Metodologije razvoja softvera

Agilno modelovanje

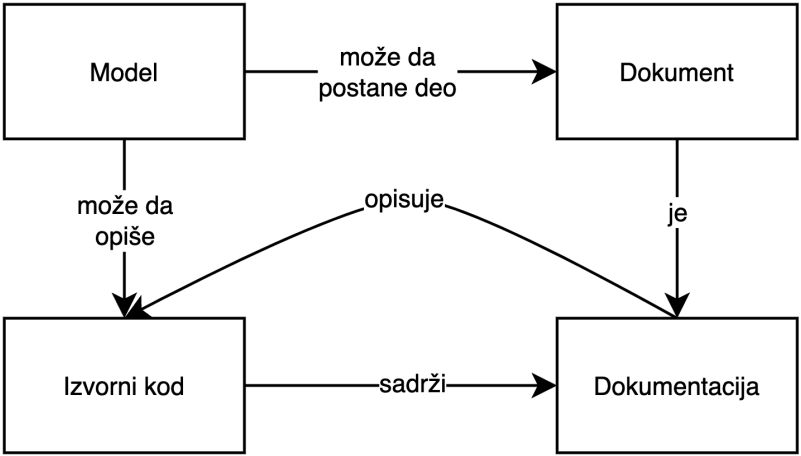
dr Milan Stojkov

Katedra za informatiku

2022.

Fakultet tehničkih nauka
Univerzitet u Novom Sadu

Modeli, dokumenti, dokumentacija, programski kod



Uvod

- Agilno modelovanje (AM) je metodologija bazirana na praksi za efektivnije modelovanje i dokumentovanje softverskih sistema
- AM metodologija predstavlja kolekciju principa za programere koji se primenjuju na dnevnoj osnovi
- AM ne definiše detaljne procedure kako kreirati određen tip modela, već pruža savete kako je moguće efektivnije modelovati

Zašto modelovati

- Dva osnovna razloga:
  - Da bi se razumelo šta se razvija. Modeluju se zahtevi sistema (npr. sa use-case dijagramima) ili poslovni procesi.
  - Razvijanje modela u cilju analize zahteva, da se definiše globalna arhitektura ili detaljan dizajn sistema
- U oba slučaja cilj je dobiti bolje razumevanje jednog ili više aspekata sistema koji se razvija/modeluje
- Model sistema olakšava i poboljšava komunikaciju unutar tima i komunikaciju tima i klijenta

Ciljevi agilnog modelovanja

- Tri osnovna cilja:
  1. Definisati i pokazati kako u praksi organizovati principe i postupke koji se odnose na modelovanje. Ideja AM nije smisljanje novih tehnika i modela za modelovanje već kako postojeće uspešno primeniti.
  2. Kako primeniti tehnike modelovanja na softverske projekte primenom agilnog pristupa – ponekad je mnogo produktivnije skicirati dijagrame i razmisliti o ideji nego odmah krenuti da se rešenje kodira
  3. Kako poboljšati aktivnosti modelovanja koristeći „gotovo agilan“ pristup, tj. primeniti agilno modelovanje na projektu koji nije agilan

Agilni modeli

- Model je apstrakcija koja opisuje jedan ili više aspekata problema ili potencijalno rešenje problema
- U tradicionalnom smislu pod modelima se podrazumevaju dijagrami i prateća dokumentacija
- Alternativne stvari poput kartica, tekstualnog opisa i sl. se takođe mogu smatraju modelima
- Agilni model je model koji je dovoljno dobar

Kada su agilni modeli dovoljno dobri I

- Agilni modeli su dovoljno dobri ako zadovoljavaju sledeće kriterujume:
  - Ispunjavaju svoju svrhu
  - Razumljivi su
    - Model zahteva može biti na jeziku razumljivom klijentu, a arhitektura može biti specificirana na jeziku razumljivom tehničkom osoblju. Bitno je da notacija modelovanja pozitivno utiče na razumevanje - npr. UML use-case dijagram nema nikakvog smisla ako korisnici ne razumeju njegovu notaciju. U tom slučaju ili treba koristiti drugi pristup ili korisnicima objasniti korišćenu tehniku.
  - Dovoljno su precizni
    - Vrlo često nije potrebno da model bude 100% precizan. Prihvatljiv nivo preciznosti često zavisi od prirode problema, konkretnog tima, organizacije, itd.
  - Dovoljno su konzistentni
    - Ne moraju biti savršeno konzistentni sa samim sobom ili sa nekim drugim elementima (modelima).

Kada su agilni modeli dovoljno dobri II

- Agilni modeli su dovoljno dobri ako zadovoljavaju sledeće kriterujume:
  - Dovoljno su detaljni
    - Dovoljan nivo zavisi od svrhe modela kao i od toga kome je namenjen, odnosno od konkretne situacije.
  - Obezbeđuju pozitivan efekat
    - Fundamentalan aspekt bilo kog dela projekta je da treba da obezbedi nekakav pozitivan efekat. Npr. ako trošak modela nadmašuje njegov benefit onda je pitanje da li je takav model potreban.
  - Treba da su što jednostavniji
    - Na jednostavnost primarno utiču detaljnost modela kao i notacija koja se koristi. Zbog toga se često koristi samo podskup celokupne notacije.

Šta je agilno modelovanje I

- AM je stav, a ne unapred definisan proces
  - AM opisuje stil modelovanja, koji kada se primeni adekvatno u agilnom okruženju rezultuje bržim razvojem softvera, boljim kvalitetom i pri tome se izbegavaju nerealna očekivanja
- AM predstavlja dodatak postojećim metodama; nije kompletna nova metodologija
  - Primarni fokus je na modelovanju, a sekundarni je na dokumentovanju. Tehnike AM bi trebalo da se koriste u cilju poboljšanja modelovanja, primarno u timovima koji koriste neku od agilnih metodologija (Scrum, Kanban, XP, ...)

Šta je agilno modelovanje II

- AM komplementarno je na druge tehnike modelovanja
  - Neke druge tehnike modelovanja sa sasvim drugim fokusom (npr. ICONIX) se mogu dopuniti agilnim modelovanjem u cilju boljih rezultata
- AM je postupak koji pruža uslove za efektivnije postizanje zahteva klijenata
  - Fokus AM je da bude efektivan što je moguće više
- AM je nešto što funkcioniše u praksi, nije samo akademska teorija
  - Cilj AM je da opiše tehnike koje će obezbediti efektivnije modelovanje sistema, ne zahteva da te tehnike budu strogo formalne
- AM je za svakog developera
  - AM principi i prakse su jednostavni, a veliki broj developera ih primenjuje od ranije, tako da prelazak na AM ne zahteve neki veliki trud

Šta je agilno modelovanje III

- AM ne predstavlja napad na dokumentaciju
  - I agilno modelovanje podržava kreiranje potrebne dokumentacije, pri čemu svrha te dokumentacije nije dokumentacija sama po sebi
- AM ne predstavlja napad na CASE (*Computer Aided Software Engineering*) alate
  - I kod AM se koriste CASE alati koje će obezbediti efektivniji rad developera

Vrednosti agilnog modelovanja

- Komunikacija
- Jednostavnost
- *Feedback*
- Smelost/hrabrost
- Smernost

Komunikacija

- Pokazalo se da mnogi problemi u razvoju softvera nastaju usled loše komunikacije
- Jedan od primarnih razloga modelovanja je baš poboljšati komunikaciju
  - Modelovanje pomaže u razumevanju ideja unutar samog tima da bi se na kraju izbeglo potencijalno pogrešno/suprotno tumačenje stvari

Jednostavnost

- Osnovni razlozi koji uzrokuju komplikovano softversko rešenje:
  - Suviše rana primena složenih šablona/obrazaca
  - Definisanje jako složene arhitekture zarad nekih potencijalnih funkcionalnosti koje se možda budu realizovale u budućnosti
  - Razvoj komplesne infrastrukture: komponente, okruženja, biblioteke

Feedback

- Nekoliko načina za dobijanje *feedback*-a za model
  - Timsko razvijanje modela omogućuje brz *feedback* od članova tima
  - Pregled/analiza modela sa klijentima. U najboljem slučaju model se razvija sa klijentima, alternativno nakon razvijanja treba da se proanalizira zajedno sa klijentima funkcionalnost koja se možda bude realizovala u budućnosti
  - Implementacija modela
  - Testiranje (*acceptance testing*)

Smelost/hrabrost

- Agilne metodologije podstiču blizak rad u timu, da postoji poverenje između članova tima i samopoverenje
- Podstiče se tim da donosi odluke iza kojih će stajati
  - Model/dokumentacija
  - Arhitektura
  - Implementacija
  - ...
- Jedan od fundamentalnih zahteva za uspešan agilni pristup

Smernost

- Najjednostavniji način da programeri imaju smernost je da priznaju da ne znaju baš sve
- Prilikom agilnog modelovanja bitno je razumeti (voditi računa) da svako u timu ima svoju ekspertizu
- Smernost dolazi do izražaja i u kontaktu sa drugim učesnicima na projektu (koji nisu direktno članovi tima)

Praktični saveti za AM

- Saveti iz prakse se mogu grupisati u 4 osnovne kategorije
  1. Iterativno i inkrementalno modelovanje
     - Primena adekvatnih modela
     - Paralelno kreiranje više modela
     - Prelazak na sledeći element
     - Modelovanje u malim inkrementalnim celinama
  2. Timski rad
     - Timsko modelovanje
     - Aktivno učešće klijenata
     - Kolektivno vlasništvo
  3. Jednostavnost
     - Kreiranje jednostavnog sadržaja
     - Jednostavan prikaz modela
     - Korišćenje jednostavnih alata
  4. Validacija
     - Testiranje
     - Dokazivanje implementacijom

Primena adekvatnih modela

- Ideja je da se kreiraju modeli koji su potrebni/adekvatni za dati slučaj
  - Svaki tip modela ima svoje dobre i loše osobine i stoga je pogodan za određene situacije dok za neke druge nije
- Bitna stvar AM je znati kada koji model (dijagram) kreirati i koliko detaljan treba da bude

Kreiranje nekoliko modela u paraleli

- Baš zato što svaki tip modela ima svoje prednosti i mane uglavnom nije dovoljno kreiranje jednog tipa modela
- Praksa je pokazala da je produktivnost mnogo veća kada se simultano radi na više različitih tipova modela koji se odnose na istu stvar, nego da se radi jedan po jedan sekvencijalno
  - Ovo se kosi sa nekim principima „tradicionalnog“ modelovanja

Prelazak na sledeći element

- U situacijama kada postoji zastoj u kreiranju odgovarajućeg tipa modela treba razmisliti o prelasku na neki drugi tip modela ili prelazak na sledeći zadatak
  - Čest slučaj je da se jednim tipom modela pokušva izmodelovati nešto što on ne podržava: npr. use-case dijagramom se pokušava opisati poslovni proces
  - Prelaskom na sledeći zadatak developer više nije u režimu „zaglavljen“.
  - Prelazak na novi zadatak često uzrokuje i to da se bolje shvati prethodni pa se onda uspešno reši.

Modelovanje u malim inkremantalnim celinama

- Ideja je da se ne modeluje čitav sistem odjednom, već da se radi u malim celinama u sesijama od 10 do 20 minuta
  - Neki duži periodi modelovanja (nekoliko dana) mogu da se po potrebi jave na početku projekta
- Princip je da se modeluje samo onoliko koliko je dovoljno da se može krenuti u programiranje

Timsko modelovanje

- Primenjuje se pravilo: „Dve glave su bolje od jedne“
- Pokazalo se da vrlo često čitav tim treba da radi zajedno kako bi se realizovali kvalitetani i korisni osnovni (bazni) modeli kritični za projekat
  - Npr. Da bi se realizovao model arhitekture čitavog sistema potrebno je rešenje sa kojim će se svi članovi tima složiti i koje će biti što je moguće jednostavnije

Aktivno učešće klijenata

- Blisko povezano sa praksom timskog modelovanja
  - Da bi model bio efikasan i da bi se dobio brz *feedback* potrebno je učešće klijenata i svih drugih zainteresovanih strana na projektu
- Posledica učešća svih zainteresovanih strana
  - Korisnici treba da se pripreme da dele poslovnu logiku sa timom, da se definiše opseg projekta i prioritet zahteva
  - Viši menadžeri imaju bolji uvid u projekat
  - Ostali timovi koji su uključeni na projekat na ovaj način lakše integrišu svoja rešenja sa rešenjima drugih timova
  - Tim koji će biti zadužen za održavanje na ovaj način će biti bolje upoznat sa sistemom pa će postupak održavanja biti lakši

Kolektivno vlasništvo

- Uključivanjem više ljudi veća šansa je da se identifikuju različiti aspekti kod modelovanja
- Omogućuje da manje iskusni članovi tima steknu znanje u modelovanju
- Sprečava negativne efekte poput „Tvoj model nije dobar“
  - Nema presonalizacije, svi zajedno rade na modelu

Kreiranje jednostavnog sadržaja

- Da bi model bio što jednostavniji zahteva se da analiza, arhitektura i dizajn treba da budu što jednostavniji, a da opet ispune svoju svrhu i zahteve klijenata
- Saveti za kreiranje što jednostavnijeg modela
  - Model sadrži samo potrebne entitete i samo neophodnu komunikaciju između njih
  - Model ne treba da sadrži duplicirane vrednosti
  - Model treba da ima što je moguće manje elemenata

Jednostavan prikaz modela

- Neka osnovna pravila za vizuelno kreiranje modela
  - Izbegavati ukrštene linije
  - Izbegavati krive linije
  - Izbegavati dijagonalne linije
  - Izbegavati elemente različite veličine
  - Izbegavati previše elemenata (7 ± 2)
  - Izbegavanje nepotrebnih detalja

Testiranje

- Kako testirati?
- Kada testirati?
- Šta testirati?
- Agilno testiranje preporučuje rano i često testiranje

Dokazivanje implementacijom

- Model predstavlja apstrakciju nekog sistema
- Da bi se zaista utvrdilo da li je on dobar potrebno je napraviti odgovarajuću implementaciju
- Preporuka
  - Malo modeluj, pa kodiraj i zatim testiraj

Dodatni saveti za AM

- Produktivnost
  - Primena standarda za modelovanje
  - Pažljiva primena šablona/obrazaca
  - Ponovno korišćenje postojećeg materijala
- Dokumentovanje modela
  - Odbacivanje privremenih modela
  - Ažuriraj samo kad je neophodno

Primena standarda za modelovanje

- Osnovna ideja je da developeri treba da se dogovore i da poštuju neki skup standarda za modelovanje
  - Slično kao što se poštuju i stnadradi prilikom programiranja
- Najčešće korišćen standard je UML
  - UML ne uključuje modele za sve aspekte (npr. UI izgled)

Pažljiva primena obrazaca

- Ako se obrasci primenjuju po svaku cenu finalno rešenje može vrlo lako da se zakomplikuje što se koči sa jednim od principa AM – *Jednostavnost*

Ponovno korišćenje postojećeg materijala

- Isto kao i kod programiranja, i kod modelovanja moguće je iskoristiti postojeće modele
  - Smanjuje troškove
  - Ubrzava rad
- Treba biti pažljiv da se ne koriste zastareli modeli

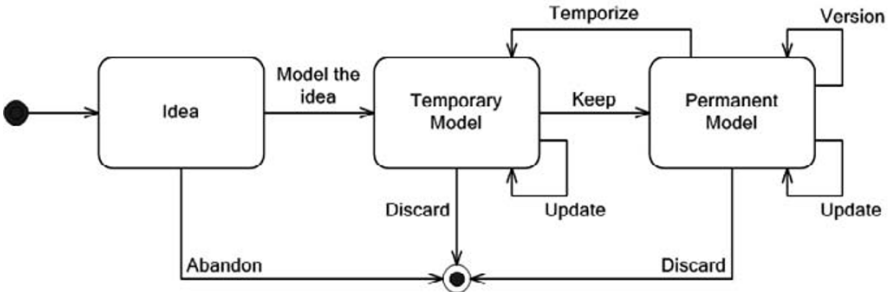
Odbacivanje privremenih modela I

- Veliki broj modela koji se kreiraju su privremeni-radni modeli
  - Modeli koji ispune svoju svrhu i potom više nemaju neku vrednost
- Modeli vrlo često i brzo postanu nesinhronizovani sa kodom. U tom trenutku treba doneti odluku da li ažurirati modele ili ih jednostavno odbaciti u slučaju da investiranje u model nema smisla

Ažuriraj samo kad je neophodno

- Model i dokumentaciju generalno bi trebalo ažurirati samo u momentu kada to postane neophodno
  - Redovno ažuriranje modela, da je u svakom momentu sinhron sa kodom, je zamoran i vremenski zahtevan proces
- Ako je model zamenjen sa kodom onda je vrlo verovatno da je model ispunio svoju funkciju i da su ga programeri razumeli tako da nije neophodno da se baš odmah ažurira

Životni tok agilnih modela



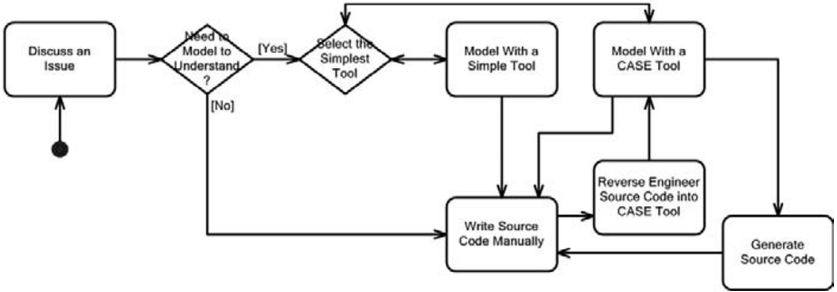
Slika preuzeta iz: *Agile modeling: effective practices for extreme programming and the unified process*, Scott Ambler, John Wiley & Sons, 2002.

Zablude AM

- Model = Dokumentacija
- Sve se može identifikovati na samom početku projekta
- Modelovanje podrazumeva Heavy-Weight softverski proces
- Zahtevi moraju biti „zamrznuti“
- Dizajn je „zabetoniran“ na samom početku
- Moraju se koristiti CASE alati
- Modelovanje je gubljenje vremena
- Sve se okreće oko modelovanja
- Svi programeri znaju kako modelovati

Evolucija modela

- Kombinuju se jednostavni alati (npr. skica na papiru) i CASE alati



Slika preuzeta iz: *Agile modeling: effective practices for extreme programming and the unified process*, Scott Ambler, John Wiley & Sons, 2002.