EKSTREMNO PROGRAMIRANJE (XP)

Marec, 2011

Primer razvoja programske opreme za vodenje delovnih procesov

Avtor: Aleš Zebec, managing partner/IT/KMO
dipl. ekon. (UN), ing. rač. in inf., MCSD

BuyITC inovativne internet rešitve d.o.o.
Prvomajska ulica 30, 2000 Maribor, Slovenija
Telefon: (02) 461 0 461
Telefax: (02) 4 624 625
www.buyitc.si
info@buyitc.si

I KAZALO

II UVOD

Razvijalci programske opreme so danes pod vedno večjim pritiskom hitrega in konkurenčnega razvoja. To se odraža tudi v modelih življenjskih ciklov razvoja programske opreme (angl. *Software Development Life Cycle - SDLC*) in uporabljenih metodologijah. V preteklosti je razvoj definiral discipliniran pristop, ki pa danes ne daje več rezultatov in je nepriljubljen predvsem zaradi preobsežne dokumentacije. Potrebne so metodologije, ki v osnovi izhajajo iz nestabilnosti in spreminjajočega se razvojnega okolja.

Podjetja se morajo odzvati hitro na nove priložnosti, tržne spremembe in spremembe v okolju, kar je nazorno pokazala zadnja finančna in gospodarska kriza. Rešitev predstavljajo agilne metodologije razvoja.

Namen seminarske naloge je opredeliti področje agilnih metodologij za doseganje krajših ciklov razvoja programske opreme. Posebej izpostaviti metodologijo ekstremnega programiranja, kot ene od uspešnejših agilnih metodologij. Nato opredeljeno metodologijo ekstremnega programiranja predstaviti na primeru projekta razvoja programske opreme v podjetju ter analizirati uspešnost uporabe izbrane metodologije.

Cilj seminarske naloge je preko analize ugotoviti ali je primer razvoja programske opreme potekal v skladu z izbrano metodologijo. Opredeliti slabosti in možne izboljšave. Ugotoviti ali je možen teoretično opredeljen pristop posamezne metodologije izvesti celovito v praksi, ali pa je potreben hibriden pristop, ki narekuje izbiro različnih agilnih metodologij za doseganje optimalnih rezultatov.

V prvem poglavju so predstavljene osnove agilnih procesov. Opisane so posamezne metodologije, bistvene lastnosti, prednosti in slabosti.

Za podrobnejšo analizo je izbrana metodologija ekstremenega programiranja (angl. *Extreme Programming – XP),* ki je opisana v drugem poglavju. Opredeljeni so ključni dejavniki, načini ter razlogi uporabe.

Sledi poglavje opisa uporabe metodologije ekstremnega programiranja na projektu razvoja programske opreme za upravljanje delovnih procesov.

Sklep povzame vsebino seminarske naloge in ključne ugotovitve, ki sledijo iz analize primera razvoja z metodologijo ekstremnega programiranja, skupaj z možnimi področji izboljšav.

III AGILNE METODOLOGIJE

Agilni procesi razvoja programske opreme skrajšajo življenjski cikel razvoja, kar pomeni pospešen razvoj. Prvi korak je razvoj prototipne različice, sledi integracija funkcionalnosti, iterativni pristop k razvoju dodatnih uporabniških zahtev in testiranje skozi celoten razvojni cikel.

Agilne metodologije izhajajo iz nepredvidljivega okolja, kjer se predpostavlja, da uporabnik pogoste ne more definirati vseh zahtev in potreb na začetku projekta. Bistvo agilnih metodologij je tako prilagajanje kontekstualnim spremembam ter spremembam specifikacij v samem razvojnem procesu. Leta 2001 je 17 posameznikov pripravilo agilni manifest (Agile Manifesto, 2001), katerega ključne točke so naslednje:

- Prednost pred procesi in orodji imajo posameznik, delovni timi, interakcija, timsko delovno okolje in ostali dejavniki, ki vzpodbujajo timsko delo.
- Poudarek je na razvoju programske opreme, ne pa na podrobni in izčrpni dokumentaciji. Nove različice se izdajajo v kratkih intervalih, vsako uro, dnevno, po navadi pa tedensko ali mesečno. Programska koda mora biti enostavna, jasna in tehnično popolna, hkrati pa se zahteve po dokumentaciji zmanjšajo.
- Pomembno je delo s strankami, ne pa pogodbe in pogajanja, ki so seveda še vedno potrebna in pomembna, vendar naravnana v partnerski odnos z naročnikom. Iz poslovnega vidika dobi naročnik rezultate že ob začetku projekta in je prisoten skozi celoten razvojni cikel, kar zmanjša poslovno tveganje.
- Poudarek je na odprtosti za spremembe, ne pa sledenje neprilagodljivim in nerealnim načrtom. Razvijalci programske opreme in predstavniki naročnika morajo biti dobro obveščeni, kompetentni in pooblaščeni za izvedbo sprememb v razvojnem ciklu. To pomeni, da so sodelujoči na projektu pripravljeni na spremembe in je ustrezno oblikovana tudi pogodba, da podpira takšen način razvoja.

Posebnost agilnih metod je zavedanje, da so glavni dejavnik uspeha projekta ljudje in njihova osredotočenost na učinkovitost in prilagodljivost dela. (Cockburn, Highsmith, 2001, str. 122).

Hawrysh and Ruprecht (2000) navajata, da enotna metodologija ne more delovati za različne projekte. Projektni management mora glede na naravo posameznega projekta določiti ustrezne metodologije. To ne pomeni samo uporabo agilnih metodologij, temveč tudi uporabo procesno orientiranih, saj enotni in univerzalni model razvoja programske opreme ne obstaja.

Cockburn opredeljuje agilni razvoj kot uporabo enostavnih, vendar učinkovitih pravil glede poteka projekta ter pravil, ki poudarjajo posameznika in komunikacijo med njimi. Predlaga naslednje prijeme, ki bi naj zagotovili večji uspeh projekta:

- Dva do osem ljudi v pisarni: optimalna komunikacija in grajenje skupnosti.
- Strokovnjaki na lokaciji: zagotavlja hiter in kontinuriran odziv na izvedeno.
- Kratki razvojni cikli: od enega do treh mesecev, kar omogoča hitro testiranje in popravke. Glede na trenutno delovno okolje se ti cikli lahko opredelijo v tednih.

- Popolnoma avtomatizirani regresijski testi: testi enot in funkcionalni testi stabilizirajo kodo in omogočajo kontinuirane izboljšave.
- Izkušeni razvijalci: pospešijo razvoj za 2 do 10 krat glede na ostale člane tima.

Glavne značilnosti agilnega procesa iz vidika hitrosti razvoja, ki omogoča krajše življenjske cikle so naslednje (Miller, 2001):

1. Modularnost na ravni razvoja.
2. Ponavljajoči kratki razvojni cikli, ki omogočajo hitro preverjanje in popravke.
3. Časovne omejitve za ponovitev ciklov, od enega do šest tednov.
4. Racionalnost v razvojnem procesu odstrani vse nepotrebne aktivnosti.
5. Prilagodljivost z upoštevanjem možnosti novih tveganj.
6. Inkrementalni procesni pristop, ki omogoča gradnjo funkcionalnosti v kratkih korakih.
7. Inkrementalni in konvergenčni pristop zmanjša tveganja.
8. Prednost imajo ljudje, pred procesi in tehnologijami.
9. Skupinski in komunikativen način dela.

Lahko izhajamo iz dejstva, da je skupni imenovalec agilnih procesov paradigma iterativnega razvoja. V okviru projekta se pojavljajo nove zahteve, obstoječe spreminjajo in umikajo v ponavljajočih se korakih. Spremembe zahtev in s tem povezane zamude v implementaciji zahtevajo nove pogodbene oblike. Tako zajemajo agilni procesi tudi spremembo poslovnega in pravnega okvirja v katerem se izvajajo projekti, saj je težko določiti končno ceno projekta, ki dopušča spremembe.

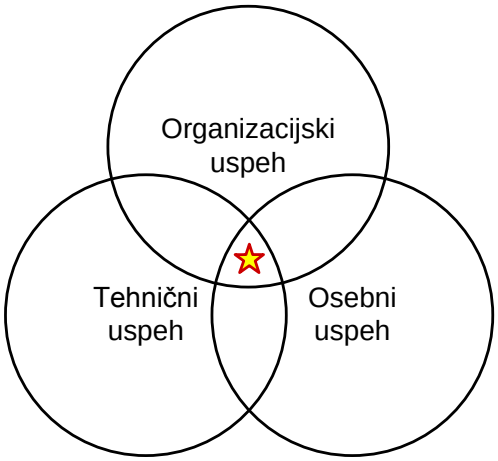
Naročnika je tako glede na novo obliko razvojnega procesa potrebno zadovoljiti v času predaje programske opreme in ne ob začetku projekta. To ne pomeni, da je potrebno ustaviti spremembe že na začetku projekta, temveč zahteva pristope, ki omogočajo obvladovanje sprememb skozi celotni razvoj (Cockburn, Highsmith, 2001). Agilne metode so tako oblikovane, da:

- Zagotovijo prve predaje v tednih, kar zagotavlja hiter uspeh in takojšnje povratne informacije.
- Uporabo enostavnih rešitev, kar pomeni manj sprememb, če pa se te pojavijo, jih je možno hitro implementirati.
- Izboljšave v kvaliteti rezultatov, kar pomeni cenejšo implementacijo kasnejših korakov.
- Neprestano testiranje za zgodnje odkrivanje napak, ki je cenejše.

Osnovni principi agilnih metod so glede na zgornje, učinkovito in motivirano delo razvijalcev, timsko delo in usmerjenost v samokritično, delujočo programsko kodo.

Shore in Warden (2007, str. 6) postavljata vprašanje ali sam agilni razvoj zagotavlja dobre rezultate? Agilni proces je osredotočen na doseganje osebnih, tehničnih in organizacijskih rezultatov. Uspeh na vseh področjih pa zagotavlja dobre rezultate.

Slika 1: Vennov diagram oblik uspeha



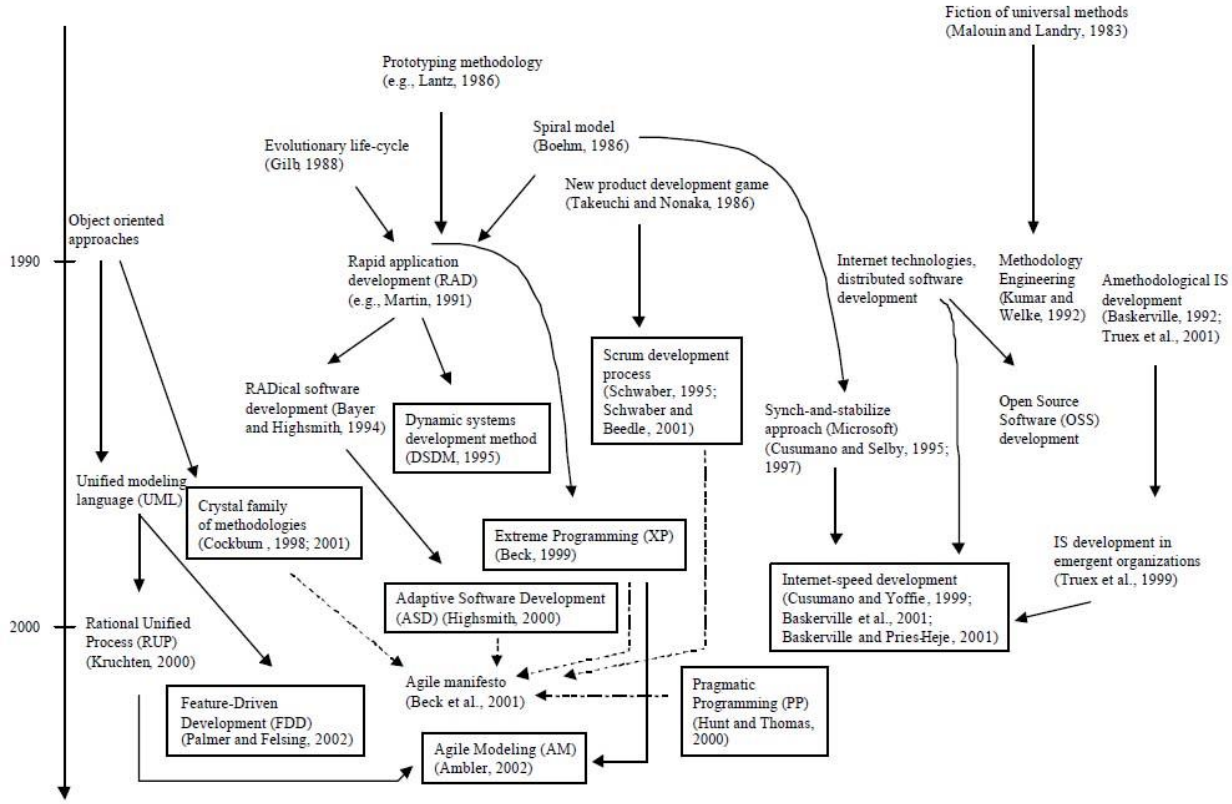
Vir: Shore, Warden, 2007, The art of agile development, str. 5

- **Organizacijski uspeh:** agilne metode dosegajo organizacijski uspeh s poudarkom na razmerju vrednost/stroški. Za naročnika se to odraža v večji donosnosti naložb. Metode postavljajo zahteve že v začetnih fazah projekta, kar omogoča podjetju, da hitro prepozna težave in ustavi projekt ter tako prepreči večje izgube. Rezultati se dosegajo z agilnimi timi, ki vključujejo strokovnjake iz določenega razvojnega področja, kar omogoča osredotočenost na ključne funkcionalnosti, katere mora programska oprema zagotavljati. Agilni projekti tako razvijejo, objavijo in predajo prvo najpomembnejše dele projekta, ostale pa dodajajo v novih različicah, ki so pogoste. Ta pristop dramatično poveča vrednost nad stroški. Ko poslovanje ali nove informacije zahtevajo spremembo, se temu prilagodijo tudi agilni timi. Izkušen agilni tim bom dejansko vnaprej iskal nepričakovane priložnosti preko katerih bo izboljšal delovanje. Agilne metode hkrati nižajo stroške, saj producirajo bistveno manj napak v kodi. Glede na izkušnje agilni tim hitreje zavrne slabe rešitve in se osredotoči na enostavne. Agilni timi komunicirajo hitro in jedrnato, ter napredujejo tudi takrat, ko ključni posamezniki niso prisotni. Ves čas so samokritični, preverjajo delo in izboljšujejo kodo, kar omogoča lažje vzdrževanje in nadgradnje.
- **Tehnični uspeh:** primer metodologije, ki je posebej usmerjena v doseganje tehničnega uspeha je ekstremno programiranje, ki bo podrobneje razloženo v naslednjem poglavju. Programerji delajo skupaj, kar omogoča neprestano usklajevanje, preverjanje in posledično boljšo kodo. Koda se ves čas integrira in tako se lahko programska oprema objavi tudi v krajših intervalih, kadar je smiselno. Tim je osredotočen na končanje določene funkcionalnosti preden se loti naslednje, kar preprečuje nepričakovane zamude in omogoča predhodne spremembe na naslednji funkcionalnosti.
- **Osebni uspeh:** ker agilne metode poudarjajo delo posameznika in tima, so ti posledično deležni tudi uspeha in nagrad, kar posamezniku prinaša osebni uspeh, zadovoljstvo pri delu in motivacijo. Management bo cenil donos naložbe preko uporabne vrednosti programske opreme. Uporabniki so vključeni v razvoj, imajo vpliv in so zadovoljni z novim orodjem. Projektni vodje bodo zadovoljni z možnostjo prilagoditve razvoja potrebam, kar jim

omogoča doseganje večjega zadovoljstva pri uporabnikih. Razvijalci bodo bolj vključeni v vse aspekte razvoja, način dela pa jim bo omogočal večjo samostojnost in boljše delovne pogoje, ki so manj stresni. Testerji so bolj integrirani v timsko delo, njihov vpliv na kvaliteto na vseh stopnjah projekta je večje, njihovo delo postane bolj zanimivo in manj monotono zaradi ponavljanja testov.

V nadaljevanju so opredeljene aktualne agilne metodolgije. Ena od metodologij je **ekstremno programiranje**, ki bo podrobneje predstavljena v naslednjem poglavju.

Slika 2: Evolucijski pregled agilnih metodologij



Vir: IEEE. Evolucija agilnih metodologij. International Conference on Software Engineering, May, 2003

III.I SCRUM

Scrum metodologija je bila razvita za managiranje razvoja programske opreme v spremenljivem okolju. Gre za empirični pristop, ki bazira na fleksibilnosti, prilagodljivosti in produktivnosti. Scrum dopušča razvijalcem, da sami izberejo ustrezne razvijalske tehnike, metode in pristope za proces implementacije. Vsebuje neprestane aktivnosti ravnanja, s poudarkom identifikacije nepravilnosti in pomanjkljivosti v razvojnem procesu in samem načinu dela.

III.II Crystal družina metodologij

Crystal družina metodologij vključuje različne metode izmed katerih se izberejo najbolj primerne za posamezni projekt. Crystal pristop vključuje načine kako posamezne metode prilagoditi, da bodo ustrezale različnim okoliščinam in potrebam posameznega projekta. Vsak član družine je označen z različno barvo, ki odraža težo posamezne metode. Crystal način predlaga izbiro pravilno obarvane metode za določen projekt, glede na velikost in prioriteto projekta. Večji projekti zahtevajo več koordinacije in tako težje metode, kot manjši. Crystal metode so odpre za različne oblike praks, orodja ali delovna okolja, kar omogoča tudi integracijo metodologij kot sta ekstremno programiranje ali Scrum.

III.III Razvoj narekovan z lastnostmi

Razvoj narekovan z lastnostmi (angl. *Feature Driven Development - FDD*) je procesno orientirana metoda za razvoj ključnih poslovnih sistemov. Metoda se osredotoča na faze načrtovanja in izvedbe. Vključuje iterativni razvoj in prakse, ki so uporabne predvsem v industriji. Različen nabor praks pomeni, da so FDD procesi unikatni za vsako situacijo. Poudarek je na kvaliteti skozi celoten proces razvoja in vključuje pogoste, konkretne objave, skupaj z natančnim spremljanjem napredka projekta.

III.IV Prilagodljiv razvoj programske opreme

Prilagodljiv razvoj programske opreme (angl. *Adaptive software development – ASD*) ponuja nov pogled na razvoj programske opreme v organizaciji. Vključuje rešitve za razvoj večjih in kompleksnejših sistemov. Metoda poudarja inkrementalni in iterativni razvoj z neprestano pripravo prototipov. Predhodnik ASD metode je radikalni razvoj programske opreme (angl. *Radical Software Development – RAD*). Metoda zagotavlja ogrodje, ki zagotavlja dovolj kontrole, usmerjanja, ravnanja, da se projekt razvija v pravo smer, ampak ne preveč, ker bi to zaviralo razvoj in kreativnost.

III.V Agilno modeliranje

Agilno modeliranje (angl. *Agile Modeling – AM*) je nov pristop k izvajanju modeliranja. Ponuja uporabo praks modeliranja z upoštevanjem agilne filozofije. Ključen pri agilnem modeliranju je poudarek na praksah modeliranja in kulturnih načelih. Osnovna zamisel je spodbujanje razvijalcev, da proizvajajo dovolj razvite modele za podporo ključnih potreb načrtovanja in dokumentacije. Cilj

je zmanjšati število potrebnih modelov in dokumentacije. Kulturna načela se odražajo v načinu spodbujanja komunikacije, načinu dela in organizaciji delovnih timov.

III.VI Metoda dinamičnega razvoja sistemov

Metoda dinamičnega razvoja sistemov (angl. *Dynamic Systems Development Method - DSDM*) je motoda, ki jo je razvil konzorcij v Veliki Britaniji. Metoda je bila prvič objavljena leta 1994. Osnovna zamisel je, da v okviru razvoja prvo prilagodimo roke in vire, nato pa ustrezno prilagodimo še zahteve glede funkcionalnosti določenega produkta. DSDM metoda izvira iz metodologije hitrega razvoja aplikacij (angl. *Rapid Application Development*). DSDM je velikokrat opredeljena kot prva celovita metoda agilnega razvoja programske opreme.

III.VIIPragmatično programiranje

Pragmatično programiranje (angl. *Pragmatic programming - PP*) predstavlja nabor dobrih praks programiranja. V ospredje postavlja tehnike, ki dejansko omogočajo izvajanje opisanih agilnih metodologij. Samo metodo sestavlja zbirka kratkih nasvetov, ki se ukvarjajo z vsakodnevnimi težavami. Skupaj je 70 nasvetov. Te prakse predstavljajo pragmatično perspektivo in postavljajo v ospredje inkrementalni, iterativni razvoj, temeljito testiranje in razvoj usmerjen v potrebe uporabnika (angl. *User-centered Design*).

III.VIII Razvoj s hitrostjo interneta

Razvoj s hitrostjo interneta (angl. *Internet-speed development – ISD*) je najmanj znan pristop k agilnemu razvoju. Uporabljen je kadar je potrebno hitro objavljanje rešitev in so razvojni cikli zelo kratki. ISD ponuja opisno, ravnalno usmerjeno ogrodje potrebno za hitre objave različic. Ogrodje sestavljajo pravila glede časovnic, opredeljene so odvisnosti kvalitete in prilagoditve procesov. Prilagoditve procesov opredeljuje osredotočenost na učinkovite zaposlene in ne procese. Če so zaposleni usposobljeni, strokovni, itd. potem je manj potrebe za procese in obratno. ISD je metodologija, ki je bolj ravnalno in poslovno orientirana, kot ostale metodologije. ISD izvira iz »sinhroniziraj in stabiliziraj« (angl. *Synch and stabilize*) pristopa, ki ga je razvil Microsoft, za obvladovanje hitro se spreminjajočega poslovnega in organizacijskega okolja.

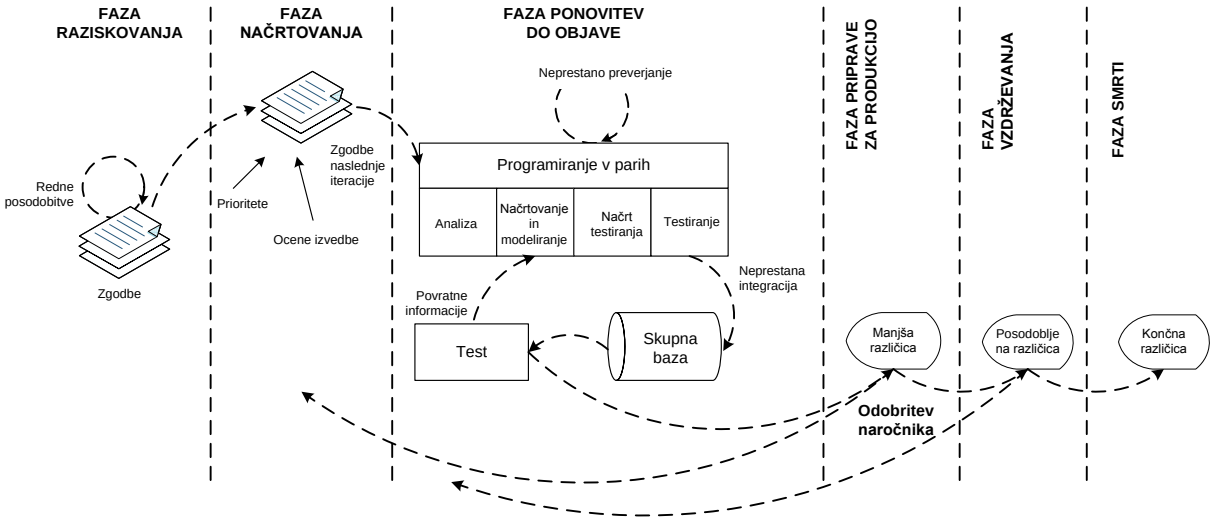
IV EKSTREMO PROGRAMIRANJE

Metodologija ekstremno programiranje se je razvila kot odgovor na težave, ki so nastale zaradi dolgih razvojnih ciklov, kot so jih poznali običajni razvojni modeli (Beck, 1999a, str. 70). Nastala je iz potreb in praks, ki so bile učinkovite v procesu razvoja. Po več preizkušnjah v praksi, je nastala teoretična oblika metodologije ekstremnega programiranja na podlagi ključnih principov in praks. Čeprav posamezne prakse uporabljene v metodi niso nove, so bile zbrane in povezane tako, da oblikujejo novo metodologijo razvoja programske opreme. Izraz »ekstremno« izhaja iz dejstva, da metodologija uporablja smiselne in že obstoječe prakse, principe na nov in ekstremen način (Beck 1999b).

IV.I Proces razvoja

Življenjski cikel metode razvoja ekstremnega programiranja sestavlja pet faz (Beck, 1999b, str. 131):

Slika 3: Življenjski cikel XP procesa



Vir: Abrahamsson, Salo, Ronkainen, 2003, Agile software development methods:

Review and analysis, str. 19

1. Faza raziskovanja

V fazi raziskovanja (angl. Exploration phase) naročniki zapiše zgodbe, katere želijo vključiti v objavi prve različice. Vsaka zgodba opisuje funkcionalnost ali lastnosti, ki jo je potrebno vključiti v program. Hkrati se razvojni tim spozna z orodji, tehnologijo in praksami, ki jih bodo uporabljali v okviru projekta. Izvede se pregled, testiranje tehnologije in arhitekturnih možnosti, ki se preverijo s

pripravo prototipa sistema. Ta faza traja nekaj tednov ali mesecev, odvisno od že obstoječega znanja in izkušenj z izbranimi tehnologijami.

2. Faza načrtovanja

Načrtovanje (angl. *Planning phase*) postavi prioritete, vrstni red zgodb in uskladi vsebino prve objavljene različice. Razvijalci ocenijo koliko časa zahteva posamezna zgodba, nato se pripravi in uskladi terminski načrt. Ta faza traja nekaj dni, terminski plan pa ne obsega več kot dva meseca.

3. Faza ponovitev do objave

Faza ponovitev do objave (angl. *The Iterations to release phase*) vključuje več ponovitev sistema pred objavo prve različice. Terminski načrt pripravljen v fazi načrtovanja se razgradi na določeno število ponovitev, kjer implementacija vsake traja od enega do štirih tednov. Prva ponovitev obsega sistem, ki vsebuje arhitekturo celotnega sistema. To je doseženo z izbiro zgodb, ki bodo povzročile in obsegale izvedbo strukture celotnega sistema. Naročnik izbere zgodbe posamezne iteracije. Funkcionalni testi, ki jih pripravi naročnik, se izvajajo na koncu vsake iteracije. Sistem je pripravljen za produkcijo na koncu zadnje iteracije.

4. Faza priprave za produkcijo

Faza priprave za produkcijo (angl. *The Productionizing phase*) zahteva dodatno testiranje in preverjanje performans sistema preden je predana naročniku. V tej fazi so možne dodatne spremembe, ki zahtevajo odločitev ali se vključijo v trenutno ali naslednjo različico. V tej fazi se lahko pojavi potreba po pospeševanju faze iz treh na en teden. Preložene ideje in predlogi se dokumentirajo za kasnejšo implementacijo.

5. Faza vzdrževanja

Ko je prva različica pripravljena za produkcijo, mora izvedba še vedno teči, medtem, ko se dodajajo nove iteracije. V ta namen je uporabljena faza vzdrževanja (angl. *Maintenance phase*), ki vključuje tudi naloge uporabniške podpore. Hitrost razvoja se po navadi po objavi prve različice upočasni. Faza vzdrževanja po potrebi zahteva vključevanje novih članov in spremembe v strukturi timov.

6. Faza smrti

Faza smrti (angl. *The Death phase*) je blizu, ko naročnik nima več dodatnih zgodb za implementacijo. To pomeni, da mora sistem zadovoljiti naročnika v ostalih pogledih, kot so npr. hitrost in zanesljivost delovanja. To je čas v procesu ekstremnega programiranja, ko se pripravi vsa zahtevana dokumentacija in spremembe arhitekture, načrtovanja ali kode niso več dovoljene. Smrt

projekta se lahko pripeti, če sistem ne zadovoljuje zahtev naročnika ali nadaljni razvoj postane predrag.

IV.II Vloge in odgovornosti

V metodi ekstremnega programiranja obstajajo različne vloge za različne naloge in namene. V nadaljevanju so predstavljene vloge (Beck, 1999b, str. 139).

Programer kodira, testira kodo. Njegova naloga je pisati enostavno in jasno definirano kodo. Komunikacija in koordinacija z ostalimi programerji in člani tima je ključni dejavnik uspeha metodologije ekstremnega programiranja.

Naročnik piše zgodbe, funkcionalne teste in odloča, kdaj je določena zahteva izvedena. Določi tudi prioritete in vrstni red implementacije glede na zahteve.

Tester pomaga naročniku pri pripravi funkcionalnih testov. Redno opravlja funkcionalne teste, objavlja rezultate in vzdržuje orodja testiranja.

Sledilec prenaša povratne informacije v proces. Sledi ocenam za izvedbo, ki jih je postavil razvojni tim in vrača povratne informacije glede natančnosti, kar se uporabi v prihodnjih ocenah. Sledi napredku posamezne iteracije in ocenjuje ali so bili cilji doseženi v okviru časovnih omejitev in z danimi viri. Na podlagi ugotovljenega predlaga spremembe.

Trener (angl. *Coach*) je odgovoren za celoten proces. Poznati in razumeti mora XP metodologijo. Njegova naloga je vodenje ostali članov razvojnih timov.

Svetovalec je zunanji član, ki ima določeno potrebno tehnično znanje. Pomaga razvojnemu timu pri reševanju specifičnih problemov.

Manager (Big Boss) sprejema odločitve. Za potrebe odločanja komunicira z ostalimi člani projektnega tima, ocenjuje trenutno situacijo, razpoznava težave in pomanjkljivosti v procesu razvoja.

IV.III Prakse

Ekstremno programiranj je zbirka idej in praks, ki so vzete iz že obstoječih metodologij (Beck, 1999a, str. 71). Naročnik sprejema poslovne odločitve, medtem, ko razvijalci sprejemajo tehnične odločitve.

Cilj metodologije je uspešen razvoj programske opreme, kljub vprašljivim in spreminjajočim se zahtevam v majhnih ali srednje velikih timih. Kratke ponovitve in konstantne majhne objave različic, hitre povratne informacije, sodelovanje naročnika, komunikacija in koordinacija, nenehna integracija in testiranje, skupno lastništvo kode, omejena dokumentacija, programiranje v parih so glavne značilnosti. Sledi opis praks:

- Načrtovanje igre (angl. *Planning game*): tesna interakcija med naročnikom in razvijalci. Razvijalci ocenijo potreben napor za implementacijo zgodb naročnika. Naročnik se nato odloči glede področja in časovnice objave različice.

- Kratke/majhne objave različic: majhen sistem je hitro pripravljen za produkcijo, vsaj enkrat na 2 do 3 mesece. Nove različice so objavljene dnevno ali vsaj enkrat mesečno.
- Metafora: sistem je definiran z metaforo ali metaforami med naročnikom in razvijalci. Ta skupna zgodba je vodilo celotnega razvoja in opisuje kako deluje sistem.
- Enostavno oblikovanje: poudarek je na oblikovanju in načrtovanju najbolj enostavne rešitve, ki jo je možno v danem trenutku implementirati. Nepotrebna kompleksnost in dodatno koda se nemudoma odstranita.
- Testiranje: razvoj programske opreme narekuje testiranje. Testi enot so implementirani pred kodo in se izvajajo pogosto. Naročniki pišejo funkcionalne teste.
- Refaktoriranje (angl. *Refactoring*): gre za rekonstrukcijo sistema, kjer se odstrani podvojena koda, izboljša komunikacija, poenostavi delovanje in doda fleksibilnost.
- Programiranje v parih: dva programerja pišeta kodo za enim računalnikom.
- Skupno lastništvo: vsak lahko spremeni katerikoli del kode, kadarkoli.
- Stalna integracija: nov del kode je integriran v osnovo sistema takoj, ko je pripravljen. Sistem je tako integriran in kompiliran večkrat na dan. Integracija zahteva izvedbo testov, ki morajo biti uspešni.
- 40 urni delavnik: je maksimalno število delovnih ur v tednu. Zaporedna tedna z nadurami nista dovoljena. Če se pojavi takšna situacija se obravnava kot problem, ki ga je potrebno rešiti.
- Naročnik na lokaciji: naročnik mora biti nenehno dostopen za razvojni tim.
- Standardi kodiranja: pravila kodiranja uporabljajo programerji. Poudarek je na komunikaciji med programerji skozi kodo.
- Odprto delovno okolje: predlagana je večja pisarna z več delovnimi mesti. Programerski pari se postavijo na sredo prostora.
- Pravična merila: timi imajo lastna pravila, ki jim sledijo, ta pa se lahko kadarkoli spremenijo. Sprejet mora biti konsenz glede spremenjenih pravil in opravljena ocena vpliva na delo.

IV.IV Uporaba in izkušnje

Beck predlaga (1999a, str. 77), da se ekstremno programiranje uvaja postopno. Predlaga uporabo metode za najtežji problem v trenutnem razvojnem procesu. Osnovna zamisel je, da ne obstaja enovit proces, ki ustreza vsakemu projektu, temveč je potrebno prilagoditi posamezne prakse.

Pojavlja se vprašanje koliko lahko spremenimo prakse in principe, da še vedno lahko govorimo o ekstremnem programiranju? Ne obstajajo primeri ali empirične raziskave v katerih bi to vprašanje bilo obravnavano.

Praktični pogled uporabe metode je dokumentiral Jeffries (2001). V njegovem delu so zbrane tehnike, ki pokrivajo večino praks, opredeljene preko projekta razvoja programske opreme v industriji, kjer je bil metoda uporabljena. Posebej obravnava težave glede določanja zahtevnosti in potrebnega časa za izvedbo določenih nalog. Predlaga uporabo “spike”, ki je kratek pogrešljiv eksperiment kodiranja, ki omogoča vpogled v določen problem, ki ga je potrebno rešiti s programiranjem.

Ekstremno programiranje je najbolj dokumentirana izmed agilnih metodologij, zato obstaja veliko raziskav, člankov, poročil in različnih praks. Različni avtorji obravnavajo področja kot je programiranje v paru, uporaba metodologije same. Predstavljene so slabosti in prednosti metodologije. Maurer in Martel (2002a, str. 87) sta predstavila nekaj konkretnih statistik glede izboljšane produktivnosti z uporabo metode pri razvoju spletnih rešitev. Poročilo podaja povečanje števila vrstic kode za 66%, 302% povečanja v številu novih metod in 282% povečanje števila implementacije novih razredov (Maurer, Martel, 2002b, str. 8).

IV.V Področje uporabe

Metodologija ni primerna za vsak projekt. Vse omejitve te metodologije še niso znane, zato so potrebne dodatne empirične in eksperimentalne raziskave iz različnih perspektiv. Je pa znanih nekaj splošnih omejitev.

Metoda je namenjena majhnim in srednje velikim razvojnim skupinam. Beck (1999b, str. 155) predlaga velikost delovne skupine med 3 in največ 20 članov. Enako pomembno je delovno okolje. Komunikacija in koordinacija med projektnimi člani mora biti ves čas omogočena, zato je pomembno, da so locirani v istem prostoru. Pomembna je tudi poslovna kultura, ki vpliva na razvojne skupine. Odpor do praks ekstremnega programiranja in principov določenih članov projekta je lahko vzrok neuspelega projekta. Omejitve in ovire velikokrat povzroči tudi izbrana tehnologija.

V RAZVOJ PROGRAMSKE OPREME ZA VODENJE DELOVNIH PROCESOV

V podjetju BuyITC, podjetje za razvoj programske opreme iz Maribora, se je v okvirju uvajanja procesne organiziranosti pojavila potreba po programskemu orodju za vodenje delovnih procesov. Glede na dejstvo, da podjetje razvija programske rešitve se je na podlagi analize splošnih, funkcijskih in tehničnih zahtev odločilo, da bo programsko rešitev razvilo samo in s tem zadržalo in razvijalo znanje s tega področja. Za razvoj rešitve je bila izbrana metodologija ekstremnega programiranja. Sledi analiza razvoja programske rešitve po fazah, ki jih predlaga ekstremno programiranje.

V.I Analiza razvoja po fazah ekstremnega programiranja

V **fazi raziskovanja** so se zbrale zgodbe zaposlenih, ki bodo to programsko opremo uporabljali. Zgodbe so vsebovale predvsem funkcionalne zahteve. Obsegale so: funkcionalnosti beleženja nalog in opravil, evidence razvijalcev, projektov, poslovnih partnerjev, funkcionalnosti za organiziranje dela, vodenja projektov in skupinskega dela.

Razvojni oddelek je znotraj te faze določil tehnologije in arhitekturo aplikacije. Izbrano je bilo razvojno okolje in tehnologije: Microsoft Visual Studio 2010, .NET Framework 4.0, ASP.NET platforma, programski jezik C# in podatkovna baza MS SQL 2008. Ker razvojni oddelek tehnologije in orodja uporablja pri razvoju drugih produktov in rešitev posebni testi in spoznavanje tehnologije ni bilo potrebno.

V tej fazi se je definiral tudi uporabniški vmesnik, ki je ključnega pomena pri razvoju rešitve, saj dialogi vsebujejo veliko število podatkov. Izbrane in izdelane so bile grafične predloge. Sledili so testi uporabniških vmesnikov (angl. U*sability*), kjer so posamezniki v podjetju na osnovi testnih strani preverjali prijaznost vmesnika, logiko, sledljivost in interaktivnost vsebinskih elementov. Sledilo je usklajevanje in prilagoditve. Rezultat je bila celostna grafična podoba rešitve (CGP) in dokument pravil za pripravo mrežnega modela aplikacije.

Projekt je dobil delovno ime “Tasks”. Bistvenega pomena je bila skupna vizija projekta, ki se glasi: *“Tasks pomaga zaposlenim pri skupinskem delu. Omogoča vodenje in upravljanje delovnih procesov. Osnovne predpostavke so organiziranje dela, sodelovanje in enostavnost. Ne uvaja striktne organizacijske hierarhije, temveč poveže zaposlene na delovnem nivoju. Rešitev predvideva dobronamerno sodelovanje zaposlenih, ki ustvarjajo lastna pravila dela za maksimalno učinkovitost. Namenoma ne vključuje mehanizmov, ki bi uveljavljali določeno vedenje med uporabniki. Rešitev odraža kvaliteto in znanje. Uporabniki ga radi uporabljajo, čeprav težko določijo zakaj. Vključuje manjše dodatke, ki povečajo uporabniško in poslovno vrednost, skozi napreden način dela.*

Sodelovanje, enostavnost, vsebina in kvaliteta imajo prednost pred širino funkcij. Razvoj je osredotočen na optimizacijo obstoječih funkcionalnosti preden se dodajo nove.

Tasks je pomemben za naše stranke, ker jim omogoča organizirano obliko dela, obvladovanje delovnih procesov in povečuje učinkovitost. Strankam omogoča, da razvijejo podjetniški način dela, ki je najbolj inovativen. Produkt bo omogočil nadaljnji in dobičkonosen razvoj podjetja.

Rešitev bomo objavljali v več fazah in v vsaki fazi povečali stopnjo uspešnosti. Prva faza bo dokazala smiselno koncepta. Razvoj druge faze bo zahteval 6 mesecev in bo vključevala razvoj celotne palete funkcionalnosti. Tretja faza bo vključevala razširitev produkta na komercialni nivo. Zastavljen cilj so 3 namestitve v prvih treh mesecih."

Faza je trajala 2 tedna.

V **fazi načrtovanja** so se določile prioritete posameznih zgodb in posamezne faze oziroma vsebina posameznih različic. Opredelile in dodelile so se posamezne naloge razvijalcem, ki so ocenili potreben čas za izvedbo. Pripravil in uskladil se je terminski plan. Predviden čas izvedbe je bil ocenjen na 2 meseca. Uporaba rešitve z naborom osnovnih funkcionalnosti je bila opredeljena za predajo v produkcijo 3 mesece od pričetka prve faze (marca 2011).

V tej fazi so se modelirali poslovni procesi, ki jih mora programska rešitev podpreti. Uporabilo se je orodje Corel iGrafX.

Sama faza je trajala 7 dni.

Slika 4: Gantogram terminskega načrta izvedbe

Vir: BuyiTC interni dokumenti

V **fazi ponovitev do objave** so se določile posamezne iteracije. Opredeljenih je bilo 8 iteracij v okviru katerih bo implementiranih 7 glavnih modulov, ki sestavljajo rešitev, ena od iteracij pa je namenjena integraciji modulov.

Prva iteracija vključuje arhitekturo celotnega sistema, ki pa uporablja že izdelane arhitekturne elemente ostalih rešitev,ki jih je razvilo podjetje. Iteracije so se izvajale v zaporedni fazi. Na koncu vsake faze so se izvedli funkcionalni testi, ki so jih izvedli izbrani uporabniki in so vključevali vpis, ažuriranje, branje in brisanje vsebine.

Faza je bila končana v 9 tednih. Zamuda na projektu je znašala en teden. Razlog za zamudo, je bil nov način izvajanja projektov v skladu z metodo ekstremnega programiranja. Težava se je zaradi

zasedenosti virov pojavila predvsem pri zagotavljanju funkcionalnih testov na koncu vsake iteracije, kar je podaljšalo fazo izvedbe za en teden.

Na podlagi poslovnih procesov modeliranih v prejšnji fazi so se oblikovali entitetno relacijski diagrami (angl. *Entity-relationship model – ERD*) za vsak modul ali iteracijo posebej. Na podlagi teh diagramov je bila izdelana shema podatkovne baze. Uporabljeno je bilo orodje Microsoft SQL Server 2008.

V **fazi priprave za produkcijo** je bilo izvedeno dodatno testiranje in preverjanje performans sistema. Izvedeni so bili dodatni testi po izvedeni zadnji iteraciji integracije.

V tej fazi so bile ugotovljene dodatne zahteve. Glede na prioritete je bilo 15% dodatnih zahtev vključenih v prvo različico in dodala se je nova iteracija v kateri so se te funkcionalnosti izvedle. Preostalih 85% zahtev se je preneslo v naslednje faze. Faza z dodatno iteracijo, je bila izvedena v enem tednu. Nazadnje je sledil test sprejemljivosti (angl. *Acceptance Test*), nakar je bila prva različica nameščena v produkcijo in uporabo 1. marca 2011. Razvojni oddelek je pričel z izvedbo naslednje faze.

Faza vzdrževanja je vključevala naloge za podporo uporabnikom. Dodale so se iteracije za izvedbo prilagoditev in popravkov, ki so bile potrebne za pravilno delovanje v produkcijskem okolju.

Ker je produkt v četrti fazi izvedbe **faza smrti** še ni dosežena. Glede na odzive uporabnikov, ki uporabljajo prvo različico so uporabniki z rešitvijo zadovoljni. Pojavljajo pa se težave z strukturo vsebine, ki se vnaša v program. Največ težav predstavlja strukturiranje vsebine na način, da bo ustrezala merilom za vzpostavitev poslovnega poročanja. Zaradi sprotnega prilagajanja in ocenjevanje rešitve je odpora pri uporabi aplikacije zelo malo, projekt ima veliko podporo v podjetju, zaposleni so ga sprejeli.

VI SKLEP

Naloga opisuje agilne metodologije, posebej ekstremno programiranje, kot eno od najbolj pogostih metodologij za razvoj programske opreme. Metoda je opredeljena na primeru razvoja programske rešitve za upravljanje delovnih procesov.

Osnovna izhodišča agilih metodologij opisuje prvo poglavje, kjer so opisane tudi najbolj pogoste metodologije. Drugo poglavje podrobno opisuje metodologijo ekstremnega programiranja. Metoda je predstavljena sistematično z naslednjimi sklopi, ki identificirajo proces metode, vloge in odgovornosti, prakse, uporabo in izkušnje ter področje uporabe. Tretje poglavje opisuje primer projekta izvedenega po metodologiji ekstremnega programiranja. Postopek razvoja je opisan po posameznih fazah, kot definirano v metodi.

Cilj seminarske naloge je bil preko analize in poteka projekta ugotoviti ali se je projekt izvajal z izbrano metodologijo. Cilj je bil dosežen in ugotovljeno, da se je projekt izvajal v skladu z metodologijo ekstremnega programiranja. V okviru priprave naloge je bila izvedena primerjava tudi z drugimi projekti, kjer pa je bilo ugotovljeno, da metodologija ekstremnega programiranja ne bi bila primerna. Poglavitni razlog je v naročniku projekta. Naročnik opisanega projekta je bilo podjetje samo, saj je produkt namenjen lastni uporabi in kasneje trženju. Pri drugih projektih pa je naročnik tretja oseba, ki ni pripravljena uporabiti in sodelovati preko opredeljene metode. Ugotovljeno je bilo, da je za podjetje pri razvoju lastnih produktov in rešitev primerna uporaba metodologije ekstremnega programiranja, saj zagotavlja predvsem hiter in kvaliteten razvoj, s stroškovno kontrolo, kljub sprotnim prilagoditvam. Za ostale oblike je potreben hibridni pristop, ki uporablja različne pristope in prakse različnih agilnih metodologij.

V fazi izvedbe projekta so se zraven pozitivnih lastnosti, kot so hitrost razvoja, pripravljenost razvojnega tima na spremembe, tekoča in bolj sproščena komunikacija, osredotočenost na rešitev brez nepotrebne formalne dokumentacije, zagotavljanje sprotne in visoke kvalitete, koračna in pregledna implementacija, pojavile tudi težave. Izvor težav je predvsem v omejenih virih. Razvijalci običajno delajo na več projektih hkrati, kar pomeni težave v razpoložljivosti. V fazi projekta je časovno usklajevanje predvsem na področju testiranja pomenilo časovne zamude. Metodologija narekuje tudi pripravljenost na spremembe definicij v času implementacije, kar v sami izvedbi ni problematično. Tukaj se občasno pojavi psihološki dejavnik, kjer posameznik razvijalec lahko zgubi motivacijo, saj lahko pride do večkratne redefinicije in ponovne implementacije posameznega modula, kar povzroči frustracije pri delu. Takšni scenariji niso težavni samo iz vidika razvijalca, temveč povzročajo nepotrebne stroške projekta, zaradi nejasno in nepopolno opredeljenih zgodb in nepripravljenosti sodelovanja naročnikov. To se pogosto pojavi pri velikih organizacijah. Dodatno težavo predstavlja ovrednotenje takšnega projekta. Večina projektov se izvaja s fiksno pogodbeno ceno, naknadna usklajevanja, nadgradnje, spremembe pa je težko uskladiti in ovrednotiti. V kolikor je naročnik pripravljen na proces razvoja, kjer bo končna cena jasna na koncu projekta, je takšna metodologija izvedljiva.

Spremembe so danes realnost razvoja programske opreme. Predvidevati vse možnosti je nemogoče, ne glede na količino načrtovanja. Tradicionalni pristopi k razvoju programske opreme pomenijo v trenutnem poslovnem okolju večanje stroškov in izgubo kontrole nad projektom. Ekstremno programiranje prinaša način razvoja kvalitetne programske opreme ob zmernih stroških, kar je

bistvo inovativnega in konkurenčnega razvoja programskih rešitev z manjšim tveganjem in večjimi možnostmi uspešnega zaključka projekta.

VII LITERATURA IN VIRI

1. Satzinger, J.W, Jackson, R.B, & Burd, S.D (2009). *Systems analysis and design in a changing world, 5. edition.* Boston, Ma: Course Technology.

2. Thomke, S., & Reinertsen, D. (1998). *Agile product development: Managing development flexibility in uncertain environments.* California Management Review, 41(1), 8. Najdeno na spletnem naslovu http://search.proquest.com/docview/216145656?accountid=16468

3. Larman, C. (2003). *Agile and iterative development: A manager's guide.* Addison-Wesley Professional.

4. Shore, J., & Chromatic. (2007). *The art of agile development.* O'Reilly Media.

5. Abrahamssona, P., Warstab, J., Siponenb, M.T., & Ronkainena J. (2003) *New Directions on Agile Methods: A Comparative Analysis.* Portland, Oregon, IEEE.

6. Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., Tesoriero, R., Williams, L., & Zelkowitz, M. *Empirical Findings in Agile Methods.*, 2002.

7. Abrahamsson, P., Salo, O., & Ronkainen, J. *Agile software development methods: Review and analysis.* Universitiy of Oulu: VTT Electronics, 2002.

8. Baumeister, H. (2005). *Extreme programming and agile processes in software engineering: 6th international conference, xp 2005, sheffield, uk, june 18-23, 2005, proceedings ... / programming and software engineering).* Springer.

9. Beck, K., Hannula, J., Hendrickson, C., Wells, D., & Mee, R. (1999a). *Embracing change with extreme programming.* Computer, 32(10), 70. Najdeno 26. marca 2011 na spletnem naslovu http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.36.8261&rep=rep1&type=pdf

10. Beck, K. (1999b). *Extreme programming explained: Embrace change.* Addison-Wesley Professional.

11. Maurer, F., & Martel, S. (2002a). *Extreme programming: Rapid development for web-based applications.* IEEE Internet Computing, 6(1), 86. Retrieved from http://search.proquest.com/docview/197328968?accountid=16468

12. Maurer, F., & Martel, S. (2002b). *On the productivity of Agile Software Practices: An Industrial Case Study.* Najdeno 24. marca 2011 na spletnem naslovu: sern.ucalgary.ca/~milos\papers\2002\MaurerMartel2002c.pdf

13. *Spletna stran Agile Alliance.* Najdeno 21. marca 2011 na spletnem naslovu http://www.agilealliance.org/.

14. *Manifesto for Agile Software Development.* The Agile Alliance. Najdeno 21. marca 2011 na spletnem naslovu http://agilemanifesto.org/.

15. *islovar - Slovar infromatike.* Najdeno 2. marca 2011 na spletnem naslovu http://www.islovar.org

16. *Slovar slovenskega knjižnega jezika - Inštitut za slovenski jezik Frana Ramovša ZRC SAZU.* Najdeno 2. marca 2011 na spletnem naslovu http://bos.zrc-sazu.si/sskj.html

17. *EVROTERM večjezična terminološka zbirka.* Najdeno 2. marca 2011 na spletnem naslovu http://evroterm.gov.si/

18. Beck, K., & Andres, C. (2004). *Extreme programming explained: Embrace change (2nd edition).* Addison-Wesley Professional.

19. Jeffries, R., Anderson, A., & Hendrickson, C. (2000). *Extreme programming installed (xp series).* Addison-Wesley Professional.

20. Miller, G.G (2001). *The Characteristics of Agile Software Processes.* The 39th International Conference of Object-Oriented Languages and Systems (TOOLS 01). Santa Barbaba, CA.

21. Hawrysh, S., & Ruprecht, J. (2000), *Light Methodologies: It's Like Déjà Vu All Over Again*, Cutter IT Journal, Vol. 13:4-12.

22. Cockburn, A. (2001). Agile *software development (agile software development series).* Addison-Wesley Professional.

23. Auer, K., & Miller, R. (2001). *Extreme programming applied: Playing to win.* Addison-Wesley Professional.

24. Wells, D. *Extreme Programming: A gentle introduction.* Extreme Programming. Najdeno 21. marca 2011 na spletni strani http://www.extremeprogramming.org/ in http://www.agile-process.org.