosti. Vsaki ekipi priporočam, da Scrum vsaj preskusi, saj prinaša drugačen pristop k delu in bo zagotovo izboljšal potek dela v ekipi.

Literatura

[1] Apache wicket. https://wicket.apache.org/. [Dostopano: 21.11.2016].

[2] Bitbucket. https://bitbucket.org/. [Dostopano: 22.11.2016].

[3] Eclipse. https://eclipse.org/. [Dostopano: 22.11.2016].

[4] Foundation. http://foundation.zurb.com/. [Dostopano: 22.11.2016].

[5] Google Maps API. Dosegljivo: https://developers.google.com/maps/. [Dostopano: 14.11.2016].

[6] Java 8. Dosegljivo: http://www.oracle.com/technetwork/java/javase/overview/java8-2100321.html. [Dostopano: 14.11.2016].

[7] Java 8 what's new. Dosegljivo: http://www.oracle.com/technetwork/java/javase/8-whats-new-2157071.html. [Dostopano: 14.11.2016].

[8] Jira. https://www.atlassian.com/software/jira. [Dostopano: 22.11.2016].

[9] Jira scrum documentation. https://confluence.atlassian.com/agile/glossary/scrum. [Dostopano: 30.11.2016].

[10] Jira task. https://confluence.atlassian.com/agile/glossary/task. [Dostopano: 30.11.2016].

[11] Postgres. https://www.postgresql.org/. [Dostopano: 22.11.2016].

[12] Project lombok. https://projectlombok.org/. [Dostopano: 22.11.2016].

[13] Scrumbut. Dosegljivo: https://www.scrum.org/scrumbut. [Dostopano: 17.11.2016].

[14] Spring. Dosegljivo: https://spring.io/. [Dostopano: 14.11.2016].

[15] Marko Bajec and Marjan Krisper. Agilne metodologije razvoja informacijskih sistemov. *Uporabna informatika, Ljubljana, april, maj, junij*, 2003.

[16] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Manifesto for agile software development. Dosegljivo: http://www.agilemanifesto.org, 2001. [Dostopano: 08.11.2016].

[17] Martijn Dashorst and Eelco Hillenius. *Wicket in action*. Dreamtech Press, 2008.

[18] Ken Schwaber and Jeff Sutherland. The scrum guide. Dosegljivo:http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf, 2016. [Dostopano: 01.11.2016].

[19] Jeff Sutherland. *V gruču do uspeha*. Pasadena, 2016.

[20] Craig Walls. *Spring in Action*. Manning, 2011.

[21] Wikipedia. Agilne metode razvoja programske opreme. Dosegljivo: https://sl.wikipedia.org/w/index.php?title=Agilne_metode_razvoja_programske_opreme&oldid=4658919, 2016. [Dostopano: 01.11.2016].