UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

MAGISTRSKO DELO

MOŽNOST UVAJANJA VITKIH PRISTOPOV V ZAGONSKEM PODJETJU: PRIMER PODJETJA ZA RAZVOJ PROGRAMSKE OPREME

Ljubljana, september 2016 BOGDANA BREZNIK

IZJAVA O AVTORSTVU

Podpisana Bogdana Breznik, študentka Ekonomske fakultete Univerze v Ljubljani, avtorica predloženega dela z naslovom Možnost uvajanja vitkih pristopov v zagonskem podjetju: primer podjetja za razvoj programske opreme, pripravljenega v sodelovanju s svetovalcem prof. dr. Borutom Rusjanom

IZJAVLJAM

1. da sem predloženo delo pripravila samostojno;

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

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

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

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

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

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

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

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

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

V Ljubljani, dne 27. septembra 2016 Podpis študentke:_____

KAZALO

KAZALO SLIK

KAZALO TABEL

UVOD

Današnji svet želi vedno več dodane vrednosti za nižjo ceno. Na področjih, kjer je prisotna močna konkurenca, lahko uporaba »vitkega pristopa« pomeni ključno prednost pred ostalimi, saj z manj viri ustvarimo enako ali višjo dodano vrednost (Šinkovec, 2012).

Osnovna ideja vitkega pristopa je maksimiziranje vrednosti za kupca, pri hkratni minimizaciji izgub. To preprosto pomeni, da z uporabo manj virov ustvarjamo večjo vrednost za kupca. Da bi to dosegli, se mora spremeniti osredotočenje managementa. Z optimiziranja posameznih tehnologij, sredstev in vertikalnih oddelkov se mora management osredotočiti na optimiziranja celotnih tokov vrednosti izdelkov in storitev, ki pa potekajo horizontalno – od tehnologij, sredstev in oddelkov pa vse do strank (What is Lean?, 2016).

Izraz »**vitka**« je termin, ki jo je prvič uporabil John Krafcik v članku z naslovom »The Triumph of the Lean Production System«, objavljenem v MIT Sloan Management Review leta 1988. Krafcik opisuje Toyotin proizvodnji sistem (angl. *Toyota Production System*, v nadaljevanju TPS), ki je nastal v proizvodnji motornih vozil podjetja Toyota Motor Corporation in je predstavljal radikalno alternativo tradicionalni masovni proizvodnji avtomobilov in serijski proizvodnji (Holweg, 2007). V članku je Krafcik objavil prve rezultate mednarodne raziskave proizvodnje motornih vozil, ki so ga izvedli na Massachusetts Institute of Technology (MIT), izraz vitko pa je uporabil za opis pristopa proizvodnega managementa, ki uporablja manj virov, manj prostora, manj skladišč, manj delovnih ur (Krafcik, 1988, str.). Med zahodnimi proizvajalci pa se je zanimanje za »vitko proizvodnjo« razširilo šele z objavo knjige »The Machine that Change the World«, v kateri so Krafcikovi sodelavci Womack, Jones in Roos (1990) poudarili razlike med učinkovitostjo zahodnih proizvajalcev avtomobilov in učinkovitostjo Toyotine proizvodnje, ki so jo poimenovali **vitka proizvodnja.** Vitka proizvodnja je v industriji skoraj soglasno prepoznana kot »univerzalna metoda«, to je metoda, ki učinkovito odpravlja izgube in kontinuirano izboljšuje proizvodnjo, kot tudi preskrbovalne procese v tovarnah širom po svetu. (Womack & Jones 1996).

Po letu 1990 se je vitek pristop postopoma širil iz proizvodnje. Ta proces širitve je pospešilo tudi promoviranje uspešnih študij primerov na zahodu, kjer so **vitka načela** proizvodnje posnemali in prilagodili tudi v drugih industrijah (Hignes et al., 2004, str.). Ta načela so vključevala prepoznavanje vrednosti za kupca, obvladovanje toka vrednosti, razvoj proizvodnje po **načelu vlečenja** in popolno odpravo vseh vrst izgub v proizvodnem sistemu. Izraz **vitko razmišljanje** (angl. *lean thinking***)** zajema uporabo nabora vitkih praks, ki sta ga prav tako uvedla Womack in Jones (1996), uporablja pa se večkrat v povezavi s storitvenimi dejavnostmi. **Vitka filozofija** (angl. *lean phylosophy*) pa je filozofija managementa, ki je osnovana na TPS.

Metode vitke proizvodnje so začele nadomeščati konvencionalne metode tudi v storitvenih dejavnostih. V javnem sektorju je največ študij primerov iz področja zdravstva in vojske ter javne uprave. Vitko obvladovanje proizvodnje se je razvilo v celovit sistem managementa. Njegova učinkovita implementacija zajema spremembe v organizaciji podjetja, nov pristop k proizvodnji ali storitvam za kupce, visoko stopnjo treninga in izobraževanja zaposlenih od višjih managerjev do zaposlenih v proizvodnji (Arnheiter & Maleyeff, 2005). Vitko projektno vodenje se od tradicionalnega projektnega vodenja ne razlikuje samo po ciljih, ki jih zasleduje, ampak tudi po strukturi njegovih faz, razmerju med fazami in udeleženci v vsaki fazi (Ballard & Howell, 2003). Vitek management je način vodenja organizacije, ki podpira koncept nenehnih izboljšav.

V samem razvoju proizvodov se je vitka miselnost začela uveljavljati kasneje. Tudi v tem smislu so Toyotin pristop k razvoju izdelkov poimenovali **vitek razvoj**. Vitek razvoj izdelkov (angl. *lean product development* ali *LPD*) je uporaba vitkih načel v razvoju izdelkov, s ciljem razviti nove ali izboljšane izdelke, ki bodo uspešni na trgu. Je navzkrižno delujoča dejavnost, ki skuša odkriti znanje o izdelkih, ki je skrito znotraj celega toka proizvodnje, tipično na točkah izročitve med posameznimi funkcionalnimi enotami. Vitek razvoj obravnava celoten proces od zbiranja in ustvarjanja idej, preko ocene potencialnega uspeha do konceptov razvoja. V ozadju vitkega razvoja izdelkov je neprekinjeno ocenjevanje in zmanjševanje tveganja za neuspeh na trgu (Mynott, 2000).

Na področju uvajanja vitkih pristopov v razvoj so prednjačili razvijalci programske opreme. Konec 90-ih let dvajsetega stoletja se je v razvoju programske opreme najprej uveljavil agilni proces, ki je bil skonstruiran na osnovi izkušenj, poskusov in napak, vedenja o tem, kaj ne deluje, ter dobrih praks. **Vitek razvoj programske opreme**, kot sta ga poimenovala Mary in Tom Poppendick v svoji knjigi Lean Software Development, pa je razširil teoretične temelje agilnega razvoja tako, da je pri razvoju programskih oprem uporabil dobro znana in sprejeta načela iz vitke proizvodnje (Poppendieck & Poppendieck, 2003).

Danes je obvladovanje inovacij izdelkov in poslovnih procesov ključnega pomena za preživetje podjetja, to pa zahteva uporabo naprednih metod in orodij. Mnoga podjetja skušajo uporabiti filozofijo odprave izgub, ki je prvo načelo vitke proizvodnje, v procesih inovacij in razvoja proizvodov. Uporaba vitkih proizvodnih konceptov v inovacijskih procesih (angl. *lean innovation*) ni tako neposredna in ima več problematičnih vidikov (Biazzo, Panizzolo & de Crescenzo, 2016).

Od leta 2008 se »vitko« uporablja v povezavi z zagonskimi podjetji (angl. *start-up* ali *startup*). **Vitko zagonsko podjetje** (angl. *lean start-up*) je metoda za razvoj dejavnosti in izdelkov, ki jo je na podlagi svojih izkušenj pri delu v nekaj zagonskih podjetij predlagal

Eric Ries (2008). Metoda vitkega zagonskega podjetja je nastala z namenom, da bi olajšala sam proces ustanovitve in začetnega razvoja podjetij (Blank, 2013). Riesova metoda vitkega zagonskega podjetja temelji na metodi razvoja strank, ki jo je njegov mentor Steve Blank opisal v knjigi The Four Steps to the Ephyphany (2005). Blank je ugotovil, da dobro definirani procesi in postopki, ki jih pri lansiranju novih izdelkov na trge uporabljajo velika uspešna podjetja, v primeru zagonskih podjetij ne delujejo. Največje tveganje in zato najpogostejši vzrok neuspeha pri zagonskih podjetjih ni razvoj novih izdelkov, ampak razvoj njihovih strank in trgov (Blank, 2005, str. 5).

Iz literature je razvidno, da se od 2009 leta vitka filozofija pojavlja tudi kot model v trajnostnem razvoju in ekologiji. Združujejo jo z zeleno proizvodnjo, metodo, ki minimizira vse vrste odpada in škodljive vplive proizvodnih in storitvenih aktivnosti na okolje. Takšen poslovni model se imenuje »**vitko in zeleno**« (angl. *lean and green*).

Kljub temu, da se termin vitka proizvodnja uporablja že več kot 25 let, standardna definicija za »vitko« ne obstaja. Nekateri raziskovalci ta termin uporabljajo z vidika prakse, kot nabor managerskih praks, orodij in tehnik, ki jih je moč neposredno opazovati (Shah & Ward, 2007). Poleg definicije vitkega, ki so osnovane na orodjih in sistemu, raziskovalci predstavljajo »vitko« kot filozofijo managementa in kot nabor smernic, ki vodijo organizacijo k dodani vrednosti (Womack & Jones, 2005), odpravljanju izgub (Holweg, 2007) in nenehnih izboljšav (Hines, Holweg & Rich, 2004; Shah & Ward, 2007).

Tudi Pettersen (2009) ugotavlja, da ni soglasja med avtorji glede definicije vitke proizvodnje. Avtorji imajo tudi različna mnenja glede tega, katere karakteristike naj bi bile povezane s konceptom vitke proizvodnje. Ta neskladnost lahko povzroči nekaj zmede na teoretičnem nivoju, je pa še bolj problematična na praktičnem nivoju, ko se organizacije odločijo uvesti koncept. Pomembno za organizacije je, da se zavedajo različnih variant in to spoznanje vključijo v implementacijski proces (Pettersen, 2009).

Namen magistrskega dela je na podlagi analize stanja v podjetju, ki se ukvarja z razvojem programske opreme, preveriti možnosti izboljšav v podjetju z uporabo pristopov vitkega razvoja in vitkega zagonskega podjetja.

Cilj naloge je sistematično prikazati razvoj »vitke miselnosti« od vitke proizvodnje, preko vitkega razvoja programske opreme do modela vitkega zagonskega podjetja ter poiskati stične točke v teh modelih. V praktičnem delu naloge bom preverila v kolikšni meri se vitka načela tako v razvoju izdelkov kot v razvoju podjetja, uporabljajo v konkretnem zagonskem podjetju. Na osnovi teoretičnih izhodišč in analize stanja v podjetju bom podala predlagale za izboljšave, ki bi lahko podjetju pomagali pri izgradnji poslovnega modela.

Magistrsko delo bo vsebovalo podroben pregled in analizo strokovne literature, znanstvenih razprav, raziskav in člankov tako domačih, kot tujih strokovnjakov s področja vitke proizvodnje, vitkega razvoja in drugih pristopov, ki se uporabljajo v zvezi z razvojem v zagonskih podjetjih. Pri tem bom uporabila deskriptivni pristop, ki daje prednost opisu strukture, delovanja ali razvoja določenega gospodarskega pojava. Za primerjanje enakih ali podobnih dejstev, procesov in pojavov pri spremljanju razvoja pojmovanja »vitkega« od proizvodnje do zagonskih podjetij in same definicije zagonskega podjetja bomo uporabili metodo komparacije.

V empiričnem delu bom v analizi stanja v podjetju uporabila sekundarne vire, kot so pretekle raziskave trga, prodajni rezultati, spletna stran podjetja in orodja za komuniciranje s kupci. Informacije o samem načinu dela in načinu komunikacije med zaposlenimi v podjetju pa nameravamo pridobiti z uporabo polstrukturiranega globinskega intervjuja, ki omogoča postavljanje dodatnih vprašanj med samo izvedbo intervjuja. Pri nestrukturiranem ali delno strukturiranem intervjuju ne uporabljamo do potankosti vnaprej pripravljenega vprašalnika, ampak zgolj vodilo ali predlogo za intervju (Mesec, 1998).

1 VITKA PROIZVODNJA IN VITEK RAZVOJ

1.1 Razvoj vitke miselnosti

1.1.1 Toyotin proizvodni sistem in njegova zgodovina

Najbolj poznan delujoč primer vitke proizvodnje je Toyotin proizvodni sistem, ki je nastal v Toyota Motor Corporation v proizvodnji motornih vozil kot radikalna alternativa tradicionalni masovni serijski proizvodnji avtomobilov. Toyotin proizvodni sistem stremi k optimalni učinkovitosti, kakovosti, hitrosti in zniževanju stroškov (Holweg, 2007). V drugi polovici dvajsetega stoletja je Toyotin način proizvodnje imel že globalni vpliv. Podjetja širom po svetu so sprejela Toyotin proizvodni sistem pod imenom vitka proizvodnja.

Ključne temelje vitke proizvodnje je že leta 1913 postavil Henry Ford. Proizvodni proces je razdelil na posamezne korake in jih vpeljal v proizvodno linijo. Proizvodnja za tekočim trakom je takrat pomenila prelomnico v načinu proizvodnje. Glavni problem serijske proizvodnje pa je bil, da z njo niso mogli zagotoviti raznolikosti in variantnih modelov avtomobila, ki jih je zahteval trg. Problem Fordovega sistema je bila tudi prekomerna proizvodnja.

Toyota Motor Company (v nadajevanju Toyota) je bila ustanovljena 1930. Toyotin proizvodni sistem pa se je začel razvijati po drugi svetovni vojni, ko je imela Japonska uničeno infrastrukturo, tovarna Toyota pa je bila močno zadolžena. Da bi zmanjšala dolg

in povečala obratni kapital, je morala Toyota v celoti spremeniti sestavo proizvodnje. Kmalu so postavili teze, ki predstavljajo začetek Toyotinega proizvodnega sistema (Piškor & Kondić, 2010):

1. Vse, kar v procesu proizvodnje ne dodaja vrednosti končnemu proizvodu, je potrebno odstraniti iz procesa.
2. Čim bolj skrajšati čas proizvodnje in znižati stroške zaradi nedokončane proizvodnje, vendar pri tem povečati fleksibilnost sistema.
3. Ne proizvajati izdelkov, ki nimajo kupca. Kupci morajo dobiti izdelek, kot ga želijo v čim krajšem času.

Inovacije v Toyotini proizvodnji so bile tako posledica pomanjkanja virov kot tudi intenzivne domače konkurence. Glavni ustanovitelji Toyotinega proizvodnega sistema so bili Taiichi Ohno, Shigeo Shingo in Eiji Toyota. Precej zgodnjega dela v Toyoti so pod vodstvom Taiichija Ohna uporabili v 50-ih letih v proizvodnji motorjev, v 60-ih pri sestavljanju vozil in šele kasneje v 70-ih letih dvajsetega stoletja v celotni oskrbovalni verigi, od koder se je ideja vitke proizvodnje kasneje tudi razširila (Holweg, 2006).

V Toyoti so proizvodni proces razvijali desetletja, vendar je njihovo načelo preprečevalo, da bi spremembe zapisovali. Eno od osrednjih načel Toyotinega proizvodnega sistema je namreč, da pravo učenje izhaja samo neposredno iz dela. Japonci to imenujejo *gemba,* kar pomeni mesto, kjer se delo opravlja. Zato skiciranje in modeliranje samega procesa nima dodane vrednosti (Liker & Morgan, 2006). Pristop k vitki proizvodnji so zaradi tega šele v 70-ih letih prejšnjega stoletja začeli deliti z ostalimi podjetji. Vsi priročniki za dobavitelje so bili napisani v japonščini in šele v naslednjem desetletju se je pojavila prva literatura v angleškem jeziku (Hines et al., 2004).

Toyota je identificirala sedem glavnih virov izgub v poslovanju ali v proizvodnem procesu (Molnar, 2009):

- **Transport:** Nepotrebno premikanje delov ali materiala v procesu ne dodaja vrednosti. Nanaša se na neučinkovit transport materiala, delov, končnih proizvodov v skladiščih, neučinkovit pretok informacij, izgubo podatkov ali nezadostnost informacij.
- **Prekomerne zaloge:** Višek surovin, polizdelkov in izdelkov povzročajo daljše pretočne čase, zastajanje blaga, poškodbe blaga, dodatne stroške v transportu. Prekomerne zaloge prikrivajo druge probleme kot so neuravnoteženost procesa, zamujanje z dobavami, napake, zastoji na strojih.
- **Nepotrebni premiki** se nanašajo na vse nepotrebne premike, ki jih pri delu naredijo zaposleni: iskanje, zlaganje proizvodov in orodja, hoja med delom.

- **Čakanje** se na naša na čas, ko delavci čakajo na delovnem mestu ampak ne delajo. To čakanje se nanaša na čas menjave orodij, izčrpane zaloge, nepripravljen polproizvod oziroma na stanje, ki ne prispeva k dodani vrednosti.
- **Prekomerna in nepotrebna obdelava** se nanaša na odvečne korake v obdelovanju proizvoda. Zaradi napak v obdelavi, ki so lahko posledica slabih orodij ali neprimerne konstrukcije, pride do popravljanj in s tem do prekomernega dela. Izgube nastajajo tudi, če je kakovost izdelka višja, kot pričakuje kupec, ki zanjo ni pripravljen plačati.
- **Prekomerna proizvodnja:** Proizvodnja izdelkov, za katere ni naročil, povzroča izgube zaradi preveč zaposlenih in dodatne stroške skladiščenja in transporta. Nanaša se tudi na prekomerno proizvodnjo polizdelkov znotraj procesa proizvodnje.
- **Napake:** predstavljajo proizvodnjo izdelkov z napakami in popravljanje napak na izdelkih in polizdelkih, škart, ponovno proizvodnjo in naknadno kontrolo, ki pomeni izgubo časa.

Liker (2004) je dodal še osmo vrsto izgub: **neizkoriščen potencial zaposlenih.** Ta predstavlja zgubljen čas, veščine, izboljšave in možnosti učenja zaradi nevključevanja ali neposlušanja zaposlenih. Avtomatizacija je dobrodošla izboljšava samo na področjih, kjer se ponavljajoče se delo na tak način izvaja boljše in varneje. Ustvarjalno razmišljanje delavcev je sicer nenadomestljiv potencial.

Eden od stebrov Toyotinega proizvodnega sistema je proizvodnja ob pravem času (angl. *just-in-time*), ki se nanaša na hiter tok materiala skozi proizvodnjo z namenom, da se pravi del znajde na pravem mestu ob pravem času (Liker & Morgan, 2006). Proizvaja se samo to, kar je potrebno, kadar je potrebno in v potrebnih količinah.

Drugi steber Toyotinega proizvodnega sistema je manj znan koncept imenovan *jidoka.* Gre za avtomatsko ustavitev proizvodnje in signaliziranje, kadar je potrebna pomoč. Proces lahko ustavi stroj, ki zazna deviacijo od standarda ali pa gre za manualni proces, ko stroj zaustavi človek. Gre za odpravo problemov na samem izvoru, preden se problem razširi ali celo vodi do defektnega izdelka, ki pride v roke kupcu. Problemi so tako takoj zaznani in odpravljeni, kar vodi h kontinuiranim izboljšavam (Liker & Morgan, 2006).

1.1.2 Vitka proizvodnja

Vitka proizvodnja (angl. *lean manufacturing*) je filozofija obvladovanja proizvodnih procesov. Osnovana je na Toyotinem proizvodnem sistemu, ki je usmerjen v odpravljanje sedmih vrst izgub z namenom, da se poveča vrednost proizvoda za končnega uporabnika.
Po Womacku in Jones-u (1996, str. 16) nam vitka proizvodnja govori o tem, kako narediti čim več s čim manj virov – z manj človeških naporov, manj opreme, manj časa in prostora – pri čemer mora biti proizvod napravljen tako, da popolnoma zadovolji kupca.

Čas od naročila do trenutka, ko je proizvod prodan, se skrajšuje z ukinitvijo del, ki ne dodajajo vrednosti proizvodu. Vitka proizvodnja temelji na petih osnovnih načelih (Womack & Jones, 1996, str. 15-26):

- natančno definiranje vrednosti proizvoda s stališča kupca;
- prepoznavanje toka vrednosti (angl. *value stream*) za določeno vrsto proizvoda;
- izenačen in kontinuiran tok materiala;
- načelo vlečenja (angl. *pull*) proizvoda skozi celoten proces proizvodnje;
- teženje k popolnosti

Definiranje vrednosti: vrednost za določen proizvod ali storitev definira kupec. Vrednost je mišljena kot funkcija, ki je vezana na proizvod ali storitev in izpolnjuje svojo osnovno nalogo – zadovoljuje želje in potrebe kupca ali uporabnika. Tako definirana vrednost je izhodiščna točka uspešne proizvodnje in poslovanja. Potrebno je analizirati potrebe kupca in značilnosti proizvoda ter ugotoviti, katere funkcije oziroma procesi dodajajo vrednost proizvodu. Ostale procese je potrebno eliminirati.

Tok vrednosti: za določene skupine proizvodov se mapirajo tokovi vrednosti s čim več natančnimi kvantitativnimi informacijami o procesu. Te informacije vključujejo čas trajanja operacij, čas potreben za tehnološki ciklus, kapaciteto strojev, čas trajanja dela, čakanje. Iz informacij je potrebno sestaviti zemljevid toka vrednosti. Zemljevid toka vrednosti je tehnika z uporabo svinčnika in papirja. Ljudem z risanjem pomaga videti in razumeti tok materiala in informacij pri procesu, ko izdelek naredi svojo pot skozi proces (Chen & Meng, 2010). S pomočjo zemljevida toka vrednosti je mogoče ugotoviti, katere so tiste aktivnosti, ki dodajo vrednost, katere aktivnosti so neobhodne, vendar ne dodajajo vrednosti in katere so čista izguba glede na vrednost izdelka, ki jo definira kupec. Slednje je potrebno izločiti.

Tok materiala: ko se na osnovi mapiranja toka vrednosti analizira proces, se določi takt proizvodnje in na osnovi tega kontinuirani tok materiala. Pri tem je potrebno pri prehodu obdelovanca iz procesa na proces eliminirati čas, ki ne doda vrednosti izdelku. Izogibati se je potrebno proizvodnje v velikih serijah z velikimi zalogami. Koncept **ravno ob pravem času** (angl. *just-in-time*, v nadaljevanju JIT) je bil prvič uporabljen v Ford Motor Company, kasneje pa so ga sprejeli in izboljšali v Toyoti. Del JIT-a je uporaba mehanizma, imenovanega **kanban**. Kanban je sistem za oskrbovanja delovnih mest z materialom ali obdelovanci. Temelji na ideji, da se tok materiala odvija po samopostrežnem načelu. V japonščini kanban pomeni znak ali tablico, v praksi je to pogosto dobesedno listek, dokument ali kartica v primernem ovitku, s katerim je opremljen material oziroma obdelovani del. Gre za preprosto kontrolo proizvodnje na načelu vlečenja materialov od enega do drugega delovnega mesta: vsak material je opremljen z identifikacijsko kanban kartico, ki je odstranjena in odložena na polico, ko se material porabi. Te kartice odnesejo

k dobavitelju materiala, ki dobavi enake dele in jih ponovno opremi s karticami (Ferjančič, 2011).

Načelo vlečenja materiala: je eden od temeljnih načel vitke proizvodnje. Za razliko od načela potiskanja, se to načelo začne s kupcem – z nakupom ali naročilom. Po tem, ko kupec izrazi potrebo po proizvodu, vsak korak v verigi vrednosti prenaša informacijo na predhodni korak v procesu in začne se proces, ki je potreben, da se iz začetnih materialov naredi nov proizvod. Pri upoštevanju tega načela ne pride do prekomerne proizvodnje, ki predstavlja izgubo.

Teženje k popolnosti: to načelo je v bistvu proces nenehnih izboljšav ali **kaizen**. Ta proces se ne sme zaustaviti, saj zagotavlja prednost pred konkurenco. Vitka proizvodnja nalaga redno organizacijo kaizen delavnic, z namenom izpopolnjevanja procesov. Odgovornost za izboljšave je razširjena na vse zaposlene.

Raziskovanje poslovnega modela, infrastrukture in praks, ki podpirajo vitko proizvodnjo, so promovirale transfer in možnost prenašanja teh načel v ostalo industrijo izven Japonske (Womack et al., 1990). Navdušeni nad visoko zmogljivostjo vitke proizvodnje v primerjavi s tradicionalno masovno proizvodnjo, so zahodni proizvajalci oponašali tehnike v delavnicah, ki so sestavni del vitkega, vendar so velikokrat ugotovili, da je težko vpeljati vitko organizacijsko strukturo in miselno naravnanost. Tako je dosti zgodnjih prenosov vitke proizvodnje pokazalo samo lokalni vpliv in na žalost je izostal pričakovan vpliv na celoten uspeh organizacije (Holweg & Pil, 2001).

1.1.3 Vitek razvoj

1.1.3.1 Usmeritve, na katerih temelji Toyotin vitek razvoj

Proizvodnja predstavlja samo polovico problema celotne izdelave avtomobila. Toyotin razvoj izdelkov je prav tako inovativen kot vitka proizvodnja. V primerjavi z ekvivalentnim razvojem v Ameriki, je bil v Toyoti projekt razvoja izdelka lahko opravljen v polovičnem času, pri čemer je bilo v razvoju udeleženih štirikrat manj inženirjev (150 produktnih inženirjev na avtomobil v Toyoti, v primerjavi 600 inženirji v Chrislerju). Zato je bil Toyotin pristop k razvoju izdelkov imenovan **vitek razvoj.** Njegova izhodiščna točka pa je, da razvojni inženirji dejansko skrbijo o tem, kaj o njihovem izdelku mislijo kupci. To jim omogoča oblikovanje močne vizije o bodočem izdelku in komuniciranje te vizije z vsemi udeleženci v razvojnem procesu (Ballé & Ballé, 2005).

V poglobljeni študiji, ki jo bila narejena na podlagi njegove direktne izkušnje z razvojem izdelkov v Toyoti, je Morgan (2005) identificiral trinajst načel, ki jih navaja kot temelj Toyotinega vitkega razvoja. Ta načela so vezana na procese, ljudi ter orodja in tehnologijo. V Toyoti dober proces ni definiran z uporabo določene tehnologije, ampak z dobrimi

načeli, na osnovi katerih ljudje kreirajo in izboljšujejo proces. Načela vitkega razvoja, ki so vezana na sam proces razvoja so (Liker & Morgan, 2006):

- **Ugotoviti vrednost, ki jo definira kupec**: je del kulture v podjetju. Dodana vrednost je vrednost, ki jo opredeli kupec. Izguba je vse, kar porabi čas, denar in človeške vire in z vidika kupca ne doda vrednosti izdelku.
- **Na začetku skoncentriran proces razvoja izdelka**: z namenom, da se resnično proučijo alternativne rešitve v času, ko obstaja še maksimalen prostor za drugačno zasnovo izdelka. Daljši čas za proučitev alternativ in reševanje predvidenih problemov že pri vzrokih, prinaša finančne prednosti. Opredelitev napačnega problema ali prezgodnje zbliževanje stališč glede napačnih rešitev povzroča stroške v vsem življenjskem ciklu izdelka.
- **Kreiranje izenačenega toka procesa razvoja**: na podlagi izkušenj iz razvoja lahko s precejšnjo natančnostjo predvidimo porabo virov. Na ta način se lahko proces stabilizira in pravilno načrtuje.
- **Uporaba standardiziranih procesov:** izziv v razvoju je, da se kljub standardizaciji ohrani kreativnost. S standardizacijo procesov se zmanjšujejo spremembe, kreirajo pa se fleksibilni in predvidljivi rezultati. To je rešitev za ciklične potrebe po virih, ki so običajno povezane z razvojem izdelkov.

V vsakem vitkem sistemu so ljudje tisti, ki zagotavljajo inteligenco in energijo. Sistem ljudi zajema rekrutiranje in izbiro inženirjev, trening in poklicni razvoj, stil vodenja, organizacijsko strukturo, institucionalno učenje in spomin ter organizacijsko kulturo. Kultura se nanaša na skupni jezik, simbole, prepričanja in vrednote. Načela vitkega razvoja, ki se nanašajo na ljudi, govorijo o razvoju ljudi, ki nenehno izboljšujejo izdelek in proces (Liker & Morgan, 2006):

- **Razvoj sistema »glavnega inženirja«:** glavni inženir je glavni načrtovalec z najvišjimi pooblastili in odgovornostjo za celoten razvojni proces. Glavni inženir je premostitveni člen pri integraciji izdelka in procesov.
- **Uravnoteženje funkcionalnih znanj in navzkrižnih funkcionalnih integracij**: ekspertno znanje, skupni višji cilji in glavni inženir vzdržujejo ravnotežje v matrični organizaciji.
- **Razviti visoko tehnično usposobljenost inženirjev**: inženirji morajo imeti dobra specialna znanja o izdelku, ki jih pridobivajo tudi z direktnim obvladovanjem veščin pri delu.
- **Integriranje dobaviteljev v razvoj izdelkov**: dobavitelji komponent, katerih zmogljivost in kultura sta združljivi s Toyotino, morajo biti smiselno vključeni v razvojni proces.

- **Učenje in stalne izboljšave**: organizacijsko učenje in stalne izboljšave so najbolj trajnostna konkurenčna prednost. Učenje in stalne izboljšave sta osnova pri vsakodnevnih operacijah in krajšanju časa potrebnega za učne cikle.
- **Izgradnja kulture, ki podpira odličnost in nenehne izboljšave:** izgradnja kulture, ki podpira odličnost, je temeljni del vodenja. Vodilni se obnašajo na način, ki je skladen s temeljnimi prepričanji, za katera se zavzemajo.

.

Tretji okvir načel zajema orodja in tehnike, ki se uporabljajo za razvoj izdelka. Tu niso zajeta samo proizvodna tehnologija, ampak tudi »mehka« orodja, ki nudijo podporo ljudem, ki so vpeti v razvojni projekt. Gre za način reševanja problemov, učenje ali standardizacijo najboljših praks (Liker & Morgan, 2006):

- **Prilagoditev tehnologije, da ustreza ljudem in procesom:** tehnologija mora biti prilagojena in vedno podrejena ljudem in procesom.
- **Komunikacija mora zajemati vse ravni organizacije vključno z vizualnim pristopom.**
- **Uporaba orodij za standardizacijo in organizacijsko učenje:** orodja so lahko preprosta (mapiranje razvojnega procesa). Konstantne izboljšave brez standardizacije niso mogoče.

Integriranje ljudi, procesov ter orodij in tehnologije v skladen sistem zahteva, da so podsistemi namensko oblikovani, povezani in da se medsebojno podpirajo. Za učinkovit proces pa so potrebni znanje ljudi in njihova organiziranost na način, da so pravi ljudje za določene naloge na razpolago ob pravem času. Na razpolago pa morajo biti tudi orodja in tehnologija, ki ustreza procesu in podpira aktivnost ljudi, ki tako lahko izkoristijo svoj potencial (Liker & Morgan, 2006).

Vitek razvoj obsega številne medsebojno povezane tehnike kot so sodelovanje dobaviteljev, oblikovanje navzkrižno delujočih timov, sočasen razvoj, integracija (kot nasprotje koordinaciji) različnih funkcionalnih vidikov vsakega projekta in strateški management vsakega razvojnega projekta. Vendar podjetje ne doseže vitkega razvoja izdelkov preprosto z uvedbo nekaterih od naštetih tehnik. Uspešen premik proti vitkemu razvoju zahteva, da se k tem tehnikam pristopi kot k elementom skladne celote (Karlsson & Ahlström, 1996). Ne gre za zbirko dobrih praks, ki jih je mogoče vpeljati po delih, ampak za sistem, v katerem je potrebno razumeti tako temeljne prakse kot odnose. Vpliv Toyotinih vitkih pristopov v proizvodnji je čutiti v vsakem vidiku njenega procesa razvoja izdelkov (Ballard, 2005). O izboljšavi procesa pogosto razmišljamo kot o tehničnem vprašanju: uporabiti pravo metodologijo, upravičiti njene stroške in omogočiti da deluje. Ker je v TPS primarni poudarek na odpravi stroškov zaradi izgub, je precej podjetij napačno razumelo, da je to bistven poudarek tudi pri razvoju izdelkov. Dejanske koristi vitkega razvoja izhajajo iz kreiranja toka, ki sočasno izboljša čas povratnih informacij,

kvaliteto izdelka in učinkovitost procesa (Reinersten, 2009). Liker in Morgan (2006) sta Toyotin razvoj izdelkov razdelila v štiri faze:

- Faza koncepta, v kateri glavni inženir povzame svojo vizijo v osnutku izdelka.
- Zasnova izdelka s sočasnim razvojem.
- Faza podrobnega načrtovanja z upoštevanjem standardov.
- Faza prototipov in orodij z vitko proizvodnjo.

1.1.3.2 Faza koncepta

V mnogih podjetjih so za različne dele razvoja izdelkov odgovorni različni funkcionalni oddelki, zato je težko prepoznati, kakšen je status projekta in kje se sprejemajo odločitve. V Toyoti je odgovor jasen – za projekt je odgovoren glavni inženir. Glavni inženir ni samo projektni manager, je tudi vodja in povezovalec sistema. Kandidate za to funkcijo izberejo in razvijajo desetletja z namenom, da dobijo najboljše in najpametnejše inženirje in povezovalce sistema (Liker & Morgan, 2006). Glavni inženir ne koordinira samo tehnološkega razvoja, ampak tudi proizvodnjo in prodajo, torej celoten projekt od koncepta do trga. Neposredno in pogosto komunicira z oblikovalci in inženirji, ima tudi neposredne stike s kupci (Fujimoto v Ballé & Ballé, 2005). V prvi vrsti je tehnični ekspert, ki ima velik vpliv na arhitekturo avtomobila. Kljub njegovi odgovornosti za razvoj izdelka od koncepta do trga, pa ima majhno formalno avtoriteto. Prepoznan in cenjen je po izkušnjah in tehničnih in komunikacijskih sposobnostih. Vodi majhno skupino produktnih in proizvodnih inženirjev. Njegova vizija avtomobila je zajeta v osnutku, ki je osnova za fazo oblikovanja izdelka (Ballé & Ballé, 2005).

1.1.3.3 Zasnova izdelka, ki temelji na sočasnem razvoju

V avtomobilski industriji se kos pločevine oblikuje v karoserijo avtomobila s pomočjo velike stiskalnice, ki vsebuje kalup, po katerem se med stiskanjem pločevina preoblikuje v ogrodje avtomobila. Oblikovanje in izrezovanje tega kalupa za izdelavo avtomobilske karoserije predstavlja polovico kapitalske investicije celotnega razvoja novega avtomobila. Inženirji med razvojnim procesom spreminjajo obliko novega avtomobila in vsaka taka sprememba se odraža tudi na oblikovanju kalupa. Ne glede na to, kako se trudijo zamrzniti končno obliko avtomobila, to med procesom dizajniranja ni mogoče. V 80-ih letih dvajsetega stoletja so v ameriški avtomobilski industriji spremembe pri oblikovanju modela stale od 30 do 50 % izdelave kalupa, medtem ko je na Japonskem ta strošek znašal samo od 10 do 20 %, kar je kazalo na to, da japonska podjetja ne dovolijo toliko sprememb, ko so specifikacije modela že sprejete. Vendar je bila razlika med ameriškim in japonskim pristopom do izdelave modela v tem, da so japonski izdelovalci kalupov vedeli, kaj vsebuje osnova kalupa za izdelavo posameznega dela šasije in samo predvidevali končno rešitev. Njihova stalna komunikacija z oblikovalci karoserije jim je omogočala, da so se inženirji, ki so izdelovali kalupe, večinoma lahko prilagodili oblikovalcem. Obvladali

so tehnike, kako v proces izdelave kalupa vpeljati pozne manjše spremembe čim kasneje, kot na primer puščanje dovolj rezervnega materiala na mestih, kjer so spremembe oblike verjetnejše. Tak znatno izboljšan razvoj izdelka se imenuje **sočasni razvoj** (Poppendick & Poppendick, 2003, str. 48)**.**

Sočasni razvoj je najbolj poznana praksa v razvoju izdelkov, ki jo je Toyota začela uporabljati v poznih 60-ih letih dvajsetega stoletja. Z gledišča konvencionalnega modela razvoja je videti Toyotin proces dizajniranja neučinkovit, drag in neroden (Ward, Liker, Christiano & Sobek, 1995). Pri sočasnem razvoju razvojni tim definira namesto ene rešitve cel set rešitev in set možnih rešitev na nivoju podsistemov. Te rešitve se raziskujejo vzporedno in na podlagi eksperimentov in analiz se niz rešitev počasi zožuje v eno samo rešitev. Ko tim oblikuje eno samo rešitev za vsak del, ki ga dizajnirajo, je več ne spreminja, razen če je to zares potrebno (Ward et al., 1995).

Toyota skuša identificirati in rešiti vse možne probleme že zgodaj v razvojne procesu. Na koncu konflikte, do katerih pri tem prihaja, rešuje z vračanjem h kriterijem, ki zadovoljujejo kupce. Kljub temu, da Toyota upošteva širši nabor možnih dizajnov kot večina ostalih proizvajalcev avtomobilov, ter odlaša s končnimi odločitvami tako, da postopoma oža nabor specifikacij in rešuje nejasnosti, je občutno skrajšala razvojni cikel. V kritičnem obdobju faze dizajniranja se pod vodstvom glavnega inženirja redno srečujejo izkušeni predstavniki vseh oddelkov od proizvodnje do nabave, da razvijejo timsko vzdušje in sprejemajo odločitve (Ballé & Ballé, 2005).

1.1.3.4 Podrobno načrtovanje in standardizacija

Zmanjšanje sprememb ob ohranjanju kreativnosti je izziv v razvoju izdelkov. Toyota je zmanjšala variabilnost s pomočjo standardizacije veščin, procesov in samega dizajna. Standardna znanja in spretnosti Toyotinih inženirjev omogoča fleksibilnost pri rekrutiranju zaposlenih in načrtovanju dela ter minimizira razlike pri opravilih. Standardizacijo procesov dosežejo z dizajniranjem izdelkov in izgradnjo proizvodih obratov osnovanih na standardih TPS. Standardizacijo dizajna pa omogoča enotna arhitektura, modularnost, možnost ponovne uporabe in skupne komponente (Liker & Morgan, 2005; Ballé & Ballé, 2005). Ta visok nivo standardizacije je ključen za odpravo predelav in izgub in paradoksalno – omogoča prilagodljivost zmogljivosti. Čeprav je za ta proces potrebno veliko kontrolnih seznamov, je natančen proces standardizacije pomemben za maksimiziranje učenja, nenehno izboljšavo procesa dizajniranja izdelka in zato, da je postane ta proces vedno hitrejši (Ballé & Ballé, 2005).

1.1.3.5 Prototipi in orodja za vitko proizvodnjo

Na splošno Toyota razvija dve vrsti prototipov, ki pa niso namenjeni testiranju rešitev, ampak preverjanju podsistemov in njihove integracije v končni izdelek. Prvi izdelki

prvega vala (1S) so zelo previdno in počasi sestavljeni, da se preverijo vse interference. Ostali izdelki iz tega vala so sestavljeni z uporabo vitkih proizvodnih tehnik. Hkrati proizvodni inženiring koordinira svojo počasno izgradnjo, da bi identificiral težave, ki lahko nastanejo pri proizvodnji in montaži. Izdelava prototipov je čas, ko najintenzivneje delajo inženirji. Je tudi čas, ko se lahko mlajši inženirji največ naučijo. To je zadnja stopnja razvoja, v kateri so inženirske spremembe še dovoljene in sprejemljive. Toyoto večkrat predstavljajo kot proizvajalca, ki izdela več modelov, kot to počnejo drugi proizvajalci avtomobilov. V praksi vsi proizvajalci avtomobilov naredijo približno enako število modelov. Toyotina edinstvenost je v tem, da poteka natančna razprava o proizvodnih težavah že v fazi modela, medtem ko konkurente v tej fazi bolj skrbi stil in inženiring (Ballé & Ballé, 2005).

1.2 Metode razvoja programske opreme

V zadnjem desetletju so se raziskave o razvojnih procesih na področju programske opreme (v nadaljevanju PO) zelo razvile. Dramatičen in neverjetno hiter razvoj številnih tehnologij in naprav, kot so mobilne internetne naprave, je še okrepil prepričanje, da raziskave procesov razvoja programske opreme potrebujejo znatne spremembe tako v obsegu kot v pristopu. Še posebej je bilo prepoznano, da zahtevajo študije teh procesov multidisciplinaren pristop – upoštevane in vključene morajo biti tudi druge discipline.

1.2.1 Proces razvoja programske opreme kot sociotehniški sistem

Raziskovalci so sedaj manj osredotočeni na modeliranje in izvajanje procesov. Strinjajo se, da na izvedbo procesa v razvoju programske opreme močno vpliva obnašanje posameznikov v organizaciji (Yilmaz, 2007). Z namenom, da bi identificirali strukture, ki jih je dobro uporabiti na področju inženiringa programske opreme, so Tamburri, Lago in Vliet (2007) analizirali literaturo s področja organizacijskih socialnih struktur. Proces razvoja programske opreme se obravnava kot sociotehniški sistem, kjer imajo organizacijski in človeški aspekti ključno vlogo in morajo biti podprti s tehnologijo na način, da jih poganja tako organizacija kot ljudje. Termin sociotehniški so prvi uporabili Trist, Bamforth in Emery (1951) v študiji o delavcih v premogovnikih. Posebej so opozorili na to, da predstavitev nove tehnologije ne vodi nujno v izboljšave. Izboljšave so rezultat pravilnega medsebojnega delovanja in dopolnjevanja med vidiki, ki se osredotočajo na ljudi, še posebej na socialni aspekt, ter tehnologijo (Fugetta 2014). Pojem sociotehniški zajema vzajemno povezanost socialnih in tehničnih vidikov neke organizacije in temelji na tezi, da se vezi med ljudmi in tehnologijo ne morejo poenostaviti na enostavno uvajanje nove tehnologije, ki mora rešiti očitne probleme pri izvajanju dela. Človeški in tehnični element procesa delujeta vedno v medsebojni odvisnosti, kar še posebej upoštevamo v raziskovalnem procesu (Fugetta, 2014). Pomembnost socialnega vidika v razvojnem procesu se kaže tudi v agilnem manifestu, ki je nastal leta 2001. V njem so določena osnovna načela agilnih metod. Kar dve od štirih osnovnih načel agilnega

manifesta poudarjata pomembnost človeškega faktorja v razvoju: človeško vedenje in kakovost interakcij med ljudmi so nujni faktorji, ki omogočajo uspeh ter dosledno in pogosto sodelovanje s kupci.

1.2.2 Tradicionalne metode razvoja programske opreme

Izboljšave v procesu razvoja programske opreme so evolucija, v kateri so se novi procesi gradili na napakah in uspehih predhodnih procesov. Za razumevanje agilnega gibanja je potrebno poznati tudi metode, ki so bile aktualne pred njim (Cohen, Birkin, Garfield & Webb, 2004).

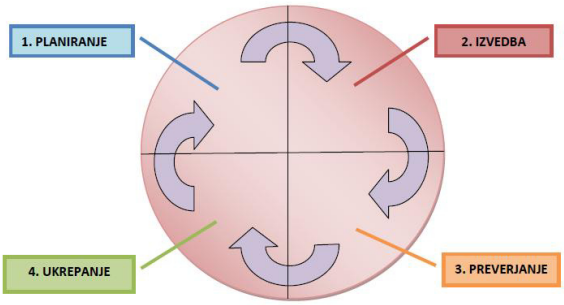
1.2.2.1 Iterativni in inkrementalni razvoj

Čeprav se **postopen in ponavljajoč se razvoj** (angl. *iterative and incrementive development* ali *IID)* večinoma obravnava kot moderna praksa, začetek uporabe datira v sredino 50-ih let dvajsetega stoletja. Kljub temu, da so se iterativne metode prakticirale in objavljale že desetletja pred tem, so nekateri videli iterativni, evolucijski in primarni razvoj, na katerih temeljijo agilne metode, kot moderne nadomestke kaskadnega modela (Larman & Basili, 2003).

Postopen in ponavljajoči se razvoj je delo Walterja Shewharta, strokovnjaka za kvaliteto v Bell Labs, ki je predlagal serijo kratih »plan-do-study-act« (v nadaljevanju PDSA) ciklov z namenom izboljšanja kvalitete proizvodnje. PDSA je v 40-ih letih dvajsetega stoletja promoviral William Edwards Deming, kar je kasneje opisal v svoji knjigi »Out of Crisis« (MIT Press, 2003).

Cikel PDSA teče v smeri urinega kazalca in se prične z načrtovanjem (angl. *plan*), ki zajema identifikacijo cilja ali namena, formuliranje teorije, definiranje meril uspeha in prenos načrta v izvedbo. Sledi korak izvajanje (angl. *do*), v katerem se načrtovane komponente, kot na primer izdelek, naredijo oziroma izvedejo. V fazi preverjanja (angl. *study*) se rezultati opazujejo v smislu opazovanja napredka in uspeha ali problemov in področij izboljšav v primerjavi z načrtom. Krog zaključuje korak ukrepanje (angl. *act*), kjer se integrira v procesu generirano znanje, ki se lahko uporabi za prilagoditev cilja, spremembo metod ali celo preoblikovanje teorije v celoti. Ti štirje koraki se ponavljajo vedno znova kot del nikoli zaključenega kroga kontinuiranih izboljšav (The PDSA Cycle, 2011).

Slika 1: Demingov krog



Vir: Povzeto po The W. Edwards Deming Institute – The PDSA Cycle, 2016

Basil in Turner (1975) sta jasno opisala klasični iterativni in inkrementalni razvoj: osnovna ideja izboljšav na osnovi ponavljanja je, da se programska oprema razvija postopoma tako, da razvijalcu omogoča, da izkoristi, kar se je naučil med razvojem predhodnih, postopnih verzij sistema. Kjer je le mogoče, poteka tok učenja tako s strani razvoja, kot s strani uporabnikov. Pri vsaki ponovitvi, se modificira oblika in dodajo nove funkcionalne zmožnosti. Vsak korak v procesu ponavljanja sestoji ali iz preproste, dobro razumljene razširitve, ali manjše spremembe oblike ali funkcionalnosti, ki jo je motiviralo boljše razumevanje problema, pridobljeno v procesu razvoja.

Postopen in ponavljajoč se razvoj programske opreme je bil uspešno uporabljen v nekaterih Nasinih projektih: projekt Mercury (od 1958 do 1963) – prvi polet človeka v vesolje in X-15 - projekt nadzvočnega letala na raketni pogon. Čeprav projekta v osnovi nista bila samo razvoj programske opreme, sta bila oba povezana z ustanovitvijo IBM-ove Federal System Division (Larman & Basili, 2003).

1.2.2.2 Kaskadni model

Kaskadni model je bil prvi predlagan način, kako razviti PO za potrebe končnega uporabnika.

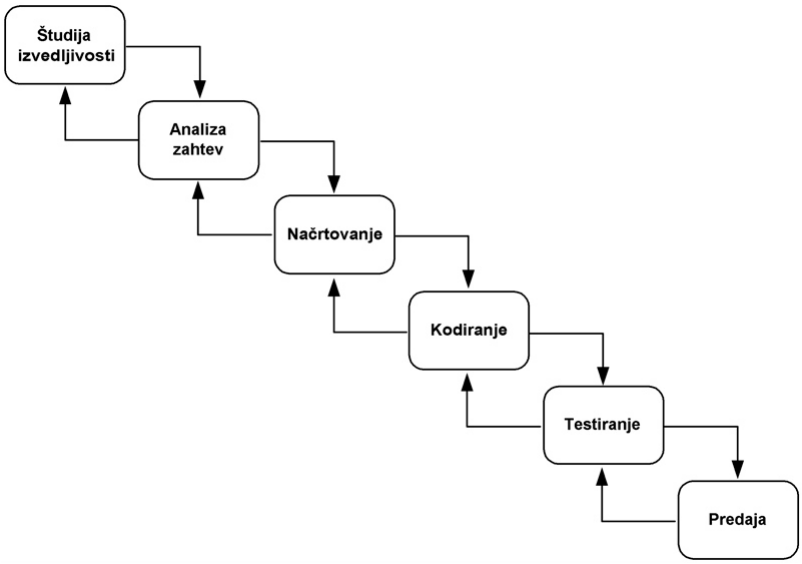
Avtor kaskadnega (angl. *waterfall*) modela je Royce (1970). V svojem članku z naslovom »Managing the Development of Large Software System« je predlagal koncept, ki je kasneje postal znan pod imenom **kaskadni model**. Kaskadni model je odigral nekakšno pionirsko vlogo med modeli razvoja programskih rešitev. Sestavljen je iz šestih faz razvojnega procesa, ki se med seboj ne prekrivajo. Verifikacija in validacija predstavljata zaključek posamezne faze in hkrati vhodno točko naslednje faze. Kaskadni model poteka v naslednjih fazah (Kampuš, 2002):

- Začne se s kompletno analizo zahtev.

- Po mesecih intenzivnih interakcij z uporabniki in strankami, inženirji predlagajo končni in obsežen set funkcij, zahtev po funkcionalnosti in ostalih zahtev, ki niso povezane s funkcijami.
- Ta informacija je dobro dokumentirana za naslednjo stopnjo - oblikovanje, pri katerem inženirji sodelujejo z drugimi strokovnjaki – eksperti za podatkovne baze in podatkovne strukture ter ustvarijo optimalno arhitekturo sistema.
- Programerji izvedejo dobro dokumentacijo sistema.
- Izdelan in oblikovan sistem je testiran.
- Sistem je dan v uporabo.

Za ta model so značilni sprotna izdelava dokumentacije in povratne zanke, s katerimi se po potrebi vrnemo v predhodno fazo, da bi odpravili napake oziroma vnesli dopolnitve.
Model se dobro obnese pri projektih, kjer so programske zahteve zelo stabilne in dobro znane že na začetku projekta. Koncept zmanjšuje količino odvečnega planiranja, saj je kompleten projektni plan narejen že ob začetku življenjskega cikla. Ker so faze med seboj ostro ločene, lahko v različnih fazah projekta sodelujejo različni posamezniki. Pogoj za prehod ene faze v drugo, je dobro dokumentirana in zaključena prejšnja faza. Kaskadni model je prikazan na Sliki 2.

Slika 2: Kaskadni model



Vir: Povzeto po eSistemi – Kaskadni model, 2016

Kaskadni proces je teoretično dovršen, vendar pa v praksi ne deluje vedno tako, kot je promoviran. Glavni vzrok za to je dejstvo, da si uporabniki premislijo in spremenijo zahteve. To se lahko zgodi, ko pri predaji izdelka ugotovijo, da izdelek ni to, kar potrebujejo. Do sprememb zahtev uporabnika lahko prihaja sredi razvoja. Ko se zahteve spremenijo, je težko ustaviti zagon projekta in sprejeti spremembe. Tradicionalne metode pokažejo slabosti že pri manjšem številu sprememb, ker morajo programerji, arhitekti in

managerji uskladiti ogromne količine dokumentacije, ki mora biti posodobljena že zaradi manjših sprememb (Boehm, 1988).

1.2.2.3 Model CMM(I) ali (sestavljen) model zrelostnih nivojev

Eden od modelov razvoja programske opreme, ki je pogosto predstavljen kot tradicionalna metoda, je **model zrelostnih nivojev** (angl. *Capability Maturity Model*, v nadajlevanju CMM) oziroma njegova razširjena različica **sestavljen model zrelostnih nivojev** (angl. *Capability Maturity Model Integration,* v nadaljevanju CMMI) , ki je nastala leta 1998, in je osnovana na istih vrednotah. Cilj CMM je doseganje doslednosti, predvidljivosti in zanesljivosti procesov. CMM je okvir, ki opisuje ključne elemente učinkovitega procesa programske opreme. CMM opisuje pot evolucijskih izboljšav od ad hoc nezrelega procesa do zrelega, discipliniranega procesa (Paulk, Weber, Garcia, Chrissis & Bush, 1993).

Prvo verzijo CMM-ja je razvil Software Engineering Institute (v nadaljevanju SEI). Nastala je leta 1987 in je bila osnovana na osnovi vprašalnika, na podlagi katerega so v SEI postali pozorni na nasvete praktikov, ki so bili udeleženi v razvoju in izboljšavah programske opreme (Paulk et al., 1993).

Modela CMM in CMMI temeljita na ideji, da za izboljšanje učinkovitosti programske opreme niso dovolj nove tehnologije in orodja, temveč je treba poskrbeti predvsem za kakovosten, dobro definiran in nadziran proces razvoja programske opreme (Košir, 2010).

Petstopenjski CMM opisuje dobre inženirske in managerske prakse in pripisuje prioritete izboljšav podjetjem, ki se ukvarjajo z razvojem programske opreme. Model definira osemnajst ključnih procesnih področij in dvainpetdeset ciljev za organizacije za dosego organizacijskega nivoja 5.

Večina organizacij, ki se ukvarja z razvojem programske opreme ima zrelostno stopnjo *kaotične*, kar ustreza prvi stopnji CMM-ja in samo nekaj jih je optimiziranih, kar predstavlja stopnjo 5. Chrisis (2003) opisuje pet nivojev zrelosti:

1. začetni ali kaotični (angl. *Initial*),
2. voden (angl. *Managed*),
3. definiran (angl. *Defined*),
4. kvantitativno voden (angl. *Quantitatively Managed*) in
5. optimiziran (angl. *Optimized*).

Vsaka raven ustrezno definira karakteristike procesa, ki jih mora izpolnjevati organizacija, če želi doseči ta nivo zrelosti. Za vsako raven so določena procesna področja, splošni in posebni cilji ter splošne in posebne prakse.

Model CMMI se uporablja za ocenjevanje organizacij za razvoj programske opreme z dvojnim namenom: z namenom izboljšanja procesa razvoja programske opreme (s strani organizacije same) ter z namenom ugotovitve zmožnosti za izvedbo naročila (s strani naročnikov). Tak pristop zagotavlja objektivnost in merljivost rezultatov. Model zahteva vgraditev mehanizmov, ki omogočajo trajno doseganje zastavljenih ciljev.

CMM se v glavnem osredotoča na velike projekte in velike organizacije. Zagovorniki CMM metode trdijo, da se ta lahko prilagodi tudi manjšim projektom, saj je formulirana precej splošno in ustreza potrebam organizacije. Cilji CMM-ja so doseči konsistentnost, predvidljivost in zanesljivost procesov (Paulk, 2001). Zagovorniki agilnih metod trdijo, da CMM verjame, da je razvoj programske opreme definiran proces, ki je lahko detajlno opredeljen, medtem ko se sami s to predpostavko ne strinjajo. Zagovorniki agilnih metod modelu CMM očitajo tudi, da je preveč osredotočen na dokumentacijo.

1.2.3 Agilne metode razvoja programske opreme

Highsmith in Cockburn (2000) sta razložila, kaj za organizacijo pomeni agilno: pomeni biti sposoben hitro dobaviti ter hitro in pogosto spremeniti. Agilne metode so pravzaprav zbirka različnih tehnik ali praks, ki pa imajo skupne vrednote in bazična načela kot so iterativni ali ponavljajoči se razvoj ter osredotočanje na interakcijo, komunikacijo in zmanjšanje vmesnih del, ki porabljajo veliko virov. Razvoj in ponavljanja omogočajo razvojni skupini, da se hitro prilagaja spremembam v zahtevah. Delo na bližnjih lokacijah in fokusiranje na komunikacijo pomeni, da skupina lahko sprejema odločitve in jih takoj sprovaja, ne da bi čakala na korespondenco. Zmanjšanje vmesnega nepotrebnega dela, ki ne dodaja vrednosti končnemu izdelku, pomeni, da se lahko viri porabijo za razvoj samega izdelka, zato je delo hitreje končano (Cohen & Lindvall, 2004).

Agilne metode so reakcija na tradicionalne načine razvoja programske opreme in prepoznano potrebo po **težkem razvojnem procesu**, ki je določen z vnaprej pripravljeno dokumentacijo (Beck et al., 2001). Pri izvajanju tradicionalnih metod razvoja programske opreme se delo začne z zbiranjem in dokumentiranjem celotnega seta zahtev, ki mu sledi oblikovanje arhitekture programske opreme na višji ravni, razvoj in pregled. Na začetku 90-ih let dvajsetega stoletja so nekateri praktiki ugotovili, da so ti začetni koraki v razvoju omejujoči in celo nemogoči. Industrija in tehnologija se razvijata prehitro, zahteve se spreminjajo s hitrostjo, ki je tradicionalne metode niso mogle dohajati, hkrati pa končni uporabniki niso znali dovolj dobro in natančno vnaprej navesti in definirati svojih zahtev, pri čemer so imeli glede programske opreme velika pričakovanja. Kot rezultat tega so različni svetovalci neodvisno razvili metode in prakse, ki so bile odgovor na nujne spremembe, s katerimi so se soočali. Agilne metode so pravzaprav zbirka različnih tehnik (ali praks), ki temeljijo na istih vrednotah in osnovnih načelih (Cohen et. al., 2004). Večina agilnih praks ni novih (Highsmith & Cocburn, 2001), novi so osredotočenje in vrednote zajete v agilnih metodah, ki se razlikujejo od tradicionalnih metod.

Po Fowlerju (2003) obstajata dve ključni značilnosti agilnih metodologij, po katerih se te ločijo od tradicionalnih metod razvoja programske opreme:

- **Agilne metode so prilagodljive in ne predvidljive:** Tradicionalne metode razvoja načrtujejo razvojni proces zelo podrobno in to v daljšem časovnem razponu. Tako zasnovane se po naravi upirajo spremembam. Nasprotno pa so pri agilnih metodah spremembe dobrodošle. Uspejo se prilagoditi spremembam do te mere, da se po potrebi spremenijo tudi same agilne metode.
- **Agilne metode so bolj usmerjene k ljudem kot k procesom:** Cilj tradicionalnih metod je, da definirajo proces, ki deluje, ne glede na to, kdo ga izvaja. Pri agilnih metodah velja, da noben proces ne more nadomestiti znanja in veščin razvojnega tima. Proces mora podpirati razvijalce pri njihovem delu.

1.2.3.1 Agilni manifest in osnovna načela agilnih metodologij

Temelji agilnih metodologij so bili postavljeni decembra 2001, ko se je sestala skupina sedemnajstih gurujev s področja **lahkih metodologij**, ki so se takrat že uporabljale (na primer ekstremno programiranje, metoda Scrum, agilno modeliranje, kristalne metode), z namenom ugotoviti, kaj imajo njihove metodologije skupnega. Na podlagi tega so želeli izpostaviti skupne metodološke osnove, čeprav njihov namen ni bil poenotenje lahkih metod (Cohen et al., 2004).

Člani agilne zaveze so v manifest agilnega razvoja PO zapisali: »Odkrivamo boljše načine razvoja programske opreme tako, da jo razvijamo, in pri tem pomagamo tudi drugim« (Manifest, 2001).

Iz tega so izpeljali štiri osnovna načela (Manifest, 2001):

- **Posamezniki in njihova komunikacija so pomembnejši od samega procesa in orodja:** To načelo izhaja iz ugotovitve, da se je potrebno individualno posvetiti članom projekta, njihovemu znanju in željam. To načelo poleg upoštevanja posameznikov poudarja pomen komunikacije. Dobra komunikacija je ključna za uspeh projekta. Rezultati so dokazano boljši v skupini, ki dobro sodeluje, četudi se člani ne držijo predpisanega procesa, kot v skupini, ki se drži procesov, vendar člani med seboj ne komunicirajo dovolj dobro.
- **Delujoča programska oprema je pomembnejša od popolne dokumentacije:** Največ, kar lahko uporabnik dobi, je delujoč program. Dokumentacija o problemski domeni in samem sistemu je koristna, vendar drugotnega pomena. Naročnik ne bo zadovoljen z dokumentacijo, če mu delujoč program ne bo prinesel rešitve. Po drugi strani je dokumentacija pomembna, ker olajšuje vzdrževanje sistema in komunikacijo med naročnikom in izvajalcem. Zato je izdelava dokumentacije pomembna, vendar s poudarkom, da ni preobsežna in da je vsak njen del utemeljen.

- **Sodelovanje uporabnika je pomembnejše od pogajanja na osnovi pogodb:** To načelo se osredotoča na odnos med naročnikom in izvajalcem, ki je v mnogih primerih preveč formalen in strog. Naročnik najbolje ve, kaj potrebuje, zato je v agilnih metodah vloga naročnika postavljena na prvo mesto. Dober odnos med naročnikom in izvajalcem lahko olajša tehnološko zahteven projekt.
- **Upoštevanje sprememb je pomembnejše od sledenja planu:** Moderne metodologije ugotavljajo pomen obvladovanja sprememb. Zaradi dinamike poslovnih okolij je v začetnih fazah projekta nemogoče zajeti vse zahteve. Projektni načrti so koristni, vendar morajo dovoljevati spremembe. Planiranje in sledenje planu sta koristna samo do stanja, ko se to ne razlikuje preveč od dejanskega.

Iz teh osnovnih načel agilnih metodologij je izpeljanih več usmeritev, ki jih vse metodologije, ki se deklarirajo kot agilne, upoštevajo (Manifest, 2001):

- Najvišja prioriteta je zadovoljstvo naročnika, ki ga dosežemo z zgodnjim in nenehnim posodabljanjem delujoče programske opreme.
- Delujočo programsko opremo je potrebno izdajati pogosto, znotraj obdobja od nekaj tednov, do nekaj mesecev. Boljši so krajši cikli, iterativen pristop je nujen.
- Spremembe zahtev so mogoče celo v poznih fazah razvoja. Spremembe, ki nastanejo med projektom in so upoštevane, lahko prinesejo stranki pomembno konkurenčno prednost, zato se jih agilni procesi ne izogibajo.
- Najboljša in najučinkovitejša metoda posredovanja informacij razvojni ekipi in znotraj ekipe same, je pogovor iz »oči v oči«.
- Projekti naj vključujejo motivirane posameznike. Omogočena naj jim bosta delovno okolje in podpora pri delu. Potrebno jim je zaupati, da bodo svoje delo opravili.
- Kakovost arhitekture programa in programske kode je potrebno stalno preverjati in izboljševati. Zaradi potrebe po čimprejšnjih rezultatih večkrat trpi kakovost. V rednih časovnih razdobjih ekipa išče načine, kako postati učinkovitejša ob rednem prilagajanju svojega delovanja.
- Enostavnost procesa je ključ do uspeha. Delo mora biti opravljeno kakovostno, vendar brez nepotrebnega kompliciranja.
- Agilni procesi promovirajo trajnostni razvoj. Sponzorji, razvijalci in uporabniki morajo biti zmožni konstantnega tempa za nedoločen čas.
- Najboljše arhitekture, zahteve in načrti izhajajo iz tistih ekip, ki so samoorganizirane.

Skupna značilnost agilnih metod je, da vsebujejo iterativni razvoj in osredotočanje na interakcijo, komunikacijo in zmanjšanje vmesnih rezultatov dela, ki zahtevajo dosti virov. Iterativni razvoj dovoljuje timu, da se hitro prilagaja spremembam zahtev. Delo na bližnjih lokacijah in osredotočanje na komunikacijo pomenita, da se tim lahko odloča takoj v skladu z dogovori, namesto da čaka na dopise. Zmanjšanje vmesnega dela ne dodaja

vrednosti izdelku, ampak pomeni, da se lahko več virov posveča razvoju in izdelku samemu: tako je delo lahko prej končano (Higsmith, 2001).

Agilne metode razvoja programske opreme poudarjajo sposobnost obravnavanja nenehnih sprememb. A četudi je kaskadni model tipično predstavljen kot zgodovinska relikvija in so bile v zadnjih desetletjih predstavljene številne agilne metode, se prepogosto dejanske aktivnosti v razvoju še vedno izvajajo po rigidnih, nefleksibilnih in pretežno sekvenčnih procesih (Fugetta 2014).

Najpopularnejše agilne metode so (Cohen et al., 2004):

- metoda Scrum,
- ekstremno programiranje (angl. *Extreme Programming* ali XP)
- kristalne metode (angl. *Crystal Methods*),
- metoda dinamičnega razvoja programske opreme (angl. *Dynamic Systems Development Method* ali DSDM),
- razvoj, ki ga usmerjajo funkcije (angl. *Feature Driven Development* ali FDD)

1.2.3.2 Metoda Scrum

Metoda Scrum je poleg ekstremnega programiranja (v nadaljevanju XP) najbolj uporabljana agilna metoda. Nastala je z namenom, da projektnemu načrtovanju in izvajanju da energijo, osredotočenje, jasnost in transparentnost. Danes se metoda Scrum uporablja v malih, srednje velikih in velikih podjetjih, ki se ukvarjajo s PO (Suthherland, 2010).

Scrum je popularna in uspešna metoda. Njeno bistvo je sprejemanje miselnosti, da lahko stranka spremeni svoje zahteve kadarkoli, tudi med samim izvajanjem projekta. Prva jo je leta 1993 uporabila skupina entuziastov v podjetju Easel Corporation (Schwaber, 1996). Schwaber (1996) je metodo Scrum prvi opisal kot metodo, ki sprejema dejstvo, da je proces razvoja nepredvidljiv, da formalizira »stori kar je potrebno« mentaliteto in kot metodo, ki se je pokazala uspešna pri številnih neodvisnih dobaviteljih PO.

Po metodi Scrum je proces organiziran v kratkih razvojnih ciklih, imenovanih **sprinti,** ki niso daljši od štirih tednov. Dolžina vsakega sprinta je vnaprej določena, sprinti pa si sledijo drug za drugim. Vsaka ponovitev je osredotočena na specifičen set zahtev naročnika, ki jih ekipa izbere in prioritizira. Tim se zaveže, da bodo zahteve rešene pred koncem sprinta. Sledenje procesu in kontrola se izvajata dnevno na kratkih sestankih, na katerih vsak član opiše, kaj je delal dan pred tem, kaj bo delal ta dan in izpostavi težave, kadar se pojavijo. Taka srečanja pomagajo identificirati težave zelo zgodaj in omogočajo načrtovanje sprememb, če so potrebne. Spremembe v zahtevah kupca so sprejete in

dobrodošle, vendar se ne upoštevajo v tekočem sprintu. So opisane ter prioritizirane in se lahko upoštevajo v naslednjih sprintih. Na koncu vsakega sprinta se funkcije, na katerih so delali, dokončane. Niso samo izvajane, ampak so tudi preverjene in dokumentirane, tako da so lahko dostavljene uporabniku. Povratne informacije, ki jih pridobijo, pa so vključene v naslednji sprint. Scrum poudarja delujoči izdelek oziroma PO na koncu vsakega sprinta, kar pomeni, da je ta PO integrirana, testirana in potencialno pripravljena za predajo (Highsmith & Cockburn, 2001).

Ključna načela razvojnega procesa metode Scrum so (Scrum Principles, 2016):

- Empirična kontrola procesa: To načelo poudarja jedro filozofije na kateri je osnovan Scrum: transparentnost, preverjanje in prilagoditev. V Scrumu odločitve bazirajo na eksperimentih in ne na vnaprejšnjem načrtovanju.
- Samoorganiziranje: Delavci predstavljajo večjo vrednost če so samoorganizirani. To ustvari inovativno in kreativno okolje, ki je bolj naklonjeno rasti.
- Sodelovanje: Nanaša se na skupinsko delo. Zagovarja, da je projektno vodenje skupni proces ustvarjanja vrednosti z različnimi timi, ki delajo in se povezujejo.
- Časovni okvirji: Ta element je zajet v skoraj vseh fazah Scrum-a kot so sprinti, dnevni sestanki, sestanki za načrtovanje sprintov in sestanki za pregledovanje sprintov.
- Iterativni razvoj: Poudarja kako bolje obvladovati spremembe in razvijati izdelke za potrebo kupcev.

V kolikor je metoda Scrum pravilno implementirana, pripomore k hitrejšemu razvoju, razvrstitvi individualnih in poslovnih ciljev, ustvarjanju okolja, ki ga poganja zmogljivost, doseganju stabilne in dosledne komunikacijo na vseh ravneh ter izboljšanju razvoja posameznika in večji kakovosti življenja (Sutherland, 2010).

1.2.3.3 Ekstremno programiranje ali XP

Ekstremno programiranje je danes ena najbolj popularnih agilnih metod za razvoj programske opreme (Beck, 1998). Po Beckovem mnenju (Beck, 2000) je ta metoda lahka, učinkovita ter prilagodljiva in ne prinaša tveganja. Ekstremno programiranje namreč naslavlja tveganja, kot so spodrsljaji pri časovnih rokih, prekinitev projekta, stopnja napak, napačno razumevanje problema, menjava zaposlenih na vseh nivojih razvojnega procesa.
Za ekstremno programiranje je značilno, da upošteva štiri spremenljivke: stroške, čas, kakovost dela in obseg dela. Tri spremenljivke lahko določajo zunanji faktorji (stranke, uprava, vodstvo projekta), četrto – običajno je to obseg dela – pa določa razvojni tim.

Kent Beck (2000) je v svoji knjigi »Extreme Programming – Explined« združil izkušnje in jih povzel v dvanajst praks ekstremnega programiranja (Cohen et al., 2004, str.13):

- **Igra planiranja:** Pri razvoju programske opreme je vedno prisoten dialog med poslovnim in tehničnim delom razvijalske ekipe o tem, kakšne so možnosti in kakšna so hotenja glede izdelka. Taki dialogi so »igra planiranja«.
- **Majhne in pogoste izdaje:** Čim manjša izdaja, ki pa predstavlja zaokroženo celoto. Izdaja mora vsebovati najbolj pomembne zahteve, priporočen cikel izdaje pa je nekaj tednov.
- **Metafora** je sistem imen in opisov, nabor nedvoumnih terminov, ki udeležencem olajša komunikacijo ter povezuje tehnični in poslovni sistem.
- **Preprost načrt:** Sistem mora biti zasnovan čim bolj preprosto, tako da še zadostuje zahtevam naročnika.
- **Testno voden razvoj** je iterativni proces, kjer razvijalec napiše avtomatske teste, pred kodo. Mnogi zagovorniki ekstremnega programiranja menijo, da je testno voden razvoj praksa, ki največ pripomore k izboljšavam produktivnosti, predvsem pa h kakovosti izdelka.
- **Preoblikovanje** naredi vrsto majhnih, a pomembnih transformacij v izvorni kodi, ki ne vplivajo na pravilno delovanje kode, ampak na preprosto načrtovanje sistema. Transformacija sistema se zgodi, ko se vse majhne transformacije kombinirajo.
- **Programiranje v parih** je zaradi posebnosti verjetno najpopularnejši termin v ekstremnem programiranju. Predstavlja razvoj programske opreme, kjer dva programerja skupaj razvijata isto celoto izvorne kode. Programiranje poteka na način da, en razvijalec tipka kodo, drugi pa išče napake in izboljšave.
- **Skupno lastništvo kode** vsakemu paru dodeli pravice, da pregleda katerikoli modul in ga izboljša. Skupno lastništvo kode omogoča svobodo razvijalcem, da se naučijo kaj novega izven njihove specializiranosti.
- **Stalna integracija:** Programski izdelki morajo biti integrirani v kratkih razvojnih ciklih. Še preden se predajo stranki v uporabo, morajo biti večkrat dnevno testirani in integrirani v delujočo celoto.
- **40-urni delovnik:** Projektno zasnovano delo zahteva od programerjev popolno koncentracijo, zato običajno terja velike napore. Ekstremno programiranje že ob načrtovanju časa razvoja predvideva 8-urni delovnik. V kolikor programerji pretiravajo z delavnostjo, se to pogosto kaže v kakovosti izdelka, lahko pa celo pripelje do izgorelosti posameznikov.
- **Kupec ob strani:** Pri razvoju programske opreme je zelo pomembno, da je pridobivanje zahtev od uporabnika hitro, kar pa je mogoče samo, če je kupec »na voljo«. S tem se tudi zmanjša tveganje za morebitno napačno interpretiranje zahtev.
- **Standardi kodiranja:** Pravila kodiranja morajo biti jasna vsem programerjem. Ta praksa se navezuje na skupno lastništvo kode, ki omogoča, da lahko vsak spreminja katerikoli del kode. Zato je skupno razumevanje zgradbe programskega jezika ključnega pomena.

Prednost ekstremnega programiranja je, da se vse prakse združujejo v smiselno celoto in se med seboj dopolnjujejo. Po letu 2004 je nastala razširjena verzija ekstremnega programiranja, vendar se osnovna načela niso spremenila (Cohen et al., 2004).

1.3 Vitek razvoj programske opreme

Dejstvo je, da je danes informacijska tehnologija (v nadaljevanju IT) ključnega pomena skoraj za vse panoge. V desetletjih, odkar so bili računalniki uvedeni na delovna mesta, se je njihova vloga spremenila iz zgolj avtomatizirane obdelave transakcij v dosti bolj strateško orodje (Maguire, 2016). Bell (2013) pravi, da živimo v času, ko je strokovna uporaba IT nujna komponenta prepoznane vrednosti vsakega podjetja, za kar navaja tri razloge: zmogljivosti IT so vgrajene v skoraj vsak izdelek in storitev, obvladovanje IT omogoča podjetjem, da bolje ponujajo svoje storitve kupcem, z uporabo IT bolje razumemo kupce in njihovo obnašanje (Bell, 2013 v Maguire, 2016).

Vitko mišljenje je uveljavljen poslovni sistem, ki se nanaša na tok, vrednost in izgube (Womack & Jones, 2003). Vendar sta »vitko« in IT tradicionalno tudi v nasprotju, saj so med njima tudi temeljne razlike. Vitko mišljenje na primer zagovarja preprostost, medtem ko računalniški sistemi vnašajo v delo kompleksnost. Iz razvoja obeh vej izhaja tudi nasprotje med uporabo načela vlečenja (vitko) in potiskanja (IT) (Crabtree & Astral, 2006).

Koncept agilnega razvoja PO je bil predstavljen leta 2001, njegov ključni cilj pa je bil sposobnost zadostnega in učinkovitega odziva na spremembe, ki jih narekujejo uporabniki. Kar nekaj načel Agilnega manifesta, kot sta preprostost in učenje z eksperimentiranjem, je tesno povezanih z vitkim mišljenjem. Približno v tem času je Mary Poppendick (2001) predstavila koncept vitkega programiranja, pri čemer je navedla, da so metode, kot so agilne, učinkovito uporabljena vitka načela v razvoju PO (Poppendick, 2001).

Pomembno vprašanje, ki se postavlja, je, kako se lahko vitke proizvodne prakse, ki so tipično ponavljanje identičnih del, in posledično proizvodnja istega izdelka, nanašajo na aktivnosti pri razvoju izdelka, katerih rezultat je vedno nekaj drugega (Cawley, Wang & Richardson, 2013). Vitek razvoj programske opreme je prenos Toyotinega proizvodnega sistema ter vitkih načel in praks v informacijsko tehnologijo, v razvoj programske opreme. Bistvo, na katerega so osredotočeni zagovorniki vitkega razvoja programske opreme, je ustvarjanje vrednosti na način, da se identificirajo in odstranijo izgube. Vendar pa lahko tako interpretacija izgub kot način, kako jih obravnavati, variirata. Medtem ko Mary in Tom Poppendick (2003) ter Hibbs (2009) identificirajo posamezne izgube in zagovarjajo njihovo takojšnjo odpravo, Reinerstein (2009) predlaga, da je smiselno in koristno odločati se, ali je določena aktivnost izguba in kako jo obravnavati samo, če se ta določena aktivnost, ki lahko predstavlja izgubo, pretvori v ekonomske pogoje. Dodatno je Ballard

(2000) že prej opozarjal na izgube pri negativnih ponavljanjih. Negativna ponavljanja je imenoval tista ponavljanja, ki se lahko opustijo, ne da bi izdelek izgubil vrednost.

Tabela 3.1. prikazuje primerjavo izgub pri razvoju programske opreme z izgubami pri vitki proizvodnji, kot so jih definirali Liker (2006), Poppendick in Poppendick (2003) in Hibbs et al. (2009):

Tabela 1: Izgube pri razvoju programske opreme

Vitka proizvodnja	**Vitek razvoj PO**
prekomerna proizvodnja	dodatne funkcije / kode
čakanje na dodeljeno delo	zaostanki pri delu
prevoz	preklapljanje med različnimi deli
prekomerno ali nepravilno procesiranje	dodatni procesi
razpoložljive (prekomerne) zaloge	nedokončano delo
premikanje	premikanje
izdelava defektnih izdelkov	napake v programski opremi
neizkoriščena kreativnost zaposlenih	neizkoriščena kreativnost zaposlenih

Vir: O. Cawley, X. Wangl. & I. Richardson, Lean Enterprise Software and Systems, 2013, str. 22.

1.3.1 Vitko ali agilno

Vitek razvoj programske opreme je paradigma, za katero je vedno več zanimanja med tistimi, ki se zavzemajo za znižanje stroškov v tej panogi. Izraz vitek privlači pozornost v poslu, vendar o tem, kako se natančno uporablja na področju programske opreme, še vedno poteka debata. Dodatno je njegovo razmerje z bolje poznanimi agilnimi metodami tudi predmet razprav (Cawley et al., 2013).

Želja po zniževanju stroškov in po večji učinkovitosti je prisotna na skoraj vsakem področju poslovanja v razvitih državah. Tradicionalna industrija je učinkovitost in zniževanje stroškov vpeljala s pomočjo avtomatizacije in organizacijskega prestrukturiranja bolj rutinskih in ponavljajočih se procesov. V mnogih industrijah uporabljajo informacijsko tehnologijo vključno z razvojem programske opreme in iščejo možnosti, kako bi te visoko tehnološke funkcije prispevale k učinkovitejšemu načinu poslovanja. Potreba postati in ostati konkurenčen je pripeljala do tega, da se pojem »vitko« uporablja na področjih izven industrije.

Izraz **vitek razvoj programske opreme** se je začel uporabljati leta 2003, ko sta ga uvedla Mary in Tom Poppendick v svojem delu »Lean Software Development: An Agile Toolkit« (2003). Zelo pogosto imajo podjetja »vitko etiko« in vodijo »vitke« kampanje, vendar ne

smemo predpostavljati, da vsaka aktivnost s pridevnikom »vitko« avtomatsko generira bolj stroškovno učinkovite procese ter sili zaposlene z novim smislom in smernicami.

S pojavom vitkega razvoja PO je nastala zmeda glede razlik oziroma podobnosti med vitkim razvojem PO in agilnim razvojem PO (Cawley et al., 2013). Medtem ko Cohen et al. (2004) navajajo vitek razvoj programske opreme kot enega od agilnih metod razvoja programske opreme, ga večina drugih avtorjev (Poppendick & Poppendick, 2003; Charette, 2007) razume kot nadgradnjo, v kateri se lahko uporabijo agilne, vitke ali splošne dobre prakse razvoja programske opreme. Cawley et al. (2013) vidijo ključno razliko med vitkim in agilnim v pristopu: agilne metode razvijajo od spodaj navzgor, medtem ko se vitek pristop začne pri vrhu in gre navzdol. Vitek razvoj uporablja več orodij, ki so bila že dobro razvita v agilnih praksah. To je skupinam, ki so že uporabljale agilne metode, olajšalo sprejetje vitkih pristopov, vendar je hkrati vneslo tudi nekaj zmede glede razlik med agilnim in vitkim. Posledično je debata o meji med vitkim in agilnim razvojem programske opreme še vedno odprta (Cawley et al., 2013).

Precej agilnih praks je uporabljenih v orodjih vitkega razvoja programske opreme, ki sta ga razvila Mary in Tom Poppendick (2003). V vitkem razvoju programske opreme pa so tudi načela, ki jih v agilnih metodah ne najdemo, kot na primer odlaševanje z odločitvami v razvoju programske opreme dokler je mogoče.

1.3.2 Elementi vitkega razvoja programske opreme

Mary in Tom Poppendick (2003) sta vitek razvoj programske opreme osnovala na sedmih modificiranih vitkih načelih in miselnih pripomočkih, poznanih iz Toyotinega proizvodnega sistema. Sedem načel, ki sta jih opisala v knjigi »Vitek razvoj programske opreme« je:

1. odprava izgub,
2. krepitev znanja,
3. pozno odločanje,
4. dobaviti, kakor hitro je mogoče,
5. pooblaščanje tima,
6. vgradnja integritete,
7. videnje celote.

Temeljno načelo vitke proizvodnje in načelo, iz katerega izhajajo vse nadaljnje usmeritve, je odpravljanje izgub. Izgube so v smislu proizvodnje jasno in natančno določene. Taiichi Ohno pa je dal izgubam nov pomen, saj je vanje vključil vse, kar ne prispeva k vrednosti za kupca: proizvajati nekaj, kar ni takoj potrebno, premikanje, transport, čakalne dobe. Toyota je tak koncept izgub prenesla iz proizvodnje v razvoj izdelkov. Tako je že na samem začetku razvojnega procesa cilj, da se ta čim hitreje zaključi. Dela, ki so vložena v

razvoj izdelka, – od dizajna do prototipov – nimajo dodane vrednosti, dokler izdelek ni ponujen tržišču.

1.3.2.1 Odprava izgub

Videnje izgub je prvi korak in razvojni preboj na področju vitkega mišljenja. Kot osnovno orodje za ocenjevanje in izboljšavo razvoja programske opreme in spregledanja izgub Poppendick M. in Poppendick T. (2003) predlagata načrt toka vrednosti: najprej za samo organizacijo, ki razvija programsko opremo, v naslednjem koraku pa še razširjen zemljevid vrednosti, ki vključuje tudi kupce. Če organizacija razume, kako kupci kreirajo vrednost, je to v veliko pomoč pri uresničevanju. Učinkovitost načrta toka vrednosti je predvsem v tem, da odtegne pozornost z organizacije, sredstev, tehnologij in procesov ter osredotoči pozornost na izdelek in njegovo vrednost. Načrt toka vrednosti se začne s sprejetjem zahteve kupca in mapiranjem časovnice, ki kaže napredek v zagotavljanju vrednosti za kupca. Potrebno je ugotoviti koliko časa je porabljeno za aktivnosti, ki dodajajo vrednost in koliko časa za čakanje (Steind, 2004).

Sedem izgub v razvoju programske opreme, ki so prenesene iz Toyotinega proizvodnega sistema so (Poppendick & Poppendick, str.4):

- **Delno opravljeno delo**: Nedokončano delo v razvojnem procesu ima tendenco, da postane zastarelo in da otežuje razvoj drugega razvoja, ki bi se mogoče moral odvijati hkrati. Problem nedokončane programske opreme je tudi, da se mogoče nikoli ne bo izkazalo, ali dejansko deluje. Nedokončano delo veže nase vire in sredstva, ki bi že morali dajati rezultate.
- **Dodatni procesi**: Velikokrat "papirologija" upočasnjuje razvoj PO, ker porablja vire, upočasnjuje odzivni čas in zakriva probleme v kvaliteti. Vendar pa razvojni procesi zahtevajo dokumentacijo kot potrdilo za kupce, za sledljivost ali zaradi odobritev sprememb. Pri vsaki dokumentaciji se je potrebno vprašati, če v očeh kupca dodaja vrednost izdelku. V kolikor je dokumentacija zahtevana, mora biti kratka in kvalitetno napisana.
- **Dodatne funkcije:** Mogoče se zdi dodati funkcije ali nove tehnične zmogljivosti v razvoj sistema za vsak slučaj, dobra ideja. To je resna izguba, saj mora biti vsak bit kode sledljiv, zbran, integriran in testiran vsakokrat, ko se koda spreminja, in potem vzdrževan do konca življenjskega cikla sistema. Vsaka dodatna funkcija povečuje kompleksnost in možnost napake.
- **Preklapljanje med nalogami:** Ko razvijalci programske opreme preklapljajo med delom, obstaja določen čas, da lahko zberejo misli in se vklopijo v tok novega dela (DeMarco & Lister, 2013). Preklapljanje med različnimi opravili je izguba časa. V kolikor so ljudje razporejeni na različna dela, je najhitrejši način za dokončanje projektov ta, da opravljajo samo eno delo naenkrat.

- **Čakanje** je ena največjih izgub v razvoju programske opreme. Začne se pri zamudah ob začetku projekta, iskanju sodelavcev in pisanju zahtev uporabnikov. Čeprav so zamude v večini procesov razvoja programske opreme pogoste, je najslabše razmišljanje, da so »normalne« in da ne vplivajo na rezultat. Zamuda onemogoča kupcem realizirati vrednost v najkrajšem možnem času.
- **Premikanje:** Razvoj je aktivnost, ki zahteva veliko koncentracijo, zato premikanje z namenom, da razvijalec dobi odgovor ali da testira izdelek, vzame bistveno več časa, kot je videti. Zaradi tega je priporočljivo, da razvojni tim dela v enem delovnem prostoru, kjer imajo vsi dostop do razvijalcev, testov, strank ali predstavnikov strank. Premikajo se tudi artefakti. Največja izguba je, če dokumenti pri predaji naslednjemu členu, ne vsebujejo vseh informacij, ki jih naslednja oseba potrebuje. Pri tem gre velikokrat za neotipljivo znanje, ki dostikrat ni predano naslednjemu v verigi.
- **Okvare:** Izguba zaradi napak je zmnožek vpliva napake in časa, ki je potekel do odkritja napake. Kritična napaka, ki je odkrita v treh minutah, velikokrat ni tak vir izgube, kot so manjše napake, za odkritje katerih potrebujemo nekaj tednov. Način, kako odpraviti izgube, je odkrivanje napak v čim krajšem času po njihovem nastanku.

Managerske aktivnosti sicer ne predstavljajo direktne dodane vrednosti izdelka, vendar imajo v organizaciji velik vpliv na zamude. Sledenje procesov in kontrola izdelkov ne dodajata vrednosti. Če je sledenje procesov prezapleteno, obstaja najverjetneje več drugih vrst izgub. Sistemi avtorizacije, ki so vzpostavljeni z namenom odobravanja sprememb v zahtevah, prav tako pogosto povzročajo zamude.

1.3.2.2 Krepitev znanja

Razvoj in proizvodnja sta zelo različni aktivnosti, zato morajo biti pri obeh uporabljeni drugačni pristopi. Ballard (2000) primerja snovanje oziroma razvoj izdelka s pripravo recepta, proizvodnjo pa s pripravo obroka. Razliko med razvojem in proizvodnjo je povzel v tabeli 3.2..

Tabela 2: Razvoj oziroma zasnova izdelkov v primerjavi s proizvodnjo

ZASNOVA IZDELKA	PROIZVODNJA
Ustvarjanje recepta.	Priprava obroka.
Kakovost je realizacija namena.	Kakovost je skladnost z zahtevami.
Variabilnost rezultatov je zaželena.	Variabilnost rezultatov ni zaželena.
Ponavljanje lahko generira vrednost.	Ponavljanje generira izgube.

Vir: G. Ballard, Positive vs negative iteration in design, 2000, str. 20.

V proizvodnji je kakovost definirana kot skladnost z zahtevami, ki so specificirane v načrtu izdelka, medtem ko kakovost v razvoju programske opreme rezultira v sistem z zaznano in konceptualno celovitostjo ter skladnostjo (Poppendick & Poppendick, 2003, str. 20).

Zaznana skladnost pomeni, da celota izdelka doseže ravnotežje med funkcijo, uporabnostjo, zanesljivostjo in ekonomiko. Kupec programske opreme bo zaznal skladnost, če bo programska oprema rešila njegov problem, če bo enostavna za uporabo in če bo njena uporaba stroškovno sprejemljiva. Konceptualna integriteta pa pomeni, da osnovni koncepti sistema delujejo skupaj gladko in kot povezana celota.

Proizvodnja predvideva homogen in nespremenljiv set zahtev kupcev, tako da je cilj izdelava vedno enakega izdelka. Ideja, da so variante v razvoju programske opreme slabe, se je pojavila, ko so z namenom zmanjšanja števila variant poskušali razviti standardiziran proces in tako doseči vsakokrat ponovljive rezultate. Vendar namen razvoja ni produkcija ponovljivih rezultatov, saj razvoj proizvaja primerne rešitve za edinstvene probleme kupcev (Poppendick & Poppendick, 2003, str. 21). Ko se organizacija sooča s težavami pri razvoju programske opreme, obstaja tendenca, da se vzpostavi bolj organiziran proces z bolj strogimi zaporednimi koraki: več zahtev se dokumentira, dogovori s kupci so napisani, spremembe so bolj kontrolirane, vsaka zahteva se sledi do programske kode. Te dodatne kontrole podaljšujejo zanko povratnih informacij in poslabšujejo proces razvoja. V primeru težav je potrebno povečati tok povratnih informacij. To se lahko doseže tako, da se namesto sprejemanja dodatnih zahtev, kupcu pokaže več variant izdelka in na podlagi teh pridobi dodatne informacije, da se testi izvajajo takoj, ko je napisana programska koda in da se namesto razmišljanja o tem, katero orodje je najprimernejše, kupcu omogoči poskusiti tri najprimernejša orodja, tako, da se kupec odloči na podlagi preizkusa (Steindl, 2004).

Zasnova novih izdelkov je proces, ki zajema odkrivanje rešitev s pomočjo kratkih, ponavljajočih se ciklov raziskav, eksperimentov in preverjanj rezultatov. Razvoj programske opreme je najbolj naravno izveden skozi takšne ponavljajoče se cikle ali iteracije. Iteracija je uporaben prirastek programske opreme, ki je zasnovan, programiran, testiran, integriran ter dobavljen v kratkem, fiksnem časovnem okvirju. (Poppendick & Poppendick, 2003, str. 23). Iteracije so tudi točke sinhronizacije – razvojni tim in kupec vidita napredke in dosežke (Steindl, 2004). Programska oprema bo izboljšana v nadaljnjih iteracijah, vendar je že od samega začetka delovna, testirana in integrirana programska koda. Iteracije zagotavljajo več povratnih informacij in dosti boljšo komunikacijo med kupci oziroma uporabniki in razvijalci, kot je to pri »standardnem« sekvenčnem razvoju programske opreme. Najučinkovitejše je učenje v veliko krajših učnih ciklih, kjer hitro dobimo povratne informacije. Razvoj PO temelji na setu rešitev, pri čemer razvijalci razvijejo set alternativnih rešitev, preverijo, kako delujejo in na podlagi tega izberejo eno alternativo ali pa združijo najboljše funkcionalnosti iz različnih alternativ. S kupci komunicirajo o omejitvah in ne o izbirah, kar vodi hitreje do boljših odločitev (Steindl, 2004).

Kadarkoli več ljudi dela na istem projektu, je potrebna sinhronizacija, pri čemer je pri razvoju PO potrebna tako sinhronizacija v timu (dnevno ali tedensko integriranje delov programa) kot sinhronizacija s kupci (prioritiziranje zahtev) (Steindl, 2004).

1.3.2.3 Pozno odločanje

Podobnost razvoja programske opreme in izrezovanja kalupa za avtomobilsko šasijo je predvsem v tem, da je vanju vloženih veliko dela in sredstev, ter da so napake lahko zelo drage. Zato se zaporedni razvoj, ko se zahteve končnega izdelka določijo, preden se začne razvoj samega izdelka, pogosto zdi smiseln. Problem zaporednega razvoja izdelka je, da sili oblikovalce, da se že na začetku ukvarjajo z načrti, ki gredo v globino izdelka, namesto da bi razvoj zastavili dovolj široko. Razvoj izdelka v globino sili v sprejemanje odločitev na nižji stopnji, še preden se preverijo posledice odločitev na višji stopnji. Najdražje napake se zgodijo, če v samem začetku pozabimo upoštevati kaj pomembnega za izdelek. Največkrat se take napake zgodijo, če se v samem začetku razvoja razvijalci prehitro začnejo ukvarjati z detajli (Poppendick & Poppendick, 2003, str. 48).

Sočasni razvoj programske opreme običajno poteka na osnovi iteracij. Predvsem se tak način dela preferira v primeru, ko so zaradi morebitnih napak možne velike izgube in ko se razumevanje problemov v zvezi z izdelkom prav tako še postopoma razvija. Sočasen razvoj omogoča odkrivanje velikih, dragih problemov, preden je prepozno. Prehod iz zaporednega na sočasni razvoj pomeni, da se programiranje funkcij, ki imajo najvišjo dodano vrednost, začne takoj po tem, ko je določena konceptualna zasnova na višji ravni, čeprav so bolj detajlne zahteve še v fazi proučevanja. Poleg tega, da zagotavlja zavarovanje pred dragimi napakami, je sočasen razvoj najboljši način, kako se ukvarjati s spreminjajočimi se zahtevami. Ne samo, da se večje odločitve odložijo, medtem ko se proučujejo vse možnosti, tudi manjše odločitve se prestavijo. Kadar so spremembe neizogibne, sočasni razvoj skrajša rok dobave in zmanjša skupne stroške ter hkrati izboljšuje delovanje izdelka.

Programska oprema se od večine drugih izdelkov loči po tem, da se od nje pričakuje, da bo redno posodabljana. V povprečju nastane več kot polovico razvojnega dela v sistemu programske opreme po tem, ko je le ta prodana, oziroma dana v proizvodnjo (Kajko – Mattsson et al., 2001). Za večino PO je pričakovano, da se bo v svojem življenjskem času redno spreminjala. Sistemi, kjer se uporablja, kot na primer: novi operacijski sistem, spremembe v podatkovnih bazah, nove aplikacije programske opreme, so predmet sprememb v njihovem okolju. Ko se PO ne posodablja več, je pogosto blizu konca svojega življenjskega cikla.

Vse spremembe niso enake. Nekaj osnovnih odločitev, kot so jezik programske opreme, odločitev o nivojih arhitekture, izbira, ki se tiče interakcije z obstoječimi podatkovnimi bazami, je potrebno sprejeti takoj na samem začetku razvoja. Te odločitve določijo

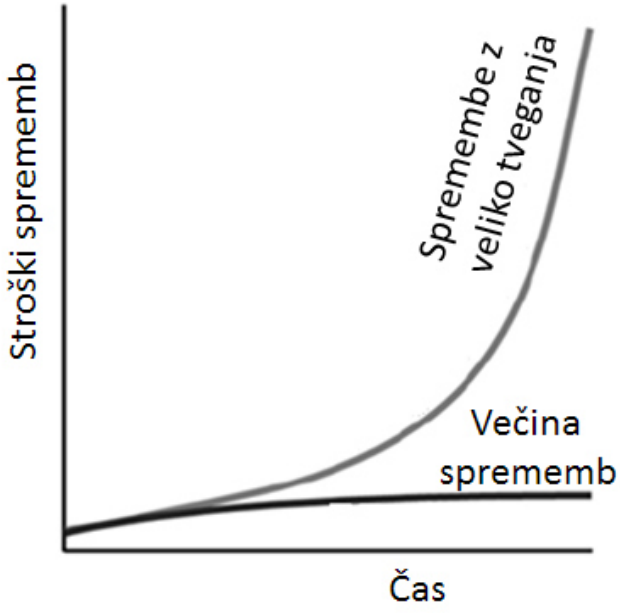
omejitve sistema v celotnem življenjskem obdobju. V primerjavi z drugimi imajo lahko te odločitve stokrat večje razmerje v stopnjevanju stroškov. Ker so tako kritične, jih je dobro minimizirati. Večina sprememb v sistemu nima takšnega faktorja stopnjevanja stroškov.

Sočasen razvoj odlaša z odločitvami, dokler je to mogoče, kar ima naslednje posledice (Poppendick & Poppendick, 2003, str. 50):

- zmanjšuje število omejitev, ki so stroškovno kritične,
- omogoča širok pristop k odločitvam, ki vključujejo verjetnost velikih denarnih izgub, zaradi česar je verjetneje, da bodo te odločitve pravilne,
- odlaša z večino odločitev, kar občutno znižuje potrebo po spreminjanju,
- dramatično znižuje stopnjevanje stroškov za večino nujnih sprememb.

Slika 3 prikazuje dve krivulji stopnjevanja stroškov za različne spremembe. Cilj agilnega razvoja je, da se kar največ sprememb pomakne iz zgornje v spodnjo krivuljo, kjer predstavljajo manjše finančno tveganje.

Slika 3: Stopnjevanje stroškov pri različnih spremembah



Vir: M. Poppendieck & T. Poppendieck, , Lean Software Development: An Agile Toolkit , 2003, str. 51.

Sočasen razvoj omogoča, da se z nepovratno odločitvijo počaka do **zadnjega odgovornega trenutka**, to je do trenutka, v katerem lahko izgubimo pomembno alternativo, če se odločimo. Strošek ne-odločitve v tem trenutku preseže strošek odločitve (The Last Responsible Moment, 2006).

1.3.2.4 Dobaviti, kakor hitro je mogoče

Hitra dobava je poslovna praksa, ki omogoča veliko konkurenčno prednost. Kupci imajo radi hitro dobavo, ker jim omogoča, da sami odlašajo z odločitvijo, za druge pa pomeni hitrejšo izpolnitev zadovoljstva. Za stranke, ki uporabljajo programsko opremo, hitra dobava pogosto pomeni fleksibilnost poslovanja. Hitra dobava pomeni za podjetja, da lahko dobavijo hitreje, kot si kupci lahko premislijo. Na tak način je manj virov vezanih na tekoče delo – lahko v obliki skladiščenega materiala ali v obliki delno dokončanega dela v razvoju (Poppendick & Poppendick, 2003, str. 66). Dobava v najkrajšem možnem času tudi dopolnjuje načelo čim kasnejšega odločanja: čim hitreje je podjetje sposobno dobaviti, več časa je za odlašanje z odločitvami. To v razvoju PO pomeni, da dopušča odprte možnosti in tako zmanjšuje negotovosti ter omogoča boljše odločitve (Steindl, 2004).

Hitra dobava se ne zgodi sama po sebi. V kolikor ljudje v delovnem procesu ne vedo natančno, kako kar najbolj učinkovito prispevati k poslu, nastajajo časovne izgube, trpi produktivnost in hitra dobava ni mogoča. Obstajata dva načina, kako zagotoviti, da delavci učinkovito izrabijo čas: ali jim povedati, kaj morajo narediti ali pa nastaviti proces tako, da delavci sami ugotovijo, kaj je potrebno narediti. V okoljih, kjer delo hitro poteka, deluje samo druga možnost, saj ni dovolj časa, da bi informacije potovale po verigi ukazov v obliki navodil za delo. Eden od ključnih načinov, kako organizirati delo, da ni odvisno od oddaljenega vira ukazov, je, da potrebe kupca delujejo kot načelo vlečenja. Za učinkovito samoorganiziranje dela je potrebno razviti metode za lokalno signaliziranje. Ena od funkcij načela vlečenja je vizualna kontrola - vsi morajo videti kaj se dogaja in kaj je potrebno narediti, kateri so problemi in kakšen je napredek (Steindl, 2004).

V razvoju programske opreme so lahko začetna točka načela vlečenja kratka ponavljanja, ki so osnovana na zahtevah kupca na začetku vsakega cikla ponavljanja. Želje kupcev se lahko na primer napišejo tudi na kartice. Programerji ocenijo potreben čas za izvedbo posamezne zahteve ali želje, kupci pa prioritizirajo kartice. Na koncu takega sestanka v zvezi z oblikovanjem zahtev projekta, postanejo kartice, na katerih so napisane zahteve, kanban kartice. Ker je osnovna ideja načela vlečenja, da delo teče samostojno, se kartice ne dodeljujejo programerjem, ampak si ti sami izberejo problem, na katerem bi radi delali. Za boljšo vizualizacijo se lahko kartice objavijo na tabli in se v skladu z napredovanjem dela premikajo s področja »potrebno narediti« v področje »testirati« in na koncu v področje »opravili teste«. Da bi programerji točno vedeli, kaj delati, kartice niso dovolj. Dobro je organizirati kratke dnevne sestanke, na katerih so aktivno udeleženi vsi člani razvojne skupine (Poppendick & Poppendick, 2003, str. 71).

Načelo vlečenja v razvoju PO zahteva kratke časovne okvirje, sicer lahko to načelo postane načelo potiskanja (angl. *push*). Način vlečenja pri razvoju PO deluje iz naročil kupcev, ki proritizirajo želene funkcije, in uporablja različne mehanizme signalizacije in predanosti delu, da se to organizira kot samovodeno.

1.3.2.5 Pooblaščanje tima

V 80-ih letih dvajsetega stoletja je postalo očitno, da s proizvodnimi tehnikami, ki jih je uvedla Toyota in ki so jih kasneje poimenovali vitka proizvodnja, lahko proizvedejo visoko kvalitetne izdelke hitreje in ceneje kot s tehnikami znanstvenega managementa, ki so se uveljavile v Ameriki. Znanstven management ali taylorizem, ime je dobil po njenem glavnem zagovorniku Fridericku Winslowu Taylorju, je teorija menedžmenta, ki analizira in sintetizira.

Delo v montaži avtomobilov je še vedno težko in ponavljajoče se. Tako v ameriški kot v japonski avtomobilski industriji je bilo strogo organizirano in kontrolirano. Razlika je bila v tem, da so inženirji v General Motorsu delavcem dajali natančna navodila, kako opraviti delo. V 1984 letu ustanovljenem obratu New United Motor Manufacturing (v nadaljevanju NUMMI), ki je bil združeno podjetje General Motorsa in Toyote, pa je bilo delo organizirano v skupinah od šest do osem ljudi, od katerih je bil eden vodja tima. Tim je sam izdelal svoje delovne postopke in koordiniral delovne standarde z ostalimi timi, ki so delali enako delo v drugih izmenah. Vloga managementa je bila trenirati, učiti in pomagati timu, medtem ko so bili inženirji na voljo, če jih je tim pozval. Vsak tim je bil odgovoren za svoje delovne procedure. Toyota v NUMMI ni prenesla japonskih proizvodnih praks v celoti, ampak je prenesla njeno prepričanje, da je osnova spoštovanja človeka, da mu zagotovi okolje, kjer lahko aktivno sodeluje pri oblikovanju in izboljšavah delovnih področij in lahko v celoti izkoristi svoje sposobnosti (Ohno, 1988 v Poppendick & Poppendick, 2003). Načelo vlečenja daje ljudem zaupanje in priložnost, da sprejemajo odločitve o svojem delu. Za dosego tega je treba ustvariti veliko znanja in omogočiti učenje vsem v ekipi z namenom, da postanejo najboljši v tem, kar počnejo. Ljudje, ki so ustrezno in pošteno plačani, so dodatno motivirani s samostojnostjo, pridobivanjem spretnosti in zaupanjem v smisel svojega dela (Pink, 2011).

Obstaja veliko dokazov, da je ljudem prirojeno skrbeti o namenu in tudi veliko dokazov, da ljudje trpijo, kadar nimajo namena (Thomas, 2000 v Poppendick & Poppendick, 2003, str. 71). Notranja motivacija prihaja iz dela, ki ga opravljamo, iz ponosa, da nekaj izdelujemo in iz občutka, da pomagamo kupcem. Notranja motivacija je posebej močna, če se člani tima zavežejo, da bodo dosegli svoj namen. Načini, da tim dobi in zadrži občutek za namen so (Poppendick & Poppendick, 2003, str. 71):

- **Jasen in prepričljiv namen**: Člani tima, ki se zavežejo k takemu cilju sodelujejo s strastjo, da bi pripeljali »svojega otroka« na tržišče.
- **Zagotoviti, da je namen dosegljiv:** Temeljno pravilo, da se skupino pooblasti za neko nalogo je, da mora biti tim sposoben doseči namen svojega dela, pri čemer pa mora imeti na razpolago tudi vse vire, ki jih za to potrebuje.

- **Dati timu možnost dostopa do kupcev:** Pogovor z uporabniki je lahko odličen način, da člani tima razumejo namen svojega dela. Članom tima daje vpogled v to, kako bo PO koristila uporabnikom in kako njihovo individualno delo koristi celotnemu projektu.
- **Dovoliti timu, da določi svoje obveznosti:** Na začetku se mora tim pogajati s kupci, da bi razumel njihove prioritete in izbrati delo za naslednji ponavljajoči se korak. Ko člani tima določijo funkcionalnosti izdelka, se obvežejo eden drugemu.
- **Vloga managementa:** Visoko motiviran tim ne potrebuje navodil za delo. Običajno potrebuje tim nekaj zaščite.
- **Odstranitev skeptikov iz skupine:** Nekdo, ki ve, da se nekaj ne da narediti, in najde za to razloge, lahko hitro uniči namen dela.

Gonilo notranje motivacije sta samoodločanje in občutek namena, vendar notranja motivacija ne bo delovala v sovražnem okolju. Raziskave so pokazale, da notranja motivacija potrebuje občutke pripadnosti, varnosti, sposobnosti in napredka (Thomas, 2000):

- **Pripadnost timu**: V zdravem okolju mora biti vsak seznanjen s cilji. Tem mora biti tudi zavezan. Tako zmage kot porazi so skupinski. Podeljevanje zaslug tima posameznikom in spodbujanje konkurence, ki ustvarja zmagovalce in poražence, ubija motivacijo.
- **Varnost :** *Miselnost nič napak* je atmosfera, ki ne tolerira nobene napake in znižuje motivacijo. Kadar posel narašča, je potrebno delegirati odgovornosti in spodbujati iniciativo, kar pa zahteva določeno toleranco, ker se bodo napake dogajale, ko bodo ljudje naredili nekaj po svoje, za kar pa ne smejo biti kaznovani.
- **Sposobnost:** Ljudje morajo verjeti, da so sposobni delati dobro, želijo sodelovati pri nečem, za kar verjamejo, da bo delovalo. Biti del zmagovitega tima visoko motivira ljudi. Občutek sposobnosti prihaja iz znanja in veščin, pozitivnih povratnih informacij, visokih standardov in s soočanjem s težkimi izzivi.
- **Napredek:** projekti morajo imeti smiselna merila, ki kažejo napredek k cilju. Ko tim doseže posebej pomemben cilj, je to potrebno obeležiti. Pomembni dosežki pa morajo biti tudi javno priznani.

Razvoj novega izdelka v Toyoti je zaupan glavnemu inženirju, ki je prvi v poznavanju izdelka. Po vsej verjetnosti je tudi napisal začetni koncept izdelka in je pridobil podporo managementa za program. Glavni inženir tudi prouči ciljni trg, dokumentira koncept vozila, vzpostavi celoten projekt in je na koncu odgovoren tudi za ekonomski uspeh vozila. Za razliko od tega, je vloga managerja v proizvodnji novih vozil v ameriških avtomobilskih tovarnah bolj koordiniranje projekta. Glavni inženir v Toyoti je bolj vodja, ki postavi smernice, uskladi organizacijo in motivira tim. V svoji študiji o procesih načrtovanja programske opreme za velike sisteme so Curtis et al. (1988) prav tako

ugotovili, da tudi v velikih sistemih glavno odgovornost za načrtovanje prevzamejo izjemni posamezniki ali manjša skupina izjemnih načrtovalcev, katerih vodenje je osnovano na njihovem znanju in ne na avtoriteti, ki jim je bila dodeljena. Ti strokovnjaki za razvoj programske opreme so lahko sistemski inženirji, glavni programerji ali arhitekti programske opreme. Mary in Tom Poppendick (2003) primerjata glavne programerje z vlogo glavnega inženirja v Toyoti. Glavni programerji so spoštovani zaradi poznavanja svojega strokovnega področja, razumejo potrebe kupcev in tima ter znajo svoje tehnične vizije tudi skomunicirati z razvojnim timom. Ker jih okolje zazna kot ljudi z največ znanja, postanejo tudi žariščna točka komuniciranja. So člani tima, seznanjeni s podrobnostmi dela. Glavni programerji običajno ne uspejo razviti svojih potencialov v okoljih, kjer dajejo prednost procesom, dokumentaciji in načrtovanju.

Pri agilnem razvoju projektni vodje velikokrat nimajo takšnega tehničnega znanja in ne rabijo razumeti vseh tehničnih aspektov projekta, ki ga vodijo. Projektni vodje v agilnem razvoju identificirajo izgube in rišejo zemljevid toka vrednosti za trenutni razvojni proces, rešujejo največja ozka grla. Projektni vodje koordinirajo sestanke in pomagajo timu, da dobi vse potrebne vire, ki jih potrebuje za delo, koordinirajo več timov hkrati ter skrbijo za redno in temeljito sinhronizacijo njihovega dela. Sodelujejo s finančno službo in kreirajo finančne modele, ki pomagajo timu do dobrih kompromisnih odločitev. Zagotavljajo tudi, da so ljudje, ki skrbijo za uvajanje, trening in podporo strankam od začetka vključeni v proces.

Razvoj programske opreme je kompleksna naloga, ki potrebuje specializirana znanja z več področij. Na eni strani so to tehnična znanja, kot so eksperti za podatkovne baze, uporabniške vmesnike in vgrajeno kodo (angl. *embedded code*) in premostitveni (angl. *middleware*) strokovnjaki. Na drugi strani je potrebno poznavanje področja, za katerega se programska oprema razvija. Tradicionalen način razvoja ekspertnih skupin v podjetju je, da razdelijo organizacijo v funkcijske enote, ki ustrezajo ključnim kompetencam, ki jih organizacija potrebuje. Četudi podjetje nima matrične organizacijske strukture, je nujno, da ima ekspertne skupine. Za oblikovanje teh pa je najprej potrebno identificirati tehnična znanja in znanja s področja, za katerega se razvija, ki so kritična za uspeh organizacije (Poppendick & Poppendick, 2003, str. 106).

1.3.2.6 Vgraditev integritete

V poznih 80-ih letih dvajsetega stoletja je Clark (1989) proučeval kritične razlike med povprečnimi podjetji in podjetji, ki so izdelovala odlične izdelke. To je poimenoval integriteta izdelka. Integriteta ima dve dimenziji – zaznavno in konceptualno. Zaznavna integriteta pomeni, da je izdelek v celoti ravnotežje med funkcionalnostjo, uporabnostjo, zanesljivostjo in gospodarnostjo, kar zadovoljuje kupca. Na zaznavno integriteto vpliva celotna uporabnikova izkušnja s sistemom, od tega, kako je oglaševan, dobavljen instaliran do enostavnosti njegove uporabe, cene in kako rešuje problem. Merilo za zaznavno

integriteto je grobo ekvivalentno s tržnim deležem. Podjetja, katerih izdelki dosegajo zaznavno integriteto, imajo načine, kako stalno ohranjati vrednote kupcev pred očmi ljudi, ki odločajo o natančnem načrtu izdelka. Imajo dober pretok informacij med kupci in razvijalci.
Konceptualna integriteta pomeni, da centralni koncept sistema deluje kot povezana celota. Komponente se dobro ujemajo, arhitektura programske opreme dosega učinkovito ravnovesje med fleksibilnostjo, učinkovitostjo, možnostjo vzdrževanja in odzivnostjo. Konceptualna integriteta je predpogoj za zaznavno integriteto. Če sistem ni konsistentno načrtovan, bo trpela uporabnost.

Ključne karakteristike sistema s konceptualno integriteto so (Poppendick & Poppendick, 2003, str. 116):

- **Preprostost:** Za skoraj vsako področje velja, da je preprost funkcionalen dizajn najboljši. Izkušeni razvijalci programske opreme razumejo, kako poenostaviti kompleksno kodo. Večina načrtovalskih vzorcev pri razvoju programske opreme je namenjena poenostavitvi kompleksnega sistema.
- **Jasnost:** Koda mora biti lahko razumljiva vsem, ki bodo morda delali z njo. Elementi programske opreme morajo biti poimenovani tako, da je brez dodatnih komentarjev razvidno, kaj so in kaj delajo.
- **Primernost za uporabo:** Da sistem, v katerem se programska oprema uporablja, dosega svoj namen.
- **Ni ponavljanja kode:** Kadar so potrebne spremembe programske opreme na več kot enem mestu, verjetnost napak raste eksponencialno, v kolikor se koda ponavlja.
- **Brez dodatnih funkcij**: Te morajo biti shranjene, integrirane in testirane vsakokrat, ko se programska koda spreminja.

1.3.2.7 Videnje celote

Sistem sestavljajo medsebojno odvisni in vzajemno delujoči deli, združeni z namenom. Sistem ni samo seštevek teh delov, ampak je produkt njihovih interakcij. Sposobnost sistema, da doseže optimalno delovanje ni v tem, kako dobri so posamezni deli, ampak kako dobro delujejo med seboj povezani (Poppendick & Poppendick, 2003, str. 137).

Sistemsko mišljenje gleda na organizacijo kot na sistem. Analizira se, kako so deli organizacije povezani in kako organizacija kot celota deluje skozi daljše časovno obdobje. K temu pripomore, da dobimo vpogled v vzorce in odnose, ter jih spreminjamo z namenom pridobitve trajne konkurenčne prednosti (Senge, 2014).

Eden od vzorcev sistemskega mišljenja se imenuje **omejitev rasti** (Senge, 2014). Četudi proces daje želene rezultate, kreira sekundarni učinek, ki uravnoteži in sčasoma upočasni uspeh. Če si še nadalje prizadevamo povečati uspeh z istim procesom, se sekundarni

učinek povečuje. Namesto prizadevati si za rast, je potrebno poiskati in odstraniti ovire za rast.

Prenos bremena je drugi vzorec sistemskega mišljenja. V tem vzorcu osnovni problem povzroča simptome, ki jih ne moremo ignorirati. S temeljnimi problemi se je težko soočiti, zato se ljudje ukvarjajo s simptomi, namesto z vzroki problemov. Hitri popravki lahko prikrijejo simptome, na žalost pa lahko na ta način osnovni problem še povečamo.

Tretji vzorec, ki se pojavlja v organizaciji, je **optimizacija oddelkov**. Bolj je sistem kompleksen, večja je želja razdeliti ga na dele in obvladovati dele organizacije ločeno. Lokalno obvladovanje pogosto ustvarja učinke, ki znižujejo skupno storilnost organizacije, vendar je vpliv lokalne optimizacije na skupen rezultat pogosto skrit. Eden večjih problemov pri meritvah uspešnosti je, da se izvajajo na lokalnem nivoju, vendar je maksimiziranje lokalnih meritev dostikrat v nasprotju z optimiziranjem organizacije kot celote.

Za sledenje napredku razvoja programske opreme so meritve potrebne, vendar je v te namene potrebno uporabiti merjenje informacij namesto merjenja uspešnosti.

1.3.3 Indikatorji učinkovitosti vitkega razvoja programske opreme

Ker je vitek razvoj programske opreme precej nov proces, so se raziskave o prednostih, ki jih prinaša, večinoma začele po letu 2010 (Feyh & Petersen, 2013). Poppendick M. in Poppendick T. (2003) predlagata, da se je potrebno pri meritvah učinkovitosti osredotočiti na celoten proces. Meritve ponavadi potekajo lokalno, po posameznih oddelkih, kar je lahko kontraproduktivno. Osredotolčanje na meritve enega procesa in njegovo optimiziranje ima lahko slab učinek na organizacijo kot celoto, saj ima tendenco, da se zavira sodelovanje z oddelki, ki so za področjem, na katerega se osredotočamo (Poppendick & Popendick, 2003, str. 137).

Feyh in Peterson (2013) sta prišla do ugotovitve, da je od leta 2010 naprej je opaziti opazen porast literature o vitkem razvoju PO, ki je pomembna za raziskave in prakso in da se večina študij in literature nanaša na izkušnje strokovnjakov. Največ študij je bilo izdelanih s pomočjo uporabe znanstvenih metod v industriji, dve sta bili izdelani v laboratoriju in predlagata rešitev ali predstavljata mnenje (Feyh & Petersen, 2013).

Vitek razvoj programske opreme stremi k izboljšavam, vendar te izboljšave potrebujejo merila, s pomočjo katerih bi se dalo izmeriti razlike, ki jih sam vitek razvoj prinaša. Merila in indikatorji, ki so jih raziskovalci uporabili za ocenjevanje učinkovitosti vitkega razvoja PO, sta Feyh in Peterson (2013) v osnovi razdelila na osnovna merila, ki se vežejo na proizvod in proces hkrati, na sam proizvod ali na udeležence v procesu. Med dvanajstimi osnovnimi merili, ki obravnavajo proizvod in proces, so najpogostejši ukrepi vezani na

merjenje časa. Največkrat gre za čas, potreben za dokončanje dela, lahko pa se upoštevata tudi čas, v katerem se dejansko dodaja vrednost izdelku in čas, v katerem ni dodane vrednosti (zamude, čas prenosov, organiziranja in čakanja v vrsti). Sedem osnovnih meril je vezanih samo na izdelek in od teh se največkrat uporablja število odkritih in nerešenih napak. Glede udeležencev v procesu, pa so v člankih največkrat omenjeni trije osnovni ukrepi in sicer: zaznano zadovoljstvo kupcev, zadovoljstvo zaposlenih in usposobljenost zaposlenih. Z uporabo funkcij, ki povezujejo dve ali več osnovnih meril so izpeljana povezana merila, kot so na primer pretok dela, čas cikla, kapaciteta (Feyh & Petersen, 2013).

Indikatorji prikazujejo merila napredka, osnovani pa so na ukrepih, ki so merljivi. V študijah, sta avtorja identificirala štirinajst indikatorjev, največkrat uporabljena kazalnika pa sta (Feyh & Petersen, 2013):

- Diagram kumulativnega toka, prikazuje skupno količino zaključenega dela v določenem času za različne faze in omogoča detektiranje zastojev
- Regresija, ki analizira nabor podatkov z regresijskimi krivuljami. Prikazuje meritve v določenem časovnem obdobju, pri čemer uporablja pretekle podatke in kaže trende (npr. dobavni rok, čas potreben za dokončanje dela)

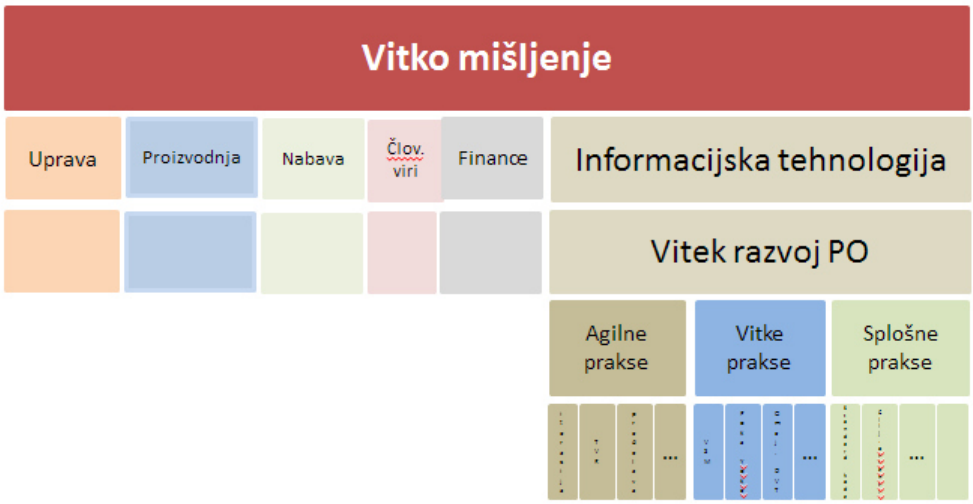
Merjenja so pomembna za sledenje napredka pri razvoju PO. K temu spada tudi štetje napak, ki vplivajo na PO in njeno primernost za objavo. Pri tem Poppendick M. in Poppendick T. poudarjata, da sledenje napakam do posameznikov velikokrat prikrije dejstvo, da je izvor napak v prevladujočem sistemu in procesih, ki jih kontrolirajo managerji (Poppendick & Popendick, 2003, str. 144).

1.3.4 Pozicioniranje vitkega razvoja programske opreme

Cowley et al. (2013) predlagajo, da se na vitek razvoj gleda kot na združitev vitkih konceptov z modernimi praksami razvoja programske opreme, kot na agilne metode in druge splošno sprejete dobre prakse razvoja.

Vitek razvoj programske opreme so pozicionirali med vitko filozofijo, ki je lahko sprejeta na organizacijskem nivoju in med agilnima praksama, ki so tipično osredotočene na čisto praktičen nivo. Agilne metode vidijo kot podporne prakse vitkega razvoja programske opreme, kar je prikazano na Sliki 4. Vitek razvoj programske opreme je pozicioniran pod okrilje vitkega mišljenja, ki zajema celotno organizacijo. To je pomembno za razumevanje, saj je vzpostavitev delovanja celotnega neprekinjenega toka vrednosti v smeri doseganja skupnega cilja pravi namen **vitkega podjetja** (Womack & Jones, 2010).

Slika 4: Pozicioniranje vitkih in agilnih praks razvoja PO v organizaciji



Vir: O. Cawley, X. Wangl & I. Richardson, Lean Enterprise Software and Systems, 2013, str. 27.

2 VITKE STRATEGIJE UPORABLJENE PRI ZAGONSKIH PODJETJIH

2.1 Opredelitev zagonskega podjetja

V zadnjih letih se pogosto govori in piše o »start-up«-ih oziroma zagonskih podjetjih, poslovnih modelih, vitkem poslovanju, vitkih poslovnih modelih itd..Vendar je pravi izziv definirati, kaj ti pojmi točno pomenijo. Ker bom v nadaljevanju opisovala poslovne modele v zagonskem podjetju, sem najprej želela definirati pojma **zagonsko podjetje** in **poslovni model**.

Izraz »zagonsko podjetje« je v slovenščini precej nov in še ne dokončno sprejet. V slovenski strokovni literaturi se dostikrat uporabljata tudi angleška izraza »start-up« ali »startup«, izraz »štartnik« pa se je kot predlog pojavil s slovenskem prevodu knjige Delaj vitko, Asha Muraya (2012) in prav tako še ni osvojen.

Zagonska podjetja so nova, inovacijsko gnana podjetja, ki razvijajo nove izdelke in storitve s potencialom za trženje in rast na globalnih trgih (Rus, 2015). Enotne definicije, kaj je zagonsko podjetje ni, vendar je jedro večine definicij podobno – gre za hitro rastoča visokotehnološka podjetja, ki delujejo v negotovih okoljih. Po mnenju Paula Grahama (2012) je zagonsko podjetje zasnovano za rast, ki je bistvena za zagonsko podjetje. Dejstvo, da je podjetje na novo ustanovljeno, še ne pomeni, da gre za zagonsko podjetje. Tudi ni nujno, da je tehnološko ali da je financirano iz skladov tveganega kapitala.

Damodoranova definicija, ki pravi, da vrednost zagonskega podjetja sloni na njegovi rasti v prihodnosti, pa bolj kot panogo ali strukturo podjetja poudarja stopnjo razvoja podjetja (Damodaran, 2012). Blank in Dorf (2012) definirata zagonsko podjetje kot organizacijo, ki je ustanovljena, da išče ponovljiv in nadgradljiv poslovni model. Eric Ries (2011) pa je v svojo definicijo zagonskega podjetja vključil okolje, v katerem podjetja delujejo. Po njegovi definiciji so zagonska podjetja institucije, ki so ustanovljene, da kreirajo nove izdelke ali servise v skrajno negotovih pogojih. Sutton (2000) razširja pojmovanje zagonskega podjetja na pojme inovacija, hitra rast, časovni pritisk, odvisnost od tretje osebe, osredotočanje na izdelek in mlada organizacijska struktura.

Pomembno je tudi, da besedo »inovacije« razumemo v širšem pomenu. Zagonska podjetja uporabljajo veliko vrst inovacij: nova znanstvena odkritja, spremembe namena obstoječih tehnologij za nov način uporabe, oblikovanje novih poslovnih modelov, ki odkrivajo vrednosti, ki so bile skrite ali preprosto pripeljati izdelek ali storitev na novo lokacijo ali k pred tem spregledanim kupcem (Ries, 2011, str. 28).

2.1.1 Značilnosti zagonskih podjetij

Glede na to, od kod izhajajo ustanovitelji zagonskih podjetij, se zagonska podjetja lahko delijo na akademska in neakademska oziroma komercialna zagonska podjetja.

Akademska zagonska podjetja, imenovana tudi *spin-out* ali *spin-off* podjetja, so začela nastajati od leta 1990 naprej, ker so bile univerze podvržene političnim pritiskom, da morajo uveljaviti, meriti in izboljšati njihov vpliv na nacionalno blaginjo, predvsem s poudarkom na gospodarski rasti, ustvarjanju delovnih mest in konkurenčnosti. Zato so se univerze morale osredotočiti na tehnološke transferje, ki bi raziskave vpeljal v produktivne organizacije (Valdivia, 2013). Namen ustanovitve akademskih zagonskih podjetij je komercialno izkoriščanje tehnološkega znanja, generiranega v javnih raziskovalnih ustanovah. To je tudi vedno bolj popularna možnost ustvarjanja bogastva s pomočjo rezultatov akademskih raziskav (Siegel, Waldman, Atwater & Link, 2003; Vohora, Wright & Lockett, 2004). Med univerzami in drugimi javnimi raziskovalnimi ustanovami je nastalo intenzivno tekmovanje, da bi povečali število visoko tehnoloških podjetij, ki izhajajo iz teh ustanov. Ta zagonska podjetja so nastala s prenosom ljudi in intelektualne lastnine iz matičnih institucij (Davenport, Carr & Bibby, 2002). Obstaja kar nekaj analiz o pogojih in strategijah pri ustanavljanju teh zagonskih podjetij (Chrisman, Hynes & Fraser, 1995; Davenport et al., 2002; Di Gregorio and Shane, 2003).

Chapple, Lockett, Siegel & Wright (2005) ugotavljajo, da sta učinkovitost in finančni donos tehnoloških transferjev pod pričakovanji, medtem ko Degroof in Roberts (2004) poudarjata, da imajo akademska zagonska podjetja tendenco ostati majhna.

Komercialna zagonska podjetja dostikrat ustanovijo posamezniki ali manjša skupina strokovnjakov z izkušnjami iz branže, ki razumejo tehnologijo in lahko z njeno pomočjo rešijo problem. Rešitev problema je nov izdelek ali storite, za katerega ustanovitelji zagonskih podjetij mislijo, da ima tržni potencial. Osnova za ustanovitev podjetja je poslovna ideja in pripravljenost tveganja ustanoviteljev, da jo uresničijo. Taka podjetja so običajno **samozagonska**, financirajo jih ustanovitelji iz lastnih sredstev. Ker so njihova sredstva omejena, temu velikokrat sledi prodaja kapitala skladom tveganega kapitala v izmenjavo za zagonska sredstva potrebna za razvoj poslovne ideje (Horton, 2016). Večkrat pa so pričakovanja investitorjev, ki priskrbijo semenski kapital za razvoj podjetja in ustanoviteljev podjetja glede zaslužka glede na vložek različni. Zunanje investitorje ponavadi bolj zanima poslovni del, kot karakteristike samega izdelka (Gatewood at al., 1995). Zagonska podjetja pa lahko pridobijo vire financiranja tudi s pomočjo poslovnih angelov ali državnih subvencij.

Colombo in Piva (2009) ugotavljata, da imajo ustanovitelji akademskih zagonskih podjetij višjo tehnično in znanstveno izobrazbo in manj tehnično – specifičnih delovnih izkušenj kot ustanovitelji ne-akademskih zagonskih podjetij. Akademiki nimajo vodstvenih izkušenj, ker je redko, da bi bili pred ustanovitvijo podjetja na kaki managerski funkciji ali da bi bili pred tem samozaposleni. Po drugi strani imajo akademska zagonska podjetja prednost pri zaposlovanju visoko kvalificiranega osebja predvsem tehničnih strok, medtem ko so manj nagnjeni k zaposlovanju ljudi na poslovnih funkcijah. Prednost akademskih zagonskih podjetij je tudi zunanje sodelovanje, saj imajo dostop do razvejane tehnološke mreže (Colombo & Pliva, 2009).

Crowne (2002) je življenski cikel zagonskih podjetij razdelil na štiri stopnje:

- Startni cikel traja od kreiranja in natančnega izoblikovanja ideje oziroma koncepta do prve prodaje.
- Faza stabilizacije traja od prve prodaje ter vse dokler izdelek ni dovolj stabilen in ga naroči novi kupec.
- Faza rasti se začne s procesom razvoja stabilnega izdelka in traja dokler niso ustvarjeni velikost trga, tržni delež in rast.
- Zagonsko podjetje se razvije v zrelo organizacijo, ko je razvoj izdelka robusten in predvidljiv in je razvoj novega izdelka definiran proces.

Medtem ko zagonska podjetja znatno prispevajo k rasti večine ekonomij, ostaja stopnja prenehanje delovanja podjetij še vedno visoka – v Silikonski dolini tudi do 90%.

Pena (2002) pa je ugotovil, da so nematerialna sredstva, ki so pozitivno povezana z uspehom podjetja, naslednja:

- človeški kapital (izobrazba, poslovne izkušnje, motivacija),

- organizacijski kapital (zmožnost podjetja hitro prilagajati se spremembam, uvedba uspešnih strategij),
- relacijski kapital (razvoj produktivne poslovne mreže in takojšen dostop do kritičnega števila zainteresiranih subjektov – npr. dobaviteljev, strank, finančnih trgov).

Brez dvoma je akumuliranje virov kritičnega pomena za uspeh zagonskih podjetij, vendar je na hitro razvijajočih se trgih, na katere vstopajo visoko tehnološko razvita podjetja, napovedovati uspeh podjetja samo na podlagi dragocenosti virov tvegano (Eisenhart & Martin, 2000). Številni (Eisenhardt & Martin, 2000; Zollo & Winter 2002; Makadok, 2001; Zott, 2003) so raziskovali **dinamično zmogljivost** podjetij, ki so jo prvi opisali Teece, Pisano in Shuen (1997). Dinamična zmogljivost je sposobnost integracije, graditve in preoblikovanja notranjih in zunanjih virov. Je nujno sredstvo za prilagajanje hitrim spremembam v poslovnih okoljih in bistveno vpliva na uspešnost podjetja (Teece et al., 1997).

Alice Trung (2015) v članku, ki analizira prenehanje delovanja 200-tih podjetij, oziroma mnenja ustanoviteljev teh podjetij, med desetimi najpogostejšimi razlogi za prenehanje navaja, da je poslovni model neizvedljiv, da ideja ni bila dovolj dobro sprejeta, da gre za pomanjkanje denarja, za pomanjkanje sredstev / investicij, da so vzrok tehnične težave, da ni potrebe na trgu, da je konkurenca premočna, da so težave z razvojem kupcev ter da so v timu nesoglasja in pomanjkanje fokusa.

Pri vzrokih za prenehanje poslovanja pa obstaja razlika med zagonskimi podjetji, ki so pridobila zagonski kapital s pomočjo vlagateljev in med samozagonskimi podjetji: pri podjetjih financiranih s tujim kapitalom, je pomanjkanje denarja na prvem mestu od vzrokov za prenehanje delovanja, medtem ko je pri samozagonskih podjetjih ta vzrok šele na desetem mestu, na prvem mestu pa je neizvedljivost poslovnega modela. Blank (2007) vidi razliko med uspešnimi in manj uspešnimi zagonskimi podjetji predvsem v osredotočanju na kupca. Veliko visokotehnoloških podjetij je pri razvoju preveč osredotočenih na izdelek in premalo na potencialne kupce, kar lahko vodi v napačno smer razvoja izdelka (Blank, 2007).

2.1.2 Pomen zagonskih podjetij v gospodarstvu

Podjetništvo velja za pomemben mehanizem ekonomskega razvoja, ker prispeva k zaposlovanju, inovacijam in blaginjo. Visoke stopnje razvoja podjetništva ni mogoče direktno prenesti na visoko stopnjo inovacij, saj le-ta zajema vse oblike samozaposlitve, ki so večkrat posledica birokratskih preprek ali pa ekonomija na ta način kreira slabše plačana delovna mesta. Porter (1990) razlikuje tri faze konkurenčnosti gospodarstev glede na ekonomski razvoj:

- Faktorska gospodarstva: države konkurirajo z nizko ceno v proizvodnji nizkocenovnega blaga ali proizvodov z nizko dodano vrednostjo.
- Učinkovita gospodarstva: države morajo povečati njihovo proizvodno učinkovitost in izobraziti delavce, da se lahko prilagodijo fazi tehnološkega razvoja.
- Inovacijska gospodarstva: za ta je značilen porast podjetniške aktivnosti. V zadnjih letih so ekonomisti prepoznali pomemben prispevek inovacij in rasti na ekonomski razvoj in blaginjo (Acs & Armington, 2006; Schramm, 2006; Audretsch, 2007).

Poleg razlik v načinu konkuriranja v naštetih fazah, obstaja tudi razlika v stopnji integracije držav v svetovno ekonomijo. Še posebej zato, ker inovacije prispevajo h konkurenčni prednosti na tujih trgih (Roper & Love, 2002; Sterlacchini 1999), so razvite ekonomije bolje globalno integrirane in imajo višjo stopnjo izvozno naravnanih podjetij (De Clercq et al., 2008).

Tehnološke spremembe in razvoj informacijskih tehnologij so pomen zagonskih podjetij, ki igrajo pomembno vlogo pri inoviranju, postavili v ospredje. Nove tehnologije omogočajo podjetjem, da razvijajo inovativne izdelke ali storitve, inovativne poslovne modele, procese in načine organizacije dela, ki na učinkovitejši način zadovoljujejo obstoječe potrebe uporabnikov in obenem soustvarijo tudi povsem nove potrebe. Ključno gonilo gospodarskega napredka torej predstavljajo inovacije, ki jim lahko po podatkih OECD (Calvino et al., 2015) pripišemo več kot polovico rasti produktivnosti v razvitem svetu (povzeto po Slovenski podjetniški observatorij 2015).

Startup manifest navaja pet razlogov za spodbujanje zagonskih podjetij **(**Beck et al., 2011).

- **Inovacije –** So glavno gonilo razvoja gospodarstva v družbi znanja. Zagonska podjetja prispevajo k hitremu razvoju novih tehnologij in nekega geografskega področja. Velika podjetja pogosto kupujejo zagonska podjetja kot tehnologijo, ki jo integrirajo in s tem ohranijo konkurenčno prednost.
- **Nova delovna mesta in gospodarska rast** – Zagonska podjetja dolgoročno ustvarjajo veliko delovnih mest in prispevajo h gospodarski rasti. Ker temeljijo na inovacijah so zdravo jedro gospodarstva, dolgoročno ustvarjajo največ delovnih mest.
- **Vnašanje nove konkurenčne dinamike v gospodarski sistem** – Zagonska podjetja so najbolj dinamične gospodarske organizacije na trgu, saj v gospodarski sistem vnašajo dodatno dinamiko in konkurenčnost.
- **Promocija inovacijskega sistema** – Visokotehnološka in na znanju temelječa storitvena zagonska podjetja so zelo povezana z institucijami znanja.
- **Vnašanje vrednot proaktivnosti v družbo** – Zagonsko podjetništvo spreminja vrednote družbe in prinaša nov miselni vzorec, skladen z družbo znanja.

Podatki raziskave Global Enterpreneuership Monitor (GEM) kažejo, da v svetu podjetniki z visokim potencialom rasti v povprečju ustvarijo kar 3-krat več delovnih mest kot podjetniki s srednjim potencialom in kar 15-krat več delovnih mest kot podjetniki z nizkim potencialom rasti. K rasti prispevajo samo zelo ambiciozni podjetniki, ki najdejo in uresničijo poslovne priložnosti. Med podjetniki pa je samo 4% takih, ki jih lahko štejemo v skupino dinamičnih podjetij s potencialom rasti na globalnih trgih (Beck, Demirgüç-Kunt, A., & Maksimovic, 2004).

2.1.3 Zagonska podjetja, ki se ukvarjajo z razvojem programske opreme

Definicija zagonskih podjetij, ki se ukvarjajo z razvojem programske opreme je bolj problematična in ni nujno vezana na velikost podjetja. Na primer Mirel (2000) imenuje zagonska tista podjetja, ki imajo šest zaposlenih, medtem ko Coleman in O'Connor (2008) imenujeta zagonska tudi podjetja z več kot 300 zaposlenimi. Tudi starost zagonskih podjetij, ki razvijajo programsko opremo, pri definiciji ni enotna. Nekateri avtorji proučujejo zagonska podjetja, ki delujejo že nekaj let, medtem ko se drugi striktno omejujejo na pred kratkim ustanovljena podjetja (Kajko-Mattison et al., 2008). Drugi avtorji obravnavajo »zagonsko« kot stanje podjetja (Crowne, 2002 in Tanabian, 2005). Nekateri zagovarjajo, da zagonska podjetja delajo na inovativnih izdelkih, pri čemer inovacije ne definirajo natančno.

Paternoster, Giardino, Unterkalmsteiner, Gorschek & Abrahamsson (2014) pravijo, da so zagonska podjetja, ki razvijajo programsko opremo, na novo ustanovljena podjetja brez zgodovine delovanja, ki hitro proizvajajo najnovejše tehnologije. Ta podjetja izdelujejo programsko opremo v skrajno negotovih pogojih, spoprijemajo se s hitro rastočimi trgi in s pomanjkanjem virov. Ugotavljajo, da raziskav o tem, kako se razvoja programske opreme lotevajo v novonastalih podjetij, ni veliko. Kitchman in Charters (2007) nadalje ugotavljata, da se nobeden od sistematičnih pregledov literature o inženiringu programske opreme, ne obravnava fenomena zagonskih podjetij. V literaturi o inženiringu programske opreme je izraz zagonsko podjetje prvič uporabil Carmel (1995), ki je ugotavljal čas do zaključitve paketov programske opreme v mladih podjetjih. Opazil je, da so bila ta podjetja posebno inovativna in uspešna in je zagovarjal potrebo po raziskovanju njihovih praks v razvoju programske opreme z namenom, da bi proces ponovili in ga skušali prenesti v druge tehnološke sektorje.

Razumevanje prednosti zagonskih podjetij, ki izhajajo iz njihovih delovnih praks je nujno, saj so zagonska podjetja generator tako novih inovativnih izdelkov kot delovnih mest. Značilen cilj zagonskih podjetij je kreiranje visoko-tehnoloških in inovativnih izdelkov in rast z agresivnim širjenjem posla na hitro rastočih trgih. Podjetja, ki se ukvarjajo z razvojem programske opreme, so soočena z velikim časovnim pritiskom trga in so hkrati izpostavljena hudi konkurenci. Delujejo v kaotičnih in negotovih okoliščinah na hitro rastočih trgih (Maccormack, 2001). S stališča procesa razvoja programske opreme so

zagonska podjetja velikokrat ukvarjajo bolj s preživetjem kot z vzpostavitvijo postopkov. Bach (1998) opisuje tipično zagonsko podjetje kot množico energičnih in cilju zavezanih ljudi brez definiranih procesov v razvoju (Bach, 1998). Sutton (2000) pa po drugi strani trdi, da so zagonska podjetja, ki razvijajo programsko opremo pri študijah razvojnih procesov večinoma prezrta.

Sutton (2000) je značilnosti zagonskih podjetij, ki se ukvarjajo z razvojem programske opreme, opredelil tudi glede na izzive s katerimi se srečujejo.

- **Mladost in nezrelost:** Podjetja so tako v svojih procesnih zmožnostih kot v svoji organizaciji neizkušena v primerjavi z bolj uveljavljenimi in zrelimi razvojnimi organizacijami.
- **Omejenost virov:** Viri, v katere najprej investira podjetje, so usmerjeni navzven. Najprej v izdelavo izdelka, potem pa v njegovo promoviranje in izgradnjo strateških povezav.
- **Različni vplivi:** V zgodnji fazi je podjetje občutljivo na vplive investitorjev, kupcev, partnerjev in konkurence, tako obstoječih kot potencialnih. Podjetje se zato nenehno prilagaja in spreminja odločitve o tem, kaj delati in kako delati.
- **Dinamične tehnologije in trgi:** Nova podjetja za razvoj programske opreme so pogosto ustanovljena za razvoj tehnološko inovativnih izdelkov, kar zahteva najsodobnejša razvojna orodja in tehnike.

Drugi raziskovalci omenjajo majhno ustanovno ekipo, ki jo večkrat sestavljajo manj izkušeni ljudje. Podjetje ima plosko organizacijsko strukturo, direktor je včasih sam glavni razvijalec. Druge študije se strinjajo z visoko tvegano naravo zagonskih podjetij in s tem, da so na novo ustanovljena ter zato brez delovnih izkušenj.

Eden večjih izzivov za inženirje v zagonskih podjetjih je, da uvedejo metode za izgradnjo in kontrolo razvojnih aktivnosti (Coleman & O'Connor, 2008). Na splošno so zagonska podjetja kreativna in fleksibilna in nerada vpeljujejo procese in birokratska merila, ki bi lahko ovirala njihova naravne sposobnosti (Sutton, 2000; Bach, 1998). Nadalje imajo zagonska podjetja zelo omejene vire in jih želijo tipično uporabiti za podporo razvoju izdelka, namesto, da bi vzpostavili proces (Coleman & O'Connor, 2008). Z zavračanjem pojma ponovljivega in kontroliranega procesa, zagonska podjetja očitno izkoriščajo nepredvidljive, reaktivne in nizko-precizne inženirske prakse (Sutton, 2000). K izdelku orientirane prakse pomagajo zagonskim podjetjem, da imajo fleksibilen tim s potekom dela, ki jim dopušča možnost, da hitro zamenjajo smer glede na ciljni trg (Sutton, 2000; Heitlager, Helms & Brinkkemper, 2007).

Zagonska podjetja pogosto raje razvijajo aplikacije, s katerimi bi ponudila rešitev ciljnemu trgu z visokim potencialom, kot da razvijajo programsko opremo za specifične kliente (Marmer, Herrmann, Dogrultan & Berman, 2011; Blank, 2013). Vprašanja vezana na ta

ciljni trg se v literaturi nanašajo na tržno usmerjen razvoj programske opreme (Regnell, Höst, och Dag, Beremark & Hjelm, 2001). Čas, potreben za vstop na trg, je ključen strateški cilj. Zagonska podjetja običajno delujejo v hitro spreminjajočih se in negotovih trgih in se spopadajo s pomanjkanjem virov. Pri razvoju programske opreme za rastoče ciljne trge, stranke in končni uporabniki povečini niso dobro poznani. Zahteve bolj narekuje trg in niso specifične za kupca. Na neraziskanih in inovativnih trgih, so zahteve slabo definirane in se hitro spreminjajo. Razvojni oddelki tako težko vzdržujejo zahteve in jih dosledno ohranjajo (Blank, 2013).

Več avtorjev (Poppendick & Poppendick, 2003; Blank, 2007; Ries 2011) priznava pomembnost vključevanja kupcev oziroma končnih uporabnikov v proces izbire in prioritizacije zahtev glede na njihove primarne zahteve. Paternoster et al. (2014) so iz triinštiridesetih primarnih študij identificirali različne delovne prakse, ki jih uporabljajo zagonska podjetja, ki razvijajo programsko opremo. Te delovne prakse so razdelili v štiri podpodročja.

- **Procesni management:** Prakse v procesnem managementu predstavljajo vse inženirske aktivnosti, ki se uporabljajo za razvoj izdelka v zagonskih podjetjih (Sutton, 2000). V zagonskih podjetjih so uporabne lahke metode (agilne metode, vitko zagonsko podjetje), ki omogočajo fleksibilnost pri izbiri prilagojenih praks in možnost reagiranja pri spreminjanju izdelka. Značilne so tudi hitre izdaje, ki omogočajo pripravo prototipa in učenje iz povratnih informacij uporabnikov.
- **Prakse pri razvoju programske opreme:** V zagonskih podjetjih je opaziti splošno pomanjkanje napisane arhitekture programov, kar predstavlja težavo, kadar se povečata baza uporabnikov in kompleksnost samega izdelka. Uporaba arhitekture in okvirjev programske opreme, ki omogočajo enostavno širitev in dodajanje ali odvzemanje funkcij je ključnega pomena.
- **Organizacijske prakse:** Ohlapna organizacijska struktura in izostanek tradicionalnega projektnega vodenja sta posledica časovnega pritiska in pomanjkanja virov v zagonskih podjetjih. Krepitev vloge članov tima, njihove odgovornosti in sposobnosti, da vplivajo na končni rezultat, je uspešna strategija zagonskih podjetij, ki povečuje možnost uspeha (Carmel, 1994; Steenhuis, 2008).
- **Orodja in tehnologije:** Prednost zagonskih podjetij je možnost uporabe najnovejših tehnologij, ne da bi jih pri tem omejevala že prej uporabljena strojna in programska oprema in delovne izkušnje. Vendar je pri tem pomanjkanje delovnih izkušenj tudi slabost zagonskih podjetij.

Zagonska podjetja so sposobna razvoja PO, ki lahko močno vplivajo na trg in bistveno prispevajo h globalni ekonomiji. Po mnenja Paternostra et al. (2014) pa sam proces razvoja PO, ki je jedro zagonskih podjetij, še ni dovolj znanstveno raziskan.

2.2 Poslovni model

2.2.1 Definicija poslovnega modela

Zaradi razmaha informacijske tehnologije v poznih 90-ih letih dvajsetega stoletja je zanimanje znanstvenikov za poslovni model izredno naraslo. Z namenom, da bi s spremembo obstoječih poslovnih modelov izboljšali poslovanje, so se na inovacije poslovnega modela osredotočali tudi vodilni delavci (Santos, Spector & Van der Heyden, 2009). Koncept poslovnega modela in posledično tudi koncept inovacij poslovnega modela temeljita na korporativnih praksah, strateškem managementu in ekonomiki podjetij (Amit & Zott, 2001; Aspara, Hietanen & Tikkanen, 2010; Carayannis, Sindakis & Walter, 2015).

Čeprav se je izraz **poslovni model** (angl. *business model*) pojavil v Langovem povzetku članka iz leta 1947, kasneje pa še v seznamu izrazov leta 1949 in v naslovu članka G.M Jonesa iz leta 1960, nobeden od avtorjev ni definiral pomena izraza (Santos et. al., 2009). Veliko število avtorjev se strinja, da »poslovni model« , čeprav zelo pogosto uporabljen v besednjaku managerjev, tudi danes nima enotne definicije (Schaffer et al., 2005; Morris, Schindehutte & Allen, 2005; Chesbrough in Rosenbloom, 2002). Schaffer navaja, da so pri pregledu znanstvenih publikacij, izdanih med leti 1998 in 2002, našli dvanajst definicij poslovnega modela, od katerih nobena ni bila v celoti sprejeta v poslovni skupnosti. Razlog za to vidijo v tem, da avtorji izhajajo iz drugačnih perspektiv (e-podjetništvo, strategija, tehnologija in informacijski sistemi) in imajo drugačna gledišča na poslovni model (Schaffer et al., 2005).

Še pred konceptom poslovnega modela, je prvi sistematično in primerjalno študijo rasti in sprememb v moderni industrijski korporaciji opisal Alfred Chandler (1962) v knjigi »Strategy and Structure« (Chandler, 1962). Čeprav sta pojma »poslovni model« in »poslovna strategija« tesno povezana, ju vendar ne smemo zamenjevati. Santos et al. (2009) navajajo, da poslovno strategijo določajo odgovori na tri vprašanja: **kaj** ponujamo, **kdo** so kupci in **kako** je ponudba proizvedena in dobavljena kupcem? V vprašanju »kako« je zajeta odločitev podjetja za poslovni model. Podjetja lahko imajo v bistvu enak izdelek ali storitev (vprašanje »kaj«), ciljajo na isti segment trga (vprašanje »kdo«) in to naredijo z različnimi poslovnimi modeli (»kako«).

Poslovni model je sestavljen iz štirih prekrivajočih se elementov, ki skupaj kreirajo in prinašajo vrednost (Jones, 2010):

- **Predlagana vrednost za kupca (angl. *customer value proposition*):** Podjetje najde način, da kreira vrednost za kupca, to je način, da pomaga kupcem opraviti pomembno delo. »Delo« je v tem primeru temeljni problem, ki v dani situaciji potrebuje rešitev.

- **Formula dobička (angl. *profit formula*):** Je načrt, kako podjetje kreira vrednost zase, ko zagotavlja vrednost za kupca. Sem spadajo model prihodkov, struktura stroškov, določanje marže, hitrost obračanja virov (zalog, fiksnih in obratnih sredstev).
- **Ključni viri:** To so ljudje, tehnologija, izdelki, objekti, oprema, kanali in znamka, ki so nujni za zagotavljanje vrednosti kupcem.
- **Ključni procesi:** Uspešna podjetja imajo operativne postopke in managerske procese, ki jim omogočajo zagotavljati vrednost na način, ki ga lahko uspešno ponovijo in nadgrajujejo. To lahko vključuje dela, ki se ponavljajo kot: usposabljanje, razvoj, proizvodnja, sprejemanje proračuna, planiranje, prodaja in servis. Ključni procesi prav tako vključujejo pravila, merila in norme, ki jih postavi podjetje.

V najbolj osnovnem smislu je poslovni model metoda poslovanja, s katero lahko podjetje ohranja svojo dejavnost, to je ustvarjanje prihodka (Chesbrough, 2002). Poslovni model je lahko tudi centralna ideja raziskav v podjetništvu (Moorris et al., 2005).

2.2.2 Inovacije poslovnih modelov

Tako kot za poslovni model, tudi za inovacije poslovnih modelov velja, da nimajo enotne definicije. Inovacija poslovnih modelov je koncept, ki je osnovan na načelu, da podjetja inovirajo, s tem da povečujejo svoje notranje zmogljivosti in zmogljivost virov (Amitt & Zott, 2001). Chesbrough (2010) je ugotovil, da tehnološki napredek sili organizacije v spremembe. Poslovni modeli se morajo odzivat na dinamiko industrije in okolja. Pri tem igrajo pomembno vlogo tehnološki transferji, ki znanstveno znanje spremenijo v inovativne tržne izdelke (Chesbrough, 2010).

Inovacije poslovnih modelov so preoblikovale celotne panoge in redistribuirale milijarde dolarjev dobičkov: diskontne prodajalne kot je Wal-Mart, ki je prišel na trg s pionirskim modelom, nizko cenovne letalske družbe so v Ameriki prevzele že 55% trga (Johnson, Christensen & Kagermann, 2008).

Za razliko od drugih inovacij, inovacije poslovnih modelov zahtevajo spremembe temeljnih odločitev o delovanju podjetja. Medtem ko se inovacija izdelka lahko postopoma vključi v proizvodnjo, je inovacija poslovnega modela običajno radikalna in v mnogih primerih preoblikuje podjetje (Smith, 2016).

Tako uveljavljenim kot zagonskim podjetjem lahko spodleti kljub temu, da imajo možnosti na trgu, nove ideje, primerne vire in talentirane podjetnike. Možen vzrok za neuspeh je lahko njihov poslovni model. Problemi, ki so lahko v samem poslovnem modelu so lahko: nasičen trg, preveč konkurence, ovire pri vstopu na trg, slabe poslovne ideje, težko uresničljive poslovne ideje, zastarela tehnologija (Smith, 2016).

Inovacije poslovnih modelov so večkrat nujne in pomembne, čeprav jih je težko doseči. Podjetja morajo imeti pozitiven odnos do eksperimentiranja s poslovnim modelom. Zato morajo določiti interne vodje za spremembo poslovnega modela, da spremljajo rezultate procesov in predlagajo nov, za podjetje boljši poslovni model. Hkrati mora organizacijska kultura v podjetju najti način za vključitev novega poslovnega modela (Chesbrough, 2010).

Zagonska podjetja lahko prinesejo pomemben vpogled v same poslovne modele. Sama zagonska podjetja so dejansko poskus na resničnem trgu, z resničnimi izdelki in resničnimi kupci (Chesbrough, 2010).

2.3 Model razvoja strank

2.3.1 Namen in osnove metode razvoja strank

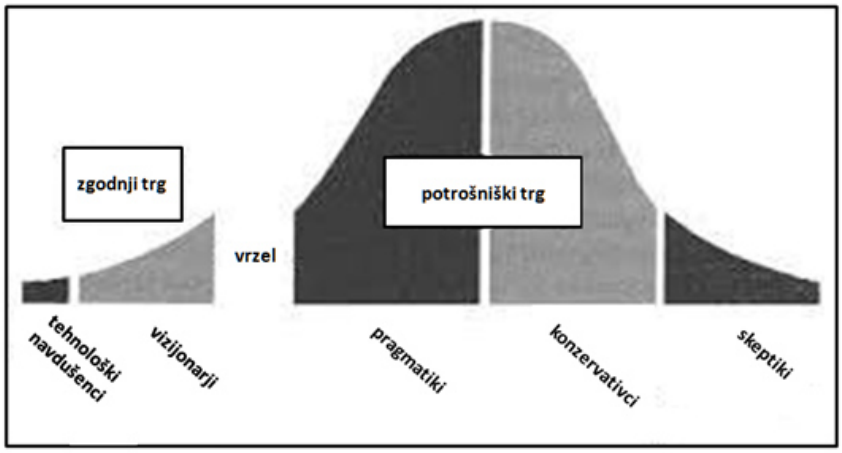
Ustanovitev novega podjetja, ne glede na to ali gre za visoko tehnološko podjetje, malo podjetje ali iniciativo v okviru večje korporacije, je vedno neke vrste tveganja. Po formuli, ki je veljala desetletja, je bilo ob ustanovitvi podjetja potrebno napisati poslovni načrt, ga predložiti investitorjem, zbrati ekipo, predstaviti izdelek in začeti prodajati kolikor mogoče. Tudi ob upoštevanju zaporedja dogodkov, so možnosti preživetja podjetja slabe. Novejše raziskave, ki jih je v okviru Harvardske poslovne šole vodil Shinkar Ghosh kažejo, da 75% zagonskim podjetjem spodleti (Blank, 2013).

Večina zagonskih podjetij nima definiranih procesov, s katerimi bi odkrivali svoje trge, našli prve stranke, preverjali domneve glede njihovega izdelka ter širili svoje poslovanje (Blank 2007). Konec 20. stoletja je večina zagonskih podjetij delovala na način, da so procese, kot razvoj in lansiranje izdelka ter obvladovanje življenjskega cikla, kot so delovali v velikih podjetjih, samo prilagodila. Blank in Dorf (2012) razlikujeta med uspešnimi zagonskimi podjetji ali »zmagovalci« in podjetji, ki jim ni uspelo ali »poraženci«. Zmagovalci so po njunem mnenju opustili tradicionalen management izdelka od razvoja do predstavitve. Spoznali so, da je vizija podjetja samo vrsta hipotez, ki jih je potrebno preveriti pri strankah. Poraženci večinoma rigidno sledijo načrtu razvoja, pri čemer izostane mnenje kupcev kot točka preverjanja v procesu (Blank & Dorf, 2012). Z namenom, da bi zmanjšal tveganje pri zagonu podjetja, je Blank (2007) na podlagi izkušenj z zagonskimi podjetji izoblikoval metodo, ki daje prednost eksperimentiranju pred planiranjem, povratnim informacijam pridobljenih od kupcev pred intuicijo ter cikličnemu procesu pri oblikovanju prototipov in testiranjih pred pretiranemu vnaprejšnjemu oblikovanju izdelkov (Blank 2007). Metodi razvoja kupcev in vitkega zagonskega podjetja sta vse bolj uporabljeni obliki podjetniškega procesa, ki slonita na testiranju hipotez, vendar ne v tradicionalnem smislu. Podjetnike vzpodbujata k opazovanju okolja, zbiranju informacij in oblikovanju in ocenjevanju domnev z namenom, da točno presojajo in sprejmejo pravilne odločitve (Jork & Danes, 2014).

Model razvoja strank temelji na nekaterih že poznanih teorijah:

- **Odkrivanje na osnovi planiranja (angl. *discovery driven planning*):** v negotovih situacijah projiciranje v prihodnost na podlagi izkušenj ne daje dobrih rezultatov. Proces odkrivanja na osnovi planiranja zahteva, da so predpostavke o primernem poslovnem modelu oblikovane in testirane (McGrath & MacMillan, 1995).
- **Raziskovalna metoda vodilnega uporabnika (angl. *lead-user research method*):** metodo je leta 1986 razvil Eric von Hippel. Zgrajena je na osnovi ideje, da nekaj vodilnih uporabnikov poseduje največ znanja in največje razumevanje novih izdelkov ali storitev (von Hippel, 1986). Podjetja lahko te uporabnike identificirajo in jih pritegnejo k skupnemu razvoju s proizvajalcem (Herstatt & von Hippel, 1992). Vodilna podjetja v industriji se poslužujejo te metode v izogib tveganjem in neuspehu pri razvoju novih izdelkov. Podjetja identificirajo vodilne uporabnike in jih vključujejo v razvoj izdelkov (Lüthje & Herstatt, 2004).
- **Krivulja sprejemanja novih tehnologij (angl. *technology life cycle adoption curve*)**, ki jo je leta 1976 razvil Rogers. Posamezniki sprejemajo nove tehnologije postopoma in procesu sprejemanja inovacij ne sledijo enako hitro. Roger je osvojitelje novih tehnologij razvrstil na osnovi obnašanja, vrednot in pristopov v pet kategorij, ki jih je kasneje Moore (1991) imenoval tehnološki navdušenci, vizionarji, pragmatiki konzervativci in skeptiki. Porazdelitev teh kategorij uporabnikov sledi normalni porazdelitvi oziroma Gaussovi krivulji, kot je prikazana na Sliki 5. Ker se osebnostne lastnosti posameznikov v teh kategorijah razlikujejo, je Moore (1991) v krivuljo trga vnesel **vrzeli**. Največja vrzel je med vizionarji in pragmatiki, saj so tehnološki navdušenci in vizionarji osredotočeni na dolgoročne prednosti, ki jih lahko imajo inovacije. Prav tako so se pripravljeni sprijazniti s slabostmi inovativnega izdelka kot sta njegova kompleksnot uporabe ali neoptimalno delovanje.
- **Nova Lanchesterjeva strategija (angl. *New Lanchester Strategy*)** je ime dobila po Fredericku Williamu Lanchesterju, ki je leta 1916 oblikoval strategijo za zračne sile. Formuliral je 1. In 2. Lanchestrov zakon, ki opisujeta vojaške operacije potrebne za zmago pri bitkah. Blank (2007) je Lanhesterjevo teorijo uporabil za tipizacijo trgov. Glede na lastnosti posameznega tipa trga je ugotavljal, kašna je smiselnost vstopa zagonskega podjetja na tržišče.
- **Rušilne tehnologije (angl. *disruptive technologies*),** ki jih je v svoji knjižni uspešnici *The Innovator's Dilemma* , ki je izšla leta 1997 opisal Clayton Christensen. Chistensen (1997) tehnologije deli na dve kategoriji:
  - **vzdrževalne tehnologije**. Vzdrževalne tehnologije se nanašajo na posamične izboljšave že sprejetih tehnologij.
  - **rušilne tehnologije** osnovane na še ne uporabljenih odkritjih. Ker so nove, imajo te tehnologije problem z učinkovitostjo. Naslavljajo omejen krog uporabnikov in mogoče tudi nimajo dokazane praktične uporabe (Find a Tech Definition, b.l.).

Slika 5: Vrzeli v krivulji življenjskega cikla sprejemanja tehnologij



Vir: G.A. Moore, Crossing The Chasm, 1991, str.33

Velike korporacije so zasnovane za delo z vdrževalnimi tehnologijami, torej z izboljšavami že uveljavljenih tehnologij. To jim omogoča atraktivne dobičke, saj se širijo v področja višje cenovnega trga, kjer ciljajo na bolj zahtevne kupce, ki niso zadovoljni z obstoječo ponudbo. Inovacije, ki omogočajo uveljavljenim podjetjem višje zaslužke s prodajo boljših izdelkov njihovim najboljšim kupcem, so vzdrževalne. Vzdrževalne tehnologije zajemajo tako preproste, postopne inženirske izboljšave kot tudi tehnološke preboje v izboljšanju delovanja izdelkov. Uveljavljena podjetja niso vedno prva na trgu z vzdrževalnimi inovacijami, so pa na koncu skoraj vedno na vrhu, ker imajo več virov in motivacije za zaslužek (Christensen, 2002).

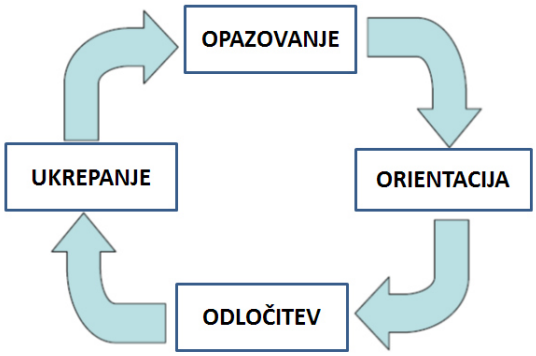
- **Model zanke OODA (angl. *observe, orient, decide, act*)** prav tako izhaja iz vojaške strategije. Zanka OODA je sestavljena iz štirih korakov, ki se povezujejo v krog, kot je prikazan na Sliki 6 (Brehmer, 2005):
  - opazovanje je zbiranje informacij o okolju. Učinkovitost te faze je uspešna, če so informacije kakovostne, primerne in pravočasno zbrane,
  - orientacija pomeni usmeriti se proti nasprotniku in pridobiti dobro pozicijo,
  - odločitev o tem, kaj bo naslednji korak,
  - ukrepanje je uresničitev odločitve.

Adolph (2006) je razložil, kako se ta model odločanja odraža v razvoju programske opreme:

- medtem, ko se razvojnemu timu ne zdi, da konkurira z nasprotnikom, lahko vsaka nepričakovana sprememba v okolju tim dezorientira, torej deluje kot nasprotnik,
- zanka OODA razlaga, kako priti v prednostno pozicijo v negotovem okolju. V prednostni poziciji je stopnja svobode večja. To se pri razvoju PO nanaša tako na okolje kot na delovanje tima,

- večina razvojnih timov deluje v konkurenčnih okoljih. Konkurenti direktno ali indirektno prispevajo k negotovosti okolja.

Slika 6: OODA zanka



Vir.: W.S. Angerman, Coming Full Circle with Boyd's OODA Loop Ideas: An Analysis of Innovation, Diffusion and Evolution, 2004, str. 2.

2.3.2 Model razvoja izdelka

V prejšnjem stoletju je narasel pritisk globalne konkurence in v večini panog je postal razvoj novih izdelkov glavna gonilna sila konkurenčnosti podjetij. Schilling in Hill (1998) ugotavljata, da mora podjetje, ki je osredotočeno na razvoj novega produkta, izpolniti dva kritična cilja: ujemanje s potrebami strank in skrajšanje časa, potrebnega za lansiranje izdelka na trg. Vendar na podlagi pregleda literature o razvoju novih izdelkov iz 80-ih in 90-ih let prejšnjega stoletja ni jasno, kako in kdaj naj bi dobavitelji in kupci bili ustrezno vključeni v proces razvoja novega izdelka (Brown & Eisenhardt, 1995).
Blank (2007) vidi vzrok za neuspeh zagonskih podjetij predvsem v njihovem osredotočanju na sam razvoj izdelka, pri čemer pozabijo na potencialne kupce. Razvoj na osnovi izdelka razdeli na štiri faze:

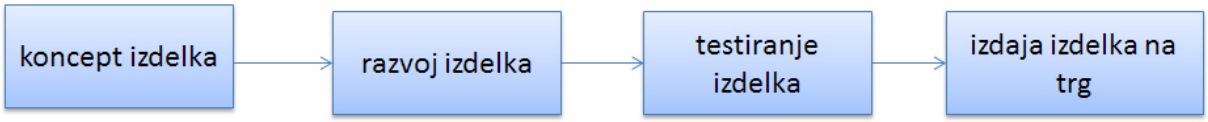
- **Koncept in semenska faza:** v tej fazi ustanovitelji oblikujejo svoje vizije glede podjetja v nekaj ključnih idej, ki so osnova za poslovni načrt. Razčistijo se vprašanja glede potrebe po izdelku in tehnična vprašanja glede samega izdelka. Na osnovi statističnih in marketinških raziskav ter intervjujev s kupci se naredi raziskava trga. Definirajo se distribucijski kanali in osnovna predpostavke glede cene.
- **Razvoj izdelka:** V tej razi razvijalci naredijo zasnovo izdelka, specifikacije, najamejo ljudi za razvoj, ocenijo čas sprostitve izdelka na trg in stroške razvoja. Marketing natančneje oceni velikost trga in hkrati načrtuje promocijski program. Velikokrat že vnaprej pripravi promocijski material in zaposli ljudi.
- **Testiranje izdelka:** Inženirji sodelujejo z manjšo skupino uporabnikov in preverjajo, če izdelek deluje v skladu s specifikacijami ter ugotavljajo morebitne napake. Tržniki izdelajo dokončen načrt trženja in marketinške materiale. Določi se pozicija izdelka.

Prodajniki poiščejo prve beta uporabnike, gradijo distribucijske kanale in zaposlijo prodajalce (Blank, 2007, str. 4).

- **Izdaja izdelka na trg:** To je zadnja faza tega modela in cilj, za katerega je podjetje živelo. Z delujočim izdelkom se podjetje prestavi v fazo trošenja sredstev za prodajo. Prodajni oddelek zaposluje ljudi in gradi prodajno organizacijo. Izgradnja prodajnih kanalov in podpora marketingu zahteva dosti finančnih sredstev.

Iz diagrama razvoja izdelka na Sliki 7 je razvidno, kaj je slabost tega modela: v celoti ignorira kupca. Zagonska podjetja, ki delujejo po tem modelu, nimajo težav z izdelkom, ampak s kupci in trgi. Podjetje je osredotočeno na lansiranje izdelka prvim kupcem, kar pa ne pomeni, da razume njihove potrebe (Blank, 2007).

Slika 7: Model razvoja izdelka



Vir: S. Blank The four steps to the epiphany, 2007, *str. 2*

Nekatere napake, ki izhajajo iz modela razvoja izdelka so (Blank & Dort, 2012, str. 8):

- podjetniki so prepričani, da vedo, kdo so kupci in katere funkcije mora imeti nov izdelek
- osredotočeni so na dan lansiranja novega izdelka
- poudarjajo izvajanje poslovnega načrta namesto hipotez, testiranja, učenja in iteracije
- domneve o uspehu vodijo v prehitro rast podjetja

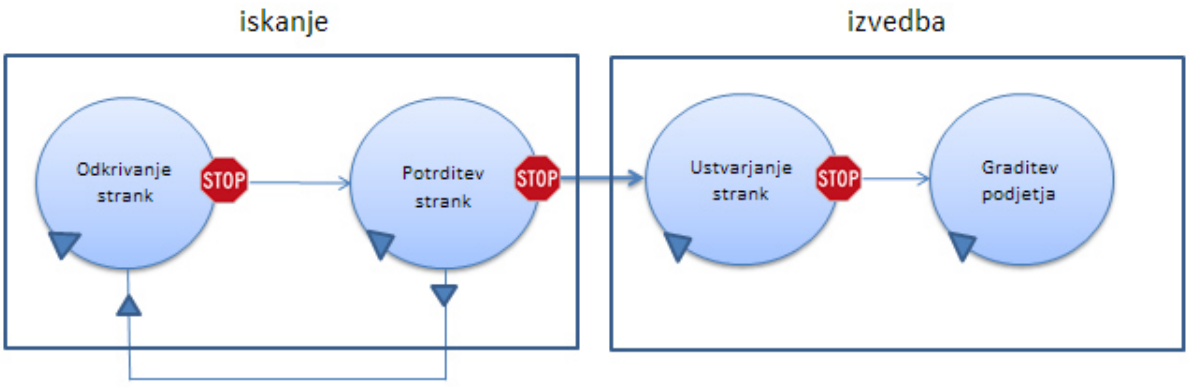
V nasprotju z zagonskimi podjetji, model razvoja izdelka dobro deluje v obstoječih podjetjih z znanimi kupci. V primeru, ko je trg dobro definiran, je smiselno funkcije izdelka specificirati vnaprej (Blank & Dorf, 2012, str 3).

2.3.3 Koraki v modelu razvoja strank

Razvoj modela strank prikazan na Sliki 8 je zasnovan z namenom, da reši probleme modela razvoja izdelka. Model loči vse aktivnosti, ki so vezane na kupca, že v zgodnji fazi nastanka podjetja in jih oblikuje v procese, oblikovane v štiri korake: **odkrivanje strank**, **preverjanje strank, kreiranje strank** in **izgradnjo podjetja**. Glavna razlika med modelom razvoja strank in modelom razvoja izdelka je, da je pri modelu razvoja strank vsak korak narisan kot krožna pot s puščicami v smeri, ki nakazujejo ponavljajoče se procese. Vsak krog v procesu je iterativen. Za razliko od modela izdelka, model razvoja

strank predvideva večje število ponavljanj pri vsakem koraku, preden naredimo prav. Pri razvoju izdelka bi korak nazaj pomenil napako, medtem ko je to pri modelu strank naraven in dragocen učenja in odkrivanja (Blank, 2007, str. 19). Krog »preverjanje strank« ima dodatno iterativno zanko, ki vodi nazaj do odkrivanja strank. Preverjanje strank je ključna točka razumevanja, ali imamo izdelek, ki ga stranka želi.

Slika 8: Model razvoja strank



Vir: S. Blank & B. Dorf, The Startup Owner's Manual, 2012, *str. 23*

2.3.3.1 Odkrivanje strank

Prva dva koraka v modelu razvoja kupcev sta učenje in odkrivanje. V fazi odkrivanja kupcev morajo podjetniki ugotoviti, kdo so kupci, od kod prihajajo in kako jih bodo dosegli. V tem koraku je potrebno vizijo ustanoviteljev glede podjetja prevesti v hipoteze o vsaki komponenti poslovnega modela (Blank, 2007, str. 42; Blank & Dorf, 2012, str. 24):

- **Hipoteze glede izdelka** se sestojijo o ugibanjih ustanoviteljev o izdelku in njegovem razvoju: na njegovo funkcionalnost, prednosti, ki naj bi jih prinašal, intelektualno lastnino, analizo odvisnosti, načrt izdaje izdelka na trg ter strošek lastništva izdelka za stranke, ki ocenjuje vse stroške, ki jih bo imel kupec ob nakupu in uporabi izdelka.
- **Hipoteze v zvezi s kupci:** v tem delu si morajo podjetniki zapisati predpostavke glede dveh ključnih področij: kdo so njihovi kupci in s katerimi problemi se srečujejo. V tem delu podjetniki obravnavajo predvidevanja o tipu kupcev, njihovih problemih, o enem dnevu v življenju kupca (omogoča boljše razumevanje ciljnih skupin), upravičenost kupčeve naložbe v izdelek in minimalen nabor funkcij, ki jih mora imeti izdelek.
- **Hipoteza o ceni in prodajnih kanalih** sta medsebojno povezani, saj je v primeru direktne prodaje ponavadi potrebna investicija v programsko opremo ali v pomoč pri fizični distribuciji izdelka. Pri izbiri prodajnega kanala mora zagonsko podjetje upoštevati naslednje kriterije: dodano vrednost prodajnega kanala, ceno in kompleksnost izdelka in obstoječe (če so) nakupovalne navade kupcev (Blank, 2007, str. 51).

- **Hipopoteza o kreiranju potreb kupcev** mora odgovoriti na vprašanje, kako bodo kupci izvedeli za podjetje in izdelek. Najmočnejši pritisk na nakupno odločitev kupcev je lahko nekaj, kar podjetje ne ustvarja. Na vsakem trgu ali v panogi so skupine ljudi, ki kreirajo trende, stile in mnenja. Potrebno je ugotoviti, kdo vpliva na nakupne odločitve (Blank, 2007, str. 53).
- **Hipoteza o tipu trga.** Pomembno je, da podjetje ve, ali bo nastopilo na obstoječem trgu ali na novo segmentiranem trgu ali pa na novem trgu, kar ni vedno čisto jasno. Podjetje mora vedeti, na katerem trgu bo nastopilo, ker je od tipa trga odvisno pozicioniranje izdelka (Blank, 2007, str. 55).
- **Hipoteza o konkurenci** je še posebej pomembna, če podjetje vstopa na obstoječi trg ali na trg, ki se je na novo segmentiral. Potrebno je vedeti, zakaj je nov izdelek boljši od konkurenčnih. Potrebno je oceniti tržne deleže, ki jih ima konkurenca. Če podjetje stopa na nov trg, kjer nima konkurence, se je potrebno vprašati, če bo nov izdelek ljudem omogočil kaj novega in če jih bo to sploh zanimalo (Blank & Dorf, 2012, str. 124).

2.3.3.2 Preverjanje strank

Prepoznavanje začetnih kupcev in ugotavljanje, koliko je izdelek ali storitev, ki ju razvijamo, zanje pomembna, je ključnega pomena za podjetniški uspeh (Cespedes et al., 2012). Na tej stopnji mora podjetje testirati hipoteze zbrane v prvi fazi (Blank, 2007). Podjetniki za to potrebujejo nekaj več kot je raziskava trga v tradicionalnem smislu. Potrebujejo proces, ki njihovo intuicijo in entuziazem obrne v tržno usmerjena dejstva, ki jih lahko uporabijo za potrjevanje, zasuk ali opustitev začetnega poslovnega modela. (Cespedes et al., 2012). Cilj tega koraka je zgraditi ponovljiv prodajni proces za prodajno in tržno ekipo, ki bosta sledili. Sestaviti je potrebno preproste teste, ki potrdijo ali ovržejo hipoteze in jih izvesti na prvih 50-ih potencialnih kupcih (Blank & Dorf, 2012, str. 189).

2.3.3.3 Ustvarjanje strank

Že sredi prejšnjega stoletja je Drucer (1954) spoznal, da je namen posla ustvarjanje kupcev (v Cespedes et al., 2012). Faza ustvarjanja strank gradi na uspehu, ki ga je imelo podjetje pri prvih prodajah (Blank, 2007 str. 22; Peršolja, 2013). Šele po večih iteracijah in pivotiranjih v fazi preverjanja, podjetniki »zadanejo« pravi poslovni model, ki ga je smiselno nadgrajevati. V modelu razvoja strank predstavlja **ustvarjanje strank** kreiranje potreb kupcev in to, da privedemo te potrebe v prodajne kanale (Osterwalder & Pigneur, 2011). Vsem zagonskim podjetjem so skupni štirje procesi, ki gradijo ustvarjanje strank (Blank, 2007, str. 129):

- cilji prvega leta delovanja
- pozicioniranje izdelka in podjetja
- lansiranje izdelka in podjetja

- ustvarjanje potreb kupcev z oglaševanjem, odnosi z javnostjo, predstavitvami...

Korak **ustvarjanje strank** se nahaja za korakom **preverjanje strank** z namenom, da večja poraba sredstev za marketing šele sledi točki, ko zagonsko podjetje pridobi prve kupce. To omogoča podjetju kontrolo nad prekomernim trošenjem sredstev (Blank, 2007).

2.3.3.4 Izgradnja podjetja

To je čas prehoda podjetja iz neformalnega učečega se razvojnega tima v bolj formalno organizacijo z oddelki kot razvoj, prodajo in marketing (Blank, 2007, str. 22; Peršolja, 2013).

Zagonsko podjetje se običajno začne z vizijo ustanoviteljev o novem izdelku ali storitvi, vizijo o tem, kako bo izdelek dosegel kupce in vizijo o tem, zakaj bi veliko ljudi kupilo ta nov izdelek ali storitev. Vendar je vse, kar ustanovitelji v začetku verjamejo o izdelku in trgu, bolj ali manj ugibanje (Blank, 2007 str.33). Prehod iz malega zagonskega podjetja v veliko podjetje ni vedno linearen graf prodaje. Rast prihodkov iz prodaje je povezan z dejstvom, da mora podjetje doseči dosti širšo skupino kupcev, kot je na primer skupina tehnoloških navdušencev. Da podjetje doseže osrednji trg pa mora oblikovati prodajno, marketinško in poslovno strategijo glede na tip trga v katerega stopa (Blank 2007, str. 163). Pri prehodu iz zagonskega podjetja v večje podjetje se mora podjetje organizirati tako, da je osredotočeno na svojo misijo, da se razširi in prestopi vrzel med zgodnjimi kupci in osrednjim trgom. Osredotočanje na misijo pomeni, da mora zagonsko podjetje postati agilno podjetje, ki se še vedno odziva s podjetniško hitrostjo, vendar z dosti večjo skupino ljudi (Blank, 2007, str. 167).

2.3.4 Model razvoja strank kot osnova za nadaljnja dela

V modelu razvoja strank je zajeta Blankova ideja o tem, da morata biti v zagonskem podjetju poslovanje in marketing obravnavana enako pomembno kot inženiring in razvoj izdelka (Ries, 2011). Metodo sta prva uporabila podjetnika Will Harvey in Eric Ries v njunem podjetju IMVU, ustanovljenem leta 2004 v Silicijevi dolini, kjer je Blank nastopal kot mentor in investitor (Blank, 2007; Ries 2011). Blankovo delo je postalo znano med investitorji s Standfordske univerze in ljudmi, ki so poznali problematiko zagonskih podjetij. Vendar je le malo kdo sledil strogo predpisanim diagramom modela razvoja strank. Glede na to, da so osnovni koraki v metodi nadalje razdeljeni v (ponavadi) štiri korake, je korakov dosti več. Blank se je v nadaljnjem delu osredotočil na relativno zrelo fazo zagonskega podjetja in poudaril razliko med **fazo iskanja** in **fazo izvajanja** (Virani, 2012). Ries je iskal ideje, ki bi potrdile njegove izkušnje, izven podjetništva. Največ je raziskoval proizvodnjo, iz katere izhaja večina modernih teorij managemanta. Ideje vitke proizvodnje in vitkega razmišljanja je uporabil v procesu inovacij in tako dobil okvir za model, ki ga je imenoval **vitko zagonsko podjetje** (Ries, 2011). Da bi podjetnikom olajšal

udejanjiti načela vitkega zagonskega podjetja je Maurya (2012) izdal priročnik »Delaj vitko«, ki je podroben pregled metode vitkega zagonskega podjetja. Pojme, ki jih je vpeljal Ries (2011) in vitki okvir, kot ga je predlagal Maurya (2012), sta Blank in Dorf (2012) ponovno združila z modelom razvoja strank v priročniku »The Startap Owner's Manual«.

2.4 Metoda vitkega zagonskega podjetja

2.4.1 Osnove metode vitkega zagonskega podjetja in njena razširjenost

Tako kot Blank (2007) tudi Ries (2011) umrljivost zagonskih podjetij povezuje z dejstvom, da metode managementa, ki se uporabljajo v zrelih podjetjih, za zagonska podjetja niso primerne. Dober poslovni načrt, trdna strategija in temeljita raziskava trga v negotovih okoliščinah, v katerih delujejo zagonska podjetja, niso zadosten temelj za uspeh zagonskega podjetja. Veliko inovatorjev je opustilo namen ali zaprlo podjetje zaradi pomanjkanja prilagojenega procesa managementa, primernega za zagonsko podjetje. Po drugi strani so podjetniki, ki so spoznali, da uporaba metod tradicionalnega managementa v zagonskem podjetju ne deluje, prevzeli »samo naredi« miselnost. Ta šola verjame, da, v kolikor je problem management podjetja, je pravi odgovor na to kaos, v katerem deluje veliko zagonskih podjetij. Po Riesovem mnenju morajo biti tudi zagonska podjetja obvladovana in procesi v podjetju definirani (Ries, 2011). Medtem ko so agilne metode razvoja prvenstveno usmerjene v rešitve in odgovore »kako« narediti izdelek, ne odgovarjajo na vprašanje »kateri« izdelek je potrebno narediti. Zato so v glavnem uporabljene v situaciji, kjer je problem znan, rešitev pa ne. V konceptu zagonskega podjetja pa ponavadi nista dobro poznana niti problem niti rešitev (Bosch el al., 2013). Cilj pristopa vitkega zagonskega podjetja je, da sistematično in natančno vodi podjetnike, da odkrivajo dragocene dele njihovega izdelka in podjetniške vizije, kakor tudi da ovržejo dele ideje, ki nimajo realnih osnov (Lalic et al., 2012).

Vitko zagonsko podjetje izhaja iz **vitkega mišljenja**, managerskega pristopa, ki je bil razvit v Toyotinem proizvodnem sistemu in je radikalno spremenil način poteka proizvodenj in oskrbovalnih verig (Virani, 2012). Metoda vitkega zagonskega podjetja je vitko miselnost vpeljala v podjetništvo. Vodilo vitke metodologije je izboljšanje učinkovitosti v zagonskem podjetju. Učinkovitost pomeni vedeti natančno, kaj kupci želijo, koliko bodo za to plačali in kakšen bo izdelek. Brez poznavanja tega bo porabljen čas in denar za sledenje napačni poti (Gaffney et al., 2014). Vitko zagonsko podjetje uporablja Blankov koncept razvoja strank, ki je kombiniran z uporabo hitrih, iterativnih in agilnih metod razvoja. Je sinteza tehnik agilnega razvoja in metod za raziskave trga in pomaga podjetnikom, da uspešno razvijajo inovativne izdelke in storitve v tesnem sodelovanju s svojimi strankami (Patz, 2013).

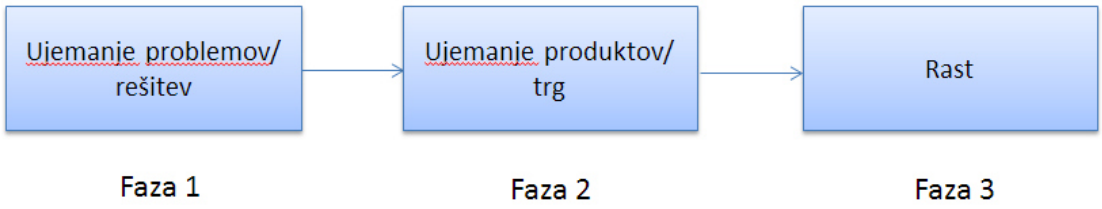
Od leta 2008 je metoda vitkega zagonskega podjetja vedno bolj priljubljena med podjetniki po vsem svetu. Nastala je iz dobrih praks podjetja v silikonski dolini in se razvila v

metodologijo, ki podjetnikom omogoča uspešen zagon inovacijskega podjetja (Bosch et al., 2013; Patz, 2013). Eric Ries je svoje ideje najprej delil na blogu in vitka skupnost jo je osvojila. Metoda se je razširila po izidu knjižne uspešnice »The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses« leta 2011, Ries pa jo je nadalje popularizira z vrsto gostujočih predavanj. **Vitko zagonsko podjetje** je postalo gibanje, ki ima privržence v sedemnajstih državah po svetu, tudi v Sloveniji. Privrženci gibanja se srečujejo na rednih sestankih v več kot 94-ih mestih po svetu. Kljub temu, da Ries metodo razvil za visoko tehnološka podjetja, ki se ukvarjajo z razvojem programske opreme, za katera je tudi delal, je gibanje preseglo okvirje tega področja. Metodo, ki je nova paradigma pri ustanavljanju podjetij, so sprejeli vlagatelji tveganega kapitala, podjetniški pospeševalniki in inkubatorji (Rogers & Duening, 2014). V podjetniških pospeševalnikih, ki so v osnovi nosilci treh tipov kapitala – ekonomskega (financiranje), socialnega (mreženje) in kulturnega kapitala (status) – je metoda postala jedro izobraževalnega načrta (Heines, 2014).

2.4.2 Načela metode vitkega zagonskega podjetja

Po metodi vitkega zagonskega podjetja gre vsako zagonsko podjetje skozi tri faze, ki so prikazane na Sliki 9.

Slika 9: Tri faze zagonskega podjetja



Vir: A. Maurya, Delaj vitko, 2014, str.8.

Prva faza - ujemanje problema in rešitve – daje odgovor na vprašanje, če imajo potencialni kupci problem, ki ga je vredno rešiti. V kolikor tak problem obstaja, je potrebno zbrati najmanjši nabor zahtevanih lastnosti in izdelati **najosnovnejši sprejemljivi produkt** v nadaljevanju NSP (angl. *minimum viable product*, MVP) (Maurya, 2014). NSP ima najosnovnejše lastnosti, ki so nujne za preverjanje domnev o izdelku, trgu ali kupcih (Rogers & Duening, 2014; Gaffney et al. 2014). NSP je osnova za testiranje rešitve oziroma za preverjanje ujemanja izdelka in trga. V kolikor se izkaže, da je izdelek privlačen za prodajo, je podjetje doseglo ujemanje izdelka in trga, ki mu sledi tretja faza-rast ali večanje obsega poslovnega modela (Maurya, 2014).

Preden pride do faze ujemanja izdelka in trga, je zagonsko podjetje osredotočeno na **učenje** in **sukanje**. **Zasuk** (angl. *pivot*) je izraz, ki opisuje spremembo smeri zagonskega podjetja, v kolikor se izkaže, da prvotne domneve o izdelku in trgu ne držijo (Maurya, 2012; Patz, 2013).

Metoda vitkega zagonskega podjetja je osnovana na petih načelih:

- **Podjetniki so povsod:** po Riesovi definiciji je zagonsko podjetje institucija, ustanovljena da kreira nove proizvode in storitve v pogojih izjemne negotovosti. To pomeni, da lahko pristop zagonskega podjetja uporabljajo podjetja ne glede na njihovo velikost ali panogo v kateri delujejo. Podjetniki so vizionarji, ki so sposobni predvidevati prihodnost v svojih panogah. Pripravljeni so tvegati v iskanju inovativnih rešitev za njihova podjetja (Ries, 2011, str. 8).
- **Podjetništvo je management:** zagonsko podjetje ni samo izdelek, je tudi institucija, ki zahteva posebej prilagojeno obvladovanje dela v nepredvidljivih okoljih. Zagonsko podjetje je nabor simultanih aktivnosti. Podjetnik mora vse aktivnosti uravnovešat in se hkrati učiti iz svojih napak (Ries, 2011, str.8). Klasični proces raziskave in razvoja je potrebno dopolniti, da deluje tudi v zagonskih podjetjih (Gaffney et al., 2014).
- **Validirano učenje (angl. *validated learning*):** namen zagonskih podjetij ni samo zaslužek ali služenje strankam, njihov namen je tudi naučiti se, kako zgraditi trajnostno poslovanje. To učenje se lahko s pomočjo eksperimentov, s katerimi lahko podjetniki testirajo elemente njihove vizije, znanstveno potrdi (Ries, 2011, str. 8). Način delovanja zagonskega podjetja mora biti iskanje in ne izvajanje. Iskanje se lahko nanaša na izdelek, prodajni kanal ali kupce (Gaffney et al., 2014).
- **Cikel naredi-meri-spoznaj (angl. *Build-Measure-Learn*):** je cikel povratnih informacij kupcev, ki je obvezen del postopka spoznavanja v vitkem zagonskem podjetju (Maurya, 2012). Vsa uspešna zagonska podjetja morajo biti usmerjena v pospeševanje hitrosti zanke povratnih informacij (Ries, 2011, str.8).
- **Računovodstvo, ki upošteva inovacije:** Potrebno se je osredotočiti tudi na to, kako meriti napredek, kako postaviti mejnike in kako prioritizirati delo. Računovodstvo, ki upošteva inovacije, predstavlja dobre izvedljive meritve, ki morajo zajeti tudi rezultate učenja in jih je mogoče uporabiti za razumevanje, kako napreduje učenje (Gaffney et al. 2014). Računovodski izkazi v zgodnji fazi zagonskega podjetja ali samega izdelka, ko večinoma še ni nobenih denarnih prihodkov, niso tako uporabni, saj bi prikazovali izgubo. Klasično računovodstvo je v zagonskih podjetjih relativno enostavno, saj so prihodki, dobiček, prosti denarni tokovi in večina drugih kazalnikov blizu ničle. Zato klasična računovodska poročila kot so izkaz uspeha, izkaz finančnih tokov in bilanca stanja ne pridejo v poštev do te mere kot pri ustaljenih podjetjih (Kos, 2013). Samo prihodek kot merilo napredka podjetja, bi v začetni fazi oviral napredek podjetja, zato je potrebno izbrati tudi druge kazalnike. Ključni pomen analitike je, da pomaga vodstvu sprejemati odločitve, ki omogočajo zagonskemu podjetju razvoj pravega

izdelka in nastop na pravem trgu (Kos, 2013). Temelj uspešne analitike so dobro izbrani kazalniki (angl. *actionable metrics*), ki dejansko merijo napredek zagonskega podjetja. Nasprotje uporabnih kazalnikov so kazalniki brez vrednosti (angl. *vanity metrics*), ki beležijo samo trenutno stanje izdelka, ne dajejo pa vpogleda v to, kako so analitiki prišli do rezultata in kako naj bi nadaljevali (Mauraya, 2013). Ključni kazalniki so tisti, ki se nanašajo na jedro poslovanja, se odražajo v prihodkih, so osredotočeni na stranke ter kažejo na vzrok in posledice. Kazalniki morajo biti razumljivi in tudi primerljivi s prejšnjim obdobjem (Kos, 2013).

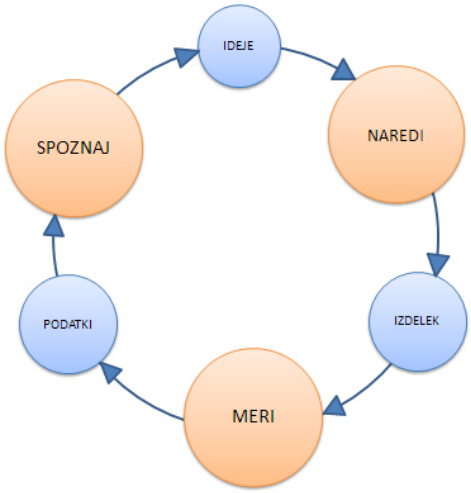
2.4.3 Uporaba načel metode vitkega zagonskega podjetja

Z namenom, da na primeru zagonskega podjetja uporabi znanstveno metodo, Ries (2011) predlaga oblikovanje hipotez, ki jih podjetje tudi testira. Vizijo podjetja oziroma podjetnikov je na začetku potrebno razgraditi na posamezne komponente. Ries imenuje dve najpomembnejši predpostavki, ki jih imajo podjetniki **hipoteza vrednosti** in **hipoteza rasti** (Ries, 2011, str. 61). Hipoteza vrednosti je osnova za testiranje, ali izdelek ali storitev zares prinašata dodano vrednost kupcem. Testiranje hipoteze rasti pa daje odgovore na to, kako bodo novi kupci odkrili izdelek ali storitev in kako se bo širil od zgodnjih uporabnikov (angl. *early adopters*). Na podlagi teh hipotez podjetje predstavi potencialnim kupcem nastajajoč izdelek oziroma NSP. Na osnovi povratnih informacij, ki jih dobi od kupcev, se podjetje odloči ali bo razvoj zasukalo v drugo smer, v drug izdelek, ali pa bo nadaljevalo razvojno pot obstoječega NSP (Rogers & Duening, 2014). Cikel povratnih informacij kupcev je obvezen del postopka spoznavanja v vitkem zagonskem podjetju in se imenuje **cikel naredi-meri-spoznaj** (Ries, 2011, Maurya 2014). Cikel je tristopenjski proces prikazan na Sliki 10 in je osnova validiranega učnega procesa (Maurya, 2014).

Slika 10 prikazuje, da je proces razvoja izdelka, kot ga zagovarja metoda vitkega zagonskega podjetja, krožen, iterativen in eksperimentalen. Ko so hipoteze postavljene, se podjetje pomakne v fazo ustvarjanja, oziroma v prvo fazo »naredi« v krogu naredi-meri-spoznaj. V tej fazi je na podlagi ideje narejen najosnovnejši sprejemljivi produkt. NSP je verzija izdelka, ki omogoča, da naredimo celoten cikel po zanki naredi-meri-spoznaj. Za razliko od tradicionalnih razvojev izdelka, ki so ponavadi dolgi in premišljeni ter stremijo k popolnosti izdelka, je cilj NSP, da se z njegovo pomočjo začne cikel učenja. NSP nima vseh funkcionalnosti, ki bodo morda potrebne kasneje, je pa dovolj izdelan, da merimo njegov vpliv na kupce (Ries, 2011, str. 77). Je poskusni predmet za empirično testiranje hipoteze vrednosti in mora biti zgrajen z minimalnimi napori v minimalnem času. Podjetje mora tudi dizajnirati poskuse, v katerih je uporabljen NSP, da z njimi potrdi ali ovrže hipotezo vrednosti in hipotezo rasti (Rancic Moogk, 2012). Kar se kompleksnosti tiče, je NSP lahko zelo preprost test ali pa zgodnji prototip, ki še nima vseh funkcij. Za odločitev o kompleksnosti NSP ne obstajajo formule, osnovana mora biti na presoji. Večina podjetnikov in razvijalcev ponavadi precenjuje število funkcij, ki jih mora imeti NSP. V kolikor obstajajo dvomi, je vedno dobro, da se NSP poenostavi (Ries, 2011, str. 95). Je pa

NPS tudi nekoliko kontroverzna ideja, ker bi kupci lahko izdelek zaznali kot da je hitro in poceni narejen (Eisenmann v Nobel, 2011). Potem, ko NSP stestirajo zgodnji uporabniki, lahko zagonsko podjetje nadaljuje z iteracijo z uporabo agilnih razvojnih praks kot so Scrum ali druga kanban načela. Kratki razvojni koraki podjetju omogočajo ustvariti majhne prilagoditve izdelka (Heines, 2014). NSP je osnova za fazo merjenja. Korak »meri« ustreza merjenju odziva kupcev, na osnovi kvalitativnih in kvantitativnih podatkov pa pridemo do novega **spoznanja** s katerim hipoteze potrdimo ali ovržemo. Pri nadaljevanju procesa se izdelek spreminja, dokler zadostno število kupcev ne potrdi njegove sprejemljivosti (Rogers & Duening, 2014; Maurya, 2014). Cilj zagonskega podjetja je, da naredi izdelek, ki so ga stranke pripravljene kupiti. Da pridejo do tega, morajo podjetniki izvajati **poskuse** (angl. *experiments*), s katerimi strogo testirajo posamezne elemente njihovega ponujenega izdelka (Rogers & Duening, 2014). Poskus je prizadevanje zagonskega podjetja, da testira svojo strategijo. Poskus sledi znanstveni metodi, začne se z jasnimi hipotezami, ki so predvidevanja o tem, kaj naj bi se zgodilo. Podjetje ta predvidevanja testira z eksperimenti. Kot so znanstveni eksperimenti zasnovani na teoriji, tako so eksperimente zagonskih podjetij vodi njihova vizija (Ries, 2011, str. 56). Poskus je cikel, ki zajema celoten validiran učni proces cikla naredi-meri-spoznaj.

Slika 10: Cikel naredi-meri-spoznaj



Vir: A. Maurya, Delaj vitko, 2014, str. 12.

Zasuk (angl. *pivot*) je sprememba, ki je potrebna, če podjetje ne napreduje dovolj, da bi verjelo, da so začetne hipoteze pravilne. Metoda vitkega zagonskega podjetja ne ponuja formule za odločitev ali nadaljevati v isti smeri ali zasukati, vedno so zato potrebni človeški elementi kot so vizija, intuicija in presoja (Ries, 2011, str.149). V zagonskem podjetju se zasuk izvede po nekaj odločilnih testih, ki ovržejo začetne hipoteze. Pri zasuku lahko podjetje spremeni ciljni segment kupcev tako, da ga zoži ali razširi, lahko spremeni sam izdelek z dodajanjem ali odvzemanjem funkcij, spremeni lahko prodajne kanale ali poslovni model. Bistvo zasuka je narediti spremembo in idealno je, da se po zasuku

ponovno oblikuje nabor domnev in hipotez, ki jih bo podjetje testiralo (Eisenmann v Nobel, 2011). Signala, ki lahko nakazujeta potrebo po zasuku sta: zmanjšana učinkovitost eksperimentov z novim izdelkom in občutek, da bi moral biti razvoj izdelka bolj produktiven. V vsaki odločitvi o zasuku pa morata sodelovati tako razvojni tim kot vodstveni tim (Ries, 2011, str.164). Zasuk je posebna sprememba, zasnovana za testiranje novih temeljnih hipotez glede izdelka. (Ries, str. 173). Z njim pa je neločljivo povezan problem in sicer je to tveganje, da bi zasuk razumeli kot nelojalnost viziji podjetja. Ustanovitelji z idejo velikokrat težko prepričajo zaposlene, investitorje, stranke in partnerje in nova spremenjena ideja se lahko zdi kot izdaja (Nobel, 2011).

Vzroki za zasuk so lahko različni (Ries, 2011, str.173):

- **Zasuk zaradi približevanja:** Kar je bilo mišljeno samo kot funkcija, postane cel izdelek.
- **Zasuk zaradi oddaljevanja:** Obratna situacija od približevanja – celoten izdelek postane funkcija v veliko večjem izdelku.
- **Zasuk zaradi segmentiranja kupcev:** Podjetje spozna, da izdelek rešuje problem strank, ampak ne tistih strank, ki so jih imeli v načrtu. Hipoteza izdelka je delno potrjena, rešuje pravi problem, ampak drugim kupcem.
- **Zasuk zaradi potreb kupca:** Kot posledica dobrega poznavanja kupca se včasih razjasni, da problem, ki ga rešuje izdelek, ni njihov največji. Pogosto se odkrijejo drugi problemi vezani na izdelek, ki jih podjetje prav tako lahko reši.
- **Zasuk zaradi platforme:** Nanaša se na spremembo iz aplikacije na platformo ali obratno. Zagonska podjetja, ki si prizadevajo kreirati nove platforme, večkrat začnejo s prodajo ene aplikacije za njihovo platformo. Šele kasneje se pojavi nova platforma.
- **Zasuk poslovne arhitekture:** izhaja iz ugotovitve Geoffreya Moora, da podjetja na splošno sledijo enemu od obeh poslovnih arhitektur: visoke marže in majhni volumni (kompleksni sistemi) in nizke marže in veliki volumni. Pri tem modelu zagonsko podjetje zamenja poslovno arhitekturo.
- **Zasuk za zajem vrednosti:** Gre za modele ustvarjanja prihodkov. Zajem vrednosti je bistveni del hipoteze izdelka, spremembe smeri pa imajo ponavadi daljnosežne posledice za ostale strategije.
- **Zasuk gonila rasti:** Podjetje spremeni strategijo rasti, da bi raslo hitreje ali da bi bilo bolj profitabilno.
- **Zasuk pri prodajnih kanalih:** Prodajni kanali so mehanizem s pomočjo katerega podjetje dobavlja svoj izdelek kupcem. Zasuk pri prodajnih kanalih je spoznanje, da bi isto rešitev uspešneje prodajali preko drugih kanalov.
- **Zasuk tehnologije:** Včasih se zgodi, da podjetje odkrije način za rešitev problema z uporabo čisto drugih tehnologij.

Zasuki so stalna dejstva v življenju rastočega podjetja. Celo po tem, ko podjetje doseže začetni uspeh, mora nadaljevati z zasuki.

Podjetništvo je grajenje organizacije v pogojih ekstremne negotovosti, zato je njegova najvitalnejša funkcija učenje. Naučiti se je potrebno, kateri elementi naše strategije zares delujejo in kateri so nesmiselni. Odkriti je potrebno, če je naša strategija prava pot do trajnostne rasti podjetja. V modelu vitkega zagonskega podjetja je uveden koncept **validiranega učenja**. Validirano učenje je proces, ki empirično dokazuje, da je tim odkril dragocene resnice o sedanjih in prihodnjih možnosti podjetja. Je konkretnejši, točnejši in hitrejši od marketinških napovedi ali klasičnih poslovnih načrtov (Ries, 2011, str. 38).

Veliko ljudi je naučenih, da pri svojem delu poudarja samo enega od elementov povratne zanke »naredi-meri-spoznaj«. Inženirji so na primer osredotočeni na to, kako izdelati najbolj učinkovit izdelek, managerji so po drugi strani osredotočeni na strategije. Veliko podjetnikov se osredotoča samo na najboljšo idejo o izdelku ali na najbolje dizajniran začetni izdelek. Posamezne od teh aktivnosti niso bistvenega pomena, zagonsko podjetje se mora osredotočiti na skupen čas, ki je potreben za cikel učenja in to je bistvo managementa zagonskega podjetja (Ries, 2011, str. 76).

Standardno računovodstvo ni vedno primerno za ocenjevanje zagonskih podjetij, ker so zagonska podjetja nepredvidljiva. S pomočjo računovodstva, ki upošteva inovacije, se domneve spremenijo v kvantitativni finančni model. Tako računovodstvo deluje v treh stopnjah (Ries, 2011, str. 118):

- uporabi NSP da pridobi dejanske podatke o tem, kje se podjetje nahaja,
- zagonsko podjetje se mora usmeriti iz izhodišča proti idealnemu izdelku tako, da naredi vse mikro spremembe in izdelek optimizira. Po tem koraku podjetje doseže točko odločitve, ki ji sledi tretji korak,
- zasuk ali ohranitev smeri. V kolikor podjetje dobro napreduje proti idealnemu izdelku – se ustrezno uči in spoznanja učinkovito uporablja – je smiselno, da nadaljuje v tej smeri. V kolikor ne, je uporabljena strategija napačna in potrebne so spremembe. Ko podjetje spremeni smer, se proces začne znova.

Izraz »vitko« v vitkem zagonskem podjetju ima korenine v Toyotinem proizvodnem sistemu, saj se metoda vitkega zagonskega podjetja nanaša na odpravo izgub, pri tem gre za izgubo časa in denarja (Nobel, 2011). Pospeševanje zanke naredi-meri-spoznaj je bistveno za konkurenčnost zagonskega podjetja. Pospeševanje cikla lahko podjetje uresniči z uporabo načel iz TPS, ki so že dolgo uveljavljeni (Ries, 2011, str. 228):

- **Delo v majhnih korakih:** omogoča zagonskim podjetjem, da minimizirajo porabo časa, denarja in truda, za katere bi se lahko izkazalo, da so zapravljeni.

- **Rast:** Hitreje, kot se bo obračala zanka, hitreje bo raslo podjetje. Za dosego trajnostne rasti uporabljajo zagonska podjetja gonilo rasti. Novi kupci prihajajo zaradi aktivnosti prejšnjih strank. Pri tem lahko gre za ustno izročilo, stranski učinek uporabe izdelka ali s financiranim oglaševanjem.
- **Prilagajanje** procesov trenutni situaciji. Kljub temu, da je hitro ukrepanje strategija zagonskih podjetij, je potrebno postopke včasih upočasnit. Kvalitete ni mogoče zamenjati za čas: v kolikor se napake ne odpravijo takoj, upočasnijo proces kasneje, kar vodi v predelave, znižano moralo in pritožbe kupcev – za vse to se porabljata čas in energija. Hiter razvoj NSP ne pomeni, da se pri njegovih karakteristikah ustavimo, ampak da ga v ponavljanjih zanke naredi-meri-spoznaj izboljšujemo. Lahko se uporabi tudi metoda **petih zakajev**, ki deluj tako, da se sprašujemo o vzrokih problema. Investiramo v odpravo simptomov posameznega problema o katerem se sprašujemo, investicije pa so sorazmerne z velikostjo simptoma.

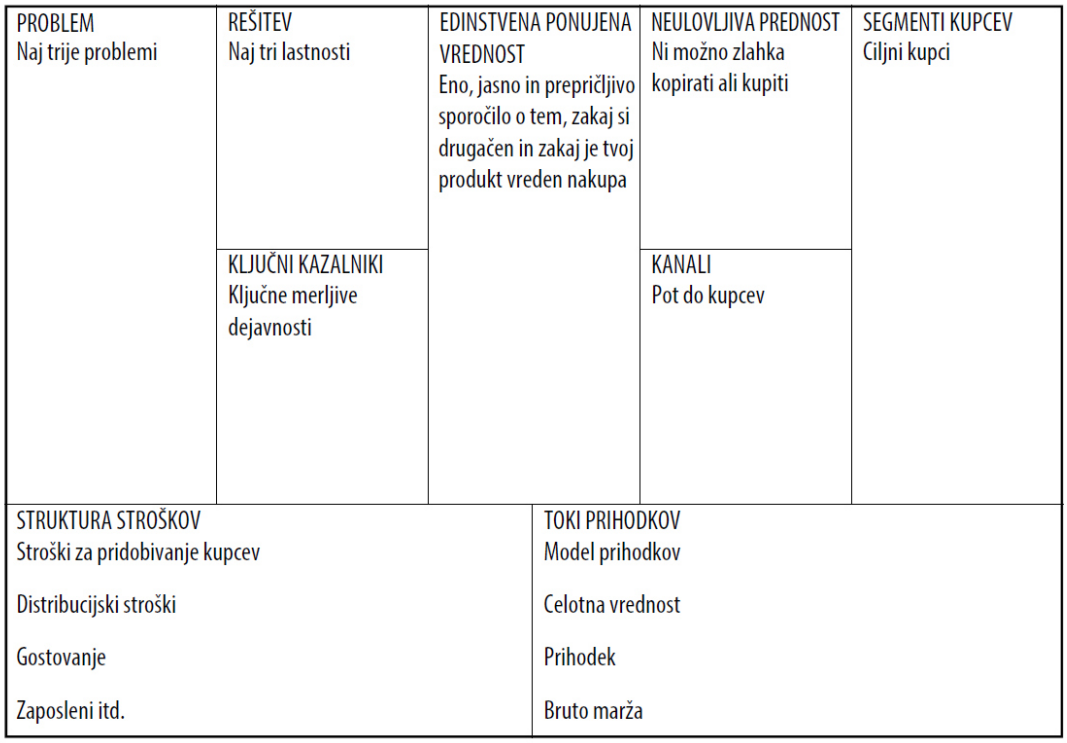
Zagonsko podjetje mora tudi takrat, ko zraste, negovati kulturo inovacij tako, da uravnoteži potrebe obstoječih strank z izzivom, da poiščejo nove stranke. Da bi uspeli, morajo biti inovacijski timi pravilno strukturirani. V omejenem času, ki ga dopuščajo sredstva, potrebuje tim v zagonskem podjetju avtonomijo za razvoj in lansiranje novih izdelkov. Dobro je, da so skupine navskrižno delujoče, da so vsi vpleteni v projekt seznanjeni z delom vseh skupin. Posegi v delo in nadzor odobritev upočasnjujeta učni cikel (Ries, 2011, str. 255). Vitko zagonsko podjetje predstavlja zanimive koncepte in ideje, ki pa jih je v praksi težje uresničit (Maurya 2012; Bosch et al. 2013). Maurya (2012) je ideje vitkega zagonskega podjetja in njihovo praktično izvajanje opisal v priročniku »Delaj vitko«, v katerem je opisan proces izvajanja usmeritev metode vitkega zagonskega podjetja v podjetju, ki razvija programsko opremo. Proces je razdeljen na tri korake (Bosch et al., 2013):

- Dokumentiranje začetnega načrta: pri tem je potrebno zajeti celoten poslovni model, ne samo izdelka oziroma rešitve. Cilj je razviti celotno vizijo podjetja.
- Identificiranje najbolj tveganih delov načrta: tveganja se lahko nanašajo na izdelek, kupce ali trg. Potrebno jih je identificirati, oceniti in prioritizirati. Z najvišje ocenjenimi tveganji se je potrebno najprej spopasti.
- Sistematično testiranje načrta: proces se osredotoči na sistematično testiranje načrta z uporabo cikla naredi-meri-spoznaj.

Da bi domneve o poslovnem modelu bile bolj pregledne, Maurya (2012) v priročniku »Delaj vitko« priporoča podjetnikom uporabo **vitkega okvirja** (angl. *lean canvas*), izpeljanega iz poslovnega okvirja (angl. business model canvas), ki ga je razvil Alex Osterwalder (Osterwalder & Pigneur, 2010). Vitki okvir pomaga identificirati najbolj tvegane domneve, ki jih je potrebno najprej testirati (Blank & Dorf, 2012; Maurya, 2012).

Vitki okvir vsebuje relevantne informacije o kupcih in izdelku, prav tako pa vključuje razmislek o finančnem toku. Zagotavlja vizualen, lahko razumljiv, pa še vedno celosten pregled poslovne ideje (Patz, 2013).

Slika 11: Vitki okvir podjetja



Vir: A. Mauraya, Delaj vitko, str.5.

V primerjavi s poslovnim načrtom je vitki okvir narejen hitro, je jedrnat in – ker je izdelan na eni strani – tudi prenosen (Mauraya (2012). Vitki okvir omogoča pogled na devet gradnikov podjetja na eni strani. Vsaka komponenta vitkega okvirja vsebuje hipoteze, ki jih je potrebno testirati (Blank, 2013).

2.4.3 Primerjava metod razvoja strank in metode vitkega zagonskega podjetja

Zaradi uporabe cikla naredi-meri-spoznaj je metoda vitkega zagonskega podjetja posebej primerna v situacijah, kjer še nista povsem jasna niti problem, niti rešitev (Patz, 2013). Model razvoja strank se sestoji iz štirih zaporednih faz: odkritja strank, preverjanje strank, ustvarjanja strank in izgradnje podjetja (Blank, 2007, Maurya, 2012; Ries, 2011). Prvi dve fazi sta namenjeni viziji podjetnikov in razčlenitvi te vizije v domneve, ki jih je mogoče testirati, drugi dve fazi pa se nanašata na izgradnjo potrebe po izdelku ter na rast podjetja in njegovo transformacijo iz zagonskega podjetja v podjetje, ki uporablja preverjen poslovni model. Metoda vitkega zagonskega podjetja oziroma njen cikel naredi-meri spoznaj pa se v glavnem uporablja v prvih dveh fazah – odkrivanju in preverjanju strank – z namenom, da podjetje pridobi dokaz o trajnostni poslovni priložnosti. Metoda vitkega

zagonskega podjetja vodi podjetje skozi faze razumevanja problema in kupcev, preverjanja prototipa izdelka in preverjanje rešitve. V kolikor rešitev ni potrjena, mora podjetje izvesti zasuk in ponovno začeti z odkrivanjem strank, dokler rešitev ni usklajena z zahtevami strank in lahko postane ponovljiv in razširljiv proces, ki vodi v nadaljnji fazi ustvarjanja strank in izgradnje podjetja (Blank & Dorf, 2012; Blank, 2007). V primerjavi z metodo razvoja strank, metoda vitkega zagonskega podjetja poudarja pomen razumevanja problema pred začetkom razvijanja rešitve (Bosch et al., 2013).

2.4.4 Kritike metode vitkega zagonskega podjetja

Vse bolj priljubljen fenomen vitkega zagonskega podjetja, ki prihaja iz prakse, še ni primerno upoštevan v akademskih revijah ali literaturi in tudi ne opisan z akademskega stališča. Nekaj let po izidu knjige »Vitko zagonsko podjetje« še ni dosti znanstvenih potrditev o uspešnosti metode, še manj pa je informacij o njenih morebitnih slabostih (Patz, 2013). Iskanje znanstvene literature po geslih »vitko« ali »zagonsko podjetje« v kombinaciji z gesli »kritika«, »napaka« ali »slabost« ni dalo rezultatov. Z uporabo enakih gesel na brskalnikih Google in Yahoo da več rezultatov, ki niso znanstvena literatura, metodo vitkega zagonskega podjetja pa presojajo z različnih vidikov. Kritike modela vitkega prihajajo v glavnem od praktikov, najdemo pa jih na njihovih blogih ali spletnih straneh namenjenih novim tehnologijam in podjetništvu. Največ kritik je deležen koncept NSP, ki ga lahko podjetniki narobe razumejo kot nekaj, kar je hitro in poceni narejeno: izdelek je lahko »preveč najosnovnejši in premalo sprejemljiv« (Dager, 2014). Izdaje preslabih izvedb izdelka lahko uničijo posel (RoutJoy, b.l.). Po drugi strani pa izdaja skromnega in nedodelanega izdelka lahko povzroči, da ga prezgodaj odkrijejo in dalje razvijejo drugi razvijalci (Burgsone, 2012). Strategija lansiranja NSP in dodajanja posameznih novih funkcionalnosti na osnovi povratnih informacij, pridobljenih od kupcev, vsebuje še eno tveganje. Intervjuji s kupci lahko pokažejo, da si ti želijo funkcionalnosti, ki so že bile v osnovnem načrtu izdelka, pa so bile kasneje odvzete, da je izdelek ustrezal kriteriju »najosnovnejši«. Posledično so potrebne ponovne potrditve teh funkcionalnosti s strani kupcev, kar povzroča nepotreben dodaten cikel ter izgubo časa in denarja (Kortmann, 2012).

Računovodstvo, ki upošteva inovacije in ki kot merilo napredka upošteva učenje, je dobrodošlo, vendar je upoštevanje standardnih računovodskih praks nujno. Standardne računovodske izkaze je potrebno v zgodnji fazi podjetja sicer drugače interpretirati, podjetniki pa jih ne smejo ignorirati ali imeti za nepomembne (Burgstone, 2012)

Želja po minimiziranju negotovosti lahko zatre umetniško vizijo. Izdelek je včasih posledica želje po tem, da bi podjetnik uresničil svojo vizijo. Pri tem lahko stalno preverjanje mnenj omejuje kreativnost. V nekaterih primerih je bolje obrniti proces: izdelati nekaj, kar je blizu vizije, brez sklepanja prevelikih kompromisov in potem najti ljudi, ki bodo z izdelkom zadovoljni (Hussein, 2015; McCahill, 2015).

Na trgu se pojavlja vedno več izdelkov, ki želijo ljudi izobraziti o tem, kako postati podjetnik in zato ni presenetljivo, da so metodo predelali, jo uporabili za druge namene in jo ponovno prodali na tečajih za podjetnike (Heines, 2014). Številni članki in tečaji o hitri priučitvi metode vitkega zagonskega podjetja preveč poenostavljajo proces. Ta potem ne vodi k miselnosti, s pomočjo katere bi posameznik razumel testiranje hipotez (Hussein, 2015; Vlaskovitz, 2012).

Proizvodna podjetja so se desetletja spoprijemala z uporabo vitkih tehnik in vitke filozofije in so v tem procesu dozorevanja filozofijo zamenjala z racionalnim in pragmatičnim razmišljanjem in precejšnjo zmernostjo. Metoda vitkega zagonskega podjetja je iz vitke metodologije povzela samo pragmatični pristop, ki ga uporablja za učinkovito izgradnjo podjetij. V primerjavi z vitko proizvodnjo, ki je naravnana v učinkovitost procesov, se metoda vitkega zagonskega podjetja bolj ukvarja s stroškovno učinkovitostjo (Pax, 2013).

3 UVAJANJE VITKIH PRISTOPOV V ZAGONSKEM PODJETJU BLOKER

3.1 Predstavitev podjetja

Podjetje Bloker[1] SAS je bilo ustanovljeno leta 2014 in ima sedež v Franciji, kjer sta bila ustanovitelja pred leti sodelavca v ameriškem multi-nacionalnem podjetju, ki je izdelovalo opremo za telekomunikacije. Leta 2015 pa je bila zaradi ameriške zakonodaje, ki ureja poslovanje ameriških šol, ustanovljena še ameriška podružnica Bloker s sedežem v Pao Altu. Podjetje ima dva redno zaposlena razvijalca, dva razvijalca pa za Bloker delata honorarno. Lastnika nista zaposlena v podjetju Bloker. Lastnik, ki živi v Franciji, je inženir računalništva in telekomunikacij, pri svojem delu se je specializiral za dostopovna omrežja. V podjetju, kjer je trenutno zaposlen, se ukvarja tudi z organizacijo in s poslovnim razvojem – razvijanjem trga za nove izdelke. Lastnik, ki je zadolžen za tehnične rešitve in razvoj, je strokovnjak za varnost na internetu. Dela v Sloveniji, kjer tudi poteka Blokerjev razvoj. Denar, ki ga vlagata v podjetje, služita kot zaposlena v podjetjih, za katera sta delala že prej. Na to, da je možno ustvariti NSP in testirati ujemanje produkta in trga ter ohraniti službo, opozarja tudi Maurya (2012). Slabša stran takega načina organizacije pa je, da sta se projektu Bloker posvečata v prostem času, oziroma v okviru, ki ga omogoča delo, za katerega prejemata denarno nadomestilo. Od motivov za ustanovitev podjetja, ki jih Sauermann (2015) opisuje kot preference posameznika glede denarnih in nedenarnih koristi povezanih z delom, sta bila za ustanovitelja podjetja Bloker pomembna intelektualni izziv in avtonomija pri delu.

[1] Ime podjetja je spremenjeno

3.2 Tehnologija, ki je podlaga inovacijam podjetja Bloker

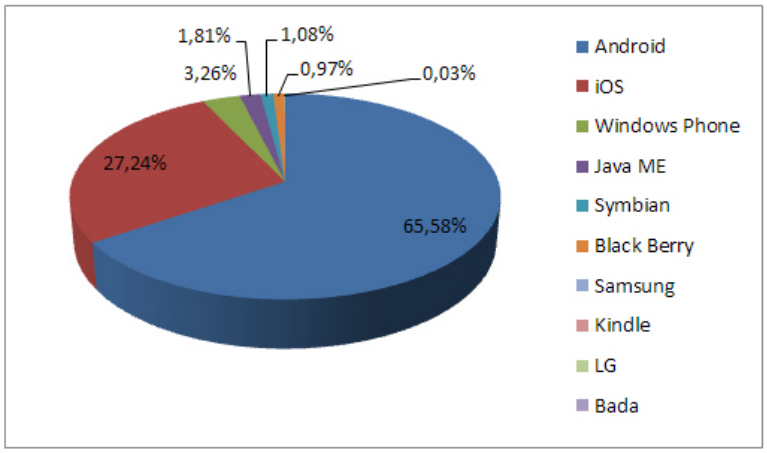
3.2.1 Računalništvo v oblaku

Računalništvo v 21. stoletju prehaja iz obdobja, ko so uporabniki bili lastniki računalnikov, v obdobje, ko uporabniki ne bodo imeli svojih računalnikov, ampak bodo imeli samo dostop do računalniške strojne in programske opreme (Regelado, 2011). Z napredkom moderne družbe so voda, elektrika, plin in telefon, ki so nujno potrebni za vsakdanjo rutino, na voljo preko javnih storitev. Vedno bolj je zaznana vizija, da bo poleg teh javnih dobrin internet postal peta javna dobrina, ki bo uporabnikom ves čas na voljo po potrebi. Uporabniki dostopajo do želenih storitev, ne glede na to, kje so te storitve nameščene, ali kako so dobavljene. Uporabniki plačajo storitev ponudniku, na podlagi tega, koliko jih koristijo (Buyya, Yeo, Venugopal, Broberg & Brandic, 2009). Za tak dostop do računalniških storitev se uporablja žargonski izraz **računalništvo v oblaku**, ki ga je leta 2006 populariziral takratni Googlov izvršni direktor Eric Schmidt v njegovem predavanju na konferenci o strategijah spletnih brskalnikov (Regalado, 2011). Računalništvo v oblaku pomeni storitve in dostop do podatkov in programov s pomočjo interneta namesto s pomočjo trdega diska uporabnikovega računalnika. Izraz »oblak« izhaja iz časov, ko so na predstavitvah in tablah upodabljali infrastrukturo oziroma internetne strežnike kot puhaste oblake, ki delijo informacije (Griffith, 2016). S tem, ko postaja uporaba storitev v oblaku bolj pogosta in število aplikacij, ki so na voljo uporabnikom, narašča, eksponentno narašča tudi naša odvisnost od interneta. Vedno več uporabnikov se zanaša na strežnike, ki omogočajo funkcionalnosti večine njihovih računalniških aplikacij, kakor tudi hranjenje podatkov. Posebej pri uporabi brskalnikov, ki delujejo na daljavo, kot je Google Chrome OS, so uporabniki skoraj pri vsakem delu z računalnikom povsem odvisni od interneta (Roberts & Hamdani, 2011). Operacijski sistem, aplikacije in strežnike sicer na daljavo uporabljamo tako kot bi jih imeli doma ali v podjetju, vendar lahko zakupljene zmogljivosti spreminjamo glede na naše potrebe, pa tudi z vzdrževanjem strojne opreme ni skrbi. Med najbolj znane infrastrukturne računalniške oblake spada Amazonov Elastic Compute Cloud ali EC2 (Čehovin, 2011).

3.2.2 Googlova operacijska sistema in Googlov Chromebook

Google igra osrednjo vlogo na trgu potrošne elektronike. Njegova strategija so kombinacije programske in strojne opreme. Google je razvil svoj operacijski sistem Android in z vodilnimi proizvajalci originalne opreme sodeloval pri razvoju mobilnih naprav, ki so bile optimizirane za njegov operacijski sistem. S pomočjo te strategije je Google dobiček generiral iz prodaj na Google Play Storu, uradni Googlovi trgovini za digitalno distribucijo in z nizkimi maržami pri prodaji pametnih telefonov in oglaševalnih storitev vezanih na Android. Kot prikazano na Sliki 12, je ta načrt deloval izjemno dobro pri prodaji pametnih telefonov in tablic, kjer je Android od leta 2013 najbolj prodajan operacijski sistem in je imel leta 2014 že 66% tržni delež (Ronen, 2015).

Slika 12: Tržni delež OS Android pri mobilnih napravah in tablicah v % za leto 2014



Vir: Prirejeno po Net Market Share, Mobile/ Tablet Operating System Market Share, 2016

Vendar Googlu s to strategijo ni uspelo prodreti na tradicionalni trg osebnih računalnikov. Za razliko od potrošne elektronike, je trg osebnih računalnikov zrel in obvladujeta ga uveljavljena ponudnika Microsoft z Windowsi in Apple z operacijskim sistemom OS X. Microsoft in Apple sta že tako uveljavljena, da ju je težko zrušiti. Google je na področje osebnih računalnikov vstopil leta 2011 s prodajo prvih Chromebookov. Chromebook je prenosni računalnik, ki deluje na podlagi spletnega operacijskega sistema Chrome OS. Je lahek, ker ima minimalen pomnilnik in omogoča takojšen dostop do interneta. Narejen je predvsem za dostop do storitev v oblaku, shranjevanje dokumentov v oblaku in za uporabo aplikacij, dosegljivih na spletu (Donovan, 2015). Ko so leta 2011 Chromebooki prišli na trg, je bila njihova cena okoli petsto ameriških dolarjev in praktično ekvivalentna ceni cenejših prenosnih osebnih računalnikov. Poleg tega, da Chromebook ni ponujal primerljive funkcionalnosti kot Microsoftov Office, je imel v začetku tudi težave z nastavitvijo tiskalnikov, zato je bila v očeh uporabnikov njegova dodana vrednost nižja. Leta 2013 so proizvajalci Chromebookov dosegli ceno med dvesto in tristo ameriških dolarjev, narejenih pa je bilo tudi dosti izboljšav, zaradi česar so se uporabniki že bili pripravljeni odreči polni uporabi Microsoft Office (Nielson, 2014). Zaradi uporabe operacijskega sistema Chrome OS, ki je osnovan na računalništvu v oblaku in cene, ki je nižja od najcenejših naprav, ki delujejo na Windowsih, je Chromebook postal rušilna tehnologija (Nielson 2014; Trounce 2015). Da bi rešili problem pri prodiranju na zreli trg osebnih računalnikov, se je Google osredotočil na razvijajoči se segment e-učenja. Po podatkih raziskovalne in svetovalne družbe Gartner, ki je specializirana za področje informacijskih tehnologij, je bilo leta 2014 na globalnem trgu okoli 70% Chromebookov prodanih izobraževalnim ustanovam.

Tabela 3: Prodaja Chromebookov-tržni deleži po segmentih in regijah leta 2014 (v %)

Regija	**Izobraževanje**	**Druga področja**	**Potrošniki**
APAC (Azija, Pacifik)	68,8	16,5	14,7
EMEA (Evropa, Srednji vzhod, Afrika)	72,3	0,9	26,8
Združene države	60,3	1,1	38,6

Vir: L. Goasduff & J. Rivera, *Gartner Says Worldwide Chromebook Sales Will Reach 7.3 Million Units in 2015, 2015*

Prodaja Chromebookov v poslovnem segmentu ostaja nizka, čeprav je Chromebook zanimiv za mala in srednje velika podjetja (v nadaljevanju MSP). Zaradi nizke cene v primerjavi z osebnimi računalniki bodo novo nastala podjetja in MSP, ki nimajo virov za nabavo drage informacijske tehnologije, v prihodnosti verjetno razmišljala o Chromebookih (Goasduff & Rivera, 2015).

Z geografskega vidika, pa je bilo leta 2014 kar 84% Chromebookov prodanih v Severni Ameriki, kar pomeni, da Severna Amerika predstavlja daleč največji trg pred področjem EMEA (Evropa, Srednji vzhod, Afrika), kamor je bilo leta 2014 prodanih 11% Chrombookov (Tabela 4).

Tabela 4: Prodaja Chromebookov po regijah leta 2014 (v tisočih)

Regija	**2014**	**2015**	***2016**
Severna Amerika	4,820	6,020	6,177
Latinska Amerika	142	178	224
EMEA (Evropa, Srednji vzhod, Afrika)	620	866	1,276
APAC z Japonsko	146	225	276
Skupaj	**5,728**	**7,289**	**7,953**

Legenda: *Pri letu 2016 gre za napoved prodaje

Vir: L. Goasduff & J. Rivera, *Gartner Says Worldwide Chromebook Sales Will Reach 7.3 Million Units in 2015, 2015*

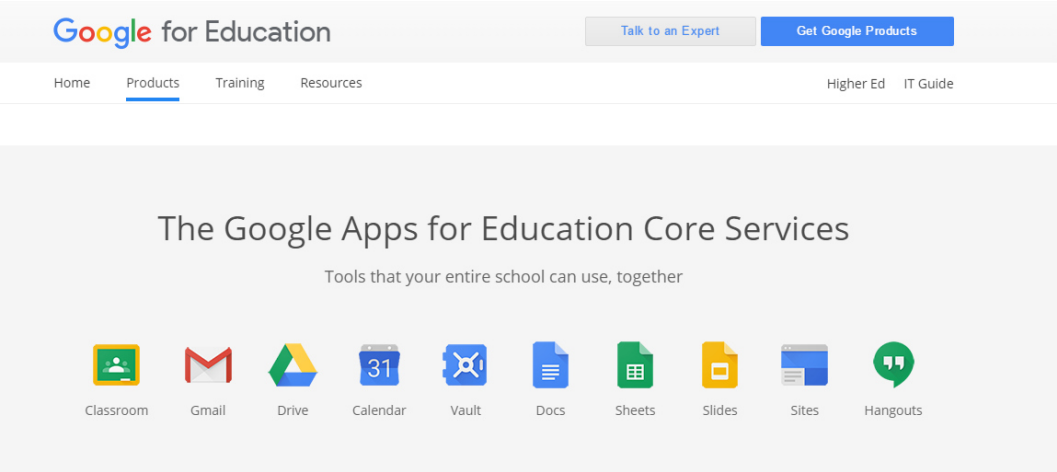
3.2.3 Googlovi spletni portali in trgovine

Za Blokerjeva izdelka Bloker razširitev in Bloker Manager so Googlovi portali in spletna trgovina osnovni prodajni kanali.

Decembra 2010 je Google predstavil *Chrome Web Store* – spletni portal, ki omogoča uporabnikom nakup in instalacijo spletnih aplikacij. Ta spletni portal je bil ena od prvih platform, ki omogoča drugim podjetjem, da prodajajo njihove spletne aplikacije in pri tem uporabljajo enoten plačilni sistem (Kincaid, 2011). Žal pa spletni portal ni omogočal prodaje aplikacij podjetjem iz večine držav srednje in vzhodne Evrope vključno s Slovenijo.

Google Apps je paket Googlovih storitev in aplikacij v oblaku namenjen podjetjem (Google Apps for Work) in šolam (Google Apps for Education). Je prostor kjer podjetje dobi osnovni nabor storitev za poslovanje na internetu kot so: registracija internetne domene, upravljanje elektronskih naslovov (gmail), pomnilniški prostor v oblaku (Drive), telefonski in video konferenčni klici (Hangouts) in ostale aplikacije kot so urejanje dokumentov, preglednic, koledar (Slika 13).

Slika 13: Ekranska slika storitve Google Apps za izobraževanje



Vir: Google for Education, 2016

Google Application Marketplace je spletna trgovina, namenjena predvsem poslovnim uporabnikom. Ponuja spletne aplikacije v oblaku, ki jih razvijajo druga podjetja in ki so kompatibilne z Google Apps. Deluje tako, da uporabnika napoti na spletno stran razvijalca aplikacije, ali pa da si uporabnik aplikacijo, ki jo je razvila tretja oseba, namesti neposredno iz spletne trgovine (Google Apps Marketplace, 2016).

3.3 Bloker – od ideje do podjetja

3.3.1 Problem neželenih spletnih vsebin

Mobilne naprave, kot so pametni telefoni in tablice, prežemajo skoraj vsa okolja, kjer ljudje opravljajo vsakodnevne aktivnosti. Z objavo Applovega iOS in Googlovega operacijskega sistem Android so mobilne aplikacije postale enostavnejše, povečale pa so se tudi hitrosti prenosov podatkov iz mobilnih omrežij. Zahvaljujoč 3G in 4G tehnologiji ter razširjenosti brezžičnih omrežij, lahko preko mobilnih naprav dostopamo do spleta skoraj od povsod (Srirama, Paniagua & Liivi, 2013). Zaradi širjenja škodljivih spletnih vsebin kot so pornografija, nasilje in sovražna sporočila, so sistemi za učinkovito filtriranje spletnih vsebin nujno potrebni. Veliko programov za filtriranje spletnih vsebin je komercialno dostopnih in uporabniki si lahko poskusne verzije prenesejo na svoje naprave iz spleta. Vendar te tehnike večinoma niso dovolj točne in se ne prilagajajo spreminjajočemu se spletu (Lee, Hui & Fong, 2002).

Tehnološki in internetni gigant Google je leta 2009 naznanil nov operacijski sistem Chrome OS, ki je bil zamišljen tako, da bodo aplikacije in uporabnikovi podatki nameščeni v oblaku. Prvi prenosni računalnik, ki je uporabljal Chrome OS, je prišel na trg maja leta 2011 pod imenom Chromebook (Chrome OS, 2016). Že istega leta se je pokazalo, da Googlov operacijski sistem Chrome OS omogoča določene prednosti v primerjavi z obstoječimi OS, kot so na primer enostavno upravljanje, robustnost na viruse in hiter zagon naprav z nameščenim Chrome OS. Slabost operacijskega sistema Chrome OS pa so njegove omejitve glede varnosti na spletu. Leta 2011 je bila na Googlovi spletni trgovini Web Store na voljo samo ena razširitev za filtriranje pornografskih vsebin, ki pa je omogočala filtriranje vsebin le na eni napravi hkrati.

3.3.2 Ustanovitev podjetja

Lastnik podjetja Bloker, ki je strokovnjak za varnost na internetu, je predvideval, da bo izboljšava varnosti na spletu pri uporabi Chrome OS postala tržna niša. S pomočjo dveh študentov računalništva je začel avgusta leta 2011 razvijati rešitev – program oziroma t.i. razširitev (angl. *extension*), s katerim so brskalniku Chrome OS dodali novo funkcionalnost. Razširitev imenovano Bloker (angl. *Bloker Extension*, v nadaljevanju BE) so decembra 2011 ponudili na Googlovi internetni trgovini Chrome Web Store kot brezplačno dodatno funkcijo za filtriranje neželenih spletnih vsebin. Hkrati so razvili tudi program Bloker Manager (v nadaljevanju BM), ki je omogočal upravljanje ene ali več naprav z instaliranimi BE. Tudi BM je bil na Web Storu na voljo brezplačno.

Po tem, ko so se leta 2014 za oba izdelka začele zanimati šole, so programa izpopolnili in ju začeli prodajati. Šele takrat, leta 2014, sta bivša sodelavca, oba z dolgoletnimi izkušnjami s področja informacijske tehnologije, ustanovila istoimensko podjetje Bloker.

Ob ustanovitvi podjetja lastnika nista izdelala poslovnega načrta, niti si svojih idej ali predvidevanj o izdelku in trgu nista zapisala. Prva leta sta lastnika projekt Bloker v celoti financirala iz svojih sredstev, od leta 2015 pa se podjetje Bloker pretežno financira od prodaje svojih izdelkov, vendar v to financiranje ni všteto delo lastnikov. Kot ugotavlja Zwilling (2015), se kar 80% uspešnih zagonskih podjetij v semenski fazi financira z lastniškimi viri, pogosto so to zneski nižji kot 10.000 USD. Prednost zagona podjetja z omejenimi sredstvi je predvsem v tem, da se lastniki izognejo pritiskom investitorjev in njihovim kritikam (Zwilling, 2015). Podjetje Bloker je torej samozagonsko podjetje (angl. *boothstraping*), kar pomeni, da deluje z omejenimi finančnimi viri, brez zunanjih investitorjev (What is Bootstrap, 2016).

To, da je podjetje samozagonsko, pa ne pomeni, da je tudi vitko zagonsko podjetje. Samozagonsko in vitko zagonsko podjetje nista isti metodi. Pri obeh metodah gre za ustvarjanje maloporabnih zagonskih podjetij (angl. *low-burn startup*) z maksimalno izrabo obstoječih virov (Maurya, 2012). Po definicijah, ki zagonska podjetja opredeljujejo kot hitro rastoča (Graham, 2012; Sutton 2000), se podjetje Bloker ne uvršča med zagonska podjetja, saj je od prvega testnega izdelka, ki so ga ponudili v spletni trgovini, minilo že pet let, podjetje pa še ni doseglo hitre rasti. Po drugi strani pa ima podjetje zaradi inovativne naravnanosti, možnosti rasti v prihodnosti (Damodaran, 2012) in možnosti prodaje na globalnem trgu (Blank, 2007), še vedno vse značilnosti zagonskega podjetja.

3.4 Bloker rešitve

3.4.1 Bloker razširitev

Podjetje Bloker je videlo tržno nišo v tem, da razvije razširitev za filtriranje nezaželenih vsebin, ki bi omogočala filtriranje na več napravah hkrati. Osnovna ideja je bila razvoj spletne aplikacije, s katero bi upravljali nastavitve filtrov na nivoju podjetij ali šol, ki so se v tistem času začele opremljati s Chrombooki. Filtriranje neželenih vsebin na nivoju omrežnih naprav se je v šolah in podjetjih že uporabljalo. Večinoma so storitev izvajali ponudniki telekomunikacijskih storitev. Filtriranje se je izvajalo s pomočjo spletnih prehodov (web gateway). Vendar je bil ta način filtriranja učinkovit, ko spletne strani še niso bile kriptirane.
Filtriranje neželenih spletnih vsebin, ki ga je razvil Bloker, je prilagojeno računalništvu v oblaku. Prednosti so enostavnejše upravljanje, performančne prednosti in funkcionalnosti (npr. filtriranje YouTube vsebin, možnosti posebnih nastavitev za določenega uporabnika (user awernes), kot so na primer drugačne nastavitve filtriranja vsebin v šoli in izven šole. Prednost Blokerjeve rešitve je tudi v tem, da je filtriranje v oblaku vedno aktivno, ne glede na ponudnika internetne povezave.

Osnovni izdelek je Bloker razširitev (BE) za spletni brskalnik Chrome in operacijski sistem Chrome OS. BE se uporablja na Chrombookih in omogoča filtriranje spletnih strani

rangiranih v devetinsedemdeset kategorij, kot so na primer orožje, droga, nasilje, pornografija, ki so določene s pomočjo Bloker storitev v oblaku. Bloker razširitev omogoča:

- filtriranje po spletnih kategorijah
- filtriranje po spletnih naslovih
- filtriranje po ključnih besedah
- filtriranje aplikacij na spletnih brskalnikih
- filtriranje YouTube video vsebin (vsak video je kategoriziran),
- filtriranje po YouTube kanalih (npr. omogočanje gledanje samo iz šolskega kanala)

Do leta 2014 je BE imela več kot sto tisoč uporabnikov, podjetje pa ni imelo poslovnega modela, s pomočjo katerega bi iz brezplačne aplikacije naredili prodajni izdelek.

Za vse razširitve je značilno, da jih napredni uporabnik lahko onemogoči. Za ta namen so sistem filtriranja nadgradili z Bloker Managerjem. Bloker Manager (BM) je spletna aplikacija za upravljanje nastavitev filtrov Bloker razširitev. BM uporabljajo administratorji (upravljavci IT v šolah, starši doma), ki lahko za vsako Googlovo napravo Chromebook posebej nastavijo filtriranje internetnih strani. BM omogoča dostop do analitičnih poročil o uporabi spletnih strani. Prvi BM je imel vse lastnosti NSP – uporabniki so lahko testirali njegovo osnovno funkcionalnost, ni pa še bil optimiziran za uporabo v šolah, saj je administrator moral vpisati ključ v vsako napravo posebej. Je bil pa BM prva rešitev za filtriranje internetnih vsebin z možnostjo centralnega upravljanja. Podjetje Bloker je spremljalo aktivnost uporabnikov BM s pomočjo Google analitike. Leta 2012 so razvoj BM opustili, saj je kazalo, da Google s prodajo Chromebookov ne bo uspel in da izdelek ne bo imel potencialnega trga.

Maja 2014, po Googlovem uspehu na trgu izobraževanja, so se za BM začele zanimati šole. Podjetje Bloker je prejel zahtevke za ponudbo iz nekaj ameriških šol. Do novembra 2014 so v podjetju BM dopolnili z naslednjimi funkcionalnostmi:

- Integrirali so ga z Googlovo spletno aplikacijo Chrombook Managment Console, s katero administratorji upravljajo nastavitve na več tisoč Chrombookih hkrati.
- Dodali so licenčno – plačilno rešitev. Sistem je spremljal število priključenih uporabnikov in tako je bilo omogočeno podeljevanje licenc – osnovna enota za prodajo.
- BM so integrirali z internetnimi plačilnimi sistemi kot so PayPal in Googlewall.
- Zaradi zahtev ameriških šol, ki zaradi davčne politike lažje kupujejo storitve od ameriških podjetij, je bila leta 2014 ustanovljena tudi ameriška podružnica Bloker.

Izpopolnjen izdelek so imenovali Bloker Manager Education Everywhere (v nadaljevanju BMEE) in je predvsem namenjen uporabi v šolah, ki so opremljene s Chromebooki in Googlovimi tablicami. BMEE lahko uporabljajo šolski administratorji, ki so zadolženi za delovanje sistema na ravni šole, učitelji lahko v času trajanja pouka dovolijo določene vsebine ali celo omogočijo posameznim učencem dostop do drugačnih vsebin, kot so sicer na programu. Starševski portal pa omogoča, da spletne vsebine učencem določajo starši, medtem ko so doma. Od februarja 2015 do junija 2016 je Bloker prodal približno šestdeset tisoč licenc petindvajsetim, večinoma ameriškim šolam. Rast prodaje BMEE je torej odvisna od Googlovega uspeha s programom Google for Education (Google for Education, https://www.google.com/edu/), ki generira prodajo Chromebookov.

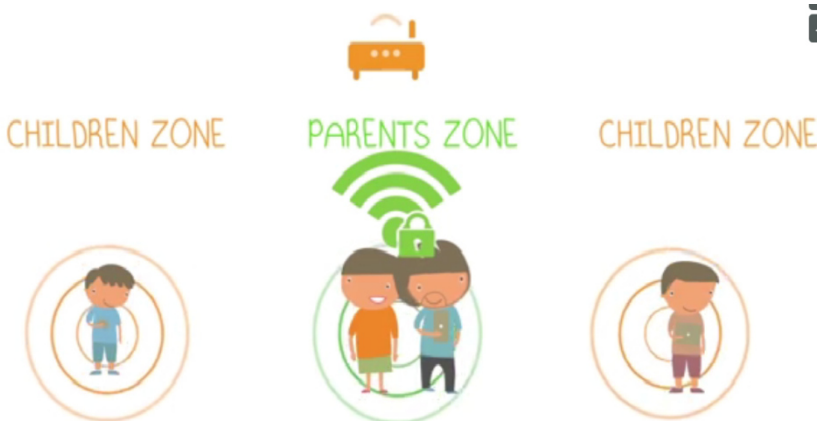
Medtem ko leta 2011 poleg BM ni bilo na trgu drugih ponudnikov celotnih sistemov filtriranja spletnih vsebin, sta bila leta 2014 na trgu že dva resna ponudnika: GoGuardian in Securly, ki sta pravočasno opazila dogajanje na trgu Chromebookov in tako pri prodaji prehitela Bloker. GoGuardian si je razvoj rešitve širitev na trgu omogočil s pomočjo investitorja tveganega kapitala, kar pa se sedaj odraža na višji ceni njihovih storitev, saj želi investitor hitro povračilo sredstev, kar pa na trgu izobraževanja, ki ima tudi v Ameriki omejena sredstva, ni lahko.

3.4.2 Bloker usmerjevalnik

Zaradi začetnega Googlovega neuspeha pri prodoru na trg osebnih računalnikov je podjetje opustilo razvoj BM in tehnologijo filtriranja spletnih vsebin uporabilo za razvoj usmerjevalnika namenjenega filtriranju nezaželenih internetnih vsebin. Iz rešitve v oblaku, ki se takrat ni zdela prodajno zanimiva, so ponovno prešli na napravo, torej strojno rešitev. Bloker usmerjevalnik (angl. *Bloker Router*, v nadaljevanju BR) je namenjen naprednejšim uporabnikom interneta kjerkoli po svetu, ki iščejo rešitev za filtriranje spletnih vsebin. V osnovi pa je bil namenjen starševskemu nadzoru spletnih vsebin, do katerih dostopajo otroci z mobilnih naprav ali računalnikov. Je dvokanalni brezžični usmerjevalnik s katerim lahko starši z enega mesta onemogočajo dostop do nezaželenih vsebin preko naprav, ki jih uporabljajo otroci, medtem pa lahko sami nemoteno dostopajo do vseh spletnih vsebin (Slika 14).

Usmerjevalnik so razvijali več kot eno leto in ga dali na trg novembra leta 2014, takoj, ko so mislili, da ustreza kriterijem NSP. Leta 2014 na trgu ni bilo dosti izdelkov za starševski nadzor nad spletnimi vsebinami. Izid novega izdelka so najavili na bazo več kot 7000 uporabnikov BE, ki so izkazali zanimanje tudi za oglaševani usmerjevalnik, vendar niso dosegli zaželenega odziva, saj se je manj kot procent uporabnikov zanimalo za plačljiv izdelek. BR je mogoče kupiti pri spletnem trgovcu Amazon, uporabniki pa jih lahko naročijo tudi neposredno na Blokerjevi spletni strani. Vendar samo s tema dvema prodajnima kanaloma ne dosegajo zadostne prodaje.

Slika 14: Ekranska slika Bloker usmerjevalnik na spletni strani podjetja Bloker



Vir: Bloker Router: The Ultimate Parental Control, 2016

Po drugi strani je podjetje z BR nepričakovano uspelo na drugem trgu. V tehnologiji dvokanalnega usmerjevalnika, ki filtrira spletne vsebine, je dodano vrednost prepoznalo južnoameriško avtobusno podjetje, ki na svojih avtobusih nudi brezžično internetno povezavo. Zaradi prevelike porabe pasovne širine, se je avtobusno podjetje odločilo, da bo na avtobusih onemogočil pregledovanje video vsebin, medtem ko so vsi ostali tipi vsebin na razpolago potnikom. Zaradi cene telekomunikacijskih storitev, so tudi omejili količino prenosa podatkov. Razvijalci so BR za potrebe avtobusnega podjetja prilagodili v dveh mesecih, izdelek je uspešno prestal testiranje in od leta 2014 je avtobusno podjetje kupilo več BR kot jih je podjetje Bloker prodalo za njihov osnovni namen.

3.5 Analiza stanja v podjetju s predlogi izboljšav

3.5.1 Metode dela uporabljene v praktičnem delu naloge

Podatke o trenutnem stanju v podjetju sem pridobila z metodo opazovanja dela v podjetju, s pomočjo pregleda orodij, ki jih uporabljajo pri delu in s pomočjo delno strukturiranih intervjujev. Individualni nestrukturirani intervju je ena od glavnih metod zbiranja podatkov pri kvalitativnih raziskavah. Nestrukturirani intervju omogoča, da pojav raziskovanja razložimo bolj osredotočeno in poglobljeno (Legard et al., 2003). Intervjuje z vsemi udeleženci v razvoju in slovenskim lastnikom sem opravila med 4. in 8. julijem 2016. Pogovor z lastnikom iz Francije pa je potekal preko Skypa teden dni kasneje, 15. julija. S pomočjo nestrukturiranega intervjuja sem od vseh udeležencev pri projektu Bloker želela izvedeti, v kolikšni meri obravnavane metode uporabljajo pri svojem delu.

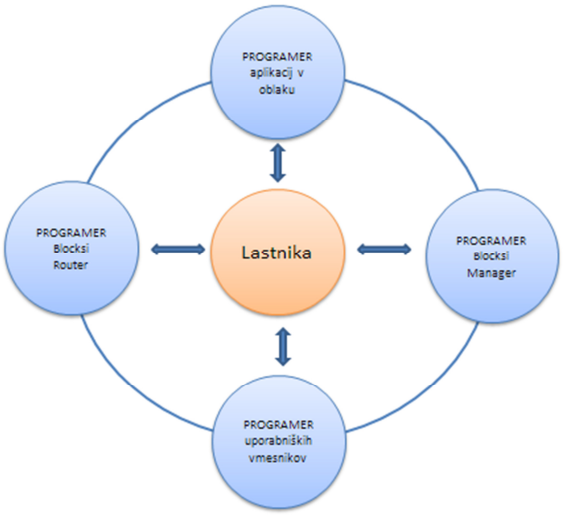
V Prilogi 1 navajam vprašanja, ki so bila postavljena posameznim intervjuvancem. Vprašanja so razdeljena na štiri sklope, ki se nanašajo na podatke o intervjuvancih, agilne metode programiranja, načela vitkega razvoja PO in uporabo metod strank in vitkega zagonskega podjetja.

Uporabo programskih orodij, ki jih uporabljajo pri delu, so mi intervjuvanci pokazali in razložili med samim intervjujem.

3.5.2 Organizacija dela in komunikacija med zaposlenimi

Delo v podjetju Bloker poteka deloma na daljavo. Lastnika si delita razvoj in prodajo izdelkov. Lastnik v Franciji, ki je zadolžen za prodajo, zbira povratne informacije in zahteve kupcev in uporabnikov ter na podlagi tega daje razvijalcem navodila za izboljšave produktov. Lastnik v Sloveniji dela s tremi razvijalci v isti pisarni, kar predstavlja neke vrste sodela (angl. *coworking*). Kljub temu, da njegovo redno delo ni povezano z zagonskim podjetjem, se lahko še vedno povezuje s sodelavci na projektu Bloker in jim je na voljo pri reševanju problemov. Od treh razvijalcev v pisarni sta dva zadolžena vsak za svoj projekt – Bloker Manager in Bloker usmerjevalnik. Oba komunicirata z obema lastnikoma in s tretjim razvijalcem, ki je odgovoren za grafiko, to je izgled uporabniških vmesnikov, ne pa tudi za funkcionalnost izdelkov. Za analitiko na strežnikih skrbi četrti razvijalec, ki dela od doma. Tudi on pri svojem delu komunicira z lastnikom in razvijalcema BR in BM. V glavnem gre za dvosmerno vertikalno komunikacijo med lastnikoma in vsakim od razvijalcev. Razvijalca odgovorna za BM in BR pa komunicirata tudi z razvijalcema, ki sta odgovorna za grafiko in analitiko. Če sistem komuniciranja upodobim z grafom kot Grabnar (1992), gre za križni sistem komunikacije, kjer vodji pošiljata sporočila vsakomur in vsakemu posebej, v kombinaciji s krožnim komuniciranjem med zaposlenimi, kot je prikazan na Sliki 15.

Slika 15: Prikaz smeri komuniciranja v podjetju



Vir: Povzeto po B. Grabnar, Retorika za managerje, 1992, str. 29

Za vitko proizvodnjo in vitka zagonska podjetja je bolj značilna izrazito horizontalna komunikacija (Hussein, 2015). Komunikacija v pisarni poteka večinoma ustno, komunikacija s sodelavci izven pisarne pa ustno in pisno preko Skypa, ki je pomembno

orodje za komunikacijo v majhnem timu, ki sodeluje na daljavo. Elektronske pošte za komuniciranje med sabo ne uporabljajo. Za komuniciranje o nalogah in za sledenje delu uporabljajo temu namenjena programska orodja.

3.5.3 Poznavanje agilnih metod programiranja in vitkih pristopov v zagonskem podjetju

Za ugotavljanje poznavanja in uporabljanja agilnih metod programiranja, metode vitkega razvoja in vitkih pristopov, ki se uporabljajo pri izgradnji zagonskega podjetja, sem uporabila nestrukturiran intervju. S pomočjo nestrukturiranega intervjuja sem od vseh udeležencev pri projektu Bloker želela izvedeti:

- Ali udeleženci poznajo agilne metode programiranja in če so jih v razvoju izdelkov uporabili.
- Če poznajo načela vitkega razvoja PO in katera od teh načel uporabljajo pri delu
- V kakšni meri so seznanjeni z metodama razvoja strank in vitkega zagonskega podjetja in v kolikšni meri sta bili metodi uporabljeni pri razvoju podjetja.

S pomočjo intervjujev sem prišla do naslednjih zaključkov:

- Razvijalci, ki še študirajo ali pa so študij zaključili pred kratkim, so bili z agilnimi metodami programiranja seznanjeni tekom študija. Agilnih metod programiranja v okviru študija niso prakticirali. Pri delu za Bloker so začeli uporabljati metodo Scrum, vendar so z njeno uporabo po šestih mesecih prenehali, ker nihče ni nadzoroval procesa. Njihova ugotovitev je bila, da v sedanji sestavi samo z dogovorom o nalogah in času izvedbe, hitreje pridejo do cilja. Metod, ki se uporabljajo za iskanje ustreznega poslovnega modela zagonskih podjetij, razvijalci niso poznali. Samo eden od njih je slišal za izraz vitko zagonsko podjetje, vendar ni poznal njegovega pomena. Nadaljnjih vprašanj o metodah razvoja strank in vitkem zagonskem podjetju razvijalcem nisem zastavljala.
- Slovenski lastnik, ki vodi razvoj izdelkov, agilnih metod programiranja in vitkega razvoja PO ne pozna, niti jih ni nikoli uporabljal pri svojem delu. Z metodo razvoja strank in knjigo »The Four Steps to the Epiphany« se je seznanil leta 2011, ko se je začel ukvarjati s projektom Bloker, vendar metode pri odkrivanju in potrjevanju strank ni dosledno uporabljal. Knjigo The Lean Startup je prebral kmalu po izidu, učil pa se je tudi izdelave poslovnega okvirja podjetja. Leta 2014, ko sta s kolegom ustanovila podjetje, sta tudi izdelala poslovni okvir za Bloker usmerjevalnik. Zaradi pomanjkanja časa je metodo opustil in ni načrtno sledil ciklu ustvari-meri-spoznaj.
- Povratne informacije od kupcev večinoma spremlja, prioritizira in posreduje razvojno ekipi lastnik v Franciji. Hkrati tudi vodi prodajo. Globalno podjetje, v katerem je zaposlen, uporablja agilne metode programiranja v razvoju izdelkov, zato je z njimi seznanjen in jih je tudi uporablja pri delu. Metodo Scrum je želel uvesti v razvoj

Bloker izdelkov, vendar zaradi pomanjkanja časa nihče ni nadzoroval sprintov (Sprint Master). Tako je Scrum metoda za razvijalce postala nepregledna, zato so jo nehali uporabljati. Z metodo vitkega zagonskega podjetja in z vitkim poslovnim okvirjem ga je seznanil solastnik. Metod sam ni proučil in jih pri delu z Blokerjem vsaj zavestno ne uporablja. Posebne metode za iskanje poslovnega modela podjetja se mu ne zdijo potrebne. Svoje znanje o poslovnem razvoju prenaša v zagonsko podjetje iz globalnega podjetja, kjer je zaposlen.

3.5.4 Uporaba agilnih metod oziroma orodij za spremljanje procesa razvoja in podporo uporabnikom

Kot ugotavljata Coleman in Connor (2008) je za veliko malih in zagonskih podjetij, ki se ukvarjajo z razvojem PO, glavna težava vzpostavljanje kontrole in struktur za pravilno managiranje aktivnosti pri razvoju PO. Proces v bistvu opisuje način, kako organizacija razvija PO in podporne storitve kot je na primer dokumentacija.

Prvi princip agilnega manifesta sicer pravi, da imajo ljudje in interakcije med njimi prednost pred procesi in orodji. Vendar tako kot narašča uporaba agilnih metod, narašča tudi trg programskih orodij za spremljanje agilnega razvoja PO (Goth, 2009).

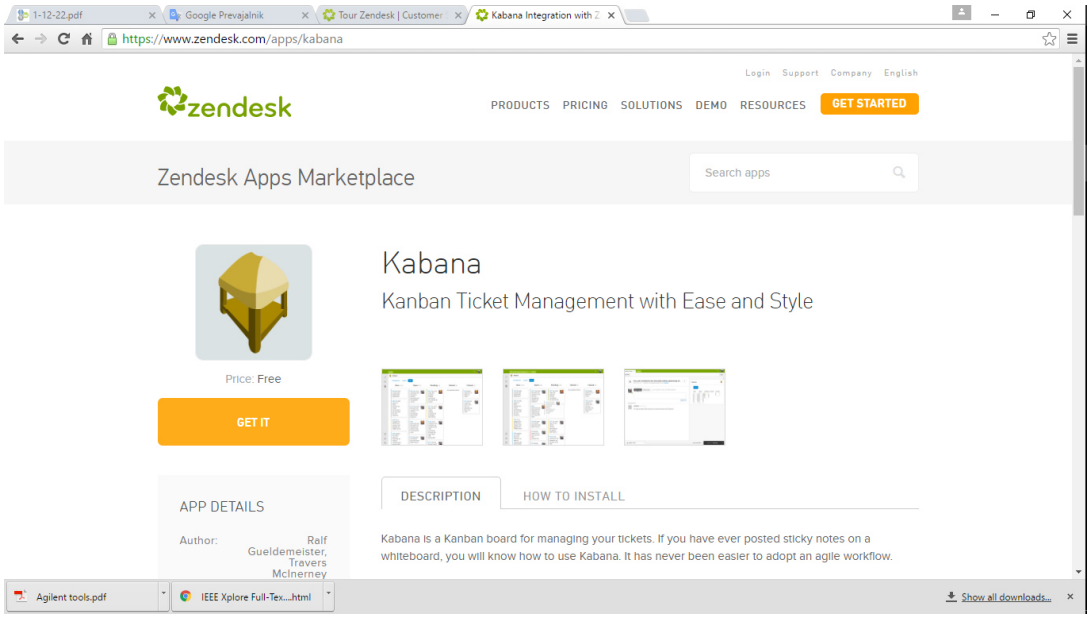
V Blokerju uporabljajo za managiranje procesov programski orodji JIRA in Zendesk.
Naloge glede razvoja izdelkov večinoma razdelita lastnika, ki se z razvijalci dogovorita o razvoju nove funkcionalnosti ali odpravah napak (angl. *bug*) v programu, načinu reševanja teh problemov (v kolikor je to potrebno) in o roku, v katerem naj bi bilo delo opravljeno. Naloga oziroma opravilo se vpiše v programsko orodje JIRA, ki ga uporabljajo vsi v podjetju Bloker. JIRA je orodje za projektno managiranje in je prvenstveno namenjena agilnim timom. Program omogoča (Atlassian – JIRA Software, 2016):

- Načrtovanje: razdeljevanje nalog na podlagi povratnih informacij od uporabnikov, načrtovanje sprintov po Scrum metodi, stalno komunikacijo med udeleženci.
- Sledenje: razpravljanje o poteku dela in prioritiziranje nalog, nudi pregled nad opravljenim delom v realnem času.
- Poročanje: izboljšanje učinkovitosti tima s pomočjo vizualnih podatkov

Podpora uporabnikom poteka na principu vlečenja, kar omogoča programsko orodje Zendesk, ki je namenjeno upravljanju odnosov s strankami. Zendesk omogoča avtomatsko managiranje toka dela ter komunikacijo s kupci. S pomočjo Zendeska tim spremlja probleme, ki jih imajo uporabniki Bloker Managerja in Bloker usmerjevalnika. Uporabniki Bloker izdelkov problem sporočijo na elektronski naslov Blokerjeve podpore, od koder se avtomatsko prekopira v Zendesk, kjer dobi številko kartice (angl. *ticket*). Status kartic vidijo vsi zaposleni. V kolikor so potrebne dodatne informacije od uporabnikov ima kartica

status *v teku,* ko je problem rešen, se označi status »zaključeno«. Od januarja 2016 program Zendesk omogoča tudi managiranje vpisanih kartic po kanban metodi (Slika 16).

Slika 16: Ekranska slika programskega orodja Zendesk



Vir: Zendesk – Integrations & Apps, 2016

Tako kot JIRA tudi Zendesk omogočata delo, ki ustreza načelom agilnih metod: tesno delo s strankami in kratka zanka povratnih informacij ter osredotočenost na to, da kupcu dobavijo vrednost (Bosch et al., 2013).

3.5.5 Elementi vitkega razvoja programske opreme, ki so izraženi v podjetju

Bloker je primer mikro podjetja v katerem imajo zaposleni dokaj jasno razdeljene naloge. Vitkega razvoja PO in njegovih načel ne poznata niti lastnika, niti razvijalci. Se pa tudi zaradi uporabe orodij za agilen razvoj PO, nekatera od načel vitkega razvoja PO nenačrtno uporabljajo pri delu v podjetju.

Predpogoj za odpravo izgub je videnje izgub, ki je prvi korak na področju vitkega mišljenja. Podjetje Bloker ima sicer zelo omejene človeške vire, vendar se pri razvoju BM in BR kažeta možnosti izgube zaradi delno opravljenega dela in pogostega preklapljanja med nalogami, do katerega prihaja zaradi menjave prioritet dela. Kar se dodatnih funkcionalnosti izdelka tiče, se je v preteklosti že zgodilo, da so tako pri BM kot pri BR razvili funkcionalnosti, ki jih uporabniki niso sprejeli, vendar se te, v takem primeru nepotrebne funkcionalnosti, pri posodabljanju z izdelka odstranijo. Kadar se zaradi spremenjene funkcionalnosti nadgrajuje tudi izgled izdelka, to je grafični vmesnik, se lahko zgodi, da se pri spreminjanju delčki kode nabirajo, vendar se tak del programa kasneje optimizira. Glede na majhnost podjetja izgub zaradi čakanja še ni, ker na vsakem

izdelku dela samo en razvijalec. Se pa zgodi, da razvijalca BR in BM včasih čakata, da razvijalec grafike zaključi s svojim delom.

Kljub sprotnim testiranjem, se napake pri razvoju PO dogajajo. Razvijalci za odkrivanje napak uporabljajo teste, ki jih razvijajo s svojega vidika - torej z vidika razvijalca, ne pa z vidika uporabnika. Lažje napake se hitro odpravijo, to pomeni v dveh do treh urah. Težje napake razvijalci odpravljajo nekaj dni. Napake v podjetju odkrijejo sami ali pa jih sporočijo uporabniki BR in BM.

Krepitev znanja je načelo, ki je zelo izraža pri delu v podjetju. Razvoj programske opreme se tudi v Blokerju izvaja v hitrih, ponavljajočih se ciklih, kjer se testira in izboljšuje v naslednjem ciklu. Tako se izboljšuje tudi kakovost izdelkov. V podjetju načrtno izboljšujejo zaznavno kakovost izdelkov – to da lahko uporabnik s čim manj koraki pride do želene funkcionalnosti izdelka.

Sočasnega razvoja PO ne uporabljajo, razvijejo eno varianto funkcionalnosti, jo testirajo in ovržejo ali sprejmejo. Ne razvijajo več alternativ PO hkrati.

Dobaviti, kakor hitro je mogoče oziroma redne in hitre izdaje nadgradenj PO izhajajo že iz agilnih orodij, ki jih uporablja tim. Orodji JIRA in Zendesk zagotavljata, da delo večinoma poteka po načelu vlečenja in da je načrt dela tudi vizualiziran. Je pa dobava Bloker usmerjevalnika odvisna tudi od dobaviteljev strojne opreme, ki pa niso tako zanesljivi.

Razvijalci lahko do določene mere sami odločajo, katera programska orodja bodo uporabljali pri svojem delu, večinoma pa se o tem kako bo opravljeno delo posvetujejo. Prav tako jim je jasen namen njihovega dela in tudi o dosegljivosti ciljev se pogovorijo z lastniki, kar jih tudi motivira pri delu. Večinoma pa razvijalci nimajo dostopa do kupcev, njihove zahteve jim posredujeta lastnika. Samo razvijalec grafičnih vmesnikov je komuniciral s kupci na sejmu, kjer so sodelovali kot razstavljavci in to izkušnjo opisuje izredno pozitivno.

3.5.6 Uporaba metode razvoja strank in metode vitkega zagonskega podjetja v primeru Bloker Manager in Bloker usmerjevalnik

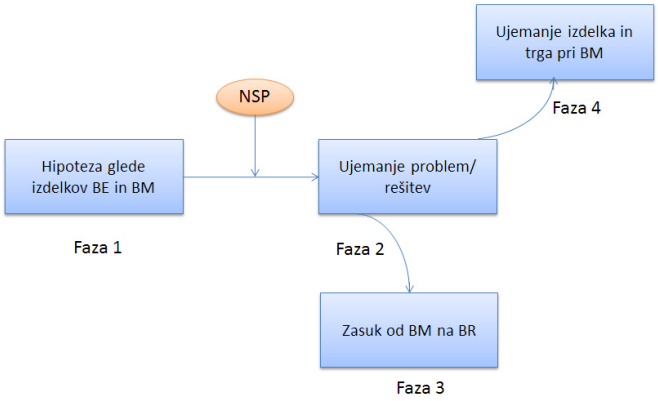
Maurya (2012) in Bosch et al. (2013) so ugotovili, da kljub temu, da je metoda vitkega zagonskega podjetja poznana in dobro dokumentirana metoda za razvoj podjetja, imajo praktiki težave pri prenašanju te metode v prakso (Maurya, 1012; Bosch et al., 2013).

Podobno se je zgodilo v podjetju Bloker. Metoda razvoja strank s postavljanjem in preverjanjem hipotez, se je lastniku zdela od začetka preveč kompleksna. Poznavanje konceptov metode vitkega zagonskega podjetja – ujemanje problema in rešitve, NSP, ujemanje izdelka in trga ter sukanje so uporabljali bolj intuitivno in ne po vrstnem redu,

kot je predviden v metodi. V primeru BE in BM je razvoj potekal v naslednjem vrstnem redu (Slika 17):

- Na osnovi poznavanja problematike neželenih spletnih vsebin, so problem predvideli in naredili Bloker razširitev, ki je bila NSP. NSP so naredili preden so preverili, ali ga kupci hočejo in če bodo pripravljeni zanj tudi plačat.
- Po tem, ko so BE in BM ponudili brezplačno, je število uporabnikov pokazalo, da je izdelek za trg zanimiv, ni pa bilo potrditve, če bodo kupci zanj tudi plačali. Podjetje ni bilo pripravljeno za predstavitev na trgu (cenovni model, možnost naročanja).
- Zasuk: razvoj BM so opustili, ker se trg Chromebookov ni razvijal. Posvetili so se razvoju Bloker usmerjevalnika.
- Do ujemanja izdelka in trga je pri BM prišlo šele na pobudo kupcev – izdelek so prilagodili zahtevam in ga naredili prodajnega. Testirali so ceno.

Slika 17: Faze pri razvoju Bloker Managerja po modelu vitkega zagonskega podjetja

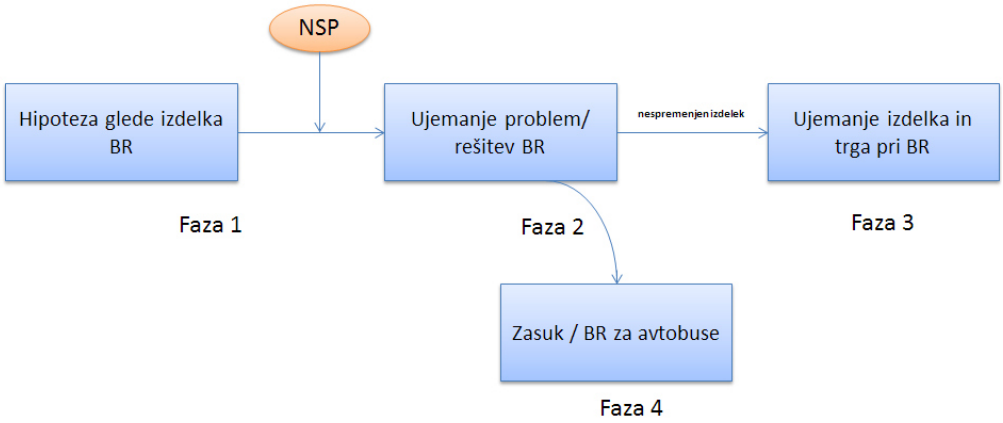


Vir: Povzeto po S. Blank, The Four Steps to The Epiphany, str. 42; A. Maurya, Delaj vitko, str. 8.

Razvoj BR je bolj sledil fazam, kot jih predvideva metoda vitkega zagonskega podjetja, vendar tudi v tem primeru do ujemanja izdelka in trga ni prišlo tam, kjer so pričakovali (Slika 18):

- S pomočjo brezplačne BE so preverili, če kupci želijo filtriranje internetnih vsebin ter BR najavili bazi uporabnikov
- V primeru starševskega nadzora, za katerega so razvili izdelek, ni prišlo do takega ujemanja produkt/ trg kot v primeru avtobusnega podjetja, ki je samo prepoznalo BR kot rešitev za svoje potrebe
- Podjetje je prilagodilo izdelek uporabniku, vendar so osnovno verzijo BR še vedno ohranili in jo prodajajo. V tem primeru gre samo za delni zasuk zaradi potreb kupca.

Slika 18: Faze pri razvoju Bloker usmerjevalnika po modelu vitkega zagonskega podjetja



Vir: Povzeto po S. Blank, The Four Steps to The Epiphany, str. 42; A. Maurya, Delaj vitko, str. 8.

3.5.7 Predlog izboljšav in izdelava vitkega poslovnega okvirja za Bloker Manager

V razvoju izdelkov, za katerega podjetje Bloker uporablja programska orodja za načrtovanje, sledenje in poročanje o procesu, podjetje uporablja metode, ki so v skladu z agilnimi metodami, nekatere pa tudi z načeli vitkega razvoja. Predlagam, da se v podjetju seznanijo z aktivnostmi, ki pri razvoju PO predstavljajo izgube, da jih bodo v prihodnje, ko bo organizacija podjetja bolj kompleksna, lahko hitreje prepoznali in odpravili. H krepitvi motivacije zaposlenih in timskemu delu bi verjetno prispevala tudi možnost, da razvijalci vsaj občasno komunicirajo s kupci.

Podjetje Bloker pa nima nobenih zapisov o uporabi vitkih pristopov ali drugih načrtov pri izgradnji podjetja samega. Glede na to, da lahko podjetje Bloker pet let od začetka razvoja ideje pokrije večino stroškov delovanja podjetja (ne pa tudi stroške dela lastnikov), lahko ugotovimo, da so tako BM kot BR dosegli ujemanje izdelka in trga. Ob nakupu obeh izdelkov uporabnik dobi eno oziroma dvoletno licenco za uporabo storitev. Merilo za ujemanje izdelka in trga so trenutno samo prihodki od prodaja BR in BM ter licenc. Podjetje nima določenih natančnejših kazalnikov za merilo napredka. Informacije o prodaji in obnovi licenc so skupaj z ostalimi podatki o kupcih vnesene v program za obvladovanje odnosov s kupci, vendar tudi če so uporabne, niso dostopne in pregledne. Obnovitev licenc za storitve pa tudi kaže na zadrževanje kupcev.

Oba lastnika imata veliko informacij o kupcih, prodajnih kanalih in konkurenci, vendar to vedenje ni nikjer zapisano. Zagonsko podjetje po definiciji deluje v skrajno negotovih pogojih, zato se tudi ugotovitve o stanju, v katerem je podjetje v določenem trenutku, spreminjajo. Ker te ugotovitve niso zapisane, je težko imeti dolgoročen pregled nad tem, kako se trg spreminja in kako podjetje napreduje. Zato sem predlagala izdelavo vitkega okvirja, ki je pomoč pri iskanju poslovnega modela, določitvi prioritet in neprekinjenemu učenju (Maurya, 2012). Bistvena prednost vitkega okvirja je, da je izdelan hitro. Sama izdelava vitkega okvirja pa je tudi povod za pogovor o nadaljnjih korakih v razvoju

izdelka, o kazalnikih za uspešnost in o prodajnih kanalih. V dnevni komunikaciji se lastnika in udeleženci pogovarjajo o nalogah in prioritetah glede funkcionalnosti izdelka. Pri tem je oteževalna okoliščina tudi fizična razdalja med lastnikoma, saj do pogovorov o strategiji ne more priti spontano kot na primer ob jutranji kavi. S pomočjo predloge iz knjige Delaj vitko (Maurya, 2014) smo v podjetju izdelali vitki okvir za Bloker Manager (Priloga 2), ki prikazuje aktualno stanje izdelka BM na trgu. Ker BM že ima svoje kupce, je več časa pri izdelavi vitkega okvirja bilo namenjenega razmisleku o prodajnih kanalih in ključnih kazalnikih. Iskanje novih prodajnih kanalov je trenutno prioriteta pri BM. Če bi vitki okvir za BM izdelali pred štirimi leti, bi verjetno več časa porabili za opisovanje problema in rešitve– kategorij vitkega okvirja, ki opisujejo sam izdelek.
Glede na to, da je podjetje Bloker doseglo ujemanje izdelkov in trgov, bi ji sedaj morala slediti faza rasti. Lastnika se zavedata, da rasti ne moreta financirati samo iz prihodkov iz prodaje, hkrati pa nista najbolj naklonjena vlagateljem tveganega kapitala, ker se je pri konkurenci izkazalo, da naložba ni bila poplačana tako hitro, kot si je želel vlagatelj. Zato bi bilo potrebno ne samo razmisliti o možnostih financiranja, ampak se o različnih možnostih tudi dejansko podučiti in ugotoviti, kateri viri bi bili za podjetje najprimernejši.

SKLEP

V magistrski nalogi sem raziskovala razvoj vitke miselnosti od Toyotinega proizvodnega sistema do trenutno zelo popularnega modela vitkega zagonskega podjetja ter uporabo načel teh metod v zagonskem podjetju, ki se ukvarja z razvojem programske opreme. Namen naloge je bil, da na podlagi analize stanja v podjetju, predlagam možnost izboljšav z uporabo vitkih pristopov v razvoju PO in pri oblikovanju poslovnega modela podjetja. V teoretičnem delu naloge sem obravnavala literaturo s področja vitke proizvodnje, vitkega razvoja, agilnih metod, vitkega razvoja PO ter metod razvoja strank in vitkega zagonskega podjetja. V vseh metodah se pojavlja načelo, da kupec opredeljuje dodano vrednost izdelka ali storitve. To načelo je osnova za analize procesov in izdelkov in prepoznavanje izgub. Izgube so pri vitkem razvoju PO dokaj natančno prevedene iz Toyotinega proizvodnega sistema. Tako kot v TPS, se tudi pri vitkem razvoju PO prepoznavanje izgub v procesih začne z izdelavo toka vrednosti. Metodi razvoja strank in vitkega zagonskega podjetja izgube obravnavata bolj kot izgubo časa za razvoj izdelkov ali storitev, ki jih kupci ne potrebujejo.

Krog učenja s sistemom stalnih izboljšav je prav tako eno od osnovnih načel vseh metod in povsod so v ta krog vključeni tudi kupci. Pri vitki proizvodnji in vitkem razvoju je cikel učenja vgrajen v sistem nenehnih izboljšav. Pri agilnih metodah in vitkem razvoju PO je pomikanje po krogu učenja izvedeno s pomočjo iteracij, ki so delujoči prirastki PO ponujeni kupcu v testiranje. Podobno se pri metodah razvoja strank in vitkega zagonskega podjetja povratne informacije od kupcev zbirajo na osnovi testiranj najosnovnejših izdelkov. V vseh metodah se izdelek ali storitev v naslednjem krogu izboljšata na osnovi

spoznanj. Vendar metode vitkega razvoja PO, metoda razvoja strank in metoda vitkega zagonskega podjetja ne omenjajo kulture odličnosti in teženja k popolnosti, ki sta značilni za TPS in vitek razvoj.

Vsem metodam je skupno tudi načelo vlečenja. Pri TPS je načelo vlečenja upoštevano pri toku materiala, medtem ko je pri vitkem razvoju in agilnih metodah to načelo upoštevano pri pretoku informacij in navodil za delo. Pri vitkem zagonskem podjetju pa so osnova za uporabo načela vlečenja zahteve kupcev, na osnovi katerih je potrebno narediti izboljšave.

Na splošno pa je v metodi razvoja kupcev in vitkega zagonskega podjetja preneseno manj načel iz TPS in vitkega razvoja, ki se nanašajo na ljudi: poleg že prej omenjene izgradnje kulture, ki podpira odličnost, sta to še pomembnost komunikacije z vizualnim pristopom, ki mora zajemati vse ravni organizacije ter prilagajanje tehnologije ljudem in procesom.

V praktičnem delu sem analizirala način dela v zagonskem podjetju, ki razvija PO z namenom, da bi proučila, ali bi bila vpeljava katerega od vitkih pristopov dobra za razvoj podjetja. Ugotovila sem, da agilne metode in nekatera načela vitkega razvoja PO v podjetju, čeprav ne vedno načrtno, že uporabljajo. Uporaba agilnih metod v podjetju Bloker je posledica prenosa znanja in izkušenj, ki sta jih lastnika pridobivala v večjih podjetjih, pa tudi uporabe agilnih programskih orodij. S pomočjo programov, ki omogočajo timom, ki razvijajo programsko opremo, da hitro in pogosteje dajo izdelke na trg, je delo v zagonskem podjetju bolj organizirano in poteka po načelu vlečenja. Uporaba teh programskih orodij je v podjetju omogočila vzpostavitev procesov, pri katerih so človeški viri bolje izkoriščeni tako pri razvoju izdelkov kot pri podpori uporabnikom. V mikro podjetju Bloker, so v okviru programskega orodja Jira začeli uporabljati tudi agilno metodo Scrum, vendar so metodo po približno pol leta opustili. Metodo so opustili, ker v podjetju niso imeli Scrum vodje (angl. *Scrum Master*), ki bi vodil proces in samo izvajanje metode. V podjetju, kjer trenutno razvijalci večinoma delajo vsak na svojem projektu, ni potrebno toliko medsebojnih usklajevanj, zato je obvladovanje same metode Scrum udeležencem vzelo več časa, kot je uporaba metode skrajšala proces razvoja. Metoda Scrum je bolj uporabna v podjetjih z več razvijalci, ki delajo na različnih delih istega projekta in katerih delo je medsebojno bolj prepleteno. Pri uporabi programskih orodij za agilni razvoj pa morajo biti v podjetju pozorni, da se ohrani prva vrednota agilnega manifesta in sicer, da imajo posamezniki in interakcije med njimi prednost pred procesi in orodji. Programsko orodje ne more nadomestiti dogovarjanja o prioritetah pri delu in rešitvah. Prav tako se morajo v podjetju zavedati, da samo programsko orodje ni dovolj za izboljšanje razvoja. Uporaba agilnih metod je v podjetjih, ki se ukvarjajo z razvojem programske opreme, prvi korak na poti k vitkemu razvoju. Vendar pogosto ostane samo pri omejeni uporabi agilnih metod. Premik, ki vodi v trajnostne spremembe in k vitkemu razvoju programske opreme, zahteva spremembe v razmišljanju o konceptih, principih in miselnih orodjih, na osnovi katerih se potem uporabljajo tehnike in agilna orodja, ki omogočajo izvedbo teh principov.

Na osnovi poznavanja metode razvoja strank in vitkega zagonskega podjetja so v podjetju izdelali NSP in preverili ujemanje izdelka in trga. Vendar o uporabi in uspešnosti teh metod ni sledljivosti. O tem, kako so bili prvi izdelki načrtovani in v kakšnem času razviti, lastnika pripovedujeta po spominu. Dosledno sledenje tem metodam zahteva dosti časa, ki je v podjetju, ki dela z omejenimi viri, ponavadi najbolj omejen vir. Predvsem sta ti metodi mišljeni kot nenehen proces, ki bi ga bilo dobro vključiti tudi v podjetje Bloker. Tudi zagonsko podjetje potrebuje kazalnike za meritev napredka podjetja. Pomembno je njihovo redno spremljanje in dostopnost teh podatkov vsem udeležencem v podjetju. Samo s pomočjo podatkov, ki so osnovani na dobro izbranih kazalnikih, lahko v podjetju sprejemajo dobre strateške odločitve. Omejitev metod razvoja strank in vitkega zagonskega podjetja pa je odločanje o ohranitvi smeri, ki si jo je zastavilo podjetje ali zasuku v drugo smer, ki temelji na ugotavljanju stanja v katerem se podjetje trenutno nahaja. Pri tem bi lahko bolj vizionarske ideje, pri katerih je večja vrzel med vizionarji in pragmatiki, prehitro opustili, ker ne bi dovolj hitro prišlo do ujemanja produkta in trga.

LITERATURA IN VIRI

1. Amit, R., & Zott, C. (2001). Value creation in e-business. *Strategic management journal*, *22*(6-7), 493-520.
2. Aspara, J., Hietanen, J., & Tikkanen, H. (2010). Business model innovation vs replication: financial performance implications of strategic emphases. *Journal of Strategic Marketing*, *18*(1), 39-56.
3. Atlassian. (b.l.). JIRA Software. Najdeno 21. julija 2016 na spletnem naslovu https://www.atlassian.com/software/jira
4. Bach, J. (1998). Microdynamics of Process Evolution, *IEEE Computer,* February 111-113
5. Ballard, G. (2000). Positive vs negative iteration in design. V *Proceedings Eighth Annual Conference of the International Group for Lean Construction, IGLC-6, Brighton, UK* (pp. 17-19).
6. Ballard, G. & Howell, G. (2003). Lean project management. *Building Research & Information*, *31*(2), 119-133.
7. Ballé, F., & Ballé, M. (2005). Lean development. *Business Strategy Review*,*16*(3), 17-22.
8. Basil, V. R., & Turner, A. J. (1975). Iterative enhancement: A practical technique for software development. *IEEE Transactions on Software Engineering*, (4), 390-396.
9. Beck, K. (1998). Extreme programming: A humanistic discipline of software development. In *International Conference on Fundamental Approaches to Software Engineering* (pp. 1-6). Lisbon: Springer Berlin Heidelberg.
10. Beck, K. (2000). *Extreme programming explained: embrace change*. addison-wesley professional.
11. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Kern, J. (2001). Manifesto for agile software development. Najdeno 9.3.2016 na spletnem naslovu http://www.agilemanifesto.org, 2001, 12-4-2002.
12. Beck, T., Demirgüç-Kunt, A., & Maksimovic, V. (2005). Financial and legal constraints to growth: Does firm size matter?. *The Journal of Finance*,*60*(1), 137-177.
13. Blank, S. (2013). *The Four Steps to The Epiphany*. Pescadero: K&S Ranch.
14. Blank, S. (2013). Why the lean start-up changes everything. *Harvard Business Review*, *91*(5), 63-72.
15. Blank, S., & Dorf, B. (2012). *The startup owner's manual*. K&S; Ranch.
16. Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer*, *21*(5), 61-72.
17. Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, *25*(6), 599-616.

18. Carayannis, E. G., Sindakis, S., & Walter, C. (2015). Business model innovation as lever of organizational sustainability. *The Journal of Technology Transfer*, *40*(1), 85-104.
19. Carmel, E. (1995). Time-to-completion factors in packaged software development. *Information and Software Technology*, *37*(9), 515-520.
20. Cawley, O., Wang, X., & Richardson, I. (2013). Lean software development–what exactly are we talking about?. In *Lean Enterprise Software and Systems*(pp. 16-31). Springer Berlin Heidelberg.
21. Chapple, W., Lockett, A., Siegel, D., & Wright, M. (2005). Assessing the relative performance of UK university technology transfer offices: parametric and non-parametric evidence. *Research Policy*, *34*(3), 369-384.
22. Chen, L., & Meng, B. (2010). The application of value stream mapping based lean production system. *International Journal of Business and Management*, *5*(6), 203.
23. Chesbrough, H. (2010). Business model innovation: opportunities and barriers. *Long range planning*, *43*(2), 354-363.
24. Chesbrough, H., & Rosenbloom, R. S. (2002). The role of the business model in capturing value from innovation: evidence from Xerox Corporation's technology spin-off companies. *Industrial and corporate change*, *11*(3), 529-555.
25. Chrisman, J. J., Hynes, T., & Fraser, S. (1995). Faculty entrepreneurship and economic development: The case of the University of Calgary. *Journal of Business Venturing*, *10*(4), 267-281.
26. Christensen, C. (1997). *The innovator's dilemma: The Revolutionary Book That Will Change the Way You Do Business*. New York: HarperCollins Publishers Inc.
27. Chrome OS. (n.d.) V *Wikipediji*. Najdeno 5. avgusta 2016 na spletni strani
28. https://en.wikipedia.org/wiki/Chrome_OS
29. Cohen, C. F., Birkin, S. J., Garfield, M. J., & Webb, H. W. (2004). Managing conflict in software testing. *Communications of the ACM*, *47*(1), 76-81.
30. Davenport, S., Carr, A., & Bibby, D. (2002). Leveraging talent: spin–off strategy at industrial research. *R&D Management*, *32*(3), 241-254.
31. Eisenhardt, K. M., & Martin, J. A. (2000). Dynamic capabilities: what are they?.*Strategic management journal*, *21*(10-11), 1105-1121.
32. *eSistemi – kaskadni model.* Najdeno 16. aprila 2016 na spletni strani http://esistemi.si/kaskadni-model
33. Feyh, M., & Petersen, K. (2013). Lean software development measures and indicators-a systematic mapping study. V *Lean Enterprise Software and Systems* (pp. 32-47). Springer Berlin Heidelberg.
34. Goasduff, L., Rivera. J. (2015, 21. maj). Gartner. Newsroom. V *Gartner Says Worldwide Chromebook Sales Will Reach 7.3 Million Units in 2015*. Najdeno 3.julija 2016 na spletnem naslovu http://www.gartner.com/newsroom/id/3058517
35. Goth, G. (2009). Agile tool market growing with the philosophy. *IEEE software*,*26*(2), 88-91.

36. Graham, P. (2012). Startup= growth. Najdeno 10. junija 2016 na spletnem naslovu http://www. paulgraham. com/growth. html

37. Griffith, E (2016, 3. maj). What Is Cloud Computing? *PC Mag*. Najdeno 13. Julija 2016 na spletnem naslovu http://www.pcmag.com/article2/0,2817,2372163,00.asp
38. Heitlager, I., Helms, R., & Brinkkemper, S. (2007, October). A tentative technique for the study and planning of co-evolution in product. In *Software Evolvability, 2007 Third International IEEE Workshop on Software Envolvability* (pp. 42-47). Paris: IEEE.
39. Herstatt, C., & Hippel, E. (1992). From Experience: Developing New Product Concepts Via the Lead User Method: A Case Study in a "Low-Tech" Field. *Journal of product innovation management*, *9*(3), 213-221.
40. Highsmith J., Orr K., Cockburn A. (2000). »Extreme programming. *E-Business Application*
41. Highsmith, J., & Cockburn, A. (2001). Agile software development: The business of innovation. *Computer*, *34*(9), 120-127.
42. Hines, P., Holweg, M., & Rich, N. (2004). Learning to evolve: a review of contemporary lean thinking. *International journal of operations & production management*, *24*(10), 994-1011.
43. Holweg, M. (2007). The genealogy of lean production. *Journal of operations management*, *25*(2), 420-437.
44. Holweg, M., & Pil, F. K. (2001). Successful build-to-order strategies start with the customer. *MIT Sloan Management Review*, *43*(1), 74.
45. What's Bootstrap (b.l.). V *Investopediji.* Najdeno 10. avgusta 2016 na spletnem naslovu http://www.investopedia.com/terms/b/bootstrap.asp
46. Johnson, M. W., Christensen, C. M., & Kagermann, H. (2008). Reinventing your business model. *Harvard business review*, *86*(12), 57-68.
47. Kajko-Mattsson, M., Westblom, U., Forssander, S., Andersson, G., Medin, M., Ebarasi, S., ... & Holmgren, M. (2001). Taxonomy of problem management activities. V *Software Maintenance and Reengineering, 2001. Fifth European Conference on Software Maintainance and Reengineering (pp. 1-10)*. Lisbon: IEEE.
48. Kampuš, A. (2002) *Projektni management pri razvoju programskih rešitev* (magistrska naloga). Ljubljana: Ekonomska fakulteta.
49. Karlsson, C., & Ahlström, P. (1996). The difficult path to lean product development. *Journal of Product Innovation Management*, *13*(4), 283-295.
50. Kincaid, J. (2011, 4. april). Sales Are At A Tricke In Google's Chrome Web Store. *TechChrunch.* Najdeno 6.8. na spletnem naslovu https://techcrunch.com/2011/01/04/sales-have-slowed-toa-trickle-on-googles-chrome-web-store/
51. Kitchman, B., & Charteres S., (2007). Guidelines for performing systematic literature reviews in software engineering. In *Technical report, Ver. 2.3 EBSE Technical Report. EBSE*.

52. Kos, B. (2013, 17. December). Matrike za start-up podjetja. *Finance-Startaj.* Najdeno 28. avgusta na spletnem naslovu http://startaj.finance.si/8701471?cctest&
53. Krafcik, J. F. (1988). Triumph of the lean production system. *MIT Sloan Management Review*, *30*(1), 41.
54. Larman, C., & Basili, V. R. (2003). Iterative and incremental development: A brief history. *Computer*, (6), 47-56.
55. Lee, P. Y., Hui, S. C., & Fong, A. C. M. (2002). Neural networks for web content filtering. *IEEE intelligent systems*, *17*(5), 48-57.
56. Liker, J. K. (2004). *The Toyota Way: 14 Management Principles from the World's Greatest Manufacturer.* New York: McGraw-Hill Professional.
57. Liker, J. K., & Morgan, J. M. (2006). The Toyota way in services: the case of lean product development. *The Academy of Management Perspectives*, *20*(2), 5-20.
58. Lean Enterprise Institute. (b.l.). What is Lean? Najdeno 15. maja 2016 na spletnem naslovu http://www.lean.org/WhatsLean/
59. Makadok, R. (2001). Toward a synthesis of the resource-based and dynamic-capability views of rent creation. *Strategic management journal*, *22*(5), 387-401.
60. Maccormack, A. (2001) How Internet Companieis Built Software, *MIT Sloan Management Review, 42 (2)*, 75-84
61. Manifest agilnega razvoja programske opreme. Najdeno 9. marca 2016 na spletnem naslovu http://agilemanifesto.org/iso/sl/
62. Marmer, M., Herrmann, B. L., Dogrultan, E. & Berman, R. (2011). *Sturtup Genome - Startup Genome Report Extra: Premature Scaling, 10*.
63. McGrath, R. G., & MacMillan, I. C. (1995). *Discovery driven planning*. Philadelphia: Wharton School, Snider Entrepreneurial Center.
64. Molnar, Š. (2009). *Uvajanje vitke proizvodnje na področju trdnih farmacevtskih oblik* (magistrska naloga). Ljubljana: Fakulteta za Farmacijo.
65. Morris, M., Schindehutte, M., & Allen, J. (2005). The entrepreneur's business model: toward a unified perspective. *Journal of business research*, *58*(6), 726-735.
66. Mynott, C. (2000). *Lean product development*. London: The Institution of Engineering and Technology.
67. Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T., & Abrahamsson, P. (2014). Software development in startup companies: A systematic mapping study. *Information and Software Technology*, *56*(10), 1200-1218.
68. Paulk, M. C. (2001). Extreme programming from a CMM perspective. *Software, IEEE*, *18*(6), 19-26.
69. Paulk, M. C., Weber, C. V., Garcia, S. M., Chrissis, M. B., & Bush, M. (1993). Key practices of the capability maturity modelSM. *Software Engineering Institute Carnegie Mellon University, Technical Report CMU/SEI-93-TR-025, ESC-TR-93*, *178*.
70. Pena, I. (2002). Intellectual capital and business start-up success. *Journal of intellectual capital*, *3*(2), 180-198.

71. Pettersen, J. (2009). Defining lean production: some conceptual and practical issues. *The TQM Journal*, *21*(2), 127-142.
72. Pink, D. H. (2011). *Drive: The surprising truth about what motivates us*. New York: Penguin US.
73. Piškor, M., & Kondić, V. (2010). Lean production kao jedan od načina povećanja konkurentnosti hrvatskih poduzeća na globalnom tržištu. *Tehnički glasnik*, *4*(1-2), 37-41.
74. Poppendieck, M. (2001). Project & Process Management-Best Practices-Lean Programming-Part 2 of 2. W. Edwards Deming's Total Quality Management still rings true for software. *Software development*, *9*(6), 71-75.
75. Poppendieck, M., & Poppendieck, T. (2003). *Lean Software Development: An Agile Toolkit: An Agile Toolkit*. b.k.: Addison-Wesley.
76. Regaldo, A. (2011, 31. oktober). Who coined "Cloud Computing"? *Technology Review.* Najdeno 13. Julija 2016 na spletnem naslovu http://www.technologyreview.com.br/printer_friendly_article.aspx?id=38987
77. Regnell, B., Höst, M., och Dag, J. N., Beremark, P., & Hjelm, T. (2001). An industrial case study on distributed prioritisation in market-driven requirements engineering for packaged software. *Requirements Engineering*, *6*(1), 51-62.
78. Reinertsen, D. G. (2009). *The principles of product development flow: second generation lean product development* (Vol. 62). Redondo Beach: Celeritas.
79. Roberts, J. C., Al-Hamdani, W. (2011). Who can you trust in the cloud?: A review of security issues within cloud computing. V *Proceedings of the 2011 Information Security Curriculum Development Conference* (pp. 15-19). Kennesaw: ACM.
80. Ronen, L. (2015, april). Amigobulls. Articles. V *New Chromebook Strategy Will Complete Google-Android EcoSystem*. Najdeno 4. Julija 2016 na spletnem naslovu http://amigobulls.com/articles/new-chromebook-strategy-will-complete-google-android-eco-system
81. Roper, S., & Love, J. H. (2002). Innovation and export performance: evidence from the UK and German manufacturing plants. *Research policy*, *31*(7), 1087-1102.
82. Rus, M. (2015). Analiza značilnosti start-up podjetij ter start-up ekosistema v Sloveniji. V M. Rebernik & K. Širec (ur.), *Slovenska podjetja in značilnosti start-up ekosistema: Slovenski podjetniški observatorij 2015* (str. 73). Maribor: Univerza v Mariboru, Ekonomsko poslovna fakulteta.
83. Santos, J., Spector, B., & Van der Heyden, L. (2009). *Toward a theory of business model innovation within incumbent firms*. Fontainebleau: INSEAD.
84. Sauermann, H. (2015, 27. junij). Fire in the belly? Employee motives and innovative performance in startups versus established firms. *4th IZZA Workshop on Enterpreneuership Research.* Najdeno 3. aprila 2016 na spletnem naslovu http://www.iza.com/conference_files/EntreRes2013/sauermann_h7739.pdf
85. Schwaber, K. (1996). Controlled chaos: Living on the edge. *American Programmer*, *9*, 10-16.

86. Senge, P. (2014). *The fifth discipline fieldbook: Strategies and tools for building a learning organization*. b.k.: Crown Business.
87. *SCRUMstudy - Scrum Principles* (b.l.). Najdeno 3. maja 2016 na spletnem naslovu http://www.Scrumstudy.com/Scrum-principles.asp
88. Shah, R., Ward, P. T. (2003). Lean manufacturing: context, practice bundles, and performance. *Journal of operations management*, *21*(2), 129-149.
89. Siegel, D. S., Waldman, D. A., Atwater, L. E., & Link, A. N. (2003). Commercial knowledge transfers from universities to firms: improving the effectiveness of university–industry collaboration. *The Journal of High Technology Management Research*, *14*(1), 111-133.
90. Smith, K. (2016). The Real Reasons Why Companies & Business Startups Are Failing, najdeno 20.6. 2016 na spletni strani http://moneycrasheres.com/reasons-buisines-failing-companies
91. Srirama, S. N., Paniagua, C., & Liivi, J. (2013, June). Mobile web service provisioning and discovery in android days. V *Proceedings of the 2013 IEEE Second International Conference on Mobile Services* (str. 15-22). Washington: IEEE Computer Society.
92. Steenhuis, H. J., & de Bruijn, E. J. (2008, September). Innovation and technology based economic development: Are there short-cuts? V *Management of Innovation and Technology, 2008. ICMIT 2008. 4th IEEE International Conference on* (pp. 837-841). Taiwan: IEEE.
93. Steindl, C. (2004). Lean software development. Najdeno 15. Avgusta 2016, na spletnem naslovu http://tem-sw.com/library/LeanSoftwareDevelopment(IBM).pdf
94. Sterlacchini, A. (1999). Do innovative activities matter to small firms in non-R&D-intensive industries? An application to export performance. *Research Policy*, *28*(8), 819-832.
95. Sutton, S.M. (2000). The Role of Process in a Software Start-up. *IEEE Softwate, 17(4), 33*
96. Šinkovec, B. (2012). »Lean« je »IN«. Najdeno 3. marca 2016 na spletnem naslovu: http://www.agencija-poti.si/Clanki/Vsi-clanki/ArtMID/637/ArticleID/115/187Lean171-je-187IN171
97. Tamburri, D. A., Lago, P., & Vliet, H. V. (2007). Organizational social structures for software engineering. *ACM Computing Surveys (CSUR)*, *46*(1), 3.
98. Teece, D. J., Pisano, G., & Shuen, A. (1997). Dynamic capabilities and strategic management. *Strategic management journal*, *18*(7), 509-533.
99. The Last Responsible Moment (17. oktober 2006). Najdeno 18. avgusta 2016 na spletnem naslovu https://blog.codinghorror.com/the-last-responsible-moment
100. *The W. Edwards Deming Institute – The Man*. PDSA Cycle. Najdeno 17.4. 2016 na spletnem naslovu https://www.deming.org/theman/theories/pdsacycle
101. Thomas, K. W. (2000). *Intrinsic motivation at work: Building energy & commitment*. San Francisco: Berrett-Koehler Publishers.

102. Trist, E. L., & Bamforth, K. W. (1951). Some social and psychological consequences of the Longwall method. *Human relations*, *4*(3), 3-38.
103. Trunk, A. (2016, 12. Maj). After analyzing 200 founders' postmortems, researchers say these are the reasoreasons startups fail. *Quartz*. Najdeno 6. junija na spletnem naslovu http://qz.com/682517/after-analyzing-200-founders-postmortems-researchers-say-these-are-the-reasons-startups-fail
104. Ward, A., Liker, J. K., Cristiano, J. J., & Sobek, D. K. (1995). The second Toyota paradox: How delaying decisions can make better cars faster. *Sloan management review*, *36*(3), 43.
105. Womack, J. P., & Jones, D. T. (1996). *Lean thinking: Banish waste and create wealth in your organisation*. New York: Simon and Shuster.
106. Womack, J. P., Jones, D. T., in Roos, D. (1990). *Machine that changed the world*. New York: Simon and Schuster
107. Valdivia, W. D. (2013). University start-ups: Critical for improving technology transfer. *Center for Technology Innovation at Brookings. Washington, DC: Brookings Institution*.
108. Vohora, A., Wright, M., & Lockett, A. (2004). Critical junctures in the development of university high-tech spinout companies. *Research policy*,*33*(1), 147-175.
109. What's Bootstrap (b.l.). V *Investopediji.* Najdeno 10. avgusta 2016 na spletnem naslovu http://www.investopedia.com/terms/b/bootstrap.asp
110. Yilmaz, L. (2007). Modelling software processes as human-centered adaptive work systems. V *European Conference on Software Process Improvement* (str. 148-159). Potsdam: EuroSPI.
111. *Zendesk – Integrations & Apps*. Najdeno 5. julija 2016 na spletnem naslovu http://www.zendesk.com/apps/kabana
112. Zollo, M., & Winter, S. G. (2002). Deliberate learning and the evolution of dynamic capabilities. *Organization science*, *13*(3), 339-351.
113. Zott, C. (2003). Dynamic capabilities and the emergence of intraindustry differential firm performance: insights from a simulation study. *Strategic management journal*, *24*(2), 97-125.
114. Zwilling, M. (2015, 25. december). 7 Ways to Bootstrap Your Business to Success. *Enterpreneuer.* Najdeno 3. julija 2016 na spletnem naslovu https://www.entrepreneur.com/article/254217

PRILOGE

KAZALO PRILOG

Priloga 1: Vodič po intervjuju

Cilj intervjujejev je bil pridobiti relevantne podatke, s katerimi bo mogoče oceniti tako intervjuvančevo poznavanje vitkih metod razvoja PO kot poznavanje metod, ki se uporabljajo pri oblikovanju poslovnega modela vitkih zagonskih podjetij. Na osnovi intervjujev sem želela ugotoviti, koliko se te metode načrtno ali nenačrtno uporabljajo pri samem delu v podjetju in pri oblikovanju poslovnega modela podjetja.

Vprašanja sem razdelila v štiri dele. V prvem splošnem delu intervjuja me je zanimalo delo, ki ga intervjuvanec opravlja v podjetju, njegov položaj in izobrazba. Drugi del vprašanj se je nanašal na metode dela oziroma razvoja programske opreme, ki jih uporabljajo v zagonskem podjetju. V tretjem delu so se vprašanja nanašala na vitka načela razvoja PO in če jih v podjetju uporabljajo. V zadnjem delu vprašanj sem se osredotočila na poizvedovanje poznavanja metode razvoja kupcev in metode vitkega zagonskega podjetja in o uporabi elementov teh metod v zagonskem podjetju, ki je bilo predmet preučevanja.

Intervju:

Splošni del:

1. Ime in priimek
2. Delovno mesto in opis nalog
3. Kakšne so vaše predhodne delovne izkušnje?
4. S kom se pri vašem delu največ povezujete, dogovarjete o nalogah, komu poročate o svojem delu?

Vprašanja, ki se nanašajo na uporabo agilnih metod programiranja:

1. Poznate agilne metode razvoja PO kot so ekstremno programiranje, Scrum metoda, kristalne metode ali vitek razvoj PO?
2. Ste katero od teh metod že kdaj uporabljali pri svojem delu pred delom v podjetju Blocsi?
   2.1. Ste katero od metod razvoja uporabili pri delu v podjetju Bloker?
   2.2. Na kak način ste uporabili to metodo (uporabljena orodja, komunikacija s sodelavci)?
3. Kako je uporaba teh metod vlivala na vaše delo?
   3.1. Pri katerem delu in v kakem obsegu ste uporabili agilne metode razvoja PO?
   3.2. Kakšne so vaše izkušnje pri uporabi agilne metode?
   3.3. Kakšne so prednosti in slabosti agilnih metod po vašem mnenju?

Vprašanja, ki se nanašajo na uporabo načel vitkega razvoja PO:

1. Prvo načelo vitkega razvoja PO se nanaša na odpravo izgub. Se katera od izgub kot delno opravljeno delo, dodatni procesi, dodatne funkcije, preklapljanje med nalogami, čakanje in okvare pojavlja pri vašem delu?
2. Kako pri vas poteka razvoj PO? Poteka v ciklu v tem smislu, da rešujete problem, ponudite rešitev kupcu in izdelek na osnovi povratnih informacij izpopolnite?
3. Ste kdaj razvijali izdelek na način, da ste hkrati razvili več alternativ, jih predstavili uporabnikom in se potem odločili za najboljšo različico?
4. Pomembno pri vitkem razvoju PO je krepitev timskega dela:
   4.1. Vam je namen vašega dela jasen?
   4.2. Se vam zdi, da je namen vašega dela dosegljiv?
   4.3.Se kdaj pogovarjate s končnimi uporabniki?
   4.4 Se vam zdi, da v podjetje deluje kot enoten tim?
5. Pomembno načelo pri vitkem razvoju je »dobaviti, kakor hitro je mogoče«.
   5.1 Se tega načela držite v podjetju?
   5.2. Kako?
6. Če konceptualna integriteta pomeni, da sistem deluje kot povezana celota in da se komponente dobro ujemajo, zaznavna integriteta pa pomeni, da je izdelek predstavlja ravnotežje med funkcionalnostjo, uporabnostjo, zanesljivostjo in ekonomičnostjo: ali skrbite za zaznavno in konceptualno integriteto izdelkov?

Vprašanja, ki se nanašajo na vitek razvoj PO:

1) Se soočate z nedokončanim delom v razvojnem procesu, dodatnimi procesi (papirologija), dodatnimi funkcijami, ki se niso izkazale, preklapljanje med delom, čakanje, premikanje, okvare oziroma napake
2) Krepitev znanja: Rešitve skozi kratke, ponavljajoče se cikle?
3) Sočasen razvoj-ali razvijate kdaj več alternativ hkrati? Kako odkrivate probleme, preden je prepozno
4) Dobaviti, kakor hitro je mogoče-kako to delate?
5) Krepitev timskega dela
6) Vgraditev integritete
7) Videnje celote-verjetno malo?

Vprašanja, ki se nanašajo na metode za razvoj poslovnega modela zagonskih podjetij:

1. Ste imeli poslovni načrt ali ste si zapisali vizijo podjetja, preden sta ga ustanovila?
2. Poznate metodo razvoja kupcev in metodo vitkega zagonskega podjetja?
3. Ko ste imeli idejo o produktu-kako ste preverili, če je izdelek primeren za trg?
4. Odkrivanje strank-kako ste jo izvajali?
   4.1.Ste svoje hipoteze napisali in stestirali?

5. Kako ste izvajali potrditev strank?
   5.1. Poznate štiri faze potrditve strank, kako ste jih izvedli?
6. Kaj ste do sedaj naredili glede ustvarjanja strank?
7. Poznate koncept najosnovnejšega spremenljivega produkta?
   7.1.Koliko časa ste razvijali NSP in kdaj ste ga dali na trg
8. Kako ste preverili ujemanje produkta in trga?
   8.1. Kdaj ste ugotovili, da se produkt in trg ujemata?
9. Ste kdaj naredili vitek okvir podjetja/ izdelka?
10. Ste kdaj sistematično testirali načrt? Kako?

Priloga 2: Vitki okvir za izdelek Bloker Manager

<table>
<tr>
<td rowspan="2">PROBLEM
Naj trije problemi

1) Filtriranje internetnih vsebin
2) Onemogočanje filtriranja vsebin
3) Centraliziranje sistema filtriranja za Chromebooke</td>
<td>REŠITEV
Naj tri lastnosti
1) natančnost filtriranja
2) integracija z Google admin okoljem
3) analitika</td>
<td rowspan="2">EDINSTVENA PONUJENA VREDNOST
Eno, jasno in prepričljivo sporočilo o tem, zakaj si drugačen in zakaj je tvoj produkt vreden nakupa

fine nastavitve filtriranja</td>
<td>NEULOVLJIVA PREDNOST
Ni možno zlahka kopirati ali kupiti

izvedba filtriranja</td>
<td rowspan="2">SEGMENTI KUPCEV
Ciljni kupci
1) šole, ki uporabljajo Chromebooke
2) domači uporabniki Chromebookov (starši)</td>
</tr>
<tr>
<td>KLJUČNI KAZALNIKI
Ključne merljive dejavnosti
1) Prodaja
2) obnova licenc
3) Goole analitika (zadrževanje na spletnih straneh)</td>
<td>KANALI
Pot do kupcev
1) Chrome web
2) distributerji računalniške opreme
3) priporočila uporabnikov</td>
</tr>
<tr>
<td colspan="3">STRUKTURA STROŠKOV
Stroški za pridobivanje kupcev

Distribucijski stroški
Gostovanje
Zaposleni itd.

1) sejmi
2) SEO (optimizacija v
3) vstopni strošek, ki ga zahtevajo distributerji
4) plače
5) infrastruktura (strežniki)</td>
<td colspan="2">TOKI PRIHODKOV
Model prihodkov

Celotna vrednost
Prihodek
Bruto marža

prodaja izdelkov in obnovitev licenc</td>
</tr>
</table>

Vir: A. Maurya: Delaj vitko, 2014, str.8, povzeto iz poslovnega okvirja (http://www.businessmodelgeneration.com)

Priloga 3: seznam kratic

TPS – Toyotin proizvodni sistem

PO – programska oprema

CMM – model zrelostnih nivojev (angl. *Capability Maturity Model*)

CMMI – sestavljen model zrelostnih nivojev (angl. *Capability Maturity Model Integration*)

NUMMI - New United Motor Manufacturing

IT – informacijske tehnologije

NSP - najosnovnejši sprejemljivi produkt

MSP - mala in srednje velika podjetja

BE – Bloker razširitev (angl. *Extension*)

BR – Bloker usmerjevalnik (angl. *Router*)

BM – Bloker Manager