Aplikacija in integracija agilnih projektnih metodologij pri projektih razvoja mobilnih aplikacij

Aljaž Daković [1] Igor Vrečko[2]
[1] Freelance project manager za projekte razvoja spletnih in mobilnih aplikacij in podiplomski študent na EPF Maribor
[2] UM – Ekonomsko-poslovna fakulteta, Inštitut za projektni management, Razlagova 14, 2000 Maribor
e-pošta: aljaz.dakovic@hotmail.com; igor.vrecko@uni-mb.si

Povzetek

Razvoj mobilnih aplikacij je nova in izjemno hitro razvijajoča se in rastoča industrijska panoga. Dinamiko tega razvoja in rasti zagotavljajo razvojni projekti mobilnih aplikacij, ki širijo svetovni spekter projektov in s svojimi specifičnimi zahtevami postavljajo pred stroko projektnega menedžmenta nove izzive. Njihove posebnosti se kažejo v relativni kratkotrajnosti in majhni kompleksnosti glede na število vključenih posameznikov v projektni sistem teh projektov od začetka njihovih življenjskih ciklusov pa do zaključka faz izvajanja. Vsled temu in ob določenih vsebinskih posebnostih teh projektov so aplikabilnost in učinkovitost klasičnih konceptov, predvsem pa metodologij projektnega menedžmenta precej omejeni. V zadnjih letih se pogosto izpostavlja primernost agilnih projektnih metodologij za projekte s podobnimi karakteristikami, kot jih imajo projekti razvoja mobilnih aplikacij. Za slednje je v članku kot primerna podlaga za pripravo in vodenje predstavljena metodologija, ki predstavlja izvirno adaptacijo in integracijo dveh agilnih metodologij – Scrum in Extreme Programming, in je bila preverjena z aplikacijo na dveh tovrstnih projektih.

***Ključne besede**: projektne metodologije, agilni projektni menedžment, agilne metodologije, mobilne aplikacije, Scrum, Extreme Programming*

1. Uvod

V zadnjih nekaj letih smo priča vzponu nove industrijske panoge – razvoju mobilnih aplikacij. Leto dni po tem, ko je leta 2007 podjetje Apple predstavilo tržišču prvi pametni telefon, je takratni direktor Steve Jobs na eni od svojih legendarnih predstavitev naznanil vzpostavitev virtualne trgovine, namenjene prodaji mobilnih aplikacij (v nadaljevanju: MA). Razvoj in prodaja MA je sicer potekala že veliko prej, vendar je bil razvoj zelo zahteven in zaradi pomanjkanja učinkovitih prodajnih kanalov ekonomsko neopravičen. Danes so razmere povsem drugačne in največji izzivi razvoja aplikacij za mobilne naprave so povezani s strojnimi posebnostmi in fizičnimi omejitvami samih naprav. Proizvajalci in ponudniki operacijskih sistemov mobilnih naprav prepoznavajo te izzive in skušajo z vsako naslednjo verzijo SDK (*Software development kit – skupek razbijalskih orodji namenjen razvoju aplikacij za določene programske pakete*) razbremeniti razvijalce MA. Slednjim je dana možnost dostopanja do številnih knjižnic in orodij za razvoj MA, kar je poenostavilo razvoj obstoječih in nastajanje novih aplikacij. V sled tega je postala industrija razvoja MA ena najhitreje rastočih industrijskih vej, katere obseg poslovanja za leto 2013 ocenjuje raziskovalna agencija ABI Research na vrtoglavih 27 milijard ameriških dolarjev.

Eksponentna rast te panoge je pritegnila številne razvijalce in mnoge podjetnike. Projekti razvoja MA predstavljajo danes pomemben delež projektov, ki se v svetu izvajajo. Zaradi določenih posebnosti tovrstnih projektov se vzporedno razvijajo tudi metodologije in koncepti projektnega menedžmenta, prilagojeni potrebam projektov razvoja MA in sorodnim projektom. Fizične omejitve mobilnih naprav – velikost zaslona, teža, kapaciteta spomina ipd. – zahtevajo, da imajo MA manjši obseg funkcij, kot to velja na primer za programske rešitve, namenjene osebnim računalnikom, zaradi česar je tudi razvoj novih aplikacij nekoliko poenostavljen. Prav kombinacija manjšega obsega novo razvitih funkcij in poenostavljen razvoj dajete posebne lastnosti projektom razvoja MA. Trajanje teh projektov je relativno kratko – razvoj lahko traja tudi samo nekaj tednov, najpogosteje nekaj mesecev, zelo redko pa projekti trajajo dlje kot leto dni. Projektni timi so zelo majhni in število sodelujočih na projektu v povprečju ne presega 10 članov (Cravens 2012, 41).

V preteklosti je razvoj MA najpogosteje temeljil na aplikaciji poznanih in uveljavljenih projektnih metodologij kot na primer RUP (orig. *Rational Unified Process*), PRINCE 2 (orig. *Projects in Controlled Environments*), SSADM (orig. *Structured systems analysis and design method*) in podobnih. Zaradi obsega in zahtevanih hitrosti izvajanja projektov razvoja MA, so prakse, opisane v omenjenih standardnih metodologijah, pri teh projektih skorajda neuporabne. Zahteve po hitrem in fleksibilnem razvoju MA vodijo v praksi do vse pogostejših aplikacij in hitrih adaptacij t. i. agilnih metodologij projektnega menedžmenta. Popularnost »agilnega« pristopa k razvoju in obvladovanju stohastičnih procesov, ki jih je zaradi same narave procesov nemogoče ali nesmiselno popolnoma načrtovati, in kateremu smo priča zadnjih približno deset let v pravzaprav vseh industrijskih panogah, je vodila tudi do razvoja številnih agilnih metodologij na področju projektnega menedžmenta (Vrečko 2011, 122).

V članku se osredotočamo na agilni metodologiji

Scrum in *Extreme Programming* (v nadaljevanju: *XP*). Obe metodologiji sta brezplačno dostopni in sta zelo dobro podprti s številnimi strokovnimi članki ter z literaturo. Metodologiji imata tudi svoje »skupnosti« –različne spletne forume, skupine in posameznike, ki so praviloma pripravljeni tudi svetovati in pomagati pri razvoju teh metodologij, glede na specifične zahteve posameznih projektov. Obstoj tovrstnih skupnosti in zagotavljanje podpore je moč zaznati tudi v Sloveniji.

Zaradi naštetih prednosti in velike kompatibilnosti metodologij *Scrum* in *XP* glede na posebnosti projektov razvoja MA, predstavljamo v članku model njune integracije kot primerne podlage za zagotavljanje učinkovitega razvoja MA, ki je preizkušen in tudi že večkrat apliciran na konkretnih projektih razvoja MA, s katerimi se ukvarjajo avtorji tega članka.

2. Predstavitev agilnih projektnih metodologij *Scrum* in *Extreme Programming*

2.1 *Scrum* projektna metodologij

Scrum je agilna projektna metodologija, ki se je prvič pojavila leta 1990 in je bila zasnovana za kompleksne projekte razvoja izdelkov. Posebnost *Scrum* projektne metodologije je, da projektnemu timu v času izvajanja razvoja zagotavlja visoko osredotočenost na problem in eliminacijo motenj projektnega tima, kar vodi do izjemne produktivnosti in učinkovitosti tima. Metodologija temelji na iterativnem pristopu, kar povečuje priložnosti za pridobivanje povratnih informacij o poteku procesa in o izmenjavi informacij med naročnikom projekta in projektnim timom. Vsaka iteracija zajema proces planiranja, izvedbe in kontrole. Proces učenja in razreševanja nejasnosti ali raznolikih interpretacij ciljev projekta ob sleherni iteraciji znižuje tveganja, povezana s pomanjkanjem vhodnih informacij.

Metodologija *Scrum* je široko zasnovana in opredeljuje: velikost in strukturo projektnega tima, vloge posameznikov v projektnem timu, organizacijska pravila, projektno dokumentacijo in posebne metodološke dogodke, ki se pojavljajo v življenjskem ciklu projekta (kot npr. dnevni sestanki, predstavitev rezultatov in podobno – več o tem v nadaljevanju). Pravila metodologije *Scrum* povezujejo člane projektnega tima, opredeljujejo njihove vloge v posameznih fazah življenjskega ciklusa projekta in ob omenjenih posebnih metodoloških dogodkih, s tem pa diktirajo odnose ter interakcije v projektnem timu.

Projektni tim po metodologiji *Scrum* sestavljajo: lastnik izdelka (orig. *Product Owner*), tim razvijalcev (orig. *Develoment team*) in t. i. vodja *Scruma* (orig. *Scrum Master*). Značilnost projektnih timov, oblikovanih po *Scrum* metodologiji, sta njihova interna organiziranost in interdisciplinarnost. Interna organiziranost timov pomeni, da sami opredeljujejo, kako najbolje opraviti delo v času projekta. Interdisciplinarni timi vsebujejo vso potrebno znanje, da uspešno zaključijo projekt, zato niso odvisni od zunanje pomoči. Slednje je eden od ključnih dejavnikov, ki pripomore k osredotočenosti na izvedbo nalog in reševanje izzivov, in je kot takšen posebnost metodologije *Scrum*.

Vloge in naloge članov projektnih timov po metodologiji *Scrum*:

- *Lastnik izdelka*. Skrbi, da izdelek dosega najvišjo možno dodano vrednost, ob tem pa skrbi tudi za optimalno delovanje tima razvijalcev. Lastnik izdelka nosi vso odgovornost za dokument opis izdelka (več o tem v poglavju 4.1). Po metodologiji *Scrum* je pomembno, da je lastnik izdelka samo ena oseba in ne menedžerski tim. Za uspešno izvajanje vloge lastnika izdelka mora ta imeti vsa pooblastila, da lahko sprejema ali zavrača spremembe izdelka v času razvoja.
- *Tim razvijalcev*. Sestavljajo ga profesionalci, ki imajo vso potrebno tehnično in drugo potrebno znanje za uresničitev razvoja v času iteracije (posamezna iteracija se po metodologiji *Scrum* imenuje sprint – več o tem v poglavju 4.2). V posamezni iteraciji oziroma razvojni fazi delo opravljajo samo člani tima.
- *Vodja Scruma*. Je del tima razvijalcev in skrbi, da razvojni tim spoštuje metodologijo *Scrum*. Sodeluje z lastnikom izdelka pri nastajanju in vzdrževanju dokumenta opisa izdelka. Je edini član tima razvijalcev, ki ima stik z zunanjo organizacijo, v kolikor je projekt del večje organizacije.
- V življenjskem ciklu projekta mora projektni tim skladno z metodologijo *Scrum* uspešno izvesti določene dogodke, katerih osnovni namen je zmanjšanje potrebe po vzpostavljanju formalnih oblik vodenja in poročanja. Vsi tovrstni dogodki so vnaprej opredeljeni in časovno omejeni. Potrebe po dodatnem sestajanju projektna tima so zmanjšane na minimum oziroma popolnoma eliminirane, kar ponovno zvišuje osredotočenost razvijalcev na izvajanje njihovih strokovnih nalog.
- Metodologija *Scrum* opredeljuje naslednje dogodke: *sprint*, sestanek, namenjen planiranju *sprint-a*, dnevni *Scrum sestanek*, predstavitev rezultatov *sprint-a* in pregled *sprint-a*. Vse navedene dogodke in način njihovega dejanskega izvajanja na primeru dveh projektov predstavljamo v nadaljevanju članka.
- Metodologija *Scrum* zvišuje produktivnost tudi z odstranjevanjem birokratskih zahtev, povezanih s projekti. Metodologija *Scrum* namesto obsežnih poročil zahteva od projektnega tima samo dva dokumenta:
  a) opis izdelka (orig. *Product Backlog*), ki je spisek vseh poslovnih in uporabniških funkcij, ki jih želimo doseči z izdelkom;
  b) opis sprinta (orig. *Sprint Backlog*), ki predstavlja spisek izbranih elementov iz opisa izdelkov, ki jih bomo razvijali v času sprinta (Schwaber in Sutherland 2011).

Prednosti *Scrum* metodologije

- Fokus – v času *sprinta* ni zunanjih motenj in projektni tim se lahko osredotoči samo na delo, določeno za posamičen *sprint*. Izrazit fokus izredno povečuje produktivnost tima.
- Agilnost – v času razvoja se lahko spreminjajo cilji in

zahteve projekta. Z vsako iteracijo je mogoče razširiti projekt, kar zagotavlja veliko fleksibilnost projektu in njegovemu izvajanju.
- Učeča se organizacija – ob zaključku vsake iteracije (*sprinta*) sledi pregled dela. Nova spoznanja in dobro prakso se *nadgrajuje* z vsako iteracijo.
- Kreativnost in inovativnost – metodologija *Scrum* s svojimi pravili spodbuja vse člane projektnega tima k soustvarjanju v času razvoja. Vsi člani tima aktivo iščejo rešitev skozi celotni potek razvoja izdelka.
- Motivacija – Člani razvojnega tima sami prevzamejo količino dela, ki ga bodo opravili v času *sprinta* ter sami predlagajo rešitve problema, ki jih bodo implementirali v času *sprinta*. Velja tudi izpostaviti, da *Scrum* načeloma ne dovoljuje prekoračevanje predvidenega časovnega obsega dela z opravljanjem t. i. nadur. Vsi ti pogoji zvišujejo osebno motivacijo sodelujočih na projektu.

Slabosti *Scrum* metodologije
- Izobraževanje projektnega tima o številnih pravilih in novih postopkih tako dela kot razmišljanja, ki so povezani z metodologijo *Scrum*, lahko močno zamaknejo pričetek projekta.
- V času sprinta je za organizacijo in izvedbo dela odgovoren tim razvijalcev. Timi, ki imajo malo izkušenj z metodologijo *Scrum* in s samoorganizacijo nasploh, bodo imeli velike težave. Odsotnost formalnega vodje, ki razdeli delo, je velik šok za novince *Scruma*.
- Tim razvijalcev mora interno imeti vso potrebno znanje, da uspešno izvede ves predviden razvoj v času *sprinta*. Zunanjo sodelovanje ni predvideno v metodologiji *Scrum*.

2.2 *Extreme Programming* projektna metodologija

Metodologija *Extreme Programming – XP* je velikokrat opisana kot projektna metodologija, čeprav bi jo dejansko bilo ustrezneje definirati kot razvojno metodologijo. Že ime metodologije razkriva, da je metodologija *XP* prvenstveno namenjena projektom razvoja programskih rešitev. Podobno kot *Scrum* tudi *XP* metodologija uporablja iterativni pristop v procesu razvoja izdelkov. Cilj *XP* metodologije je znižati visoke stroške kasnejših sprememb, do katerih pogosto prihaja v razvojnih projektih zaradi pomanjkanja vhodnih informacij ob zagonu njihovega izvajanja. Metodologija *XP* podrobno opredeljuje sestavo tima razvijalcev in različne vloge programerjev, ki jih izvajajo v času razvoja. *XP* opredeljuje faze procesa razvoja programskih rešitev in najboljše prakse za razvoj programskih rešitev v posameznih fazah. Posebnost metodologije *XP* je, da ob zelo konkretnih primerih razvoja programskih rešitev opredeljuje tudi vrednote, katerih se morajo držati razvijalci, ki delajo na projektih po *XP* metodologiji. Za razliko od *Scrum* metodologije, metodologija *XP* ne posveča posebne pozornosti organizacijskim pravilom ali strukturi projekta. Podrobno predeljuje le proces razvoja programskih rešitev.

Metodologijo *XP* lahko opredelimo kot enostavno metodologijo, zelo primerno za manjše projektne time in projekte, v katerih vse vhodne zahteve niso znane ali pa se bodo v času razvoja programske rešitve prav gotovo spreminjale. Metodologija *XP* je zato prava izbira za projekte razvoja programskih rešitev z nejasnimi izhodiščnimi zahtevami in za projekte razvoja programskih rešitev, kjer so spremembe pričakovane (Beck 1999).

Prednosti *XP* metodologije
- Agilnost. Tudi za to metodologijo velja, da se lahko v času razvoja spreminjajo cilji in zahteve projekta. Podobno kot pri *Scrum* metodologiji je tudi pri *XP* metodologiji z vsako iteracijo mogoče razširiti projekt, kar zagotavlja veliko fleksibilnost projektom in njihovemu izvajanju.
- Delujoči prototip. Metodologija *XP* zahteva delujočo programsko rešitev ob koncu vsake iteracije. Delujoč izdelek je po prvi iteraciji možno plasirati na tržišče, nakar se na podlagi povratnih informacij uporabnikov nenehno izboljšuje.
- Kakovost. Tehnike razvoja programskih rešitev, kot jih predvideva metodologija *XP*, predvsem zvišujejo kakovost in odstranjujejo možnosti napak v končni programski kodi, ki nastane v času projekta.
- Preglednost. Metodologija *XP* zagotavlja veliko preglednost nad projekti, kar olajša delo projektnemu menedžerju in vrhovnemu menedžmentu.
- Povratne informacije. Razvoj novega izdelka je vedno povezan z visokim tveganjem zavrnitve novitete s strani trga. Metodologija *XP* povečuje interakcijo izdelka s trgom in vodi nadaljnji razvoj na podlagi povratnih informacij bodočih kupcev. S tem se pomembno znižujejo tveganja zavrnitve izdelka s strani trga.

Slabosti *XP* metodologije
- Metodologija *XP* zahteva, da naročnik projekta nastopa kot član projektnega tima in je stalno dostopen vsem članom projektnega tima.
- Tehnike razvoja po metodologiji *XP* zahtevajo, da so vsi člani projektnega tima na skupni lokaciji. V primeru, da to ni izvedljivo, uporaba metodologije *XP* kot metodološke podlage za vodenje nekega projekta ni primerna.
- Tehnike razvoja programske kode, kot jih predvideva metodologija *XP*, so za mnoge programerje neobičajne in za nekatere celo kontroverzne. Zato lahko preteče veliko časa, da razvijalci sprejmejo postopke dela, ki jih zahteva metodologija *XP*.
- Metodologija *XP* je relativno mlada metodologija in je še vedno v svojem razvoju in testiranju, zato jo je težko brez zadržkov priporočati za uporabo pri kompleksnih projektih.
- Za uspešno izvedbo projektov po metodologiji *XP* mora imeti organizacija primerno organizacijsko kulturo. Tako je ta metodologija izjemno primerna za organizacije, ki vzpodbujajo kreativnost, svobodno razmišljanje in inovativnost svojih zaposlenih, ni pa primerna ob togih organizacijskih strukturah.

Metodologiji *Scrum* in *XP* imata kar nekaj skupnih lastnosti, hkrati pa ima vsaka tudi nekaj pomembnih posebnosti, zaradi katerih je uporabnost teh metod pri projektih razvoja MA enkrat bolj drugič manj primerna. Primerjavo metodologij *Scrum* in *XP* prikazujemo v tabeli 1, pri čemer za posamezen kriterij tudi podajamo oceno, katera metodologija je primernejša za projekte razvoja MA.

Tabela 1: Primerjava *Scrum* in *Extreme Programming (XP)* metodologij

Kriterij primerjave metodologij	Metodologija *Scrum*	Metodologija *XP*	Primernejša metodologija za razvoj MA
Organizacijska struktura projekta	Samoorganizacija vseh članov projektnega tima. Opredeljuje organizacijske dogodke in dokumentacijo.	Samoorganizacija vseh članov projektnega tima.	*Scrum*
Opredelitev projektnega tima	Projektni tim sestavljajo: • lastnik izdelka, • vodja Scruma, • razvijalci.	Projektni tim sestavljajo: • naročnik projekta, • razvijalci.	*Scrum*
Optimalna velikost projektnega tima	4 do 6 članov	8 do 10 članov	*Scrum* in *XP*
Opredelitev lokacije projekta	Priporoča skupno lokacijo projektnega tima. Omogoča tudi delo z različnih lokacij.	Zahteva delo na skupni lokaciji. Predpisana je celo velikost in postavitev skupne pisarne, kjer poteka razvoj.	*XP*
Časovna razsežnost projekta	Ni formalnega zaključka projekta. Projekt poteka, dokler je smiseln za naročnika. Razvoj je razdeljen na iteracije.	Ni formalnega zaključka projekta. Projekt poteka, dokler je smiseln za naročnika. Razvoj je razdeljen na iteracije.	Scrum in *XP*
Trajanje posameznih iteracij razvoja	1 do 4 tedne	1 do 6 mesecev	*Scrum*
Primernost okolja za uporabo metodologije	Vodenje vseh vrst kompleksnih projektov razvoja, tako v velikih kot v majhnih organizacijah.	Vodenje razvoja programskih rešitev v manjših organizacijah s primerno organizacijsko kulturo.	*Scrum*
Organizacijske vrednote	Ne predlaga spremembe kulture in vrednot organizacije.	Podrobno predpisuje organizacijske spremembe in spremembe vrednot, potrebnih za udejanje metodologije.	*XP*
Opredelitev procesa razvoja med iteracijami	Ne predlaga tehnik ali najboljših praks poteka razvoja izdelka.	Podrobno opisuje tehnike razvoja in dodaja primere dobre prakse razvoja programskih rešitev.	*XP*

Primerjava v tabeli 1 prikazuje, da niti metodologija *Scrum* niti *XP* nista sami po sebi popolnoma idealni za aplikacijo pri projektih razvoja MA. Zato je za tovrstne projekte smiselno razmišljati o razvoju nove metodologije oziroma o ustrezni adaptaciji in integraciji metodologij *Scrum* in *XP*.

3. Integracija agilnih projektnih metodologij *Scrum* in *Extreme Programming* za vodenje projektov razvoja mobilnih aplikacij

Na področju razvoja MA lahko opazimo pogosto aplikacijo metodologije *Scrum*, ki sicer postaja dnevna praksa velikega dela razvijalcev v Silicijevi dolini. Izbrana pravila metodologije *Scrum*, predvsem vezana na vprašanja organizacijske strukture projekta, bomo v nadaljevanju uporabili kot podlago za oblikovanje integriranega modela agilnih metodologij *Scrum* in *XP*.

V tabeli 2 prikazujemo sistem organiziranja in priporočene metodološke podlage pri izvajanju projektov razvoja MA. Poznavanje in razumevanje celotnega sistem organiziranja nam razkrije potrebe in izzive projektnega menedžmenta pri razvoju MA, predvsem na organizacijski in operativni ravni. Na te izzive se lahko po izkušnjah avtorjev članka pripravimo in zelo uspešno odgovorimo z integracijo metodologij *Scrum* in *XP*.

Na najvišji strateški ravni mora menedžment opredeliti poslovne cilje in vizijo projekta. Poslovni cilji vplivajo na vrsto MA, ki naj bi bila razvita. Poznamo tri vrste MA:

1. Aplikacije, prilagojene specifičnemu operacijskemu sistemu (OS); te omogočajo razvijalcem popoln dostop do vseh funkcionalnosti, ki jih ponujata tako naprava kot operacijski sistem.
2. Spletne aplikacije; gre za spletne strani, ki so posebej optimizirane za delovanje na mobilnih napravah. Prednost spletnih aplikacij je, da delujejo na vseh mobilnih operacijskih sistemih. Njihova največja pomanjkljivost je, da uporabniki dostopajo do njih s pomočjo brskalnika, kar je za mnoge uporabnike

Tabela 2: Sistem organiziranja in metodološka podlaga pri projektih razvoja MA

Raven delovanja	Naloge / Opravila	Priporočena metodološka podlaga
Strateška raven	Opredelitev poslovnih ciljev in vizije projekta. Izbira vrste razvijajoče MA (prilagojene OS / spletne / hibridne).	
Organizacijska raven	Oblikovanje projektnega tima, organizacijske strukture in pravil delovanja – izbor organizacijske projektne metodologije.	*Scrum*
Operativna raven	Razvijanje MA z vestnim in učinkovitim izvajanjem aktivnosti in doseganje opredeljenih dogodkov.	*Extreme Programming*
Raven nadzora kakovosti	Nadzor nad potekom razvoja in doseganja ustrezne kakovosti MA. Certifikacija MA za prodajo. Merjenje in analiziranje odziva uporabnikov.	

nezaželena uporabniška izkušnja.

3. Hibridne aplikacije – kombinacija predstavljenih specifičnih in spletnih MA; gre za spletne aplikacije, ki so obdane s kodo aplikacij, specifičnih vsakemu mobilnemu operacijskemu sistemu. Zaradi te kombinacije se obnašajo na vsakem operacijskem sistemu kot matična aplikacija, ki v ozadju zaganja spletno aplikacijo.

S sprejetjem poslovnih ciljev in vrsto MA, ki bo lahko zagotovila zastavljene poslovne cilje, se lahko pričneta oblikovati projektni tim in organizacijska struktura, v kateri bo tim deloval. Pri oblikovanju projektnega tima je potrebno upoštevati vse tehnične in poslovne izzive, ki jih predstavlja razvoj MA. Potrebno je sestaviti majhen, interdisciplinaren tim, ki ima možnost delovanja na skupni lokaciji. V tej fazi priporočamo uporabo *Scrum* metodologije kot ustrezne metodološke podlage. V kolikor oblikovan projektni tim ni seznanjen s *Scrum* metodologijo, jih je smiselno nemudoma pričeti izobraževati o pravilih in dogodkih, ki jih ta metodologija predpisuje. Zagon izvajanja projekta se izvede šele po tem, ko se tim in menedžment seznanita in poenotita o organizacijskih pravilih.

V času razvoja je potrebno skrbeti, da se dogodki in aktivnosti, ki jih zahtevajo pravila metodologije *Scrum*, izvajajo vestno in učinkovito. Metodologija *Scrum* namreč pozitivno vpliva na fokus in motivacijo projektnega tima, vendar pa po drugi strani ne vsebuje predlogov tehnik, primernih za razvoj dejanske kode MA. V ta namen je kot metodološko podlago smiselno uporabiti metodologijo *XP*, katera podrobno opisuje učinkovite agilne tehnike razvoja. Ker je primerna za majhne time in zahteva delo na skupni lokaciji, je ni težko vključiti v organizacijsko strukturo, ki jo zagotavlja *Scrum* metodologija.

Načeloma zgoraj predstavljene aktivnosti zadoščajo za uspešen zagon projekta razvoja MA, s čimer pa seveda še zdaleč ni zagotovljen končen uspeh projekta. Težko bi govorili o agilnem pristopu razvoja, če ne bi poskušali z vsak iteracijo zagotoviti delujočega prototipa izdelka. Zato na najnižjo raven postavljamo nadzor in kakovost aplikacije, ki jo mora projektni menedžer upoštevati pri razvoju. Predstavlja informacijsko raven, ki posreduje povratne informacije uporabnikov. Na podlagi le-teh morata projektni menedžment in naročnik z vsako naslednjo iteracijo odpraviti napake in napačne poslovne domneve, ki so se pojavile pri razvoju MA.

4. Vodenje projektov razvoja MA s kombinacijo metodologij *Scrum* in *XP*

Aplikacija integriranih metodologij *Scrum* in *XP*, kot je predstavljena v tem članku, je bila izvedena in preizkušena na dveh uspešno izvedenih projektih razvoja MA. Avtorji članka integrirano metodologijo uporabljajo nekoliko manj kot eno leto. Ob tem je potrebno poudariti, da projektov razvoja MA niso nikoli poskusili voditi s pomočjo tradicionalnih projektnih metodologij, zato niti ne morejo niti ne poskušajo trditi, da tovrstnih projektov ne bi bilo mogoče uspešno izvajati tudi s kakšnimi drugimi metodologijami.

V času aplikacije integrirane metodologije na dveh projektih razvoja MA so bile v procesu razvijanja MA preizkušene različne kombinacije integriranja metodologij *Scrum* in *XP*. Nekateri poskusi so bili bolj, spet drugi pa manj uspešni. Na podlagi spoznanj in zaznanih številnih napak, ki so bile v veliki meri posledica pomanjkanja metodoloških izkušenj na tovrstnih projektih, je bila opredeljena in za nadaljnje tovrstne projekte izbrana kombinacija metodologij *Scrum* in *XP*, ki se je izkazala za uspešno.

V nadaljevanju je podrobneje predstavljeno, kako so bili izvajani aktivnosti in dogodki, kjer je kot podlaga bila uporabljena metodologija *Scrum* in kako so bile implementirane tehnike razvoja kode, metodološko podprte z metodologijo *XP*.

4.1 Priprava *opisa izdelka*

Opis izdelka je osnovni in verjetno najpomembnejši dokument v procesu izvajanja projekta po *Scrum* metodologiji in je tudi pogoj za začetek izvajanja projekta. Opis izdelka je spisek t. i. »zgodb«, ki jih pripravi lastnik izdelka. Vsaka zgodba predstavlja opis funkcionalnosti izdelka, ki jo je potrebno razviti v času projekta. Metodologija *XP* prav tako zahteva od naročnika, da pripravi zgodbe, vendar definira zgodbe kot »uporabniške zgodbe«. Uporabniške zgodbe predstavljajo celoten scenarij, kako bodo uporabniki uporabljali določeno funkcijo izdelka. Uporabniške zgodbe zahtevajo več konteksta in pripomorejo k boljšemu razumevanju posameznih funkcij izdelka.

Avtorji članka so pri pripravi zgodb razširili metodologijo *Scrum* z uporabniškimi zgodbami, kot jih

koncipira *XP*. Primer *opisa izdelka* je prikazan v tabeli 3. Pri delu na projektih so avtorji članka tabelo zasnovali kot enostaven Excelov dokument, do katerega so z uporabo brezplačne storitve Dropbox (Drobox omogoča, da uporabniki med seboj delijo slike, datoteke, mape in podobno) imeli zagotovljen dostop vsi člani projektnega tima.

Tabela 3: Primer opisa izdelka

ID	Ime zgodbe	Pomembnost	Čas	Kakšen bo demo ob koncu sprinta?	Opombe
1	Urejanje lastnega profila	30	8	V nastavitvah odpri lasten profil in spremeni prikazano sliko in ostala polja z osebnimi podatki.	Prikazane slike so dimenzije 50 X 120 px. Polja z osebnimi inf. še niso oblikovana.

Vir: lasten

Vsaka zgodba v tabeli ima svojo t.im. kartico (kartice predstavljajo način vizualizacije posameznih uporabniških zgodb). Nekoliko podrobneje so predstavljene v nadaljevanju., katera vsebuje podroben opis uporabniške zgodbe. Po metodologiji *Scrum* je dokument opisa izdelka last naročnika in ga kot takega lahko spreminja samo naročnik. Ob delu na projektih smo sami opazili, da je veliko bolje, če imajo vsi člani tima pravico spreminjati ta dokument, vendar ne brez vednosti naročnika projekta. Zato smo vzpostavili način delovanja, da je vse spremembe formalno in končno potrdil ali zavrgel naročnik projekta. Posebno pozornost je potrebno nameniti zagotovitvi enotnega razumevanja zgodb. V ta namen je lahko posebej koristno polje »opombe« v *opisu izdelka*, kjer imajo razvijalci možnost natančneje opredeliti posamezne zgodbe. S tem se pogosto prihrani veliko časa v kasnejših fazah izvajanja projekta, ko ni potrebnih toliko popravkov in spreminjanja neustreznih rešitev, do katerih pride zaradi neusklajenosti med naročnikom projekta in timom razvijalcev.

Dobro pripravljen *opis izdelka* zato nikakor ne sme predstavljati samo spiska potrebnih izvedbenih aktivnosti. Zgodbe morajo opisovati poslovne cilje in naročnikova naloga je, da natančno predstavi poslovne cilje s pomočjo scenarijev predvidene uporabe končanega izdelka. Tehnično plat realizacije projekta mora naročnik prepustiti razvijalcem, kar zna pogosto predstavljati poseben izziv za naročnike, ki imajo sicer sami relativno dobro tehnično znanje in izkušenost na tem področju.

4.2 Planiranje *sprinta*

Planiranje sprinta je najpomembnejši dogodek, ki ga predvideva metodologija *Scrum*. Ustrezna organizacija in izvedba tega dogodka vplivajo na uspešnost dela v času naslednjega sprinta in s tem na uspešnost celotnega projekta. Slaba izvedba sestanka, namenjenega planiranju naslednjega *sprinta*, vpliva na potreben čas izvajanja razvoja med sprintom, kar lahko predstavlja tudi več tednov dela.

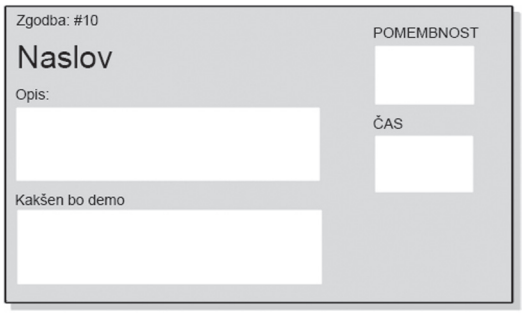
Večji del sestanka je namenjen ocenjevanju uporabniških zgodb, katerih avtor je lastnik izdelka. Vsaki zgodbi določi lastnik izdelka t. i. *domet* in *oceno pomembnosti*. Z dometom se opredeli stopnja dokončanosti uporabniških zgodb, vključenih v *sprint* – večji kot je domet, več funkcionalnosti bo implementiranih z vsako zgodbo. Ocena pomembnosti vpliva na čas, kdaj bo tim implementiral določeno uporabniško zgodbo – višja kot je ocena pomembnosti, prej v življenjskem ciklu projekta bo uporabniška zgodba implementirana. Vsaki uporabniški zgodbi doda razvojni tim še *oceno časa*, potrebnega za implementacijo zgodbe. Dinamika spreminjanja ocen uporabniškim zgodbam s strani lastnika izdelka in tima razvijalcev je ključna za uspešno planiranje *sprinta*.

V času sestanka oziroma *planiranju sprinta* se projektni tim pogaja z lastnikom izdelka, katere zgodbe bodo vključene v naslednji sprint in katere bodo prepuščene kasnejšim sprintom. Do pogajanj prihaja, ker tako *Scrum* kot *XP* metodologiji predlagata, da si razvijalci sami naložijo količino dela. S tem se sicer pri razvijalcih praviloma doseže večjo mero odgovornosti za kvalitetno in pravočasno realizacijo dela, povzroča pa pogosto konflikte z lastnikom izdelka. Lastnik namreč želi vključiti čim več zgodb v naslednji *sprint*, medtem ko razvijalci željo vključiti tolikšno število zgodb, ki še ne predstavlja prevelike obremenitve in stresa. Ko se lastnik in razvijalci uskladijo, se uporabniške zgodbe, ki so bile izbrane za naslednji *sprint*, umaknejo iz opisa izdelka in se dodajo v opis sprinta. *Opis sprinta* je ključni dokument v času sprinta.

V zvezi s sestankom, namenjenem planiranju naslednjega *sprinta*, je ključnega pomena napotilo metodologije *Scrum*, da naj bo sestanek izveden v metodološko opredeljenem časovnem okvirju. Metodologija *Scrum* predvideva osemurni sestanek v primeru, da *sprint* traja mesec dni, medtem ko je za krajše *sprinte* potrebno pripraviti krajše sestanke. Pogosto se dogaja, da tim razvijalcev in lastnik izdelka (t. i. *Scrum* tim) ne opravi potrebnega dela v predvidenem času in nato sestanek ali podaljša ali ponovi naslednji dan. Podaljšanje sestanka se je izkazalo kot nesmiselno, saj v primeru, da člani tima niso zmožni doreči vseh detajlov sprinta v prvih osmih urah, tega tudi ne bodo zmožni v naslednjih urah, ko nastopi utrujenost. Slednja pogosto povzroča še dodatna in nepotrebna trenja ter konflikte med člani tima, kar lahko predstavlja dodatno oviro v nadaljevanju projekta. Izkušnje avtorjev članka kažejo, da je pri za projekt manj pomembnih sprintih smiselno sestanek enostavno prekiniti. Čeprav bo zaradi tega razvoj v času *sprint* nekoliko počasnejši in manj kvaliteten, bo prekinitev predstavljala dobro lekcijo članom tima, kako delovati in sodelovati pri naslednjih *sprintih*. Dobri sodelavci niso zadovoljni s slabo opravljenim delom, zato se bodo pri naslednjem planiranju *sprinta* bolj potrudili, da bo sestanek pravočasno zaključen. Za pomembnejše *sprinte* – pri katerih bi zaradi slabega planiranja lahko prihajalo do pomembnih sistemskih napak in katerih odpravljanje bi precej dvignilo stroške razvoja – je smiselno ponoviti sestanek.

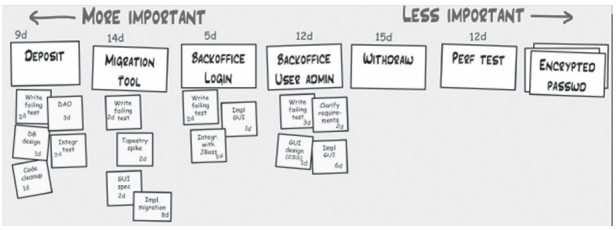
V času pogajanja glede uporabniških zgodb se je kot izredno dobra praksa izkazala uporaba posebnih kartic (slika 2), ki pomagajo pretvoriti abstraktno idejo uporabniške zgodbe v nekaj oprijemljivega. Prednosti, ki jih prinese delo s takšnimi karticami so (Kniberg 2007, 40):

- Člani tima morajo vstati in se gibati po prostoru, kadar jih željo preurediti. Sestanek je s tem bolj dinamičen in člani tima imajo več energije za nove ideje.
- Vsi člani tima so del interaktivnega procesa in ne samo tisti posameznik, ki sedi pred računalnikom in spreminja tabelo.
- Zagotovljena je večja preglednost nad celotnim projektom.
- Spreminjanje pomembnosti uporabniških zgodb je enostavno.



Slika 1: Primer kartice kot podloge za zapis uporabniške zgodbe (Vir: lasten)

Kartice pripravi vodja *Scrum*a pred sestankom, tako da tim s tem ne izgublja časa. Izpolnjene kartice so nato razvrščene na steni ali tabli na osnovi ocenjenih pomembnosti posamezne uporabniške zgodbe. Kartice z nižjo oceno pomembnosti uporabniške zgodbe pomaknemo bolj desno, tiste z višjo oceno pa bolj levo, kot to prikazuje slika 3.



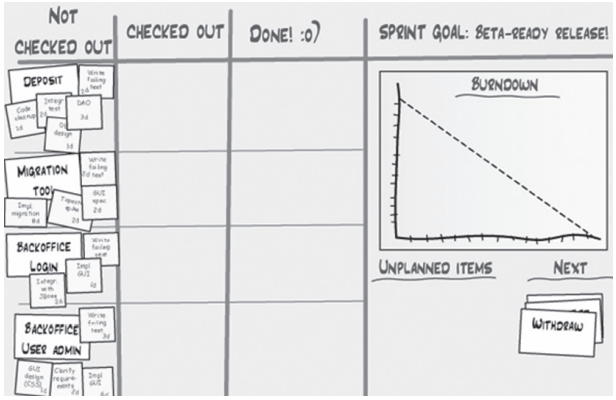
Slika 2: Prikaz razvrstite kartic gleda na njihovo oceno pomembnosti (Vir: Kniberg, 2007)

Tudi za določanje časa, potrebnega za implementacijo posamezne zgodbe, priporočamo tehniko vizualizacije. Ta popestri proces, imenuje pa se časovni poker. Vsak član ekipe prejme set trinajstih kart. Pri ocenjevanju zgodb vsak član izbere karto, ki po njegovi oceni predstavlja čas njene implementacije. Na kartah čas predstavljajo točke, kot to predvideva metodologija *Scrum*. Vsi člani sočasno razkrijejo svojo karto, s čimer je zagotovljeno, da vsak član tima poda svojo lastno oceno in pri tem ni pod vplivom drugih članov tima. V primeru, da je med ocenami velika razlika, se je potrebno o zgodbi in ocenah dodatno pogovoriti. Velika odstopanja namreč nakazujejo, da nekateri člani ne razumejo uporabniške zgodbe. Ko so nejasnosti odpravljene, se partija pokra ponovi. Ponavljanje igranja poteka vse dokler vsi člani ne izberejo iste časovne ocene.

Slika 3: Prikaz kart tehnike *časovni poker* (Vir: http://www.planningpoker.com/)

4.3 Izdelava opisa sprinta

Opis sprinta (orig. *Sprint Backlog*) je dokument, ki vsebuje vse uporabniške zgodbe, vključene v *sprint*. V *opisu sprinta* zgodbe razčlenimo na naloge. Tradicionalno je *opis sprinta*, tako kot *opis izdelka*, nek dokument, do katerega morajo imeti dostop vsi člani tima. Predstavlja skupno vizijo *sprinta* in predstavlja referenčni dokument za vse razvijalce, vključene v *sprint*. Člani tima ga tako morajo imeti vedno na očeh. Opazili smo, da tega ni mogoče doseči z elektronskim ali papirnatim formatom. V trenutku, ko člani tima razvijalcev nehajo dnevno spremljati *opis sprinta*, se pričnejo pojavljati odstopanja od plana. Veliko boljše rezultate smo dosegali z uporabo še ene tehnike vizualizacije – s spremembo *opisa sprinta* v enostavno tabelo, na katero smo pritrjevali kartice, ki smo jih uporabljali pri *planiranju sprinta* (koncept je prikazan na sliki 4).



Slika 4: Prikaz opisa sprinta v obliki tabele (Vir: Kniberg, 2007)

Na prosto steno, tablo ali velik kolaž narišemo razpredelnico s štirimi stolpiči. Vsaka kartica dobi svojo vrstico. Kartice razvrstimo po pomembnosti, in sicer tako, da so najpomembnejše uporabniške zgodbe na vrhu. Pod rubriko »Next« pritrdimo vse odvečne kartice, ki niso bile vključene v sprint. Zgodbe, ki jih pričnemo izvajati, pomaknemo za en stolpec v desno. Ko jih zaključimo, jih pomaknemo v zadnji stolpec.

V zadnji stolpec narišemo t. i. »Burn-down chart«. Graf na horizontalni osi prikazuje čas izražen v dnevih, na vertikalni osi pa seštevek časa vseh zgodb, ki jih je potrebno zaključiti v *sprintu*. Po vsakem dnevnem *Scrum* sestanku narišemo novo točko na graf. Graf prikazuje trend razvoja. Je dober vizualni pripomoček, ki prikazuje, ali bomo v času sprinta zaključili predvidene zgodbe. Kadar delo poteka hitreje, kot smo predvidevali, mora ekipa sprejeti dodatne zgodbe v sprint, sicer se delo proti koncu sprinta upočasni. V času razvoja se pojavljajo nepredvidene dodatne zahteve, za katere se pripravijo nove uporabniške zgodbe.

Dinamika ekipe se z uporabo predstavljene metodologije in tehnik občutno poveča. V prostoru naj obstaja centralna točka, kjer se ljudje zbirajo, debatirajo o novih idejah in nalogah, se pogovarjajo o možnih implementacijah in podobno, s čimer se občutna izboljša komunikacija celotnega tima.

4.4 Izvajanje dnevnega *Scrum sestanka*

Metodologija *Scrum* opredeljuje zahtevo po vsakodnevnem izvajanju 15-minutnega sestanka, t. i. *Scrum sestanka*. Gre za pomemben dogodek, ki naj vsak dan poteka ob istem času in v istem prostoru, s čimer dnevni *Scrum sestanek* kmalu postane običajna rutina tima. Poteka naj pred tablo ali steno, na kateri je opis *sprinta*. Tudi v tem primeru vizualizacijski pripomočki pomagajo za izboljšanje komunikacije v timu.

Namen *Scrum sestanka* je, da vsak član tima odgovori na tri vprašanja:

1. Kaj si naredil od zadnjega *Scrum sestanka*?
2. S katerimi ovirami si se srečal?
3. Kaj boš naredil do naslednjega *Scrum sestanka*?

Člani tima naj ob podajanju odgovorov vključijo kartice na tabli ali steni. Zaključene uporabniške zgodbe naj premaknejo v ustrezni stolpec, na zgodbe, katere bodo pričeli implementirati, pa naj napišejo svoje ime in jih premaknejo v ustrezni stolpec. S tem, ko napišejo svojo ime, sprejmejo odgovornost za določeno zgodbo, kar pozitivno vpliva na motiviranost za uspešno izvedbo. Na koncu sestanka tim vriše novo točko v graf, ki prikazuje napredek projekta. Priporočljivo je, da vsak dan graf posodobi drug član tima, s čimer je doseženo, da se vsi člani tima zavedajo pomembnosti pravočasne izvedbe.

4.5 Vodenje proces razvoja mobilnih aplikacij

Metodologija *Scrum* prepušča celotno izvedbo projekta timu. Obe predstavljeni agilni metodologiji zahtevata, da se menedžment nekoliko distancira od razvoja, kar pa ne pomeni, da za projektne menedžerje ni prostora v agilnih projektih. Po metodologiji *Scrum* lahko projektni menedžerji nastopajo v vlogi lastnika izdelka, ki pa ni enaka vlogi naročnika (velja pa dodati, da lahko naročnik prevzame tudi vlogo lastnika). V primeru, da imajo projektni menedžerji dovolj tehničnega znanja, lahko nastopijo tudi v vlogi *vodje scrum*a.

Tudi metodologija *XP* prepušča vso delo v času razvoja programerjem. Za razliko od metodologije *Scrum*, predlaga *XP* zelo koristne tehnike razvoja programskih rešitev. Posebnost metodologije *XP* je tudi v tem, da v sklopu metodologije predpisuje vrednote, po katerih bi se moral uspešen razvojni tim ravnati. Vloga projektnega menedžerja na projektu, ki uporablja metodologijo *XP*, je, da izobražuje razvijalce o različnih tehnikah *XP* in pomaga celotnemu timu, da se poenoti z vrednotami metodologije *XP*. Projektni menedžerji morajo ob uporabi agilnih metodoloških pristopov na projektih nastopiti v mentorski vlogi. Projektni menedžer je zadolžen za stalno izobraževanje tima, podajanje nasvetov in izobraževanja tako članom tima, kot vodji *Scrum*a in celo lastniku izdelka, v kolikor sam ne nastopa v tej vlogi. Metodologija *XP* zahteva od udeležencev projekta, da se držijo štirih vrednot:

- komuniciranje,
- enostavnost,
- zagotavljanje povratnih informacij,
- pogumnost.

Bolj konkretno pa *XP* predpisuje delavne procese za razvoj programskih rešitev. Produktivnost metodologije izhaja iz sinergijskih učinkov, ki nastajajo pri uporabi vseh tehnik razvoja hkrati. Potrebno pa je opozoriti, da bodo neizkušene ekipe imele precej težav pri uvajanju nekaterih *XP* tehnik. Ob tem se morda lahko pojavi dilema, ali je sploh smiselno vpeljevati tovrstne procese dela, če zagotavljajo uspeh le, kadar uporabljamo vse tehnike hkrati. Avtorji članka smo opazili, da produktivnost naraste tudi, če ne vpeljemo vseh tehnik hkrati. Razvoj je še vedno bolj strukturiran in pregleden, zmanjša se število napak, kvaliteta razvite MA pa močno naraste.

V tem članku bomo samo omenili tehnike razvoja po metodologiji *XP*, saj bi predstavitev njihovega vpliva na projekt in postopkov, kako jih vključimo v projekte razvoja, prekoračila namen in obseg tega članka (sicer pa so vse tehnike zelo podrobno predstavljene v okviru agilne metodologije *XP*). Naj dodamo, da zelo priporočamo projektni menedžerjem, ki bi želeli voditi projekte razvoja programskih rešitev s pomočjo agilne metodologije *XP*, da se poglobijo v posamezne tehnike, saj je mentoriranje tima pravzaprav nemogoče, v kolikor projektni menedžer sam ne razume tehnik razvoja.

Metodologija *XP* predlaga uporabo naslednjih tehnik razvoja programskih rešitev, ki jih lahko uspešno uporabimo tudi na projektih razvoja MA:

1. Celovit tim
2. Testiranje s končnimi uporabniki
3. Številne delujoče verzije
4. Enostavni dizajn in arhitektura
5. Programiranje v parih
6. Testno usmerjen razvoj
7. Optimizacija kode (*Refactoring*)
8. Skupno lastništvo kode
9. Upoštevanje standardov kodiranja
10. Uporaba metafor
11. Vzdržljivi ritem razvoja.

Uporabo *XP* tehnik razvoja priporočamo vselej, ko je opazen počasen, neučinkovit in/ali nekakovosten razvoj MA. V našem primeru smo *XP* tehnike vpeljevali

postopoma. Nov proces dela smo pričeli testirati šele tedaj, ko je tim dobro sprejel prejšnjo novost, ki smo jo uvajali. Hkratno uvajanje številnih novosti v procesu dela lahko povzroči, da se razvijalci pričnejo ukvarjati več s procesom dela, kot z razvojem. Najzahtevnejša tehnika med naštetimi je zagotovo testno usmerjen razvoj. V primeru, da programerji nimajo predhodnih izkušenj s tovrstnim razvojem in tim nima dostopa do nekoga, ki bi lahko prenesel znanje o testno usmerjenem razvoju na ekipo, je implementacija tovrstnega procesa dela skorajda nemogoča. Testno usmerjen razvoj je več kot le tehnika, je način razmišljanja. Koncept prinaša očitne prednosti in ga predlagamo vsem timom, ki imajo potrebno znanje, da uresničijo testno usmerjen razvoj.

4.6 Izvedba *zaključnega dema*

Ob zaključku *sprinta*, še posebej pri prvih nekaj *sprintih*, se lahko zgodi, da končni rezultat ni zelo zanimiv. V teh primerih se lahko pojavi vprašanje smiselnosti priprave *zaključnega dema*. Ne glede na rezultat *sprinta*, je vedno potrebno izvesti *zaključni demo* iz naslednjih razlogov (Kniberg 2007, 75):

- Tim ob tej priložnosti prejme zasluženo pohvalo, kar pozitivno vpliva na motivacijo in moralo za naslednji *sprint*.
- Drugi timi morajo biti seznanjeni z rezultatom *sprinta*. V primeru, da uporabljamo metodologijo *Scrum* in razvijamo izdelek z več kot enim timom, je lahko zelo koristno, da drugi člani timov vidijo končni rezultat vsakega *sprinta*.
- Zaključni demo običajno izzove neprecenljive povratne informacije.
- Predstavitev je družabni dogodek, kjer ponudimo timu možnost, da se sreča z novimi ljudmi, ki lahko ponudijo drugo perspektivo k rešitvi nekega kompleksne problema.
- *Zaključni demo* zagotavlja delujočo verzijo izdelka. Pogosto so namreč projekti razvoja programskih rešitev skorajda končani, izvedba zadnjih potrebnih popravkov pa je preložena v kasnejše faze življenjskega ciklusa projekta. To je seveda v nasprotju z metodologijo *Scrum* in ni v nikakršno korist naročniku, saj skorajda končan izdelek deluje povsem enako kot povsem nedokončan izdelek – torej ne deluje.

Po drugi strani pa je *zaključni demo* lahko tudi zelo učinkovita »kazen« za slabo izvedene *sprinte*. Kadar ne dokončamo planiranega dela v času *sprinta*, ne smemo odpovedati zaključne predstavitve *sprinta*. Slaba predstavitev je neprijeten dogodek za tim in člani tima takšne izkušnje ne bodo želeli ponoviti, zato bodo v naslednjem sprintu veliko bolj motivirani, da uspešno zaključijo planirano delo. *Zaključni demo* mora biti pomemben dogodek, do katerega mora tim čutiti odgovornost. Zato naj bo predstavitev javna in naj bodo na njo povabljeni vsi, za projekt in za tim pomembni ljudje.

4.7 Pregled *sprinta*

Ob zaključku vsakega *sprinta* je potrebno izvesti pregled *sprinta*. Sestanek je namenjen pregledu dogodkov, ki so se zgodili med *sprintom*. Tako člani tima prenašajo dobre prakse in znanje v naslednje *sprinte*. Sestanek je namenjen oblikovanju predlogov izboljšave procesa organiziranja, vodenja in izvajanja prihajajočih sprintov, ki jih poskušamo uvesti v naslednjem *sprintu*.

Mnogi trdijo, da je pregled ob koncu *sprinta* drugi najpomembnejši dogodek metodologije *Scrum*. Pregled *sprinta* daje edinstveno priložnost, da projektni tim skupaj poišče rešitve za težave, ki so se pojavljale v času razvoja in jih tako sprejme kot lastne rešitve. Nobena dobra rešitev problema ne bo tako učinkovita kot tista, za katero člani timi verjamejo, da so jo predlagali sami, saj bodo do lastnih rešitev čutili večjo odgovornost.

5. Sklepna spoznanja in priporočila

Številne prednosti in kompatibilnost metodologij *Scrum* in *XP* s posebnostmi projektov razvoja MA, hkrati pa njune določene pomanjkljivosti, so bile osnova za zamisel o njujem prilagajanju, predvsem pa integriranju. Tako integrirano metodologijo so avtorji tega članka večkrat aplicirati na konkretnih projektih razvoja MA in samo metodologijo ob tem stalno izpopolnjevali. Ob tem je potrebno poudariti, da namen članka ni zagovarjanje stališča, da je predstavljena metodologija edina primerna metodološka podlaga za tovrstne projekte, zato tudi ne analizira ali prikazuje drugih možnih metodologij.

Članek je pripravljen na način, da bi bralcem na kar najbolj enostaven in hiter način podal napotke za uporabo predstavljene metodologije v praksi, bodisi na projektih razvoja MA ali pa na drugih projektih podobnih karakteristik. Po uvodnem delu v obe metodologiji – *Scrum* in *XP* – v katerem so kratko predstavljene njune posebnosti in terminologija, članek prikazuje konkretne procesne korake razvoja MA in možnost uporabe metodoloških podlag *Scrum* in *XP* v tem procesu. Članek je dopolnjen tudi s slikami uporabljanih metodoloških pripomočkov, kar naj bralcem dodatno olajša aplikacijo predstavljene metodologije.

Viri in literatura

1. *Beck, K. 1999. Extreme Programming Explained: embrace change. Addison-Wesley: Boston.*
2. *Cravens, A. 2012. A demographic and business model analysis of today's app developer. ZDA: GigaOm.*
3. *Kniberg, H. 2007. Scrum and XP from the Trenches. ZDA: C4Media.*
4. *Schwaber K., Sutherland J. 2011. Scrum Handbook. Somerville: Scrum Training Institute.*
5. *Vrečko I. 2011. Obvladovanje strateških kriz z uporabo projektnega menedžmenta kot celovit invencijsko-inovacijski proces: doktorska disertacija. Maribor: I. Vrečko, Ekonomsko-poslovna fakulteta. http://dkum.uni-mb.si/Dokument.php?id=23821.*

Aljaž Daković, dipl.ekon.(UN), rojen 15.03.1988 v Mariboru je šolanje sicer zaključil na Ekonomsko-poslovni fakulteti, vendar ga njegovi interesi in neformalna izobrazba kot razvijalca spletnih aplikacij postavljata na križišče med poslovnimi vedami in informacijsko tehnologijo. Njegov močan podjetniški duh ga je vodil k ustanovitev prvega podjetja ob dopolnitvi polnoletnosti. Podjetje je ponujalo razvoj spletnih strani za mala podjetja. Sočasno vodenje podjetja in opravljanje študija je bila vzrok, da so se vrata prvega podjetniškega poskusa zaprla po dobrem letu dni. Od leta 2009 do začetka leta 2013 je prevzel operativno vodenje programa zbiranja sredstev »Starši otrok sveta« v nevladni organizaciji UNICEF. V večjih slovenskih mestih je organiziral »fundraising« ekipe in aktivnosti. Naloge so vključevale vodenje timov do 24 zaposlenih in pomoč pri organiziranju različnih marketinških ter P.R. kampanj. S sodelovanjem na številnih nevladnih projektih je pridobil veliko izkušenj z organiziranjem in realiziranem projektov in neprecenljive vodstvene izkušnje. Ljubezen do informacijske tehnologije in podjetniška usmerjenost sta bila vzrok za odhod iz organizacije UNICEF. Kmalu za tem se je s timom z imenom »Jumbooz« uvrstil na regionalne finale Microsoftovega inovacijskega tekmovanja Imagine Cup 2013. Po tekmovanju se je pridružil razvoju dveh mobilnih aplikacij, z namenom uvajanja agilnih projektnih metodologij za hitrejši in bolj učinkovit razvoj.

Dr. Igor Vrečko, univ. dipl. gosp. inž., je docent za področje »management poslovanja« na Ekonomsko-poslovni fakulteti Univerze v Mariboru. Središče njegovega strokovnega in raziskovalnega dela je področje projektnega menedžmenta in integracija projektnega s strateškim in kriznim menedžmentom. Ima bogate praktične izkušnje, pridobljene v svetovalnih projektih za številna domača in tuja podjetja. Je predstojnik Inštituta za projektni management na Ekonomsko-poslovni fakulteti Maribor, direktor mednarodnega programa certificiranja projektnih menedžerjev v Sloveniji, podpredsednik Slovenskega združenja za projektni management, vodja programa usposabljanja s področja projektnega menedžmenta na Centru za poslovno usposabljanje pri GZS, presojevalec usposobljenosti kandidatov za pridobitev nacionalne poklicne kvalifikacije za poklic »vodja projekta« in »vodja projektne naloge«, član nacionalne komisije za inovacije – strokovnjak za presojo tehnološko-podjetniških vidikov, nosilec mednarodnega certifikata »IPMA Certified Senior Project Manager« ter glavni in odgovorni urednik revije Naše gospodarstvo/Our Economy.