UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

MAGISTRSKO DELO

ANALIZA METODE RAZVOJA INFORMACIJSKIH REŠITEV V IZBRANEM PODJETJU

Ljubljana, april 2022 VID ROTAR

IZJAVA O AVTORSTVU

Podpisani Vid Rotar, študent Ekonomske fakultete Univerze v Ljubljani, avtor predloženega dela z naslovom Analiza metode razvoja informacijskih rešitev v izbranem podjetju, pripravljenega v sodelovanju s svetovalcem red. prof. dr. Mirom Gradišarjem

I Z J A V L J A M

1. da sem predloženo delo pripravil samostojno;

2. da je tiskana oblika predloženega dela istovetna njegovi elektronski obliki;

3. da je besedilo predloženega dela jezikovno korektno in tehnično pripravljeno v skladu z Navodili za izdelavo zaključnih nalog Ekonomske fakultete Univerze v Ljubljani, kar pomeni, da sem poskrbel, da so dela in mnenja drugih avtorjev oziroma avtoric, ki jih uporabljam oziroma navajam v besedilu, citirana oziroma povzeta v skladu z Navodili za izdelavo zaključnih nalog Ekonomske fakultete Univerze v Ljubljani;

4. da se zavedam, da je plagiatorstvo – predstavljanje tujih del (v pisni ali grafični obliki) kot mojih lastnih – kaznivo po Kazenskem zakoniku Republike Slovenije;

5. da se zavedam posledic, ki bi jih na osnovi predloženega dela dokazano plagiatorstvo lahko predstavljalo za moj status na Ekonomski fakulteti Univerze v Ljubljani v skladu z relevantnim pravilnikom;

6. da sem pridobil vsa potrebna dovoljenja za uporabo podatkov in avtorskih del v predloženem delu in jih v njem jasno označil;

7. da sem pri pripravi predloženega dela ravnal v skladu z etičnimi načeli in, kjer je to potrebno, za raziskavo pridobil soglasje etične komisije;

8. da soglašam, da se elektronska oblika predloženega dela uporabi za preverjanje podobnosti vsebine z drugimi deli s programsko opremo za preverjanje podobnosti vsebine, ki je povezana s študijskim informacijskim sistemom članice;

9. da na Univerzo v Ljubljani neodplačno, neizključno, prostorsko in časovno neomejeno prenašam pravico shranitve predloženega dela v elektronski obliki, pravico reproduciranja ter pravico dajanja predloženega dela na voljo javnosti na svetovnem spletu preko Repozitorija Univerze v Ljubljani;

10. da hkrati z objavo predloženega dela dovoljujem objavo svojih osebnih podatkov, ki so navedeni v njem in v tej izjavi.

V Ljubljani, dne _____ Podpis študenta: _____

KAZALO

KAZALO SLIK

SEZNAM KRATIC

angl. – angleško
FDD – (angl. Feature-Driven Development); Funkcionalno vodeni razvoj
XP – (angl. Extreme Programming); Ekstremno programiranje
DSDM – (angl. Dynamic Systems Development Method); Dinamični sistemi razvoja metode

UVOD

Široko sprejetje agilnih metod je močno spremenilo svet razvoja programske opreme. Agilna metoda scrum je postala izredno razširjena v omenjeni panogi. Jasno je tudi, da v praksi končna implementacija scruma zaradi različnih razlogov redko upošteva priporočena vodila (Eloranta, Koskimies & Mikkonen, 2016).

V agilnem svetu scruma je običajno, da se načina dela ne določi vnaprej, temveč se velik del prepusti sami razvojni ekipi, ki se odloči za to metodo, to pa zaradi tega, ker se po njej najbolje rešuje podane probleme (Cohn, 2019).

Scrum je pomembna strategija pri razvoju programske opreme. Pogosto pa se dogaja, da se pred razvijalci pojavijo problemi, ki jim onemogočijo uresničiti dobre rešitve ali pa jim podaljšajo čas za zaključek nalog. Literatura razvijalcem predlaga veliko »najboljših praks«, s katerimi si lahko pomagajo. Večinoma so omenjena vodila zelo teoretična in se jih v nekaterih organizacijah ne more upoštevati (Morandini, Coleti, Oliveira & Corrêa, 2021).

V agilnem procesu razvoja programske opreme je najpomembneje ustvariti produkt z največjo poslovno vrednostjo. Za dosego takega cilja se posamezne naloge razporedi po njihovih prioritetah. Agilne metode, kot je na primer scrum, opredeljujejo vodila za razvrstitev nalog (Jarzębowicz & Sitko, 2020).

Scrum je v razvoju programske opreme ustvaril vlogo skrbnika metode. Ta je predstavljen kot pomočnik in vodja, ki na različne načine pomaga razvojni ekipi. Nekateri načini od njih so promoviranje scruma, organizacije sestankov ter odstranjevanje različnih ovir (Shastri, Hoda & Amor, 2021). Ekipa, v kateri delujem, je sestavljena iz sedmih razvijalcev, skrbnika izdelka in skrbnika metode. Izkazalo se je, da je ekipa z vidika organizacij sestankov in obvladovanja različnih ovir pri delu zelo samostojna. Zaradi tega se ji je zastavilo vprašanje glede učinkovitosti vloge skrbnika metode.

Med delom v izbranem podjetju smo kot ekipa opazili, da trenutno uporabljena oblika scruma še ni dobro prilagojena produktu, ki ga razvijamo. Ena izmed težav je v tem, da opravljeno delo nosilcem interesov ni povsem razvidno. Zunanji opazovalci namreč veliko opravljenega dela razvijalcev trenutno ne vidijo. Glede tega je potrebno najti ustrezno rešitev. Poleg tega nekatere težavnosti naloge niso dobro ocenjene, kar se posledično kaže z zamudami pri opravljanju nalog. Pokazala se je tudi možnost za boljše organiziranje dnevnih in tedenskih sestankov, saj nekateri sestanki trajajo predolgo, kar pa zmanjšuje učinkovitost razprave.

Namen magistrskega dela je izboljšati trenutni proces razvoja informacijskih rešitev po metodi scrum.

Cilj magistrskega dela je podati predlog prilagojene metode scrum za potrebe izbrane razvojne ekipe. Predlagana metoda bo izboljšala trenutni način dela v ekipi.

Metoda dela v teoretičnem delu je pregled relevantne literature v zvezi z agilnostjo in agilno metodo scrum. Pregledal sem literaturo, v kateri so bili opisani različni pristopi uvajanja prilagojenih oblik te metode. V empiričnem delu magistrskega dela sem sodeloval v razvojni ekipi v izbranem podjetju. Med našim delovanjem sem lahko prepoznal glavne pomanjkljivosti trenutne metode. Z analizo obstoječega in željenega stanja, analizo literature in ob sodelovanju članov razvojne ekipe sem po sintezi predlagal rešitve za trenutne pomanjkljivosti.

Magistrsko delo je sestavljeno iz štirih poglavij. V prvem sta po literaturi narejena pregled agilnosti ter primerjava različnih agilnih metod. Nato je predstavljena metoda scrum, ki jo analiziram v izbranem podjetju.

Drugo poglavje opisuje orodja, s katerimi si lahko ekipe pomagajo za boljše izvajanje metode scrum. Opisana orodja so predlagana tudi v kasnejših izboljšavah trenutno izvajane metode.

V tretjem poglavju je najprej predstavljena analiza trenutne implementacije metode scrum v izbranem podjetju. Opisana je trenutna struktura tekov (sprintov). Poleg tega so podrobno opisani vsi sestanki znotraj posameznega teka. Zatem so predstavljene glavne pomanjkljivosti trenutne metode.

V četrtem poglavju je predstavljena predlagana izboljšana metoda scrum. Za vsako izmed ugotovljenih pomanjkljivosti je predlagana ustrezna izboljšava. Predlagane izboljšave so rezultat proučene literature različnih implementacij metode scrum in opazovanja načina dela v ekipi izbranega podjetja.

1 RAZVOJ PROGRAMSKE OPREME PO AGILNI METODI SCRUM

1.1 Agilnost

Za razumevanje agilnih metod je dobro najprej poznati pomen besede agilnost. »Agile Alliance« razloži agilnost kot »zmožnost ustvarjanja in odzivanja na spremembe. Je način uspešnega delovanja znotraj negotovega in turbulentnega okolja.«

Ideja agilnosti se je začela v letu 2001 z »Manifestom agilnega razvoja programske opreme«. Skupina 17 razvijalcev se je v ameriški zvezni državi Utah zbrala z namenom identificirati

in uvesti nov način razvoja programske opreme (Appelbaum, 2021). Vsi člani skupine so se strinjali, da je imel trenutni razvoj programske opreme preveč pomanjkljivosti. Glavni problem je bil delovanje podjetij na način, pri katerem je bila prevelika teža na nenehnem načrtovanju in dokumentiranju lastnih procesov razvoja programske opreme. Zaradi tega je podjetje spregledalo ključno stvar za uspešno poslovanje – ugajanje svojim strankam.

Podjetja so prodajala svoje storitve na podlagi vrednot, kot so odličnost in celovitost. Težava pri tem je bila, da omenjene vrednote niso bile učinkovita vodila, še posebej pri usmerjanju boljšega opravljanja dela razvijalcev programske opreme. Kot zasnovo rešitve podanih problemov je omenjena skupina 17 razvijalcev ustvarila »Manifest agilnega razvoja programske opreme«. Ta dokument, napisan v 68 besedah, je za vedno spremenil način razvoja programske opreme. Napisani principi in vrednote dokumenta so bili do današnjega dne obširno sprejeti (Atlassian, 2021).

12 principov agilne programske opreme

Zdi se, da je današnje agilno okolje poplavljeno z metodami, ki obljubljajo uresničitev agilnih idealov v realnem svetu. Toda trenutna številčnost metod ni nič povsem novega.

Manifest agilnega razvoja programske opreme je bil ustvarjen v želji po iskanju skupnih točk med scrumom, ekstremnim programiranjem (angl. Extreme Programming), Crystal Clearom in drugimi ogrodji. Sestavljen je na podlagi skupnih vrednot omenjenih ogrodij, kot so (Atlassian, 2021):

- **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.

Drugače rečeno: četudi cenimo dejavnike na desni, vseeno bolj cenimo tiste na levi (Agilemanifesto.org, 2001).

Poleg omenjenih skupnih vrednot so na sestanku določili tudi dvanajst principov razvoja programske opreme, ki dodatno obrazložijo njihov pomen (Atlassian, 2021). Principi v ozadju agilnega manifesta so (Agilemanifesto.org, 2001):

- Naša najvišja prioriteta je zadovoljiti stranko z zgodnjim in nepretrganim izdajanjem vredne programske opreme.
- Sprejemamo spremembe zahtev celo v poznih fazah razvoja. Agilni procesi vključijo tovrstne spremembe v prid konkurenčnosti naše stranke.
- Delujočo programsko opremo izdajamo pogosto, znotraj obdobja nekaj tednov do nekaj mesecev, z dajanjem prednosti krajšemu časovnemu okviru.

- Poslovneži in razvijalci morajo skozi celoten projekt dnevno sodelovati.
- Projekte gradimo okoli 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 nje 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 nenehnega 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 obdobjih ekipa išče načine, kako postati učinkovitejša ob rednem prilagajanju svojega delovanja.

Do današnjega dne se je manifest agilnega razvoja programske opreme ohranil z minimalnimi spremembami. Na drugi strani pa je današnji svet agilnosti glede na njegovo začetno stanje povsem drugačen (Atlassian, 2021).

1.2 Vrste agilnih metod

Agilne metode so ogrodja, s katerimi ekipe in organizacije lahko v svojih projektih izvajajo agilno miselnost. Denimo, če agilnost predstavlja kaj storiti, agilne metode predstavljajo kako to storiti. Obstaja veliko agilnih metod, ki jih podjetja lahko uporabijo za učinkovito širjenje svoje agilnosti znotraj organizacije. Najbolj priljubljene so naslednje:

- scrum,
- ekstremno programiranje,
- kanban,
- Crystal,
- funkcionalno vodeni razvoj (angl. Feature-Driven Development – FDD),
- dinamični sistemi razvoja metode (angl. Dynamic Systems Development Method – DSDM).

Zanimivo je, kako so si omenjene agilne metode med seboj podobne in v čem so si različne (Appelbaum, 2021).

1.2.1 Scrum

Scrum je ena izmed najbolj priljubljenih agilnih metod. Ogrodje je izredno učinkovito pri upravljanju ponavljajočih se in postopnih projektih. V omenjeni metodi skrbnik izdelka

določi seznam prednostnih nalog, ki jih mora razvojna ekipa opraviti. Ekipa mora na koncu določenega dvo- do štiritedenskega teka uvesti funkcionalno programsko opremo.

Metoda scrum je zaradi svoje preprostosti med agilnimi ekipami izredno priljubljena. Omogoča jim prepoznati probleme že v zgodnjih fazah projekta, poleg tega pa vzpodbuja tesna sodelovanja. Omenjeno metodo bom bolj podrobno predstavil v nadaljevanju (ScrumExplainer, 2020).

1.2.2 Ekstremno programiranje

Ena izmed bolj priljubljenih agilnih metod je ekstremno programiranje (angl. Extreme Programming – XP). Znana je po svojem poudarku na hitrosti in neprekinjeni dostavi programske opreme. Kot pri metodi scrum ekstremno programiranje omogoča tesno sodelovanje znotraj ekipe z namenom dostavljanja delujoče programske opreme od enega do treh tednov. Metoda se zanaša na strankino sodelovanje glede najpomembnejših željenih funkcionalnosti programske opreme. Poleg tega pa se zanaša na razvojno ekipo, da omenjene strankine želje učinkovito uresniči v produktu.

Ekstremno programiranje je priporočena metoda pri manjših razvojnih ekipah, ki so vešče te metode. Poleg tega mora biti ekipa sposobna dobro komunicirati s strankami, ki ne poznajo dobro informacijskih tehnologij (Beck, 1999).

1.2.3 Vitek razvoj programske opreme

V primerjavi z metodama scrum in ekstremno programiranje je vitek razvoj programske opreme (angl. Lean Software Development) bolj preprost, saj vsebuje manj stroge smernice in pravila. Metoda se je razvila sredi dvajsetega stoletja. Temelji na principih, ki zagotavljajo koristnost in uspešnost proizvodnje. Nato se je preobrazila v metodo razvoja programske opreme. Ta temelji na petih principih upravljanja:

1. identificirati vrednost,
2. opredeliti tok vrednosti,
3. ustvariti nepretrgan potek dela,
4. ustvariti vlečni (angl. pull) sistem,
5. nenehno izboljševati.

Vitek razvoj programske opreme še posebej poudarja pomembnost zmanjševanja »odpadkov«. Pri razvoju programske opreme se to odraža v zmanjševanju časa, izgubljenega z neproduktivnimi nalogami, učinkovitem izkoriščanju delovnih virov, omogočanju ekipi in

posameznikom pomembneje odločati in razvrščati naloge, ki prinesejo največjo vrednost (Poppendieck & Cusumano, 2012).

1.2.4 Kanban

Kanban se poleg drugih agilnih metod osredotoča na medsebojno pomoč ekipam in omogoča stalno dostavljanje kvalitetne programske opreme. Omenjena metoda je edinstvena zaradi svoje vizualne predstavitve, s katero lahko učinkovito vodimo ustvarjanje produkta.

Kanban temelji na šestih temeljnih dejavnostih:

1. vidna predstavitev poteka dela,
2. omejitev trenutnega dela,
3. upravljanje poteka dela,
4. jasna določitev procesnih pravil,
5. uvedba zanke povratnih informacij,
6. izboljšanje medsebojnega sodelovanja.

Kanban dosega zgoraj omenjene dejavnosti s pomočjo kanban table. Prikazuje jo slika 1. Omenjeno orodje se uporablja za razdeljevanje nalog v stolpce, ki predstavljajo nedokončane naloge (Narediti), trenutno aktivne (V delu) ter že opravljene (Opravljeno). Kanban je agilna metoda, ki izboljša medsebojno sodelovanje in učinkovitost, poleg tega pa omogoča izbiro najbolj učinkovitega poteka dela (Ahmad, Markkula & Oivo, 2013).

Slika 1: Kanban tabla

Prirejeno po Birade (2020).

1.2.5 Crystal

Agilna metoda Crystal se bolj osredotoča na interakcije med ljudmi, na njihove spretnosti, orodja in tehnike razvoja.

Crystal razvršča projekte glede na tri določene kriterije:

1. velikost ekipe,
2. kritičnost sistema,
3. projektne prioritete.

Način dela je glede na druge agilne metode zelo podoben. Velik poudarek je na neprekinjenem dostavljanju programske opreme, veliki vključenosti uporabnikov in izničitvi neproduktivnih aktivnosti. Zaradi predpostavke, da je vsak projekt unikaten, je metoda Crystal postala znana kot ena izmed najbolj fleksibilnih agilnih metod (Anwer, Aftab, Waheed & Muhammad 2017).

1.3 Predstavitev metode scrum

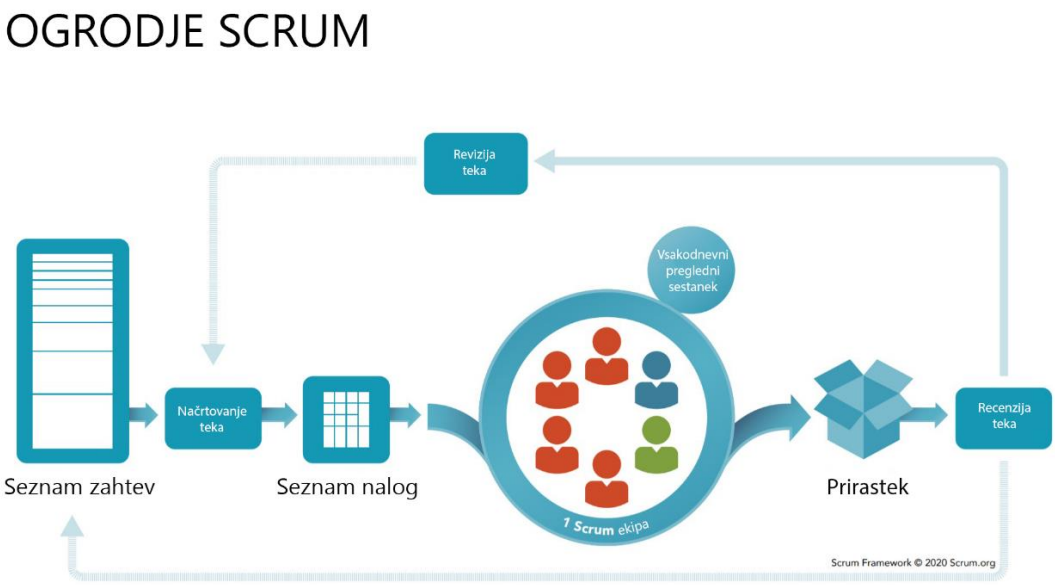
Scrum je ogrodje, ki ekipam omogoča razvoj in vzdrževanje kompleksnih produktov. Medtem ko se ogrodje najpogosteje uporablja za ekipe pri razvoju programske opreme, se lahko njegove smernice in principe uporablja tudi v različnih drugih okoljih. Prav zaradi tega je scrum trenutno tako priljubljena metoda (Atlassian, 2021).

Scrum od skrbnika metode (angl. Scrum Master) zahteva, da znotraj ekipe ustvarja okolje, kjer (Scrum Guide, 2020):

1. skrbnik izdelka v seznamu zahtevkov določi potrebno delo za kompleksni problem,
2. scrum ekipa med tekom izbrano delo vgradi v novonastalo dodano vrednost produkta,
3. scrum ekipa in nosilci interesov pregledajo rezultate in pripravijo naloge za naslednji tek,
4. se ponovijo vsi koraki od začetka.

Opisani proces prikazuje slika 2, podrobnosti pa bodo opisane v naslednjih poglavjih.

Slika 2: Proces metode scrum



Prirejeno po Scrum.org (2016).

1.4 Vloge v scrumu

Scrum ima tri vloge: člana ekipe, ki opravlja delo, skrbnika metode, ki ekipi pomaga delo bolje opravljati, in skrbnika izdelka, ki se odloči, kakšno naj bo delo. Skrbnik metode je zadolžen za način dela, skrbnik izdelka pa za njegovo vsebino (Southerland, 2016).

1.4.1 Skrbnik izdelka

Celotno podpoglavje je povzeto po poglavju »Skrbnik izdelka« (angl. Product Owner) v knjigi (Southerland, 2016). Avtor knjige je predstavil štiri glavne lastnosti, potrebne za skrbnika izdelka:

Skrbnik izdelka mora dobro poznati področje dela. To pomeni, da mora proces, ki ga izvaja ekipa, dovolj dobro razumeti, da ve, kaj člani ekipe lahko dosežejo in česa ne morejo, kar je enako pomembno. Prav tako mora dobro poznati vsebino dela, da jo lahko prenese v dejansko vrednost. Skrbnik izdelka mora dovolj dobro poznati trg, da ve, kaj stranke potrebujejo.

Druga lastnost je, da mora imeti ustrezna pooblastila za sprejemanje odločitev. Tako kot uprava ne bi smela posegati v delo ekipe, mora imeti skrbnik izdelka manevrski prostor za odločanje o viziji izdelka in korakih, ki so potrebni, da se vizija uresniči. Ker je skrbnik izdelka pod pritiskom notranjih in zunanjih interesnih skupin, mora imeti dovolj avtoritete,

da se jim po potrebi lahko upre. Odgovoren je za rezultate, hkrati pa mora imeti možnost sprejemanja odločitev.

Skrbnik izdelka mora biti ekipi na voljo, da ji pojasni, kaj je treba storiti in zakaj. Čeprav je on tisti, ki je odgovoren za seznam nalog, mora med njim in ekipo potekati nenehen dialog. Strokovno znanje ekipe pogosto vpliva na njegovo odločitev. Biti mora zanesljiv, dosleden in na razpolago, saj brez njega člani ekipe ne bi vedeli, kaj naj počnejo in v kakšnem vrstnem redu naj se tega lotijo. Na skrbnika izdelka se zanašajo glede vizije in znanja o tem, kaj je na trgu pomembno. Če skrbnik izdelka ekipi ni na razpolago, lahko celoten proces propade. Zaradi tega je običajno odsvetovano, da bi to vlogo prevzeli direktorji ali vodje oddelkov, saj večinoma nimajo časa, ki je za to potreben.

Zadnja lastnost je, da je skrbnik izdelka odgovoren za vrednost. V poslovnem svetu se uspeh meri v obliki donosa. Uspeh skrbnika izdelka se meri s prihodkom, ki ga je ustvaril za vsako točko dela. Če je hitrost ekipe štirideset točk dela na teden, se izmeri prihodek, ki ga je prinesla vsaka točka. Merilo je lahko tudi uspeh ekipe. Ključnega pomena je, da se določi merilo uspeha in da se odgovornost za njegovo povečanje prenese na skrbnika izdelka. Pri scrumu je omenjenim meritvam preprosto slediti, saj je metoda izredno pregledna.

Vloga skrbnika izdelka je včasih za eno osebo prezahtevna, zato imajo lahko večji projekti običajno ekipo skrbnikov izdelka.

1.4.2 Skrbnik metode

Scrum je v razvoju programske opreme ustvaril vlogo skrbnika metode (angl. Scrum Master). Ta je predstavljen kot pomočnik in vodja, ki na različne načine pomaga razvojni ekipi, npr. s promoviranjem scruma, organizacijo sestankov, odstranjevanjem različnih ovir (Shastri, Hoda & Amor, 2021).

Skrbnik metode je odgovoren za učinkovitost ekipe. Omogoča ji, da lahko izboljša svoje prakse v okviru scruma. Je pravi voditelj, ki služi ekipi in celotni organizaciji.

Skrbnik metode služi **ekipi** na več načinov, med drugim:

- usposablja člane ekipe pri samoupravljanju in navzkrižnih funkcionalnostih,
- pomaga ekipi, da se osredotoči na ustvarjanje dodanih vrednosti produkta, ki ustrezajo definiciji končanega,
- odpravlja ovire za napredek ekipe,
- zagotavlja, da se vsi scrum dogodki odvijajo in so pozitivni, produktivni ter ne presegajo določenega časa.

Skrbnik metode služi **skrbniku izdelka** na več načinov, med drugim:

- pomaga pri iskanju tehnik za učinkovito opredelitev ciljev izdelka in obvladovanje seznama zahtevkov,
- pomaga scrum ekipi, da razume potrebo po jasnih in jedrnatih zahtevkih v seznamu zahtevkov,
- pomaga pri vzpostavitvi empiričnega načrtovanja izdelkov v kompleksnih okoljih,
- olajšuje sodelovanje z nosilci interesov, kadar je to zahtevano ali potrebno.

Skrbnik metode služi **organizaciji** na več načinov, med drugim (Scrum Guide, 2020):

- vodi, usposablja in trenira organizacijo pri njenem uvajanju v scrum,
- načrtuje in svetuje pri izvajanju scruma v organizaciji,
- pomaga zaposlenim in nosilcem interesov pri razumevanju in uvedbi empiričnega pristopa k razvoju in reševanju kompleksnih problemov,
- odstranjuje ovire med nosilci interesov in scrum ekipami.

1.4.3 Ekipa

Ekipa (programerji) je po scrumu sestavljena iz programerjev, ki opravljajo delo. Člani ekipe morajo imeti sposobnosti, ki so potrebne za uresničevanje vizije skrbnika izdelka. Ekipe naj bi bile majhne, najbolje med 3 in 9 članov (Southerland, 2016).

Posebne veščine, ki jih potrebujejo razvijalci, so pogosto široke in se razlikujejo glede na področje dela. Razvijalci so odgovorni za (Simplilearn, 2020):

- ustvarjanje načrta za tek – torej seznam zahtevkov,
- ustvarjanje kakovostnih izdelkov z upoštevanjem definicije končanega,
- vsakodnevno prilagajanje za namen uresničitve cilja teka,
- medsebojno zaupanje v ekipi.

1.5 Sestanki po metodi scrum

Sestanki so po metodi scrum sestavni del delovnega okolja. Veljajo za neprecenljiv vir zbiranja informacij in povratnih informacij razvojne ekipe ter pomagajo pri usklajevanju ekipe s cilji teka.

Obstaja pet vrst sestankov po metodi scrum. Odvijajo se v določenem omejenem času med tekom. Vsak sestanek služi svojemu namenu (nTask, 2018).

1.5.1 Sestanek za načrtovanje teka

Na začetku vsakega teka (Sprinta) se celotna ekipa zbere na sestanku za njegovo načrtovanje z namenom dogovora o izpeljavi naslednjih nalog v seznamu zahtevkov. Skrbnik izdelka je zadolžen za predstavitev nalog, ki so za nosilce interesov najbolj pomembne. Programerji so zadolženi za izbiro količine dela, katerega mislijo, da ga bodo lahko opravili znotraj teka. Pri tem morajo biti pozorni, da ne prevzamejo preveč zahtevkov, saj to lahko vodi v neuspešno končanje teka. Ekipa iz seznama zahtevkov potegne zahtevke v seznam nalog.

Ko se ekipa loti kompleksnega dela, za katerega je značilna negotovost dokončanja, mora intuitivno izbrati količino zahtevkov glede na prejšnje, že dokončane teke. Načrtovanje dela v obliki delovnih ur in primerjava ocen zahtevkov z njihovim dejanskim opravljenim delom poda lažen občutek natančnosti samega načrtovanja. Če je delo resnično predvidljivo, bi morali takšne prakse zavreči v prvih nekaj tekih ali pa se jim popolnoma izogniti.

Dokler se ekipa ne nauči, kako v vsakem teku dokončati potencialno povečanje funkcionalnosti sistema v razvoju, bi morala zmanjšati obseg načrtovanih del. Neuspeh pri spreminjanju starih navad vodi do tehničnega dolga in morebitne smrti projekta. Če seznam zahtevkov še ni izpopolnjen, je lahko del sestanka za načrtovanje teka porabljen za izpopolnitev zahtevkov. Na koncu sestanka ekipa določi, kdo bo opravil katero delo. Izbrane zahtevke se lahko tudi razbije na manjše podzahtevke (James & Walter, 2010).

1.5.2 Vsakodnevni pregledni sestanek

Razvojna ekipa se vsak dan ob istem času in na istem mestu dobi na preglednem sestanku (angl. Daily meeting). Ta traja največ 15 minut. V tem času se pregleda napredek pri doseganju cilja teka in izdela dnevni načrt dela. Člani ekipe si povejo, kaj so prejšnji dan storili, da bi pomagali doseči cilj teka. Poleg tega vsak predstavi svoj dnevni delovni načrt in s kakšnimi ovirami se srečuje.

Pravilo, da mora vsak član ekipe, ki poroča o svojem trenutnem delu, vstati, pripomore k temu, da je poročanje kratko. O temah, ki zahtevajo dodatno pozornost, lahko razpravljajo tisti, ki jih tema zanima, ampak šele po poročanju vseh članov ekipe.

Za ekipo je zelo koristno, da vzdržuje trenutni seznam zahtevkov in graf napredovanja. Med izvajanjem teka je običajno, da se odkrijejo dodatne naloge, potrebne za doseganje cilja teka. Ovire, ki jih povzročajo težave, na katere ekipa nima vpliva, se štejejo za organizacijske ovire.

Vsakodnevni pregledni sestanek je namenjen prekinjanju starih navad samostojnega dela. Na primer gledanje le skrbnika metode med poročanjem je eden izmed simptomov, da se ekipa ni naučila delovati kot samoorganizirana skupina (Obergfell, 2019).

1.5.3 Sestanek za recenzijo teka

Na koncu vsakega teka se organizira sestanek za njegovo revizijo (angl. Sprint Review Meeting). Na sestanku ekipa vsem, ki jih to zanima, še posebej zunanjim nosilcem interesov, prikaže napredek v funkcionalnosti izdelka. Prikaz napredka naj bi bil v obliki predstavitve funkcionalnosti in ne kot poročilo.

Skrbnik izdelka pregleda zahtevke, izbrane na sestanku za načrtovanje teka, in pojasni, kateri zahtevki se štejejo za dokončane. Na primer programska oprema, za katero je napisana zgolj koda, se šteje za nedokončano, saj netestirane programske opreme ni mogoče izdati. Nepopolni zahtevki se vrnejo v seznam zahtevkov in se razvrstijo glede na prioritete skrbnika izdelka kot kandidati za prihodnje teke.

Skrbnik metode pomaga skrbniku izdelka in nosilcem interesov, da svoje povratne informacije pretvorijo v nove zahtevke. Te nato prioritizira skrbniku izdelka. Odkrivanje novih obsegov funkcionalnosti pogosto prehiteva stopnjo razvoja ekipe. Če skrbnik izdelka meni, da je novo odkriti obseg funkcionalnosti pomembnejši od prvotnih pričakovanj, bo novi obseg dobil prednost pri izvedbi. Normalno je, da zaradi tega nekateri zahtevki nikoli ne bodo izvedeni.

Sestanek za revizijo teka je pomemben tako za nosilce interesov kot tudi končne uporabnike aplikacij. To je priložnost za pregled in prilagajanje izdelka med procesom njegovega razvijanja. Nove izdelke, zlasti izdelke programske opreme, si je težko predstavljati, ne da bi videl vsaj del njihove izvedbe. Številne stranke se morajo znati odzvati na del delujočega izdelka, da odkrijejo, kaj si bodo pravzaprav želele. Ponavljajoči se razvoj programske opreme, ki temelji na ustvarjeni dodani vrednosti, omogoča razvoj izdelkov, za katere vsi načrti niso vnaprej točno določeni (Knowledgehut, 2021).

1.5.4 Sestanek za revizijo teka

Ko ekipa pokaže, kaj je med tekom dosegla, koliko zahtevkov je dokončala in koliko izdelkov lahko posreduje strankam in nosilcem interesov za njihovo oceno, se vsi člani sestanejo in razmislijo o stvareh, ki so bile uspešne, in takih, ki bi lahko potekale bolje, ter izboljšavah, ki jih lahko uvedejo v prihodnjem teku.

Da bi bili učinkoviti, zahteva omenjeni sestanek za revizijo teka (Sprint Retrospective Meeting) določeno stopnjo čustvene zrelosti in zaupanja med člani. Pomembno je, da se ekipa ne osredotoča na iskanje krivca, ampak na ocenjevanje procesa. Člani kot ekipa morajo prevzeti odgovornost za proces in rezultate ter kot ekipa poiskati rešitve. Imeti morajo pogum, da stvari, ki jih motijo, omenijo na način, ki se namesto na krivdo osredotoča na napredek. Drugi člani morajo zrelo sprejeti kritiko in namesto izgovorov ponuditi rešitve.

Ob koncu srečanja morajo člani ekipe in skrbnik metode doseči dogovor o izboljšavi procesa, ki jo bodo uporabili pri naslednjem teku. Omenjena izboljšava mora biti zabeležena v seznamu zahtevkov naslednjega teka, ekipa pa jo mora ustrezno preizkusiti. Tako člani ugotovijo, ali so izboljšavo uspešno uvedli in kakšen učinek je imela na hitrost ekipe (Southerland, 2016).

1.5.5 Sestanek za izpopolnitev zahtevkov

Večino zahtevkov je potrebno na začetku izpopolniti, saj so preveliki in nejasno napisani. Čeprav sestanek za izpopolnitev zahtevkov ni obvezen, je izpopolnjevanje zahtevkov še posebej potrebno. Za ekipo je zelo koristno, da si znotraj vsakega teka vzame nekaj časa za to aktivnost. Vsi člani ekipe se na sestanku zberejo in pripravijo seznam zahtevkov za naslednje sestanke za načrtovanje tekov.

Na sestanku za izpopolnitev zahtevkov (angl. Sprint Refinement Meeting) se velike nejasne zahtevke razbije na manjše, bolj razumljive dele glede na poslovne in tehnične lastnosti. Včasih se le del ekipe vnaprej dobi s skrbnikom izdelka in drugimi nosilci interesov ter izpopolni zahtevke, preden v to vključi celotno ekipo.

Med izpopolnjevanjem zahtevkov ekipa oceni, koliko truda bi porabili za njihovo dokončanje. Poleg tega ekipa posreduje druge tehnične podatke o zahtevkih, ki skrbniku izdelka pomagajo določiti njihovo prednost v seznamu zahtevkov. Izkušeni skrbnik metode lahko pomaga ekipi pri prepoznavanju ključnih delov funkcionalnosti, ki imajo največjo poslovno vrednost, hkrati pa spodbuja strogo opredelitev pojma »opravljeno«, ki vključuje ustrezno testiranje in predelavo.

Običajno se zahtevke zapisuje v obliki zgodb (angl. User Stories). Prevelike zgodbe se imenujejo epika (angl. Epic). Tradicionalni pristopi v razvoju programske opreme razdelijo funkcionalnosti v »horizontalne« naloge, ki jih ne morejo dobro razvrstiti po prednosti, nosilci interesov pa ne vidijo njihove poslovne vrednosti. Takih pristopov se je težko odvaditi.

Agilnost na drugi strani zahteva učenje razdeljevanja velikih epik v uporabniške zgodbe, ki predstavljajo zelo majhne funkcionalnosti. Dober primer epike je npr. v aplikaciji za

zdravstvene kartoteke »prikaži zdravniku celotno vsebino pacientovih alergijskih zapisov«. Iz tega se ustvari uporabniško zgodbo: »prikažite, ali obstajajo zapisi o alergijah«. Čeprav so inženirji pričakovali pomembne tehnične izzive pri razčlenjevanju notranjih vidikov evidenc o alergijah, je bila sama prisotnost ali odsotnost alergije najpomembnejša stvar, ki bi jo morali vedeti zdravniki. Sodelovanje med stranko in ekipo pri razdelitvi te epike je prineslo uporabniško zgodbo, ki predstavlja 80 % poslovne vrednosti za le 20 % truda začetno načrtovane celotne epike.

Ker večina strank ne uporablja večine funkcionalnosti izdelkov, je pametno epike razdeliti tako, da se najprej implementira najpomembnejše zahtevke. Celotni opisani proces obsega seznam zahtevkov, obdelavo tega seznama ali zgodb (James & Walter, 2010).

1.6 Scrum artefakti

Scrum artefakti predstavljajo delo ali vrednost. Zasnovani so tako, da najbolje poudarijo preglednost ključnih informacij. Z njihovo pomočjo imajo vsi, ki jih pregledajo, enako podlago za prilagoditev svojega dela.

Vsak artefakt vsebuje zavezo, da bo zagotovil informacije, ki povečujejo preglednost in osredotočenost, na podlagi katere je mogoče meriti napredek:

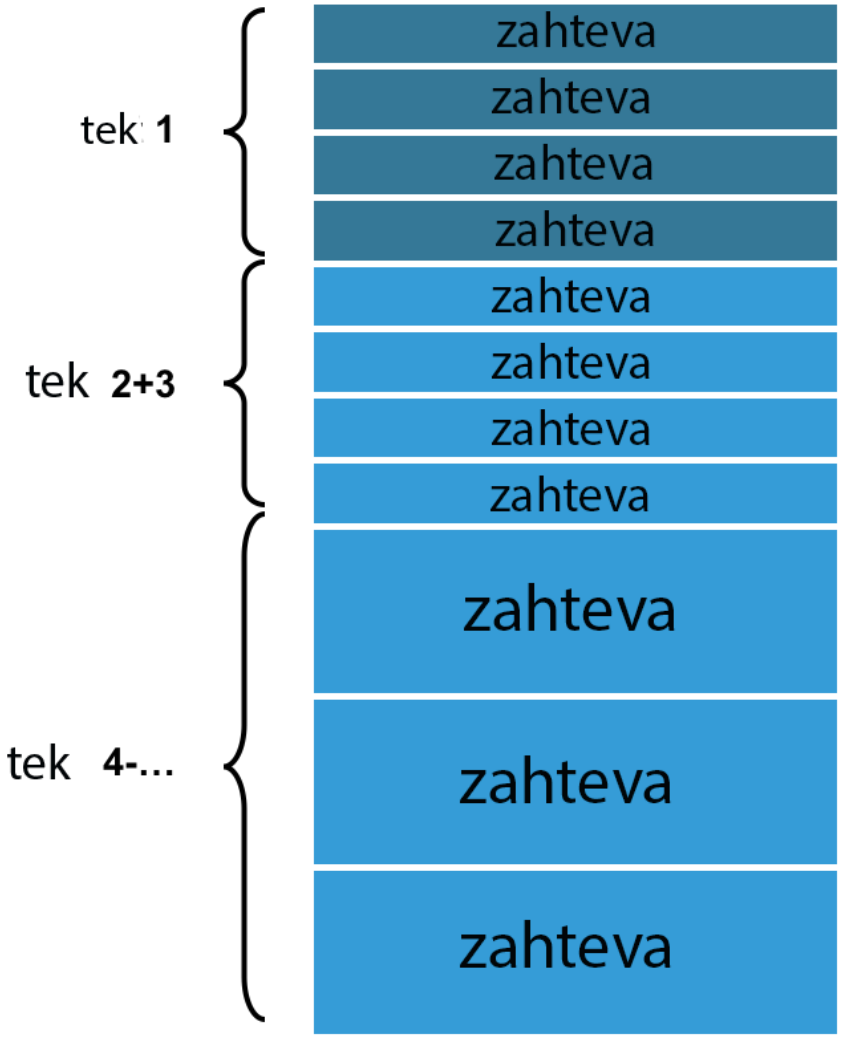
- za seznam zahtevkov je to cilj produkta,
- za seznam nalog je to cilj teka,
- za prirastek je to definicija končanega.

Te zaveze obstajajo za krepitev empirizma in scrum vrednot za ekipo in njene nosilce interesov (Scrum Guide, 2020).

1.6.1 Seznam zahtevkov

Seznam zahtevkov (angl. Product Backlog) je nastajajoč urejen seznam, ki prikazuje, kaj je potrebno za izboljšanje izdelka. Predstavlja edini vir dela, ki ga opravlja scrum ekipa. Primer seznama zahtevkov prikazuje slika 3.

Slika 3: Seznam zahtevkov



Prirejeno po Scrum.org (2016).

Zahtevki v seznamu zahtevkov, ki jih lahko scrum ekipa opravi v enem teku, se štejejo kot pripravljene za izbiro na sestanku za načrtovanje teka. To stopnjo preglednosti običajno pridobijo po opravljenem izpopolnjevanju zahtevkov. Izpopolnjevanje zahtevkov je dejanje razčlenitve in nadaljnje opredelitve postavk na manjše, natančnejše zahtevke. To je stalna dejavnost za dodajanje podrobnosti, kot so opis, vrstni red in velikost. Atributi zahtevkov se pogosto razlikujejo glede na področje dela.

Za ocenjevanje zahtevkov so odgovorni razvijalci, ki bodo opravljali delo. Skrbnik izdelka lahko vpliva na programerje tako, da jim pomaga razumeti in izbrati prave kompromise (Scrum Guide, 2020).

Zahteve za izdelek, ki ga razvija ekipa, so navedene v seznamu zahtevkov. Skrbnik izdelka je odgovoren za seznam zahtevkov, njihovo vsebino, razpoložljivost in prednostno

razvrstitev. Seznam zahtevkov ni nikoli dokončan. Poleg tega glede na načrt projekta odraža le prvotno znane in najbolje razumljene zahteve (Schwaber & Southerland, 2010).

Dodaten primer seznama zahtevkov je predstavljen v poglavju »Sestanek za načrtovanje teka«.

1.6.2 Cilj produkta

Cilj produkta opisuje prihodnje stanje produkta, ki ga lahko scrum ekipa načrtuje. Nahaja se v seznamu zahtevkov. Preostanek seznama zahtevkov se sproti prilagaja z namenom opredelitve, kaj bo izpolnilo zadani cilj produkta.

Produkt je sredstvo za doseganje vrednosti. Ima jasne meje, znane nosilce interesov in dobro opredeljene uporabnike. Produkt je lahko storitev, fizični izdelek ali kaj bolj abstraktnega. Cilj produkta je dolgoročni cilj ekipe scrum. Preden se lotijo naslednjega, morajo izpolniti (ali opustiti) trenutni cilj (Pichler, 2021).

1.6.3 Zahtevek

Zahtevki (angl. Product Backlog Item) so elementi, ki sestavljajo seznam zahtevkov. Omenjeni zahtevki produktov segajo od specifikacij, epik, uporabniških zgodb, hroščev, opravil do časovno omejenih raziskovalnih nalog. Vsak zahtevek mora imeti naslednje lastnosti (James & Walter, 2010):

- opis: kaj je cilj zahtevka,
- vrednost: poslovna vrednost zahtevka, ki jo določi lastnik izdelka,
- ocena: ekipa mora oceniti relativni trud, ki bo potreben za dokončanje zahtevka,
- prioriteta: skrbnik produkta mora določiti vrstni red zahtevkov glede na njihovo poslovno vrednost.

1.6.4 Uporabniška zgodba

Uporabniška zgodba (User story) je ena izmed oblik elementov v seznamu zahtevkov. Njihov namen je čim bolj jasno opisati željeno funkcionalnost zahtevka. Uporabniška zgodba ima določeno obliko:

"Kot _____ želim _____, da bom lahko _____."

Mnoge ekipe si prizadevajo, da bi bile uporabniške zgodbe dovolj majhne in razumljive za uspešno načrtovanje in dokončanje tekov. Opredelitev uporabniških zgodb tako, da so

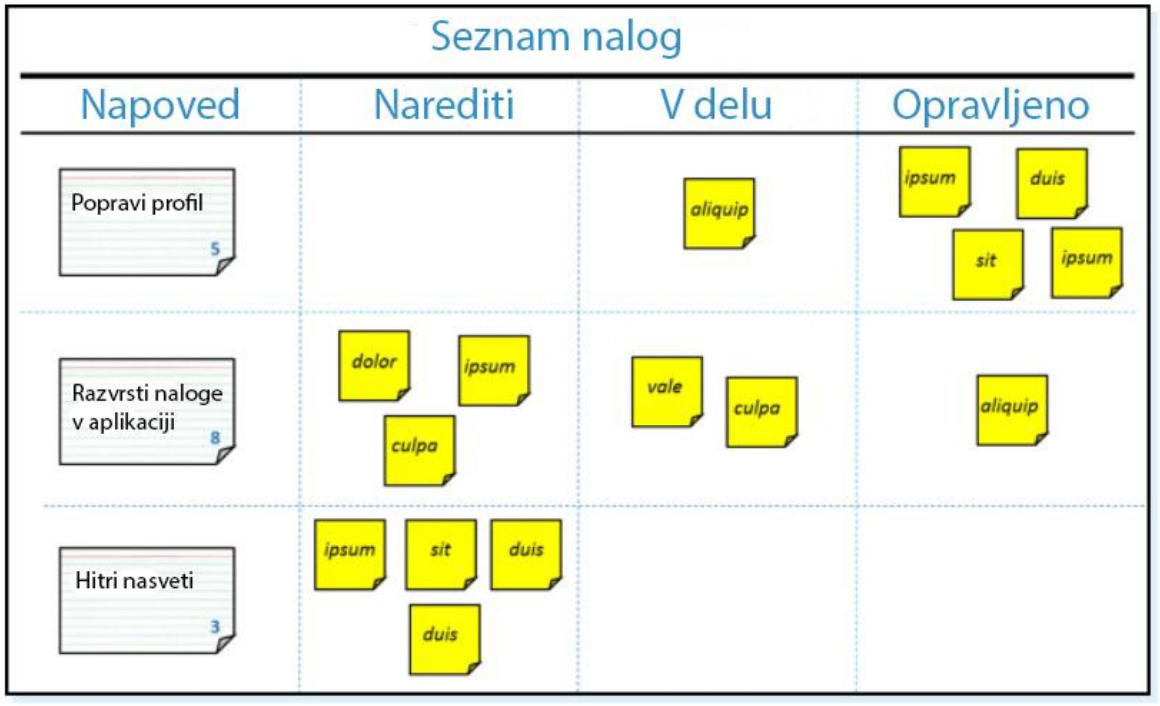
razumljive in uporabne, je skupno delo, ki vključuje skrbnika izdelka, skrbnika metode in ekipe (Scrum.inc, 2016).

1.6.5 Seznam nalog

Seznam nalog (angl. Sprint Backlog) je sestavljen iz cilja teka, manjšega niza seznama zahtevkov, izbranih za tek, in načrta za implementacijo prirastka.

Seznam nalog je načrt programerjev za njihovo delo. Prikaz seznama nalog je podan na sliki 4. To je v realnem času vidna slika o delu, ki ga načrtujejo razvijalci med tekom, da bi dosegli njegov cilj. Posledično se seznam nalog teka posodablja med celotnim tekom. Imeti mora dovolj podrobnosti, da lahko ekipa na vsakodnevnem preglednem sestanku preveri svoj napredek (ProductPlan, 2021).

Slika 4: Seznam nalog



Prirejeno po Scrum.org (2016).

1.6.6 Cilj teka

Cilj teka (angl. Sprint Goal) predstavlja glavni namen teka. Čeprav je cilj teka zaveza razvijalcev, zagotavlja prilagodljivost v smislu načina dela, ki je potrebno za njegovo dosego. Cilj teka v ekipi ustvarja občutek skladnosti in osredotočenosti pri delu ter jo spodbuja k skupnemu delu.

Cilj teka se določi na sestanku za načrtovanje teka. Izbrani cilj se doda tudi v trenutni seznam nalog. Med celotnim tekom imajo izbrani cilj razvijalci vedno v mislih. Če se izkaže, da je trud za načrtovano delo drugačen, kot so pričakovali, ekipa sodeluje s skrbnikom izdelka pri pogajanjih o spremenjenem obsegu seznama nalog v okviru teka. Sprememba obsega pri tem ne sme vplivati na dokončanje cilja teka (Objective Based, 2021).

1.6.7 Prirastek

Prirastek (angl. Increment) predstavlja temeljni korak k dosegu cilja produkta. Vsak prirastek je dodatek k prejšnjemu. Je temeljito preverjen, kar zagotavlja, da vsi prirastki delujejo skupaj in s tem ustvarjajo vrednost.

V teku je mogoče ustvariti več prirastkov. Njihova vsota se predstavi na sestanku za recenzijo teka, po potrebi pa se prirastek lahko nosilcem interesov dostavi še pred koncem teka. Opravljenega dela ni mogoče šteti za del prirastka, če ne ustreza definiciji končanega (Roopesh, 2019).

1.6.8 Definicija končanega

Opredelitev končanega je uradni opis stanja prirastka, če izpolnjuje merila kakovosti, ki je potrebna za produkt.

V trenutku, ko nek zahtevek ustreza definiciji končanega, se ustvari prirastek.

Opredelitev končanega ustvarja preglednost, tako da vsem omogoči skupno razumevanje tega, kar je bilo opravljeno v okviru prirastka. Če zahtevek ne ustreza definiciji končanega, ga ni mogoče izdati ali celo predstaviti na sestanku za recenzijo teka. Namesto tega se ga vrne v seznam nalog za prihodnjo obravnavo.

Če je opredelitev končanega za prirastek del standardov organizacije, jo morajo upoštevati vse njene scrum ekipe. Če organizacija nima določene splošne definicije končanega, mora vsaka ekipa ustvariti svojo definicijo, primerno svojemu izdelku.

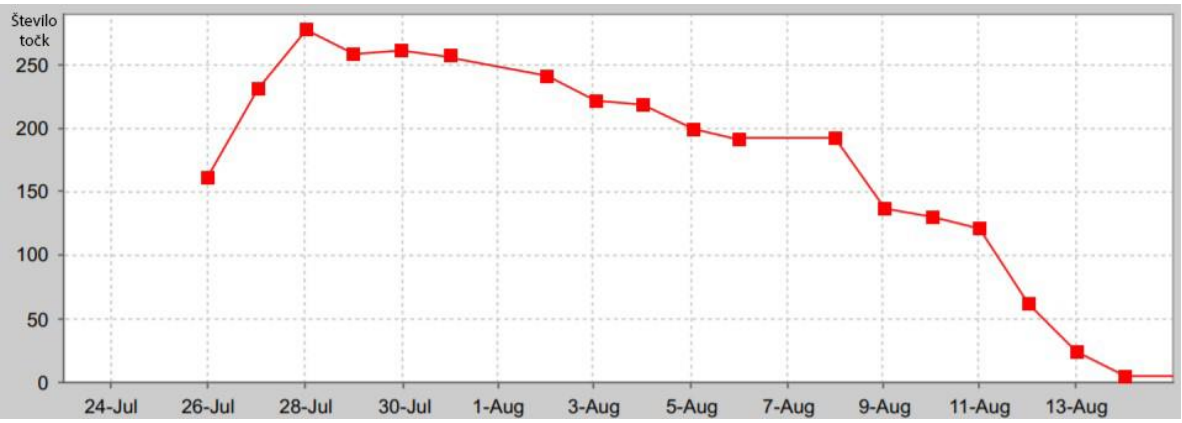
Razvijalci se morajo držati definicije končanega. Če na produktu sodeluje več scrum ekip, morajo medsebojno opredeliti in upoštevati svojo skupno definicijo končanega (Bowes, 2015).

1.6.9 Graf napredovanja

Graf napredovanja (angl. Burndown Chart) prikazuje količino preostalega dela skozi čas. Je

odličen način za vidno predstavitev soodvisnosti med količino dela, ki je preostalo v vsakem trenutku, in napredkom razvojne ekipe pri zmanjševanju tega dela. Primer grafa napredovanja kaže slika 5. Navpična os predstavlja število točk, ki jih ekipa namerava opraviti v prihodnjem teku, vodoravna pa število dni. Praviloma naj bi se krivulja v zadnjih dnevih teka strmo spustila do številke 0 (Schwaber & Southerland, 2010).

Slika 5: Graf napredovanja



Prirejeno po James & Walter (2010).

2 ORODJA ZA BOLJŠO IZVEDBO METODE SCRUM

Komunikacija med člani ekipe, v kateri delujemo, poteka v celoti na daljavo. Zaradi tega je bilo potrebno za uspešno sodelovanje prilagoditi način dela. Vsi sestanki potekajo preko aplikacije za video klice (Microsoft Teams). Poleg tega za različne sestanke potrebujemo aplikacije, ki nadomestijo potrebne procese v izvajanju metode scrum. Za prav ta namen obstaja že veliko različnih aplikacij. V tem poglavju bomo opisali tiste, ki jih uporabljamo v trenutni ekipi.

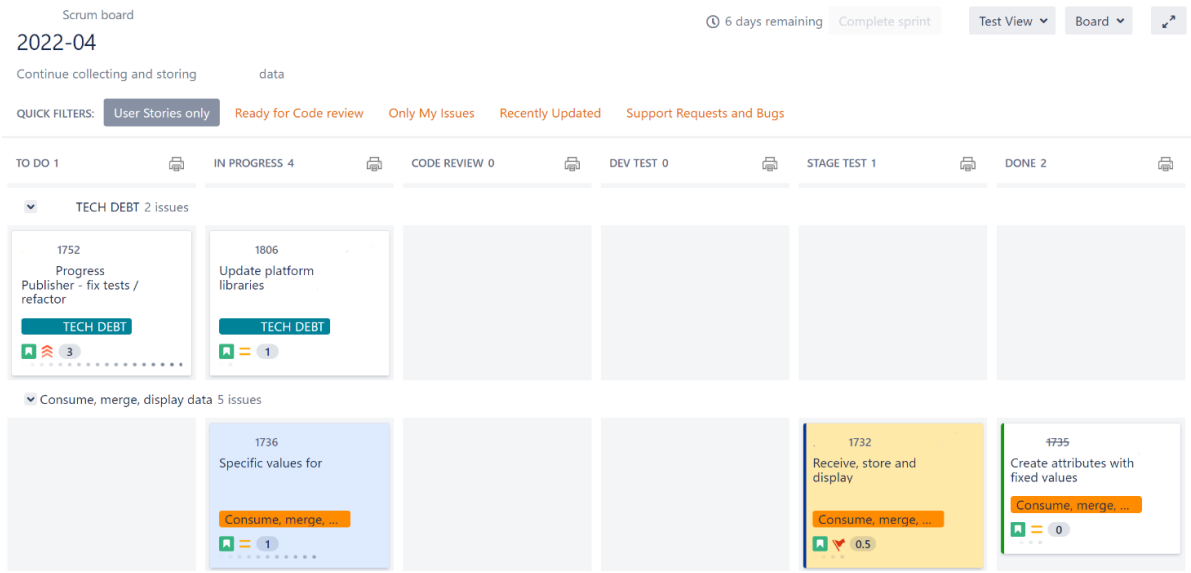
2.1 Jira

Eden izmed glavnih pripomočkov, ki pomagajo ekipi pri izvajanju metode scrum, je aplikacija Jira. Namenjena je nadzorovanju zahtevkov in upravljanju projektov. Večinoma jo uporabljajo podjetja pri razvoju programske opreme. S pomočjo njenega orodja scrum table lahko uporabniki hitro in učinkovito ustvarjajo nove zahtevke, jim dodajo željene opise in točkovne ocene.

Primer scrum table je viden na sliki 6. Slika je bila zajeta iz uporabljene aplikacije, zato tudi ni prevedena. Naloge so vidne tako članom ekipe kot tudi drugim nosilcem interesov. Poleg tega Jira ekipi omogoča učinkovito načrtovanje tekov. Časovne in druge parametre teka se

lahko prilagodi po želji ekipe. Zaradi tega scrum tabla pregledno prikazuje celoten napredek ekipe skozi svoje teke (Atlassian Software, 2021).

Slika 6: Primer scrum table v aplikaciji Jira



Prirejeno po Atlassian Software (2021).

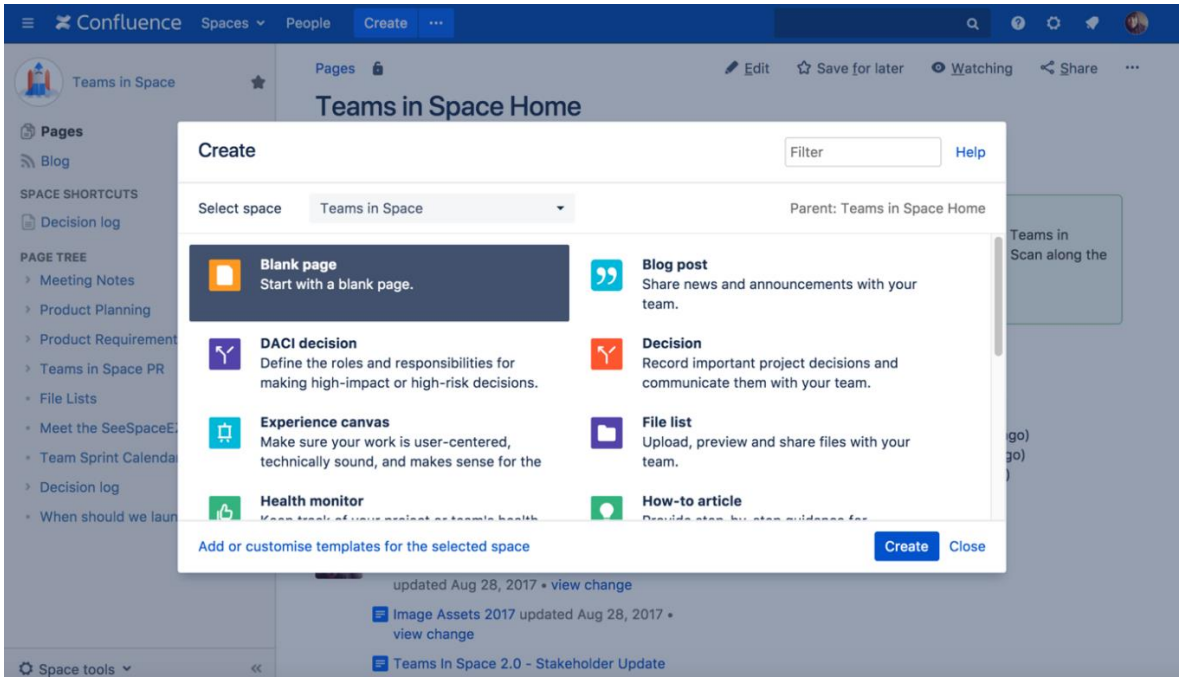
2.2 Confluence

Confluence je spletna aplikacija, namenjena beleženju in deljenju dokumentov. Namenjena je ekipam vseh vrst, ki potrebujejo učinkovit način dokumentiranja informacij.

Uporabnik lahko ustvari svoj delovni prostor, prilagojen njegovi ekipi. V njem lahko ustvari različne vrste dokumentov. Kot je vidno na sliki 7, lahko uporabnik izbere obliko dokumenta iz vnaprej pripravljenih primerov. Slika je bila zajeta iz uporabljene aplikacije, zato tudi ni prevedena. Obstajajo dokumenti za podporo marketinškim informacijam, kadrovskim oddelkom, produktom programske opreme in drugim.

S tem se učinkovito zmanjša čas priprave dokumentacije. Dokumenti v confluenceju v ekipah predstavljajo glavni vir uporabnih informacij. V aplikaciji je vgrajen tudi brskalnik, preko katerega lahko uporabnik poišče vse željene vsebine. Confluence se uporablja v vseh vrstah podjetij, saj je zaradi enostavne uporabe izredno učinkovit (Atlassian Software, 2021).

Slika 7: Primeri možnih oblik dokumentov v confluenceju

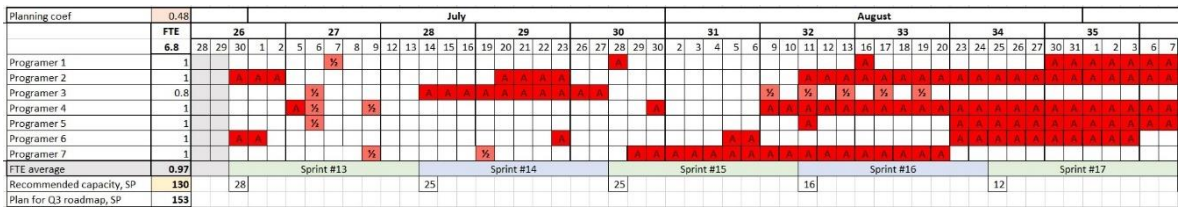


Vir: Rauer (2021).

2.3 Koledar odsotnosti in izračun kapacitete teka

Kot je bilo že omenjeno, ekipa v teku doseže določeno število opravljenih točk. V vsakem teku je število opravljenih točk načeloma različno. Eden izmed ključnih razlogov za to so odsotnosti članov ekipe. Zaradi tega je bil znotraj ekipe ustvarjen pripomoček za načrtovanje točk teka. Koledar odsotnosti je ustvaril skrbnik metode v programu Excel. Kot je vidno na sliki 8, koledar prikazuje termine bodočih tekov. V zgornjem delu so prikazani meseci, v spodnjem pa oštevilčeni teki. Na levi strani so napisani vsi člani ekipe (vidno kot Programer 1). V vrstici vsakega programerja so skozi vse prikazane teke vpisane njegove odsotnosti. Rdeča celica s črko A predstavlja, da bo član ekipe odsoten ves dan, 1/2 pa, da bo odsoten polovico dneva.

Slika 8: Koledar odsotnosti



Vir: lastno delo.

Pod vsakim začetkom teka je vidno določeno število točk. Te so izračunane po enačbi, ki jo je implementiral naš skrbnik metode. Izračun točk je odvisen od odsotnosti članov ekipe, začetno možno število točk pa temelji na opravljenem številu točk preteklih tekov. Tako se izračun z vsakim končanim tekom prilagodi. Točna enačba izračuna je znana le skrbniku metode. Z njeno pomočjo lahko ekipa natančneje načrtuje realno oceno opravljenih točk za naslednje teke.

2.4 Igra načrtovanja

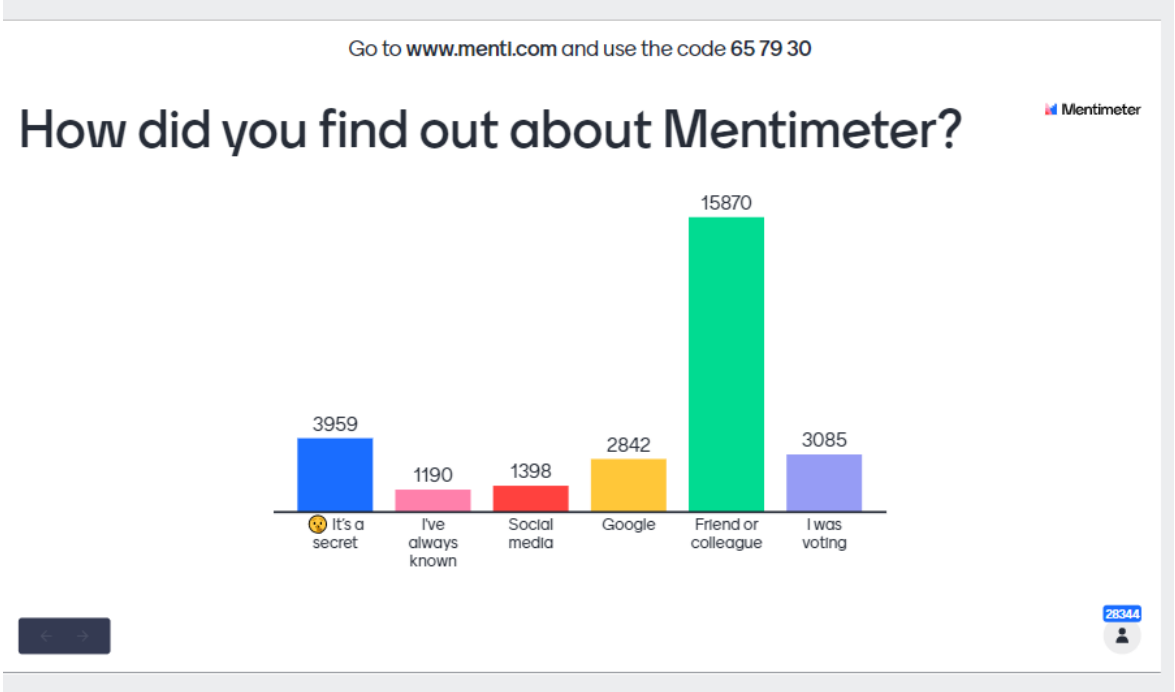
Igra načrtovanja (Planning poker) je način zbiranja ocen točk programerjev za določene zahtevke. Igra je preprosta. Vsaka oseba dobi karte s številkami iz opevanega Fibonaccijevega zaporedja – 1, 2, 3, 5, 8, 13 itd. Vsak zahtevek se ocenjuje ločeno. Vsak sodelujoči izbere karto s številko, ki jo želi dodeliti nalogi, in jo postavi na mizo tako, da je številka obrnjena navzdol. Vsi naenkrat obrnejo karte. Če se ocene med seboj razlikujejo le za eno karto (na primer petica, dve osmici in trinajstica), se upošteva povprečje (v tem primeru 6,6) in začne se ocenjevanje naslednje naloge. Če se ocene razlikujejo za tri karte ali več, morata osebi z najvišjo in najnižjo oceno razložiti razloge za svojo odločitev. Postopek se nato ponovi. Če se ocena ne spremeni, se upošteva povprečje vseh ocen (Southerland, 2016).

Igra načrtovanja je učinkovita spletna aplikacija, s katero scrum ekipe lahko ocenjujejo zahtevke. Njen način delovanja je enak, kot da bi igralci igrali s fizičnimi kartami. Z njeno pomočjo vsak član ekipe lahko oceni določen zahtevek, ne da bi pred tem lahko videl ocene svojih sodelavcev (We agile you S. L., 2021).

2.5 Mentimeter.com

Mentimeter je spletna aplikacija, s katero lahko uporabniki ustvarjajo vprašalnike in interaktivne vizualizacije. Z njeno pomočjo uporabniki lahko hitro in učinkovito ustvarijo svoj vprašalnik ter ga delijo komurkoli preko spletne povezave. Drugi uporabniki lahko nato takoj preko omenjene povezave vprašalnik rešijo. Med vsem časom pa lahko tisti, ki je vprašalnik ustvaril, vidi izpolnjene rezultate. Aplikacija mu glede na število izbranih odgovorov izriše tudi pregledne vidne predstavitve (Mentimeter, 2019). Primer vprašalnika in narisane vizualizacije prikazuje slika 9. Slika je bila zajeta iz uporabljene aplikacije, zato tudi ni prevedena.

Slika 9: Primer vizualizacije odgovorov na vprašalnik v aplikaciji mentimeter



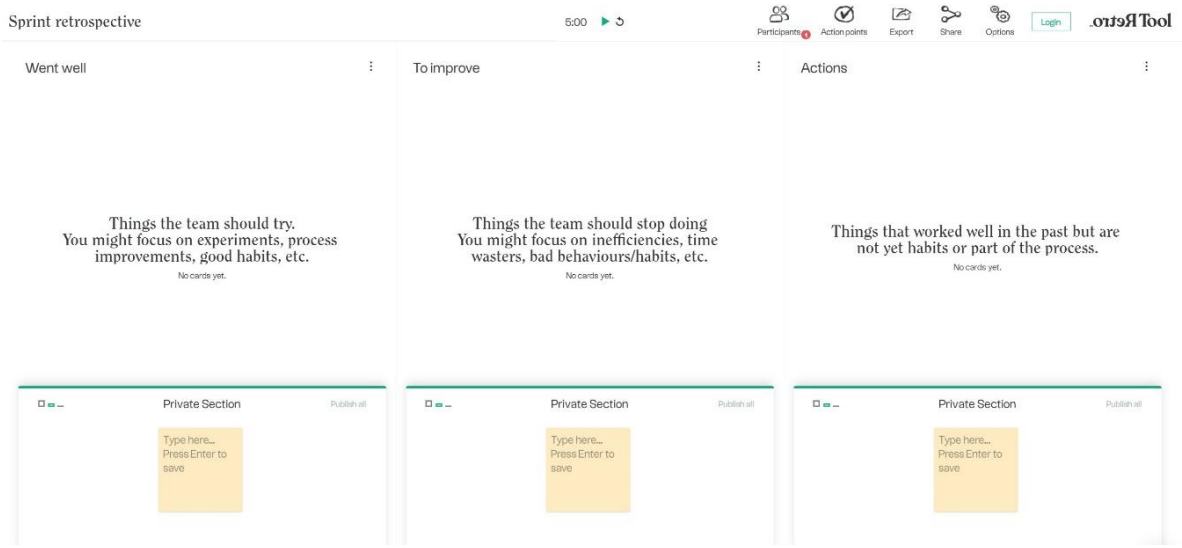
Vir: RUBeL (2021).

2.6 Retrotool.com

RetroTool je aplikacija, namenjena ekipam pri izvajanju sestankov za revizijo teka. Primerna je predvsem za ekipe, ki delujejo na daljavo. Implementirana je na način, da lahko uporabnik v zelo kratkem času s pomočjo že narejenih predlog pripravi vse za sestanek za revizijo teka. Kot prikazuje slika 10, je predloga za revizijo teka sestavljena iz treh stolpcev. Slika je bila zajeta iz uporabljene aplikacije, zato tudi ni prevedena.

Prvi stolpec z leve strani je namenjen stvarem v teku, ki so se dobro naredile. Drugi stolpec je namenjen izpostavitvi stvari, ki bi se lahko bolje izpeljale. Tretji stolpec pa je namenjen rešitvam, s katerimi se lahko izboljša stvari, izpostavljene v drugem stolpcu. Aplikacija ima dodan tudi odštevalnik časa. Z njim si uporabniki pomagajo pri določanju časa, namenjenega vpisovanju točk v posamezne stolpce (RetroTool, 2021). Primer uporabe aplikacije je opisan v poglavju »Sestanek za revizijo teka«.

Slika 10: Predloga za revizijo teka



Vir: RetroTool (2021).

3 ANALIZA RAZVOJA PROGRAMSKE OPREME V IZBRANEM PODJETJU

Kot je bilo že omenjeno, se praktične implementacije metode scrum med seboj pogosto razlikujejo. Vse je odvisno od načina dela in okolja, v katerem dela določena ekipa. Če ekipa opazi, da pri svojem delovanju upoštevanje smernic scruma zanjo ni logično ali ustrezno, se lahko odloči, da bo delovni proces prilagodila glede na svoje potrebe.

3.1 Analiza trenutne implementacije metode scrum

Analiza razvoja programske opreme po metodi scrum se nanaša na izbrano podjetje, v katerem sem zaposlen. Delamo v ekipi, sestavljeni iz sedmih programerjev, skrbnika izdelka in skrbnika metode. Dva programerja živita v Sloveniji, ostali del ekipe pa je nastanjen v drugi državi. Zaradi tega celotna komunikacija med ekipo poteka na daljavo. To pomeni, da so vsi sestanki organizirani preko video klicev, ostala komunikacija pa preko e-sporočevalne aplikacije Slack. Ekipa je po številčnosti članov velika. Pri samem razvoju produkta uporabljamo prilagojeno metodo scrum.

3.1.1 Opis teka v delujoči ekipi

Ekipa deluje na popolnoma novem produktu. Njena naloga je implementacija spletne aplikacije, ki pomaga uslužbencem podjetja pri upravljanju delovnih enot podjetja. Zaradi tega se zahteve po novih funkcionalnostih zelo pogosto spreminjajo in prilagajajo.

V obravnavani ekipi je en tek dolg dva tedna. Glede na spremenljive razmere, v katerih delujemo, nam to ustreza. Dober primer je, če na primer nosilci interesov zahtevajo, da se čim hitreje implementira neka nova funkcionalnost. V tem primeru lahko skrbnik izdelka v seznamu zahtev naredi ustrezen zahtevek – uporabniško zgodbo – ter ga uvrsti na vrh seznama zahtev. Posledično se ekipa v naslednjem teku, pred vsemi drugimi, najprej loti novo narejenega zahtevka. Željena funkcionalnost se zaradi tega lahko implementira v relativno kratkem času.

Med celotnim tekom programerji implementirajo različne funkcionalnosti znotraj aplikacije. Med njimi so lahko popravki obstoječe programske kode, izboljšave trenutnih procesov ali pa popolnoma novi dodatki. Na koncu teka pa se vse dokončane funkcionalnosti združi v eno celoto (prirastek), ki se objavi na okolje, kjer je na voljo končnim uporabnikom. Aplikacija, torej vsak tek, dobi novo nadgradnjo, v kateri so po prioritetah uporabnikov implementirane najpomembnejše funkcionalnosti.

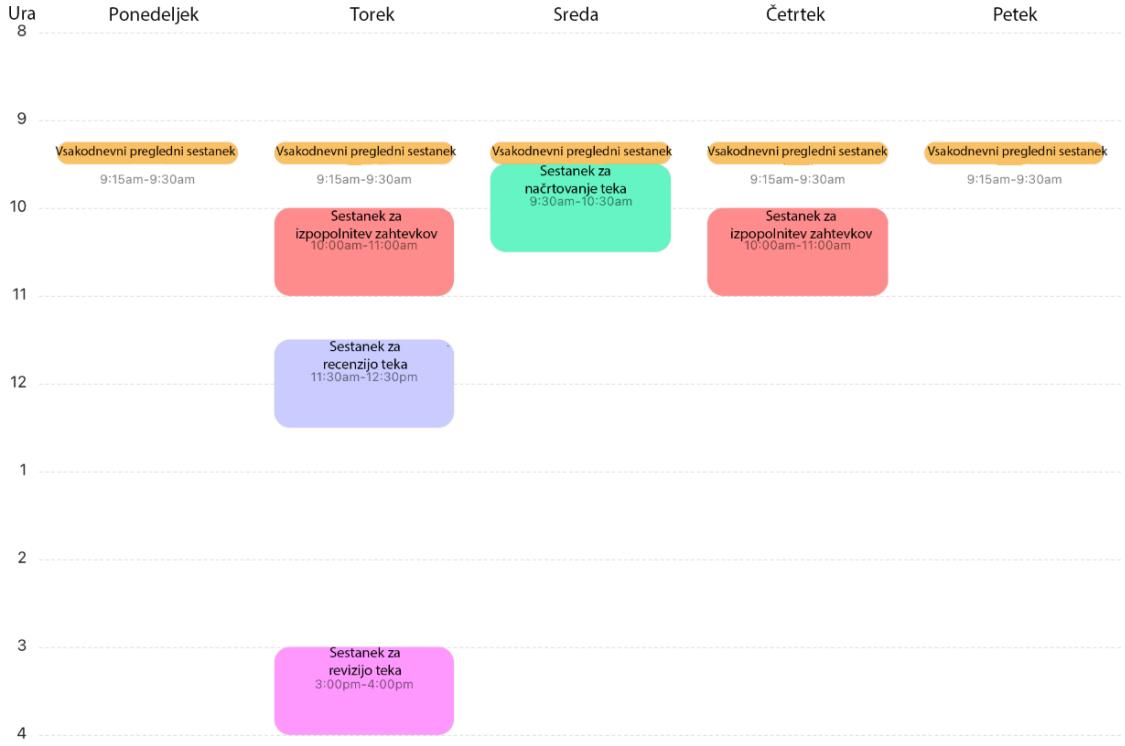
3.1.2 Struktura teka v delujoči ekipi

Kot je bilo že omenjeno, tek v trenutni ekipi traja dva tedna. Glede na produkt, ki ga razvijamo, je to ustrezno, saj se zahteve nosilcev interesov pogosto spreminjajo. Med celotnim tekom imamo načrtovanih tudi več sestankov. Vsak sestanek je strukturiran tako, da služi točno določenemu namenu. Kot je prikazano na sliki 11, imamo vsak dan ob 9:15 redni pregledni sestanek, vsak teden ob torkih in četrtkih ob 10:30 pa imamo tudi sestanek za izpopolnitev zahtevkov.

Na sliki 11 je prikazan začetek novega in konec starega teka. Tek se začne na vsako drugo sredo v mesecu. Začne se z vsakodnevnim preglednim sestankom ob 9:15. Takoj za njim, ob 9:30, pa imamo sestanek za načrtovanje teka. Tek se nato nadaljuje skozi ves teden. Prikazan je na sliki 12. Vsak dan imamo pregledni sestanek, vsake dva dni pa sestanek za izpopolnitev zahtevkov. Konec teka je sestavljen iz ponedeljka in torka – prikazana sta na sliki 11. Glavni dan predstavlja torek, ko imamo načrtovanih več sestankov. Običajno imamo dva. To sta vsakodnevni pregledni sestanek in sestanek za izpopolnitev zahtevkov. Poleg tega pa imamo na koncu teka dva dodatna sestanka. To sta sestanek za recenzijo teka in sestanek za revizijo teka. Oba sestanka trajata eno uro.

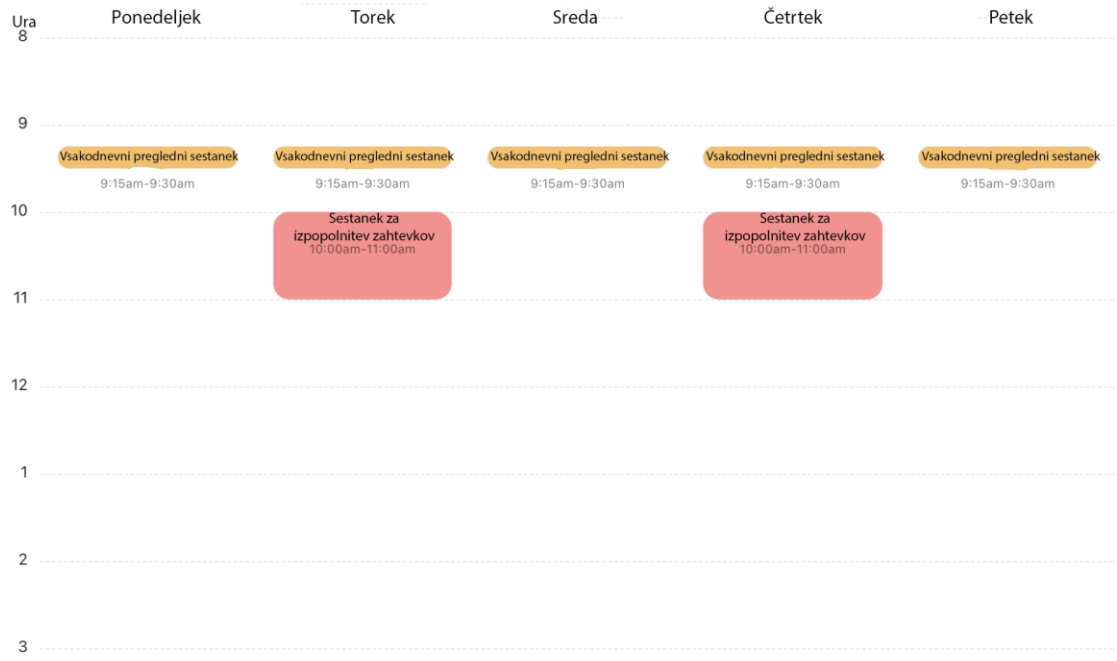
Vsak sestanek ima svoj namen. Temu ustrezno je tudi razporejen na določen dan in čas. Med delovanjem po trenutnem opisu teka pa je ekipa opazila, da bi lahko časovne termine sestankov znotraj teka bolje prilagodili. Uvedene spremembe strukture sestankov teka smo opisali v poglavju »Sprememba strukture sestankov teka«.

Slika 11: Teden začetka in konca teka



Vir: lastno delo.

Slika 12: Vmesni teden teka



Vir: lastno delo.

3.1.3 Vsakodnevni pregledni sestanek

Med celotnim tekom imamo vsakodnevni pregledni sestanek. Udeležujejo se ga skrbnik izdelka, skrbnik metode in vsi programerji. Sestanek po navadi traja 15 minut. Če kdo potrebuje kako pomoč, lahko traja tudi dlje časa. Sestanek se začne s tem, da vsak programer predstavi trenutno stanje naloge, ki jo opravlja. To se pravi, da pove, kaj je delal prejšnji dan teka, kaj bo delal današnji dan in ali ima mogoče kakšne težave. Če je naletel na težavo, lahko druge člane ekipe prosi za pomoč. Ko vsi programerji predstavijo svoje trenutno delo, enako naredi tudi skrbnik izdelka. Na koncu sestanka skrbnik metode vpraša, če kdo potrebuje kakršnokoli drugo pomoč. Če ne, se sestanek zaključi.

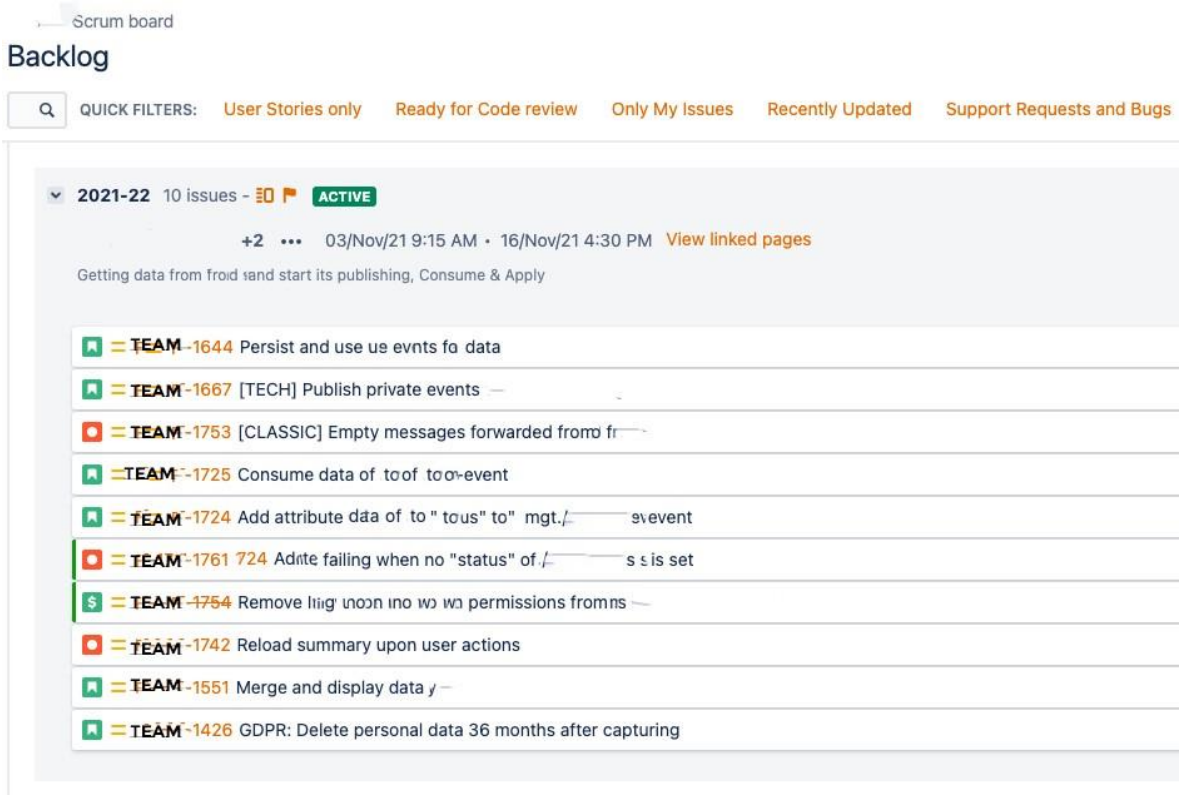
Občasno se zgodi, da morajo med svojim delom programerji komunicirati z uslužbenci zunaj svoje ekipe. Pri tem se tudi večkrat zgodi, da se omenjeni uslužbenci ne odzovejo ali pa se ne morejo uskladiti pri določenih stvareh. V takih primerih skrbnik metode priskoči na pomoč in pomaga razrešiti morebitne težave pri komunikaciji med sodelavci.

3.1.4 Sestanek za načrtovanje teka

Začetek vsakega teka se začne s sestankom za njegovo načrtovanje. Sestanek traja eno uro. Udeleži se ga celotna ekipa. Sestanek vodi skrbnik izdelka ob pomoči skrbnika metode. Najprej se pregleda odsotnosti članov ekipe skozi trenutno načrtovani tek. To se vidi v koledarju odsotnosti, v katerega morajo člani ekipe pred samim sestankom vnesti svoje načrtovane odsotnosti. Glede na vnešene odsotnosti se izračunajo tudi možne točke, ki bodo določale kapaciteto teka (opisano v poglavju »Orodja za boljšo izvedbo metode scrum«). Ko je znano končno število točk teka, skrbnik izdelka ekipi prikaže seznam zahtevkov. Kot je bilo omenjeno, celotna komunikacija poteka preko video klicev. To pomeni, da na sestanku skrbnik izdelka »deli« svoj zaslon in prikazuje programsko orodje Jira. Sestanek se nadaljuje z izbiro zgodb, ki bodo sestavljale tek. Skrbnik izdelka začne pri zgodbah, ki so na vrhu seznama nalog, saj imajo najvišjo prioriteto. Za vsako zgodbo vpraša ekipo, če se strinja, da se jo umesti v naslednji tek. Vse zgodbe, ki so na vrhu seznama nalog, morajo biti pred samim sestankom za načrtovanje teka napisane razumljivo, poleg tega pa morajo vsebovati vse potrebne informacije za njihovo uspešno implementacijo. Te zgodbe ekipa predstavi na sestanku za izpopolnitev zahtevkov. Če je zgodba razumljiva celotni ekipi, se jo umesti v tek. Če ni, tisti, ki jim zgodba ni razumljiva, skrbniku izdelka ali razvijalcem programske opreme postavijo ustrezna vprašanja. Nanja odgovorijo člani ekipe, ki poznajo podrobnosti zgodbe. Zgodbi, ki je nerazumljiva, se sme naknadno vpisati dodatne informacije. Vsaka zgodba je ocenjena z določenim številom točk, vsak tek pa je določen s kapaciteto možnih točk. Zaradi tega se proces umeščanja zgodb v tek toliko ponovi, da se seštevek točk vseh zgodb ujema s kapaciteto teka. Ko je ta dosežena, mora ekipa določiti tudi cilj teka.

Vsak tek ima določen cilj. Ta je sestavljen iz enega ali več kratkih stavkov. Namen cilja teka je, da na kratek način predstavi, kaj bo ekipa poskušala doseči v teku (viden nad zahtevki na sliki 13 - slika je bila zajeta iz uporabljene aplikacije, zato tudi ni prevedena). Ekipa sestavi cilj teka tako, da v njem zajame najpomembnejše funkcionalnosti, ki jih bo med tekom implementirala. Pomembno je, da je napisan tako, da nosilci interesov vedo, kakšne dodatke ali popravke bo aplikacija dobila. Zaradi tega se pri sestavljanju cilja teka izogibamo specifičnim tehničnim izrazom, ki so razumljivi le tistim, ki so vešči tehničnih znanj. Ko je cilj teka opredeljen, skrbnik metode vpraša, če ima pred začetkom novega teka kdorkoli kakšne zadržke ali vprašanja ter če je ekipa prepričana, da lahko vse umeščene naloge izpolni med tekom. Če zadržkov ni, skrbnik metode začne tek.

Slika 13: Seznam nalog in cilj teka med sestankom za načrtovanje teka



Prirejeno po Atlassian Software (2021).

3.1.5 Sestanki za izpopolnitev zahtevkov

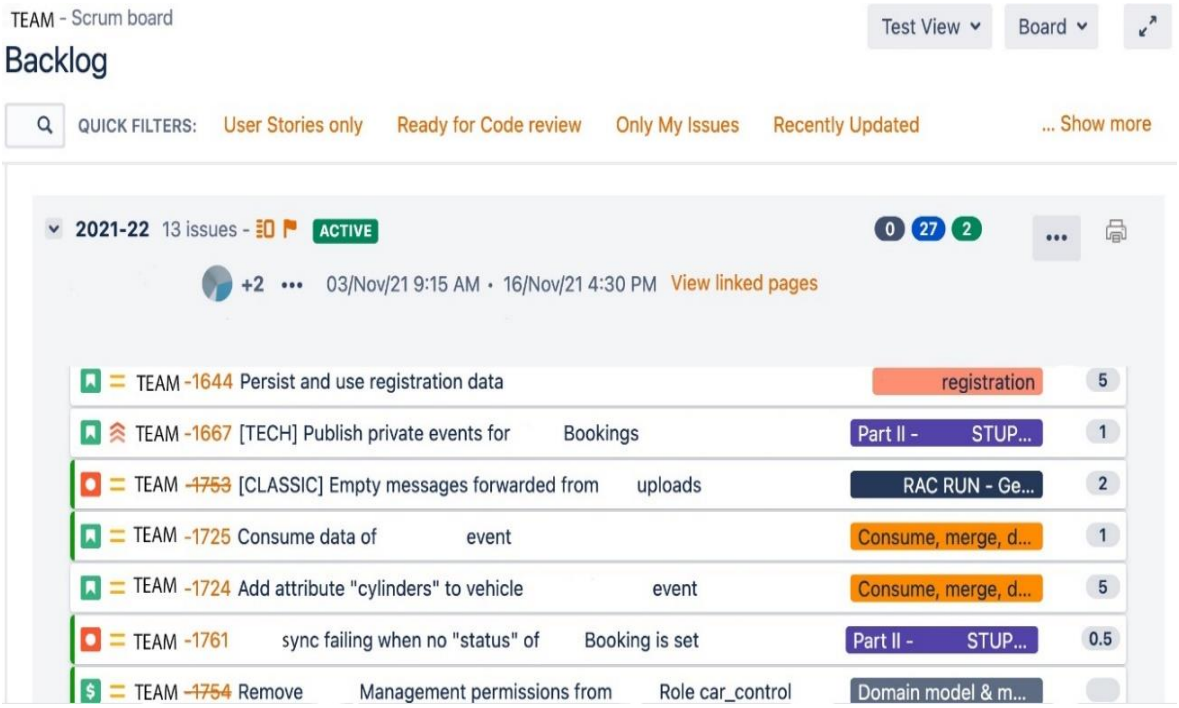
Okolje, v katerem deluje naša ekipa, se nenehno spreminja. Zahteve nosilcev interesov se spreminjajo glede na razmere na trgu in način dela zaposlenih v podjetju. Posledica tega je, da mora imeti skrbnik izdelka dober vpogled, kaj je potrebno narediti v določenem času. To pomeni, da mora dobro načrtovati, katera funkcionalnost mora biti narejena v določenem času ter do kdaj mora biti dokončana. Takšno načrtovanje je težko. Poleg tega pa mora

skrbnik imeti dovolj znanja, da lahko oceni trud in čas, potreben za implementacijo večjih funkcionalnosti.

Za pomoč pri načrtovanju dela v prihodnjih tekih se ekipa dvakrat na teden (v torek in četrtek) zbere na sestanku za izpopolnitev zahtevkov. Sestanek traja eno uro. Prisotni so vsi člani ekipe. Sestanek vodi skrbnik izdelka ob pomoči skrbnika metode. Skrbnik izdelka si pred sestankom pripravi uporabniške zgodbe, ki bi jih rad izpopolnil s pomočjo programerjev. Izpopolnitev uporabniške zgodbe pomeni, da ima zgodba vpisane vse potrebne informacije, da jo lahko ekipa umesti v tek. To tudi pomeni, da lahko omenjeno uporabniško zgodbo prevzame in izpelje katerikoli programer v ekipi.

Skrbnik izdelka pred začetkom sestanka za izpopolnitev zahtevkov na vrh seznama zahtevkov umesti uporabniške zgodbe, katere želi, da jih ekipa izpopolni. Primer seznama zahtevkov je prikazan na sliki 14. Slika je bila zajeta iz uporabljene aplikacije, zato tudi ni prevedena. Na sestanku skrbnik v programski opremi Jira na svojem zaslonu prikaže vsako uporabniško zgodbo posebej. Ekipa skupaj prebere vsebino zgodbe. Če je zgodba razumljiva celotni ekipi, je ni potrebno dodatno izpopolnjevati. Če ni, mora član ekipe, kateremu kaj ni popolnoma razumljivo, to povedati in postaviti ustrezna vprašanja za razrešitev nejasnosti. Na zastavljena vprašanja odgovorijo programerji ali skrbnik izdelka. Pridobljene dodatne informacije skrbnik izdelka zapiše v besedilo uporabniške zgodbe. Na enak način se razreši ostala vprašanja. Po rešenih vseh vprašanjih se lahko preide na naslednji korak – ocenjevanje težavnosti zgodbe.

Slika 14: Primer seznama zahtevkov



Prirejeno po Atlassian Software (2021).

V trenutnem procesu ocenjevanja zgodb ekipa ne uporablja nobenega temu namenjenega orodja. Kot je bilo že omenjeno, vsi sestanki potekajo preko aplikacije za video klice. Znotraj aplikacije obstaja tudi možnost pisanja sporočil, vidnih vsem članom video klica.

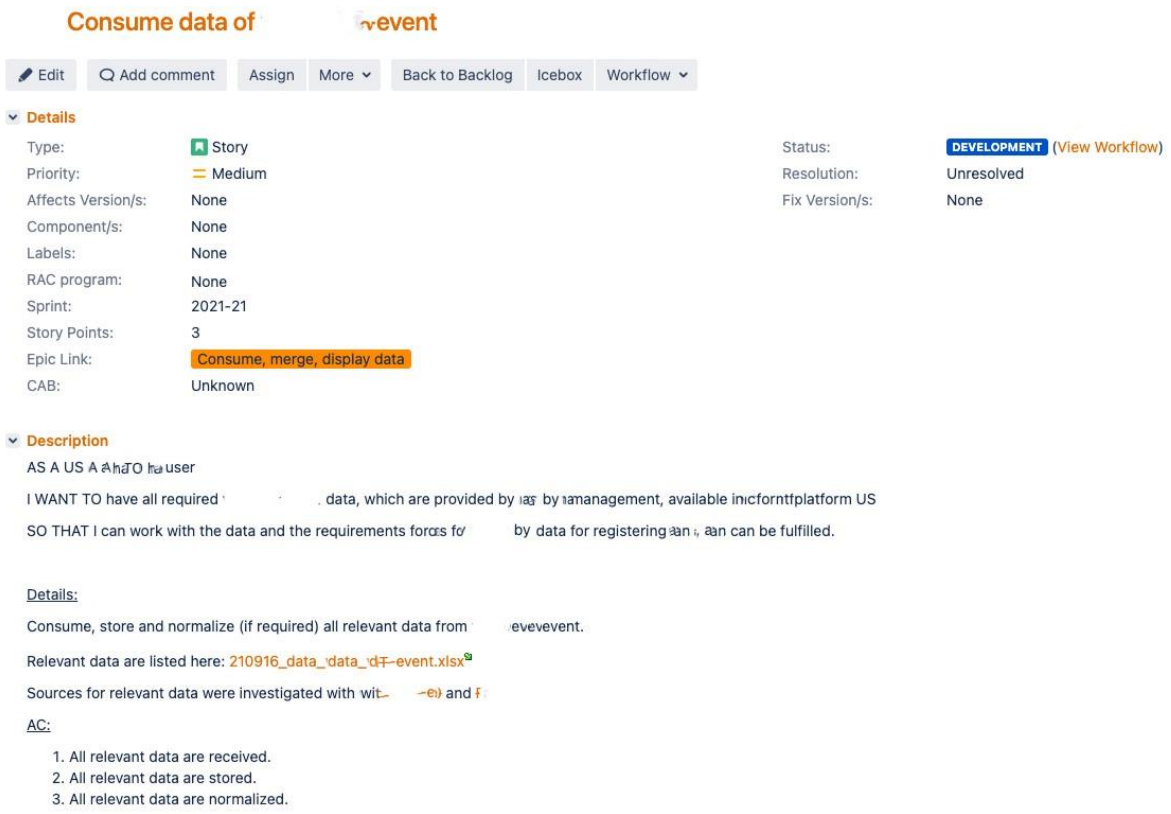
Trenutni proces ocenjevanja zgodb poteka tako, da skrbnik metode v skupni klepetalnik vpiše ime zgodbe, ki jo bo ekipa ocenila. Nato vpraša ostale programerje, če so pripravljeni na ocenitev težavnosti zgodbe. Skrbnik izdelka v procesu ocenjevanja zgodb ne sodeluje, saj ni zadolžen za njeno izpeljavo. Če nobeden od programerjev nima dodatnih vprašanj ali zadržkov, se zgodbo lahko oceni. To poteka tako, da skrbnik metode poda znak, po katerem morajo vsi programerji naenkrat vpisati svoje ocenjeno število točk za zgodbo. Pomembno je, da to res naredijo vsi naenkrat. To pa zaradi tega, da vpisane točke predstavljajo realno oceno vsakega programerja. Če tega pravila ne bi bilo, bi lahko programerji, ki bi vpisali število točk kasneje kot drugi, svojo oceno prilagodili ocenam drugih programerjev.

Jasno je, da trenutni način ocenjevanja ni optimalen, saj lahko vsak programer za trenutek počaka in vidi, kakšno oceno je vpisal njegov sodelavec, ki bolje pozna problem, ter za njim vpiše enako število točk. Več o posledicah takšnega delovanja je opisano v poglavju »Glavne pomanjkljivosti trenutne metode«. Ko so ocene vseh programerjev vpisane v skupni klepetalnik, jih skrbnik metode pregleda. Če so vse ocene točk enake, jih vpiše v zgodbo v programski opremi Jira (vidno na sliki 15 kot »story points« - slika je bila zajeta iz uporabljene aplikacije, zato tudi ni prevedena).

Če se točke med seboj razlikujejo, skrbnik metode pregleda, kdo je vpisal najvišje in kdo najnižje število točk. Nato programerje vpraša, zakaj menijo, da je zgodba vredna toliko več ali toliko manj točk. Vsak programer utemelji, zakaj meni, da je zgodba vredna zapisanega števila točk. Možno je, da določeni argumenti prepričajo ostale člane ekipe, da spremenijo svojo prvotno oceno.

Na koncu razprave skrbnik metode prosi člane ekipe, da ponovijo ocenjevanje zgodbe. Če so v naslednjem krogu vse ocene enake, se ocenjevanje trenutne zgodbe konča, če pa ne, se ponovi razprava o razlogih za različne ocene zgodbe. Če se tudi po drugi ali tretji ponovitvi ocenjevanja zgodbe ekipa še zmeraj ne more uskladiti glede končne ocene, je zgodbo treba prestaviti na naslednji sestanek za izpopolnitev zahtevkov. Večinoma je to potrebno, kadar je za razrešitev nejasnosti potreben dodaten čas ali pa posvet z zunanjimi sodelavci ali nosilci interesov, ki so povezani z zgodbo. Opisani proces predstavlja trenutno stanje ocenjevanja zgodb.

Slika 15: Primer zahtevka



Prirejeno po Atlassian Software (2021).

3.1.6 Sestanek za recenzijo teka

Zadnji dan vsakega teka pride na vrsto sestanek za recenzijo. Izveden je preko video klica, traja pa eno uro. Poleg celotne ekipe se ga udeležijo tudi vsi nosilci interesov. Vodi ga skrbnik metode. Namen sestanka je predstaviti, kaj je ekipa dosegla v tekočem teku. Na začetku skrbnik metode predstavi dnevni red. V prvem delu sestanka skrbnik izdelka predstavi, katere naloge je ekipa uspešno opravila v teku in ali je dosegla njegov cilj, v drugem delu pa demonstrira dejansko novo implementirano funkcionalnost v aplikaciji. Na tak način nosilci interesov dobijo dober vpogled v dejansko napredovanje aplikacije.

Kot je bilo navedeno, v prvem delu sestanka skrbnik izdelka predstavi, katere naloge znotraj teka je ekipa uspešno opravila. Poleg tega prikaže načrte ekipe za bližnjo prihodnost. Te vsebujejo časovno razporeditev večjih funkcionalnosti za aplikacijo. Tako lahko nosilci interesov dobijo dober vpogled, kaj bo implementirano v naslednjih mesecih.

Po predstavitvi opravljenih nalog sledi demonstracija implementiranih funkcionalnosti. To opravi eden izmed programerjev. Predstavitev poteka tako, da programer deli svoj zaslon in na njem prikaže aplikacijo. Nato ob razlagi prikazuje, kaj vse novo implementirana

funkcionalnost omogoča uporabniku. Dober primer tega je implementacija novega gumba v aplikaciji. Programer v svoji predstavitvi prikaže novonastali gumb. Ob tem razloži, čemu je namenjen in kaj vse je potrebno, da se gumb prikaže v aplikaciji. Nato na ta gumb pritisne in razloži, kaj se med tem dogaja tako v zunanjosti kot v notranjosti aplikacije.

Za predstavitvijo opravljenih nalog sledi zadnji del sestanka – vprašanja nosilcev interesov. Te skrbnik metode povabi, da kaj vprašajo ali komentirajo predstavljeni tek. Ko kdo vpraša ekipo o določeni funkcionalnosti ali bodočih načrtih za aplikacijo, mu odgovori ali skrbnik izdelka ali določeni programer. Načeloma na tehnična vprašanja vskoči programer, medtem ko na vprašanja v zvezi z načrti in drugimi temami odgovori skrbnik izdelka. Vsako vprašanje ali komentar sta izredno dobrodošla, saj ekipa dobi pomembne povratne informacije glede svojega dela. Če vprašanj ali komentarjev ni, ekipa težko ve, ali so nosilci interesov zadovoljni z njihovim delom.

3.1.7 Sestanek za revizijo teka

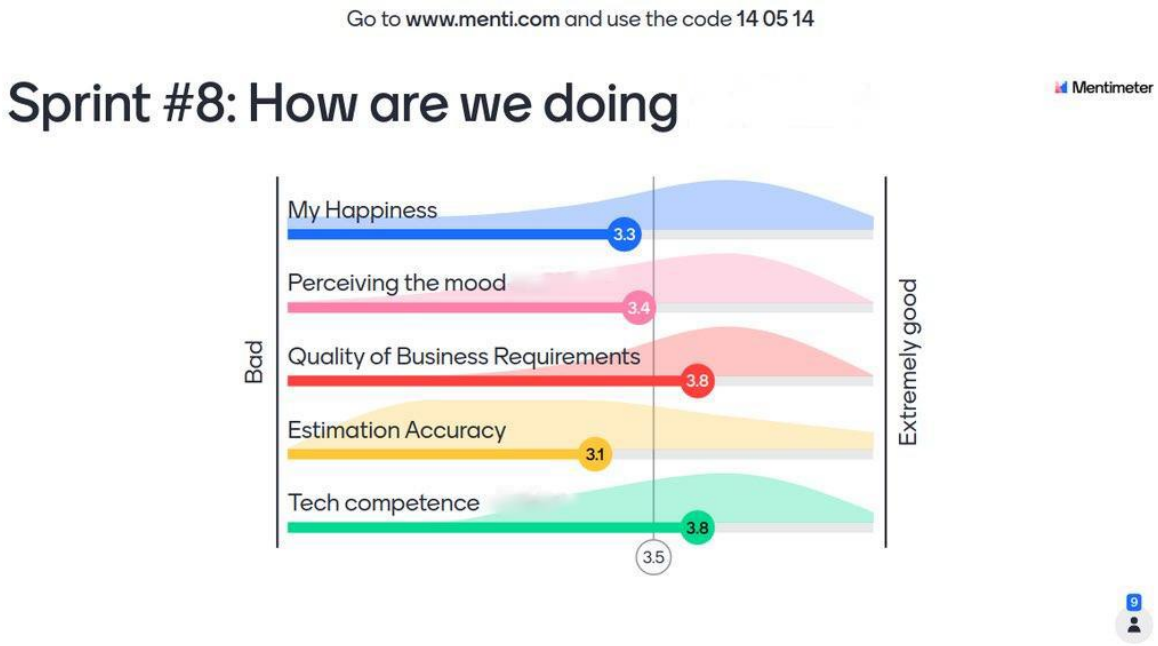
Vsak konec teka se zaključi s sestankom za revizijo teka. Sestanek traja eno uro, prisotna pa je celotna ekipa. Vodi ga skrbnik metode. Sestanek se začne s tem, da skrbnik metode povabi vse člane ekipe, naj obiščejo spletno stran mentimeter.com (opisano v poglavju »Orodja za boljšo izvedbo metode scrum«), na kateri bodo podali svoje ocene glede končanega teka.

Na obiskani strani mora vsak član ekipe podati oceno svojega prepričanja za pet različnih mnenj. Ta so enaka za vsak tek, saj s tem lahko spremljamo, kako se ocene gibajo skozi čas. Kot je razvidno s slike 16, so mnenja naslednja (slika je bila zajeta iz uporabljene aplikacije, zato tudi ni prevedena):

- moje zadovoljstvo,
- ocenitev zadovoljstva celotne ekipe,
- kvaliteta poslovnih zahtev,
- točnost ocenitev nalog,
- tehnične kompetence ekipe.

Za vsako mnenje lahko sodelujoči člani ekipe podajo oceno od 1 do 5. V našem primeru 1 pomeni najslabša ocena, medtem ko je ocena 5 najboljša. Vsa mnenja skupaj odražajo neko oceno teka. Ko vsi člani ekipe glasujejo, skrbnik metode komentira rezultate. Če je neko mnenje dobilo izrazito slabe ocene, povpraša ekipo, zakaj so ga tako ocenili. V tem primeru član ekipe, ki je podal slabo oceno, poda svoj komentar. Ostali člani ekipe lahko na to dodajo svoja opažanja. Če so kateri problemi manjši, jih lahko ekipa predebatira že v tem koraku. V nasprotnem primeru lahko član ekipe omenjeni problem izpostavi v drugem delu sestanka. Na koncu prvega dela sestanka skrbnik metode komentira dobljene ocene glede na pretekle teke. Omenjeni del sestanka je namenjen splošnemu vpogledu v kvaliteto končanega teka.

Slika 16: Vprašalnik o mnenju o preteklem teku

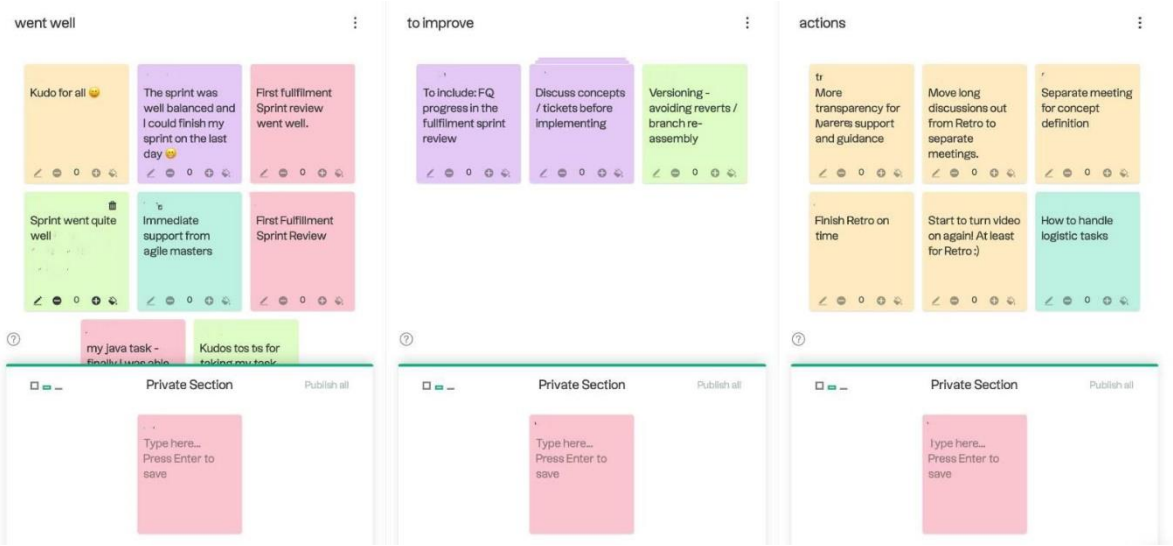


Vir: menti.com (2019).

V drugem delu teka se celotna ekipa vpiše na spletno stran retrotool.com. Orodje je podrobneje opisano v poglavju »Orodja za boljšo izvedbo metode scrum«. Skrbnik metode v spletni aplikaciji nastavi odštevalnik časa na pet minut.

Kot kaže slika 17, je spletna aplikacija razdeljena na tri dele: »kaj je bilo dobro« (vidno kot »went well«), »za izboljšati« (»to improve«) in »dejanja« (»actions«). Slika je bila zajeta iz uporabljene aplikacije, zato tudi ni prevedena. Naloga drugih članov ekipe, torej skrbnika izdelka in programerjev, je, da v določenem času v stolpca »went well« in »to improve« vpišejo svoja opažanja. V stolpec se lahko doda nov element z opažanji tako, da se pod njim v listek vpiše besedilo in ga vanj povleče. V stolpec »went well« lahko člani ekipe vpišejo, kaj se jim zdi, da je bilo dobro izvedeno v trenutnem teku. Dober primer opazke kaže slika 17: »Naloge v teku so bile dobro razporejene. Tek sem lahko uspešno zaključil.« Poleg tega lahko v stolpec vpišejo pohvale drugim sodelavcem. V drugi stolpec (»to improve«) se lahko vpiše stvari, za katere menimo, da bi se jih lahko izboljšalo. V poštev pride katerakoli stvar, ki je bila prisotna skozi tek. To je na primer lahko opazka, da je bilo v teku preveč nalog, ki so bile med seboj tesno povezane. Dodatne primere za vsak stolpec posebej prikazuje slika 17.

Slika 17: Spletna aplikacija RetroTool



Vir: RetroTool (2021).

Ko se čas petih minut izteče, ekipa preneha z vnašanjem v omenjena stolpca. Skrbnik metodolgije se najprej osredotoči na stolpec »went well«. Vsak vnos – besedilni list – na glas prebere. Če je prebrani vnos nerazumljiv, vpraša tistega člana ekipe, ki je to napisal. V nasprotnem primeru prebere naslednji vnos. Na tak način prebere vse podane vnose in se nato osredotoči na stolpec »to improve«.

V stolpcu »to improve« člani ekipe vpišejo stvari v teku, za katere vidijo, da bi jih bilo treba izboljšati. Omejitve glede ustreznosti opazk ni. Vsak član lahko vpiše težavo, ki se nanaša na celotno ekipo ali določenega posameznika. Seveda ni nujno, da je težava samo v ekipi. Možno je dopisati katerokoli stvar, ki je na nek način slabo vplivala na tek. Vpišemo lahko tudi stvari, za katere vidimo, da bodo v prihodnosti povzročale težave za uspešno zaključevanje teka.

Po obravnavanem stolpcu »went well« se skrbnik metode osredotoči na vnose, vpisane v stolpcu »to improve«. Načeloma je za vsak vnos potrebno več razprave kot ob vnosih v stolpcu »went well«. Kot omenjeno, skrbnik metode vsak vnos na glas prebere. Nato člana ekipe, ki je vnos vpisal, prosi, da poda dodatno mnenje ali pri nejasnosti obrazloži napisani problem. Ostali člani ekipe lahko obravnavani problem pokomentirajo in dodajo svoje mnenje. Vnosi v stolpcu »to improve« nosijo dodatni pomen glede na vnose v stolpcu »went well«. Cilj vsakega vnešenega problema je, da se zanj najde ustrezna rešitev. Za ta namen je v aplikaciji tudi zadnji stolpec »dejanja« (na sliki 17 glej pod »actions«). Kot je bilo omenjeno, je na sestanku do tega trenutka stolpec »actions« še prazen. Naloga skrbnika metode je, da za vsak vnos v stolpcu »to improve« s pomočjo ekipe poskuša ustvariti nov vnos z ustrezno rešitvijo v stolpcu »actions«. Ekipa skupaj sodeluje pri iskanju rešitev. Ko

so za vsak problem uspešno zasnovali rešitev, se ekipa zaveže, da jih bo v naslednjem teku aktivno poskušala upoštevati.

Dober primer problema in zanj ustrezne rešitve je pripravljenost na sestankih. Dogajalo se je, da nekateri člani ekipe na sestankih za izpopolnitev zahtevkov niso učinkovito sodelovali. Poleg tega je ekipa porabila preveč časa za predstavitev in diskusijo kompleksnejših nalog. To se je dogajalo predvsem za večje in bolj zahtevne naloge. Rešitev za podani problem je bila, da mora pred sestankom za izpopolnitev zahtevkov vsak član ekipe sam prebrati vse obravnavane naloge. Na tak način so vsi člani ekipe pred začetkom sestanka poznali vsebino nalog. Poleg tega so lahko hitreje razrešili nejasnosti in skrajšali čas za razpravo o nalogi. To je lep primer nekega problema in najdene ustrezne rešitve.

Naslednji konec teka ekipa preveri, ali je katero od zadanih nalog v stolpcu »actions« uspešno opravila. To poteka na način, da skrbnik metode pred začetkom vnašanja novih vnosov v stolpca »went well« in »to improve« pregleda vnose v stolpcu »actions«. Skrbnik metode prebere vsak vnos in se skupaj s pomočjo ekipe odloči, ali je bil zastavljeni cilj uspešno dosežen. Če cilj ni bil izpolnjen, omenjeni vnos prepiše v stolpec »actions« za naslednji tek (za vsak tek je aplikacija retrotool.com na začetku brez vnosov). To pomeni, da bo ekipa morala na prestavljenem cilju aktivno delati tudi naslednji tek. Če je bil cilj uspešno dosežen, skrbnik metode vnosa ne prestavi v stolpec »actions« za naslednji tek. Na tak način delovanja lahko ekipa aktivno spremlja reševanje svojih problemov v teku. Poleg tega lahko dobro spremlja svoje napredke. Na koncu sestanka skrbnik metode vpraša, če bi kdo hotel predebatirati še kakšno dodatno stvar. Za tem se sestanek zaključi.

3.2 Glavne pomanjkljivosti trenutne metode

Trenutna metoda scrum, ki je bila opisana v prejšnjem poglavju, se je izkazala kot zadovoljiva. Opazili pa smo, da se nekatere stvari v trenutnem delovanju lahko dodatno izboljša. V tem poglavju bomo predstavili glavne pomanjkljivosti trenutno uporabljene metode scrum.

3.2.1 Problem časovnega termina in podaljševanja vsakodnevnih preglednih sestankov

Namen vsakodnevnega preglednega sestanka je hitro in učinkovito dobiti vpogled v trenutno stanje teka vsakega člana ekipe. Opazili smo, da se velikokrat zgodi, da se pri predstavitvi trenutnega stanja dela programerjev zgodi, da se pojavijo vprašanja, ki zahtevajo več časa za razrešitev. To se je že večkrat zgodilo pri več kot enem članu ekipe. Posledica tega je, da se je sam sestanek zavlekel tudi do trideset minut – kar je dvakrat toliko kot načrtovano. Problem tega je, da ima lahko nekdo takoj po vsakodnevnem preglednem sestanku

načrtovane druge sestanke. Poleg tega se izgubi sam pomen kratkega in učinkovitega sestanka.

Pri vsakodnevnem preglednem sestanku se je tudi izkazalo, da ekipi termin sestanka ne ustreza. Trenutni začetek sestanka je ob 10:00 dopoldne. Večina članov ekipe prične delo ob 7:30 ali celo prej. Razlika od začetka dela do sestanka je torej vsaj 2 uri. To lahko vpliva na učinkovitost dela zaposlenih. Primer tega je, če ima programer določen problem, za katerega bi se potreboval posvetovati s celotno ekipo. Od začetka delovnega dne do vsakodnevega preglednega sestanka bi moral počakati 2 uri, preden bi lahko svoj problem predstavil ekipi. Zaradi različne delovne kulture obeh držav, iz katerih so člani ekipe, nekateri programerji pridejo na delo tik pred vsakodnevnim preglednim sestankom. Njihove delovne navade so precej drugačne od navad programerjev iz Slovenije. Zaradi tega ne moremo pričakovati, da bodo vsi člani ekipe na delu prisotni pred vsakodnevnim preglednim sestankom. To pomeni, da je potrebno za pomoč pri določenih problemih večkrat do omenjenega sestanka počakati. Čas sestanka bi zaradi tega bilo potrebno ustrezno prilagoditi.

3.2.2 Pripravljenost na sestankih za izpopolnitev zahtevkov in problem obravnavanja večjih zahtevkov

Kot je bilo že omenjeno, pri trenutni uporabi metode scrum obstaja problem obravnavanja bolj zahtevnih in večjih nalog. To se vidi predvsem na sestankih za izpopolnitev zahtevkov. Ko je treba predstaviti kako večjo nalogo, skrbnik izdelka za sam uvod potrebuje ogromno časa. Med predstavitvijo morajo programerji hitro razumeti pomen zahtevkov in pripraviti ustrezna vprašanja za razrešitev nejasnosti. Večkrat se je zgodilo, da se zaradi tega kakšna pomembna stvar znotraj naloge ni obravnavala. Programerji v kratkem času za večje naloge preprosto ne morejo predvideti vseh možnih težav, ki se lahko pojavijo med opravljanjem naloge. Zaradi tega se je že večkrat zgodilo, da se je med tekom za tako nalogo ugotovilo, da je ekipa pozabila dodati ali razrešiti enega izmed ključnih delov naloge. Posledično je bilo za to nalogo treba nameniti dodatni čas za razrešitev problema. Tako naloga zaradi tega mnogokrat ni bila uspešno izpeljana znotraj teka.

Na sestankih za izpopolnitev zahtevkov je ekipa opazila še eno težavo, povezano z večjimi nalogami. Večkrat se zgodi, da skrbnik izdelka ustvari zahtevek s pomanjkljivimi informacijami, ker preprosto nima dovolj tehničnega znanja, da bi lahko opisal, kako se mora neka funkcionalnost implementirati. V tem primeru na sestanku za izpopolnitev zahtevkov prosi ekipo, da skupaj vpišejo manjkajoče zahteve ali informacije. Večinoma mu pri tem pomagajo programerji, ki se najbolj spoznajo na obravnavano temo. Problem takega načina dela je, da ostali programerji med tem samo poslušajo in v postopku ne sodelujejo. Zaradi tega se veliko časa porabi za razreševanje problemov, pri katerih ni nujno prisotna celotna ekipa.

3.2.3 Način ocenjevanja zahtevkov

Trenutni način ocenjevanja težavnosti zahtevkov – dodeljevanje točk – ima veliko težavo. Kot je bilo že omenjeno, je postopek ocenjevanja tak, da se za ocenjevani zahtevek v klepetalnik vpiše ocena vsakega programerja. Težava je v tem, da programer lahko za kratek čas vidi oceno, ki jo je pred njim vpisal drug sodelavec. Zaradi tega ta ocena lahko vpliva na vpis njegove ocene. Posledica tega je, da so lahko nekatere naloge ocenjene z nerealnim številom točk. Če naj bo naloga dobro ocenjena, jo mora neodvisno oceniti vsak programer zase. Vsak programer ima namreč svojo raven znanja o določenih tehničnih in poslovnih znanjih. To pomeni, da nekateri bolje poznajo določene teme kot drugi, zato lahko tudi bolj natančno ocenijo, koliko truda zahteva izpeljava določene naloge. Na drugi strani pa bodo drugi programerji, ki nimajo toliko znanja o določeni temi, za isto nalogo potrebovali več časa in truda. Točkovna ocena naloge mora tako izražati srednjo vrednost, ki je potrebna za implementacijo naloge. Če programerji vidijo oceno drugih in zaradi tega podajo svojo prilagojeno oceno, lahko to povzroči problem. Naloga je posledično lahko ocenjena z nerealnim številom točk. Programerji, ki dobro poznajo tehnično in poslovno stran naloge, jo bodo implementirali pred tistimi, ki imajo manj znanja. Zaradi tega je potrebno, da se v ocenjevanju naloge upoštevata znanje in tehnična usposobljenost vseh programerjev.

3.2.4 Praktična vloga skrbnika metode

Naša ekipa je sestavljena iz skrbnika izdelka, skrbnika metode in sedmih programerjev. Vsi programerji imajo že več let izkušenj z uporabo agilnih metod. Zelo dobro poznajo tudi metodo scrum. Zaradi tega jim namen in struktura sestankov v scrumu nista tuja.

V ekipi je skrbnik metode zadolžen za organiziranje sestankov. Zaradi dela na daljavo morajo vsi sestanki biti zabeleženi v koledarju v aplikaciji za komunikacijo. Časovni termini sestankov se zelo redko spreminjajo, zato dodatnega dela pri tem ni. Na vsakem sestanku pa je prisoten skrbnik metode, ki je zadolžen za tekoče potekanje sestanka. V primeru vsakodnevnega preglednega sestanka skrbnik metode naredi uvod in določi prvega člana ekipe, da pove svoje stanje dela. Nato določi naslednjega člana, dokler vsi ne pridejo na vrsto. Če kakšen član ekipe razlaga določen problem predolgo, ga skrbnik metode na to opozori in predlaga, da se problem reši, ko bodo vsi člani ekipe prišli na vrsto. Na sestankih za izpopolnitev zahtevkov ima podobno vlogo. Po narejenem uvodu preostanek sestanka vodi skrbnik izdelka. Če določena razprava o zahtevkih postane predolga, skrbnik metode na to opozori. S tem se ne izgubi preveč časa za posamezne zahtevke. Izkazalo pa se je, da so po določenem času delovanja v ekipi sodelavci že sami ugotovili, katera je trajala predolgo. Poleg tega se je že večkrat zgodilo, da je zaradi ponavljajoče se strukture vsakodnevnih preglednih sestankov vlogo skrbnika metode občasno prevzel ali skrbnik izdelka ali eden izmed programerjev.

Poleg tega je skrbnik metode zadolžen za varovanje ekipe pred zunanjimi motečimi dejavniki. Kot je bilo omenjeno, naša ekipa sodeluje na daljavo. To pomeni, da ne sedi v eni pisarni, ampak se nekateri člani ekipe nahajajo v Sloveniji, drugi pa v tuji državi. Komunikacija med njimi poteka preko spletnih klepetalnikov in video klicev. Če katerikoli od nosilcev interesov ali uslužbencev izven ekipe kaj potrebuje, to lahko napiše v spletni klepetalnik programerju, ki ga za to potrebuje. Programer se glede na velikost dela, potrebnega za izpolnitev prošnje, odloči, ali bo problem rešil takoj ali ne. Če je problem prevelik, prosilca napoti, naj v seznamu zahtevkov ustvari nov zahtevek z željenim izidom. Ker so programerji v naši ekipi že dobro izkušeni in samostojni, menimo, da v takih primerih ne potrebujejo pomoči skrbnika metode. Menimo tudi, da zaradi izkušenosti programerjev v metodi scrum in sami naravi dela na daljavo vloga skrbnika metode ni tako pomembna in učinkovita, kot je to opisano v teoriji metode scrum. Zaradi tega se je postavilo vprašanje praktične vloge skrbnika metode.

3.2.5 Problem dokončevanja večjih zahtevkov

Med delovanjem v ekipi smo opazili velik problem. Večkrat se je dogajalo, da določeni zahtevki niso bili uspešno končani do konca teka. Posledično smo morali nedokončane zahtevke premestiti v naslednji tek. Zaradi tega ekipa uradno ni izpolnila cilja teka. To je vplivalo tudi na njen ugled pri nosilcih interesov.

Ekipa je opazila, da imajo nedokončane naloge v tekih eno ključno lastnost. Večina nalog je bila ocenjena s težavnostjo 8 točk. Redko se je zgodilo, da je imela naloga oceno 5 ali manj točk. Poleg tega je bilo za izpostavljene zahtevke znano, da je bilo treba za njihovo uspešno implementacijo pred samim začetkom narediti dodatne raziskave problema. Primer take raziskave je, ko je bila za neko funkcionalnost, ki jo je bilo treba narediti v eni aplikaciji, potrebna tesna povezava z drugo, zunanjo aplikacijo. Zaradi tega je najprej treba raziskati, kako deluje omenjena zunanja aplikacija. Eno izmed bolj zamudnih del med samim raziskovanjem je dogovarjanje z ekipo, ki je zadolžena za aplikacijo. Večkrat se zgodi, da določeni člani ekipe niso odzivni ali pa ne poznajo vseh stvari, ki jih potrebujemo. Zaradi tega je večkrat treba stopiti v stik z več kot eno osebo. Preden dobimo vse potrebne informacije, lahko traja tudi do nekaj dni ali celo tednov. Glede na našo dvotedensko strukturo teka to predstavlja velik problem. Programer lahko dobi vse potrebne informacije šele proti koncu teka. To pomeni, da ima večinoma za samo dokončanje premalo časa. Na tak način se zahtevek ne dokonča znotraj načrtovanega teka.

Naslednji problem večjih zahtevkov je preglednost opravljenega dela. Če se nek zahtevek ni uspešno dokončal v teku, ga je bilo potrebno prestaviti v naslednjega. Večinoma je bila vsaj polovica implementacije zahtevka že narejena. Za dokončanje zahtevka je bilo sicer potrebno malo dela, vendar z vidika uspešnosti teka zahtevek ni bil dokončan. Nosilci interesov ne morejo videti, koliko dela je že opravljenega, saj jim tehnična plat ni vidna. To

se še posebej opazi v procesu prestavljanja obstoječega zahtevka iz končanega v novi tek. Ko ekipa ne opravi določenega zahtevka v teku, se njegovo število točk težavnosti ne spremeni. To je predvsem problem, kadar je opravljenega že večino dela. To posledično vpliva tudi na natančnost načrtovanja naslednjega teka.

Dober primer je, da je nedokončana naloga ocenjena z 8 točkami. To je načeloma zelo veliko. Poleg tega pa je bilo v prejšnjem teku za omenjeni zahtevek opravljenega večino dela. V resnici je zahtevek vreden na primer 1 ali 2 točki. Zaradi trenutnega načina dela v ekipi pa se zahtevek premesti v naslednji tek z nespremenjenim številom točk. To pomeni, da kapaciteta celotnega teka v resnici ni dosežena. Dejansko je v tek zajetih premalo zahtevkov. To se posledično vidi po tem, da je že pred sredino teka omenjeni zahtevek dokončan. Do konca teka pa mora ekipa večinoma vzeti nove zahtevke, ki niso bili načrtovani v teku, saj jim dela zmanjka predčasno.

3.2.6 Problem reševanja hroščev

Ena izmed težav, ki preprečuje uspešnost končanih tekov, so zahtevki o hroščih. Reševanje hroščev je težavna naloga. Programer težko oceni, koliko dela je treba za njihovo razreševanje. Vzrokov za nepravilno delovanje aplikacije je lahko več. To pomeni, da mora programer preveriti več različnih delov kode, preden najde vzrok nepravilnosti. Sam proces lahko traja zelo dolgo časa. V drugih primerih pa so popravki za hrošče lahko izredno kratki. Problem je v tem, da ko programer prebere opis hrošča v zahtevku, večinoma ne more vedeti, kaj povzroča neželjeno delovanje.

Zaradi tega je trenutni način ocenjevanja hroščev drugačen od drugih zahtevkov. Ker programerji ne morejo vedeti, kaj bi lahko bil vzrok, da se je hrošč pojavil, njihovim zahtevkom ne določamo težavnostnih točk. Hkrati pa je skrbnik metode v svojem izračunu kapacitete teka (opisanem v poglavju »Koledar odsotnosti in izračun kapacitete teka«) upošteval, da bo za vsak tek potreben dodaten čas za reševanje hroščev. Tako se je splošna kapaciteta teka zmanjšala. Z vidika programerjev je to sprejemljivo, saj težko ocenimo nek zahtevek, če ne vemo, koliko dela je zanj potrebnega. Problem nastane pri transparentnosti opravljenega dela v teku. Nosilci interesov lahko ocenijo uspešnost določenega teka le po številu opravljenih točk. Vsi zahtevki, razen hroščev, imajo svoje določeno število točk. To pomeni, da hrošči nimajo vpliva na kapaciteto teka. Dober primer je, če bi bila kapaciteta nekega teka 20 točk. Tek bi vseboval 4 zahtevke, ocenjene s 5 točkami. Poleg tega pa bi bili dodani 3 hrošči. Predpostavimo lahko, da tek ni bil uspešno dokončan. Za nekatere hrošče se je izkazalo, da je bilo treba opraviti veliko raziskovanja, preden se je našel njihov vzrok. Posledično se je opravilo le polovico ocenjenih zahtevkov. Velik problem tega je, da nosilci interesov ne vidijo dela, opravljenega za razreševanje teh hroščev. Vidijo le število opravljenih točk. Z njihove strani se vidi le, da je ekipa uspela opraviti le polovico načrtovanega teka. To predstavlja velik problem glede transparentnosti, saj se velik del truda

ekipe ne vidi. Zaradi tega je potrebno najti rešitev za pregledno prikazovanje opravljenega dela pri reševanju hroščev.

3.2.7 Problem osredotočenja na pomembne pomanjkljivosti na sestanku za recenzijo teka

Na sestankih za recenzijo teka se je večkrat zgodilo, da je ekipa občutno prekoračila časovni okvir sestanka. Večinoma je bil problem v tem, da je bilo vnosov v stolpca »went well« in »to improve« zelo veliko. Kot je bilo že omenjeno, je ekipa vsak vnos posebej prebrala in ga komentirala. Za vsak vnos se lahko razvije pogovor, v katerem poskušamo ugotoviti, zakaj je do tega prišlo in kako bi lahko to rešili. Če je bilo vpisanih vnosov veliko, so razprave postale predolge. Zaradi tega se je sestanek moral končati, preden je ekipa pregledala vse vnose.

4 PREDLOG IZBOLJŠANE METODE SCRUM

Široko sprejetje agilnih metod je močno spremenilo svet razvoja programske opreme. Agilna metoda scrum je postala zelo razširjena v omenjeni panogi. Jasno je tudi, da v praksi končna implementacija scruma zaradi različnih razlogov redko upošteva priporočena vodila. Nekatere spremembe so seveda sprejemljive, zgodi pa se lahko, da podjetja ne vedo, kakšne posledice lahko povzroči njihova spremenjena uporaba metode (Eloranta, Koskimies & Mikkonen, 2016).

Trenutno uporabljena metoda scrum ima veliko pomanjkljivosti. Opisane so bile že v poglavju »Glavne pomanjkljivosti trenutne metode«. Med trenutno uporabo metode scrum je postalo jasno, da je pomanjkljivosti treba odpraviti. S pomočjo strokovne literature in praktičnega sodelovanja v ekipi smo pripravili glavne predloge za izboljšavo. Ob njihovem upoštevanju lahko ekipa izboljša svojo trenutno uporabo metode scrum.

Spodaj našteti predlogi so bili predstavljeni tudi nadrejenim. Strinjali so se, da jih ekipa začne uporabljati v naslednjih tekih.

4.1 Strukturirani vsakodnevni pregledni sestanki

Pri izvajanju vsakodnevnih preglednih sestankov smo opazili več pomanjkljivosti. Ena izmed njih je struktura samega sestanka. Kot je bilo omenjeno, vsak član ekipe na sestanku predstavi svoje stanje dela in probleme, s katerimi se spopada. Velikokrat se je že zgodilo, da se je razprava o možnih rešitvah problema tako razvila, da se je sestanek podaljšal na skoraj dvojno dolžino. Poleg tega vsi člani ekipe v razpravi niso sodelovali. Zaradi tega je bil njihov čas porabljen neučinkovito. Za rešitev tega problema smo se odločili, da se

struktura sestanka prilagodi. Če član ekipe za svoj problem prosi ekipo za pomoč, se glede na kompleksnost možne rešitve določi čas razprave. Če je takoj razvidno, da bo za problem potrebno več minut razprave, se najprej vpraša, kdo mora pri njej sodelovati. Določeni člani ekipe se bodo na koncu zbrali na dodatnem uskladitvenem sestanku. Ta je zelo učinkovit, saj v njem sodelujejo le tisti člani ekipe, ki so zanj potrebni. S tem razbremenimo vse ostale člane, da se lahko osredotočijo na svoje delo. Sestanek traja približno 15 minut. Če je za razpravo treba več časa, za to ni ovir, saj sestanek časovno ni omejen. To je pomembno predvsem zato, da se za vsak izpostavljeni problem najde ustrezna rešitev. S pomočjo sestanka za uskladitev problemov se časovna učinkovitost vsakodnevnega sestanka poveča. Možnost podaljševanja vsakodnevnega sestanka pa se zelo zmanjša, saj se vse večje razprave premesti v ločeni sestanek.

Problem vsakodnevnih preglednih sestankov je predstavljal tudi termin samega sestanka. Kot je bilo omenjeno, je slovenska kultura dela drugačna od tiste v tuji državi, v kateri dela preostanek ekipe. Za možni posvet, pri katerem bi bila potrebna celotna ekipa, moramo počakati do vsakodnevnega preglednega sestanka, ki je po trenutni metodi dela ob 10:00. Med začetkom delavnika, ki je za nekatere člane od 7:30 naprej, do potrebnega sestanka je torej vsaj dve uri. To predstavlja prevelik časovni premor. Zaradi tega predlagamo termin, ki bi bolje ustrezal celotni ekipi. Vsakodnevni pregledni sestanek bi prestavili z 10:00 na 9:15. S tem se občutno zmanjša neželjena časovna luknja med začetkom dela in sestankom. Poleg tega ta termin glede na kulturo tuje države ni zastavljen prezgodaj.

4.2 Vnaprejšnje priprave na sestanke za izpopolnitev zahtevkov in uvedba zahtevkov spike

Kot je bilo omenjeno, znotraj ekipe obstajata dva pereča problema. Prvi je zagotovo pripravljenost programerjev na sestankih za izpopolnitev zahtevkov, drugi pa neuspešno zaključevanje večjih zahtevkov. Za oba problema bomo predlagali ustrezno rešitev.

Posledice nepripravljenosti programerjev na sestankih za izpopolnitev zahtevkov so časovno podaljševanje sestankov in nenatančno izpopolnjevanje informacij, potrebnih za njihovo dokončanje. Zaradi tega za opisani problem predlagamo naslednje: Pred vsakim sestankom za izpopolnitev zahtevkov naj skrbnik izdelka po ekipnem klepetalniku sporoči zahtevke, ki jih bo ekipa morala obravnavati. Priporočljivo je, da se omenjene zahtevke sporoči vsaj en dan prej, da ima ekipa dovolj časa za vpogled vanje. Obveza vsakega člana ekipe je, da pregleda vse podane zahtevke. Če mu je katerikoli del opisa zahtevka nerazumljiv, si zanj pripravi vprašanja. Možno je, da bo veliko vprašanj sodelavcev enakih ali vsaj podobnih. Na sestanku za izpopolnitev zahtevkov tako ekipa lahko porabi veliko manj časa. Pri predstavitvi vsakega zahtevka skrbnik izdelka ne bo potreboval več ogromno časa, saj bo ekipa že poznala njegovo vsebino. Poleg tega bo s pomočjo vnaprej pripravljenih vprašanj rešenih veliko več nejasnosti. Vsak programer lahko vidi drugačne možne zaplete, ki se

pojavijo med reševanjem zahtevka. S tem se večje število možnih problemov lahko odkrije vnaprej in hkrati upošteva v sami točkovni oceni njegove težavnosti. Tako bodo tudi točkovne ocene zahtevkov bolj natančne, saj bodo programerji imeli več informacij o tem, kaj vse gre lahko med reševanjem zahtevkov narobe in kaj je še dodatno treba opraviti. S tem se bo izboljšala tudi preglednost opravljenega dela, saj bo ekipa porabila približno toliko dela, kot so ga tudi točkovno ocenili. Bolj redko se bo dogajalo, da bo za kak zahtevek zaradi neopaženih problemov treba veliko več dela, kot je bilo zanj ocenjeno. S tem se bo tudi možnost nedokončevanja tekov zelo zmanjšala, saj bo načrtovano delo dobro opredeljeno. Posledično bodo nosilci interesov bolj zadovoljni s celotnim delovanjem ekipe.

Vnaprejšnja priprava na sestanke za izpopolnitev zahtevkov bo zagotovo pripomogla k izboljšanemu dokončevanju zahtevkov in skrajševanju trajanja sestankov. Obstaja pa še problem definiranja informacij za zahtevke, pri katerih skrbnik izdelka nima dovolj tehničnega znanja in razumevanja. To se je zgodilo že za več funkcionalnosti, ki so izključno tehnične. Za take zahtevke je do zdaj skrbnik izdelka vpisal pomanjkljive zahteve in opomnil ekipo, da bo za zahtevek pred samo implementacijo potrebno narediti še dodatno raziskavo, s katero bo zahtevek lahko izpeljan. Zaradi tega je bilo za ocenjevanje takega zahtevka predlagano, da se ga oceni z več točkami. Tak način ocenjevanja pa se je izkazal za neučinkovitega, saj zahtevek ni vseboval skoraj polovice informacij, kako se funkcionalnost lahko implementira. Višja ocena takega zahtevka tudi ni dobra ocena potrebnega dela, saj je informacij za dokončanje zahtevka malo. Zaradi tega predlagamo boljšo rešitev za opredelitev zahtevkov, pri katerih skrbnik izdelka nima ustreznega tehničnega znanja.

Avtorji Morandini, Coleti, Oliviera in Corrêa so v svojem članku podali dobre rešitve, kako se lahko izboljša opredelitev zahtevkov. Ena izmed njih je, da je pomembno, da programerji pomagajo skrbniku izdelka in možnim nosilcem interesov pri opredelitvi vsebine in rangiranju zahtevkov. S tem se čas opredeljevanja zahtevkov zmanjša, sama kvaliteta informacij pa izredno izboljša (Morandini, Coleti, Oliveira & Corrêa, 2021). Glede na predstavljene rezultate raziskave v članku predlagamo naslednjo rešitev. Če je določeni zahtevek za skrbnika izdelka preveč tehnično zahteven, prosi za pomoč enega do dva programerja. Če tadva ocenita, da je vsebino zahtevka možno opredeliti v kratkem času (ne več kot 2 uri), mu pri tem pomagata. Če je za izpolnitev zahtevka treba porabiti veliko več časa zaradi raziskovanja problema in opredeljevanja zahtev, skrbniku izdelka predlagata dodatno rešitev. Eden izmed programerjev v seznamu zahtev ustvari nov zahtevek, imenovan spike.

Spike je posebna vrsta uporabniške zgodbe, ki se uporablja za pridobivanje znanja, potrebnega za zmanjšanje tveganja tehničnega pristopa, boljše razumevanje ali povečanje zanesljivosti ocene zahtevka. Spike ima lahko največji časovni okvir enega teka. Na koncu teka se za spike ugotovi, ali je opravljen ali ne, tako kot kateri koli drug zahtevek. Spike je odličen način za zgodnje zmanjšanje tveganj in omogoča ekipi, da pridobi povratne informacije in razvije razumevanje za prihajajočo kompleksnost zahtevka. Včasih ekipa ni

prepričana, ali lahko dokonča zgodbo zaradi nekaterih potencialnih blokad in verjetno ne more niti oceniti zahtevka. Tako lahko spike upoštevamo kot naložbo za skrbnika izdelka, da ugotovi, kaj je treba zgraditi in kako bo ekipa to zgradila. Skrbnik izdelka en tek pred implementacijo željenega zahtevka določi omejeno delovno kapaciteto ekipe. Ta kapaciteta se bo porabila za raziskovanje spika. Ko je ta rešen, se lahko izpopolnjeni zahtevek vzame v naslednji tek (Visual Paradigm, 2021).

Menimo, da bo uvedba spikov ekipi pomagala pri določanju bolj zahtevnih zahtevkov. Kot je bilo omenjeno, se spikom dodeli termin, v katerem naj bi skrbnik izdelka ali programer raziskal in pridobil potrebne informacije. Za razvidnost opravljenega dela spikov predlagamo naslednje. V trenutno uporabljeni koledar odsotnosti se za vsak vzeti spike v teku vpiše posebna odsotnost programerja, ki ga bo raziskoval. Odsotnost je na sliki 18 predstavljena kot zelen kvadrat s črko »S« ali »S½«. To pomeni, da bo programer v teku raziskoval dodeljeni spike ves delovni dan ali le polovico. Posebna označitev v koledarju odsotnosti je pomembna predvsem zaradi izračuna točkovne kapacitete teka. Kot je bilo omenjeno, je kapaciteta izračunana glede na prejšnje številke doseženih točk tekov in odsotnosti programerjev. Ker spiki nimajo določenih točkovnih ocen, jih je za pravilen izračun kapacitete teka treba upoštevati. V izračunu kapacitete teka ima označitev celega ali polovičnega dela za spike enako vrednost, kot če bi programer vpisal odsotnost za ves dan ali le polovico dneva. S tem bi se razvidnost opravljenega dela ohranila, saj se točkovna kapaciteta teka prilagaja glede na vneseno delo za izbrane spike. Menimo, da bi tak način dela izredno pomagal ekipi pri dokončevanju večjih zahtevkov, saj bi vse nejasnosti raziskali znotraj spikov že vsaj en tek prej. Neuspešno dokončevanje večjih zahtevkov bi se posledično zmanjšalo, povečala pa bi se možnost uspešnega zaključevanja naslednjih tekov.

Slika 18: Koledar odsotnosti z vključenimi spiki

Planning coef	0.48	July																										August												
	FTE	26					27					28					29					30					31					32					33				
	6.8	28	29	30	1	2	5	6	7	8	9	12	13	14	15	16	19	20	21	22	23	26	27	28	29	30	2	3	4	5	6	9	10	11	12	13	16	17	18	19	20
Programer 1	1								½															A													A				
Programer 2	1			A	A	A												A	A	A	A													A	A	A	A	A	A	A	A
Programer 3	0.8								½					A	A	A	A	A	A	A	A	A	A						S	S		½		½		½		½		½	
Programer 4	1						A		½		½															A						A	A	A	A	A	A	A	A	A	A
Programer 5	1								½									S	S ½									S	S					A					S	S	
Programer 6	1			A	A					S ½											A									A	A			S							
Programer 7	1										½						½							A	A		A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
FTE average	0.97			Sprint #13										Sprint #14										Sprint #15										Sprint #16						
Recommended capacity, SP	125			28										25										24										14							
Plan for Q3 roadmap, SP	153																																								

Vir: lastno delo.

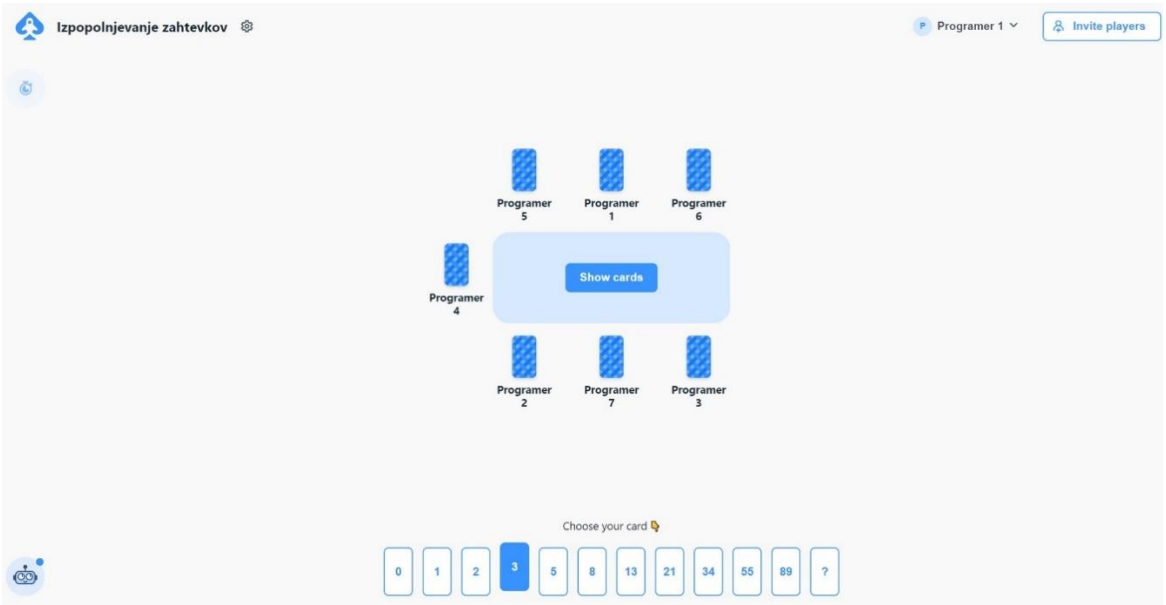
4.3 Ocenjevanje zahtevkov po metodi Igra načrtovanja

Trenutno ocenjevanje zahtevkov, opisano v poglavju »Način ocenjevanja zahtevkov«, je zelo vprašljiv glede avtentičnosti ocen. Kot je bilo omenjeno, člani ekipe v klepetalnik v video klicu vpisujejo svoje ocene zahtevkov. Zaradi tega se večkrat zgodi, da vsi ne vpišejo

svojih ocen naenkrat, zato lahko za kratek čas vidijo ocene sodelavcev. To lahko vpliva tudi na hitro spremembo njihove ocene, spremenjene ocene pa ne odražajo mnenja programerja, ampak vpliv drugih. Zaradi tega so lahko nekateri zahtevki ocenjeni s premalo ali preveliko točkami, saj vsak programer pozna določene teme bolje kot drugi. Zaradi tega bi predlagali izboljšan način ocenjevanja zahtevkov.

V poglavju »Orodja za boljšo izvedbo metode scrum« smo predstavili zelo uporabno spletno orodje Igra načrtovanja. Predlagamo, da ekipa začne uporabljati to orodje, saj menimo, da bo s tem rešila problem vpliva videnja ocen drugih sodelavcev. Aplikacija simulira fizično igranje s kartami. Skrbnik izdelka v aplikaciji ustvari novo igro in preko spletnega naslova povabi vse potrebne sodelavce. Kot je razvidno s slike 19, ima za vsako ocenjevanje zahtevka igralec na voljo različne karte z vrednostmi Fibonaccijevega zaporedja. Ko igralec izbere svojo oceno, se njegova karta modro obarva. Ko so vse karte izbrane, skrbnik izdelka pritisne na gumb »show cards« in karte se naenkrat obrnejo. Razprava o ocenah se zatem lahko prične, kot je opisano v poglavju »Sestanek za izpopolnjevanje zahtevkov«. Menimo, da se bo z uporabo opisane aplikacije postopek ocenjevanja zahtevkov izredno izboljšal.

Slika 19: Aplikacija Igra načrtovanja



Vir: We agile you S. L. (2021).

4.4 Odstranitev skrbnika metode

V članku »Agile Requirements Prioritization in Practice: Results of an Industrial Survey«, kjer je bila opisana raziskava o praktičnem izvajanju procesov po metodi scrum, je bilo prav tako ugotovljeno, da se načrtovane ekipne vloge znotraj procesov v realnosti razlikujejo. Iz

rezultatov raziskave je razvidno tudi, da je za različne ekipe skrbnik metode sodeloval v procesih, ki niso predlagani v smernicah metode scrum. Zgodilo pa se je tudi, da ni sodeloval v ključnih aktivnostih, kjer naj bi sodeloval po smernicah metode scrum (Jarzębowicz & Sitko, 2020). Kot je bilo omenjeno, se uporabe metode scrum v praksi lahko zelo razlikujejo. To je predvsem odvisno od okolja in načina dela v podjetju. Naša trenutna ekipa je sestavljena iz skrbnika izdelka, skrbnika metode in programerjev. Tako skrbnik izdelka kot vsi programerji imajo že več let izkušenj v uporabi metode scrum. V poglavju »Praktična vloga skrbnika metode« smo poudarili, da je vpliv skrbnika metode v ekipi vedno manjši. Menimo, da je eden izmed razlogov za to predvsem delo na daljavo. Ekipa tudi ni tako povezana, kot če bi delala v enem prostoru. Po drugi strani pa je vsak programer veliko bolj sposoben in samostojen pri omejevanju zunanjih ovir (kot je bilo omenjeno v poglavju »Praktična vloga skrbnika metode«). Opazili smo, da se je skozi čas veliko zadolžitev skrbnika metode počasi prenašalo na programerje in skrbnika izdelka. Eden izmed primerov je organiziranost na vsakodnevnih preglednih sestankih. Ker je imel skrbnik metode večinoma enako zadolžitev – uvod v sestanek in določitev vsakega člana ekipe k predstavitvi stanja njegovega dela, so člani ekipe skozi čas včasih posamično prevzemali njegovo vlogo. To se je zgodilo tudi takrat, ko je bil skrbnik metode odsoten. Poleg tega se je vedno pogosteje dogajalo, da se je ekipa sama ustavila, če je kakšen sestanek trajal čez časovno določitev, in predlagala nadaljevanje v naslednjem sestanku. Tako je bilo potrebnih posredovanj skrbnika metode vedno manj. Enako se je začelo dogajati tudi na ostalih sestankih. Vodenje sestankov za izpopolnitev zahtevkov je vse pogosteje prevzemal skrbnik izdelka. To pa predvsem zaradi tega, ker je imel več domenskega znanja o predebatiranih zahtevkih. Samo vodenje sestankov se je zaradi tega prenašalo tudi na skrbnika izdelka.

Organiziranost sestankov skozi celotni tek je vgrajena v aplikacijo za video klice. Termin sestankov je vpisal skrbnik metode. Dobra stran aplikacije je, da se termini za vsak naslednji tek ponovijo. Zaradi tega ni dodatnega dela pri njihovi organizaciji. Če je časovni termin kakšnega sestanka potrebno spremeniti, to lahko opravi tudi skrbnik izdelka. S tem se lahko tudi organizacija vseh sestankov znotraj teka prenese nanj.

Zaradi zgoraj omenjenih razlogov menimo, da vloga skrbnika metode v trenutni ekipi ni potrebna. Skozi čas delovanja v ekipi smo opazili, da je predvsem na sestankih skozi tek skrbnik metode vse pogosteje prepuščal vodenje sestankov programerjem ali skrbniku izdelka. Vsi člani ekipe so dovolj izkušeni, da so lahko izmenično vodenje uspešno prevzemali. Poleg tega se je časovna prekoračitev sestankov dogajala vse redkeje. Pri uvedbi odstranitve skrbnika metode pa predlagamo možno rešitev pri obvladovanju zunanjih ovir. Če kdo izmed zunanjih nosilcev interesov ali sodelavcev potrebuje pomoč, ki ni načrtovana znotraj teka, naj za to ne sprašuje neposredno programerja, ampak skrbnika izdelka. Menimo, da bo to boljši pristop predvsem zaradi tega, ker skrbnik izdelka zelo dobro pozna aplikacijo in delo v ekipi. Z njegovo pomočjo bo najverjetneje lahko rešil svoj problem. Poleg tega se s tem ne moti dela programerjev, ki so med tekom osredotočeni na svoje delo. Predlagana odstranitev skrbnika metode naj ne bi poslabšala trenutnega načina dela ekipe.

To menimo predvsem zaradi tega, ker so se znaki prevzemanja njegove vloge začeli kazati že skozi čas delovanja v ekipi.

4.5 Ponovno ocenjevanje nedokončanih zahtevkov

Eden izmed večjih problemov v zvezi s preglednostjo opravljenega dela je bilo nedokončevanje večjih zahtevkov. Trenutni način dela je, da se v primeru nedokončanega zahtevka njegova točkovna ocena v naslednjem teku ne spremeni. To predstavlja velik problem nenatančnosti načrtovanega dela v naslednjem teku. Kot je bilo omenjeno, lahko zaradi nespremenjene ocene zahtevka že na sredini teka zmanjka dela za programerje. Posledično je potrebno v tek vzeti nove zahtevke. Tak način dela slabša realno oceno točkovne kapacitete tekov. Za izognitev temu problemu predlagamo naslednjo rešitev: vsak nedokončan zahtevek v prejšnjem teku naj se pred vključitvijo v naslednjega ponovno točkovno oceni.

Predlagamo poseben prilagojen proces ponovnega ocenjevanja nedokončanih zahtevkov. Če na koncu teka ostanejo nedokončani zahtevki, jih ekipa obravnava na sestanku za načrtovanje naslednjega teka. V delu sestanka, kjer se ekipa odloča o zahtevkih, ki jih bo vzela v tek, se najprej posveti tistim, ki so ostali še nedokončani iz prejšnjega teka. Za vsak nedokončani zahtevek programer, ki je na njem delal, predstavi, kaj je do zdaj že naredil in kaj je za dokončanje zahtevka še potrebno storiti. Zatem predlaga svojo prilagojeno točkovno oceno. Načeloma je ta manjša od prvotne. Lahko pa se seveda tudi zgodi, da programer predlaga enako ali celo višjo oceno. Primer tega je lahko, da je programer začel na zahtevku delati šele na koncu teka, hkrati pa se je izkazalo, da je bil zahtevek ocenjen s premalo točkami. Menimo, da so taki scenariji malo verjetni, a še zmeraj mogoči. Po programerjevi podani točkovni oceni skrbnik izdelka vpraša preostali del ekipe, če se s predlagano oceno vsi strinjajo. Če se ekipa strinja, se predlagane točke vpiše v zahtevek. Člani ekipe, ki se z oceno ne strinjajo, predstavijo svoje argumente ali možna vprašanja. Po razpravi se ekipa odloči glede nove točkovne ocene.

Menimo, da bo predlagana metoda ponovnega ocenjevanja nedokončanih zahtevkov pripomogla k boljši preglednosti dela v ekipi, saj bodo zahtevki vedno ocenjeni s točkami, ki predstavljajo realno oceno preostalega dela. Z upoštevanjem te se bo realna ocena kapacitete teka z vsako ponovitvijo izboljšala. Možnost uspešnih zaključkov teka se bo povečala, zmanjšala pa se bo možnost, da bi pred koncem teka ekipi zmanjkalo dela.

4.6 Točkovno določanje hroščev

Kot je bilo omenjeno v poglavju »Problem reševanja hroščev«, v trenutni implementaciji scruma ekipa točkovno ne ocenjuje hroščev. To predstavlja problem nepreglednosti opravljenega dela.

V članku (Eloranta, Koskimies & Mikkonen, 2016) so ugotovili podoben problem nepreglednosti dela. Kadar ekipa ni prikazovala opravljenega dela z grafom napredovanja, njeni člani niso imeli občutka, da je pomembno med tekom končati vse zadane naloge. Take ekipe niso uspešno zaključevale svojih tekov. Posledično zaradi tega tudi niso mogle natančno načrtovati svoje kapacitete prihodnjih tekov, saj niso vedele, koliko točk lahko uspešno opravijo.

Zelo podoben problem predstavlja trenutno neocenjevanje hroščev. Kot je bilo omenjeno, je za vsak tek vštet določen čas za njihovo reševanje. Izračun kapacitete celotnega teka je načrtoval skrbnik metode. Ker pa hrošči niso ocenjeni s številom točk, se delo za njihovo reševanje ne prikazuje na grafu napredka. Za rešitev tega problema predlagamo naslednje. Za boljšo preglednost dela je treba dodeliti ustrezne točke tudi hroščem. Njihovo ocenjevanje pa ima eno pomembno razliko od ocenjevanja drugih zahtevkov. Kot je bilo omenjeno, programer težko ve, kaj je vzrok, da se je nepravilno delovanje aplikacije (hrošč) sploh pojavilo. Zaradi tega težko oceni težavnost dela, ki je potrebno, da se hrošč odpravi, zato predlagamo prilagojeno ocenjevanje. Vsak hrošč ima lahko razpon točkovne ocene od 1 do 3. Točkovna ocena 1 pomeni, da je vzrok za hrošč že znan ter da ekipa ne vidi možnosti dodatnega nenačrtovanega dela. Točkovna ocena 3 pa pomeni, da ekipa ne ve, kaj bi lahko bil vzrok za nepravilno delovanje. Zaradi tega je treba upoštevati, da bo za odpravo hrošča s točkovno oceno 3 potrebno podrobno raziskovanje delovanja aplikacije.

Predlagani razpon točkovne ocene od 1 do 3 ekipi pomaga pri prikazovanju opravljenega dela. Maksimalna ocena 3 prihaja iz razloga, da v primeru, ko za nek hrošč ne vemo njegovega vzroka, ne moremo realno oceniti potrebnega dela. Zaradi tega bi ga lahko v skrajnem primeru ekipa ocenila s točkovno oceno 13 ali več. Če bi se tako ocenjeni hrošč na koncu rešil z lahkoto, bi to močno vplivalo na realno kapaciteto naslednjih tekov. Zaradi tega je še zmeraj bolje, da se to potrebno delo prikaže tako, da ne more preveč vplivati na realno kapaciteto teka. Iz tega razloga predlagamo omejeno ocenjevanje hroščev.

Posledično predlagamo, da se iz izračune kapacitete teka odstrani del, v katerem je upoštevan trud reševanja hroščev, saj bi ti v tem primeru dobili svojo točkovno oceno. Zaradi tega bi bili tudi prikazani na grafu napredka. Ekipa in nosilci interesov bi tako dobili bolj jasen vpogled v opravljeno delo v bodočih tekih.

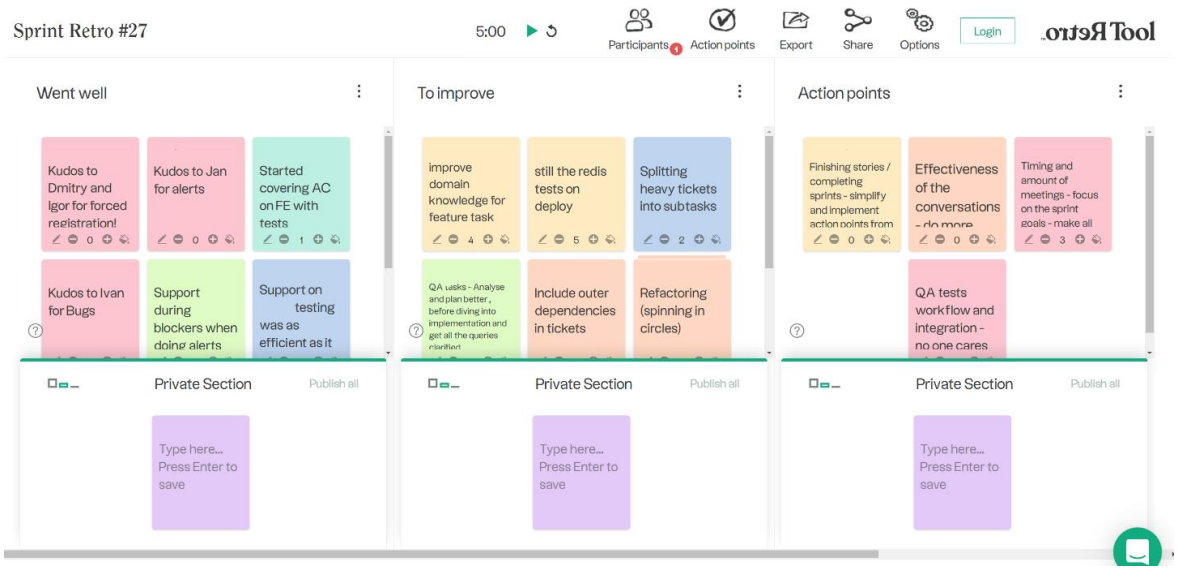
4.7 Izboljšana struktura sestankov za recenzijo teka

Kot je bilo omenjeno, ima ekipa težave pri učinkoviti razpravi na sestankih za recenzijo teka. Problem nastane pri tem, ker ekipa vse vnose v stolpcih »went well« in »to improve« posamično predebatira. Velikokrat se je zgodilo, da so nekateri vnosi bili zelo podobni, zato se je razprava o njih na določen način ponavljala. Posledično je ekipa prekoračila za sestanek določeni čas, ne da bi obravnavala vse vnose.

V organizaciji Mountain Goat Software, kjer se že vrsto let ukvarjajo z vpeljavo scruma v vseh vrst podjetja, za naš problem predlagajo učinkovito rešitev. Ko na sestanku za recenzijo teka vsi člani ekipe vnesejo svoja pozitivna opažanja in ideje za izboljšave, se organizira kratek čas za glasovanje. Vsak član ekipe lahko glasuje za vnose, ki se mu zdijo najpomembnejši. Nato ekipa predebatira vnose, ki so dobili največ glasov (Cohn, 2019).

Menimo, da bi predlagana izboljšava pomagala ekipi pri bolj učinkoviti razpravi med sestankom za recenzijo teka. Poleg tega pa predlagamo dodatno izboljšavo. Pred glasovanjem za najpomembnejše vnose bi lahko same vnose preorganizirali. Vnose, ki so si med seboj podobni, bi lahko združili v manjše skupine, šele za tem pa bi izvedli glasovanje. Primer glasovanja je prikazan na sliki 20, kjer ima vsak vnos pod besedilom prikazano število glasov. Slika je bila zajeta iz uporabljene aplikacije, zato tudi ni prevedena. Na sliki so podobni vnosi tudi že združeni – prikazan je le eden izmed njih. Na tak način bi se izognili ponavljanju razprav glede podobnih vnosov. Poleg tega pa bi ekipa imela več časa za pogovor o najpomembnejših temah. Menimo, da bi se s tem kvaliteta razprave izboljšala, poleg tega pa bi se zmanjšale prekoračitve za sestanek določenega časa.

Slika 20: RetroTool z možnostjo glasovanja za zahtevke



Vir: RetroTool (2021).

SKLEP

V magistrskem delu sem s pomočjo literature raziskal agilnost in v njenem okviru metodo scrum. Uporaba agilne metode scrum je v panogi razvoja programske opreme izredno razširjena. Scrum kot ogrodje predlaga določena vodila, ki pa se v končni implementaciji v praksi redko upoštevajo. Vsaka ekipa si metodo prilagodi glede na produkt, ki ga razvija, in

okolje, v katerem deluje. Namen magistrskega dela je bil v izbrani ekipi izboljšati trenutni proces razvoja programske opreme po metodi scrum, cilj pa najti izboljšano metodo za izbrano ekipo. Izboljšana metoda je prilagojena glede na način dela ekipe v izbranem podjetju. V magistrskem delu sem dosegel vse cilje, zastavljene v uvodu.

V magistrskem delu sem opravil podrobno analizo implementacije metode scrum v izbranem podjetju. Med izdelavo dela sem bil član razvojne ekipe, ki je delala po tej metodi. S pomočjo analizirane literature in predlogov ostalih članov smo kot ekipa nadrejenim predlagali opisane predloge za izboljšavo trenutne metode scrum. Nadrejeni so se strinjali, da ekipa lahko opisane izboljšave začne izvajati v nadaljnjem delu.

Podrobno analizirana metoda scrum v izbranem podjetju lahko podobnim ekipam pomaga pri izboljšavi njihove implementacije trenutno uporabljane metode. Ugotovil sem, da se implementacije te metode med različnimi ekipami razlikujejo. Vsaka ekipa si metodo prilagodi glede na svoj način dela. Mi smo kot ekipa za trenutno uporabljano metodo scrum poiskali dodatna orodja, s katerimi si ekipa lahko pomaga pri njeni izboljšavi. Večina orodij je dostopna na svetovnem spletu, zato so na voljo vsem. Za tem smo predstavili tudi pomanjkljivosti trenutne metode. Čeprav ekipa dela učinkovito, smo opazili možnosti za izboljšave. Za vsako pomanjkljivost smo predlagali tudi ustrezno rešitev. Na enak način se lahko podobne ekipe lotijo izboljšave svoje implementacije metode scrum.

Zaradi same narave metode scrum se lahko novo predlagano metodo vedno da še izboljšati. V vsakem delovnem okolju se način dela in struktura ekipe spreminjata. V skladu s tem se mora prilagoditi tudi uporabljena metoda. Kot je bilo omenjeno, bo izboljšano metodo ekipa v izbranem podjetju začela uporabljati v nadaljnjih tekih. Zelo verjetno je, da se bo skozi čas pokazalo, da so nekatere izboljšave bolj učinkovite kot druge, glede na dobljene rezultate pa bo lahko ekipa uporabljeno metodo še dodatno prilagodila. Jasno namreč je, da se bo skozi nadaljnje delovanje ekipe implementacija metode scrum stalno spreminjala.

LITERATURA IN VIRI

1. Agilemanifesto.org. (2001). *Manifest agilnega razvoja programske opreme.* Pridobljeno 8. februarja 2021 iz https://agilemanifesto.org/iso/sl/principles.html
2. Ahmad, M. O., Markkula & Oivo, M. (2013). Kanban in software development: A systematic literature review. *39th Euromicro Conference on Software Engineering and Advance Applications* (str. 9-16). Pridobljeno 5. marca 2022 iz https://ieeexplore.ieee.org/abstract/document/6619482
3. Anwer, F., Aftab, S., Waheed, U. & Muhammad, S. S. (2017). Agile Software Development Models TDD, FDD, DSDM, and Crystal Methods: A Survey. *International Journal of Multidisciplinary Sciences and Engineering, 8*(2).

4. Appelbaum, B. (2021). Plainview. *Agile Methodologies: A Beginner's Guide.* Pridobljeno 20. marca 2021 iz https://www.planview.com/resources/guide/agile-methodologies-a-beginners-guide/
5. Atlassian. (2021). *What is Agile?* Pridobljeno 8. februarja 2021 iz https://www.atlassian.com/agile/
6. Atlassian Software. (2021). *Team collabor and incremental delivery starts on the scrum board.* Pridobljeno 25. marca 2021 iz https://www.atlassian.com/software/
7. Beck, K. (1999, oktober). Embracing change with extreme programming. *In Computer*, 32, 70-77. Pridobljeno 5. marca 2022 iz https://ieeexplore.ieee.org/abstract/document/796139
8. Birade, O. (2020, 2. februar). *Omkar's Newsletter. Kanban Board.* Pridobljeno 17. julija 2021 iz https://omkarbirade.substack.com/p/kanban-board-b42daf7578e9
9. Bowes, J. (2015, 10. marec). *The definition of done.* Pridobljeno 27. decembra 2021 iz https://manifesto.co.uk/definition-done/
10. Cohn, M. (2019). *Scrum Methodology and Project Management.* Pridobljeno 3. januarja 2022 iz https://www.mountaingoatsoftware.com/agile/scrum
11. Eloranta, V.-P., Koskimies, K. & Mikkonen, T. (2016). Exploring ScrumBut – An empirical study of Scrum anti-patterns. *Information and Software Technology, 74,* 194–203. Pridobljeno 3. januarja 2022 iz https://doi.org/10.1016/j.infsof.2015.12.003
12. James, M. & Walter, L. (2010). *Scrum reference card.* Brisbane: CollabNet Inc.
13. Jarzębowicz, A. & Sitko, N. (2020). Agile Requirements Prioritization in Practice: Results of an Industrial Survey. *Procedia Computer Science, 176,* 3446–3455. Pridobljeno 3. januarja 2022 iz https://doi.org/10.1016/j.procs.2020.09.052
14. Knowledgehut. (2021). *Sprint Review Meeting.* Pridobljeno 22. decembra 2021 iz https://www.knowledgehut.com/tutorials/Scrumtutorial/Sprintreview
15. Mentimeter. (2019). *Interactive presentation software.* Pridobljeno 15. oktobra 2021 iz https://www.mentimeter.com/
16. menti.com. (2019). *Join a presentation – Mentimeter.* Pridobljeno 15. oktobra 2021 iz https://www.menti.com/
17. Morandini, M., Coleti, T. A., Oliveira, E. & Corrêa, P. L. P. (2021). Considerations about the efficiency and sufficiency of the utilization of the Scrum methodology: A survey for analyzing results for development teams. *Computer Science Review, 39,* 100314. Pridobljeno 3. januarja 2022 iz https://doi.org/10.1016/j.cosrev.2020.100314
18. nTask. (2018, 12. november). *Types of Scrum Meetings and Scrum Best Practices* [objava na blogu]. Pridobljeno 31. julija 2021 iz https://www.ntaskmanager.com/blog/Scrummeetings/
19. Obergfell, Y. (2019, 1. November). Scrum Institute. *What Is A Daily Scrum/Stand-Up Meeting? This Might Surprise You!* Pridobljeno 22. decembra 2021 iz https://www.Scruminstitute.org/daily-Scrummeeting-daily-stand-up-meeting-the-Scrumframework.php

20. Objective Based. (2021, 17. februar). *Scrum Sprint Goal Examples* [objava na blogu]. Pridobljeno 27. decembra 2021 iz https://objectivebased.blog/ScrumSprintgoal-examples/
21. Pichler, R. (2021, 9. marec). *Product goals in Scrum* [objava na blogu]. Pridobljeno 27. decembra 2021 iz https://www.romanpichler.com/blog/product-goals-in-scrum/
22. Poppendieck, M. & Cusumano, M. A. (2012). *Lean Software Development: A Tutorial. IEEE Software*, 5, 26-32. Pridobljeno 5. marca 2022 iz https://ieeexplore.ieee.org/abstract/document/6226341
23. ProductPlan. (2021). *Sprint Backlog.* Pridobljeno 27. decembra 2021 iz https://www.productplan.com/glossary/Sprintbacklog/
24. Rauer, M. (2021). *Confluence 6.7 Neues visuelles Nutzererlebnis und neu konzipierte Erwahnungen* [objava na blogu]. Pridobljeno 2. decembra 2021 iz https://blog.seibert-media.net/blog/2018/02/07/confluence-6-7/
25. RetroTool. (2021). *Free, simple and flexible online retrospectives.* Pridobljeno 15. oktobra 2021 iz https://retrotool.io/
26. Roopesh, J. (2019, 8. februar). *What is a Product Increment in Scrum?* [objava na blogu]. Pridobljeno 27. decembra 2021 iz https://staragile.com/blog/what-is-a-product-increment-in-scrum
27. RUBeL. (2021). *Abstimmungstools: Erweiterte Funktionen von Mentimeter nutzen.* Pridobljeno 2. decembra 2021 iz https://www.rubel.rub.de/news/abstimmungstools-erweiterte-funktionen-von-mentimeter-nutzen
28. Schwaber, K. & Southerland, J. (2010). *Scrum.* Pridobljeno 8. februarja 2021 iz http://www.volaroint.com/wp-content/uploads/dlm_uploads/2014/03/DC-VOLARO-Training-ScrumWhat_Is_Scrum.pdf
29. Scrum.inc. (2016, 24. februar). *Writing Better User Stories.* Pridobljeno 8. februarja 2021 iz https://www.scruminc.com/independent-user-stories/
30. Scrum.org. (2016). *The Home of Scrum.* Pridobljeno 19. avgusta 2021 iz https://www.scrum.org/resources/what-is-scrum
31. Scrum Guide. (2020). *Scrum Guides.* Pridobljeno 17. julija 2021 iz https://scrumguides.org/scrum-guide.html
32. ScrumExplainer. (2020). *What is the Scrum Methodology? A Thorough Guide to the Scrum Framework.* Pridobljeno 3. januarja 2022 iz https://scrumexplainer.com/scrum/Scrummethodology/
33. Shastri, Y., Hoda, R. & Amor, R. (2021). Spearheading agile: the role of the scrum master in agile projects. *Empirical Software Engineering, 26* (1). Pridobljeno 3. januarja 2022 iz https://doi.org/10.1007/s10664-020-09899-4
34. Simplilearn. (2020). *What is Scrum Team: Structure, Roles and Responsibilities*. Pridobljeno 5. marca 2020 iz https://www.simplilearn.com/what-is-Scrumteam-article
35. Southerland, J. (2016). *V gruču do uspeha.* Ljubljana: Založba Pasadena d. o. o.
36. Visual Paradigm. (2021). *What is a Spike in Scrum?* Pridobljeno 28. oktobra 2021 iz https://www.visual-paradigm.com/scrum/what-is-Scrumspike/

37. We agile you S. L. (2021). *Planning Poker Online.* Pridobljeno 15. oktobra 2021 iz https://planningpokeronline.com/