UNIVERZITET DŽEMAL BIJEDIĆ U MOSTARU
FAKULTET INFORMACIJSKIH TEHNOLOGIJA MOSTAR

ZAVRŠNI RAD br. 9999

Agilni *software development*

(Nacrt)

Student: *Ernad Husremović, DL 2792*
Mentor: *mr. Adil Joldić*

ver: 0.0.5

Mostar, decembar 2012.

Sadržaj

1. Uvod

Šta znači 'biti agilan'?
Agilni razvoj software-a ne predstavlja specifični proces. Agilni razvoj je način na koji se razmišlja o razvoju software-a(J.Shore i S.Warden, 2008, str. 9).
Osnovna polazišta ovog pristupa opisuje "Agilni manifest"[1], koji je definisan kroz četiri vrijednosti i 12 principa:
Vrijednosti:

- Ljudi i interakcije ispred procesa i softverskih alata
- *Software* koji funkcioniše ispred detaljne dokumentacije
- Komunikacija sa klijentima ispred pregovora
- Odgovor na promjene ispred slijeđenja plana

Principi:

- Glavni prioritet je zadovoljiti zahtjeve klijente kroz ranu i kontinuiranu *isporuku* software-a
- Blagonaklono prihvatiti *promjene* funkcionalnih zahtjeva, čak i u kasnijim fazama razvoja.
- Funkcionalan software treba isporučivati *često*, nakon par hefti ili mjeseci, nastojeći da taj period bude što kraći.
- Najefikasniji način razmejene informacija unutar razvojnog tima je direktna - 'face-to-face' komunikacija.
- Software koji *funkcioniše* je primarna mjera uspjeha projekta.
- Agilni procesi promoviraju održivi razvoj. Finansijeri, developeri i korisnici trebaju biti u stalnoj koordinaciji, bez obzira na dužinu trajanja projekta.
- Kontinuirano pažnja na *kvalitet* tehničkih operacije i dobar dizajn povećava agilnost.
- *Jednostavnost*, kao vještina postizanja maksimalnog učinka sa što manje rada, je krucijalni agilni princip.

[1] `http://agilemanifesto.org/iso/en/principles.html`

– Najbolja arhitektura, funkcionalni zahtjevi i dizajn se postižu u *samo-organizovanim* timovima.

– Tim redovno analizira predhodne operacije u cilju bolje efektivnosti (*refleksija*). Na osnovu tih rezultata, tim utvrđuje buduće operacije.

1.1. Agilne metode

Najpoznatije Agilne metode[2]:

– XP - *Extreme programming*

– *Scrum*

– *The Crystal Methods*

– *Feature Driven Development*

– *Lean Development*

– *Agile Modeling*

Metoda	Dužina iteracije	Veličina tima	Distribucija tima[3]
XP	1-2 hefte	2-10	Nije moguće
Scrum	2-4 hefte	1-7	Moguće

Tablica 1.1: Karakteristike pojedinih agilnih metoda

[2](D.Cohen et al.)

[3]Mogućnost da članovi tima budu geografski dislocirani, da ne sjede zajedno

2. Početak projekta

2.1. Vizija

TODO

3. Organizacija i uloge unutar tima

3.1. Product menadžer

3.2. Couch

3.3. Klijent

3.4. Tester

3.5. Developer

4. Analiza

4.1. *Stories* vs *Requirements*

Incremental requirements

4.1.1. *Story card*

Njihova glavna osobina je da su ”customer-centric” - usmjereni na korisnika. Svaka *Story card* treba za cilj imati isporuku nove vrijednosti kljentu.

4.2. Prva iteracija

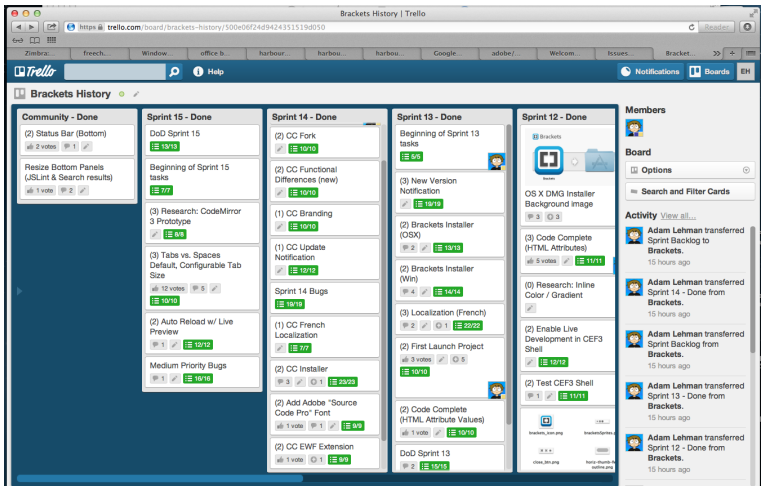
TODO

5. Planiranje i praćenje projekta

5.1. Uvod

5.1.1. Empirijska kontrola procesa

Defined Process Control vs Empirical Process control(A.Tomasini i M.Kearns, 2012)
Trello agilni projekt menadžment sistem



Slika 5.1: trello

5.2. Slack

Slack - "među-story" prostor(J.Shore i S.Warden, 2008, str. 275)

5.3. Estimating

Velocity, points, Consistent Estimates

5.3.1. Timeboxing, Iteration Timebox

5.4. Commitment

6. Komunikacija

6.1. Meetings

TODO

6.2. *Issue management* (ISSUE)

Zadatak, Aktivnost, ticket
"Issue"[1] predstavlja određeni kokretni projektni zadatak:

- prijedlog za realizaciju (ideja)
- prijava programske greške (eng. bug)
- realizacija nove ili nadogradnja postojeće funkcije sistema (eng. new feature, feature upgrade)

6.2.1. Poznati alati za *issue management*

- gitlab issues[2]
- redmine issues[3]

Sve što je relevantno za projekat treba publikovati na ISSUE sitem.
Ako razvojni tim nije kolociran[4] većina komunikacija između članova obavlja se elektronskim putem.
"Issues management" sistem je po pitanju realizacije konkretnih projektnih zadataka najbolji način komunikacije.[5]

[1] developeri često koriste termin "ticket", ili "bug" (čak i kada se ne radi o programskog grešci)
[2] (Husremović, 2012b)
[3] `http://www.redmine.org`, `http://redmine.bring.out.ba`
[4] Nalazi se i djeluje na jednom mjestu, nije geografski distribuiran
[5] email kao sredstvo komunikacije u ove svrhe treba izbjegavati. Email komunikacija brzo postaje nepregledna za većinu projektnih zadataka.

Komunikacija putem ISSUE sistema time obuhvata komunikaciju na temu novih ideja i prijedloga, realizacije ili nadogradnje funkcija sistema, kao i prijavu i otklanjanje programskih ”bug”-ova (grešaka).
Parcijalni pristup (npr. ograničite se samo na ”bug”-ove) stvara prostor da bitne informacije budu nedostupne svim članovima tima.
Redovno nakon gornje preporuke slijedi pitanje:
Kako ću znati da li je nešto relevantno ili nije ?
Ako ne znaš da li je relevantno - stavi na ”issue”.
Nije nikakav problem da se na ticketima u početku nalaze suvišne informacije. Problem je kada informacije nedostaju.
Takođe je vrlo bitno da se informacije na sistem publikuju bez kašnjenje, a ne retroaktivno u formi izvještaja.
Mnoge informacije nakon dan ili dva postaju beskorisne.
Issue management treba reflektovanti život i dinamiku projekta.

6.2.2. *Issues* i novi član tima

”Issues” su sredstvo komunikacije.
Kada se novi član uključi u ekipu, redmine komentari (ili nedostatak komentara) su odličan indikator kompetencija člana.
Iz njih se vrlo brzo dođe do informacija kojim vještinama član raspolaže, koje su njegove profesionalne navike. Na osnovu tih informacija se može djelovati:

- Organizvati fokusiranu edukaciju za novog člana
- Zajednički rad sa iskusnijim kolegama itd.

7. Inkrementalni dizajn

8. Development

8.1. *Source code management* (SCM)

Detaljno u materijalu "Agilni *software development*, GIT SCM"(Husremović, 2012b)

8.2. Testiranje

Testiranje i refactoring etaljno u materijalu "Agilni *software development*, Tehnike testiranja"(Husremović, 2012c)

8.3. Stalna integracija (CI)

Detaljno u materijalu "Agilni *software development*, *Continuous Integration* (CI)"(Husremović, 2012a)

8.4. *"Jednom i samo jednom"*

"Jednom i samo jednom" (eng. "Once and Only once") princip:

> Svaki koncept izrazi jednom. (i samo jednom !).[1]

8.5. *Spike* rješenja

Spike (bos. ekser, smeč) razjašnjavaju tehička pitanja koje susrećemo, pri čemu se izbjegava kompleksnost produkcijskog kôda.(J.Shore i S.Warden, 2008, str. 334)

[1] (J.Shore i S.Warden, 2008, str. 319)

9. Release management

9.1. Testno vs produkcijsko okruženje

Detaljno u materijalu "Agilni *software development*, *Test* & *deploy* infrastruktura"(Husremović, 2012d)

10. Agilno = haotično ?!

Striktne granice između faza dizajna, implementacije i isporuke rješenja korisniku nestaju.
Prakticiranjem TDD-a developer stalno mijenja dizajn[1] u malim koracima u toku implementacije.
Breakthrough faze obezbjeđuju veće promjene. Bitno je uočiti da se te promjene dešavaju u najbolje, najproduktivnije vrijeme - onda kada developer "sazrije" po pitanju konkretnog problema, kada dobro ovlada postojećim stanjem - njegovim ograničenjima i dobrim stranama.
Što ne trebaš (čega nema u *story*-jima) - ne implementiraj !
Nemamo potrebu anticiparti dizajnerska i arhitektonska rješenja na duge staze.
Nedostatak anticipacije će voditi "siromašnim", "kratkovidnim" rješenjima ?
::ODGOVORITI, ELABORIRRATI::

[1] inkrementalni dizajn TODO: referenciraj se na poglavlje

11. Zaključak

Agilni *software development* naravno nije *Silver Bullet*[1]
Međutim, on iz temelja mijenja uvriježene inžinjerske prakse. Agilni pristup unosi "životnost" u razvojni proces software-a.
U toku razvoja sofware-a[2] se od članova razvojnog tima u znak rezignacije mogu čuti konstatacije slijedećeg tipa:

> *Hah, sve bi bilo u redu da živimo u idealnom svijetu ... Da imamo dovoljno vremena i/ili developera ... Da smo prije implementacije "xyz" funkcije dobili sve potrebne informacije ...*

Agilni pristup kao temeljno polazište uzima realni svijet i realne potrebe korisnika software-a.
On razvojni tim stalno podsjeća i usmjerava da je glavni cilj sofverskog projekta ostvariti vrijednost za *korisnika software*-a.[3] Ta vrijednost (upotrebljivost, korisnost) je sama po sebi vremenski *dinamična* kategorija. Agilni *software development* nastoji razvojni ciklus usaglasiti sa tom činjenicom.

[1] `http://en.wikipedia.org/wiki/No_Silver_Bullet`

[2] ali i sveukupnog životnog ciklusa

[3] Prvi slogan moje firme je bio: *Iskoristite računar*. Međutim, slogan se nije puno koristio. Još gore, firma u svom djelovanju u mnogim segmentima odstupila od principa ovog slogana. Nakon 16 godina, rad na ovoj temi me je podsjetio na taj slogan. Iz ove perspetkive mogu konstatovati da je to bio sjajan slogan. Šteta što smo ga zaboravili - i slovom i djelom.

Literatura

A.Tomasini i M.Kearns. *Agile transition, What you Need to Know Before Starting*, 2012.

D.Cohen, M.Lindvall, i P.Costa. *Agile Software Development, (DACS State-of-the-Art/Pratice Report)*.

Ernad Husremović. *Agilni software development, Continuous Integration (CI)*, 2012a. URL `https://github.com/hernad/agile_dev_env/raw/master/agile_ci.pdf`.

Ernad Husremović. *Agilni software development, Git SCM*, 2012b. URL `https://github.com/hernad/agile_dev_env/raw/master/agile_git.pdf`.

Ernad Husremović. *Agilni software development, Tehnike testiranja*, 2012c. URL `https://github.com/hernad/agile_dev_env/raw/master/agile_test.pdf`.

Ernad Husremović. *Agilni software development, test & deploy infrastructure*, 2012d. URL `https://github.com/hernad/agile_dev_env/raw/master/agile_test_release.pdf`.

J.Shore i S.Warden. *The Art of Agile Development*. O'Reilly, 2008.

Dodatak A
Software toolset

1. Mac OS X 10.8.2
2. mvim, vim tekst editor ver 7.3
3. MacTex (TeX Live 2012)

Dodatak B
Software repozitoriji

– Agilni developerski environment - `https://github.com/hernad/agile_dev_env`