Sveučilište Josipa Jurja Strossmayera u Osijeku Odjel za matematiku

Eugen Markovinović

Metode agilnog razvoja softvera

Scrum, Crystal i Dynamic system development method

Osijek, 2018.

Sadržaj

1. Uvod

Tokom godina razvoja softvera, planska izrada sa svom potrebnom dokumentacijom koja uključuje punu specifikaciju i izradu modela se pokazala kao vrlo pouzdana i dobra. Međutim, ubrzani razvoj tehnologije, sve veća potreba tržišta i u novije vrijeme ogromna potreba za brzinom izbacivanja novog i updateanog softvera čine ovu staru i pouzdanu metodu ne učinkovitom. Kod detaljnog pisanja dokumentacije i pripreme gdje ona možda i nije potrebna se gubi dragocjeno vrijeme te softver koji na kraju izađe, bude već zastario i možda i više totalno ne upotrebljiv.

Da to ne bi bilo dovoljno, u današnje vrijeme vrlo često ni ne znamo sve potrebne zahtjeve te ih možda čak ne zna ni korisnik. Pisanje specifikacije je u ovakvome slučaju noćna mora te ona možda i ako ju uspijemo napisati postaje ne učinkovita ili čak u nekim slučajevima netočna! Zbog toga ju moramo mijenjati što troši dodatno vrijeme i stvara dodatne financijske probleme, ali i nezadovoljstvo samih korisnika.

Prilagodba tržištu je jedan od osnovnih uvjeta opstanka na tržištu, ne samo u IT industriji nego i općenito. Za male kompanije ovo igra presudnu ulogu, moraju biti brzi i fleksibilni jer u suprotnom su osuđeni na propast. Stalno prilagođavanje korisnicima i njihovim zahtjevima se pokazuje bitno ovdje te nije više dovoljno samo na početku napisati specifikaciju.

Svi gore spomenuti argumenti su doveli do promjene pogleda na razvoj softvera. Upravo potreba za brzinom i fleksibilnošću dovela je do zamjene starog planskog razvoja sa novim metodama, takozvanim agilnim metodama razvoja softvera.

2. Agilne metode

Kako smo spomenuli, drugačije potrebe i uvjeti tržišta su uzrokovali nastanak agilnih metoda. Postoje razne agilne metode no sve imaju otprilike iste karakteristike:

- Procesi specifikacije, dizajna i implementacije su isprepleteni, nema detaljne specifikacije a dokumentacija dizajna je automatizirana ili minimizirana.
- Sistem se razvija nizom verzija, gdje korisnici igraju ulogu u izmjeni iduće verzije. Zato su agilne metode inkrementalne gdje su inkrementi mali i brzi.

Iz svega dosada je poprilično jasno što agilne metode žele postići, brzinu i rješavanje procesa koje se smatraju nepotrebnima. Zato je dosta vodećih inženjera koji zastupaju ovakve metode došlo do manifesta njihovih takozvanih zajedničkih vrijednosti.

Njihove vrijednosti su:

- Pojedinci ispred procesa i alata
- Softver koji funkcionira ispred detaljne dokumentacije
- Surađivanje korisnika ispred pregovaranja u sklapanju ugovora
- Odgovor na promjene ispred praćenja plana

Na temelju tih vrijednosti došli su do zajedničkih principa, koje zapravo sve agilne metode dijele:

- Uključenje korisnika – korisnici su uključeni kroz cijeli proces razvoja softvera, njihova uloga je da predlažu i prioritiziraju nove zahtjeve te da evaluiraju iteracije.
- Inkrementalno izbacivanje softvera - Softver se razvija u inkrementima
- Ljudi ne procesi - Vještine ljudi u razvojnom timu trebaju biti prepoznate i iskorištene, inženjere treba pustiti da rade na svoj način bez da im se prethodno kaže kako.
- Prihvati promjenu – Očekuj da se zahtjevi sistema mijenjaju te napravi sistem da se može prilagoditi tim promjenama
- Održavaj jednostavnost – Kada god je moguće, radi na izbacivanju složenosti iz sistema, fokusiraj se na jednostavnost softvera i razvojnog procesa.

Koristeći ove vrijednosti i principe, agilne metode su se pokazale učinkovitima i dobrima kod produkta gdje kompanija radi mali ili produkt srednje veličine. Također su se dobrima pokazali kod projekata gdje je jasna inicijativa korisnika da se uključi u proces razvoja softvera.

Treba napomenuti da je nekada vrlo teško provesti navedene principe. Primjerice uključivanje korisnika je nekada vrlo teško izvedivo jer korisnici nisu voljni žrtvovati svoje vrijeme da bi pomogli u razvoju softvera. Kada interes korisnika znatno padne, vrlo je teško ispuniti ovaj princip. Stalan fokus na promjene je isto tako vrlo stresan za pripadnike tima, kako nisu svi prilagođeni na takav rad, zna se dogoditi da pojedinci jednostavno ispadnu ili odustanu pod pritiskom. Nadalje, princip održavanja jednostavnosti je nekada izrazito teško

provesti jer zahtijeva dodatan rad, a kako je u ovim metodama potrebna brzina i vremenski rokovi su uvijek vrlo kratki, jednostavnost nekada pada u drugi plan.

Još jedan problem koji se javlja kod ovih metoda je što baš taj nedostatak dokumentacije i oslanjanje na ljude uzrokuje ovisnost o pojedincima. Ako se dogodi da netko bitan za projekt, napusti projekt često zahtjeva dosta vremena da se novi inženjer uputi u dosadašnji rad.

Unatoč svemu, ove metode su bitne u današnje doba i vrlo ih je bitno istraživati i upoznati. Postoje brojne metode koje su zasnovane na već spomenutim principima i vrijednostima, pa navedimo neke od njih:

- Ekstremno programiranje
- Scrum
- Crystal
- Adaptive Software Development
- DSDM – Dynamic system development method
- Feature Driven Development

U nastavku ću opisati neke od njih te objasniti malo kako funkcioniraju.

3. Scrum

Scrum je agilna metoda razvoja softvera koja se fokusira na upravljanje i menadžment iterativnog procesa umjesto na pojedine tehničke pristupe. Naravno, kao i ostale agilne metode, iterativni proces je ključ Scrum-a te ga spominjem odmah u prvoj rečenici. Veliku zaslugu za razvoj Scrum-a ima Ken Schwaber koji je napisao brojne članke i upute za njega te napisao Scrum Guide uz Jeffa Sutherlanda, što se smatra temeljem Scrum metode.

Scrum Guide naglašava kako je temelj njihove metode empirijski pristup i znanje iz iskustva, donošenje odluke na temelju onoga što je znano. Navodi da svaki empirijski pristup karakteriziraju tri stupa:

- Jasnoća (transparency)
- Inspekcija
- Prilagodba

Također navodi i vrijednosti Scrum-a:

- Predanost
- Hrabrost
- Fokus
- Otvorenost
- Respekt

Ove vrijednosti bi trebale karakterizirati članove Scrum tima.

Scrum je zamišljen za rad sa malim i agilnim timovima, preporuka kaže od tri do devet članova koje vodi takozvani Scrum master. Scrum master pazi da se tim pridržava pravila i vrijednosti Scrum-a te on vrši interakciju sa ljudima izvan tima. Zanimljivo je da Scrum master ne govori timu kako da radi svoj posao, oni to sami odlučuju, on samo upravlja općenitim procesom i daje općenite upute.

Sam proces Scrum-a je jednostavan te mu je glavni dio koji ga određuje takozvani sprint. Sprint je općenito vremenski okvir od mjesec dana ili manje (često dva tjedna) u kojemu se pravi jedan inkrement iterativnog procesa.

Sam sprint se sastoji od:

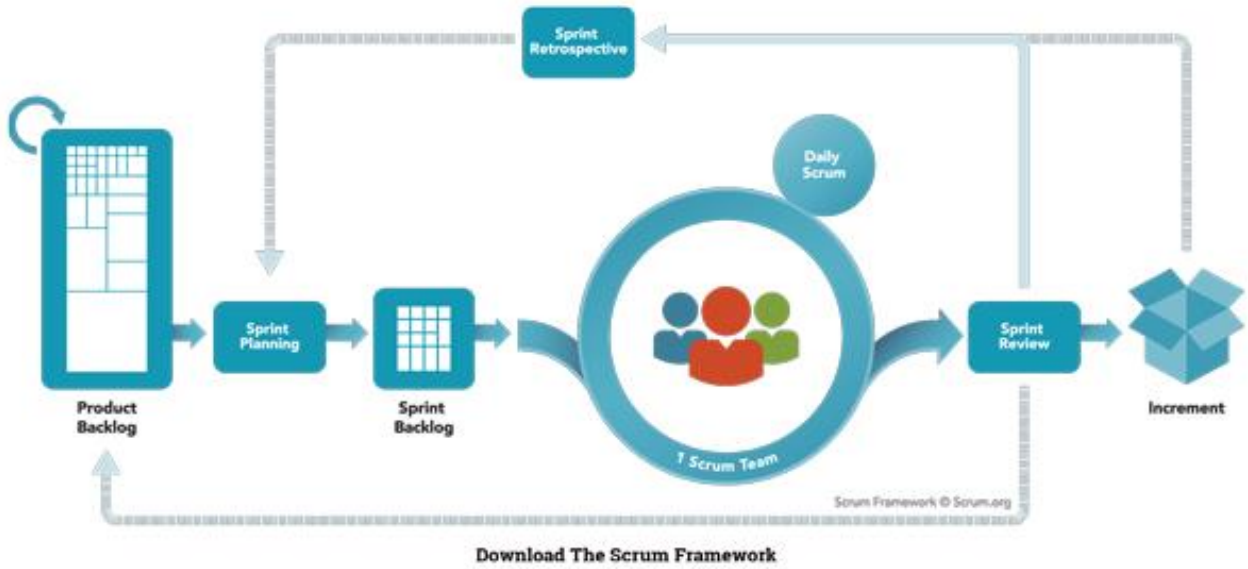
- Planiranja
- Dnevnih Scrum-ova
- Inženjerskog rada
- Pregleda
- Revizije

Dnevni Scrum-ovi su otprilike 15-minutni dnevni sastanci koje vodi Scrum master. Na njima se planira što bi se trebalo raditi u iduća 24 sata.

Pregled sprinta se odvija na samom kraju sprinta, na kojem se gleda što je sve napravljeno i uspoređuje s prethodno zamišljenim. Ovdje mogu sudjelovati i potencijalni kupci proizvoda.

Revizija(retrospekcija) je prilika da sam razvojni tim iznese svoje mišljenje. Ovaj se sastanak događa nakon pregleda, a prije planiranje idućeg sprinta. Na njemu sagleda kako se sam proces odnosio na ljude, što poboljšati u idućem sprintu i priča se o samoj implementaciji.

Osim sprinta postoje još dva glavna dijela Scrum-a. Samo dizajniranje takozvanog „product backlog-a“ što predstavlja dokumentaciju i traženu specifikaciju. Treba napomenuti da dokumentacija nikada nije kompletirana i ona se dopunjava tokom samoga procesa (u takozvanim „sprink backlog-ovima“). Zadnji događaj bi bio zatvaranje projekta, o ovome neću puno govoriti jer sama Scrum metoda nije zamišljena da ima jasan kraj. Do ovoga dolazi ako je projekt napušten ili jednostavno više nije potreban.



4. Crystal

Devedesetih godina 20. stoljeća Alistair Cockburn je provodio istraživanje u nastojanju da izradi novu metodu razvoja softvera. Nakon brojnih intervjua sa inženjerima i djelatnicima izradio je takozvane Crystal metode. Glavna značajka ovih metoda agilnog razvoja je fokus na ljude i njihove interakcije prije svega ostaloga. Po njemu razvoj softvera trebamo tretirati kao igru gdje potičemo sve na interakciju, kreativnost i stvaranje ideja.

Kako je kod Crystal metoda naglašeno da je svaki projekt drugačiji i svi projekti trebaju drugačiji pristup, Cockburn je razvio više različitih familija Crystal metode. Cockburn je zaključio da svojstva projekta ovise o dvije stvari, veličini tima i razini kritičnosti projekta. Nadalje, zaključio je da broj ljudi povećava kritičnost projekta. Da bi se lakše razumjelo što je htio reći, nazvao je metode zajedničkim imenom Crystal na temelju svojstva kristala, boji i tvrdoći koji simboliziraju veličinu tima i kritičnost. Isto tako je i imenovao svoje metode po bojama kristala.

- Clear(čista, prozirna)
- Yellow(žuta)
- Orange(naračasta)
- Red(crvena)

Svaka od ovih metoda je namijenjena za različitu veličinu tima, prikazano na slici ispod.

Jasno je da Crystal metoda stavlja veliki naglasak na same ljude što je čini posebnom među ostalim agilnim metodama. Malo bolji opis Crystal metoda daju slijedeće karakteristike:

- Česta dostava – Isporuka koda korisnicima je vrlo česta
- Reflektivno poboljšanje – Uvijek je moguće na neki način unaprijediti produkt
- Osmotska komunikacija – Informacije teku između tima koji radi na istoj lokaciji što omogućuje da se upiju bitne informacije bez da ljudi direktno sudjeluju u samom razgovoru
- Osobna sigurnost - Svima mora biti zagarantirana sigurnost i svi moraju moći govoriti bez straha
- Fokus – Svaki član zna točno što mora raditi i ne mora mijenjati stalno svoje zadatke
- Lagan pristup ekspertnim korisnicima – Crystal dobiva redoviti feedback od korisnika
- Okruženje sa automatskim testovima i stalnom integracijom

The 7 properties of Crystal Clear

mandatory

Osmotic communication
- Sit close
- Communicate often
- Include all

Reflective improvement
- Improve methodology
- Improve the team
- Reflection workshop

Frequent delivery
- Delievery, not iteration
- Every two month
- Min. 2 deliveries per project

non-mandatory

Easy access to expert users
- Written specs are not enough
- Feedback early and often
- Real users, not your manager

Personal safety
- If you're scared, you don't perform
- When feeling safe, you can take critique

Focus
- Minimize disruptions
- Loud and quiet time
- Multitasking get's less done

Agile technical enviornment
- Automated tests
- Configuration management
- Frequent integration

5. Dynamic systems development method (DSDM)

Prva verzija DSDM-a je nastala otprilike 1995. kada se pokušavalo uvesti malo discipline u takozvanu „Rapid Application development“ (RAD) metodologiju. Posljednja verzija DSDM-a poznata kao DSDM Atern je izašla 2007. no postoje ljudi koji govore da je DSDM bio preteča svih agilnih metoda. 2014. je objavljen DSDM handbook u koji se može koristiti kao osnova DSDM principa.

Kao i ostale agilne metode, DSDM funkcionira na inkrementalno iterativnom principu te rad s ljudima. No postoje i vrlo značajne razlike zbog kojih se DSDM ističe. Navedimo osnovne principe DSDM-a:

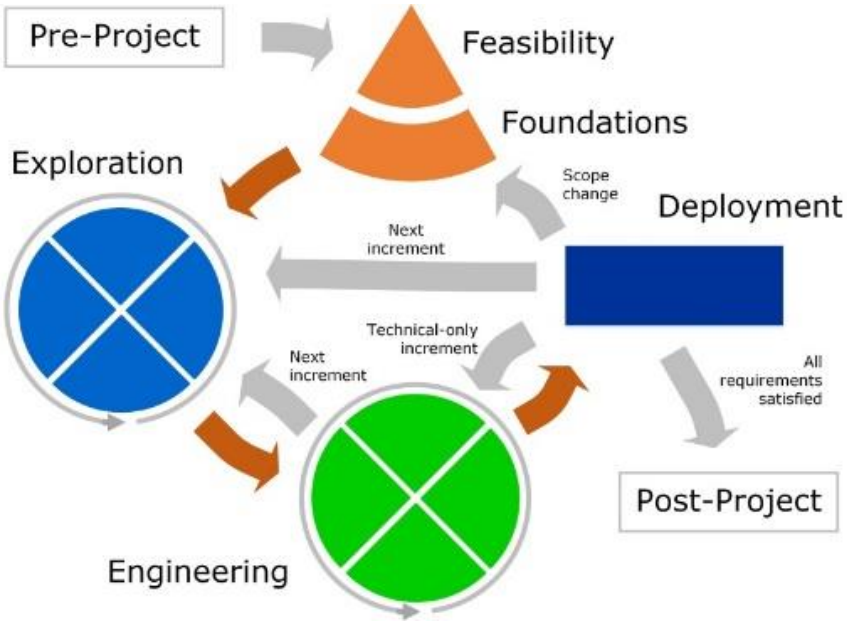
- Fokusiraj se na potrebe proizvoda (zahtjeve)
- Izbaci sve na vrijeme
- Surađuj
- Nikada nemoj ugroziti kvalitetu
- Gradi inkrementalno od stabilnih temelja
- Razvijaj iterativno
- Komuniciraj stalno i jasno
- Demonstriraj kontrolu

DSDM za razliku od ostalih agilnih metoda stavlja kvalitetu i vrijeme na prvo mjesto. Čak stavlja kvalitetu ispred funkcionalnosti! Premda on je agilna metoda zbog inkrementalnog iterativnog razvoja i izražene vrijednosti na komunikaciji s ljudima, očito je da dijeli dosta karakteristika i sa standardnim planskim razvojem softvera. Dokumentacija i propisi kod DSDM-a su puno stroži nego kod ostalih agilnih metoda te zbog toga on može imati vrlo zanimljive upotrebe.

Navedimo koje su propisane tehnike DSDM-a kojima se ostvaruju gore navedeni principi:

- Time-boxing – sve je u vremenskim okvirima u kojima se MORA odraditi
- MoSCoW – tehnika prioritiziranja rada, naziv je dobila po: MUST, SHOULD, COULD, WON'T
- Prototipiranje
- Testiranje
- Modeliranje

Bez ulaska u previše detalja samog DSDM-a naglasimo da je on već pomalo stara tehnika, nastao negdje u razvoju agilnih metoda. Ukratko, slika ispod demonstrira kako izgleda DSDM razvoj.


Pre-Project
Feasibility
Foundations
Exploration
Scope change
Deployment
Next increment
Technical-only increment
Next increment
All requirements satisfied
Post-Project
Engineering

6. Zaključak

Na kraju, kada pogledamo sve dosada napisano, jasno je da svijet razvoja današnjeg softvera ne bi mogao postojati bez agilnih metoda. SCRUM, Crystal i DSDM se svi zalažu za komunikaciju i stavljaju veliki naglasak na same ljude. Inkrementalno iterativni razvoj uz stalnu komunikaciju s korisnicima se pokazuje kao bitno svojstvo za razvoj velikog broja današnjih softvera.

Premda ove metode nisu uvijek dobre te planski razvoj ima značajne prednosti u nekim pogledima, razvijači softvera moraju sagledati sve i procijeniti koja metoda im najbolje paše. Uz manje kompanije gdje se možda razvija samostalni softver koji se plasira na tržište i uz stalan tim, agilne metode bi bile idealne. Dok kod velikih projekata gdje sve treba biti zabilježeno i dobro provjereno, planski pristup je neupitan.

Konačno, postoje improvizacije koje koriste neke značajke planskog pristupa i neke značajke agilnih metoda. Kako je razvijač Crystala, Alistair Cockburn rekao:

> „To je grupa primjera koju sami prilagođavamo svojim okolnostima.“

Literatura

1. Ian Sommervile: Software Engineering, Ninth Edition
2. Ken Schwaber, Jeff Sutherland: Scrum Guide
3. https://www.scrum.org
4. https://activecollab.com/blog/project-management/crystal-methods
5. DSDM handbook
6. https://en.wikipedia.org/wiki/Dynamic_systems_development_method