
Lekcija 2 –
Modeli Životnog
Ciklusa Softvera

Koraci kroz koje prolaze svi programi

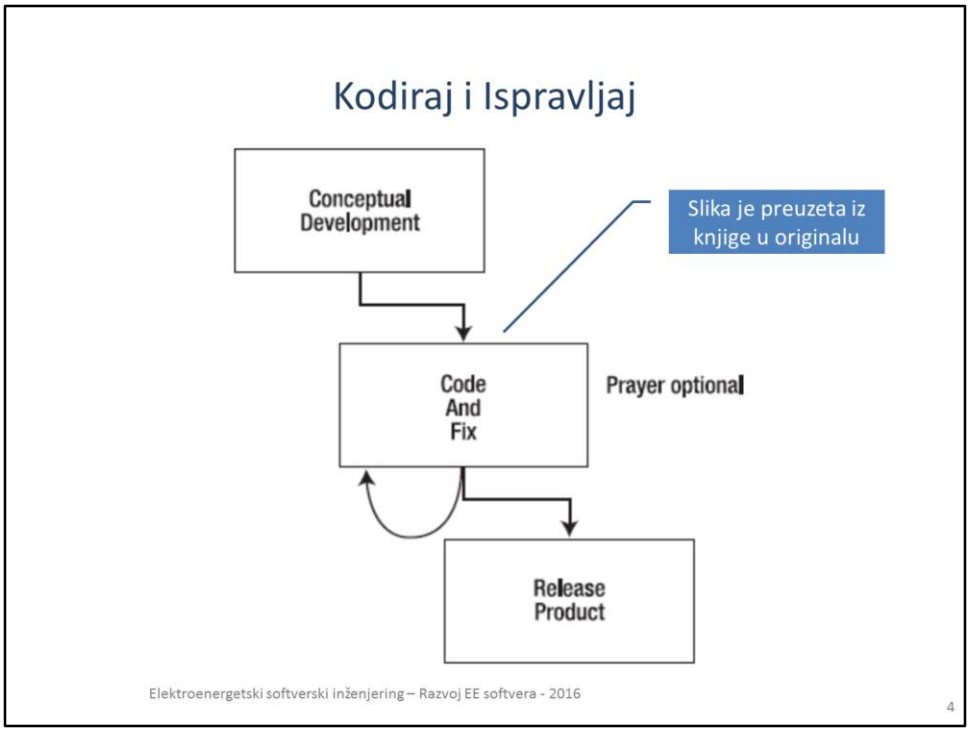
1. Koncepcija/Ideja
2. Prikupljanje zahteva/Istraživanje/Modelovanje
3. Dizajn
4. Konstrukcija i ispravljanje grešaka
5. Testiranje
6. Instalacija
7. Održavanje/evolucija
8. Arhiviranje

Ove korake ne treba shvatiti da se uvek izvršavaju u ovom redosledu. To zavisi od konkretnog procesa, koji se koristi. Međutim, konceptualno svi ovi koraci su eksplicitno ili implicitno prisutni. Ako neki korak nije eksplicitno podržan od strane razvojnog procesa, tada u tom delu se ne može postići poboljšanje niti reprodukcija u drugim situacijama.

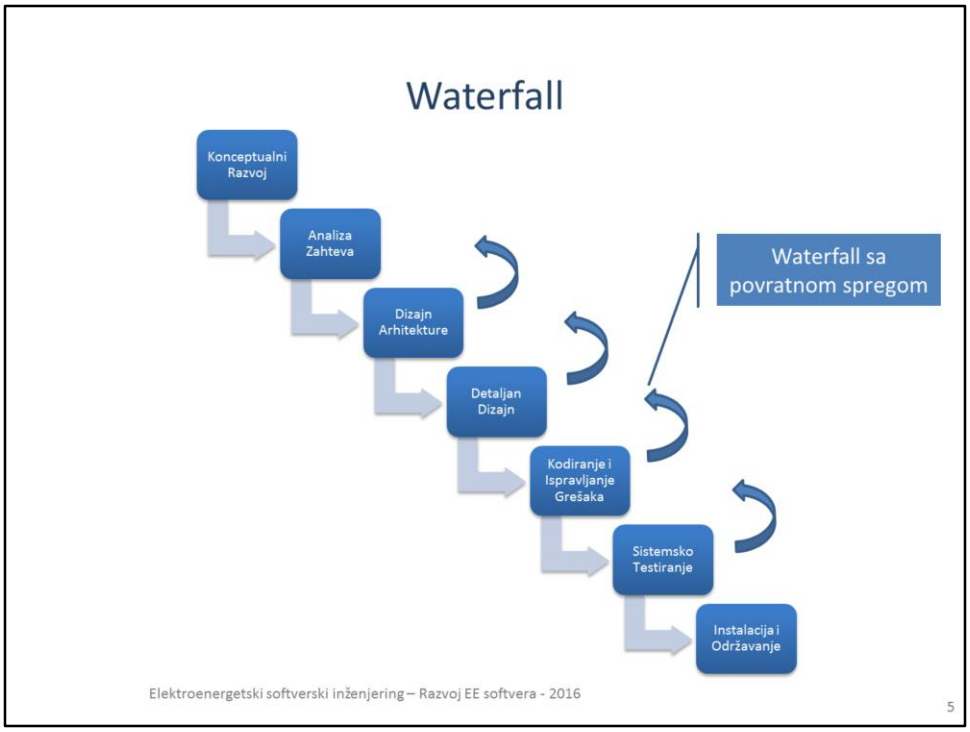
Rukovanje softverskim razvojnim projektima

- Plansko-rukovođeni procesi
- Agilne razvojne metode (Agile Manifesto: http://agilemanifesto.org)
- Ovde je citat iz knjige dat u originalu:

*"Each project must decide on the model that works best for its particular application and base that decision on the **project domain**, **the size of the project**, **the experience of the team**, and **the timeline of the project**."*



Ova metoda je dobra za brze i “prljave” solucije, pogotovo za izgradnju minimalnog mogućeg proizvoda (minimal viable product), koji je popularan u tzv. “lean” pristupima.



Klasični waterfall je fenomenalan kao teoretski model (okvir). Ono je nalik sedmoslojnom OSI mrežnom modelu. Njena realizacija nije nikad oživela u praksi, ali je dao dobar konceptualni okvir o računarskim mrežama.

Nijedna od ovde navedenih varijanti waterfall modela ne adresira probleme rasporeda (scheduling) niti stvari oko nepoznanica (uncertainty) u projektu.

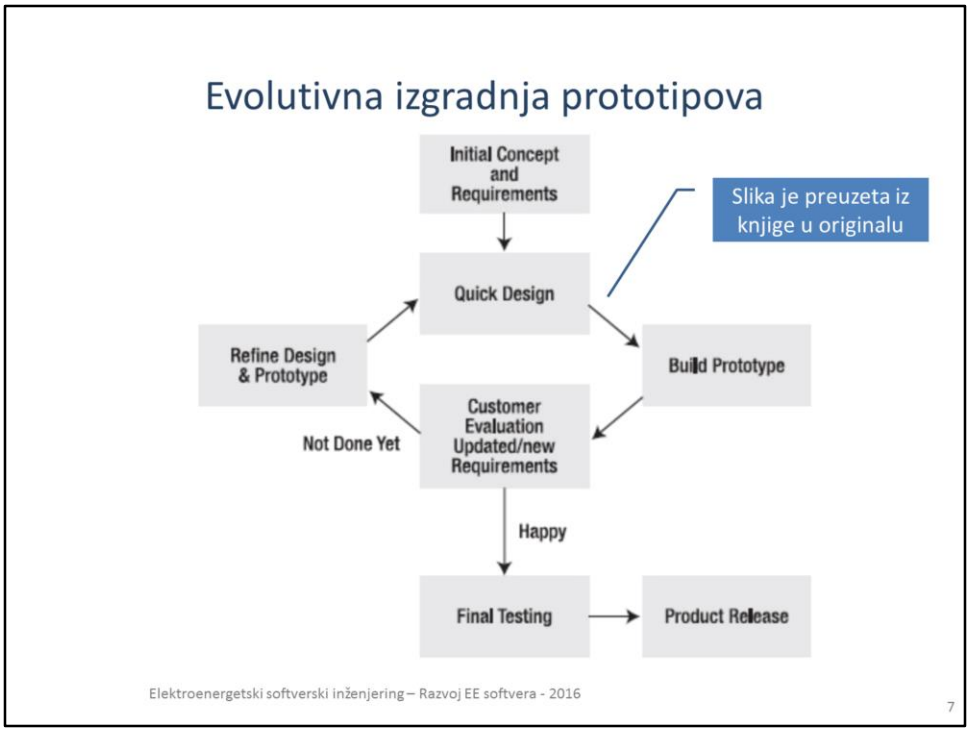
Iterativni i inkrementalni model

- Ovde je citat iz knjige dat u originalu:

"The best practice is to iterate and deliver incrementally, treating each iteration as a closed-end "mini-project," including complete requirements, design, coding integration, testing, and internal delivery. On the iteration deadline, deliver the (fully-tested, fully-integrated) system thus far to internal stakeholders. Solicit their feedback on that work, and fold that feedback into the plan for the next iteration."

www.adaptionsoft.com/on_time.html

Dok waterfall sa povratnom spregom prepoznaje činjenicu da nisu svi zahtevi unpared poznati, i da će se pojaviti greške u kasnijim fazama, ono se ne trudi da ova saznanja aktivno ugradi i u sam proces. Iterativni procesni modeli proaktivno ugrađuju sve aspekte promena u sam proces, pa čak definišu i vremena kada promene treba da se dese. Proizvod se izgrađuje korak po korak.



S obzirom da dizajn evoluira kako se menjaju zahtevi, postoji značajna šansa za lošim dizajnom, osim ako nema mogućnosti planskog redizajniranja – nešto što je sve teže i teže postići kako projekat odmiče i klijent sve više vezuje za tekuću verziju proizvoda.

Evolutivna izgradnja softvera pomoću prototipova najbolje funkcioniše sa striktnim rokovima, i iskusnim timom, koji je već radio zajedno na nekoliko prethodnih projekata. Ovo je slučaj sa većinom Agilnih metoda.

Kod ovog pristupa, inicijalni skup zahteva ima ogroman uticaj na samu arhitekturu. Ovo ne treba da čudi, jer drugačija prioritizacija zahteva, odnosno drugačiji njihov skup dovodi do različitih rešenja. Ovo je pogotovo naglašeno u tzv. use-case rukovođenim procesima (na primer, RUP). Arhitektura diktira aspekte daljeg održavanja prozivoda.

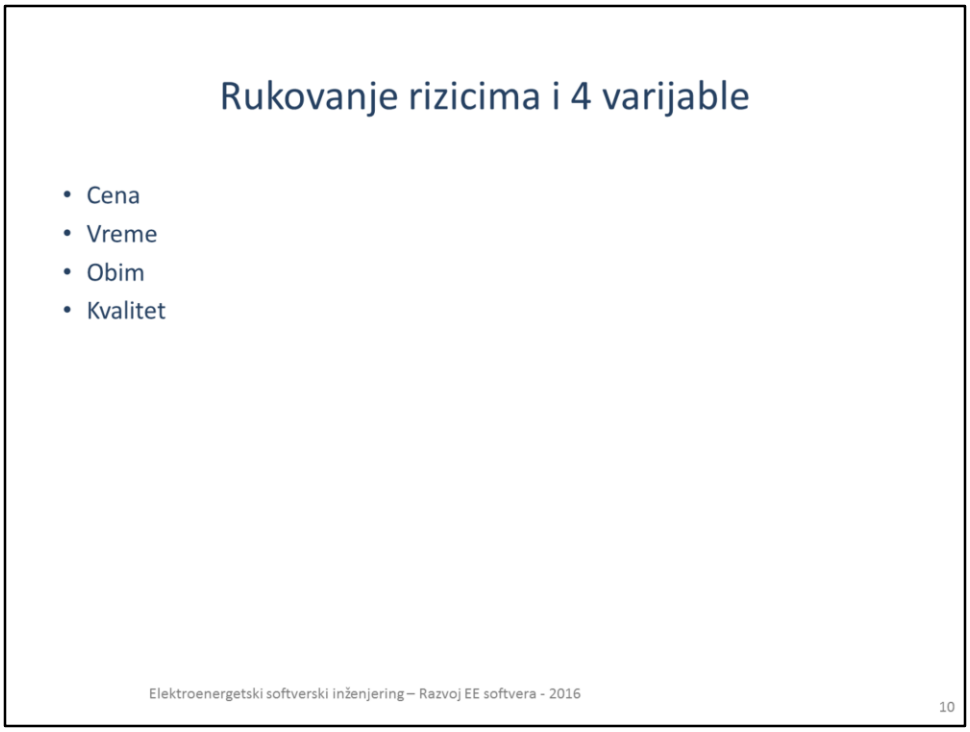
Lagane (lightweight) Agilne metode

- Naglašava pisanja testova pre samog kôda
- Frekventne instalacije proizvoda
- Značajna uloga klijenta tokom razvoja
- Zajedničko vlasništvo nad izvornim kôdom
- Refaktorizacija – kreiranje niza semantički ekvivalentnih izmena radi pojednostavljenja (poboljšanja) kôda

Dobar za male i srednje po veličini projekte. Zahteva ekstremnu disciplinu i energečni (agilni) tim.

eXtreme Programming (XP)

- Intenzivno učešće klijenta
- Test-driven razvoj (pogotovo kod unit testiranja)
- Programiranje u paru (pair programming)
- Kratke iteracije i česte instalacije



XP-ova motvicaija je redukcija rizika. U programiranju, jedina konstanta je sama promena. Ako se pažnja uvek obrati na promene i adekvatno njima rukuje, tada se može kontrolisati cena promena unutar predviđenih granica.

4 vrednosti

- Komunikacija
- Jednostavnost
- Povratna sprega
- Hrabrost i otvorenost (iskrenost)

Principi XP-a – deo 1

- Brze povratne informacije – pogotovo kroz automatizovane testove
- Pretpostavka jednostavnosti – refaktorizacija održava kôd u najjednostavnijoj formi i smanjuje greške
- Inkrementalne promene – kontinualna integracija
- Obuhvaćanje promena
- Kvalitetan rad
- Učiti učenje
- Mala incijalna ulaganja
- Raditi na uspesima
- Konkretni eksperimenti – koristi se *spike* (brzi eksperiment da se testiraju neke tehnologije i pretpostavke)

Principi XP-a – deo 2

- Otvorena i iskrena komunikacija – konstruktivne kritike ne bi trebale da uvrede nikoga
- Raditi u skladu sa instinktima ljudi, a ne protiv njih
- Prihvatanje odgovornosti
- Lokalna adaptacija
- Dislociranost uvećava kašnjenja u komunikaciji
- Iskrena merenja – tačnost i preciznost

4 osnovne aktivnosti

- Konstrukcija
- Testiranje
- Slušanje – znanje domena (klijent) i tehničko znanje (softverski inženjeri)
- Dizajniranje, ovde citiran iz knjige i dat u originalu:

"Designing is creating a structure that organizes the logic in the system. Good design organizes the logic so that a change in one part of the system doesn't always require a change in another part of the system. Good design ensures that every piece of logic in the system has one and only one home. Good design puts the logic near the data it operates on. Good design allows the extension of the system with changes in only one place."

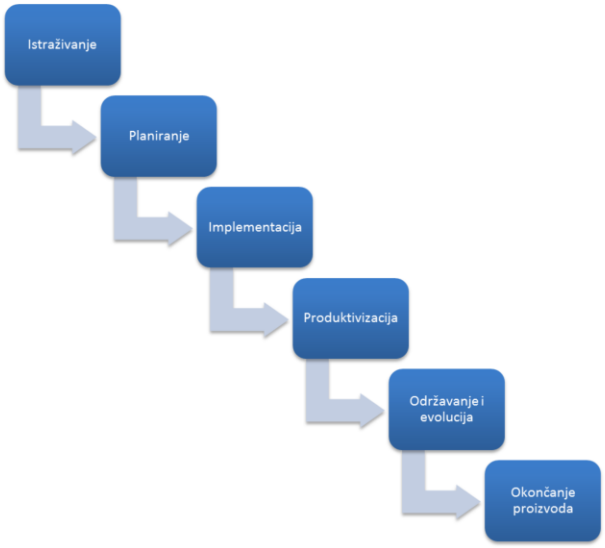
12 praksi – deo 1

- Igra planiranja (the planning game)
- Male instalacije
- Metafore – zamenjuje formalni opis arhitekture. Ovo treba da bude koherentan opis sistema, koji se može dekomponovati na manje delove – slučajeve korišćenja (user stories). Mora koristiti jezik razumljiv i za klijenta.
- Jednostavan dizajn
  - zadovoljava sve unit testove
  - nema dupliranog kôda
  - izražava šta koji slučaj korišćenja znači preko kôda
  - ima minimalan broj klasa i metoda, koje imaju smisla u kontekstu slučajeva korišćenja obuhvaćenih do datog trenutka
- Testiranje
- Refaktorizacija
- Programiranje u paru
- Zajedničko vlasništvo
- Kontinualna integracija

12 praksi – deo 2

- 40-satna nedelja
- Klijent deo razvojnog tima
- Standardi za konstrukciju kôda

Životni ciklus XP-a



Scrum

- Metodologija za rukovanje projektima, ali koristi prakse XP-a
- Iteracija se zove *sprint*
- Postoje dva skladišta (backlogs): proizvodni (product) i iterativni (sprint)
- Glavne uloge: Scrum master, Product owner i Developer
- Regularni scrum sastanci:
  - Koje zadatke si završio od poslednjeg Scrum sastanka?
  - Da li ima nešto što te blokira?
  - Šta planiraš da uradiš do narednog Scrum sastanka?
- Sličan životni ciklus (faze) kao kod XP-a