Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko

Od plansko usmerjenih do agilnih razvojnih metod na praktičnih primerih

Študijski program: Informatika in tehnologije komuniciranja

Interno učno gradivo pri predmetu »Razvoj informacijskih sistemov«

Različica »Sept21«

Luka Pavlič, Tina Beranič, Lucija Brezočnik

Maribor, 2021

Naslov	Od plansko usmerjenih do agilnih razvojnih metod na praktičnih primerih
Avtorji	Luka Pavlič, Tina Beranič, Lucija Brezočnik
Strokovni recenzent	Sašo Karakatič
Tipologija	2.05 - Drugo učno gradivo (skripta)
Založnik	Fakulteta za elektrotehniko, računalništvo in informatiko, Univerza v Mariboru
Kraj založbe	Maribor
Povezava	https://dk.um.si/IzpisGradiva.php?id=80966
Datum izida	November, 2021

Vsebina

1. Vloga metod agilnega razvoja

Agilni manifesto [1] je dopolnil že skoraj dve desetletji. Leta 2001 je bil rezultat dolgoletnih prizadevanj, da bi na področje razvoja programske opreme uvedli nove pristope, ki bi omogočali hitrejše odzivanje na spreminjajoče zahteve in (tudi s tem) dvig učinkovitosti razvojnih aktivnosti. Skladnost z manifestom je izkazalo kar nekaj metod. Med njimi so najbolj popularne XP (eXtreme Programming), Crystal, DSDM (Dynamic Systems Development Method), FDD (Feature-Driven Develop-ment), Scrum, SaFE (Scaled Agile Framework), LeSS (Large Scale Scrum) in druge. Danes preko 70% vseh agilnih metod temelji na metodi Scrum [2] [3].

Na prvi pogled so agilne metode že od začetka predstavljale pomemben odklon od desetletja uveljavljenih procesnih modelov, ki temeljijo na natančnem srednje- in dolgoročnem planiranju, jasnem zaporedju opravil, dobro definiranimi procesi in vlogami v njih ipd. Po drugi strani tudi agilne metode, z namenom uveljavljanja inženirskih praks, dokumentiranja rešitev in zagotavljanja kakovostnih izdelkov, od izvajalcev zahtevajo discipliniranost in sledenje pravilom. Prav tako agilne metode niso čarobna paličica, ki bi jo uporabili med razvojem, temveč jih lahko razumemo predvsem kot pomembno orodje obvladovanja tveganj, povezanih s stroški, obsegom in kakovostjo informacijskih rešitev in storitev.

Ker agilnih praks v industriji več ne dojemajo kot neko posebnost, temveč vidijo njihove prednosti, stremijo k uveljavljanju agilnih metod. Ugotavljamo lahko, da se le-te uveljavljajo tudi v projektih, kjer njihova uporaba ni najbolj primerna. Agilnost je vodstveni kader začel dojemati kot pomemben kazalnik zrelosti, sodobnosti razvojnih oddelkov. Sploh, če so vodstva podjetij že bila izpostavljena negativnim izkušnjam zaradi toge zasnove plansko naravnanih, tipično sekvenčnih metod razvoja. Tako je ena izmed dveh poglavitnih ovir pri vpeljavi agilnih metod v prakso, t.j. podpora vodstva, doživela preobrazbo. Pogosto moramo vodstvenemu kadru pojasniti, zakaj v določenih projektih (popolnoma) agilna metoda morda ni najbolj primerna, ali jih odvrniti od vpeljave najbolj popularne metode in predlagati ustrezne prilagoditve. Drugi izziv pri uvajanju agilnih metod v prakso, sploh če ideja za vpeljavo prihaja od zgoraj navzdol, ostaja: spreminjanje načina razmišljanja vpletenih izvajalcev. Zato vpeljava novih metod razvoja ni razumljena kot nepotrebno in dodatno delo. V podjetjih se prenove lotevajo s pomočjo polno zaposlenih posameznikov, ki med drugim opravljajo tudi vlogo t.i. agilnega trenerja (angl. agile coach). Ti so lahko zaposleni v podjetju, zelo pogosto so tudi zunanji izvajalci, ki lahko predhodno naredijo presojo in svetujejo glede izbire in prilagoditve agilnih praks ter jih v drugem koraku pomagajo vpeljati in dodatno prilagoditi glede na izkušnje pilotnega uvajanja.

Izzivi, povezani z izbiro, prilagoditvijo in vpeljavo agilnih metod, so poleg ostalih dejavnikov, različni tudi glede na implicitno stopnjo vitkosti podjetja, ki se za vpeljavo odloči. Agilne metode namreč sledijo vrednotam, ki so zbrane v agilnem manifestu [1]. Izkazalo se je, da je z uporabo tradicionalnih procesov te vrednote težje naslavljati. Zato se agilne, vitke, praviloma iterativne razvojne metode udejanjajo prek več uveljavljenih pristopov, kot so:

- redni dnevni sestanki,
- izvajanje retrospektiv ob določenih mejnikih, tipično ob koncu iteracije,
- redne demonstracije izdelkov,
- ciljno usmerjeno delo v času iteracij,
- kratkoročno planiranje na osnovi prioritet.

Agilnost pomeni tudi, da so v večino zgoraj naštetih pristopov vpleteni tako razvijalci, kot tudi vsebinski delavci in vodstven kader. V odvisnosti od tega, kako določene pristope pozdravljajo ali celo že izvajajo v podjetju, se je možno odločiti tudi za alternativno izvedbo določenih pristopov. Končen cilj pa mora biti izboljšanje komunikacije med vpletenimi v razvojne projekte, možnost hitrega odzivanja na spremembe ter minimiziranje režijskega dela, predvsem v smislu kratkih in usmerjenih sestankov. Zato različne splošno namenske (t.j. neprilagojene) agilne metode poznajo različne izraze in tehnične prijeme za izvedbo pristopov. V metodi Scrum imamo tako tri vloge vpletenih (predstavnik stranke, skrbnik metodologije, razvojna ekipa), nujne dogodke (sprint, planiranje sprinta, dnevni sestanki, demo, retrospektive) in orodja (zahteve rešitve, zahteve sprinta, definicija narejenega). Način dela je tipično organiziran v PDCA (angl Plan-Do-Check-Adjust) cikle s sprotnimi kontrolnimi točkami:

- definiranje in zaveza merljivemu cilju,
- definiranje obvladljivih (majhnih) in vidnih korakov (nalog) do cilja,
- izvedba nalog,
- sprotno merjenje napredka,
- refleksija, uvedba izboljšav in ponovitev.

Ker se pri izboru praks snežna kepa šele začne kotaliti, so glavni izvor izzivov, povezani predvsem z uspešnim uvajanjem dnevnih razvojnih praks. Le-te so pogosto diametralno nasprotne tem, ki jih razvojniki dolga leta izvajajo in zelo pogosto v njih ne vidijo nekih posebnih pomanjkljivosti. Ni namreč nujno, da pred uvedbo agilnih metod razvojno osebje sploh razmišlja na način, da je njihovo delo primarno namenjeno reševanju informacijskih problemov svojih strank preko manjšega števila dorečenih funkcionalnosti in ne npr. razvoj cele množice funkcionalnosti, za katere niti natančno ne vedo, kakšen je njihov namen.

2. Način zasnove in vpeljave metode agilnega razvoja

Ekipe se za prestop na agilni razvoj odločajo v največji meri zaradi želje po pospešitvi dostave izdelkov, možnosti upravljanja sprememb in povečanja produktivnosti (slika 1). Zanimivo je, da stroški projekta in vzdrževanje programske opreme pri tem nimajo velikega vpliva [12, 16]. Iz navedenega lahko sklepamo, da je največja težava tradicionalnih pristopov ravno dolg čas razvoja in upravljanje sprememb. Ravno to pa je strankam najbolj pomembno [7]. Pri uporabi agilnih metod se zavedamo pomembnosti zadovoljne stranke, ki jo želimo kar se da aktivno vključiti že v razvojni proces. Stranka je posledično zadovoljna s hitro dostavo funkcionalnosti, razvojna ekipa pa je zadovoljna, ker se omeji na razvoj manjših delov produkta in od stranke hitro prejema povratne informacije (zahteve postanejo bolj jasne) [4].

Slika 1. Razlogi za vpeljavo agilnega pristopa

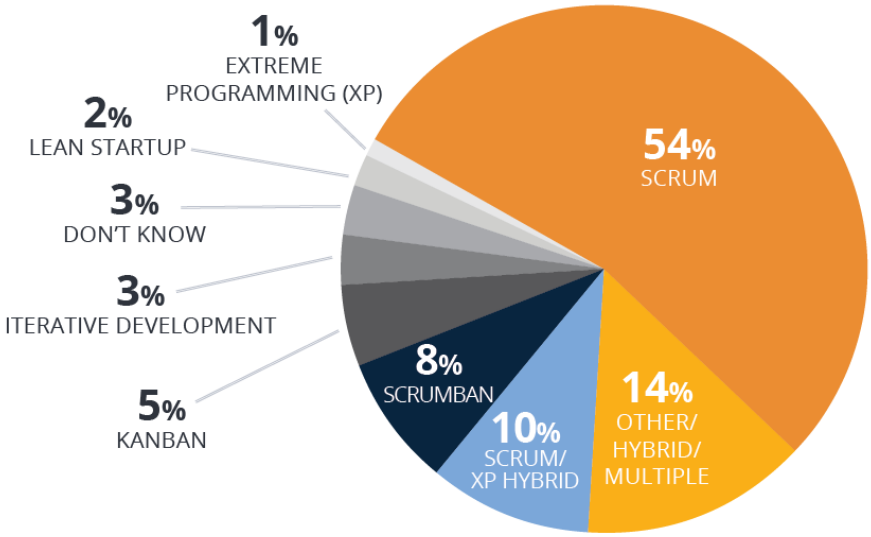
Zasnova agilne metode je pogosto izziv že sama po sebi. V projektih vpeljave agilne metode besedico »zasnova« pogosto izpustijo ali jo nadomestijo z »izbira«. V tem primeru v podjetju uvedejo že obstoječo metodo razvoja brez upoštevanja mnogih posebnosti podjetja in razvojnih ekip. Po nekaj pilotnih projektih sledijo pogosto večja prilagajanja izbrane metode, ki bi se jim bilo ravno z uporabo predhodne zasnove metode možno izogniti. Poudariti je treba, da kot zasnovo lahko razumemo tudi izbiro neke obstoječe metode, če izbira temelji na poglobljeni študiji apliciranja na obstoječe stanje. Zgolj izbira in izvajanje neke uveljavljene »of-the-shelf« agilne razvojne metode je mogoča samo v primerih, ko izvajalec izpolnjuje vse predpogoje, ki jih izbrana metoda pričakuje (npr. za Scrum je to majhna, multifunkcijska ekipa, ki dela na eni lokaciji in se sočasno ukvarja z enim projektom ipd.).

Zasnova (agilne) razvojne metode je izrazito inkrementalen proces. Če izhajamo iz pregleda najboljših praks v industriji ali iz svoje lastne, morda celo zastarele metode razvoja, bomo do pravega nabora praks prišli šele v nekaj iteracijah in ciljno usmerjenih sestankih, ki so namenjeni vrednotenju predlaganih pristopov in identifikaciji izboljšav. Primer aktivnosti, povezanih z zasnovo ustrezne agilne razvojne metode je sledeč:

- začetni sestanek vpletenih izvajalcev z namenom identifikacije pomanjkljivosti obstoječe razvojne metode in njihovih implikacij,
- študija potenciala, ki ga imajo agilni pristopi oz. obstoječe agilne metode za naslavljanje identificiranih pomanjkljivosti,
- pregled dobrih praks iz sorodne industrije pri naslavljanju identificiranih pomanjkljivosti,
- snovanje začetne različice nove / prilagojene razvojne metode,
- sestanek z namenom pregleda in kritične diskusije predlaganih sprememb ter identifikacija nadaljnjih izboljšav,
- iterativno izboljševanje predlagane agilne metode,
- odločitev za uvedbo po metodi postopnega uvajanja ali uvedbo »vse naenkrat«.

3. Pregled uveljavljenih industrijskih praks

Po najnovejših raziskavah se med najbolj uporabljene enoekipne agilne metode uvrščajo Scrum, Kanban in Scrumban. Raziskave kažejo, da približno polovica podjetij še vedno uporablja slapovni model, druga polovica pa uporablja agilne in iterativne pristope. Podjetja, ki uporabljajo agilne metode, se po podatkih raziskave VersionOne (slika 2) največkrat odločajo za Scrum oziroma Scrum + XP (78%), Scrumban (8%) in Kanban (5%).



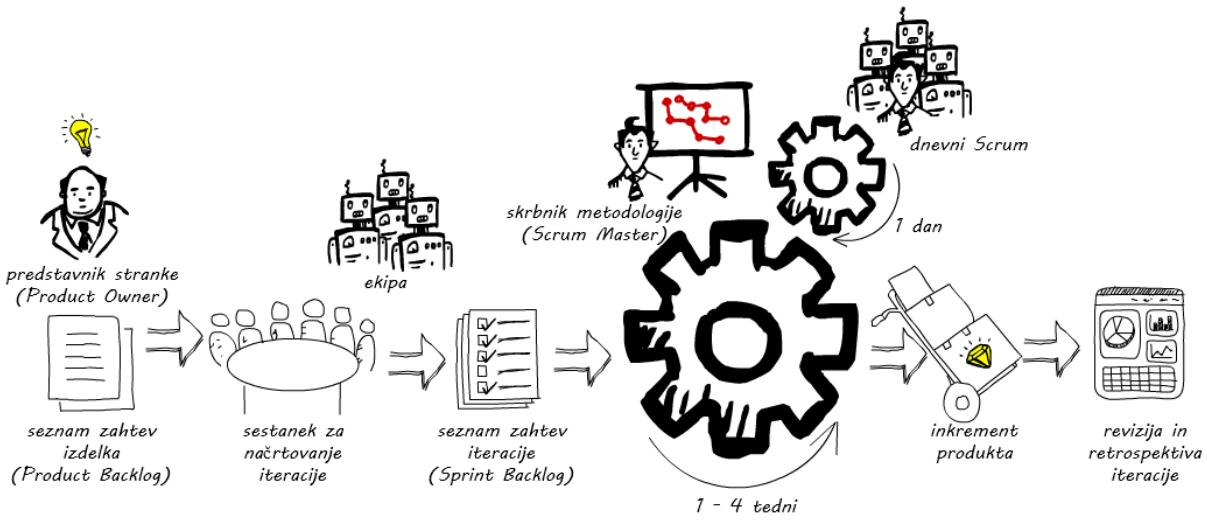
Slika 2. Tržni deleži enoekipnih metod agilnega razvoja [3]

Scrum

Scrum je organizacijsko ogrodje, sestavljeno iz ekipe, povezanih vlog, dogodkov in artefaktov, ki jih povezujejo pravila, s katerimi se urejajo odnosi med njimi. Metoda Scrum določa tri ključne vloge. Te so predstavnik stranke / skrbnik izdelka (angl. Product Owner), ekipa (angl. Scrum Team) in skrbnik metodologije (angl. Scrum Master). Predstavnik stranke je zadolžen za maksimizacijo donosnosti naložbe s pomočjo oblikovanja seznama zahtev in določitve njihovih prioritet. Zahteve podaja v posamezne Sprinte. Ne smemo ga razumeti kot produktnega managerja. Razvojna ekipa je zadolžena za razvoj funkcionalnosti produkta. Bistvenega pomena je večfunkcijskost, kar pomeni, da imajo člani ekipe vso potrebno znanje za implementacijo izdelka in samoorganiziranost. Skrbnik metodologije je zadolžen za nemoten potek projekta in pomoč razvojni ekipi pri pravilnem izvajanju pravil Scrum za

dosego poslovne vrednosti. Nekateri avtorji navajajo še pomožne vloge, kot so deležniki (stranke, prodajalci) in managerji.

Razvojni proces se začne z vizijo projekta, ki jo pripravi predstavnik stranke / skrbnik izdelka (slika 3). Zadolžen je za oblikovanje seznama zahtev (angl. Product Backlog), ki jih mora razvojna ekipa izpolniti za uspešno zaključen projekt. Zbrane zahteve prioritetizira in jih uvrsti v predvidene Sprinte (t.j. iteracije). Priporočeno trajanje posameznega Sprinta sta dva tedna (najmanj en, največ štirje tedni). Ko se na začetku projekta njegovo trajanje določi, ga več ne spreminjamo.

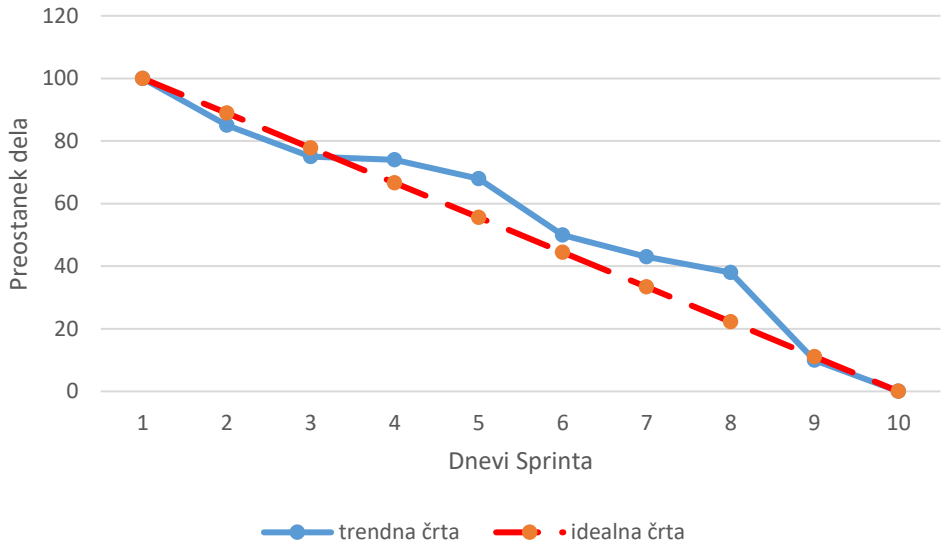


Slika 3. Metoda Scrum

Pred vsakim Sprintom se skupaj s predstavnikom stranke sestane razvojna ekipa na sestanku za načrtovanje Sprinta (angl. Sprint Planning Meeting). Skupaj pregledajo seznam zahtev in določijo, katere zgodbe bodo realizirane v prihodnjem Sprintu. Na podlagi izbranih zgodb razvojna ekipa sestavi seznam zahtev (angl. Sprint Backlog) in jih v primeru, če je katera zahteva prevelika, razdeli na več manjših. Ocenjevanje zahtevnosti posameznih zahtev se izvede s pomočjo uveljavljenih tehnik ocenjevanja, npr. igre planiranja (Poker Planning).

Po začetku Sprinta se razvojna ekipa vsak dan zbere na kratkem, 10-minutnem dnevnem Scrum sestanku (angl. Daily Scrum Meeting). Na tem sestanku si izmenjajo stanje napredka prejšnjega dne, se dogovorijo o delu v tekočem dnevu in izpostavijo morebitne težave. Po koncu vsakega Sprinta se izvede njegova retrospektiva (angl. Sprint Review Meeting), pri kateri sodeluje razvojna ekipa in predstavnik stranke. Pred vsakim začetkom novega Sprinta skrbnik metodologije skliče še sestanek za oceno kakovosti razvojnega procesa (angl. Sprint Retrospective Meeting). Na tem sestanku se izpostavijo možnosti izboljšav v samem razvojnem procesu, ki jih v naslednjih Sprintih skuša vpeljati razvojna ekipa.

Za merjenje uspešnosti ekipe je odgovoren skrbnik metodologije. Tipična metrika za uspešnost vodenja procesov je obseg preostalega dela na projektu. Ta metrika se navadno predstavi v obliki sprotne grafične predstavitve – grafa preostalega dela (angl. Burn Down Chart). Graf prikazuje, kako se v odvisnosti od časa spreminja obseg preostalega dela na projektu. Primer takšnega grafa je prikazan na sliki 4, ki ima na vertikali prikazan obseg preostalega dela, na horizontali pa čas (navadno v dnevih). Poleg omenjene metrike se uporabljajo še druge metrike in pristopi, kot sta diagram prislužene vrednosti in funkcijska razgradnja delovnih vlog.



Slika 4. Graf preostalega dela – sprotna predstavitev metrike

Uspešnost ekipe pri razvoju lahko merimo še z več metrikami [6]. Število dejansko zaključenih zgodb v primerjavi z načrtovanimi je metrika, ki govori o tem, ali razvojna ekipa sploh pozna in razume svoje sposobnosti. Primerja se število dejansko zaključenih zgodb s številom zgodb, ki si ga je ekipa zadala opraviti v enem Sprintu. Uspešnost Sprinta se lahko meri tudi s popolnoma tehničnimi metrikami, kot npr. merjenje tehničnega dolga, rast napak ipd.

Hitrost ekipe je izražena kot realizacija števila točk. V Scrumu se pričakuje, da ekipa pospešuje in s tem v vsaki iteraciji opravi večje število točk kot v predhodni iteraciji. Ob koncu vsakega Sprinta razvojna ekipa pokaže rezultate svojega dela stranki z namenom, da pridobi morebitne predloge za izboljšave in druge pripombe. Odziv stranke in/ali deležnikov lahko merimo na različne načine, kot je število na novo definiranih funkcionalnosti, ki morajo biti zajete v končnem produktu ipd.

Zavzetost ima ključno vlogo pri uspešnosti ekipe. Problem, ki se pojavi, je: “Kako jo naj merimo objektivno?”. Še vedno je najpreprostejši način ta, da skrbnik metodologije ekipi enostavno postavi vprašanje, ali so motivirani, veseli ipd. Možnost izboljšav lahko odkrijemo tudi med

izvajanjem retrospektive iteracije. Namen retrospektive je pregled izvajanja preteklega Sprinta, ki zajema ljudi, odnose, procese in orodja.

Kanban

Kanban je metoda upravljanja procesov, ki gradi na izkušnjah starejših agilnih metod. Pri Kanbanu sta glavna cilja odprava potrat (angl. waste) in zamikov, kar pozitivno vpliva na optimizacijo delovnega toka. Temelji na mehanizmu razporejanja opravil, imenovanem ravno-ob-pravem-času (angl. Just-In-Time – JIT), ki so ga razvili in uporabili pri Toyoti [12, 14].

Kanban temelji na treh osnovnih pravilih:

- **Vizualizacija poteka dela**

  Proces dela je po navadi predstavljen s tablo Kanban (slika 5), s katere je jasno razvidna pot od pričetka do zaključka opravila. Opravila so prioritizirana, kar pomeni, da so na tabli vedno samo najpomembnejša opravila. Prav tako je potrebno jasno definirati, kdaj so opravila zaključena. Časovno se opravila pomikajo med različnimi stanji, dokler ne pridejo v končno stanje. Pri tem je glavni cilj čim hitreje končati opravilo, kakor hitro se to pojavi na tabli. Glede na kompleksnost procesa definiramo primerno število stanj – v osnovi potrebujemo tri stanja (Todo, Doing, Done) – tem pa lahko poljubno dodajamo še lastna [14].

- **Omejitev WIP (Work In Progress)**

  Z omejitvijo Dela v teku se držimo načela “naredimo več tako, da delamo manj”, ki je bilo potrjeno že mnogokrat [12]. Ne glede na to, ali je projekt preprost ali zahteven, ali je ekipa velika ali majna, vedno obstaja optimalno število opravil v teku, da z njim ne žrtvujemo učinkovitosti razvijalcev. Sočasno izvajanje več opravil vodi do kasnejših časov zaključkov kot zaporedno izvajanje manjšega števila opravil. S pomočjo metode Kanban želimo omejiti število opravil v teku na optimalno raven. Ekipa si namreč prizadeva k čim hitrejšemu končanju posameznih opravil, odkar se ta pojavijo na tabli – ni pomembno, koliko opravil se izvaja sočasno [12, 14]. Zaradi krakih časov dostave novih funkcionalnosti je zaželena, skoraj potrebna, avtomatizacija dostave produktov. Pri tem si lahko pomagamo s tehnikami DevOps, ki združujejo vrzel med razvojem in operativo. Cilj DevOps je čim bolj zanesljiv, ponovljiv in prilagodljiv proces prehoda novih funkcionalnosti iz razvojnega v produkcijsko okolje [15].

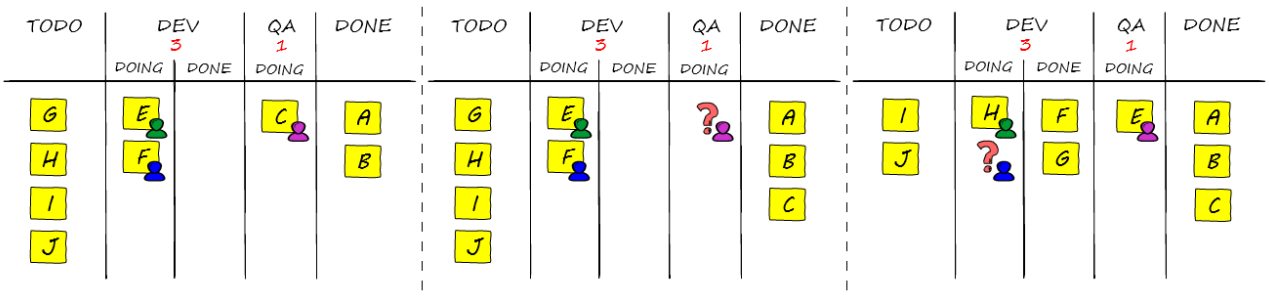
- **Minimalizem, meritve in izboljšave**

  Kanban ohranja obstoječe ekipe, procese, vloge in zadolžitve ter vpeljuje minimalne spremembe za njegovo adaptacijo. Ekipe lahko ohranijo obstoječe pristope, ki so že vpeljani in dobro delujejo, hkrati pa pridobijo na produktivnosti ter kontroli procesov s pomočjo dela po Kanbanu. Ta prav tako vzpodbuja vpeljavo metrik in opravljanje objektivnih meritev, s katerimi se spremlja napredek, izboljšuje delovni tok in veča učinkovitost [12, 14].

Kanban ne predvideva posebnih vlog za upravljanje metode. Prav tako za začetek ne potrebujemo daljših priprav. Potrebujemo zgolj tablo s stanji, ki vizualizirajo proces gradnje izdelka. Za posamezna stanja določimo omejitve WIP in dodamo kartice z opravili. Napredovanje po stanjih procesa deluje po principu povleči (angl. pull), kar pomeni, da lahko z delom na opravilu začnemo šele takrat, ko imamo na stanju prosto mesto, kar določa omejitev WIP. Vsakokrat, ko se opravilo pomakne v stanje Končano, se proži signal Povleči. To pomeni, da je ekipa/posameznik uspešno zaključil/a opravilo, zato se je sprostilo novo mesto za stanje Delo v teku. Vsa prejšnja stanja, ki so morebiti bila blokirana zaradi omejitev v kasnejših stanjih, lahko ponovno nadaljujejo z delom na novih opravilih. Člani ekip, ki nimajo opravil v delu, lahko izvlečejo novo opravilo.

Nasprotje temu principu predstavlja princip potisni (angl. push), kjer se omejitev WIP ne upošteva. Takoj, ko je opravilo na določenem stanju končano, se potisne na naslednjo stanje. S potiski lahko hitro pridemo do zasičenja na stanjih, ki v procesu predstavljajo ozko grlo. Zasičenje ne bo neposredno vplivalo na ostale člane, saj lahko ti prosto nadaljujejo z novim delom, bo pa privedlo do preobremenitev članov na kritičnih stanjih, saj bodo občutili vedno večji pritisk, medtem ko bodo preostali člani normalno nadaljevali z delom.

Pri principu Povleči lahko zaradi obtičanja dela na enem stanju zastane delo celotne ekipe. Ekipa je zato prisiljena hitro prepoznati ozka grla, se prilagajati, si medsebojno pomagati in tesneje sodelovati [12, 14].



Slika 5. Primer table Kanban

Na levi tabli slike 5 lahko vidimo primer table Kanban, ki je razdeljena na štiri stanja: Todo, Dev (Development), QA (Testing) in Done. Stanje Development je dodatno razdeljeno na dve stanji (Doing in Done), tako, da lahko člani v ekipi za testiranje prepoznajo končana opravila, ki so pripravljena za povleko.

Na srednji in desni tabli slike 5 je prikazana situacija stagniranja (angl. Slack Time), kjer član ekipe ne more povleči novih opravil zaradi pomankanja opravil na prejšnjem ali zasičenja opravil na kasnejšem stanju. Omejitev WIP prepoveduje povleko novega opravila in vzpodbuja dokončanje dela, ki je že v procesu (ima zaradi tega višjo prioriteto). V takšnih primerih je zaželeno, da člani ekipe pomagajo na stanjih, kjer se je pojavilo ozko grlo, lahko pa se posvetijo tudi pisanju avtomatiziranih testov, preoblikovanju programske kode (angl. Refactoring), inoviranju ali drugim doprinosom k projektu [8].

Uspešnost vodenja procesov z metodo Kanban lahko merimo z različnimi metrikami, med katerimi je najbolj uveljavljena čas dobave. Ta metrika meri čas od nastanka zahtevka po spremembi (stranka želi nov inkrement produkta) do njegove realizacije. Pogosto se uporablja tudi metrika čas cikla, ki meri čas med pojavitvijo opravila na tabli Kanban in končanjem opravila – je čas, ki ga ekipa dejansko porabi za izvedbo [12, 14].

Scrumban

Scrumban je kompozit metod Scrum in Kanban, saj vsebuje osnovne lastnosti Scruma in fleksibilnost Kanbana. Planiranje je izvedeno v smislu zapolnitve prostega mesta in dodelitve opravil po principu Povleči, kot pri Kanbanu [9]. Tabla tako ostaja enaka kot pri Kanbanu, z omejitvami WIP na posameznih stanjih. Zaželeno je, da se opravila na tabli prioritetizirajo v času planiranja v velikosti t.i. vedra. To se ponavadi zgodi takrat, ko ekipi zmanjka opravil v seznamu (angl. Planning On Demand). Tako kot pri Kanbanu tudi tu ostaja ocenjevanje dolžine opravil opcijsko. Ekipe, katerim je pomembno napovedovanje trajanja iteracije, se raje odločajo za določanje opravil na način, da so vsa zelo podobnih zahtevnosti (zato bodo trajala enako dolgo).

Scrumban podpira večfunkcijske in specializirane (ekspertne) ekipe, nima pa jasno definiranih vlog za njihove člane [10, 11]. Kontrola dela procesa (stanje) znotraj Scrumban lahko pripada

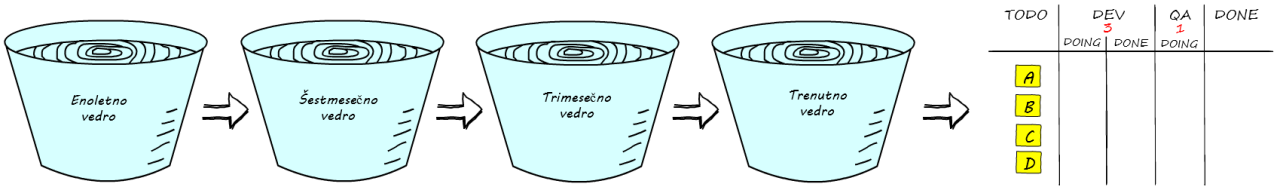
različnim ekipam, kot so npr. razvojna ekipa, ekipa za zagotavljanje kakovosti, DevOps in druge.

Tako kot pri Kanbanu so sestanki in drugi dogodki opcijski. Najbolj pogosto se uporabljajo kratki, petnajstminutni dnevni sestanki t.i. Standup in Kaizen. Kaizen je dogodek, namenjen skupnemu reševanju problemov in konstantnemu izboljševanju, tako da ne ponavljamo istih napak znova in znova. Njihov cilj je hitro odkriti in odpraviti (t.i. angl. Breakthrough) jedro problema, pri čemer sodelujejo člani iz različnih ekip oziroma oddelkov.

Planiranje v velikosti vedra je del agilnega projektnega vodenja in je grafično predstavljeno na sliki 6 [5]. Gre za sistem, pri katerem razvojna ekipa premika potencialne projekte skozi različna vedra. Prvo vedro se imenuje “Enoletno vedro”, ki vsebuje ideje, cilje in velike načrte potencialnih projektov. Drugo vedro, t.i. “Šestmesečno vedro”, vsebuje projekt, ki ga želi podjetje izpeljati. Ideje so definirane bolj specifično, vendar še vseeno ne vsebujejo vseh podrobnosti. V navezavi s Scrumom bi bilo to najbolj primerljivo z logično povezanim naborom opravil (angl. Sprint epics). Ko se odločimo, da bomo nadaljevali z razvojem določenega logično povezanega nabora opravil, ga premaknemo v t.i. “Trimesečno vedro”. V tem vedru se nahajajo podrobneje definirana opravila v obliki uporabniških zgodb. Nazadnje se opravila premaknejo v t.i. “Trenutno vedro” in razčlenijo na ločeno definirana opravila. Ta opravila si lahko predstavljamo kot opravila pri Scrumu. Ko je razvojna ekipa pripravljena, se omenjena opravila prestavijo v projektni seznam zahtev.

Iteracije pri Scrumbanu naj ne bi presegale dveh tednov. Ko se projekt približuje koncu, nastopi faza zamrznitve zahtev (angl. Feature Freeze). V njej je prepovedano dodajanje novih opravil, saj je prvo potrebno dokončati vsa nedokončana. Tik pred izidom produkta nastopi faza triaže (angl. Triage), v kateri se projektni manager odloči, katera opravila v fazi razvoja je potrebno zaključiti in katerih ne.

Najbolj uporabljena metrika pri Scrumbunu je povprečen čas cikla.



Slika 6. Planiranje v velikosti vedra

Primerjava vodilnih enoekipnih metod agilnega razvoja

Tabela 1: Primerjava metod Scrum, Kanban in Scrumban

	SCRUM	**KANBAN**	**SCRUMBAN**
Tabla/Artifakti	Preprosta tabla Product backlog Sprint backlog Izdelek Burndown chart	Preprosta tabla	Preprosta tabla
Sestanki	Daily scrum Sprint planning Sprint review Sprint retrospective	Ni zahtev	Daily scrum Ostali Scrum sestanki, če so potrebni
Prioritete	Del Product Backloga, določa predstavnik stranke	Ni zahtev	Ni zahtev (vedra!)
Kdo daje delo?	Predstavnik stranke / skrbnik izdelka	Odvisno od vlog na projektu	Odvisno od vlog na projektu
Iteracije	Da - sprinti.	Ne - kontinuirano delo	Ni potrebno Lahko sprinti
Ocenjevanje dela	Da (točke)	Ne - pričakujemo naloge podobnega obsega	Ne - pričakujemo naloge podobnega obsega
Ekipe	Večfunkcijske	Večfunkcijske ali specializirane	Večfunkcijske ali specializirane
Vloge	Predstavnik stranke Skrbnik metodologije, ekipa	Ni določeno	Ekipa + dodatne vloge po potrebi

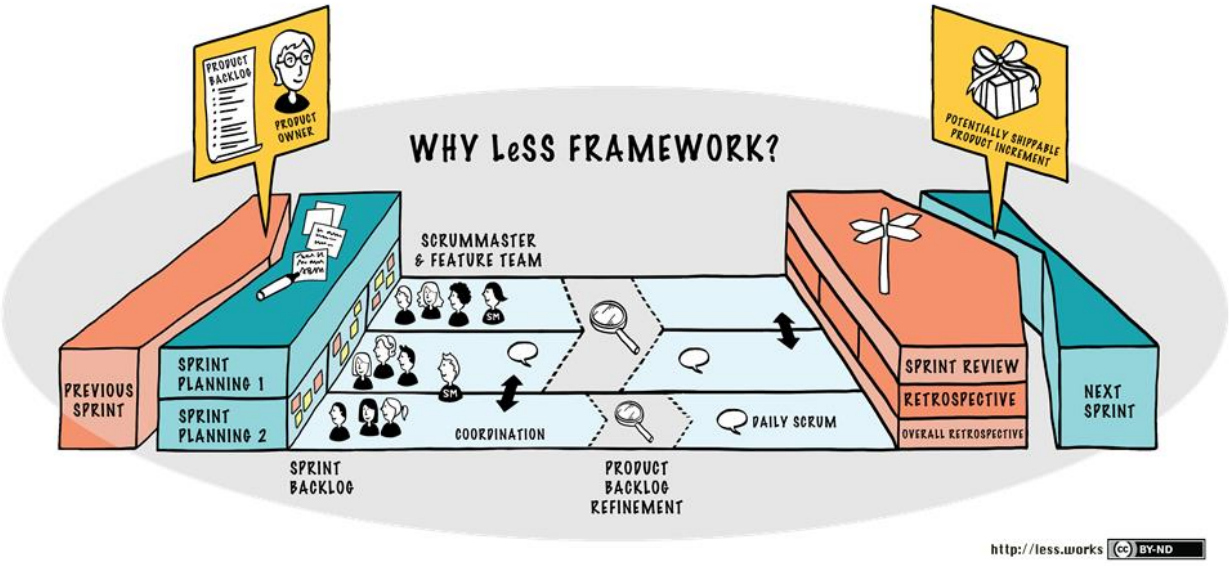
	SCRUM	**KANBAN**	**SCRUMBAN**
Skupinsko delo	Sodelovanje	Temelji na »Pull« principu	Temelji na »Pull« principu
WIP (omejitev trenutnega dela)	Planirano na nivoju sprinta	Sprotno nadzorovanje s stanjem na tabli	Sprotno nadzorovanje s stanjem na tabli
Spremembe v naboru opravil	Morajo počakati na naslednji sprint	Spremenjeno po potrebi (JIT)	Spremenjeno po potrebi (JIT)
Seznam zahtev	Uporabniške zgodbe s prioritetami	Ne obstaja (JIT)	Ne obstaja (JIT)
Odzivanje na ovire	Naslovljene takoj	Naslovljene takoj	Naslovljene takoj
Kdaj ustreza?	Razvoj izdelka Majhne nadgradnje Specifikacije zahtev so dobre	Podpora in vzdrževanje (operativa)	Razvoj izdelka, če so nejasne zahteve Spremenljive spremembe (a smer je jasna) Dogodkovno naravnana podpora in vzdrževanje
Grafi	Graf preostalega dela (Burndown chart)	Cumulative flow diagram	Kombinacija

LeSS in SAFe

V podjetjih je praktično že postala stalnica, da en produkt razvija večje število razvojnih skupin. Vodenje takšnega sistema se navadno lotimo z uporabo pristopa LeSS (Large-Scale Scrum). LeSS ni nov in izboljšan Scrum. Gre za ugotavljanje, kako vpeljati načela, namen, elemente in eleganco Scruma v obsežnejšem kontekstu, kar se da preprosto. V njem najdemo:

- en Product Backlog (za produkt in ne za ekipo),
- en spisek kriterijev, ki morajo biti izpolnjeni pred naslednjih inkrementov produkta, za vse ekipe,
- en inkrement produkta ob koncu vsakega Sprinta,
- en Product Owner,
- več več-funkcijskih ekip (brez ekip z zgolj specifičnim znanjem),
- en Sprint.

Ogrodje LeSS prikazuje slika 7. Ekipa je ista kot pri enoekipnem Scrumu, vendar vse ekipe same koordinirajo in integrirajo delo s preostalimi ekipami. Če pri razvoju sodeluje več kot osem ekip, govorimo o LeSS Huge.

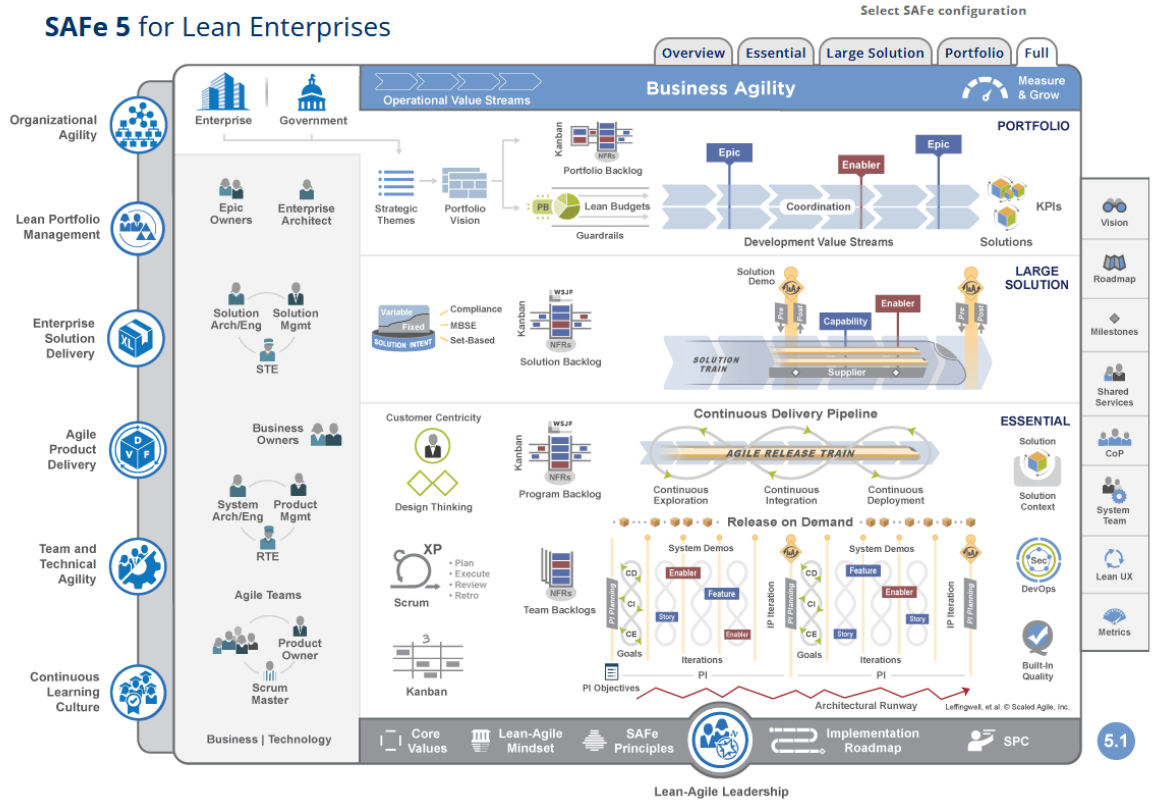


Slika 7. Ogrodje LeSS

Pristop LeSS so uspešno vpeljali v mnogih velikih podjetjih, kot na primer Ericsson & Nokia Networks, Cisco in UBS. Ena izmed zadnjih najbolj odmevnih zgodb pa je vpeljava LeSS v

podjetje BMW, kjer so z omenjeno metodo rešili izziv vzpostavitve USP (Unified Sales Platform), ki je vključevala integracijo več kot 30 zunanjih sistemskih storitev. Rezultat vpeljave je bil v dogovorjenem roku dostavljen UPS, visoko zadovoljstvo strank in visoka kakovost končne storitve.

SAFe (angl. Scaled Agile Framework) je pristop, v katerem so integrirana načela, prakse in kompetence, ki so kombinacija agilnih pristopov, vitkega razvoja in DevOps. Uspešno vpeljavo SAFe lahko predstavlja študija primera Air France - KLM. Slednji so želeli izboljšati čas dostave izdelkov na tržišče in učinkovitost razvoja poslovnih aplikacij, pri vsem tem pa so bili primorani upoštevati stroge regulative. Vpeljali so tri ART-e v komercialno-digitalni poslovni domeni. ART (Agile Release Train) je organizacijski konstrukt agilnih ekip, ki praviloma vključuje 50 - 125 ljudi. Za namen vodenja vpeljave agilnih pristopov v 11 poslovnih domenah so oblikovali Agile Release Plan, prilagojen za letalsko domeno, temelječ na SAFe. Oblikovali so tudi Transversal Tracks – skupine povezane z določeno poslovno domeno. Po končani vpeljavi so poročali o 20 % povečanju učinkovitosti dostave izdelkov.



Slika 8. Ogrodje SAFe

Ogrodje SAFe (slika 8) je možno vpeljati v štirih različicah (osnovni, celotni, ter dvema vmesnima: SAFe za velike rešitve in SAFe portfolio). Ogrodje je razdeljeno na več nivojev, pri čemer se najnižji posveča tehnično-organizacijskim vidikom razvoja, podobno kot to počne LeSS. Vmesni nivoji pa preko ekipne organizacije pripeljejo do portfolia produktov in strateških odločitev podjetja. SAFe se tipično uporablja za razvojne projekte z več kot 125 udeleženci.

4. Primer: zasnova in uvedba agilnih praks v proces razvoja in vzdrževanja IT rešitev

Izbor oz. snovanje ustrezne metode agilnega razvoja je pogojen z mnogimi spremenljivkami razvojnega okolja. V praksi bomo le redko srečali stabilne razvojne ekipe, ki delujejo ves čas skupaj in na podobnih rešitvah. Večina izzivov med snovanjem in uvajanjem agilnih metod dela je povezanih ravno s tem, kako v takšnih okoljih kompenzirati realno stanje in vsaj delno poustvariti okolje, primerno za razvoj na agilen način. Tukaj so še drugi viri situacij, ki večino agilnih metod (če bi jih izvajali popolnoma po črki pravil) postavijo pod vprašaj. Vključno z najbolj osnovnimi, kot so bolniške in druge neplanirane odsotnosti kadra. Zelo pogosto se srečamo s potrebami po zelo specializiranih znanjih znotraj več ekip. Tu so tudi kritične napake, ki napredujejo do strank in vsekakor zahtevajo od razvijalcev takojšnje ukrepanje – ne glede na stanje trenutne iteracije. Prav tako bomo (razen v primeru majhnih, mikro ali zagonskih podjetij) zelo redko srečali člane razvojnih ekip, ki ne bi bili hkrati zadolženi za več rešitev. Prepletanja med nalogami različnih rešitev se sicer tipično ne dogajajo na dnevni ravni, a na nivoju tedna je zelo pogosto velik izziv, da bi razvijalci (sploh v primeru specializiranih znanj) bili ves čas na voljo le za točno določen projekt. Prav tako razvijalci ne zaključijo dela na rešitvi ob predaji stranki – tukaj so še vzdrževalna dela, ki prav tako terjajo določen angažma razvijalcev.

Agilne metode tipično naslavljajo »le« razvojni del projekta. Zelo redko v vodilnih agilnih metodah srečamo pristope za enostavno integracijo drugih aktivnosti projektov razvoja informacijskih rešitev in storitev – pa naj bo to sprotno snovanje zahtev, določanje prioritet, preliminarno ocenjevanje projektov, dokumentiranje (končne ali celo vmesne različice) rešitve, podpora operativnim ekipam pri nameščanju in vzdrževanju rešitev ipd.

Če bi slepo sledili denimo metodi Scrum, ki od razvojnih ekip pričakuje ne samo nenehno »šprintanje«, temveč celo nenehno pospeševanje, nam takšna agilna utegne predstavljati prej oviro kot dodano vrednost.

Pogosto ločena ekipa za zagotavljanje kakovosti skrbi za testiranje večjega števila rešitev, ki prihajajo iz različnih razvojnih ekip. Je v tem primeru sploh možna uveljavitev načela odprte neposredne komunikacije? Tudi če jo dopustimo, bo predstavljala nenehne motnje razvojni ekipi. V praksi se uveljavljajo t.i. nerazvojne iteracije, namenjene stabilizaciji kompleksnih rešitev, skupnim integracijskim naporom.

Našteti so le nekateri izzivi pri snovanju agilne metode, ki lahko tekom uvajanja dobijo popolnoma novo dimenzijo skozi vidik uveljavljanja pristopov v sicer utečene razvojne ekipe. Tovrstni izzivi so povezani na eni strani s povsem tehničnimi elementi metode, kot so izvedba

vodenja dela (npr. fizična tabla z opravili), na drugi strani pa z dojemanjem opravljenega dela (npr. samoocenjevanje opravil, osredotočenost na funkcionalnost po vertikali tehnologij namesto osredotočenost na tehnične elemente rešitve ipd.).

Izhodišča študije primera za snovanje lastne metode

Denimo, da v podjetju Podjetje d.o.o. obstaja želja in namera, da bi v razvojne cikle rešitev informacijskih tehnologij (v nadaljevanju ITZ rešitev), sploh v razvoj / vzdrževanje / nadgrajevanje lastne rešitve ERP, vpeljali iterativne oz. agilne metode dela. Na takšen način bi izvedli postopen prehod iz obstoječih metod dela. Izhodiščna želja je povečati usklajenost strategije podjetja z IT rešitvami, ki le-to udejanjajo, ter povečati učinkovitost ter odzivnost ekip, ki izvajajo razvoj IT rešitev. Obstoječi procesi razvoja Podjetja d.o.o. izhajajo iz plansko usmerjenega, slapovnega (»waterfall«) modela razvoja. Interne stranke podajajo zahteve v pisni obliki in jih predajo v razvoj, nato po določenem časovnem obdobju sledi predaja rešitve v uporabo. V podjetju se kažejo mnoge slabosti sekvenčne narave trenutnega razvoja IT rešitev. Prenova procesov razvoja IT rešitev v podjetju bi torej bila osnovana na lastnih in tujih pozitivnih in negativnih izkušnjah tako tradicionalnih, sekvenčnih, kot tudi sodobnejših, iterativnih in agilnih, metod razvoja.

Indikatorji iz industrije, ki jih skrčeno povzemamo tudi v tem gradivu, kažejo mnoge pozitivne učinke uvedbe agilnih metod razvoja. Ob prenovi procesov razvoja Podjetja d.o.o. bi torej bilo smiselno izhajati iz trenutnih trendov agilnih metod ob upoštevanju uveljavljenih dobrih praks.

Zaradi specifične narave dela in tipične matrične organiziranosti Podjetja d.o.o. bodo prilagoditve izbrane metode nujne. Velik izziv razvojnim skupinam predstavlja razdrobljenost delovnih nalog ter slaba definiranost odgovornosti vlog, ki nastopajo znotraj procesa razvoja IT rešitev. Skupine se v kadrovskem smislu spreminjajo tekom razvojnega cikla. Posamezniki v istem trenutku tipično sodelujejo v večjem številu razvojnih ciklov, tudi ne nujno od začetka do konca, temveč tipično le za čas, ko se na projektu potrebujejo njihova specialistična znanja. Vzporedno z razvojem se posamezniki ukvarjajo tudi z vzdrževalnimi deli na rešitvah v produkciji, kar jim vzame veliko časa. Zaradi visokih odvisnosti med produkti so tudi postopki preverjanja in testiranja relativno nedorečeni. Posledično se težave IT rešitev tipično pokažejo šele ob uporabi rešitve, t.j. tipično mesece po snovanju specifikacij.

Primer povzema pristope, s katerimi bi lahko uspešno spremenili obstoječe procese Podjetja d.o.o., da bodo le-ti omogočali čim boljše naslavljanje navedenih obstoječih in novih, prihajajočih, izzivov.

Glavno izhodišče snovanja predlogov sta iterativnost in agilnost. Obstoječi (pretežno) sekvenčni način razvoja bi zamenjal / dopolnil iterativni način razvoja. Hkrati bi formalizirali tudi vidike agilnega razvoja, kot so hitre izdaje, tesno sodelovanje med produktnimi vodji, razvojnimi in testnimi skupinami, zbiranje in odzivanje na mnenja in zahteve uporabnikov ipd.

Pri tem želimo prilagoditi le organizacijo aktivnosti, njihove odvisnosti in mejnike. Nabor aktivnosti, način dokumentiranja ter ostale inženirske discipline želimo spremeniti kakor malo je le mogoče. Zmanjšati želimo čas, ki ga zaposleni preživijo na sestankih, hkrati želimo dvigniti kakovost krajših in bolj usmerjenih sestankov. Povečati želimo tudi vključenost in informiranost razvijalcev o poteku projektov.

Izhajamo iz prepričanja, da bi v primerih, kjer se je izkazalo za dobro, ohranili delno sekvenčni model. In sicer ob predpostavki, da se formalizira neformalne prakse, ki obstoječi sekvenčni proces izboljšajo.

Prav tako bi namesto transformacije uveljavljene matrične organiziranosti podjetja in nazivov ter vlog raje le-te ustrezno organizirali v smislu iterativnega in agilnega procesa. V ta namen prilagamo tudi nabor pričakovanih vlog in ekspertnih znanj, ki jim je v podjetju možno poiskati obstoječo enakovredno vlogo. Najbrž pa se ob uspešnem prehodu ne bo možno izogniti delni spremembi razmerja moči posameznih vlog na projektih.

Zaradi velikega števila razvojnih ciklov in relativno majhnega števila razvijalcev izhajamo iz predpostavke, da ekskluzivnih skupin na projekt ne bo (primer, ko bi en zaposlen oz. ena skupina določen čas delala le na enem projektu). Posledično ohranjamo delno matrično organiziranost. Hkrati izhajamo iz potrebe po maksimiranju časa, ko razvijalci usmerjeno delajo le na enem projektu brez vmesnih prekinitev. V ta proces želimo na dnevni bazi vključiti tudi možnost hitrega odzivanja na kritične napake in vzdrževalne zahtevke iz terena.

Kot osnovo predlaganega procesa (procesov) smo izbrali metode Scrum, Kanban in Scrumban, saj so le-te najširše zastopane v industriji . Zanje prav tako obstaja največ smernic, dobrih in slabih praks. V smislu koordiniranja ekip pri skupnem razvoju istega izdelka uvajamo pristope, ki jih najdemo tudi v metodah LeSS in SaFE.

Uvedba iterativnosti v nove procese Podjetja d.o.o.

Iterativnost, kot jo definiramo v tem primeru, razumemo na dva načina:

- neprestano prepletanje vsebinskih, razvojnih, testnih aktivnosti in izdaj (mnogo mikro slapovnih ciklov namesto enega samega) ter
- vzpostavitev in sledenja rutin v smislu dnevnih, (dvo)tedenskih in dolgoročnih opravil, sestankov ipd.

V splošnem predlagamo iterativnost na naslednjih nivojih:

- **Dolgoročno planiranje** (Izven strogo razvojnega cikla, npr. 3 meseci) - tekom dolgoročnega planiranja se doseže konsenz med predstavniki uporabnikov, katere funkcionalnosti / izdelki bodo šli v kakšnem sosledju v razvoj. V ta namen uporabljamo prioretizacijo dolgoročnih ciljev ter selekcijo kratkoročnih ciljev po principu izločanja dolgoročnih (“bucket-size-planning”).
- **Sprint** (iteracija, 2-tedenski mikrocikel) - sprint zajema delo na podmnožici funkcionalnosti - sledenje prioritetam. Pred sprintom je potrebno v okviru razvojne skupine doseči konsenz o količini in naboru funkcionalnosti - planiranje sprinta. Sprint pa naj da zaključen inkrement v smislu novih funkcionalnosti, ki jih naj potrdi skrbnik izdelka / predstavnik uporabnikov. Sprint se začne s planiranjem sprinta ter konča z demonstracijo novosti. Pomembno je, da so sprinti vseh ekip v podjetju sinhronizirani, saj bo le tako mogoče prosto prehajanje posameznikov med ekipami in/ali njihova sinhronizacija pri skupnih razvojih.
- **Dnevna rutina** - tekom sprinta je jasno, katera opravila morajo biti opravljena. Dnevna rutina je namenjena učinkovitemu opravljanju konkretnih opravil in dobri povezavi s predstavnikom uporabnikov. Dnevna rutina se prične s dnevnim sestankom in konča ob vnaprej določeni uri v dnevu.

Pred začetkom razvoja (sprinta) mora biti definirana vsebina naslednjega sprinta in ne več vsebina celotnega izdelka. Predstavniki uporabnikov med sprintom spremljajo razvoj, potrjujejo oz. zavračajo implementacijo, ter pripravljajo podrobno dokumentacijo za naslednji sprint. Aktivnosti vodenja kakovosti (testiranja) potekajo že kmalu po začetku razvoja. Izdaje rešitve naj bodo konstantne, vendar vsaj na koncu vsakega sprinta. Na nivoju projekta se lahko odločimo, za kakšen nivo izdaj gre: interne, eksterne, za uporabnike ipd.

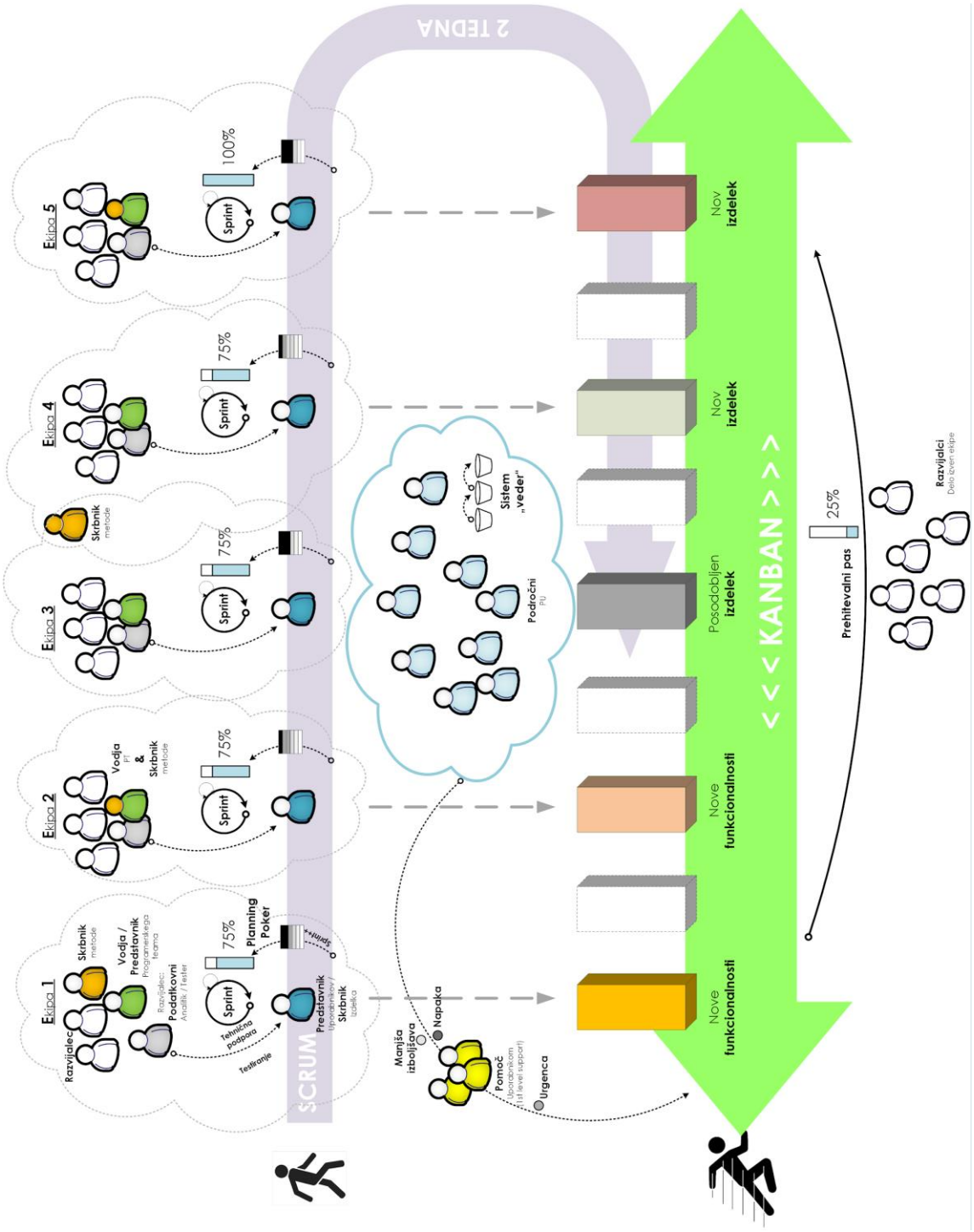
Ločeno in vzporedno pa poteka proces vzdrževanja ostalih rešitev in odprave napak. Razmerje med razvojnimi in vzdrževalnimi aktivnostmi mora zaradi možnosti planiranja biti znano pred

vsakim sprintom. Slika 9 prikazuje celoten pogled na metodo, ki poleg iterativnih konceptov vsebuje tudi sekvenčne - organizirane v smislu metode kanban. Kanban ("prehitevalni pas") je namenjen opravilom, ki so nujni in neplanirani (odprava napak, vzdrževalna dela, odprava urgenc ipd.) ter na njem sodelujejo vsi razvijalci, ki so sicer (ko poteka sprint) člani zaključenih ekip. Znotraj prehitevalnega pasa, ob času, ki je določen v skladu z dogovorjenim upravljanem prekinitev, posamezniki opravljajo dela na svojih preteklih izdelkih, ki so že v produkciji in niso predmet trenutnega dela. Predlagano razmerje delovnega časa razvijalcev za novosti napram vzdrževanju je 75% proti 25%, v korist novostim. Razmerje je mogoče za krajši ali daljši čas tudi prosto spreminjati - tudi ločeno po ekipah.

Ločitev iterativnega procesa razvoja novih funkcionalnosti od kontinuiranega vzdrževalnega procesa prinaša mnoge pozitivne učinke podjetju d.o.o.:

- Formalnih mejnikov je manj. Analiza, načrtovanje in implementacija se smiselno ciklično dopolnjujejo.
- Aktivnosti funkcionalnega testiranja se testerji lahko lotijo že zelo kmalu po razvoju, nakar sledijo sistemski testi, odprava napak pa je konstantna.
- Posledično bo predvsem v sklopu aktivnosti testiranja vzpostavljeno vedno več avtomatiziranih postopkov - kar je nujno, če želimo kakovostne konstantne in pogoste izdaje.
- Za zagotavljanje dnevne rutine ob omejitvah vključenosti posameznikov v večje število projektov istočasno je potrebna vloga vodje skupine.
- Planiranje projekta se bistveno poenostavi: pred začetkom razvojnih aktivnosti je potrebno določiti le dolgoročno smer rešitve ter v podrobnosti definirati prvi sprint. Zaradi dnevne rutine (stranka ob strani) je tudi manj možnosti šuma v komunikaciji med razvijalci in predstavniki uporabnikov.
- Predstavniki uporabnikov na nivoju sprinta sprejemajo ali zavračajo nove funkcionalnosti. Tako se ne more zgoditi, da bi končna rešitev vsebovala (vsebinsko) napačno rešitev. Predstavniki uporabnikov tako prevzamejo odgovornost za vsebinsko neoporečnost rešitve, razvijalci pa za tehnično odličnost.
- Ukinitev oz. vsaj zmanjšanje prekinitev ki jih med delom občutijo razvijalci s strani odprave nujnih napak oz. nujnih vzdrževalnih del: Ta izziv bi bilo možno nasloviti (kot opišemo v nadaljevanju) z uvedbo fiksnega dela dneva, ko se razvijalci usmerjeno ukvarjajo z razvojem brez vsakih motenj in uvedbo fiksnega dela dneva, ki ga imajo poleg razvoja razvijalci rezerviranega za neplanirane sprotne naloge (odprava napak, vzdrževanje ipd.). V smislu avtomatizacije bi tudi tukaj pomagalo orodje, ki bi prijavljene nejasnosti in napake peljalo skozi celoten postopek (prijavljena, prevzeta, odpravljena napaka ipd.) in na takšen način predstavljalo formalen komunikacijski

kanal med testnimi skupinami in tehnično pomočjo na eni strani ter razvijalci na drugi strani. Podpora orodja bi posledično omogočala tudi številne metrike, ki bi dolgoročno gotovo izboljšale kakovost in s tem zmanjšale potrebo po prekinitvah (npr. če bi se izkazalo, da večina težav podporne službe izhaja iz navodil in ne napak v tehničnem smislu, bi lahko več napora vložili v izboljšanje navodil).



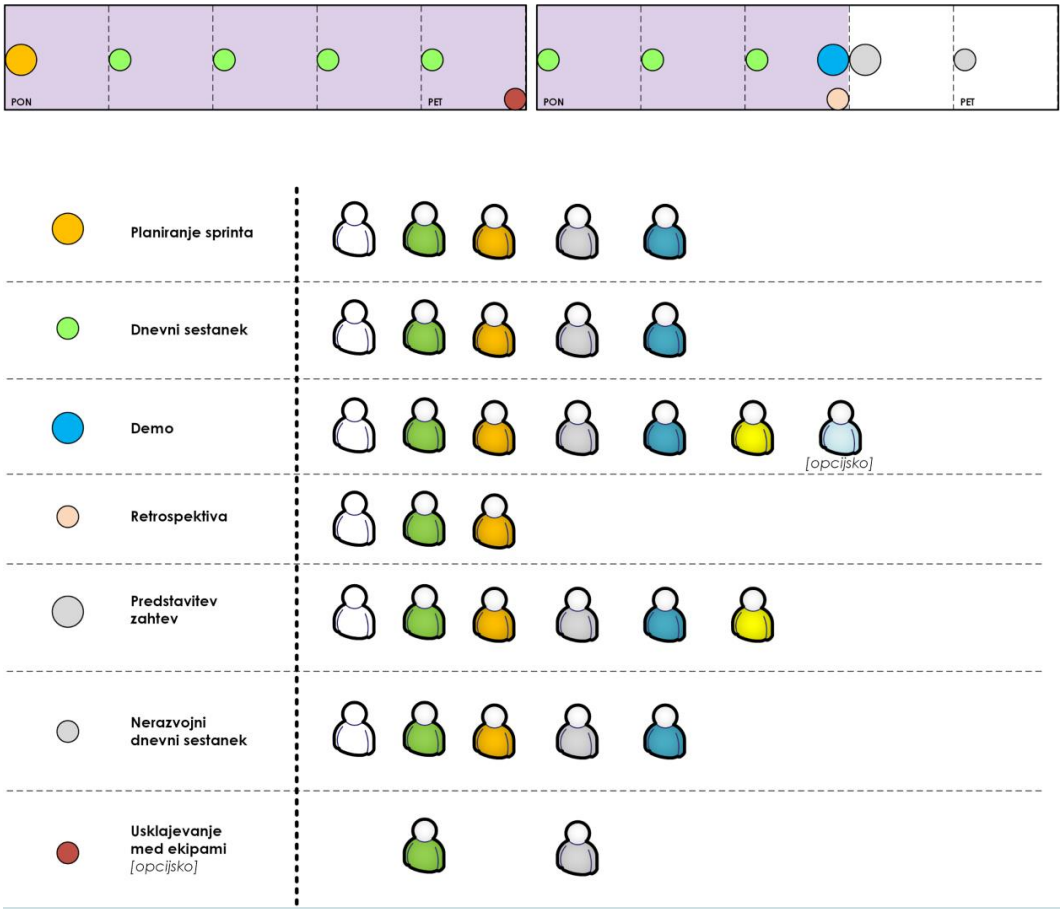
Slika 9. Prikaz celotnega pogleda na predlagano metodo

Cikel razvoja / nadgradnje novih funkcionalnosti

Razvojni cikel / iteracijo po vzoru metode Scrum imenujemo kar “sprint”. Tudi sam potek iteracije je podoben temu, kar predvideva Scrum. Sprint naj bo dolg 2 tedna, njegov začetek in konec naj bosta sinhronizirana na nivoju vseh ekip. Sprint je namenjen razvoju novih funkcionalnosti oz. večji nadgradnji obstoječih (ko je obseg nalog ocenjen na vsaj 2 tedna za celotno ekipo IT razvijalcev).

Sprint se začne s sestankom planiranja sprinta. Na dnevni bazi se srečamo z dnevnimi sestanki, zaključek sprinta pa ni konec zadnjega tedna, kot je to praksa pri metodi Scrum, temveč se sprint zaključi 1 ali 2 dni pred tem. Na takšen način dobijo ekipa ter predstavnik uporabnikov možnost, da v celoti in v detajle proučijo specifikacije zahtev za naslednji sprint. V kolikor se predvidena 2 dneva za študijo zahtev izkažeta kot predolga, je možno čas, namenjen študiji zahtev, skrajšati tudi samo na 1 dan. Od razvijalcev se tekom študije zahtev pričakuje, da se dogovorijo tudi o tehničnih in arhitekturnih rešitvah, ki so potrebne za naslovitev zahtev naslednjega sprinta. Čas za študijo zahtev bo tudi neprimerno skrajšal planiranje sprinta, ki je predvidevamo za začetek, prvi dan naslednjega sprinta.

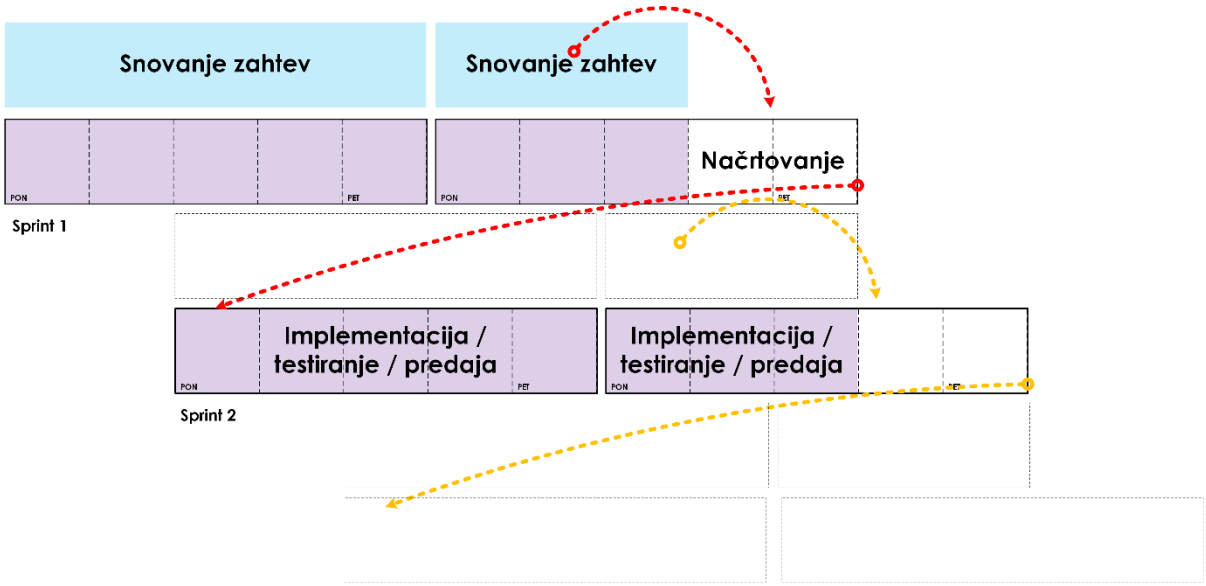
Retrospektiva je po vzoru Scruma kratek sestanek, namenjen ekipi. Novost lastne metode pa je sestanek za usklajevanje med ekipami. Ta sestanek je pomemben v primeru, ko dve ali več ekip zaradi velikega obsega hkrati dela na zahtevah enega predstavnika stranke. Na sliki 10 je poleg opisane strukture sprinta prikazana tudi dinamika sestankov ter zasedba vlog (opisano v nadaljevanju) na posameznem sestanku.



Slika 10. Podroben pregled Sprinta

Vmesni dnevi med sprinti omogočajo, da se uveljavljen razvojni cikel Podjetja d.o.o. relativno skladno zlije v ponavljajočo rutino brez, da bi določene aktivnosti bile podhranjene. Način pretvorbe uveljavljenega sekvenčnega načina dela v sprinte je prikazan na sliki 11:

- Snovanje zahtev je aktivnost, ki poteka kontinuirano, vzporedeno z razvojnimi aktivnostmi. Pri tem snovanje vedno razvoj prehiteva za 1 sprint.
- Načrtovanje rešitve / nadgradnje poteka takoj po koncu sprinta in pred začetkom sprinta. Osnova so zahteve, za katere ima ekipa dovolj časa, da se z njimi seznani.
- V teku klasičnega sprinta, kot ga predvideva metoda Scrum se znotraj ekipe izvaja preletanje aktivnosti implementacije, testiranja in odprave napak ter predaje.

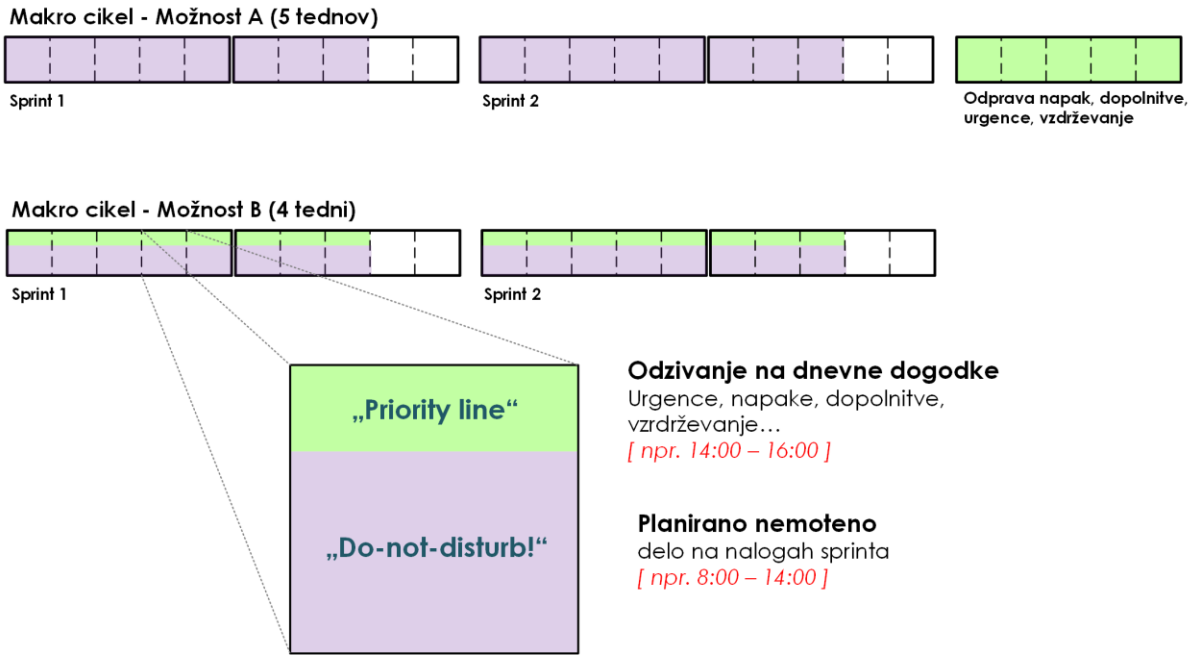


Slika 11. Prepletanje aktivnosti v sprintu

Obvladovanje prekinitev v razvojnem ciklu - Prevzemanje in naslavljanje napak in vzdrževalnih nalog

Delo je najbolj učinkovito opravljeno, ko je za to namenjen dovolj velik časovni interval, kjer ni prekinitev. Kljub temu se prekinitve pogosto zgodijo. Za to je potrebno pustiti nekaj časa pri izdelavi časovnega plana projekta/sprinta.

V zadnjem času se vedno bolj uveljavlja prehitevalni pas (angl. Fast-lane), ki je namenjen popravkom, urgentnim zahtevam iz podpore uporabnikom, asistence operativi in podobno. Tako naj bo tudi v Podjetju d.o.o. (slika 12).



Slika 12. Vgradnja prehitevalnega pasu v razvojni cikel

Definirali smo dva možna scenarija (glej sliko 12):

- **Makro cikel - Možnost A** prikazuje tri Sprinte, od katerih sta dva namenjena izključno razvoju inkrementa produkta (vijolični kvadratki) in en odpravi napak, vzdrževanju in urgentnim spremembam (zeleni kvadratki). Razvojni Sprinti trajajo dva tedna oziroma pet delovnih dni in vključujejo zgolj implementacijo novih funkcionalnosti. Dva dneva ob koncu drugega tedna vsakega Sprinta sta namenjena predstavitvi zahtev in nerazvojnemu delovnemu sestanku. Sledi tretji Sprint, ki traja zgolj en teden. V tem času razvijalci ne razvijajo novih funkcionalnosti, vendar zgolj odpravljajo napake, naslavljajo urgentne zahteve iz podpore uporabnikom, vzdržujejo sisteme ipd.
- **Makro cikel - Možnost B** sestoji iz dveh dvotedenskih Sprintov. Za razliko od prej opisane možnosti A, je tukaj vsak dan razdeljen na dve fazi. Ti sta “priority line” in “do-not-disturb!”. Slednja je namenjena nemotenemu delu na nalogah sprinta in traja cca šest ur na dan. V tem času se razvijalcev ne sme motiti (prepovedani klici, elektronska sporočila ipd.). S tem, ko razvijalci vedo, da jih noben ne more doseči oziroma kontaktirati, so bolj zbrani in se lahko osredotočijo zgolj na implementacijo ter so posledično bolj učinkoviti. Takšnemu intenzivnemu osredotočenemu delu sledi čas za “priority line”. V tem obdobju razvijalci ne razvijajo novih funkcionalnosti, ampak se odzivajo na dnevne dogodke. Sem sodi odprava napak, urgence in vse preostalo, kar smo že omenili v tretjem Sprintu možnosti A. S takšnim pristopom lahko agilno naslavljamo pereče probleme, ki morebiti nastanejo, in jih je potrebno obravnavati prioritetno.

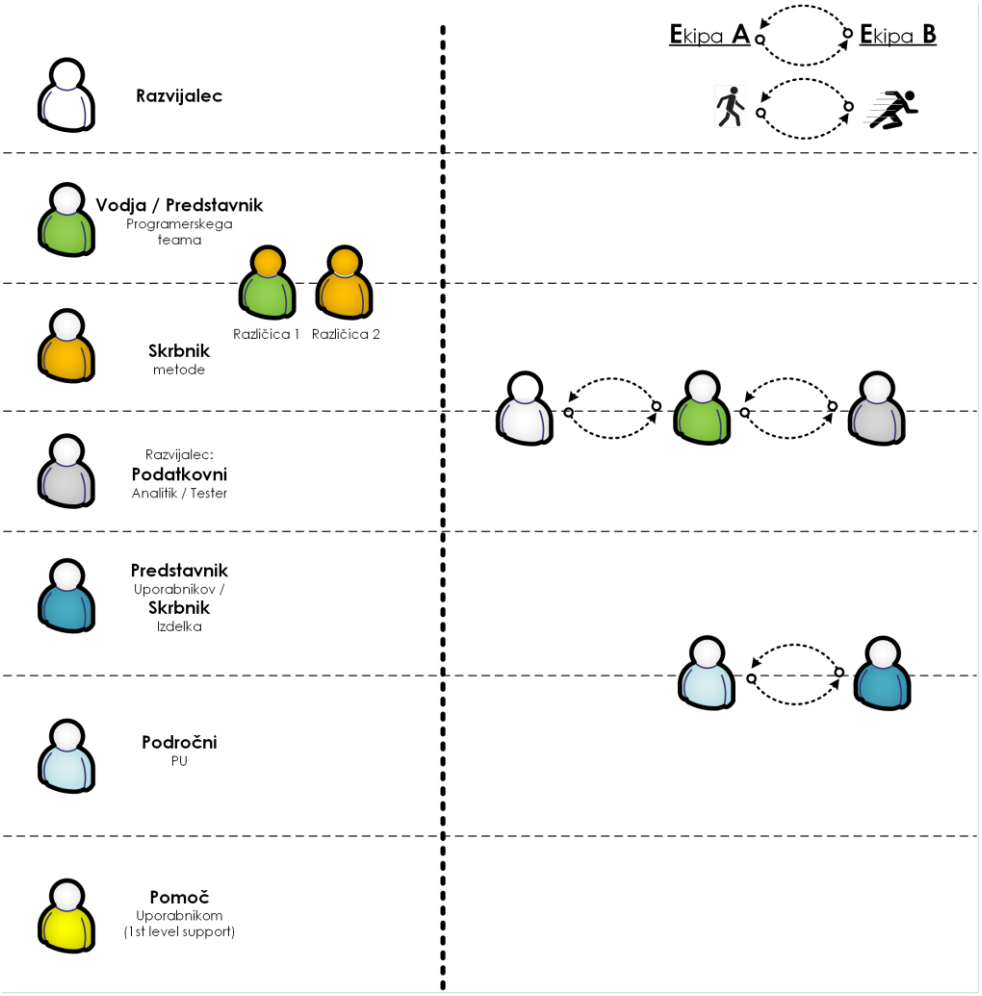
Prenos rešitev iz razvoja v produkcijo

V okviru sprinta programerska ekipa Podjetja d.o.o. implementira funkcionalnosti rešitve v razvoju za katere so bile prevzete specifikacije. Ob koncu vsakega sprinta je na voljo inkrement rešitve v razvoju, za katerega predstavnik uporabnikov potrdi oz. zavrne funkcionalnosti tekočega sprinta. Le to stori s pomočjo prevzemnega testiranja, ki ga v povezavi s podatkovnim analitikom izvaja že med samim razvojem.

Prevzemno testiranje ni edina oblika preverjanja kakovosti in sledenja skladnosti s specifikacijami v okviru posameznega sprinta. Razvijalec: podatkovni analitik/tester med sprintom preverja kakovost rešitve v razvoju. Prav tako, predstavnik uporabnikov prisostvuje rednim dnevnim sestankom na katerih lahko spremlja potek opravljenega dela. Prav tako programerski team predstavniku uporabnikov praviloma vsak dan zagotovi sprotne interne dnevne izdaje rešitev v razvoju, ki prikazuje trenutno stanje implementiranih funkcionalnosti. Dnevne izdaje predstavniku stranke služijo za sprotno preverjanje kakovosti, pri čemer lahko s svojim vidikom pomembno poveča kakovost rešitve v razvoju.

Vloge v razvojnem ciklu

Slika 13 povzema vloge vpletenih v razvoj IT rešitev Podetja d.o.o., njihove naloge in zakonitosti glede možnosti menjavanja vlog:



Slika 13. Vloge v metodi

Razvijalec, član programerskega teama

Razvijalec je del skupine, ki se ukvarja z razvojem tekom sprinta. Specializacije namenoma ne predvidevamo, je pa za pričakovati, da se bo sčasoma določena ekipa in posamezniki znotraj nje specializirali za določene funkcionalnosti. Glede na priporočila agilnih metod, razvojna skupina naj ne bi presegala sedem članov. Naloge vloge razvijalca so izključno razvojne narave. Razvijalec je tekom enega sprinta lahko aktiven zgolj na enem izdelku / v eni ekipi. Izmenjava razvijalcev med programskimi teami je tako za namen zagotavljanja večje kakovosti možna le na nivoju sprintov. V sklopu dnevnih usklajevanj razvijalci poročajo o napredku na projektu. Aktivne ure razvijalcev so glede na definiran časovni okvir porazdeljene med naloge povezane z aktualnimi projekti in naloge povezane z naslavljanjem napak in vzdrževanjem preteklih projektov. Razvijalci tako med delom na projektu ne opravljajo sprotnih nalog in obratno. Potrebna znanja vloge razvijalca so:

- Tvorjenje seznama nalog na osnovi uporabniških in tehničnih zgodb,
- Ocenjevanje obsega razvojnih nalog,
- Inženirska znanja iz področja vsebine projekta,
- Vzpostavitev okolja za samodejno gradnjo rešitve,
- Sodelovanje pri načrtovanju rešitve za podane zahteve,
- Priprava namestitvenega paketa rešitve,
- Nameščanje razvite rešitve,
- Osnovno poznavanje agilne metode razvojnega cikla,
- Konstruktivno moderirano sodelovanje na sestankih.

Vodja / predstavnik programerskega teama

Vodja/predstavnik programskega teama je član razvojne skupine, ki je izbran za posrednika med skupino in ostalimi udeleženci. S postavitvijo vloge vodja/predstavnik programskega teama se izognemo prepogostim (en razvijalec je lahko član več projektov - hkrati dela samo na enem) in prevelikim (prisotni bi bili vsi razvijalci vseh programskih teamov) projektnim sestankom. Potrebna znanja vloge vodja/predstavnik programskega teama so enaka potrebnim znanjem vloge razvijalec z dodatnimi zahtevami:

- Podrobno poznavanje vsebine celotnega projekta / izdelka,
- Širši vpogled v arhitekturo rešitve,
- Konstruktivno zastopanje mnenja razvojne ekipe napram predstavniku uporabnikov.

Razvijalec: podatkovni analitik / tester

Podatkovni analitik/tester je član programerske ekipe in hkrati predstavlja pomoč skrbniku izdelka / predstavniku uporabnikov v tehničnem smislu. Podatkovni analitik zbira, procesira in izvaja analize za namen iskanja povezave med in pomena podatkov. V vlogi testerja na drugi strani skrbi za preverjanje kakovosti rešitve v razvoju s pripravo testnih scenarijev na osnovi podanih zahtev in za izvajanjem potrebnih testov. Podatkovni analitik/tester v odvisnosti od zadolžitev opravlja tudi naloge razvijalca v ekipi. Vloga podatkovnega analitika ne more biti hkrati združena v isti osebi, kot je skrbnik metode ali vodilni razvijalec. Potrebna znanja vloge razvijalec: podatkovni analitik/tester so enaka potrebnim znanjem vloge razvijalec z dodatnimi zahtevami:

- Podrobno poznavanje vsebine celotnega izdelka,
- Podrobno poznavanje toka podatkov rešitve v razvoju,
- Priprava testnih scenarijev na osnovi uporabniških in tehničnih zgodb.

Skrbnik metode

Skrbnik metode (v metodi Scrum je to vloga "Scrum Master") je ključna in nujna vloga, ki skrbi, da metoda deluje dobro in pravilno. Gre za vlogo, ki ji naloge zapolnjujejo približno polovični delovni čas. Zato sta možni dve izvedbi skrbnika metode - (a) preostali čas je skrbnik lahko tudi razvijalec v ekipi, za katero skrbi in (b) oseba je lahko skrbnik metode pri dveh ekipah. Opcija "a" ima izrazite prednosti, saj v tem primeru skrbnik zelo dobro pozna in sodeluje pri delu ekipe, ter je venomer z njo - kar je v agilnih metodah zelo pomembno. S tem, ko je skrbnik metode tudi v vlogi razvijalca, ga preostali člani programskega tema dojemajo kot svojega zaveznika. Skrbnik metode ne more biti zunanji sodelavec. Priporočamo, da se vlogo skrbnika metode združi v isti osebi z vodjo / predstavnikom programerskega teama.

Skrbnik metode skrbi za moderiranje, izvajanje sestankov, izvajanje metrik, razreševanje organizacijskih in osebnih težav, ščiti skupino pred zunanjimi motnjami ipd. Potrebna znanja vloge skrbnik metodologije so:

- Priprava in posodabljanje Scrum table (razvojni projekti),
- Priprava in posodabljanje Kanban table (odprava napak in vzdrževanje),
- Organizacija sestankov,
- Moderiranje sestankov, umirjeno glajenje konfliktnih situacij,
- Poznavanje dnevne rutine (časovnice sestankov, usmerjenega dela, termina dnevnih opravil),
- Poglobljeno poznavanje agilne metode razvojnega cikla.

Predstavnik uporabnikov / skrbnik izdelka

Predstavnik uporabnikov/skrbnik izdelka je vsebinsko orientirana vloga katere naloga je zbiranje, usklajevanje in zapisovanje zahtev rešitve, vključno s postavljanjem prioritet posameznih zahtev na osnovi tega podajanjem zahtev v posamezne sprinte. Vsak programerski team v okviru razvoja določene rešitve sodeluje z enim predstavnikom uporabnikov. Predstavnik uporabnikov lahko ima širšo podporo področnih predstavnikov uporabnikov z globokim znanjem iz istega ali povezanih področij. Ti mu lahko pomagajo pri oblikovanju zahtev, pri čemer pa z programskim

teamom komunicira izključno ena oseba. Predstavnik uporabnikov preko rednih izdaj rešitve v razvoju spremlja razvoj izbranih funkcionalnosti v okviru sprinta. Odgovoren je za prevzemni preizkus implementirane zahteve in njeno potrditev oz. zavrnitev, s čimer pa skrbi za napredovanje rešitve. Vloga predstavnika uporabnika se lahko menja z vlogo področnega predstavnika uporabnikov, vendar ne znotraj sprinta. Obstaja torej možnost, da z eno ekipo e enem sprintu sodeluje en predstavnik uporabnikov, v naslednjem sprintu pa drugi. Potrebna znanja vloge predstavnik uporabnikov so:

- Podrobno dobro poznavanje vsebine celotnega izdelka,
- Snovanje zahtev rešitev,
- Tvorjenje uporabniških in tehničnih zgodb na osnovi zahtev rešitve,
- Poznavanje razvojnega cikla,
- Konstruktivno moderirano sodelovanje na sestankih,
- Testiranje izdelkov,
- Konstruktivno komuniciranje z ostalimi deležniki (uporabniki, razvijalci ipd.).

Področni predstavnik uporabnikov

Področni predstavnik uporabnikov sodeluje s predstavnikom uporabnikov/skrbnikom izdelka ter mu nudi podporo s svojim poglobljenim znanje določenega domenskega področja v katerega posega rešitev v razvoju. Področni predstavnik uporabnikov sodeluje izključno s predstavniki uporabnikov/skrbniki izdelkov posameznih programskih teamov in ne direktno z razvijalci ali vodjo/predstavnikom programskega teama. Področni predstavniki uporabnikov so praviloma prisotni tudi na demo predstavitvi rešitve v razvoju. Področni predstavnik uporabnikov lahko v določenem sprintu postane tudi predstavnik uporabnikov / skrbnik izdelka. Potrebna znanja vloge področni predstavnik uporabnikov so:

- Snovanje zahtev rešitev,
- Poznavanje razvojnega cikla,
- Konstruktivno moderirano sodelovanje na sestankih,
- Testiranje izdelkov,
- Poglobljeno domensko znanja določene domene.

Pomoč uporabnikom

Pomoč uporabnikom (žargonsko "first-level support") razrešuje vprašanja/zahteve povezane z nujnimi situacijami, napakami, vzdrževanjem in manjšimi dopolnitvami ter izboljšavami v preteklosti razvitih rešitev. Naloga te vloge je razreševanje situcaij, kolikor je le mogoče. Zadeve napredujejo do razvijalcev le v primeru, ko jih ni možno rešiti že na tem nivoju. Pri tem se zahteve dodajo na Kanban tablo za napake / urgence / dodatne zahteve in se določi njihova prioriteta. Razrešitev je lahko nujna (presoja vloge "pomoč uporabnikom") ali pa - privzeto - ko ima ekipa rezerviran čas za naslavljanje zadev na kanban tabli. V ta namen je nujno, da poznajo funkcionalnosti in delovanje razvitih rešitev, ki so v uporabi. V namen čim boljšega informiranja se pričakuje prisotnost vsaj na demonstraciji novosti, še bolje pa tudi na sestanki, kjer se predstavijo zahteve za nov razvoj. Potrebna znanja vloge pomoč uporabnikov so:

- Poznavanje funkcionalnosti razvitih rešitev v uporabi,
- Proaktivno reševanje zadev,
- Zmožnost ovrednotenja nujnosti prispelih prijav.

Izdelki in odgovornosti

Poglavitni izdelki / aktivnosti, ki jih v Podjetju d.o.o. obvladujejo z namenom dostave informacijskih rešitev oz. nadgradnje le-teh so naslednje:

- Vizija izdelka,
- Cilj iteracije,
- Seznam zahtev izdelka,
- Seznam zahtev iteracije,
- Opredelitev opravljenega,
- Graf preostanka dela v sprintu,
- Graf preostanka dela do predaje,
- Revizija in retrospektiva iteracije,
- Visokonivojska arhitektura,
- Kadrovski plan sprinta,
- Seznam nalog,
- Kartica za Scrum tablo,

- Kartica za Kanban tablo,
- Scrum tabla,
- Kanban tabla za napake,
- Rešitev,
- Testni primerki enot,
- Okolje za testiranje rešitve,
- Testni primeri za funkcionalne teste,
- Testni primeri za sistemske teste,
- Nadgradnja rešitve,
- Nameščena rešitev,
- Sestanki - na nivoju razvojne skupine,
- Sestanki - na nivoju usklajevanja razvojnih skupin.

Vizija izdelka (angl.: Product Vision) (odgovorni: Predstavnik uporabnikov/skrbnik izdelka)

Dokument s katerim opredelimo dolgoročne cilje projekta / rešitve. Določa smer razvoja in je glavno vodilo razvojne ekipe. Vizija projekta mora biti jasno in natančno definirana.

Cilj iteracije (angl.: Sprint Goal) (odgovorni: Razvojna skupina, predstavnik stranke)

Dokumentirati je potrebno cilj posamezne iteracije. Razvojna ekipa se osredotoči na dosego cilja, saj je jasno definirano zakaj se gradi določen inkrement končne rešitve.

Seznam zahtev izdelka (angl.: Product Backlog) (odgovorni: Predstavnik stranke)

Dokument (sklop dokumentov - uporabi se obstoječ način podajanja zahtev) podaja podrobne zahteve za funkcionalnosti, ki se v planu realizacije v naslednji iteraciji. Zahteve morajo biti opremljene s prioritetami, ki bodo služile izboru umeščanja funkcionalnosti v posamezne sprinte.

Seznam zahtev iteracije (angl.: Sprint Backlog) (odgovorni: Razvojna skupina)

Glede na urejene zahteve iteracije (po prioriteti) se kreira t.i. seznam zahtev iteracije, ki bodo realizirane v naslednjem sprintu. Konkretnih zahtev ni potrebno prestavljati iz zahtev iteracij v seznam zahtev (sprinta). Obstajati pa mora dokument, ki s pomočjo

sklicev jasno pove, katere funkcionalnosti so v planu sprinta. Tudi ta seznam mora biti opremljen s prioritetami - v primeru, da količina nalog in/ali kadrovska zasedba sprinta ne bo omogočala realizacije vseh planov, je prioriteta ključ urejanja po vrstnem redu.

Opredelitev opravljenega (angl.: Definition of Done) (odgovorni: Razvojna skupina, Skrbnik metodologije)

Vsak seznam zahtev izdelka ima merila sprejemljivosti, ki jih je potrebno doseči, da je delo opravljeno. Opredelitev opravljenega se lahko izdela v obliki seznama, ki običajno vsebuje tudi merila kakovosti, omejitve in splošne nefunkcionalne zahteve.

Graf preostanka dela v sprintu (angl.: Sprint Burndown Chart) (odgovorni: Skrbnik metodologije)

Graf v padajoči obliki prikazuje število preostalih točk v odvisnosti od dneva. Jasno mora tudi biti vidna idealna črta, kar pomaga ekipi vizualno spremljati napredek.

Graf preostanka dela do predaje (angl.: Release burn-up chart) (odgovorni: Skrbnik metodologije)

Naraščajoč graf ponuja razvojni ekipi, da lahko prepozna in spremlja napredek do roka izdaje.

Inkrementalne rešitve (angl.: Potentially shippable product increment) (odgovorni: Razvojna skupina)

Izhod vsakega izmed sprintov je inkrementalna rešitev. Delo vseh ekip mora biti integrirano pred koncem vsakega sprinta (integracija se izvaja tekom sprinta). S tem zvišamo osredotočenost, agilnost in fleksibilnost z zmanjšanjem dela v teku (angl.: Work In Progress). Inkrementalne rešitve ponujajo možnost za kasnejše izboljšave. Inkrementalna rešitev je primerna, ko imamo dosežene vse točke iz seznama opravljenega (angl.: Definition of Done).

Revizija in retrospektiva iteracije (odgovorni: Razvojna skupina)

Na koncu sprinta vsaka izmed ekip opravi retrospektivo iteracije, ki se izvede v obliki sestanka. Sestanek služi odkrivanju pomanjkljivosti tekom izvajanja iteracije (medosebne ovire, način dela) in je podlaga za izboljšave v kasnejših iteracijah.

Visokonivojska arhitektura (HLA) (odgovorni: Koordinator funkcionalnosti, tehnični koordinator)

Dokument, ki podaja kratko in jasno sliko nad celotno (tehnično) arhitekturo rešitve - vključeni izdelki, komunikacijski vmesniki in protokoli, namestitveni diagram ipd.

Kadrovski plan sprinta (odgovorni: Skrbnik metodologije)

Dokument (ali preglednica, vpogled v obstoječ sistem kadrovske službe ipd.) nedvoumno odgovori, katere osebe (razvijalci) bodo udeleženi v sprintu ter koliko časa (ure) bodo na voljo.

Seznam nalog (odgovorni: Razvojna skupina)

Iz seznama zahtev (sprinta) razvojna ekipa sestavi seznam nalog (sprinta). Minimalna narava izdelka je kolekcija kartic v fizični obliki in z ocenjenim trudom (točke). Če poslovnih podjetja in ostali interni predpisi to zahtevajo, se seznam nalog tudi vnese v elektronsko obliko. V tem primeru priporočamo, da se poleg planiranega truda predvidi tudi (prazno) polje dejanskega truda, ki se izpolni po opravljenih nalogah.

Kartica (za scrum tablo - naloge) (odgovorni: Razvojna skupina, skrbnik metodologije)

Kartice ustvarjajo razvijalci tekom planiranja sprinta v skladu z metodo Scrum. Vodja razvijalcev lahko v izogib predolgemu sestanku predhodno pripravi seznam kartic s predlagano oceno truda - kar ekipa potrdi oz. dopolni. Ena uporabniška ali tehnična zgodba tipično rezultira v večjem številu kartic - po filozofiji metode Scrum. Posamezna kartica za nalogo (opravilo) mora vsebovati vsaj: naziv naloge, (kratek) opis, (kratek) opis kriterijev kakovosti ter oceno truda. Kartica lahko vsebuje tudi kratico uporabniške ali tehnične zgodbe, ki ji pripada.

Če poslovni in ostali interni predpisi podjetja to zahtevajo, se kartice tudi vnese v elektronsko obliko.

Kartica (za kanban tablo - napake in pomanjkljivosti) (odgovorni: Predstavnik skupine)

Kartice za kanban tablo vsebujejo identifikator napake (še posebej, če napake vodimo v sistemu za vodenje napak) in/ali opis napake. Kartice nimajo ocene truda, saj bodo umeščene na kanban tablo. Kartica naj vsebuje tudi čas prijave napake, saj bo tako možno meriti čas cikla.

Scrum tabla (odgovorni: Skrbnik metodologije)

Tabla za vodenje kartic sprinta.

Kanban tabla za napake (odgovorni: Skrbnik metodologije)

Tabla za vodenje kartic v obdobju odprave napak in pomanjkljivosti.

Rešitev (odgovorni: Predstavnik skupine V&V)

Izdelek je končna ali vmesna rešitev razvojnega cikla.

Testni primerki enot (test case-i) (odgovorni: Razvojna skupina)

Teste enot pripravljajo razvijalci, ki jih tudi poganjajo. Testi enot predstavljajo uveljavljen mehanizem samodejnega testiranja. Namen testiranja enot je, da se že v postopek preverjanja kakovosti (aktivnost V&V) preda tehnično gledano čim kakovostnejša rešitev.

Okolje za testiranje rešitve (odgovorni: Predstavnik skupine V&V)

Izdelek predstavlja nabor programske in strojne opreme, ki je potrebna za nemoteno testiranje rešitve, ki nastaja znotraj razvojnega cikla.

Testni primeri za funkcionalne teste (odgovorni: Predstavnik skupine V&V)

Testni primeri so dokumentirani (dokument na testni primerek) postopki za funkcionalne teste. Še bolje je, če se tekom aktivnosti V&V uporablja (vsaj delno)

avtomatizirano testiranje - v tem primeru je izdelek samodejni test izbranega / uporabljenega testnega orodja.

Testni primeri za sistemske teste (odgovorni: Predstavnik skupine V&V)

Podobno kot testni primer za funkcionalne teste. Le da so ti izdelki testi, ki ne testirajo funkcionalnih, temveč sistemske zahteve (odzivnost, varnost, testi preobremenitev ipd.)

Nadgradnja rešitve (odgovorni: Razvojna skupina)

Izdelek je del programske in/ali strojne opreme, ki deluje kot dodatek k neki obstoječi rešitvi.

Namestitveni paket rešitve (odgovorni: Razvojna skupina)

Izdelek omogoča nameščanje izdelane rešitve pri stranki. Tipično takšen paket sestoji ne samo iz namestitvenega paketa v tehničnem smislu, temveč tudi podrobnih navodil, kako ta paket uporabiti. Izhajamo iz obstoječih dobrih praks in dokumentacije podjetja.

Nameščena rešitev (odgovorni: Osebje podjetja po navodilih razvojne skupine (izdelek izven obsega razvojnega cikla))

Izdelek je razvita, testirana, nameščena in delujoča rešitev pri stranki.

Posodobljena rešitev (odgovorni: Osebje podjetja po navodilih razvojne skupine (izdelek izven obsega razvojnega cikla))

Izdelek je razvita, testirana, nameščena in delujoča rešitev pri stranki - v primeru, da gre le za nadgradnjo rešitve, ki jo je stranka pred tem že uporabljala.

Sestanki - na nivoju razvojne skupine (odgovorni: Skrbnik metodologije)

Sestanki na nivoju ene razvojne skupine - npr. planiranje sprinta, dnevni sestanki ipd.

Sestanki - na nivoju usklajevanja razvojnih skupin (odgovorni: Skrbnik metodologije iz vodilne razvojne skupine)

Sestanki nad nivojem skupine - npr. sestanek za usklajevanje skupin, v primeru da eno rešitev (a njene različne vidike) razvija več ločenih razvojnih skupin.

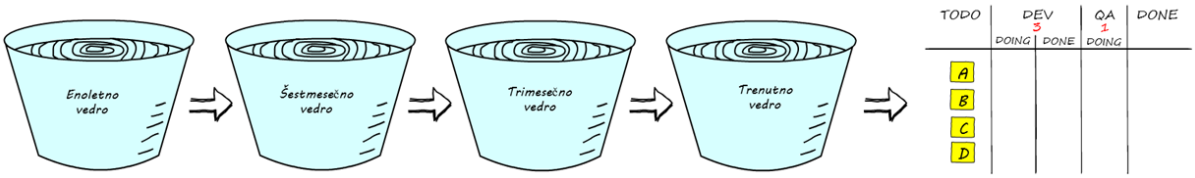
Definiranje funkcionalnih in nefunkcionalnih zahtev informacijskih rešitev ter njihovo podajanje v razvoj

V okviru planiranja se zahteve projekta zberejo v seznamu zahtev (ang. Product Backlog). Predstavnik uporabnikov/skrbnik izdelka oblikuje vizijo projekta, s čemer se začne razvojni proces. Predstavnik uporabnikov/skrbnik izdelka oblikuje seznam zahtev (ang. Product Backlog), katere mora programski team izpolniti za uspešen zaključek projekta. Pri oblikovanju so mu lahko v primeru potrebe po dodatnem domenskem znanju v pomoč področni predstavniki uporabnikov. Le ti s svojim poglobljenim domenskim znanjem lahko pomembno izboljšajo opise zahtev rešitve v razvoju. Zbrane zahteve predstavnik uporabnikov/skrbnik izdelka prioritizira in glede na to razporedi v sprinte na osnovi česar se določi trajanje posameznega projekta.

Pred začetkom vsakega spinta se predstavnik uporabnikov/skrbnik izdelka sestane z programskim teamom na predstavitvi zahtev rešitve v razvoju katerega namen je seznanitev programskega teama z zahtevanimi funkcionalnostmi. V začetku sprinta se na planiranju sprinta

sestane programski team ter predstavnik uporabnikov/skrbnik izdelka ter skupaj pregledajo seznam zahtev in določijo katere izmed njih bodo realizirane. Na podlagi izbranih zgodb, programski team sestavi seznam zahtev (ang. Sprint backlog). Izbrane funkcionalnosti so razbite na posamezne naloge, ki so prioritizirane in z ocenjenim trudom potrebnim za izvedbo.

Poleg tekočega planiranja rešitev v razvoju pa pomemben vidik predstavlja dolgoročno planiranje razvoja. Dolgoročno planiranje omogoča metoda planiranja v velikosti vedra (ang. Bucket Size Planning) je prikazano na sliki 14. Metoda je del agilnega projektnega vodenja pri katerem se potencialni prihodnji projekti premikajo skozi različna navidezna vedra, ki so lahko predstavljena v obliki table. Štiri zaporedna vedra ločijo dolgoročne od kratkoročnih planov, pri čemer imena veder ne izražajo dejanske oddaljenosti rešitev od postavljenih rokov.



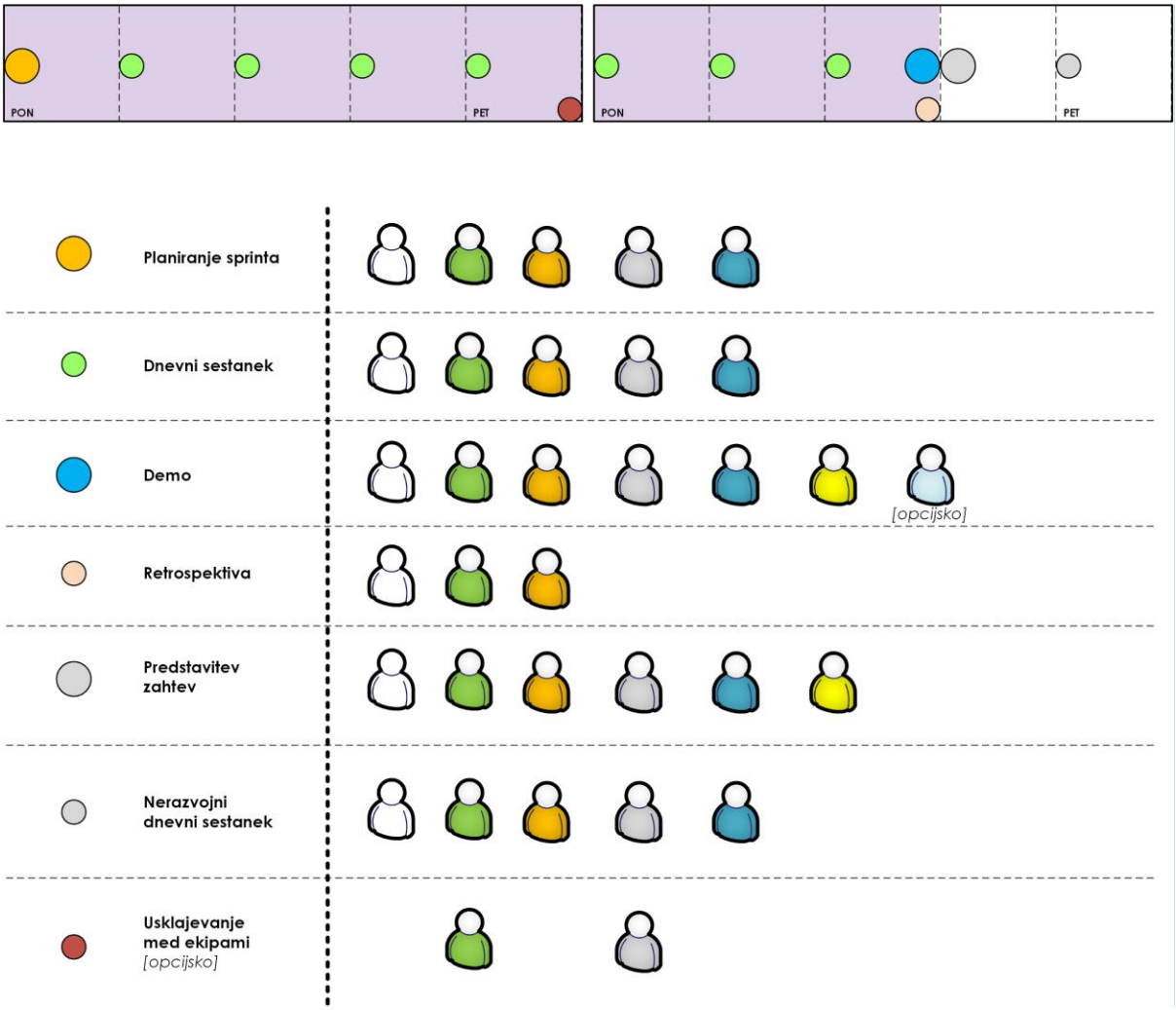
Slika 14: Planiranje v velikosti vedra

Prvo vedro je “enoletno” vedro, ki vsebuje ideje, cilje in načrte potencialnih projektov. Razvoj ideje do cilja, projekt prenese v drugo, “šestmesečno”, vedro. Le to vsebuje projekte, ki jih podjetje želi izpeljati. Ideje so definirane bolj specifično, vendar še zmeraj brez vseh podrobnosti. Naslednje vedro je “trimesečno” vedro v katerega premaknemo projekt, ki ga nameravamo razviti, iz cilja pa se oblikujejo zahteve. Vedro že vsebuje podrobno definirana opravila v obliki uporabniških zgodb. Zadnje vedro je “trenutno” vedro, ki vsebuje ločeno definirana opravila in ga je mogoče enačiti s Scrum tablo.

Dolgoročno planiranje poteka na sestankih predstavnikov uporabnikov/skrbnikov izdelkov. Pri tem posamezni predstavniki uporabnikov/skrbniki izdelkov glede na potrebe predlagajo ideje potencialnih projektov in le te na kratko predstavijo preostalim predstavnikom uporabnikov/skrbnikom izdelkov skupaj z dodeljeno prioriteto nujnosti razvoja. Prehodi projektov med vedri potekajo na osnovi prioritet potencialnih projektov. Pri tem se primarni prioriteti, ki jo dodeli predlagatelj pridružijo še prioritete preostalih predstavnikov uporabnikov/skrbnikov izdelkov, ki se na vsakem sestanku dolgoročnega planiranja posodobijo. Potencialni projekti z najvišjimi prioritetami se prenesejo v naslednje vedro do natančnejšega planiranja funkcionalnosti in prenosa potencialnega projekta v rešitev v razvoju.

Sestanki in mejniki

V agilnih razvojnih metodah je število sestankov minimalno. Na drugi strani pa je njihovo redno in pravilno izvajanje kritičnega pomena. Povzemamo sestanke in njihove podrobnosti tekom sprinta.



Slika 15. Podroben pregled Sprinta s sestanki in udeleženci sestankov

Planiranje sprinta

Namen: Sestanek planiranja sprinta je namenjen prenosu uporabniških zahtev v razvojni cikel (sprint). Na osnovi predstavljenih zahtev in njihove študije razvijalci kreirajo konkretne naloge za celoten sprint in jih po končanem sestanku začnejo izvajati v skladu s postavljenimi prioritetami. Izbor zahtev, ki bodo predmet razvoja v printu se izvede na osnovi ocenjevanja nalog. Npr. predstavnik uporabnika želi izvedbo zahtev A, B, C, D in E. Razvijalci pa na osnovi ocene nalog ugotovijo, da so na podlagi kapacitete sprinta sposobni dostaviti le izdelke A, B, C in D. Predstavnik uporabnikov se strinja, da zahteva E tekom sprinta ne bo izvedena, razvojna ekipa pa se zaveže, da bodo zahteve A - D izvedene in se zaveze praviloma (razen v izjemnih okoliščinah) tudi drži.

Vsaka ekipa ima svoj sestanek planiranja sprinta.

Termin: Ponedeljek, prvi dan sprinta, ista ura in prostor, kot je namenjeno sicer za dnevne sestanke. Npr. 8:30.

Predvideno trajanje: Tipično sestanek traja po metodi Scrum 2-4 ure. Zaradi predhodne študije zahtev in možnosti preliminarne priprave nalog pa je trajanje ocenjeno na 1-2 uri.

Udeleženci: Vsi člani ekipe (razvijalci, vodilni razvijalec, podatkovni analitik), predstavnik uporabnikov, skrbnik metode.

Potek: Predstavnik uporabnikov predstavi zahteve v padajočem vrstnem redu glede na prioritete. Skrbnik metode izračuna kapaciteto sprinta. Razvijalci razbijejo zahteve na naloge in s pomočjo tehnike "planning poker" ocenijo naloge. Ocenjevanje poteka s pomočjo točk. Preko kapacitete sprinta in ocen se razvijalci zavežejo k izvedbi izbranih zahtev. Skrbnik metode sestanek moderira.

Izhod: Izhod sestanka je seznam nalog sprinta, prazen diagram preostanka dela, ter zaveza za izvedbo dogovorjenih nalog.

Dnevni sestanek

Namen: Dnevni sestanek je namenjen ekipi, da preveri trenuten napredek, posodobi graf preostanka dela, in poroča / demonstrira predstavniku uporabnikov novosti. Vsak izmed razvijalcev mora spregovoriti v svojem imenu - vsaj našteti opravljene naloge prejšnji dan, odprte izzive in okviren plan za prihajajoči dan. Na tem sestanku NE razrešujemo težav - temu je namenjen razvojni cikel!

Predstavnik uporabnikov na sestanku praviloma spregovori le, če je pozvan k besedi.

Termin: Vsak dan sprinta, ista ura in prostor. Npr. 8:30.

Predvideno trajanje: Maksimalno 10 minut, izjemoma 15 minut.

Udeleženci: Vsi člani ekipe (razvijalci, vodilni razvijalec, podatkovni analitik), predstavnik uporabnikov, skrbnik metode.

Potek: Sestanek poteka na tak način, da vsi udeleženci vidijo trenutno stanje nalog sprinta in graf preostanka dela. Spregovorijo eden po eden. Skrbnik metode sestanek moderira.

Izhod: Posodobljen graf preostanka dela. V primeru, da člani ekipe naloge na tabli prestavljajo (v delu → narejeno) le na dnevnem sestanku, se posodobi tudi tabla z nalogami.

Demo

Namen: Demo je namenjen prikazu rezultatov tekočega sprinta. Njegov namen je seznanjanje predstavnika uporabnikov/skrbnika izdelkov s trenutnim stanjem in implementiranimi funkcionalnostmi. Na tem sestanku se predstavnik uporabnikov odloči, katere funkcionalnosti so zrele za produkcijo in katere ne.

Termin: Ob zaključu sprinta, zadnji 2 uri. Morda bi bilo smiselno, da se demo termini ne prekrivajo po ekipah.

Predvideno trajanje: Praviloma traja sestanek do 1 ure. Pred tem ima ekipa 1 uro, da pripravi in preizkusi delujoč demo.

Udeleženci: Vsi člani programskega teama (razvijalci, vodilni razvijalec, podatkovni analitik/tester), predstavnik uporabnikov/skrbnik izdelka, skrbnik metode, pomoč uporabnikom, opcijsko tudi področni predstavniki uporabnikov. Demo je načeloma odprt za vse, ki bi se ga želeli udeležiti.

Potek: Programerski team predstavi in prikaže razvite funkcionalnosti tekočega sprinta. Predstavnik uporabnikov/skrbnik izdelka ter področni predstavniki uporabnikov podajo morebitne pripombe in predloge.

Izhod: Posodobljen seznam zahtev.

Retrospektiva

Namen: Namen retrospektive je pregled razvojne metodologije v smislu potencialnih izboljšav razvojnega procesa.

Termin: Ob zaključu sprinta, konec 8 dneva sprinta.

Predvideno trajanje: Predvidoma sestanek traja 15-30 minut.

Udeleženci: Razvijalci programskega teama, vodilni razvijalec, razvijalec: podatkovni analitik/tester in skrbnik metodologije.

Potek: Sestanek vodi skrbnik metodologije, pri čemer člani razvojne ekipe v obliki diskusije delijo svoje izkušnje in morebitne pomanjkljivosti.

Izhod: Izbrana potencialna izboljšava razvojnega procesa v ekipi.

Predstavitev zahtev

Namen: Namen sestanka je podrobnejša predstavitev zahtev še pred uradnim začetkom sprinta. Programerski teama se seznani z domeno in vsebino razvojnih nalog naslednjega sprinta. Na ta način je začetek sprinta lažji in bolj tekoč. V primeru, da bodo zahteve implementirali v več ekipah, poteka sestanek 2-fazno – najprej z vodilnimi razvijalci, nato znotraj ekip.

Termin: Četrtek (2 dni) pred začetkom novega sprinta. Ob terminu dnevnih sestankov.

Predvideno trajanje: 1-2 uri.

Udeleženci: Vsi člani programskega teama (razvijalci, vodilni razvijalec, podatkovni analitik/tester), predstavnik uporabnikov/skrbnik izdelka, skrbnik metode, pomoč uporabnikom.

Potek: Predstavnik uporabnikov/skrbnik izdelka predstavi vsebini zahtev naslednjega sprinta. Člani programskega teama se seznanijo z domeno pri čemer predstavnik uporabnikov/skrbnik izdelka odgovori na morebitna vsebinska vprašanja.

V primeru, da bodo zahteve implementirali v več ekipah, poteka sestanek 2-fazno – najprej z vodilnimi razvijalci, nato znotraj ekip.

Izhod: Okvirno razumevanje zahtev naslednjega sprinta. Osnova za načrtovanje.

Nerazvojni dnevni sestanek

Namen: Nerazvojni dnevni sestanek je sestanek z enakimi zakonitostmi kot dnevni sestanek. A poteka na dan, ki je namenjen študiji zahtev. Razvijalci preverijo in uskladijo svoje razumevanje zahtev s predstavnikom uporabnikov.

Termin: Enako kot dnevni sestanek: ista ura in prostor. Npr. 8:30.

Predvideno trajanje: 10 - 15 minut

Udeleženci: Vsi člani programskega teama (razvijalci, vodilni razvijalec, podatkovni analitik/tester), predstavnik uporabnikov/skrbnik izdelka, skrbnik metode.

Potek: Moderiran pogovor s predstavnikom uporabnikov.

Izhod: Dopolnjeno razumevanje zahtev naslednjega sprinta. Osnova za načrtovanje.

Usklajevanje med ekipami

Namen: Namen sestanka je uskladitev med ekipami, če se 2 ali več ekip spopada z razvojem skupnih funkcionalnosti.

Termin: Vnaprej dogovorjen dan in ura. Tipično na sredini sprinta v popoldanskem času.

Predvideno trajanje: 15 - 30 minut.

Udeleženci: Vodje razvijalcev programskih teamov, podatkovni analitiki/testerji.

Potek: Sestanek poteka na tak način, da vsi udeleženci predstavijo trenutno stanje nalog sprinta in graf preostanka dela. Razrešijo morebitne odvisnosti med ekipami. Spregovorijo eden po eden.

Izhod: Posodobljena razpredelnica odvisnosti med ekipami.

Poleg strogo razvojnih sestankov metoda predvideva tudi nabor sestankov, namenjenih snovanju in izboru zahtev za naslednji sprint:

Dolgoročno planiranje / Sestanki med PU

Namen: Namen sestanka je postaviti dolgoročen plan razvoja IT rešitev. Planiranje se izvaja po principu “bucket-size-planninga”, kjer posamezne sklope funkcionalnosti v odvisnosti od nujnosti in definiranosti, umeščamo v enega izmed “veder”. Vsi predstavniku uporabnikov in področji predstavniki uporabnikov se dogovorijo o tem, katere sklope funkcionalnosti prestaviti v vedro, ki bo šlo prej v razvoj.

Termin: Vnaprej dogovorjeni termini; npr. vsaka druga sreda (sinhronizirano z razvojnimi sprinti) popoldan ob 14:00.

Predvideno trajanje: Do 1 ure.

Udeleženci: Predstavniki uporabnikov/skrbniki izdelkov teri področni predstavniki uporabnikov. Vodstvo podjetja opcijsko.

Potek: Planiranje poteka po metodi planiranja v velikosti vedra (ang. Bucket Size Planning). Dodajajo se ideje potencialnih projektov, izmed idej se definirajo projekti kateri postanejo cilj in nato se izmed teh izberejo tisti, katerih razvoj se bo začel v kratkem. Premik projektov med vedri poteka na osnovi prioritet. Le to v prvem koraku določi predstavnik uporabnikov, ki projekt predlaga ter njegovo prioriteto smiselno utemelji. Tej prioriteti se nato dodajo še prioritete preostalih predstavnikov

uporabnikov ter skupaj posamezen projekt utežijo do prehoda oz. ne prehoda v naslednje vedro.

Izhod: Dolgoročni plan razvoja IT rešitev.

5. Zaključek

Agilnejši razvoj informacijskih rešitev in storitev je vse prej kot nova želja razvojnih ekip. V kombinaciji s spreminjajočimi poslovnimi razmerami, ko vitkost in visoka odzivnost razvojnih ekip več ne predstavlja le konkurenčne prednosti, temveč nujo za obstanek na trgu, so agilne metode razvoja postale ustaljena praksa.

V želji po osvojitvi agilnih pristopov in čim hitrejši uvedbi sodobnih agilnih metod razvoja, v podjetjih pogosto ne sprejmejo optimalnih rešitev. Tako vse prepogosto vidimo primere, ko podjetja prehitro zavržejo dolgoletne metode razvoja, ki so tipično sekvenčno in plansko usmerjene. Zamenjujejo jih z agilnimi metodami, ki so pomanjkljivo ali celo neustrezno prilagojene njihovemu poslovnemu okolju. Posledice takšnega ravnanja so žal pogosto lahko tudi negativne.

Pri predmetu »Razvoj informacijskih sistemov« se študentje poglobljeno ukvarjajo tudi z metodami agilnega razvoja informacijskih rešitev. Interno učno gradivo je zato bilo izdano z namenom demonstracije izbora, prilagoditve in celovitega pogleda na konkretne razvojne procese namišljenega Podjetja d.o.o.

Viri

[1] The Agile Manifesto, http://agilemanifesto.org, nazadnje obiskano 2021/09.

[2] Scrum.org, https://www.scrum.org, nazadnje obiskano 2021/09.

[3] Version One State Of Agile Survey Report, http://stateofagile.versionone.com, nazadnje obiskano 2021/09.

[4] TAKEUCHI Hirotaka, NONAKA Ikujiro, The New New Product Development Game, Harvard Business Review, 1986.

[5] http://www.eylean.com/blog/2014/11/scrumban-on-demand-vs-long-term-planning/, Scrumban: on demand vs. long-term planning, obiskano 2021/09.

[6] http://implementingagile.blogspot.si/2011/06/scrum-metrics.html, Scrum metrics, obiskano 2021/09.

[7] BITTNER Kurt, LO GIUDICE Diego, DeMARTINE Amy, MINES Christopher, HAMMOND Jeffrey S., TURRISI Taylor, IZZI Matthew, Forrester Research – Boost Application Delivery Speed And Quality With Agile DevOps Practices, 2016.

[8] http://brodzinski.com/2012/05/slack-time.html, “Slack Time” Pawel Brodzinski, obiskano 2021/09.

[9] http://goo.gl/dkrbGE, Whitepaper – Scrum vs Kanban vs. Scrumban, 2021/09.

[10] http://goo.gl/JgfaaA, Eyelean “Scrum-ban for project management”, obiskano 2021/09.

[11] https://switchingtoscrum.files.wordpress.com/2013/12/implementing-scrumban_v1-32.pdf, Implementing Scrumban: A short guide to implementing Scrumban at your organization, obiskano 2021/09.

[12] BRECHNER Eric, Agile Project Management with Kanban, Microsoft Press, 2015.

[13] http://stateofagile.versionone.com/, 10th Annual State of Agile Report, obiskano 2021/09.

[14] KLIPP Paul, Getting Started with Kanban, Amazon Digital Services LLC, 2014.

[15] BITTNER Kurt, LO GIUDICE Diego, DeMARTINE Amy, MINES Christopher, HAMMOND Jeffrey S., TURRISI Taylor, IZZI Matthew, Forrester Research – Boost Application Delivery Speed And Quality With Agile DevOps Practices, 2016.

[16] Gartner, Bill Holz presentation “Agile in the Enterprise”, 2015.