**UNIVERZA V LJUBLJANI**
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

**Matic Kašnik**

# UPORABA AGILNIH METOD PRI RAZVOJU PROGRAMSKE OPREME V SLOVENSKIH START-UP PODJETJIH

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Mentor: doc. dr. Mojca Ciglarič
Ljubljana, 2014

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00506 / 2013
Datum: 15.4.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MATIC KAŠNIK**

Naslov: **UPORABA AGILNIH METOD PRI RAZVOJU PROGRAMSKE OPREME V SLOVENSKIH START-UP PODJETJIH**
**AGILE METODOLOGIES USE FOR SOFTWARE DEVELOPMENT IN SLOVENIAN START-UPS**

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Primerjajte tradicionalni in agilni razvoj programske opreme. Raziščite pomembnejše agilne metodologije in jih tudi podrobneje opišite. Osredotočite se na prednosti, ki jih prinaša njihova raba v majhnih razvojnih skupinah. Zasnujte raziskavo, ki bo osvetlila različne vidike uvajanja, uporabe in koristi agilnih metodologij v slovenskih start up podjetjih. Pripravite strukturirano anketo in pristopite z njo do izbranih start-up podjetij. Odgovore analizirajte in v razpravi pojasnite, kakšno je stanje glede uporabe agilnih metodologij v Sloveniji, primerjajte stanje s podobnimi raziskavami iz tujine in skušajte podati tudi vzroke za takšno stanje.

Mentor:
M. Ciglarič
doc. dr. Mojca Ciglarič

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Nikolaj Zimic

# Izjava o avtorstvu diplomskega dela

Spodaj podpisani Matic Kašnik, z vpisno številko 63070044, sem avtor diplomskega dela z naslovom:

Uporaba agilnih metod razvoja programske opreme v slovenskih start-up podjetjih

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom doc. dr. Mojce Ciglarič,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne _____ Podpis avtorja: _____

# KAZALO

# ZAHVALA

Rad bi se zahvalil mentorici diplomskega dela doc. dr. Mojci Ciglarič za podporo ter dobro voljo in vsem ostalim, ki so pripomogli h nastanku tega diplomskega dela.

# POVZETEK

Diplomsko delo predstavi in opiše agilne metodologije, ki jih uporabljajo slovenska tehnološka start-up podjetja. Opisane so osnovne značilnosti, prednosti in slabosti uporabe posameznih metodologij kot so Scrum, ekstremno programiranje, vitko podjetništvo in metoda Kanban. Naredili smo pregled uporabe agilnih metodologij v slovenskih tehnoloških start-up podjetjih. Predvsem smo se osredotočili na vrsto orodij, tehnik in pristopov, ki jih razvojne ekipe uporabljajo ter kje vidijo največje prednosti in slabosti pri uporabi agilnih metodologij za razvoj produktov.

KLJUČNE BESEDE:
Agilne metodologije, start-up, vitko podjetništvo, Scrum, Kanban

# ABSTRACT

The thesis presents and describes the use of agile methodologies which are being used by Slovenian tech start-up companies. It describes the basic features, advantages and disadvantages of various agile methodologies such as Scrum, extreme programming, lean start-up and method Kanban. We made a review of the most used agile methodologies in Slovenian technology start-up companies. In particular, we focus on variety of tools, technics and approaches used by the development teams and where do they see the greatest strengths and weaknesses in the use of agile methodologies for product development.

KEY WORDS:
Agile methodologies, start-up, lean start-up, Scrum, Kanban

# 1 UVOD

Vse več posameznikov ter projektnih skupin se zaradi ekonomskih razmer in drugih razlogov na slovenskem trgu odloča oditi na svojo podjetniško pot. Zaradi razvoja informacijskih tehnologij se vse več izmed njih odloča prav za pot, usmerjeno v razvoj aplikacij programske opreme na področju mobilnih in spletnih tehnologij. Samo v letu 2012 je Slovenski podjetniški sklad v razpisu ponudil skoraj 60 milijonov evrov za razvoj tehnološko inovativnih projektov, namenjenih 181 start-up podjetjem [1]. Toda koliko slovenskih start-up podjetij je pri razvoju programskih tehnologij res uspešnih in so si zagotovila finančno samo vzdržnost?

V globalnem merilu se vse več start-up podjetij poslužuje agilnih metodologij razvoja programske opreme, s katerimi želijo optimizirati razvoj in s tem zmanjšati porabljene vire (človeške in finančne). V diplomskem delu si bomo najprej v teoretičnem delu pogledali različne lastnosti in pristope do metodologij – predvsem se bomo osredotočili na agilne metode. Razlog za to je, da omogočajo hitrejši, bolj dinamičen in usmerjen razvoj za potrebe ciljnih skupin. S tem, ko se podjetje odloči za razvoj novega produkta, lahko vzame razvoj po tradicionalnih metodah veliko časa in virov, medtem ko z agilnimi metodami dosežemo večjo efektivnost in skrajšamo pot produkta na trg.

V praktičnem delu si bomo pogledali, kakšne so metodologije razvoja programske opreme pri slovenskih start-up podjetjih s področja računalništva in informatike, s katerimi dosegajo učinkovite ter uspešne projektne rezultate. Z raziskavo ter personaliziranim pristopom do slovenskih start-up podjetij želimo klasificirati profile podjetij, ki jim je uspelo uspešno realizirati projekte z uporabo agilnih metodologij razvoja programske opreme. Pogledali si bomo tudi, ali ima vpliv na razvoj različnih informacijskih tehnologij (mobilne aplikacije, spletne aplikacije, drugo) tudi uporaba različnih agilnih metodologij in katerih. V zadnjem, sklepčnem delu bomo naredili še celoten pregled raziskave ter poskušali razjasniti postavljene hipoteze.

Cilj diplomskega dela je ugotoviti profil uspešnih slovenskih start-up podjetij s področja računalništva in informatike ter kakšne metodologije razvoja programske opreme uporabljajo glede na usmeritev svoje dejavnosti razvoja programskih tehnologij. Naredili bomo pregled profilov uspešnih podjetij, kot so velikost in izkušenost razvojne ekipe, čas ukvarjanja z razvojem produkta in uspešnost preteklih projektov. S tem bomo poskusili ugotoviti, kaj so ključne kompetence za uspeh in kako nam lahko uporaba agilnih metod poveča možnost za uspeh na trgu.

# 2 MODELI RAZVOJA PROGRAMSKE OPREME

Za boljše razumevanje diplomskega dela si najprej poglejmo, kaj pomenita pojma, kot sta modeli ter proces razvoja.

Model procesa je abstraktna predstavitev procesa razvoja. Z njim opišemo proces z določene perspektive, organiziramo aktivnosti za učinkovitejši razvoj, služi kot vodilo za obnašanje razvojne skupine, koordinacijo in sodelovanje. Vsaka posamezna metodologija ima določene procese razvoja, s katerimi se določijo smernice ter posamezni koraki za dosego našega cilja.

Proces razvoja programske opreme lahko definiramo kot strukturirano množico oziroma zaporedje aktivnosti in korakov, ki vključujejo vire in omejitve, ter so jasno časovno usklajeni. Imamo jasno zastavljen cilj, ki vodi h končnemu produktu. Lahko je sestavljen iz podprocesov, ki so med seboj hierarhično strukturirani in povezani. Osnovne aktivnosti procesov razvoja so:

- analiza in definicija zahtev (uporabnikove zahteve, specifikacije, podrobnejši opis zunanjega obnašanja sistema ter rezultati),
- načrtovanje (priprava strukture sistema, ki ustreza specifikacijam, definiranje implementacije in rezultati načrta izvedbe ter hierarhične strukture modulov),
- implementacija (pretvorba načrta izvedbe v delujoč program in razvoj programske kode).

Osnovna procesa načrtovanja in implementacije sta tesno povezana, saj se velikokrat prepletata in med seboj dopolnjujeta – še posebno pri agilnih metodologijah razvoja programske opreme.

Modele razvoja delimo na štiri osnove pristope:

1. tradicionalni modeli (kaskadni ali zaporedni in inkrementalni modeli),
2. evolucijski modeli (iterativni, prototipni in spiralni modeli),
3. agilne metode (ekstremno programiranje, Scrum, Lean itn.),
4. kombinirani modeli.

Tradicionalni modeli veljajo za najstarejše modele razvoja programske opreme. Strokovnjaki so z njimi definirali prve oblike strukturiranega pristopa k razvoju programske opreme. Še vedno so veliko v uporabi, ker so enostavni in lahko razložljivi uporabniku. Slabosti tradicionalnih modelov so naslednje:

- ni iteracij in popravkov med razvojem,
- ne nudi načina za obravnavo sprememb med razvojem,
- ni ponavljajočih se aktivnosti, ki bi vodile k ustvarjanju končnega izdelka,
- ne omogoča vzporednega izvajanja aktivnosti,
- dolg čas čakanja na končni izdelek.

Pri evolucijskih modelih velja, da faze razvoja izvajamo v več iteracijah (pri iterativnih modelih) ali pa temeljijo na predhodno izdelanih in navadno še nepopolnih različicah sistema ali dela sistema. Tipična dolžina ene iteracije pri iterativnih modelih je od enega do dva tedna. Vsaka naslednja iteracija se začne šele na koncu predhodne in je določena na osnovi rezultatov predhodne. Velja, da med izvajanjem iteracije ni nobenih sprememb. Začetne povratne informacije omogočijo zgodnje povratne informacije s strani uporabnikov. Evolucijski model nam ne omogoča predvidevanja, koliko iteracij bo potrebnih za razvoj dokončnega in dovolj dobrega izdelka.

Očitno je, da v današnjem svetu obstaja mnogo različnih modelov razvoja programske opreme, kar posledično povzroča hiter razvoj različnih metodologij pri procesih razvoja. Zaradi optimizacije virov (finančnih, človeških, časovnih itn.) vse več podjetij poskuša odkriti, kateri model razvoja ustreza njihovim zahtevam. Ker so se novejše metodologije, kamor spadajo agilne metode, izkazale za zelo učinkovite pri optimizaciji stroškov, se tega poslužuje tudi vse več slovenskih start-up podjetij na področju informacijskih tehnologij. Zaradi trenutnih ekonomskih razmer je omenjenih start-up podjetij vse več, prav tako narašča tudi število posameznikov, ki si želijo v svet podjetništva. V nadaljevanju si bomo pogledali definicijo, lastnosti in razloge za uporabo različnih agilnih metodologij.

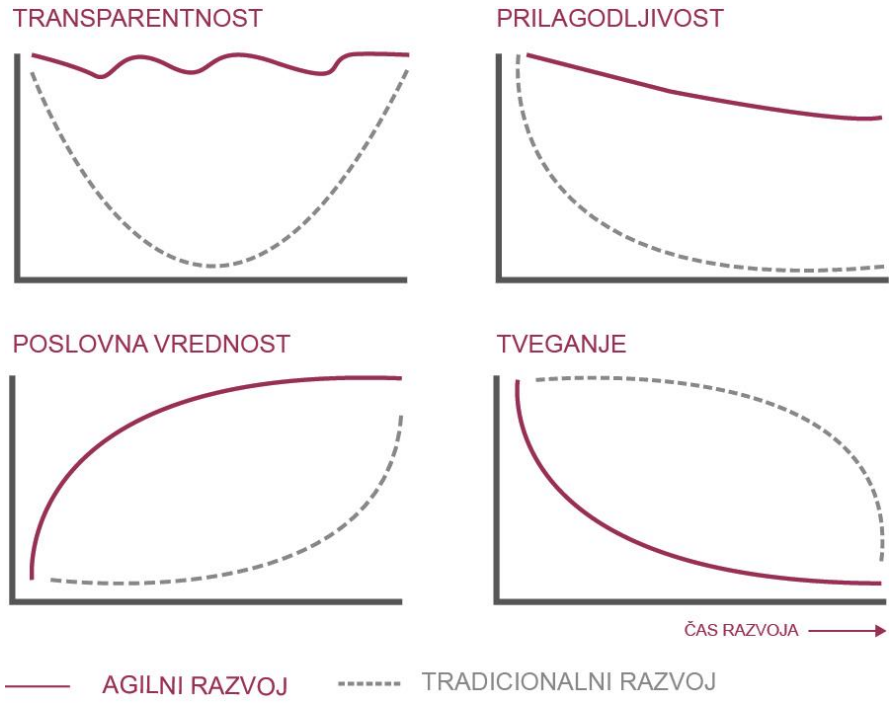
## 2.1 ZAKAJ AGILNE METODE

Agilne metodologije razvoja programske opreme so se pojavile kot alternativa težko obvladljivim, tradicionalnim metodologijam. Pri klasičnih ter tradicionalnih metodologijah so se projektni vodje začeli srečevati s kar nekaj problemi. Ker želimo ugotoviti, kaj je pripeljalo do potrebe novih in bolj učinkovitih metodologij, si poglejmo najprej probleme, s katerimi so se vodje projektov in razvijalci soočali [2]. Ugotovili so, da so problemi pri plansko vodenih metodologijah naslednji:

- imajo dolg življenjski cikel,
- so zahtevne za učenje in uporabo,
- so težko prilagodljive,
- imajo obsežne dokumentacije,

- težko vključevanje sprememb med razvojem programske opreme,
- novejše vrste programske opreme zahtevajo učinkovitejši razvoj,
- postopek razvoja je preveč zapleten in nepredvidljiv, da bi ga lahko bolj natančno planirali vnaprej.

Agilne metode so skupek več različnih metodologij za razvoj programske opreme. Bistvo agilnih procesov je ideja o preprostosti ter fleksibilnosti pri hitri izdelavi zmogljive programske opreme [glej sliko 1]. Agilna razvojna metodologija zagotavlja možnosti za ocenjevanje poteka projekta skozi celotni razvojni krog. Agilne metode temeljijo na iterativnem in inkrementalnem razvoju, ki omogoča kratke razvojne cikle – šprinte ali iteracije, katerih rezultat so delujoči in uporabni deli kode, ki gredo takoj v uporabo. Ta pristop nam omogoča močno zmanjšanje stroškov razvoja ter časa, ki je potreben, da pride izdelek do trga. Metodologija temelji na intenzivni komunikaciji ter sodelovanju z uporabniki, kar omogoča sprotno preverjanje rezultatov ter doseganje zastavljenih ciljev. S konstantno komunikacijo se preverja, ali se dosegajo želeni cilji naročnika, hkrati pa omogoča tudi vpeljavo sprememb, ki velja za eno izmed poglavitnih lastnosti razvoja programske opreme pri uporabi agilnih metod. Pri agilnem razvoju je pomembna enostavnost metod, ki omogočajo enostavno uporabo in se jih hitro nauči. To nam omogoča tudi poenostavitev pri načrtovanju in samem razvoju ter nam dovoljuje sposobnost sprotnega vključevanja sprememb zahtev naročnika [27].




Slika 1: Prednosti razvoja z agilnimi metodologijami pred tradicionalnimi metodami [33]

Poznamo več različnih modelov agilnih metodologij:

- SCRUM [3],
- LSD – »vitek« razvoj programske opreme (Lean Software Development) [4],
- XP – ekstremo programiranje (ang. *Extreme programming*) [5],
- AD – agilne tehnike podatkovnih baz (ang. *Agile Database Techniques*) [6],
- AM – agilno modeliranje (ang. *Agile Modeling*) [7],
- ASD – prilagodljiv razvoj programske opreme (ang. *Adaptive Software Development*) [8],
- Crystal,
- FDD – funkcijsko voden razvoj programske opreme (ang. *Feature Driven Development*) [9],
- DSDM – metoda dinamičnega razvoja sistemov (ang. *Dynamic Systems Development Method*) [10],
- TDD – testno voden razvoj (ang. *Test-Driven Design*) [11].

Za boljše razumevanje razvoja agilnih metodologij si poglejmo še malo zgodovine.

## 2.2 MANIFEST AGILNOSTI

Začetek agilnih metodologij sega v leto 2001, ko se je v Utahu v ZDA zbrala skupina 17 svetovalcev in praktikov, ki so razpravljali o lahkih metodologijah razvoja. Sestavili in objavili so t.i. Manifest agilnosti, s katerim so definirali družino metodologij, ki jo danes imenujemo agilni razvoj programske opreme. Dokument predpisuje principe in postopke, ki jih morajo upoštevati agilne metode.

Dokument vsebuje štiri osnova načela agilnosti [2]:

- posamezniki in njihova komunikacija so pomembnejši kot sam proces in orodja:
  - ✓ poudarja komunikacijo in povezanost med razvijalci,
  - ✓ ustvarja boljše delovno okolje, povečuje motivacijo in ustvarja občutek pripadnosti,
  - ✓ kompetence razvijalcev vključujejo tudi komunikacijske sposobnosti, skupinsko delo in prilagodljivost;
- delujoča programska oprema je pomembnejša kot popolna dokumentacija:
  - ✓ omogoča preprostost pri programiranju ter napredne rešitve,
  - ✓ s pisanjem dokumentacije ne oviramo razvoja,
  - ✓ spišemo le nujno potrebne dokumente, glavna prioriteta je razvoj delujoče programske kode;

- vključevanje in sodelovanje uporabnika je pomembnejše kot pogajanje na osnovi pogodb:
  - ✓ uporabnik najbolje ve, kaj potrebuje,
  - ✓ sodelovanje in komunikacija sta pomembnejši kot pogodba, ki velja le kot osnova;
- odzivanje na spremembe je pomembnejše od sledenja planu:
  - ✓ pogodba mora biti napisana tako, da omogoča naknadno spreminjanje ciljev na podlagi komunikacije z naročnikom,
  - ✓ nemogoče je že na začetku razvoja predvideti vse zahteve.

## 2.3 NAČRTOVANJE IN VODENJE PROJEKTA

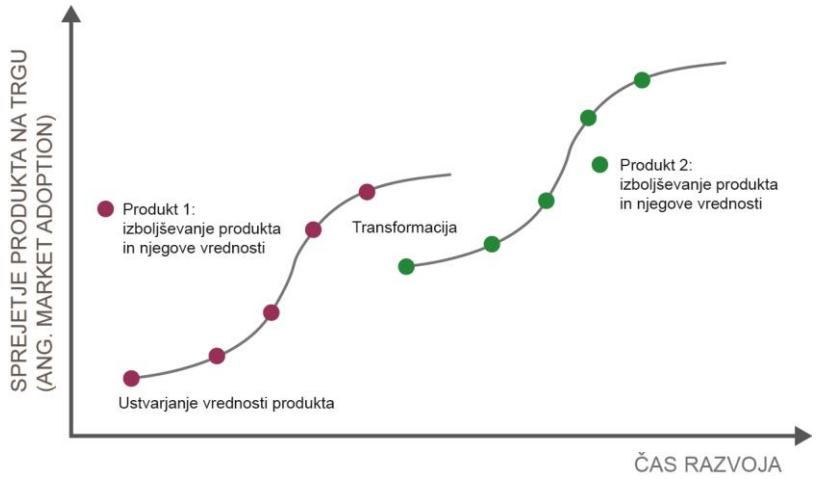
Velikokrat velja prepričanje, da se pri agilnih procesih ne posveti dovolj virov načrtovanju projekta. V resnici pa se pri agilnih metodologijah posveti veliko več virov načrtovanju ter zmanjševanju tveganja kot pri tradicionalnih metodologijah. Agilen pristop se namreč osredotoča na načrtovanje zelo pogosto v primerjavi z celovitim načrtovanjem in predpostavko, da se načrtovanje izvede samo enkrat.

Načrtovanje pri agilnih metodologijah je sestavljeno iz šestih plasti agilnega načrtovanja – »čebule« (ang. *Agile Planning Onion*). Vsaka izmed plasti določa cilje, postavi časovne okvirje ter definira lastnika in interakcijo posameznih plasti:

1. agilno načrtovanje strategije (ang. *Agile Strategy Planning*),
2. agilno načrtovanje portfelja (ang. *Agile Portfolio Planning*),
3. agilno načrtovanje različice (ang. *Agile Release Planning*),
4. agilno načrtovanje iteracije (ang. *Agile Iteration Planning*),
5. agilno dnevno načrtovanje (ang. *Agile Daily Planning*),
6. agilno neprekinjeno načrtovanje (ang. *Agile Continuous Planning*).

**Agilno načrtovanje strategije (ang. *Agile Strategy Planning*)** je prva plast, kjer se načrtuje strateška vizija, ki definira podjetje, in kaj si želi postati. Projekti in prizadevanja za razvoj produktov se idealno začnejo z vizijo, ki je povezana s poslovno potrebo. Vizija je tipično umeščena v kontekst strategije podjetja ter s tem tudi povezanimi zastavljenimi cilji. Agilno strategijo lahko predstavimo s strateško krivuljo – S-krivuljo (ang. *strategy curve/S-curve*) (glej sliko 2). S krivuljo prikazujemo, kako se produkt, storitev, tehnologija ali poslovni napredek razvija skozi čas. Na S-krivuljo se lahko gleda na primarnem nivoju kot mapiranje razvoja produkta in priložnosti ali na makro ravni za opis razvoja poslovanja in industrije. S-krivulje so na produktnem,

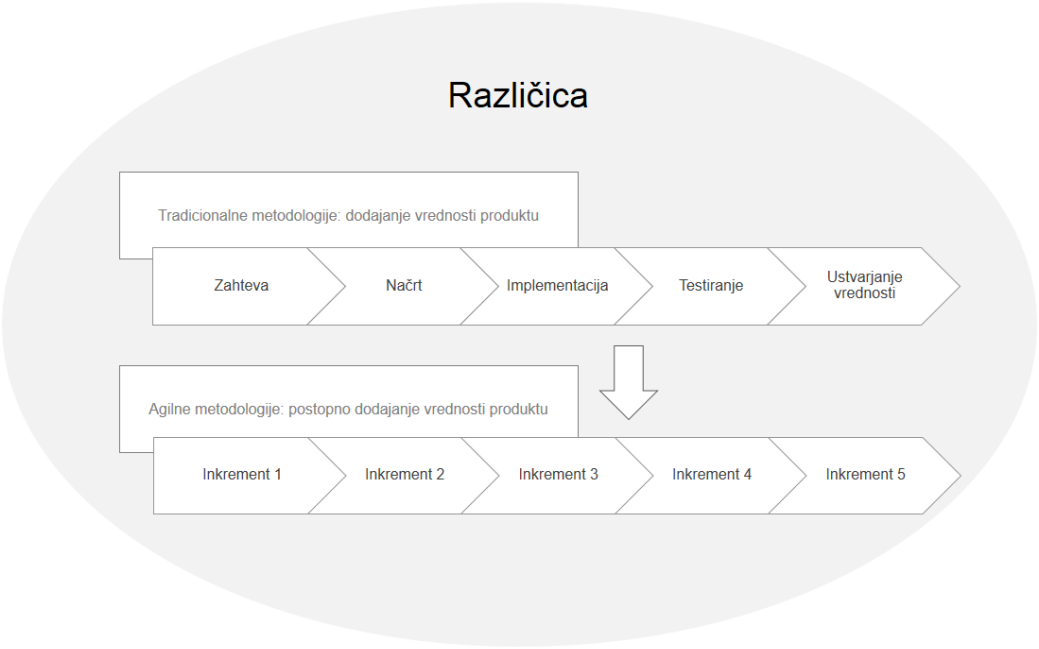
storitvenem ali tehnološkem nivoju povezane s tržno sprejetostjo oz. uspešnostjo(ang. *market adoption*), saj začetek krivulje kaže začetek tržne potrebe po nekem izdelku oziroma storitvi, konec krivulje pa prikaže zastarelost ali konec produkta, storitve ali tehnologije. Običajno konec ene S-krivulje označuje transformacijo za novo S-krivuljo, ki jo nato nadomesti.




Slika 2: S-krivulja [34]

**Agilno načrtovanje portfelja (ang. *Agile Portfolio Planning*)** je plast, ki predstavlja celovito ponudbo izdelkov, ki so sestavljeni iz aplikacijskih zbirk ter orodij in načinov medsebojnih interakcij. Cilj upravljanja portfelja (izvedba strategije) je, da se zmanjšuje izgubo na vrednosti produkta ter podaljšuje zadrževanje vrednosti. Strategija ustvarja vrednost, vendar produkt izgubi del vrednosti zaradi slabe izvedbe ali drugih organizacijskih trenj. Za razumevanje izgubljanja vrednosti produkta predvidevajmo, da imamo pravo strategijo in s tem ustvarjamo vrednost. Cilj organizacije je, da pripravi strategijo, s katero ohranjajo in podaljšujejo vrednost produkta. Rešitev za ohranitev vrednosti je upravljanje portfelja s pregledovanjem in prilagajanjem v majhnih iteracijah, glede na odziv notranjih in zunanjih uporabnikov.

**Agilno načrtovanje različice (ang. *Agile Release Planning*)** je tretja plast, ki predstavlja dostavni cikel pri agilnem razvoju. Različica ima tipičen razpon od enega do šest mesecev, vendar lahko v nekateri okoljih traja tudi dlje. Ključ pri tej plasti je neprestano dodajati vrednost v majhnih inkrementih in tako upoštevati povratno informacijo odziva notranjih in zunanjih uporabnikov. Spodnji diagram (glej sliko 3) prikazuje, kako se razlikuje tradicionalna od agilne izvedbe. Pri tradicionalni izvedbi se vrednost nalaga in doda le na koncu cikla. Ravno nasprotno se pri agilni izvedbi neprestano ustvarjajo dodane vrednosti, ki jih načrtuje portfelj.




Slika 3: Agilno načrtovanje različice [35]

**Agilno načrtovanje iteracije (ang. *Agile Iteration Planning*)** predstavlja četrto plast, imenovano tudi šprint (ang. *Sprint*). Iteracije so kratke, z določenim razvojnim številom različic, ki v praksi trajajo od enega do štirih tednov. Iteracije predstavljajo glavno izvedbo celotnega projekta. V vsaki iteraciji je cilj ekipe, da izdela del delujoče programske opreme (glej sliko 4), ki se jo tudi predstavi in demonstrira deležnikom. S tem zmanjšamo skupno tveganje in omogočamo hitro prilagajanje projekta. Pri iteraciji ne uspemo vedno dodati dovolj funkcionalnosti, da bi lahko izdali novo različico, vendar je cilj narediti različico z minimalnim številom hroščev (ang. *bugs*). Za izdajo novega produkta ali novih funkcionalnosti je potrebnih tudi več iteracij [31]. Iteracije vključujejo tri ključne faze:

- načrtovanje iteracije oz. šprinta,
- odstranjevanje ovir med izvajanjem (dnevni »stand-up« sestanek),
- opravljeno – dostavi vrednost.

**Agilno dnevno načrtovanje (ang. *Agile Daily Planning*)** predstavlja peto plast, kjer se ekipa dnevno osredotoča na dokončanje funkcije z najvišjo prioriteto. Ker se dodane lastnosti in funkcionalnosti oddajajo v iteracijah, jih lastnik produkta (ang. *product owner*) pregleda in, če so sprejemljivi, tudi sprejme. Vsak dan potekajo 15-minutni »stand-up« ali »scrum« sestanki, na katerih poteka facilitirana komunikacija pregleda napredka vsakega posameznika. Vedno se postavi tri ključna vprašanja:

- Kaj je bilo narejeno pretekli dan?

- Kaj bo narejeno danes?
- Kaj so morebitne ovire pri razvoju?

Na omenjenih sestankih se vključi tudi predstavnika stranke ter tudi druge zainteresirane deležnike kot opazovalce. Člani ekip tako poročajo drug drugemu, kaj je bilo narejeno pretekli dan, kaj načrtujejo za danes in kakšne so njihove morebitne ovire. S tem komuniciranjem lahko izpostavimo morebitne ovire. Sestanki vsak dan potekajo ob isti uri na isti lokaciji in naj bi trajali maksimalno 15 minut. Ime »stand-up« sestanek izhaja iz tega, da vsi člani ekipe stojijo, s čimer poskušajo sestanke omejiti na omenjen časovni okvir. Eno izmed ključnih orodij za ekipo so »Burndown« grafikoni, s katerimi se naredi pregled dnevnega napredka. Orodje omogoča hitro sprejemanje popravkov.

**Agilno neprekinjeno načrtovanje (ang. *Agile Continuous Planning*)** predstavlja zadnjo, šesto in najbolj kritično plast. Agilne ekipe za razvoj produktov so nenehno v stanju neprekinjenega, prilagodljivega načrtovanja, sodelovanja, razvoja, sodelovanja, razvoja, testiranja in integracije. S tem se spodbudi dinamično in visoko produktivno okolje, v katerem je produkt vedno visoko kvaliteten del programske opreme.

Ekipe pri agilnem razvoju projekta so običajno samoorganizirane in medsebojno povezane, brez posebno določenih hierarhij pravnih oseb. Običajno člani ekipe prevzamejo odgovornost za naloge, ki zagotovijo funkcionalnost, ki jo iteracija zahteva. Vsak izmed članov se sam odloči, kako bo izpolnili zahteve iteracije.

Agilne metode poudarjajo medosebno »face-to-face« komunikacijo, pri kateri se celotna ekipa zbere na enem mestu. Večina ekip deluje v odprtem prostoru (ang. *bullpen*), ki omogoča omenjeno komunikacijo. Velikost ekip je običajno majhna in šteje 5 do 9 ljudi za poenostavljeno komuniciranje in timsko sodelovanje. Pri večjih razvojnih produktih deluje več ekip skupaj v smeri skupnega cilja. Če delujejo ekipe na različnih lokacijah, lahko med seboj komunicirajo preko videokonferenc, glasovnih klepetov, e-pošte itn.

Ne glede razvoj, vsaka ekipa potrebuje predstavnika stranke. Osebo določijo predstavniki deležnikov in jih predstavlja v njihovem imenu. S tem se tudi zaveže, da bo na voljo razvijalcem za vprašanja med posameznimi iteracijami. Po vsaki končani iteraciji se deležniki ter ekipa posvetujejo o napredku ter naredijo evalvacijo prioritet funkcionalnosti z namenom optimiziranja donosnosti naložbe (ang. *return on investment*) in zagotavljanja usklajevanja s potrebami kupcev in cilji podjetja.

Agilni razvoj poudarja delujoč del programske opreme kot primarno merilo napredka. To merilo v kombinaciji z zgoraj omenjeno komunikacijo producira tudi manj pisne dokumentacije kot pri drugih metodah in se tako osredotoča bolj na razvoj. Agilen način spodbuja vse deležnike, da prioritizirajo želje in tako lažje določijo cilje posamezne iteracije.

## 2.4 OPIS AGILNIH METODOLOGIJ

Za lažje razumevanje analitičnega dela diplomskega dela si bomo pogledali podrobnejši opis nekaterih trenutno najbolj uporabnih metodologij za razvoj programske opreme v start-up podjetjih.

### 2.4.1 METODOLOGIJA SCRUM

Scrum je bil prvič definiran kot »prilagodljiva, holistično produktno razvojna strategija, kjer razvojna ekipa deluje kot enota za dosego skupnega cilja« v nasprotju s »tradicionalnim, slapovnim (ang. *waterfall*) pristopom«, ki sta ga leta 1986 definirala Hirotaka Takeuchi in Ikujiro Nonaka v članku »New New Product Development Game«.

Avtorja sta opisala nov pristop k tržnemu razvoju izdelkov, ki bi povečal hitrost in prilagodljivost in temelji na študijah primerov iz proizvodnih podjetij v avtomobilski, fotokopirni in tiskalniški industriji. Omenjeni pristop sta imenovala holistični ali »rugby« pristop, saj celoten proces izvaja medfunkcijska (ang. *cross-functional*) ekipa v več prekrivajočih se fazah, v katerih poskušajo premagati daljavo kot enota s podajanjem žoge naprej in nazaj.

V »rugby« žargonu se v ameriškem nogometu scrum nanaša na ponovni začetek oziroma nadaljevanje igre po manjšem prekršku. Beseda scrum pomeni vrnitev žoge v igro, kar nakazuje na iterativen in inkrementalen pristop k razvoju programske opreme. V začetku leta 1990 sta v različnih podjetjih Ken Schwaber in Jeff Sutherland skupaj z Johnom Scumniotalesom ter Jeffom Mckennom razvila podoben pristop, na katerega se nanaša prva uporaba besede scrum [3, 25, 26].

Scrum velja za splošno agilno metodo, ki daje večji poudarek na vodenje iterativnega razvoja, manj pa na posebne agilne postopke, kot so tehnični pristopi k agilnemu razvoju in procesu razvoja. Metodologija ne predpisuje uporabe programerskih postopkov in se uporablja kot ogrodje za vodenje projektov.

Metodologijo sestavljajo 3 faze (glej diagram 1):

1. Začetna faza – osnutek načrta:
    - pripravijo se splošni cilji projekta, načrtuje se osnutek arhitekture programske opreme;
2. Zaporedje tekov oz. šprintov:
    - v vsakem šprintu razvijemo inkrement sistema;
3. Zaključna faza – projekt izpeljemo do konca:
    - dopolnimo zahtevano dokumentacijo, kot je uporabniški priročnik ali pomoč sistema,
    - pregled znanja, ki smo ga pridobili iz projekta.

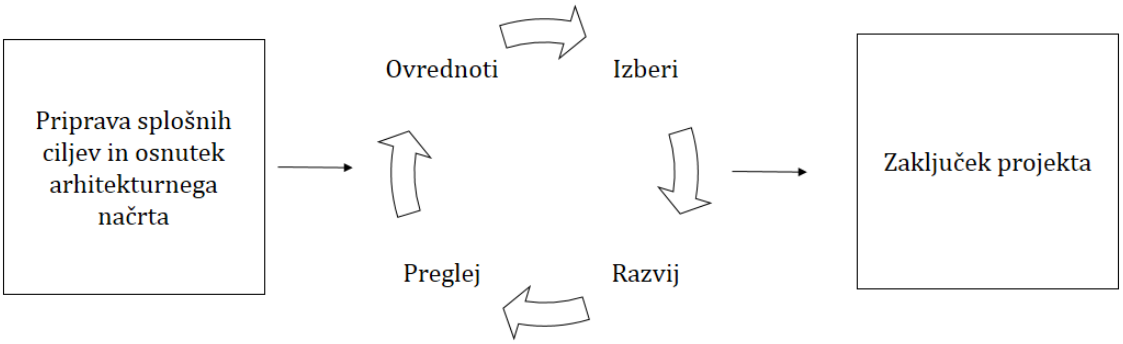



Diagram 1: Faze metodologije Scrum

**ZNAČILNOSTI**

Proces razvoja poteka v 30-dnevnih iteracijah, imenovanih *Sprint*, v katerih si iteracije sledijo ena za drugo in rezultat vsake je nova funkcionalnost izdelka. Razvojna skupina je samoorganizirana in avtonomna; več skupin lahko hkrati vzporedno implementira inkremente. Značilnost so tudi dnevni statusni sestanki, ki potekajo znotraj enega šprinta. Potrebni so za koordinacijo projekta ter pregled opravljenega dela in prilagoditve.

Vloge pri metodologiji Scrum so naslednje:

- predstavnik naročnika (ang. *Product Owner*),
- razvojna skupina (ang. *Scrum Team*),
- skrbnik metodologije (ang. *Scrum Master*),
- opazovalci (ang. *Observers*).

**Predstavnik naročnika** je oseba, ki je uradno zadolžena za celotno izvedbo projekta in zastopa interese uporabnika ter skrbi za financiranje projekta. Predstavlja vse, ki so

zainteresirani za projekt in tudi končne rezultate, skrbi za zagotavljanje sredstev upravičenosti projekta (ang. *ROI*). Prav tako predstavnik naročnika vzdržuje seznam zahtev, ki obsegajo: izdelavo začetnega seznama, prioritete posameznih zahtev, oceno časa razvoja s pomočjo razvijalcev in plan predaje posameznih različic izdelka. Seznam zahtev ni dokončen, ampak se v času izvajanja projekta dopolnjuje glede na potrebe naročnika in zahtev.

**Razvojna skupina** odgovarja za razvoj novih funkcij pri posamezni iteraciji. Kot že omenjeno, deluje po principu samoorganizacije, kar pomeni, da si sami izberejo način realizacije posamezne zahteve in so sami odgovorni za uspeh posameznih iteracij in projekta v celoti. Velikost razvojne skupine v povprečju vključuje 7 ± 2 članov, ki imajo potrebno znanje za dosego ciljev posameznega šprinta. Naloga vodstva ni predpisovanje dela razvojni skupini, temveč ustvarjati okoliščine, ki bodo čim manj motile delo ekipe.

**Skrbnik metodologije** je zadolžen za nadzor skladnosti celotnega procesa razvoja z metodologijo Scrum. Delo obsega učenje metodologije vseh sodelujočih pri projektu, skrb, da se metodologija vklopi v kulturo organizacije in daje pričakovane rezultate ter da se upoštevajo pravila in prakse metodologije. Zadolžen je za organizacijo dnevnih sestankov, spremljanje seznama zahtev, ki jih je potrebno opraviti, beleženje odločitev in merjenje napredka, odpravlja tudi ovire pri delu skupine in zagotavlja potrebne vire, pomaga pri odločitvah razvojne skupine in komunicira z naročniki ter vodstvom izven skupine.

**Opazovalci** so vsi, ki jih projekt zanima in se ne smejo neposredno vmešavati v razvoj produkta. Lahko so prisotni na dnevnih sestankih kot gosti, vendar jim ni dovoljeno aktivno participirati na sestanku.

Zelo pomembni so tudi sestanki, ki jih metodologija narekuje, in sicer:

- planiranje teka (ang. *Sprint Planning Meeting*),
- dnevni sestanek (ang. *Daily Scrum*),
- pregled teka (ang. *Sprint Review Meeting*),
- ocena teka (ang. *Sprint Retrospective Meeting*).

Za realizacijo so potrebni naslednji izdelki metodologije Scrum:

- seznam zahtev (ang. *Product Backlog*),
- plan teka (ang. *Sprint Backlog*),
- diagram izgorevanja (ang. *Sprint Burndown Chart*),

- povečanje funkcionalnosti izdelka (ang. *Increment of Potentialy Shippable Product Functionality*).

Potek Scrum metodologije se začne s seznamom zahtev (ang. *Product Backlog*) naročnika. Seznam, kot že omenjeno, dopolnjuje in vodi predstavnik naročnika. Glede na potek razvoja se ta seznam dopolnjuje in po potrebi dodaja nove zahteve. Vsaka zahteva ima tudi svojo prioriteto, ki se jo določi glede na potrebe uporabnika. Večja kot je prioriteta zahteve, večjo prednost ima pri samem razvoju.

Na začetku vsakega *Sprinta* se predstavnik naročnika sestane skupaj z razvojno skupino na 8-urnem sestanku (ang. *Sprint Planning Meeting*), na katerem poteka planiranje iteracije in se določa zahteve, ki jih je potrebno realizirati. Sestanek je razdeljen na dva dela po 4 ure. V prvem delu se predstavnik naročnika in razvojna skupina dogovorita, katere zahteve iz seznama, ki imajo najvišjo prioriteto, bodo vključene v naslednjo iteracijo. V drugem delu izdela razvojna skupina seznam nalog (ang. *Sprint Backlog*), ki jih bodo realizirali do konca iteracije. [13]

Vsakodnevno potekajo tudi kratki 15-minutni sestanki (ang. *Daily Scrum Meeting*), na katerih vsak član razvojne ekipe odgovori na 3 vprašanja:

- Kaj si delal od prejšnjega sestanka?
- Kaj nameravaš delati do naslednjega sestanka?
- S katerimi težavami se srečuješ pri delu?

Z dnevnimi sestanki in vprašanji se tako zagotovi vpogled v potek projekta in v primeru težav tudi pomoč.

Po vsaki končani iteraciji razvojna skupina predstavi izdelane funkcionalnosti predstavniku naročnika ter zainteresiranim uporabnikom. Na sestanku (ang. *Sprint Review Meeting*) lahko tako uporabniki predstavijo svoja mnenja ter podajo predloge za naslednjo iteracijo.

Med iteracijami se razvojna skupina sestane s skrbnikom metodologije (ang. *Scrum Master*), da ocenijo potek dela v pretekli iteraciji. Pri tem se dogovorijo tudi za morebitne izboljšave, da bo potekal nadaljnji razvoj še bolj učinkovito, in da zagotovijo večjo kakovost končne programske opreme.

Metodologijo uporabljajo pri svojem delu tudi največja svetovna podjetja, kot so Microsoft, Oracle, Google, IBM in drugi, kar dokazuje učinkovitost in kvaliteto omenjene metodologije.

### 2.4.2 EKSTREMNO PROGRAMIRANJE (XP)

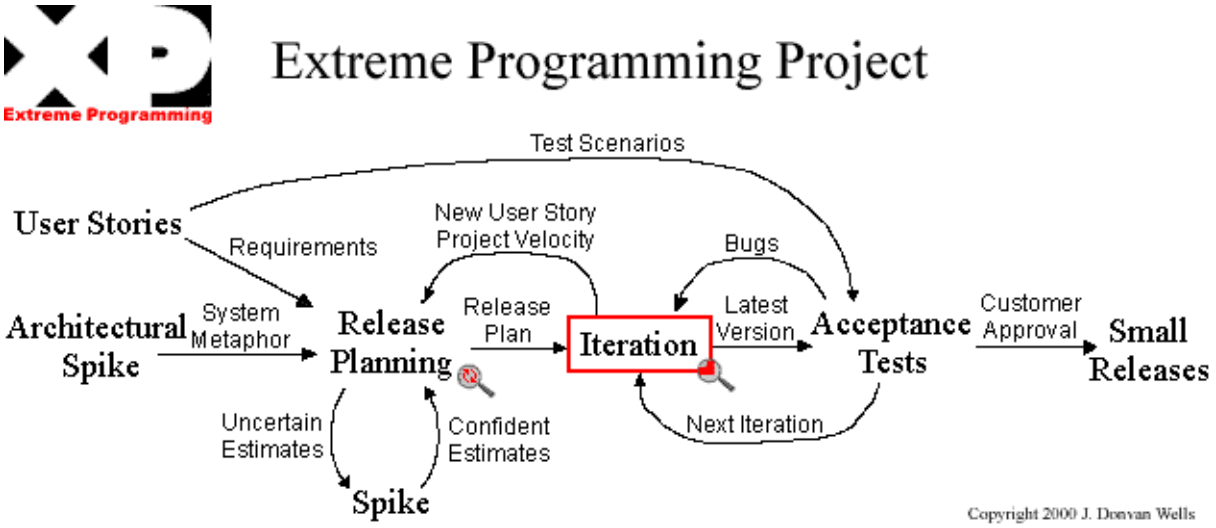
Prvi znani projekt, ki je vključeval proces ekstremnega programiranja, se je začel leta 1996 kot eden izmed mnogih procesov agilnih metodologij. Glavni razlog za uspešnost ekstremnega programiranje je fokusiranje na zadovoljstvo uporabnikov.

Ekstremno programiranje je proces, pri katerem ne dostavimo vseh ciljev in funkcionalnosti, ki bi si jih želeli na začetku, vendar le tiste dele programske opreme, ki jih potrebujemo. Zaradi tega se lahko razvijalci povsem prilagajajo zahtevam končnih uporabnikov in naročnika, ki se lahko hitro spreminjajo, tudi ko je produkt že v končni fazi produktnega cikla.

Predvsem se pri ekstremnem programiranju poudarja timsko delo. Manager, naročnik ali razvijalec – vsi imajo enakovredno vlogo pri procesu izdelave in veljajo za enakovredne partnerje. Ekstremno programiranje implementira preprosto, vendar učinkovito delovno okolje, ki omogoči ekipam visoko produktivnost. Visoko produktivnost se dosega na način samoorganizacije ekipe za efektivno reševanje problemov.
Ekstremno programiranje vključuje 5 principov za izboljšanje projekta:

- komunikacija (ang. *communication*),
- preprostost (ang. *simplicity*),
- povratna informacija (ang. *feedback*),
- spoštovanje (ang. *respect*),
- pogum (ang. *courage*).

»Ekstremni programerji« oziroma razvojni del ekipe konstantno komunicira z naročniki in končnimi uporabniki kot tudi s sodelavci. Njihov koncept programiranja temelji na preprosti in »čisti« programski kodi. Povratne informacije začnejo zbirati že s prvim dnem razvoja programske opreme. Celoten produkt ponudijo naročniku takoj, ko je to mogoče in nato implementirajo želene spremembe. Vsaka majhna nadgradnja in uspeh še poglobita medekipno spoštovanje do unikatnih prispevkov vsakega člana razvojne ekipe. Našteti principi so zgolj temelji za pogumen odziv na spreminjajoče se zahteve in tehnologije pri razvoju.




Slika 5: Potek projekta z uporabo metodologije ekstremnega programiranja [5]

Najbolj presenetljiv aspekt ekstremnega programiranja je v preprostosti pravil, ki so sama po sebi zelo preprosta in posamezno ne predstavljajo nekega smisla. Ko vsa pravila kombiniramo skupaj, lahko šele opazimo celotno sliko. Sprva se lahko pravila zdijo celo naivna, vendar temeljijo na principih in vrednotah.

Najpomembnejša pravila in koncepti ekstremnega programiranja so:

- pogosta integracija (ang. *often integration*): razvojna ekipa naj bi dnevno implementirala in dodajala novosti produktu;
- hitrost poteka projekta (ang. *project velocity*): hitrost je kazalnik, koliko dela se opravi na projektu, s tem kazalnikom se napoveduje planiranje izdaje novih različic in napovedanih izboljšav;
- programiranje v paru (ang. *Pair programming*): programsko kodo razvijata dva razvijalca hkrati na enem računalniku, s čimer se zagotovi višjo kvaliteto produkta;
- uporabniška zgodba (ang. *user story*): opisuje problem, ki ga bo končni produkt reševal; zgodba mora biti predstavljena s strani uporabnika in mora biti dolga vsaj 3 povedi. Zgodba ne opisuje rešitve in ne vsebuje tehničnih izrazov.

Cikel projekta, ki sloni na ekstremnem programiranju, se začne z začetno fazo planiranja, ki ji sledijo iteracije. Na koncu vsake iteracije sledi testiranje s strani uporabnika oziroma naročnika. Ko ima produkt dovolj funkcionalnosti, da zadovolji potrebe uporabnika, razvojna ekipa zaključi iteracijo in izda del programa.

### 2.4.3 METODOLOGIJA VITKEGA RAZVOJA (ANG. *LEAN METHODOLOGY*)

Vitka metodologija (ang. *Lean methodology*) spada med najmlajše izmed metodologij za razvoj start-up podjetij. Metodologija se uporablja že vrsto let, vendar sta jo prvič predstavila leta 2003 avtorja Mary in Toma Poppendiecka v prispevku Lean Software Development: An Agile Toolkit.

Filozofija omenjene metodologije bazira na razvoju vitke proizvodnje (ang. *lean manufacturing*). Z njo so se začeli ukvarjati japonski proizvajalci avtomobilov v začetku osemdesetih let. John Krafic, ki je bil v tem času inženir pri Toyoti, je med opazovanjem procesa in orodij, ki se uporabljajo za odstranitev odpadkov (ang. *eliminate waste*) in množično proizvodnjo avtomobilov, skoval izraz *lean manufacturing*.

Medtem ko je metodologija Scrum skupek pravil in vlog, je metodologija vitkega razvoja skupek načel in konceptov s peščico orodij. Obe tehniki delita isto ideologijo, katere cilj je čim hitrejši razvoj, hkrati pa poskušata zmanjšati pomanjkljivosti in napake. Agilne metodologije poudarjajo prilagodljivost, vendar je dejstvo, da je metodologija Scrum skupek obveznih pravil. Metodologija vitkega razvoja je bolj odprta in predstavlja proces razvoja kot skupek visoko prilagodljivih priporočil. To pomeni, da spodbuja ekipe in podjetja, da sprejemajo odločitve in se tako prilagajajo vsakodnevnim problemom in oviram.

Za boljše razumevanje si poglejmo, kaj pomeni vrednost (ang. *value*) produkta. Vrednost definiramo kot storitev ali proces, za katerega je kupec pripravljen plačati. Koncept vitkega razvoja je torej ohranjanje vrednosti z manj dela.

## PROCES LEAN START-UP METODOLOGIJE

Metodologija »Lean Start-up« start-up podjetja uči, kdaj je potrebno zamenjati fokus in kako ohraniti rast podjetja z maksimalno hitrostjo skozi optimizacijo procesov. Z uporabo »Lean Start-up« pristopa lahko podjetja ustvarijo red in ne kaos z uporabo orodij, s katerimi se neprestano testira vizijo produkta.

Poglejmo si nekaj glavnih procesov »Lean Start-up« metodologije. [4, 28]

### 1 Odstrani negotovost (ang. *Eliminate uncertainty*)

Zaradi pomanjkanja prirejenih in prilagojenih procesov znotraj managementa je pripeljalo mnogo start-upov do tega, da opustijo vse procese. Iz tega se je razvil pristop »Just do it«, kot ga omenja Eric Rie. S tem se izognemo vsem procesom managementa.

»Lean start-up« metodologija ni samo porabiti  manj sredstev, temveč tudi postaviti proces oziroma metodologijo kot okvir za razvoj produkta.

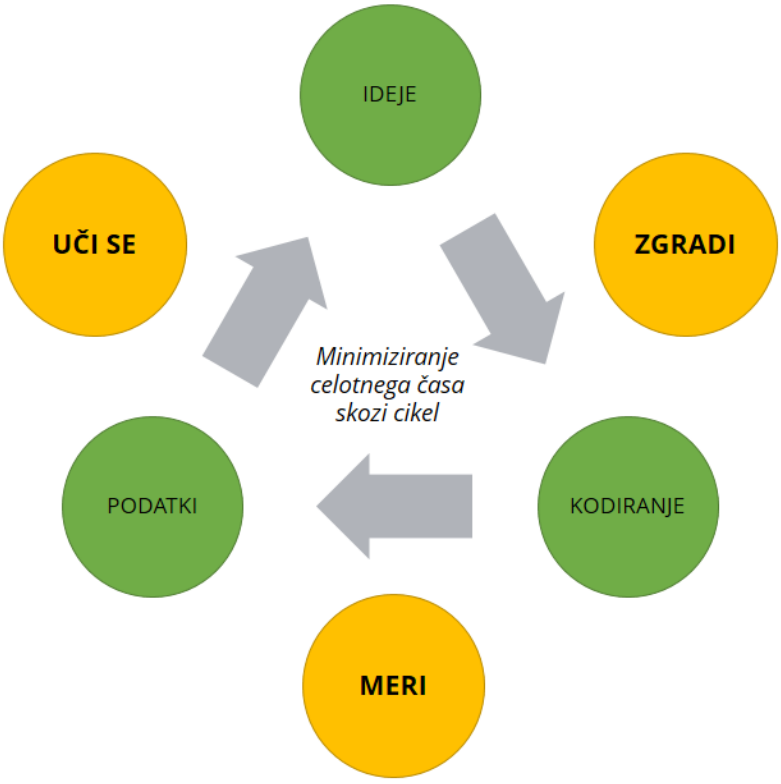
## 2 Delaj pametno (ang. *Work smarter not harder*)

»Lean Start-up« metodologija predpostavlja, da je vsak nov projekt ali start-up podjetje eksperiment, ki poskuša odgovoriti na neko vprašanje. Dandanes ni več vprašanje:  »Ali je možno ta izdelek zgraditi?«, temveč: »Ali bi bilo smiselno ta izdelek zgraditi?« in »Ali lahko zgradimo trajnostni model in poslovanje okoli tega izdelka ali storitve?«.

Če nam uspe odgovoriti na ti dve vprašanji, potem lahko pričnemo s procesom pridobivanja prvih strank in postopno gradnjo našega izdelka glede na povratno informacijo. To nam omogoča, da bo do časa, ko bo produkt pripravljen na široko distribucijo, že imel uveljavljene stranke.

## 3 Minimalno sprejemljiv produkt (ang. *Miminum viable product – MVP*)

Bistvena sestavina »Lean Start-up« metodologije je povratna zanka zgradi – meri – uči se (ang. *Build-Measure-Learn*), kot jo prikazuje slika 6.




Slika 6: Povratna zanka zgradi – meri – uči se [36]

Prvi korak je, ugotoviti problem, ki ga je potrebno rešiti. Drugače povedano, podjetnik mora biti pripravljen na test postaviti vse svoje predpostavke ter prepričanja o trgu.

Drugi korak je razviti minimalno sprejemljiv produkt – izdelek z minimalno vrednostjo donosa, da lahko čim hitreje pričnemo s postopkom učenja. MVP je sprejemljiv produkt brez mnogih funkcionalnosti, ki se kasneje lahko izkažejo za pomembne. Gre za prototip produkta. MVP je lahko spletna stran, video predstavitev ali pa zgodnji prototip produkta, ki ga pokažemo potencialnim strankam. Stopnja (ne)razvitosti MVP-ja ne pomeni, da je produkt površen ali pa da ne odraža profesionalnosti podjetnika.

MVP je tista stopnja produkta, ki omogoči ekipi zbrati največ informacij preko učenja z najmanj vloženega napora. Omogoča začetek procesa učenja. Ko imamo naš MVP, se prične merjenje in učenje iz pridobljenih podatkov, ki lahko dokažejo vzročno-posledična vprašanja. Na podlagi povratnega mehanizma podjetnik spoznava, kdo so stranke, in pridobiva povratne informacije o vrednosti produkta, ki služijo kot osnova za opustitev ali ohranitev funkcionalnosti.

Eden izmed načinov pridobivanja informacij je, ali bomo vztrajali pri trenutnem razvoju ali pa bo potrebno narediti obrat (ang. *pivot*). »Pivot« pomeni preobrat v poslovanju oz. bolj strokovno strukturiran popravek, namenjen testiranju nove temeljne hipoteze o produktu, strategiji ali glavnem motorju rasti. Obrat lahko razumemo tudi kot novo strateško hipotezo, ki potrebuje nov MVP za testiranje. Uspešen obrat postavi podjetje na pot proti trajnostnemu poslovnemu modelu.

Med preizkušanjem hipotez, v ciklu zgradi – meri – uči, podjetnik spoznava, ali funkcionalnost prinaša končnim kupcem vrednost ali ne. Če podjetnik pridobi potrditev, se razvoj nadaljuje, v primeru negativnega odgovora je potreben obrat. Posledično to pomeni, da je potrebno celoten proces učenja začeti ponovno in prav tako tudi postaviti nove hipoteze o produktu in strategiji [28].

## 4 Validirano učenje (ang. *Validated learning*)

Napredek v proizvodnji se meri s količino kvalitetno izdelanih produktov. Enota za merjenje napredka pri »Lean Start-up« metodologiji je validirano učenje. Validirano učenje je rigorozna metoda za demonstriranje napredka produkta, ko je produkt še v skrajni negotovosti. Sprejetje validiranega učenja bistveno skrajša razvojni proces, saj omogoči pravilno fokusiranje razvoja na tiste funkcionalnosti, ki jih končni uporabniki želijo in so za njih pripravljeni plačati. Ravno to je prednost pred običajnim lansiranjem

»beta« produktov podjetij, kjer lahko čakanje traja občutno dlje časa. »Lean Start-up« omogoča hiter odziv na povratne informacije in s tem spremembo razvoja in prilagajanje produkta končnim uporabnikom.

## PRINCIPI METODOLOGIJE VITKEGA RAZVOJA

### 1 Odstranjevanje odpadkov (ang. *Eliminate waste*)

Najprej se moramo vprašati, kaj sploh štejemo kot odpadek (ang. *waste*), če želimo podrobno razumeti omenjeno metodologijo. Kot odpadek štejemo:

- vse, kar ne doprinese vrednosti končnemu produktu (vrednost je definirana glede na percepcijo kupca),
- nepotrebne vrstice kode in funkcionalnosti,
- nejasne ali spreminjajoče se zahteve,
- neučinkovito in počasno notranjo komunikacijo ali proces,
- birokracijo,
- delno dokončano delo,
- napake in vprašanja kakovosti.

Skupna praksa vse agilnih metodologij je retrospektiva na preteklo delo. Omenjali smo že vsakodnevne sestanke pri Scrum metodologiji. Zelo pomemben del procesa razvoja je namreč sestanek po vsaki končani iteraciji, da se omogoči prostor za diskusijo, kaj je šlo dobro, kaj ni šlo dobro in kaj bi bilo lahko narejeno drugače pri naslednji iteraciji. To je pomemben del identificiranja in eliminacije odpadkov.

Proces identificiranja in eliminacije odpadkov naj bi potekal regularno in bil implementiran v vsako iteracijo. Frekvenco procesa določi vsaka ekipa zase in ga obravnava pravočasno v majhnih korakih. Nenehno izboljševanje in učno okolje doseže z majhnimi, vendar pogostimi izboljšavami. Na ta način se ustvarja tudi sama kultura dela in s tem omogoči prednost pred konkurenco [29].

### 2 Vgraditi kakovost (ang. *Build Quality In*)

Razvojni proces ne sme dovoliti pojavljanja napak. Če to ni mogoče, je potrebno spremeniti proces. Najprej se napako potrdi, nato se nastalo težavo popravi, nakar sledi ponovitev iteracije. Pregledovanje in popravljanje napak, ki se lahko nabirajo v vrsti v nekem trenutku prihodnosti, se je izkazalo za neučinkovito.

Učinkoviti postopki za zagotavljanje kakovosti in preprečevanje težav s kakovostjo že obstajajo, in sicer obe spodaj omenjeni metodi spadata med dobre prakse metodologije ekstremnega programiranja (XP).

**Programiranje v paru (ang. *Pair programming*) -** metoda se poskuša izogniti težavam s kakovostjo z uporabo dveh razvijalcev za vsako nalogo. Naloga s tem pridobi kvaliteto, saj upošteva izkušnje dveh razvijalcev, kar pogosto privede do večje produktivnosti. Omogoči tudi pogled iz druge perspektive in rešitve, ki jih prvi razvijalec ne bi opazil. Ena izmed prednosti je tudi izboljšana kakovost, saj lahko ena oseba misli malo pred drugo in razreši morebitna vprašanja še preden se pojavijo.

**Razvoj, osredotočen na testiranje (ang. *Test Driven Development - TDD*)** preprečuje težave s kakovostjo, tako da pripravi teste pred samim pisanjem kode. V najpreprostejši obliki za zagotavljanje kvalitete razmišljamo o preskusnih pogojih vsake funkcije, preden se jo razvije. Če razvijalec razume, kako se funkcijo testira, je verjetnost, da bo napisana koda obravnavala vse možne scenarije, veliko večja. V bolj sofisticirani obliki zagovarja metoda ekstremnega programiranja pisanje avtomatiziranih testov za vsakega od preskusnih pogojev, predno se napiše kodo.

Še eden izmed učinkovitih pristopov za preprečevanje težav s kakovostjo je minimiziranje časa med stopnjami procesov razvoja. To pomeni, da želimo minimizirati čas med samim razvojem, testiranjem in razhroščevanjem. Boljše kot zapisovanje hroščev za poznejšo obravnavo, se jih je lotiti takoj. Samo beleženje vzame čas, kar uvrstimo med odpadek. Prav tako lahko časovni premori povzročijo izgubo na konstantnem razvoju ter fokusu.

Če povzamemo, ima kvaliteta izjemen pomen, v nasprotnem primeru ustvarjamo nepotrebne odpadke v vseh možnih različnih oblikah. Zato je pomembno, da v vseh procesih vgradimo kvaliteto v naš produkt čim prej v začetnih fazah in jo izboljšujemo skozi celoten razvojni proces.

## 3 Ustvariti znanje (ang. *Create Knowledge*)

Zelo pomemben aspekt in princip vitke metodologije je tudi ustvarjanje in kopičenje znanja. Verjetno se večini zdi ta princip trivialen in samoumeven, vendar je znanje bistvenega pomena. Za zagotavljanje znanja in večje produktivnosti na daljši časovni rok so na razpolago nekatere tehnike, in sicer:

- programiranje v paru,

- pregled kode,
- dokumentacija,
- wiki, kjer lahko znanje dopolnjujemo inkrementalno,
- temeljito dokumentirana koda,
- prenosi znanj in dobrih praks,
- treningi,
- različna orodja za management.

## 4 Odložiti odločitev (ang. *Defer Commitment (Decide as late as possible)*)

Odločitve, ki so nepopravljive in jih ne moremo kasneje spremeniti, naj bi se sprejemalo čim pozneje. Pomembno je tudi, da se te odločitve ne pusti čakati predolgo. To lahko upočasni ekipo in bo naredilo projekt še težji. Vendar so te kritične odločitve lahko veliko lažje v kasnejšem odločanju, saj lahko v tem času zberemo veliko več informacij in si s tem zagotovimo pravilno odločitev.

Odlaganje nepopravljivih odločitev torej pomeni imeti naše možnosti odprte čim dlje. Do trenutka, ko je treba sprejeti odločitev, je potrebno izkoristiti vsako priložnost, s katero lahko izvemo, katera je najboljša odločitev. Prav tako nam omogoča čas, da lahko spoznamo različne možnosti in se bolj poglobimo v odločitev, ki nam lahko pomagajo, da pridemo do pravega zaključka.

Pri tem koraku je tudi pomembno dobro arhitekturno zasnovati naše rešitve, da bodo prožne, če želimo omogočiti sprejetje odločitve čim kasneje.

## 5 Dostaviti hitro (ang. *Deliver Fast*)

Zdi se očitno, da vedno stremimo k temu, da čim prej dokončamo naš produkt, vendar ni vedno tako. Vse prepogosto se v svetu razvoja programske opreme zgodi, da se razvoj zavleče. Velikokrat posamezniki preveč razmišljajo o prihodnjih zahtevah, ki se lahko ali pa se ne pojavijo vedno.

Pogosto je problem, da razvijalci dobijo blokado pri reševanju problemov in potrebujejo pomoč ali informacije preostalih vpletenih v projekt, zato ne odreagirajo s pozornostjo, kakršno problem v resnici zahteva. Gotovo je tudi, da se posveča preveč pozornosti »over-engineeringu« rešitvi, tako v svetu programske arhitekture kot poslovnih zahtev. Posledica tega je, da ekipe začnejo razvijati komplekse rešitve,

namesto da bi se osredotočile na preprostost produkta, za katerega se lahko nato hitro pridobi povratno informacijo od uporabnikov in gradi na tem.

Zelo pomembna je medekipna komunikacija, saj se v praksi velikokrat zgodi, da ekipa čaka nekoga, da dokončna nalogo, medtem ko bi se lahko lotila že drugega zahtevka. Če ekipa potrebuje nekaj, s čimer bi dosegli svoje cilje, je potrebna pomoč drug drugemu, če tudi to delo ni v njihovi specifični domeni.

Hitrost produkta na trgu je brez dvoma prednost pred konkurenco. Veliko je podjetij, ki se uspejo prebiti na trg prvi in tako pridobijo konkurenčno prednost. Nekatera podjetja lahko tudi kopirajo in naredijo boljši produkt, vendar se pogosto zgodi, da je tisto podjetje, ki prvo vzpostavi stik s trgom, dolgoročni voditelj na svojem področju.

### 6 Spoštuj ljudi (ang. *Respect People*)

Še eden izmed principov, ki se zdi na prvi pogled logičen. V realnosti velikokrat prihaja do tega, da posamezniki pozabijo na osnovno človeško vljudnost, posebno na delovnem mestu. Največkrat se taki izkažejo prav managerji oz. bolj izkušeni »senior« razvijalci.

Kaj to pomeni v praksi? Pomemben je takojšen odziv na mnenje ljudi, ki jih moramo pozorno poslušati, čeprav se njihovo mnenje razlikuje od našega. Pomemben aspekt je tudi empatija in videti stvari iz perspektive drugih posameznikov. Seveda to ne pomeni, da se morajo vedno strinjati z drugimi. Pomembno je, da se naučimo biti asertivni in pokazati naše nestrinjanje brez agresije in napadalnosti.

Ekstremno pomembno je pokazati ljudem zaupanje in jim dati priložnost, da sprejemajo odločitve o svojem delu. Za dosego tega je potrebno zgraditi veliko znanja in omogočiti strokovni razvoj posameznikom, ki lahko razmišljajo izven okvirjev.

### 7 Optimizirati celoto (ang. *Optimize the Whole*)

Ekipe, ki se ravnajo po vitki metodologiji, stremijo k temu, da optimizirajo celoten proces razvoja in ne samo delo znotraj ekip ter posameznih področij. Velikokrat se v svetu razvoja zgodi, da prihaja do velikih zamud projektov in procesov, prav tako kot tudi problemov v medekipni komunikaciji. Rešitev je v ekipah, ki so multidisciplinarne, zgrajene iz posameznih strokovnih področij, ki imajo znanja in sposobnosti, da končajo zahtevo od začetka do konca, brez potrebe po referencah drugih ekip.

V praksi se takšne ekipe in strukture težko doseže, zato je zaželeno, da se agilne pristope implementira od vodstva do razvijalcev.

Optimizacija delovnega procesa se izkaže na dolgi rok ključnega pomena – ne samo v ekipnih dosežkih, temveč tudi v sami kvaliteti produkta. Dolgoročno to pomeni biti bolj kompetenčen na trgu.

### 2.4.4 KANBAN

Kanban spada med tehnike upravljanja procesa razvoja programske opreme na zelo učinkovit način. Uporablja se predvsem pri uporabi agilnih metodologij, kot so Scrum in Lean. Deluje na konceptu »ravno v pravem času« (ang. *just-in-time*), medtem ko med razvojem ne preobremenjuje članov razvojne ekipe.

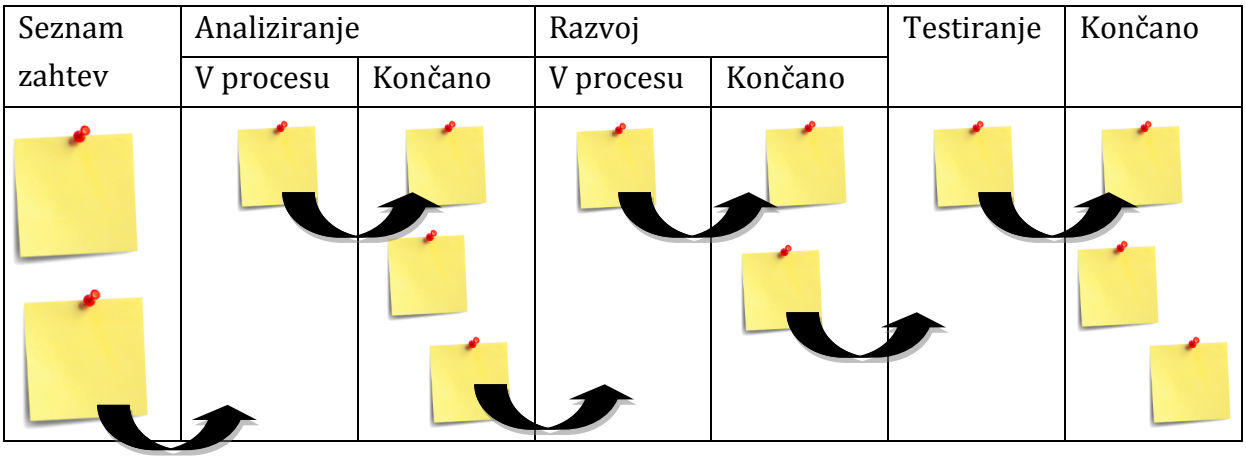
Ime Kanban ima izvor na Japonskem in se ga lahko prevede v signalizacijska kartica (ang. *signal card*). Kanban metodo je formuliral David J. Anderson kot pristop k inkrementalnemu, evolucijskemu procesu razvoja v organizacijah. Kanban omogoča transparentnost v razvoju, kjer je naročnik del razvojne ekipe. [15]

Poglejmo si 5 glavnih lastnosti uporabe Kanban pristopa:

1 **Vizualizacija poteka dela (ang. *Visualize the workflow*)**

Običajno je potek dela sam po sebi neviden, zato metoda poudarja vizualizacijo poteka dela. Pomembno je, da naredimo naš razvojni proces viden (razberemo lahko že iz samega pomena besede Kanban), kar je ključno za naše razumevanje, kako bo delo potekalo. S tem omogočimo tudi lažje sprejemanje pomembnih odločitev, če je treba spremembe sprejeti.

Običajen način za vizualizacijo poteka dela je, da uporabimo steno ali tablo za kartice s karticami in stolpci. Stolpci predstavljajo različne stopnje ali korake v samem procesu razvoja in poteka dela (glej sliko 7).




Slika 7: Primer vizualizacije s pomočjo Kanbana

Proces razvoja programske opreme si lahko predstavljamo kot cevovod, kjer na eni strani vstopajo novi zahtevki za razvoj, na drugi strani pa izhaja izboljšana programska oprema. Iz seznama si tisti razvijalec, ki področje zahtevka najboljše pozna, vzame en zahtevek, ki ga nato prestavi v stolpec za analizo. Po končani analizi razvijalec postavi v stolpec končano, kjer ga lahko nato drug razvijalec vzame in zahtevek pripelje do konca razvoja, ko je pripravljen za testiranje. Nato zahtevek, pripravljen za testiranje, vzame tretja oseba, ki preveri, ali je zahtevek končan, ali pa ga vrne nazaj v prejšnje stolpce z zahtevki za popravek. S takšno preprosto vizualizacijo omogočimo razvojni ekipi ter naročniku transparentnost.

### 2 Omejevanje razvoja (*ang. Limit work-in-process*)

S tem ko omejimo (količino dela, ki nas čaka, čas za razvoj itn.), da bomo razvili le določene zahtevke, dosežemo, da se izpostavijo problemi. Če želimo problem uspešno razrešiti, je treba problem najprej jasno izpostaviti. Pomembno je, da zahtevkom določimo prioriteto, s katero določimo, kaj je treba najprej končati. S tem lahko iz seznama zahtevkov izločimo tiste, ki imajo nizko prioriteto in z njimi ne izgubljamo virov. Omejevanje razvoja je eden izmed ključnih temeljev Kanbana. [15]

### 3 Upravljanje poteka (ang. *Manage flow*)

Ključni del pri vpeljavi uporabe Kanban modela v razvoj je spremeniti razvoj v bolj efektivno uporabo virov. Preden lahko to naredimo, moramo vedeti, kaj sploh spremeniti. To dosežemo tako, da ugotovimo, kako vrednost produkta potuje skozi celoten cikel razvoja. Potrebno je analizirati problematične odseke, kjer naša vrednost produkta porablja največ virov. Po analizi lahko implementiramo spremembe. Ko nato

ponovimo cikel razvoja, lahko vidimo, kakšen učinek imajo naše spremembe na celoten potek – ali je bila sprememba dobra ali ne. Ta celoten cikel se nikoli ne končna, saj vedno želimo naš potek razvoja izboljšati in iščemo efektivnejše rešitve.

### 4 Narediti jasne politike (ang. *Make policies explicit*)

Kot smo že omenili, je nemogoče izboljšati nekaj, česar ne razumemo. Zato moramo poskrbeti, da so vsi procesi razvoja definirani, objavljeni in razumljivi celotni ekipi, ki sodeluje pri razvoju. Brez definiranih procesov in eksplicitnega razumevanja, kako mora delo potekati, lahko hitro pride do konfliktov. Šele ko vsi razumemo, zakaj, kako in kaj delamo ter kaj so naši cilji, lahko pričnemo z vpeljavo sprememb, ki bodo naredile naše delo bolj optimalno in efektivno.

### 5 Uvedba zank za povratne informacije (ang. *Implement feedback loops*)

Kanbanov model spodbuja majhne, konstantne in inkrementalne spremembe, ki izboljšujejo razvojni proces. Ko ima ekipa skupno razumevanje politik dela, procesov razvoja in tveganj, je bolj verjetno, da lahko zgradi skupno razumevanje problemov in predlaga ukrepe za izboljšanje, ki jih dosežejo s konsenzom.

Za dosego teh ciljev je potrebno implementirati zanke, s katerimi lahko pridobimo povratne informacije, kako poteka naše delo in razvoj. Povratno informacijo lahko razumemo kot vse informacije, ki nam dajejo vpogled, kako potekajo procesi razvoja, delo, razumevanje znotraj ekipe itn. Če nimamo vzpostavljenih teh mehanizmov, ne moremo razumeti in izboljšati poteka dela in razvoja.

Kanbanov model razvoja uporabljajo tudi v podjetju Zemanta [16] pri eni izmed razvojnih ekip. Razlog za implementacijo so našli ravno v tem, da omogočijo transparentnost v razvoju ter tako vključijo naročnika kot del razvojne ekipe.

Za vizualizacijo uporabljajo spletno orodje Trello [17] in imajo stolpce za zahtevke dela razdeljene na »Incoming«, »Waiting«, »Doing«, »Review« in »Done«. Preden pa vnesejo vse zahtevke v Trello, naredijo na skupnem sestanku analizo in določijo zahtevke, potrebne za razvoj produkta ali storitve.

Planiranje dela ne poteka vnaprej, temveč poskušajo iz posameznih manjših razvojnih »zgodb« zgraditi večjo »zgodbo«. Tako poteka celoten razvojni proces po pristopu

gradnje produkta od spodaj navzgor. To omogoča konstantno prilagajanje produkta kupcem in efektivno razporeditev virov za razvoj.

Kako se Kanban razlikuje od Scruma?

Obe metodi imata enak cilj, in sicer izdati različico programske opreme čim prej ter čim bolj pogosto. Metodi zahtevata visoko sodelovanje in samoupravljanje v ekipah. Obstajajo pa razlike med obema pristopoma [15]:

| Kanban | Scrum |
|---|---|
| Ni vnaprej predpisanih vlog. | Vnaprej določene vloge (skrbnik metodologije, predstavnik naročnika, član razvojne ekipe). |
| Neprekinjeno izdajanje različic. | Šprint (izdajanje različic je odvisno od dolžine šprinta). |
| Zahtevki gredo posamezno skozi razvojni cikel. | V razvojni proces ali šprint gre skupek zahtevkov iz plana teka (ang. *Sprint backlog*). |
| Spremembe so dovoljene kadarkoli. | Med šprintom spremembe niso dovoljene. |
| Pristop je bolj primeren za operativna okolja z visoko stopnjo variabilnosti in prioritet. | Pristop je bolj primeren v situacijah, v katerih lahko delo prioritiziramo kot skupke zahtevkov (ang. *batch*), ki jih lahko pustimo za kasnejši razvoj. |

Iz zgoraj naštetih lastnosti ter temeljev pristopa lahko zapišemo nekaj prednosti Kanbana, sicer:

- uporablja krajše razvojne cikle in tako lahko ekipe dostavijo nove funkcionalnosti hitreje,
- je zelo odziven na spremembe,
- ko se prednostne naloge spreminjajo zelo pogosto, je Kanban idealen,
- zahteva manj virov za organizacijo ter začetek uporabe,
- zmanjšuje količine odpadkov in odstranjuje dejavnosti, ki ne prinašajo vrednosti skupini/oddelku in organizaciji,
- omogoča hitre povratne informacije, ki izboljšujejo možnosti za hitrejši razvoj, vključenost vseh članov ekipe ter skupno motivacijo.

# 3 UPORABA AGILNIH METODOLOGIJ V SLOVENSKIH START-UP PODJETJIH

V analitičnem delu diplomskega dela si bomo pogledali, kakšne metodologije uporabljajo slovenska start-up podjetja pri razvoju programske opreme in kakšne so prednosti in slabosti uporabe agilnih metodologij.

V zadnjih dveh letih zaradi ekonomskih razlogov vse več posameznikov vstopa v svet podjetništva. Razlog za to je predvsem močno zmanjšanje števila kariernih priložnosti, predvsem za mlajšo generacijo. Prav tako je trend računalništva v oblaku odprl veliko možnosti za nove ideje, produkte ter storitve. To je povzročilo velik porast za ustanovitev start-up podjetij tudi v Sloveniji.

Po podatkih Zavoda Republike Slovenija za zaposlovanje število brezposelnih v Sloveniji narašča [32]. To je eden izmed dejavnikov, da se vse več ljudi odloča za ustanovitev start-up podjetja. V prvi četrtini leta 2013 se je v primerjavi s prvo četrtino v letu 2012 v Sloveniji število start-up podjetij povečalo za 7,8 odstotka po raziskavi, ki jo je opravilo podjetje Bisnode [14].

V članku na spletni strani Silicon Gardens je navedeno, da gradi slovenski krog start-up podjetij več kot 1400 ljudi, zaposlenih v več kot 150 podjetjih. Tukaj so vključene le pravno registrirane oblike podjetij, med tem ko je projektnih start-up skupin še mnogo več. Prav tako spletna stran navaja, da so slovenska start-up podjetja v svetu v letih 2007 in 2013 zbrala več kot 53 milijonov dolarjev kapitala za njihovo delovanje. Na omenjeni spletni strani lahko tudi vidimo, da je slovensko tehnološko start-up okolje zelo raznoliko. Približno 35 podjetij se je po podatkih zgoraj omenjenega vira do leta 2013 ukvarjalo z razvojem programske opreme kot storitev oziroma SaaS (ang. *Software as a Service*), približno 25 podjetij se je ukvarjalo z razvojem mobilnih aplikacij, sledi razvoj spletnih trgovin ter strojne opreme [18].

Iz leta v leto število start-up podjetij narašča [14] in želeli smo preveriti, ali je uspeh start-up podjetij, predvsem tehnoloških, pogojen z uporabo agilnih metodologij. Uporaba agilnih metodologij omogoči start-up podjetju, da razvije produkt, ga čim prej predstavi na trgu in si tako zagotovi pomembne povratne informacije na trgu. Ker je danes tudi konkurenčnost na trgu vsa večja, je ravno zgoraj omenjen razlog eden izmed poglavitnih za izdelavo tega diplomskega dela. Vse več je v slovenskem start-up prostoru

različnih programov, start-up šol in pospeševalnikov, ki učijo razvojne ekipe agilnih metodologij in uporabe različnih orodij za izboljšanje njihovega uspeha.

Razlog za analizo uporabe agilnih metodologij v slovenskem tehnološkem start-up svetu glede na zgoraj naštete razloge je upravičen. V ta namen smo izvedli anketo, s katero smo vprašali slovenska start-up podjetja in start-up projekte, kako poteka njihov razvoj.

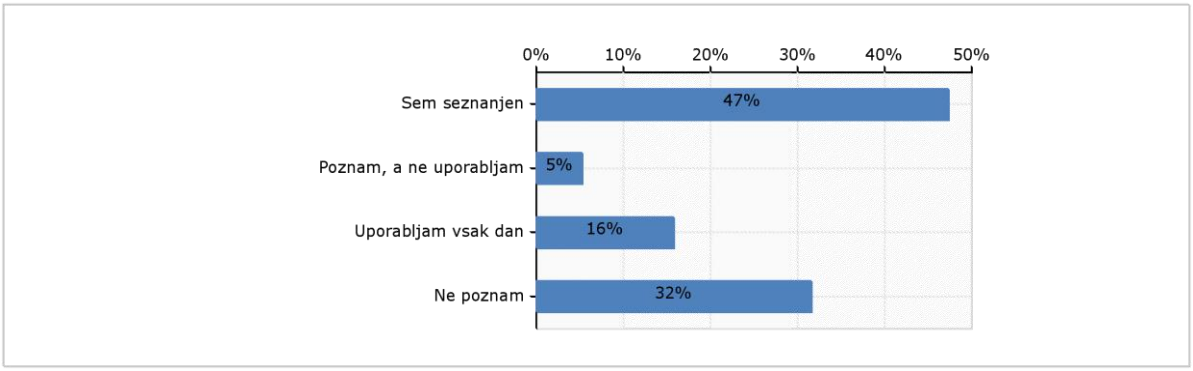
Želeli smo preveriti odgovore na naslednja vprašanja:

- Ali so ekipe, ki sodelujejo v start-up projektih, seznanjene z agilnimi metodologijami?
- Zakaj se start-up podjetja odločajo za uporabo agilnih metodologij?
- Kolikšen odstotek start-up podjetij uporablja za svoj razvoj agilne metodologije?
- Če projekt ne uporablja agilnih metodologij, na kakšen način razvijajo svoj produkt?
- Katere agilne metodologije so najbolj prepoznavne?
- Ali start-up projekti, ki uporabljajo agilne metodologije, dosegajo večjo učinkovitost pri delu oz. pridejo hitreje do končnega produkta?
- Kateri so problemi in dejavniki, s katerimi se srečujejo pri uporabi agilnih metodologij?
- Katere so slabosti in prednosti uporabe agilnih metodologij?
- Katera so orodja, ki jih uporabljajo pri svojem delu?

Za potrebe raziskave smo anketirali 19 start-up projektnih skupin in podjetij, fokusirali smo se predvsem na tiste, ki so udeleženi v kateri izmed slovenskih start-up šol. Med anketiranimi so sodelovala start-up podjetja, ki so mlajša od enega leta, in tudi tista, ki so že uveljavljena, vendar jih zaradi želje po anonimnosti ne bomo izpostavljali.

Vzorec sestavljajo podjetja, ki so udeležena v programih start-up šol na področju Ljubljane, saj je start-up okolje tu najbolj razvito (start-up šole Hekovnik, Ustvarjalnik, Tehnološki park itn.) [19]. Ekipe anketiranih v povprečju sestavlja od 3 do 5 članov, povprečna starost je med 25 in 29 let. Povprečna formalna izobrazba je pri malo manj kot polovici ekip (40 %) univerzitetna, medtem ko je tretjina anketiranih še študentov. Več o vzorcu sledi na koncu ankete.

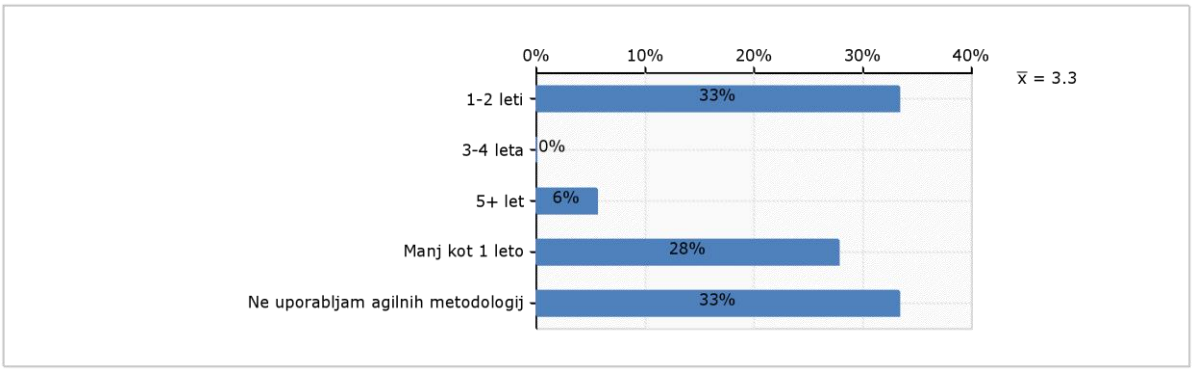
1. Ali poznate agilne metodologije razvoja programske opreme? (n = 19)




**Graf 1: Seznanjenost z agilnimi metodologijami pri razvoju programske opreme**

Skoraj polovica anketiranih pozna agilne metodologije razvoja programske opreme, nekaj več kot četrtina vprašanih pa agilne metode že uporablja. Vseeno ostaja tretjina ljudi, ki ne pozna metod za agilen razvoj programske opreme.

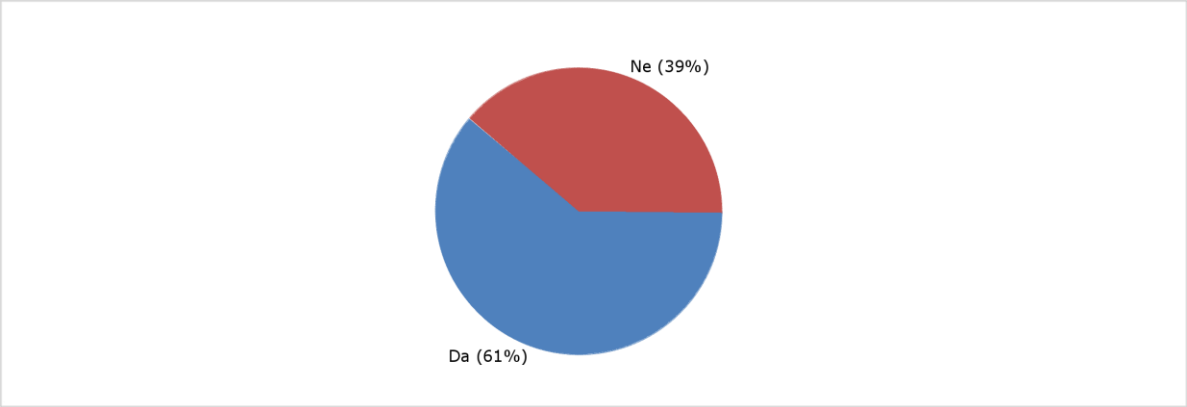
2. Odgovorite, če ste odgovorili pritrdilno na prejšnje vprašanje. Koliko let izkušenj imate z razvojem po agilnih metodologijah? (n = 18)




**Graf 2: Izkušnje z razvojem programske opreme po agilnih metodologijah**

Tretjina anketiranih ima že eno do dve leti izkušenj z razvojem programske opreme po agilnih metodologijah, medtem ko je skoraj tretjinski delež seznanjen s tovrstnim razvojem manj kot eno leto. To tudi nakazuje, da postaja uporaba agilnih metodologij vse bolj pogosta. Povprečna uporaba agilnih metodologij znaša 3,3 leta izkušenj. Odgovori nakazujejo, da je še vedno potencial za promocijo o uporabnosti agilnih metodologij pri slovenskih start-up podjetjih.

3. Ali uporabljate pri vašem projektu agilne metodologije? (n = 18)




**Graf 3: Uporaba agilnih metodologij pri trenutnih projektih**

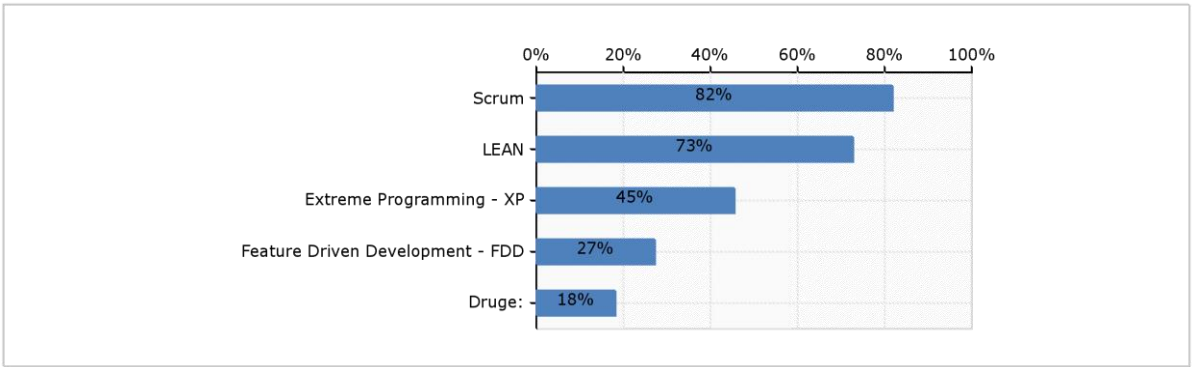
Skoraj dve tretjini vprašanih pri svojih projektih pri razvoju programske opreme že uporablja agilne metodologije. Ostali delež sestavljajo ekipe, ki so manjše od treh članov in se jim ne zdi smiselno vpeljati uporabo agilne metodologije za potrebe njihovega razvoja.

4. Če za razvoj programske opreme ne uporabljate agilnih metodologij, v nekaj besedah napišite, kako poteka vaše delo.

Kot glavne razloge za neuporabo agilnih metodologij so vprašani navedli:

- »Ne vem, kaj je to.«
- »Skušamo se držati neke svoje variacije, ki je kombinacija kanban in lean metodologije, a pri tem nismo povsem dosledni, predvsem ker posamezne ekipe pogosto štejejo le enega člana.«
- »Sem računalniško neizobražen, bom pa pri razvoju svojega produkta potreboval nekoga, ki to znanje ima.«
- »Zaenkrat delam sam, tako da se aplikacije pišejo straight forward.«

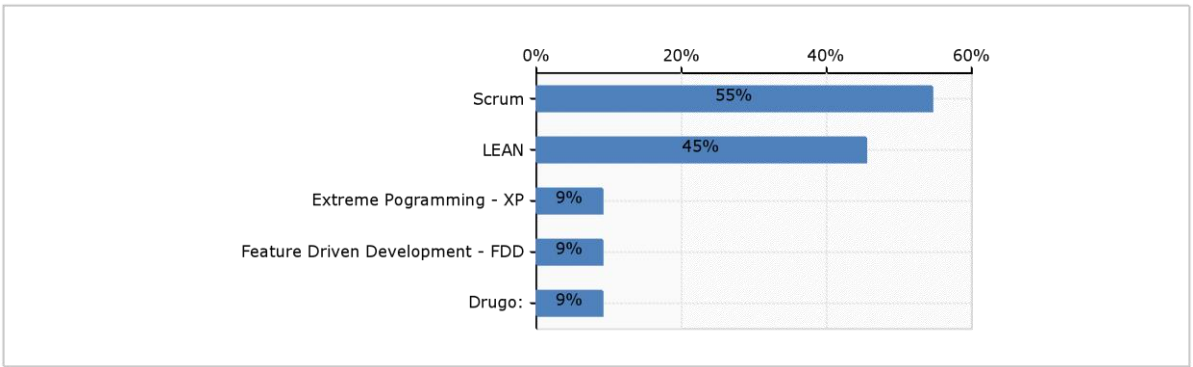
5. Katere izmed agilnih metodologij poznate ali ste vsaj slišali za njih? (n = 11) Možnih je več odgovorov.




**Graf 4: Poznavanje agilnih metodologij**

Najbolj poznana agilna metodologija pri vprašanih je Scrum, sledi ji metodologija Lean, nato ekstremno programiranje ter FDD. Kot druge metodologije so vprašani navedli še uporabo Kanban, TDD (ang. *Test Driven Development*) in BDD (ang. *Behavior-Driven Development*) metodologije. Glede na pridobljene podatke prevladujeta metodologiji Scrum in LEAN, ki se osredotočata predvsem na to, da podjetju omogočita predstaviti produkt čim prej na trgu potencialnim kupcem.

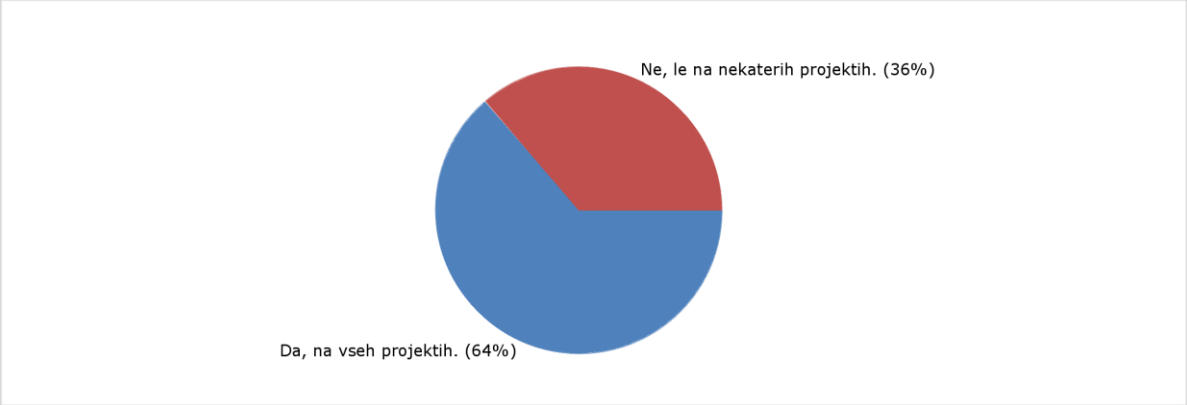
6. Katero izmed naštetih metodologij uporabljate pri vašem projektu? (n = 11) Možnih je več odgovorov.




**Graf 5: Uporaba agilnih metodologij pri trenutnih projektih**

Pri vprašanih sta najbolj razširjeni uporabi metodologij Scrum in Lean, sledi metodologija ekstremnega programiranja, FDD ter druge. Menim, da bo tudi v prihodnosti zelo velik poudarek na uporabi metodologij Scrum in LEAN, saj postaja konkurenčni trg vse večji in s tem čas za razvoj novih produktov vse manjši. Z uporabo omenjenih tehnologij se lahko tako slovenska start-up podjetja postavijo ob bok svetovni konkurenci, saj pri tehnoloških start-up podjetjih postane trg kar cel svet.

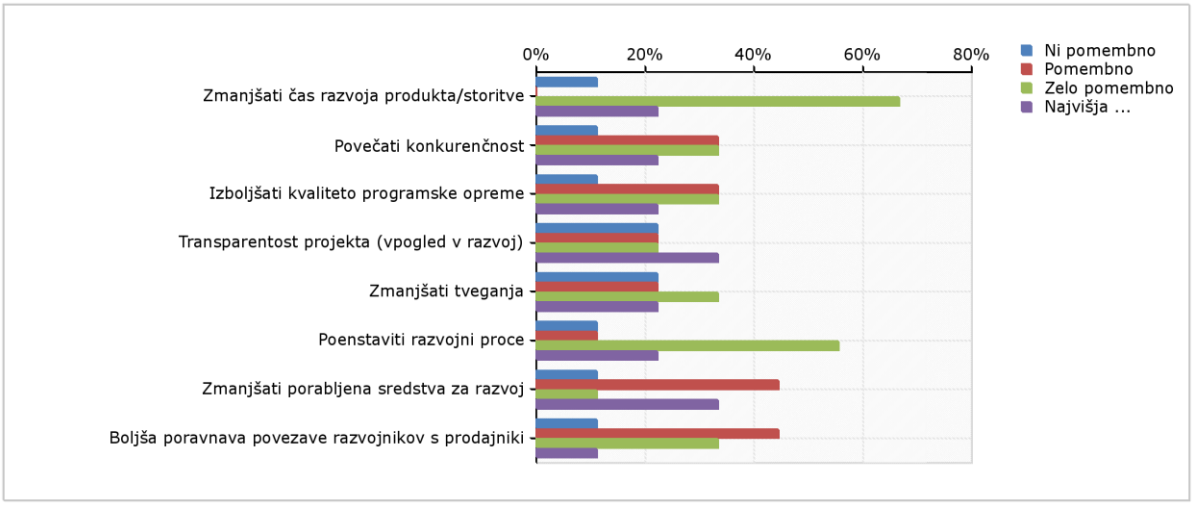
7. Ali omenjeno metodologijo uporabljate pri vseh projektih? (n = 11)




**Graf 6: Uporaba agilnih metodologij pri vseh projektih**

Malo manj kot dve tretjini vprašanih uporabljajo agilne metodologije pri vseh projektih, medtem ko dobra tretjina uporablja druge agilne metodologije. Sedem podjetij ima trenutno aktiven le en projekt. Menim, da glede na rezultat, še vedno obstajajo ekipe, ki še niso našle metodologije, ki bi najbolj ustrezala njihovemu načinu dela.

8. Spodaj po prioriteti razvrstite, zakaj uporabljate agilne metodologije. (n = 9)

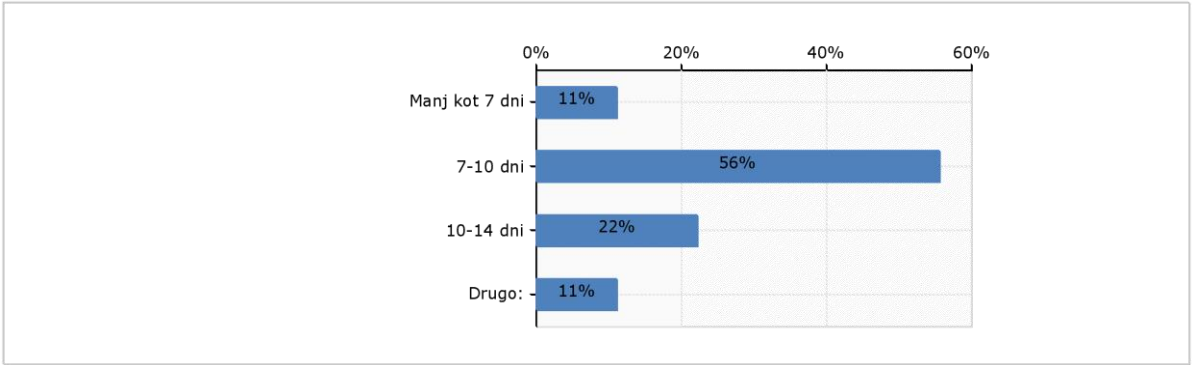



**Graf 7: Razlogi za uporabo agilnih metodologij**

Najvišjo prioriteto so vprašani postavili transparentnosti projekta ter manj porabljenim sredstvom za razvoj programske opreme. Velika večina vprašanih želi predvsem zmanjšati čas za razvoj produkta in storitev, to velja tudi za poglavitni razlog, zakaj so se odločili za uporabo agilnih metodologij. Naslednji zelo pomemben dejavnik, ki vpliva na uporabo agilnih metodologij, je višina porabljenih sredstev za razvoj. Zanimiv podatek je tudi, da večino sredstev financiranja pridobijo tehnološka start-up podjetja predvsem v

tujini [18]. Vprašani si želijo tudi boljšo povezavo med razvojno in prodajno ekipo, iz česar lahko sklepamo, da želijo predvsem izboljšati svojo transparentnost in interno komunikacijo, kar lahko razberemo tudi iz rezultatov.

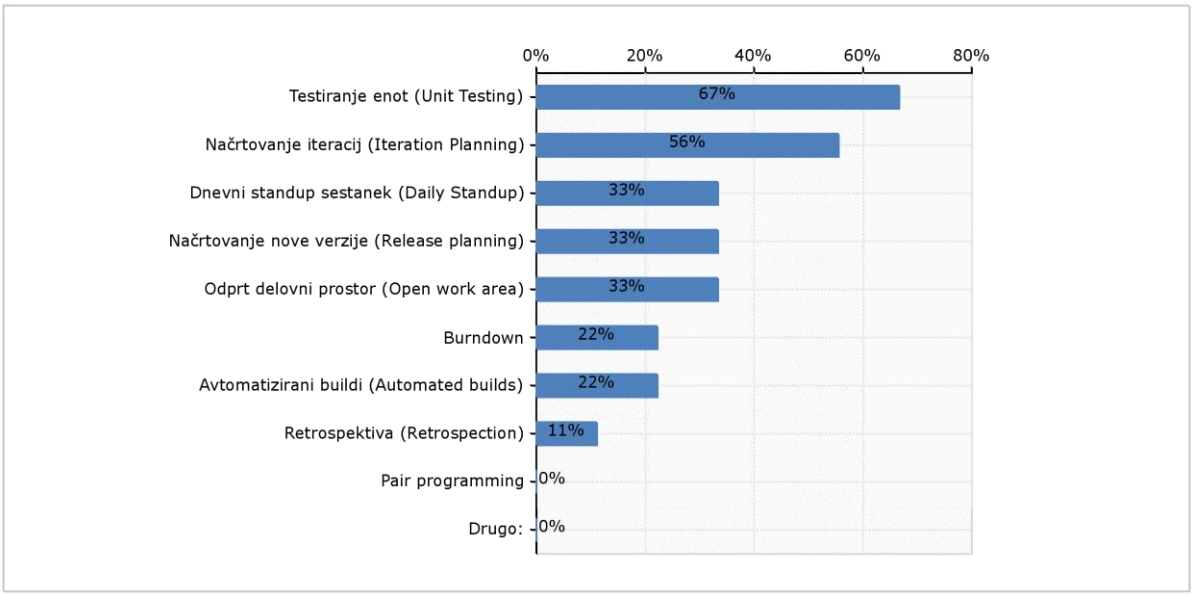
9. Kako dolg je vaš razvojni cikel oziroma iteracija? (n = 9)




**Graf 8: Dolžina razvojnega cikla**

Razvojni cikel pri več kot polovici vprašanih traja od sedem do deset dni, medtem ko pri dobrih 20 odstotkih traja od deset do štirinajst dni. Krajši razvojni cikel (od sedem do deset dni) se tudi zdi najbolj smiselna izbira, saj želijo razvojne ekipe čim prej razviti produkt, ki ga lahko ponudijo na trgu in si s tem zagotovijo povratne informacije.

10. Katere izmed naštetih tehnik in metod uporabljate pri svojem delu? (n = 9) Možnih je več odgovorov.

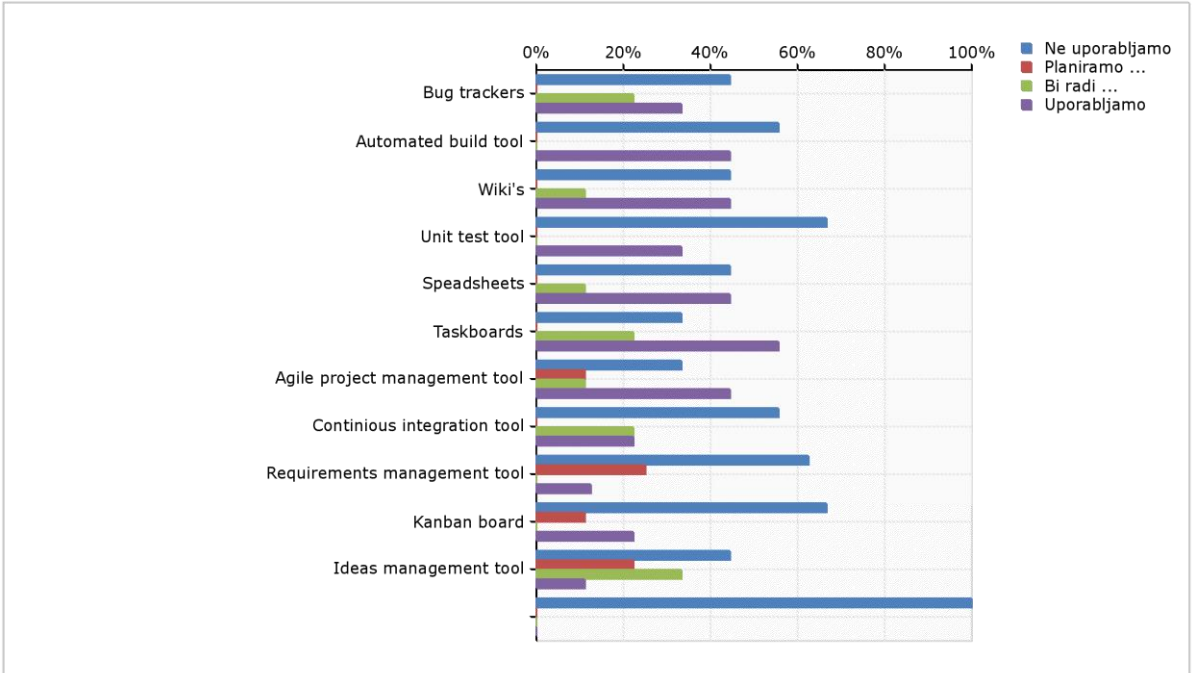



**Graf 9: Uporaba tehnik in metod pri razvoju**

Največji odstotek vprašanih uporablja tehnike testiranja enot ter načrtovanje iteracij, saj želijo na trgu predstaviti že delujoč produkt s čim manj pomanjkljivostmi. Sledijo tehnike dnevnih »stand-up« sestankov, načrtovanja novih verzij ter uporaba odprtega delovnega prostora. Kot velik trend se pojavlja pri slovenskih start-up podjetjih tudi delo v tako imenovanem »co-workingu« ali odprtem delovnem prostoru v start-up šolah, saj omogoča razvijalcem hitro pridobiti povratno informacijo in najti pomoč pri ostalih posameznikih, ki imajo bolj podrobno znanje glede razvoja produktov in storitev.

Najmanjši odstotek vprašanih je dosegla metoda programiranja v paru (ang. *Pair programming*). Menim, da so poglavitni razlogi za to pomanjkanje virov – predvsem kadrovskih, saj želijo podjetja doseči čim večjo efektivnost pri razvoju in s tem tudi vstop na trg. Zagovorniki metode programiranja v paru zagovarjajo, da je kvaliteta kode pri programiranju v paru toliko boljša, da je potrebno manj testiranja in odpravljanja napak kasneje v življenjskem ciklu produkta. Posledično se izboljša tudi produktivnost.

11. Katera izmed naštetih orodij uporabljate pri vašem delu? (n = 9)

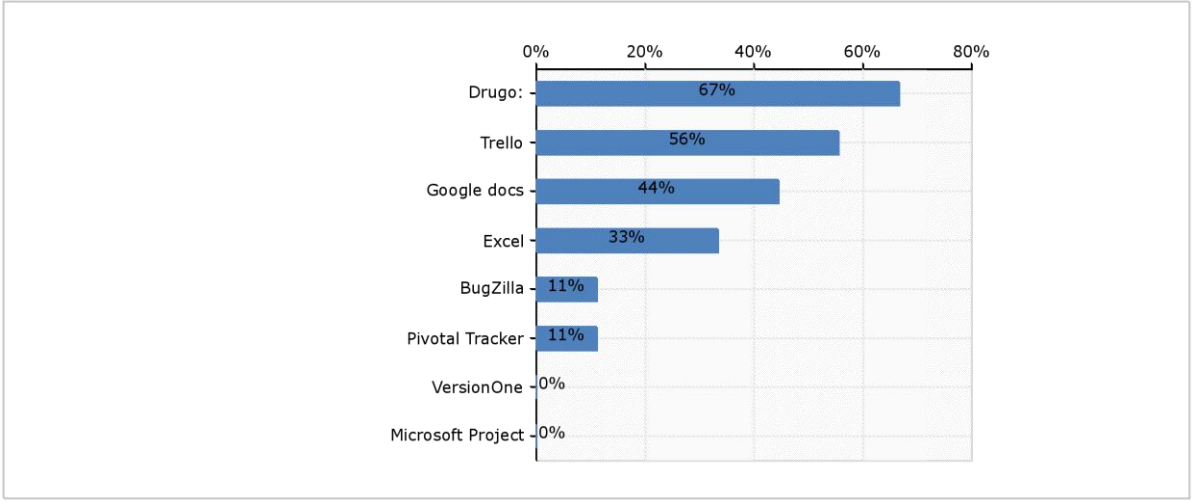



**Graf 10: Uporaba orodij pri razvoju programske opreme**

Največje število vprašanih uporablja »Taskboard« oziroma prostor za dodeljevanje in sledenje zadolžitvam. Sledi uporaba projektnih managerskih orodij, avtomatiziranih »build« orodij, wikijev ter razpredelnic. Glede na rezultate želijo ekipe doseči večjo transparentnost svojih projektov, s tem pa tudi bolj jasne procese razvoja.

Nihče od vprašanih ne uporablja orodja za management idej, vendar ima več kot tretjina vprašanih v načrtu začetek uporabe le-teh. Predvidevam, da je razlog za to pomanjkanje virov ter časa, ki bi ga lahko podjetja namenila temu.

12. Katera specifična orodja uporabljate pri delu? (n = 9) Možnih je več odgovorov.




**Graf 11: Uporaba specifičnih orodij pri razvoju programske opreme**

Več kot polovica vprašanih uporablja za lažje delo orodje Trello, sledi uporaba Google orodij ter Excel. Kot druga uporabljena orodja so vprašani navedli še orodja TeamBox, TeamWorkPM, GitHub Issue Tracker, Podio, Asana, Jura in SVN.

Razvojne ekipe iščejo in uporabljajo predvsem orodja, ki temeljijo na preprosti uporabi, saj ne želijo izgubljati časa za spoznavanje novih kompleksnih orodij. Prav k temu napeljuje tudi metodologija LEAN in uporaba pristopa Kanban. Tudi v prihodnosti se lahko pričakuje porast uporabe preprostih orodij za razvoj in upravljanje procesov.

13. Kateri je največji problem pri uporabi agilnih metodologij?

Vprašani so navedli naslednje razloge kot največji problem pri uporabi agilnih metodologij:

- »Nadzor.«
- »Vpeljava v delovni proces.«
- »Neizkušenost razvijalcev z agilnimi metodologijami – težko je mešati tradicionalne in agilne metodologije.«
- »Prepričati ekipo v uporabo.«
- »Vsi člani ekipe ne poznajo pristopov agilnih metodologij dovolj dobro.«
- »Implementacija agilne metodologije v podjetje.«

- »Nepoznavanje uporabe agilnih metodologij in posledično slaba volja.«

14. Na kakšen način ocenjujete prednosti uporabljene metodologije?

Vprašani ocenjujejo prednosti uporabe agilne metodologije:

- »Hitrejši razvoj, koda je vedno mlada, early releases.«
- »Boljšo preglednost.«
- »Hiter razvojni cikel, nižji stroški razvoja, iskanje product-market fita«.
- »Večjo hitrost razvoja in kvalitete.«
- »Dober pregled nad projektom, lahek pregled nad verzijami, dobro znane prioritete«.

Vsi komentarji vprašanih nakazujejo, da želijo svoje delo narediti bolj transparentno ter pripeljati kvaliteten produkt čim prej do svojih kupcev. Ravno zaradi naštetih razlogov se pričakuje tudi vse bolj pogosto uporabo agilnih metodologij.

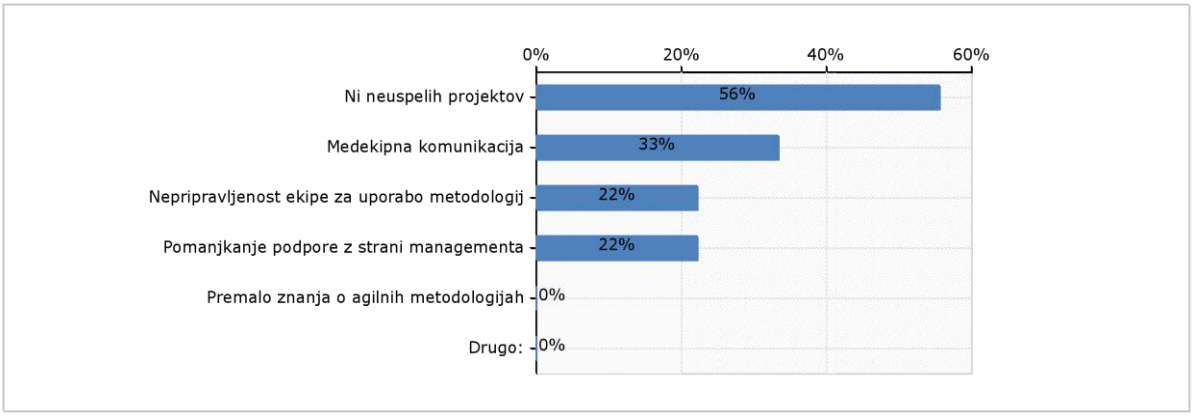
15. Na kakšen način ocenjujete slabosti uporabljene metodologije?

Vprašani navajajo naslednje slabosti uporabe agilne metodologije:

- »Včasih je potrebno veliko časa za planiranje.«
- »Nezadovoljstvo razvojne ekipe in projektnih vodij.«
- »Slabo sprejetje agilnih metodologij v ekipi.«
- »Dodatno delo.«

Glavni razlogi slabosti nakazujejo na nepopolno poznavanje in uporabnost agilnih metodologij med razvijalci. Sklepam lahko, da bi ekipe dosegle večjo efektivnost, če bi se udeležili izobraževanj o uporabnosti agilnih metodologij ter orodij, ki bi naredila upravljalni in razvojni proces bolj preprost.

16. Če imate kakšen neuspeli projekt, kjer ste uporabili agilne metodologije, kaj so glavni razlogi? (n = 9) Možnih je več odgovorov.

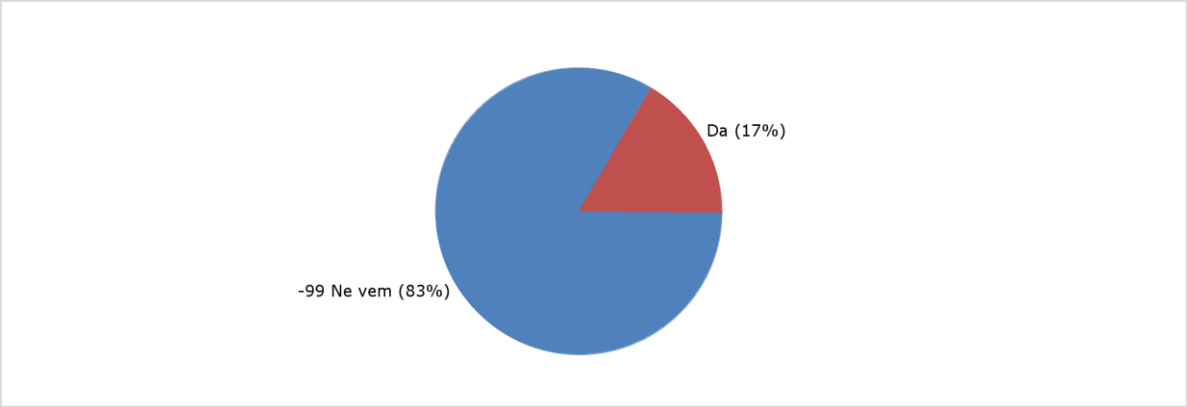



**Graf 12: Glavni razlogi za neuspeh projektov pri uporabi agilnih metodologij**

Najbolj pogost razlog za neuspeh projekta so vprašani navedli med-ekipno komunikacijo, sledita nepripravljenost ekipe za uporabo agilnih metodologij in pomanjkanje podpore s strani managementa.

Sklepamo lahko, da bi z vizualizacijo in transparentnostjo razvoja dosegli boljši pregled nad produktom in s tem dosegli boljšo komunikacijo. Prav tako bi lahko z uporabo preprostih orodij za zbiranje povratnih informacij in upravljanje procesov razvoja znotraj ekipe dosegli večjo produktivnost.

17. Ali želite implementirati agilne metodologije? (n = 6)



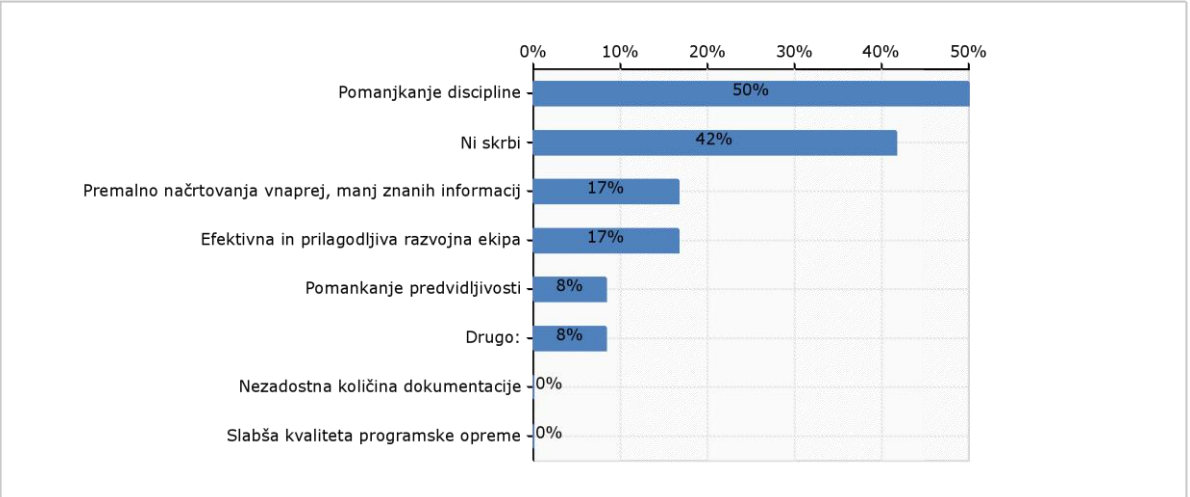

**Graf 13: Implementacija agilnih metodologij**

Izmed šestih vprašanih jih velika večina ni prepričanih, ali bi želeli implementirati uporabo agilnih metodologij v razvojni proces, medtem ko jih 17 % to želi storiti. Menim, da še vedno obstaja velik potencial za uporabo agilnih metodologij v slovenskem start-up okolju. Začeti pa bi bilo treba z izobraževanjem razvojnih ekip in jih naučiti, na kakšne načine lahko agilne metodologije uporabljajo, kaj so njihove prednosti in kako jih implementirati.

18. Kaj vas ustavlja pri implementaciji? (n = 1) Možnih je več odgovorov.

Edini razlog, da ekipa ne implementira ene izmed agilnih metodologij v razvojni proces, je, da je v ekipi samo en član.

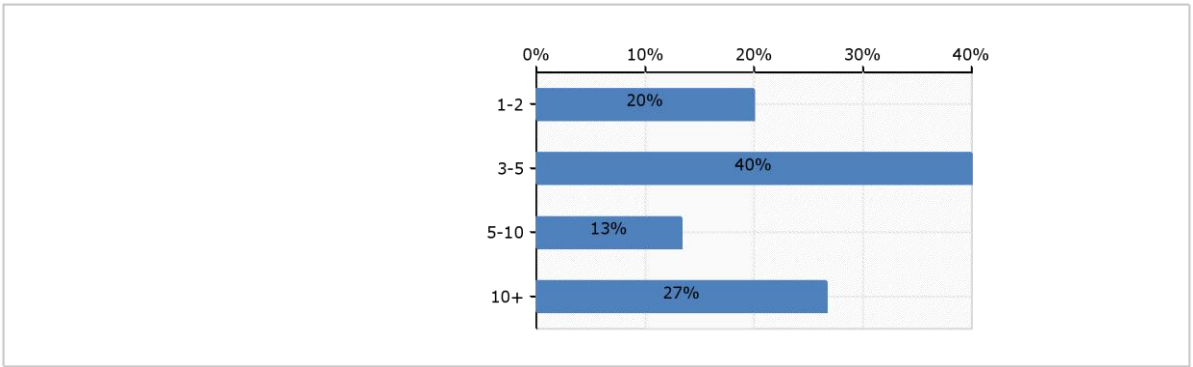
19. Katere so vaše največje skrbi glede implementacije agilnih metodologij? (n = 12) Možnih je več odgovorov.




**Graf 14: Skrbi pri implementaciji agilnih metodologij**

Polovica vprašanih ima največ skrbi pri implementaciji agilnih metodologij s pomanjkanjem discipline. Malo manj kot polovica nima skrbi pri implementaciji agilnih metodologij.

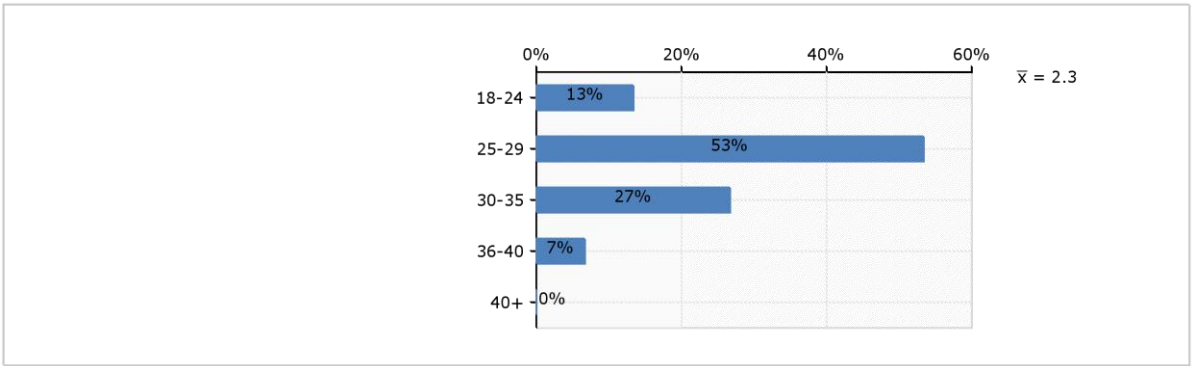
20. Število ljudi v vaši ekipi. (n = 15)




**Graf 15: Število ljudi v ekipi**

Velikost ekipe je v malo manj kot polovici primerov od tri do pet ljudi. Tretjina ekip ima več kot 10 ljudi, petina ekip pa je sestavljena le iz enega ali dveh članov.

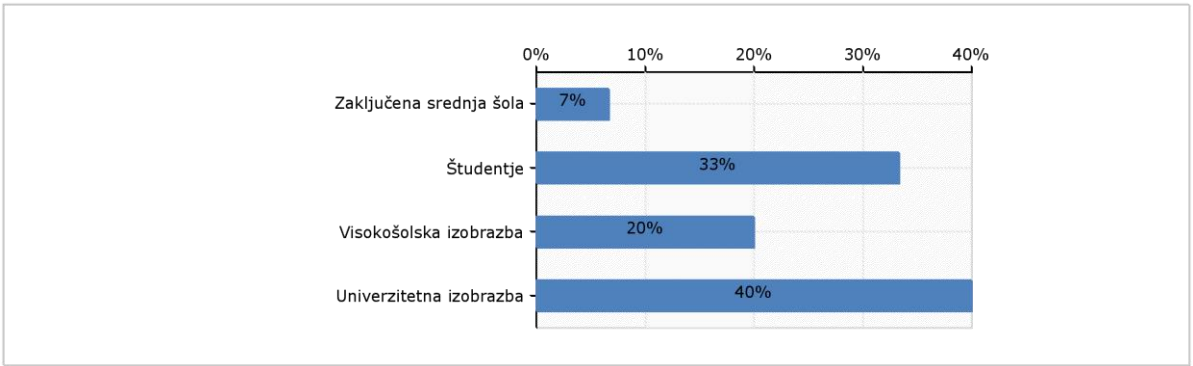
21. Povprečna starost vaše ekipe. (n = 15)




**Graf 16: Povprečna starost ekip**

Povprečna starost več kot polovice ekip je med 25 in 29 let, s slabo tretjino sledi starostna skupina med 30 in 35 let. Sklepamo lahko, da sestavljajo slovenska start-up podjetja v večini predvsem mladi do 30 oziroma 35 let.

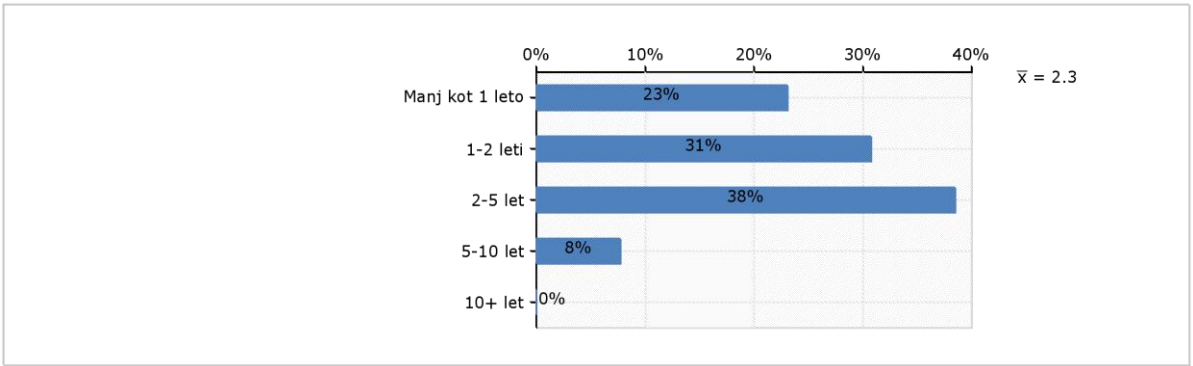
22. Vaša povprečna formalna izobrazba ekipe. (n = 15)




**Graf 17: Povprečna formalna izobrazba ekip**

Povprečna stopnja izobrazbe v ekipah je v 40 % univerzitetna stopnja, tretjino ekip sestavljajo v povprečju še študenti, ena petina ekip pa ima opravljeno visokošolsko izobrazbo.

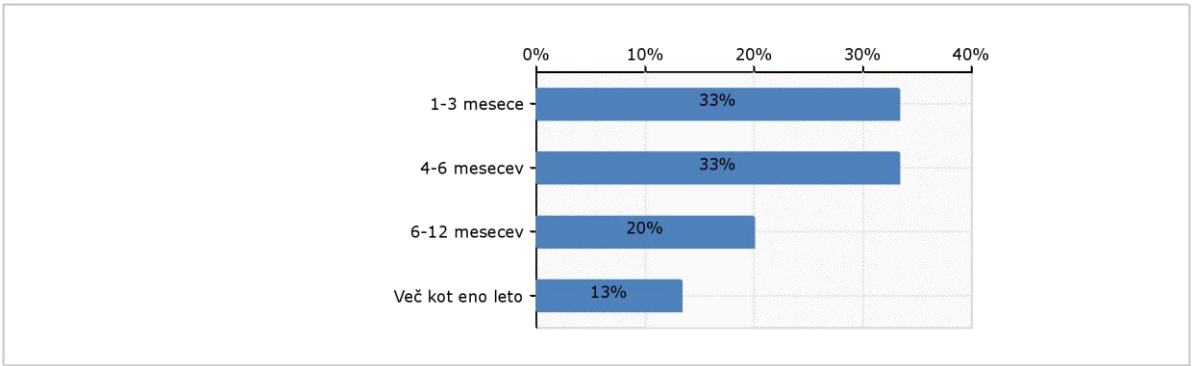
23. Koliko let izkušenj imate z razvojem programske opreme? (n = 13)




**Graf 18: Število let izkušenj z razvojem programske opreme**

Povprečno število let izkušenj z razvojem programske opreme je med vprašanimi 2,3 leti. Največji odstotek (38 %) ima od 2 do 5 let izkušenj, dobra petina ljudi pa manj kot eno leto.

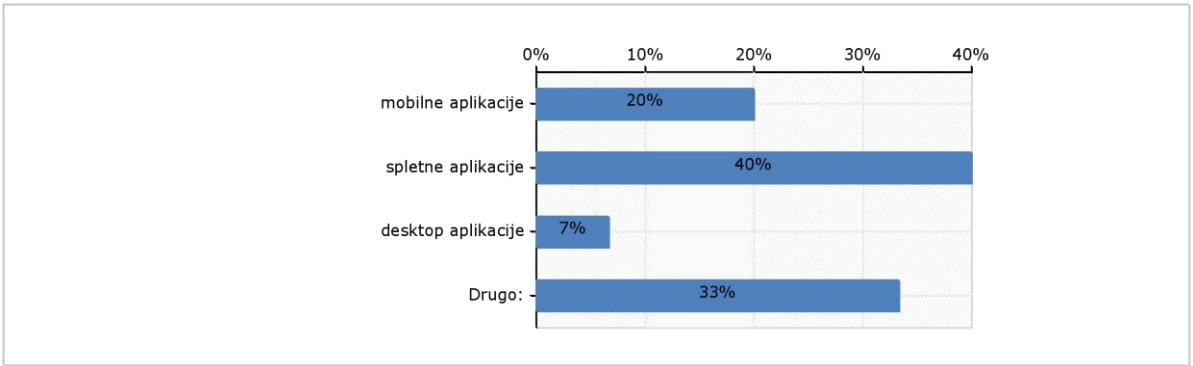
24. Koliko časa že razvijate vaš produkt oz. sodelujete skupaj pri projektu? (n = 15)




**Graf 19: Čas razvijanja produkta**

Ena tretjina vprašanih razvija produkt manj kot 3 mesece, ena tretjina manj kot 6 mesecev, ena petina vprašanih razvija produkt manj kot 12 mesecev in preostali več kot eno leto.

25. Smer razvoja programske opreme. (n = 15)




**Graf 20: Smer razvoja programske opreme**

Največji odstotek vprašanih se ukvarja z razvojem spletnih aplikacij, petina vprašanih z razvojem mobilnih aplikacij, 7 % pa z namiznimi aplikacijami.

Vsi vprašani, ki so navedli kot odgovor drugo, se ukvarjajo s spletno trgovino in kombinacijo razvoja mobilnih in namiznih aplikacij. Na že omenjeni spletni strani Silicon Gardens [14] si lahko ogledamo, s katerimi smermi razvoja programske opreme so se ukvarjala slovenska start-up podjetja do leta 2013. Menim, da bo narasel trend razvoja mobilnih aplikacij, spletnih trgovin ter razvoj programske opreme z uporabo velikih podatkov (ang. *big data*) [20]. Nekaj izmed svetovnih trendov za leto 2014 pa si lahko pogledamo na spletni strani tech.co [21, 22].

# 4 SKLEPNE UGOTOVITVE

Uporaba agilnih metodologij, glede na anketo, postaja v slovenskem start-up okolju vse bolj pogosta, saj jih uporablja 61 % vprašanih start-up podjetij (glej graf 3). Razvidno je, da je polovica vprašanih seznanjena z vsaj eno izmed agilnih metodologij (glej graf 1). Kot najbolj pogost odgovor na vprašanje, zakaj uporaba agilnih metodologij ni del vsake projektne ekipe, je razlog, da ekipa šteje le enega člana. V procesu zbiranja podatkov sem opazil, da ekipe želijo uporabljati agilna orodja pri razvoju produktov. Glavni razlog je želja predstaviti produkt trgu čim prej ter zmanjšati porabo virov, predvsem finančnih.

Kot zelo pomemben razlog za uporabo agilnih metodologij pri razvoju sta dve tretjini vprašanih navedli (glej graf 7) zmanjšati porabljena sredstva za razvoj produkta, kot najpomembnejša dejavnika pa transparentnost projekta in čas, potreben za razvoj produkta. Iz tega lahko sklepamo, da uporaba agilnih metodologij skrajša razvojni čas, ki je potreben, da s produktom vstopimo na trg, prav tako zmanjšamo tudi vložena sredstva za realizacijo projekta.

Med anketiranjem sem ugotovil, da ekipe, ki ne uporabljajo agilnih metodologij za upravljanje procesov, vseeno uporabljajo orodja, ki izhajajo iz sveta agilnih metodologij. Med ta orodja uvrščamo tudi »Business Model Canvas« [23], ki ga uporabljajo v večini priznanih slovenskih start-up šolah, kot so Hekovnik, Start-up center TP in Ustvarjalnik [19]. Z uporabo orodij lahko razvojne ekipe prej postavijo svoj poslovni model in definirajo procese razvoja, kar jim omogoči hitrejši razvoj produkta.

V analizi poročila razvoja agilnih metodologij, ki jo vsako leto izvede podjetje VersionOne so bila v letu 2012 za uporabo pri razvoju programske opreme z uporabo agilnih metodologij največkrat priporočena orodja VersionOne (93 %) in LeanKit (92 %), medtem ko je bila uporaba Excela med najbolj pogosto uporabljenimi orodji (69 %) kot efektivno orodje za uporabo pri agilnem managementu [24].

Najbolj poznana je metodologija med anketiranimi slovenskimi start-up podjetji je Scrum (pozna jo 80 % vprašanih), takoj za njo pa sledi Lean metodologija (73 %). Lean metodologija se je predvsem v zadnjem obdobju izkazala kot zelo hitro rastoča pri uporabi v start-up podjetjih. Prav s pomočjo konceptov in procesov Lean metodologije so hitro zrastla podjetja, kot so npr. DropBox in Groupon. Omenjeni podjetji danes veljata za vodilni na svojem področju delovanja. Sklepam, da bo tudi v prihodnjih letih

uporaba pristopa, kot ga ponuja Lean metodologija, vse več v uporabi zaradi že zgoraj omenjenih razlogov.

Za najbolj uporabni tehniki pri razvoju programske opreme pri vprašanih podjetjih, ki uporabljajo agilne metodologije, veljata testiranje enot in načrtovanje iteracij (glej graf 9). Podjetja še vedno dajejo velik poudarek na kvaliteto produkta, preden ga predstavijo trgu, zato lahko sklepamo, da sta ti tehniki v slovenskem start-up okolju med vodilnimi. V svetovnem merilu sta bili po analizi, ki jo je opravilo podjetje VersionOne, med najbolj uporabljenimi orodji v porastu t.i. »Taskboardi« (+11 %) ter Kanban (+10 %), iz leta 2011 v leto 2012.

V pogovorih z eno izmed start-up šol sem tudi izvedel, da pri njih zelo spodbujajo uporabo orodij, ki jih ponuja metodologija Lean ter pristop Kanban. Cilj uporabe omenjenih pristopov je predvsem predstaviti produkt trgu in si tako pridobiti povratne informacije. S tem želijo spodbuditi razvoj produkta skupaj s trgom in tako omogočiti produkt, ki rešuje problem kupca. Posledično se tako zmanjšajo porabljeni viri ter čas za analizo samega trga.

Predpostavljam, da se bodo start-up podjetja v prihodnosti vse več odločala za vstop v katero izmed start-up šol. Na ta način bodo lahko hitro spoznala najbolj uporabne tehnike, orodja in metodologije za razvoj produktov in storitev. Nekaj predlogov za razvoj start-up okolja v Sloveniji je podala tudi iniciativa Start-up Slovenija [30]. Želijo si več promocije start-up podjetništva, podpore talentom, izobraževanja, infrastrukturo v obliki univerzitetnih in regijskih središč ter kapital, ki bo omogočal start-up podjetjem rast.

Sklepam, da je poznavanje agilnih metodologij v slovenskem start-up okolju že dobro, vendar še vedno ostaja potencial za promocijo, izobraževanje in spoznavanje. Glede na opravljeno analizo menim, da se bo vse več podjetij samoizobraževalo tudi na področju uporabe omenjenih tehnik in uporabe metod. Za hitrejši razvoj start-up podjetij je potrebno v Sloveniji več nepovratnega kapitala, ki bi lahko omogočil brezskrbno začetno pot mnogim podjetnikom. Predvsem bi se tako lahko podjetja bolj posvetila hitrejšemu in kvalitetnejšemu razvoju svojih produktov in storitev ter si tako priborila hitrejši vstop na trg.

# 5 LITERATURA IN VIRI

[1] (2014) Definicija slovenskega start-up podjetja. Dostopno na: http://www.podjetniskisklad.si/datoteke/Rezultati/Tekocirezultati2013/Razpisisklada31.12.2012.pdf

[2] (2014) Manifest agilnega razvoja programske opreme. Dostopno na: http://agilemanifesto.org/iso/sl/

[3] (2014) Scrum. Dostopno na: https://www.scrum.org/

[4] (2014) LSD – »vitek« razvoj programske opreme (Lean Software Development). Dostopno na: http://www.allaboutagile.com/7-key-principles-of-lean-software-development-2/

[5] (2014) XP – ekstremno programiranje (ang. *Extreme programming*). Dostopno na: http://www.extremeprogramming.org/

[6] (2014) AD – agilne tehnike podatkovnih baz (ang. *Agile Database Techniques*). Dostopno na: http://www.ambysoft.com/books/agileDatabaseTechniques.html

[7] (2014) AM – agilno modeliranje (ang. *Agile Modeling*). Dostopno na: http://www.agilemodeling.com/

[8] (2014) ASD – prilagodljiv razvoj programske opreme (ang. *Adaptive Software Development*). Dostopno na: http://www.exa.unicen.edu.ar/catedras/agilem/cap23asd.pdf

[9] (2014) Crystal. Dostopno na: http://sharif.edu/~ramsin/index_files/sdmlecture12.pdf

[10] (2014) FDD – funkcijsko voden razvoj programske opreme (ang. *Feature Driven Development*). Dostopno na: http://www.featuredrivendevelopment.com/

[11] (2014) DSDM – metoda dinamičnega razvoja sistemov (ang. *Dynamic Systems Development Method*). Dostopno na: http://www.dsdm.org/

[12] (2014) TDD – testno voden razvoj (ang. *Test-Driven Design*). Dostopno na: http://www.agiledata.org/essays/tdd.html

[13] Viljan Mahnič, Strahil Georgiev, Tomo Jarc: Poučevanje metode Scrum v sodelovanju s podjetjem za razvoj programske opreme

[14] (2014) Rast števila slovenskih start-up podjetij.
Dostopno na: http://mladipodjetnik.si/novice-in-dogodki/novice/pregled-stanja-gospodarstva-v-evropi-1.-cetrtletje

[15] (2014) Kaj je Kanban. Dostopno na: http://www.versionone.com/what-is-kanban/

[16] (2014) Zemana. Dostopno na: http://www.zemanta.com/

[17] (2014) Trello. Dostopno na: https://trello.com/

[18] (2014) Slovenski start-up ekosistem.
Dostopno na: http://www.silicongardens.si/ecosystem2013/

[19] (2014) Slovensko podporno okolje za start-upe.
Dostopno na: http://mladipodjetnik.si/novice-in-dogodki/novice/slovensko-podporno-okolje-za-startupe

[20] (2014) Veliki podatki. Dostopno na: http://www.bigdata-startups.com/big-data-trends-2014/

[21] (2014) Svetovni trendi, ki se pojavljajo v svetu iz leta 2012 do leta 2014. Dostopno na: http://tech.co/17-tech-startups-trends-will-emerge-2014-2013-12

[22] Svetovni trendi za razvoj start-up podjetja v letu 2014.
Dostopno na: http://tech.eu/features/169/technology-trends-2014-startups/

[23] Business Model Canvas. Dostopno na: http://www.businessmodelgeneration.com/

[24] Poročilo razvoja agilnih metodologij podjetja VersionOne. Dostopno na: http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf

[25] (2014) Metodologija Scrum. Dostopno na: http://scrummethodology.com/

[26] (2014) Wikipedia, Scrum.
Dostopno na: http://en.wikipedia.org/wiki/Scrum_%28software_development%29

[27] (2014) Agilni razvoj.
Dostopno na: http://www.versionone.com/agile101/agile_development.asp

[28] (2014) Blaž Kos in vitko podjetništvo.
Dostopno na: http://www.blazkos.com/vitko-startup-podjetje.php

[29] (2014) Pristopi Lean metodologije.
Dostopno na: http://www.allaboutagile.com/lean-principles-1-eliminate-waste/

[30] (2014) Iniciativa start-up Slovenija. Dostopno na:
http://www.startup.si/sl-si/iniciativa-startup-slovenija/programi-iniciative

[31] (2014) Agilno načrtovanje iteracije. Dostopno na:
http://www.agilehelpline.com/2011/04/agile-iteration-planning.html

[32] (2014) Brezposelnost v Sloveniji po podatkih Zavoda Republike Slovenije za zaposlovanje. Dostopno na:
http://www.ess.gov.si/trg_dela/trg_dela_v_stevilkah/registrirana_brezposelnost

[33] (2014) Prednosti razvoja z agilnimi metodologijami pred tradicionalnimi metodami. Dostopno na:
http://www.versionone.com/Agile101/Agile-Software-Development-Benefits/

[34] (2014) S-krivulja. Dostopno na:
http://www.agilehelpline.com/2011/04/agile-strategy-planning.html

[35] (2014) Agilno načrtovanje različice. Dostopno na:
http://www.agilehelpline.com/2011/04/agile-release-planning.html

[36] (2014) Povratna zanka zgradi – meri – uči se. Dostopno na:
http://lean.st/principles/build-measure-learn