Softversko inženjerstvo

Softverski procesi i metodologije razvoja softvera

dr Miloš Stojanović*

Visoka tehnička škola strukovnih studija Niš

2017.

*milos.stojanovic@vtsnis.edu.rs

Uvod

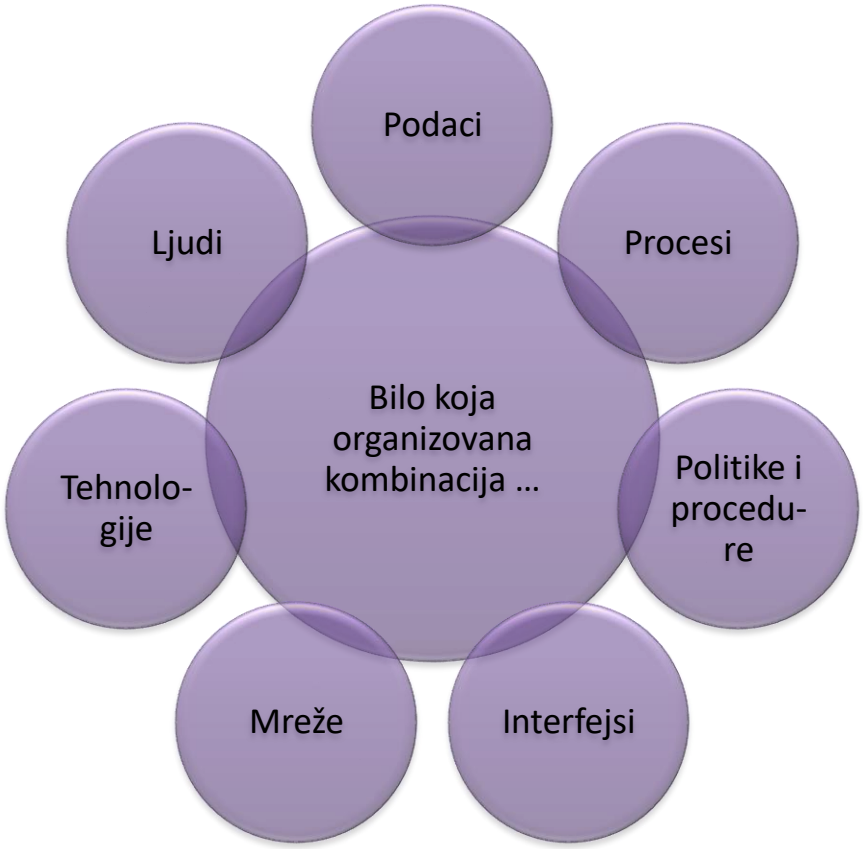
- Softversko inženjerstvo možemo definisati kao sistemski pristup razvoju, korišćenju, održavanju i odumiranju softvera.
- Pored isporuke softvera, visok kvalitet, mali troškovi i kratak vremenski rok su dodatni ciljevi koje softverski inženjer mora da dostigne.
- Kvalitet i produktivnost konačnog proizvoda zavise od veštine ljudi uključenih u projekat, **procesa** koji koriste za različite zadatke u projektu i alata koje koriste.
- **U softverskom inženjerstvu glavni fokus je dat na procesima.**
- Osnovni zadatak procesa je da pomogne ljudima da postignu veći kvalitet i produktivnost kroz specifikaciju zadataka koje treba uraditi i definisanjem načina na koji ih treba uraditi.

Životni ciklus razvoja softvera

- Šta je životni ciklus razvoja softvera? (Eng. System Development Life Cycle – SDLC)
- Proces kroz koji stručnjaci različitih profila (analitičari, projektanti, inženjeri, programeri, ...) i korisnici informacionog sistema prave informacioni sistem.

Šta je informacioni sistem (IS)?

- Uređeni i integrisani skup podataka, procesa, interfejsa, mreža, tehnologija i ljudi koji su u međusobnoj korelaciji u cilju podrške i poboljšanja svakodnevnih poslovnih operacija i podrške menadžmentu u rešavanju poslovnih problema i donošenja odluka.



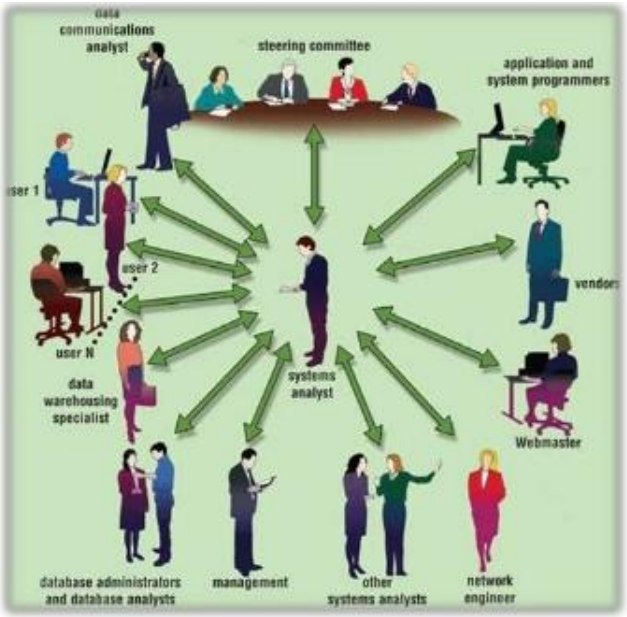
Koje su preporuke za razvoj sistema?

- Grupisanje zadataka po fazama (grupama aktivnosti)
- Uključivanje korisnika (lice za koje se sitem implementira)
- Razvoj jasno definisanih standarda (procedure za koje kompanija očekuje da zaposleni treba da ih primenjuje)

Ko su učesnici u životnom ciklusu razvoja softvera?

- Sistem analitičar (ili arhitekta sistema, sistem inženjer …) premošćava komunikacioni jaz između onih kojima trebaju informacioni sistemi i onih koji dobro poznaju tehnologije.
- Osoba odgovorna za dizajn i razvoj informacionog sistema
- Veza između korisnika

  i IT profesionalaca



Ko su učesnici u životnom ciklusu razvoja softvera?

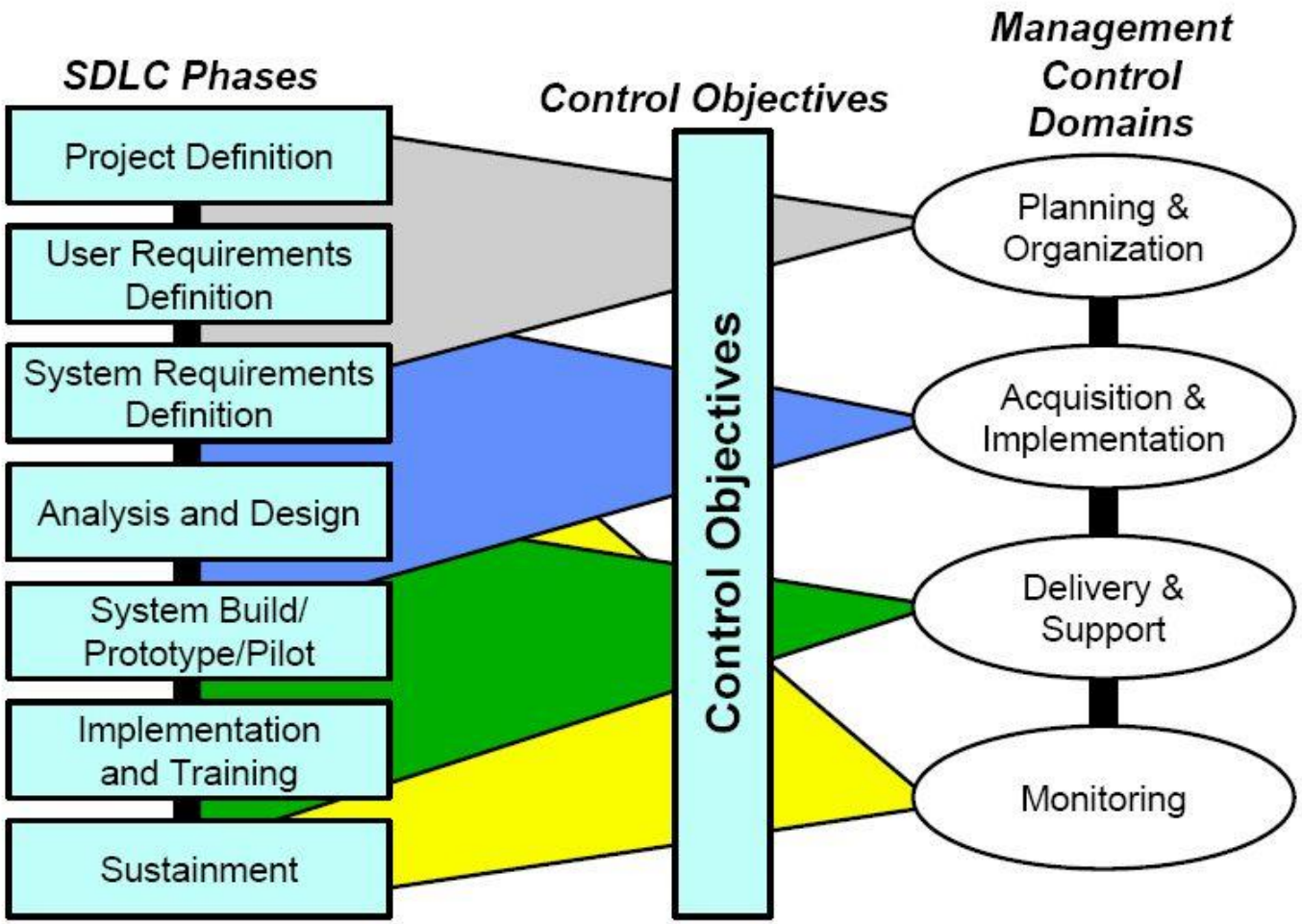
- Šta je projektni tim?

➢ Grupa ljudi koja radi na projektu od početka do kraja

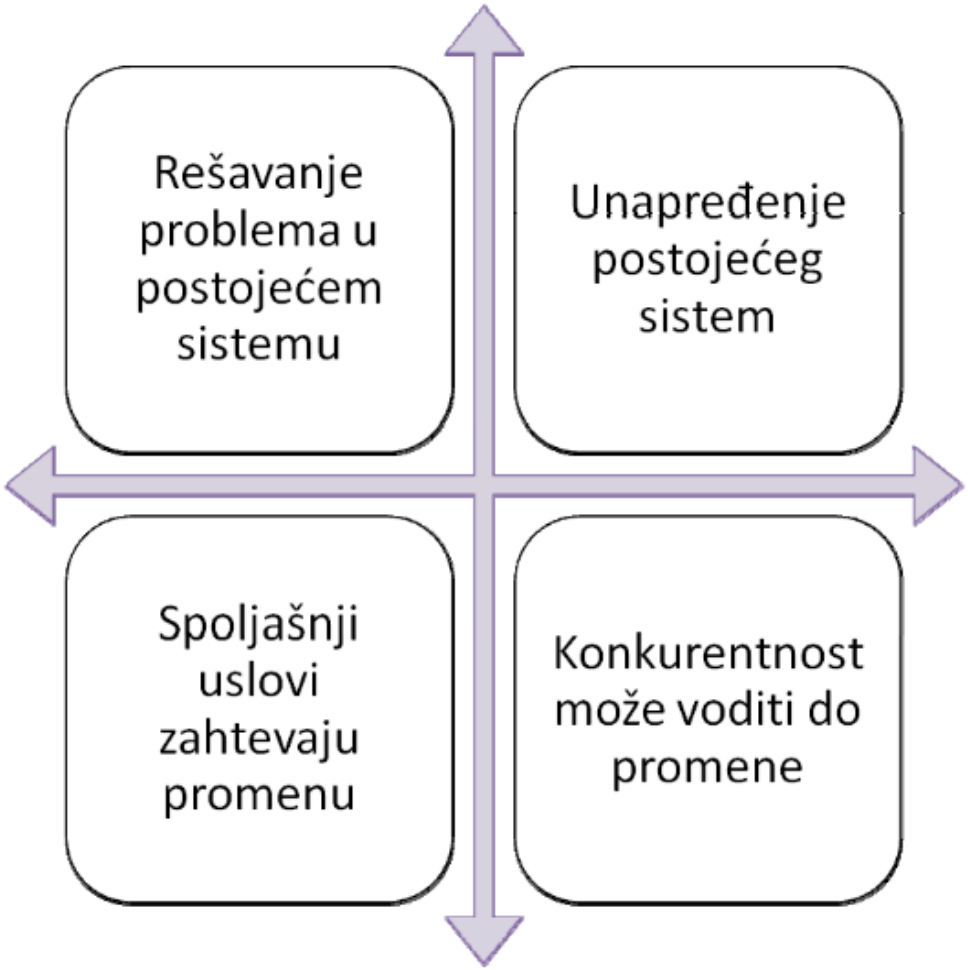
➢ Sastoji se od korisnika, analitičara sistema i drugih IT profesionalaca

➢ Vođa projekta – jedan od članova tima koji upravlja i kontroliše budžetom projekta i vremenskim planom

Koje su faze životnog ciklusa razvoja softvera?

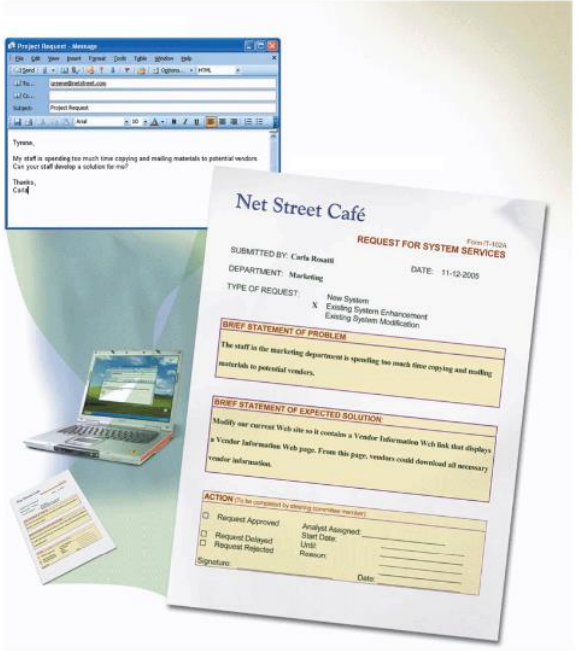


Koji su razlozi za kreiranje ili modifikovanje informacionog sistema?



Šta je zahtev za održavanjem sistema?

- Formalni zahtev za novim ili modifikovanim informacionim sistemom.
- Takođe zvan projektni zahtev.



Šta je faza planiranja?

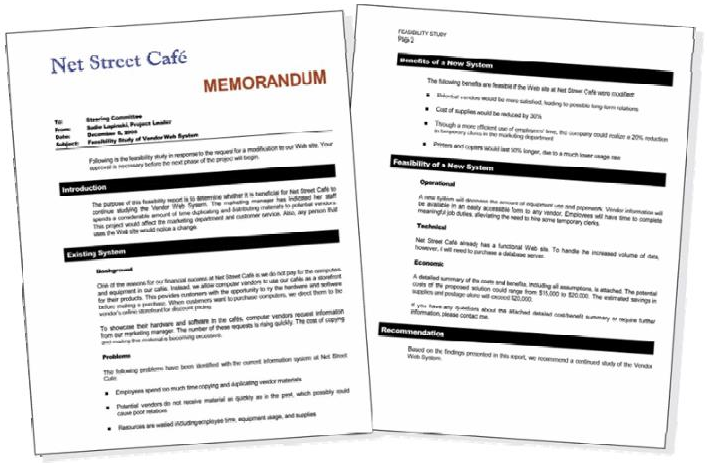
- Počinje kada upravni odbor primi projektni zahtev
  - Upravni odbor – rukovodeće telo kompanije
- Funkcije odbora:
  - Recenzija i odobravanje projektnih zahteva
  - Određivanje prioriteta projektnih zahteva
  - Dodeljivanje resursa
  - Formiranje razvojnog tima za svaki odobreni projekat

Šta je faza analize?

- Izvršenej preliminarnog istraživanja – studija izvodljivosti
- Izvršenje detaljne analize

Šta je preliminarno istraživanje?

- Utvrđivanje tačne prirode problema ili unapređenja i utvrđivanje da li je vredno truda.
- Zaključci se prezentuju u izveštaju o izvodljivosti, poznatom kao studija izvodljivosti.



Šta je detaljna analiza?

- Proučavanje kako trenutni sistem funkcioniše
- Utvrđivanje šta korisnik želi, šta mu je potrebno i definisanje zahteva
- Predlaganje rešenja

Šta su moguća rešenja?

- **Kupovina** paketskog softvera – dostupan na tržištu:
  - Za horizontalno tržište – za potrebe mnogih kompanija
  - Za vertikalno tržište – za određenu industriju
- **Izrada** sopstvenog softvera – razvijen prema korisničkim zahtevima
- **Outsource** – “spolja” razvijen softver

Šta je faza dizajna?

- Definiše se arhitektura sistema (komponente, njihov interfejs i ponašanje):

➢ Dokument koji opisuje arhitekturu

➢ Plan implementacije

➢ Analiza osnovnih prioriteta

➢ Plan testiranja

Šta je detaljni dizajn?

- Detaljna specifikacija dizajna za komponente u predloženom rešenju:

➢ Dizajn baze podataka

➢ Dizajn ulaza i izlaza

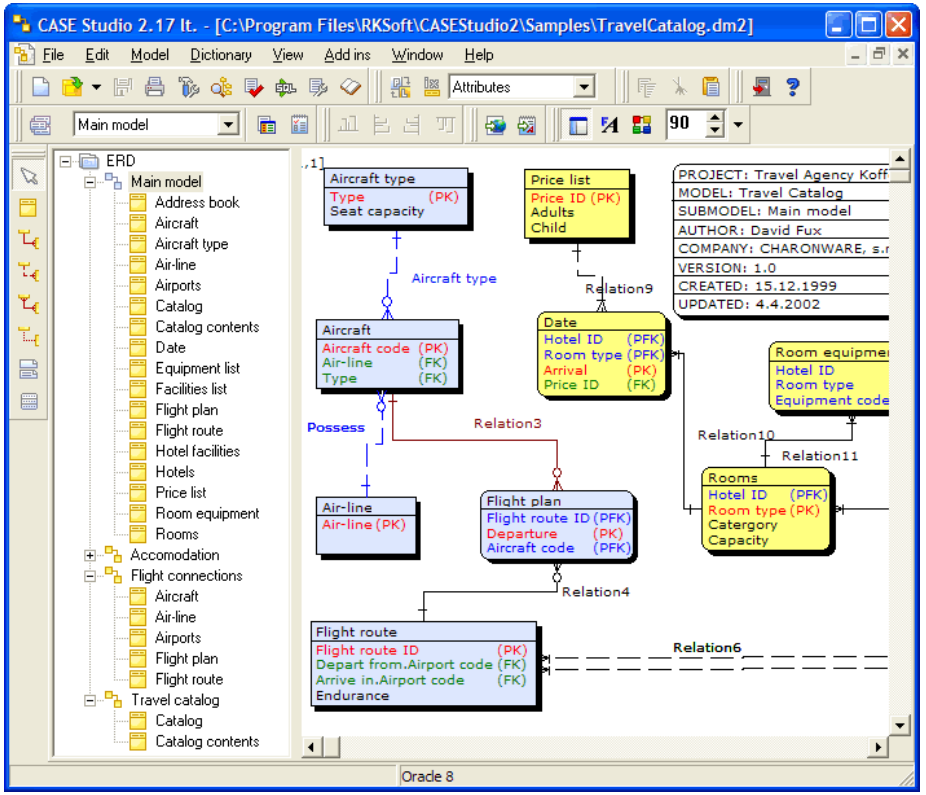
➢ Dizajn programa

Šta je faza implementacije?

- Prelazak na novi sistem
- Obuka korisnika
- Instalacija i testiranje novog sistema

Šta je computer-aided software engineering (CASE)?

- Softverski alati dizajnirani da podrže aktivnosti životnog ciklusa razvoja softvera.



Šta je prototip?

- Nekompletna verzija softvera koji se razvija.
- Obično simulira samo nekoliko aspekata funkcionalnosti eventualnog softvera i može biti potpuno drugačiji od kranjeg proizvoda.
- Omogućava korisnicima softvera da evaluiraju ono što su developeri predložili kao rešenje.
- Izrada prototipa previše rano može dovesti do problema.

Koja su tri tipa testova koje izvršava projektni tim?

1. “Jedinično” - komponentno testiranje – proverava da li svaka pojedinačna komponenta sistema funkcioniše.
2. Testiranje sistema – proverava da li svi delovi sistema (komponente) rade zajedno kao celina.
3. Testiranje integracije – poverava da li sistem kao celina funkcioniše ispravno sa drugim sistemima.

Šta je faza podrške?

- Obezbeđuje tekuću pomoć nakon implementacije sistema:

➢ Upravljanje analizom sistema nakon implementacije

➢ Identifikovanje grešaka

➢ Identifikovanje potencijalnih unapređenja

➢ Praćenje performansi sistema

Šta je obuka?

- Pokazivanje korisnicima kako će koristiti hardver i softver u sistemu.

Softverski proces

- Struktuiran skup aktivnosti neophodan za razvoj softverskog sistema.
- Aktivnosti zajedničke za različite procese:
  - Specifikacija – definisanje šta sistem treba da radi;
  - Projektovanje i implementacija – definisanje organizacije sistema i njegova implementacija;
  - Validacija – provera da li sistem radi ono što naručioc želi;
  - Evolucija – promena sistema kao odgovor na promenu potreba naručioca.
- Model softverskog procesa predstavlja apstraktnu reprezentaciju procesa.

Softverski proces

- Realni softverski procesi predstavljaju isprepletenu mešavinu tehničkih, kolaborativnih i menadžerskih aktivnosti čiji je krajnji cilj specifikacija, projektovanje, implementacija i testiranje softverskog sistema.
- Četiri osnovne aktivnosti softverskog procesa specifikacija, razvoj, validacija i evolucija su različito organizovane u različitim procesima.

Planom vođeni i agilni procesi

- Planom vođeni procesi su procesi kod kojih se sve aktivnosti unapred planiraju, te se i progres meri na osnovu tog plana.
- Kod agilnih procesa planiranje je inkrementalno, pa je lakše promeniti proces nakon promene zahteva naručioca.
- U praksi najpraktičniji procesi uključuju elemente kako planom vođenog tako i agilnog pristupa.
- Ne postoje dobri i loši softverski procesi.

Šta je model procesa

- Razvojni plan koji definiše opšti proces razvoja softverskog proizvoda: koje aktivnosti se izvršavaju, od strane kojih osoba u kojim ulogama, redosled aktivnosti, koji proizvodi će biti razvijani i kako će se proceniti njihov kvalitet, ...

Pregled postojećih modela softverskih procesa

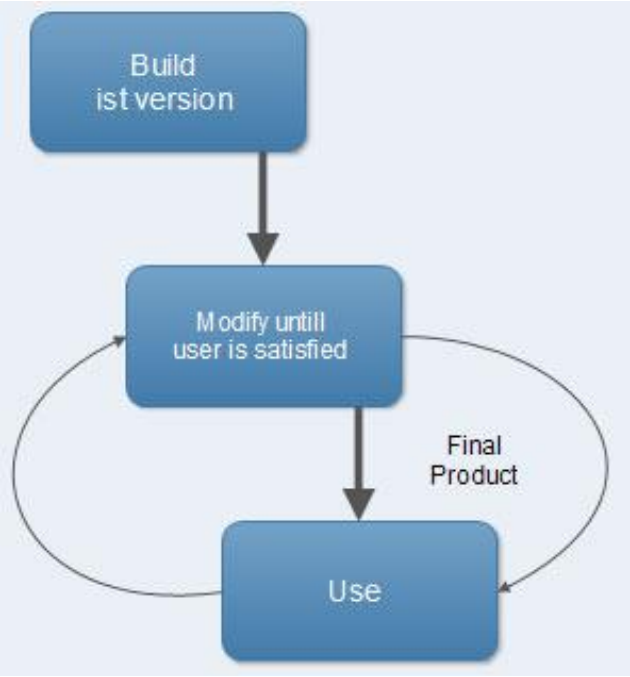
- Šta je metodologija razvoja softvera?
- **Metodologija je formalizovani proces** ili skup iskustava za kreiranje softvera
- Skup pravila koje razvojni tim prati
- Skup konvencija koje je organizacija odlučila da prati
- Sistematičan, inženjerski pristup za organizovanje softverskih projekata

Metodologije razvoja

- Među najpoznatije metodologije spadaju:
  - Waterfall model
  - Waterfall model sa prototipom
  - Metodologija za paralelni razvoj
  - V model
  - Inkrementalni model
  - Spiralni model
  - Rapid Application Development (RAD)
  - Agilne metodologije
  - Rational Unified Process (RUP)

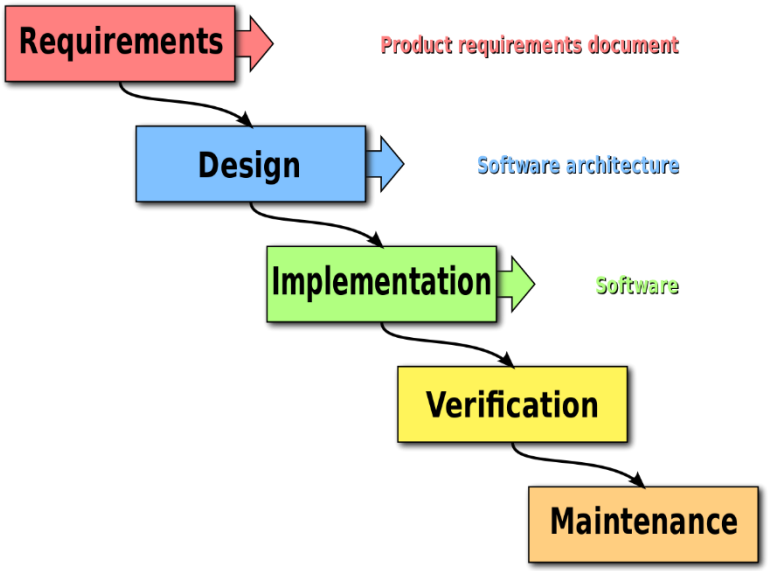
Build and Fix model ☺

- Većina softvera je razvijena korišćenjem ovog modela.
- U suštini ovde nema modela, specifikacije, ni dizajna.



Waterfall model

- Jedan od najstarijih modela
- Dobar za rešavanje potpuno jasnih problema bez izmena u zahtevima
- Jednostavan i razumljiv za korisnika

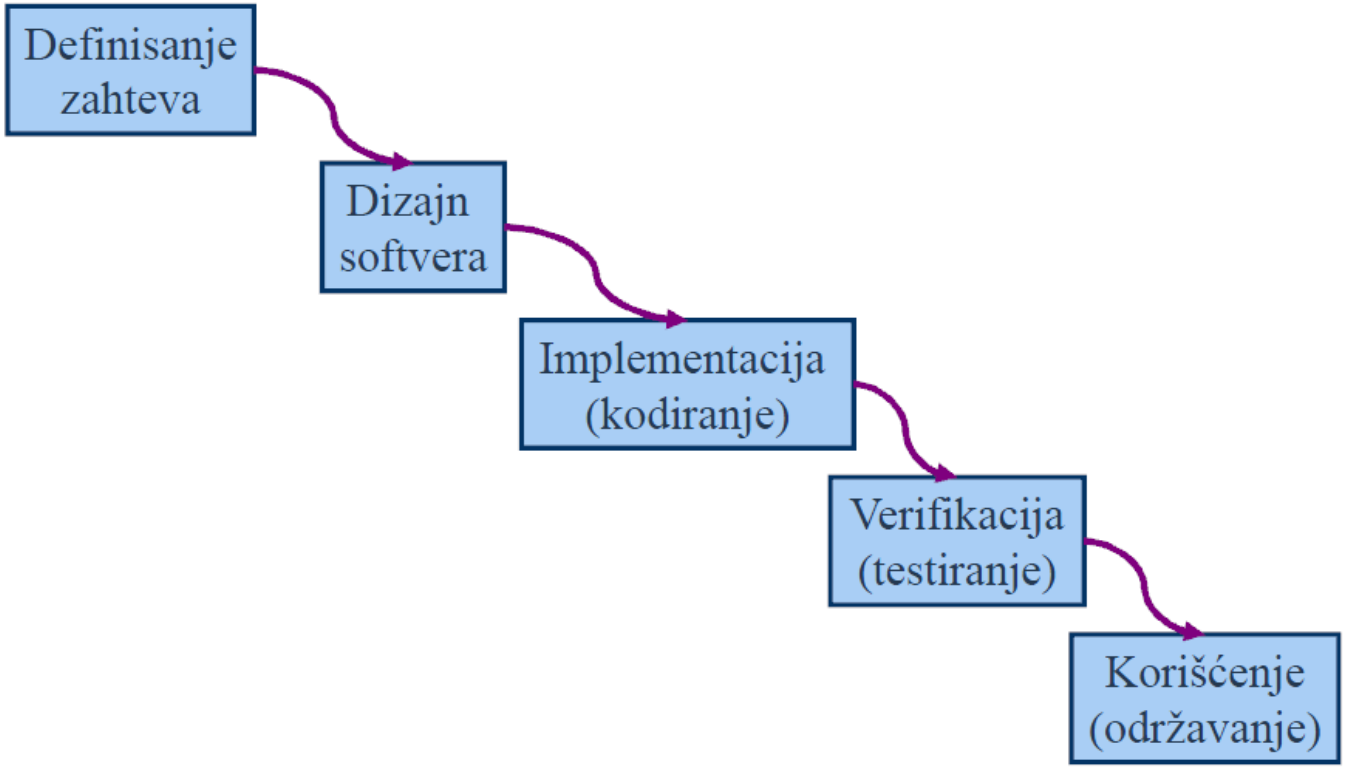


Waterfall model

- Predstavlja vrlo apstraktan prikaz razvojnog procesa opisan kroz niz aktivnosti
- Jasne granice između faza sa izlaznim produktima
- Naziva se još i linearno-sekvencijalni model.
- Svaka faza mora biti u potpunosti završena pre nego što započne sledeća faza.
- Na kraju svake faze, vrši se analiza da bi se utvrdilo da li je projekat na dobrom putu i da li se treba nastaviti ili obustaviti.
- Ova analiza se naziva završna analiza faze.

Waterfall model

- Predstavlja tradicionalni pristup razvoju:



Waterfall model

- Prednosti:

➢ Svaka faza ima specifične izlazne produkte i proces analize.

➢ U jednom trenutku radi se samo na jednoj fazi.

➢ Dobar je za projekte kod kojih su potpuno definisani i jasni zahtevi.

➢ Potkrepljuje stav “definiši pre dizajna” i “dizajniraj pre implementacije”.

Waterfall model

- Nedostaci:

➢ Nemogućnost efikasnog prihvatanja izmena u zahtevima korisnika kada je proces u toku.

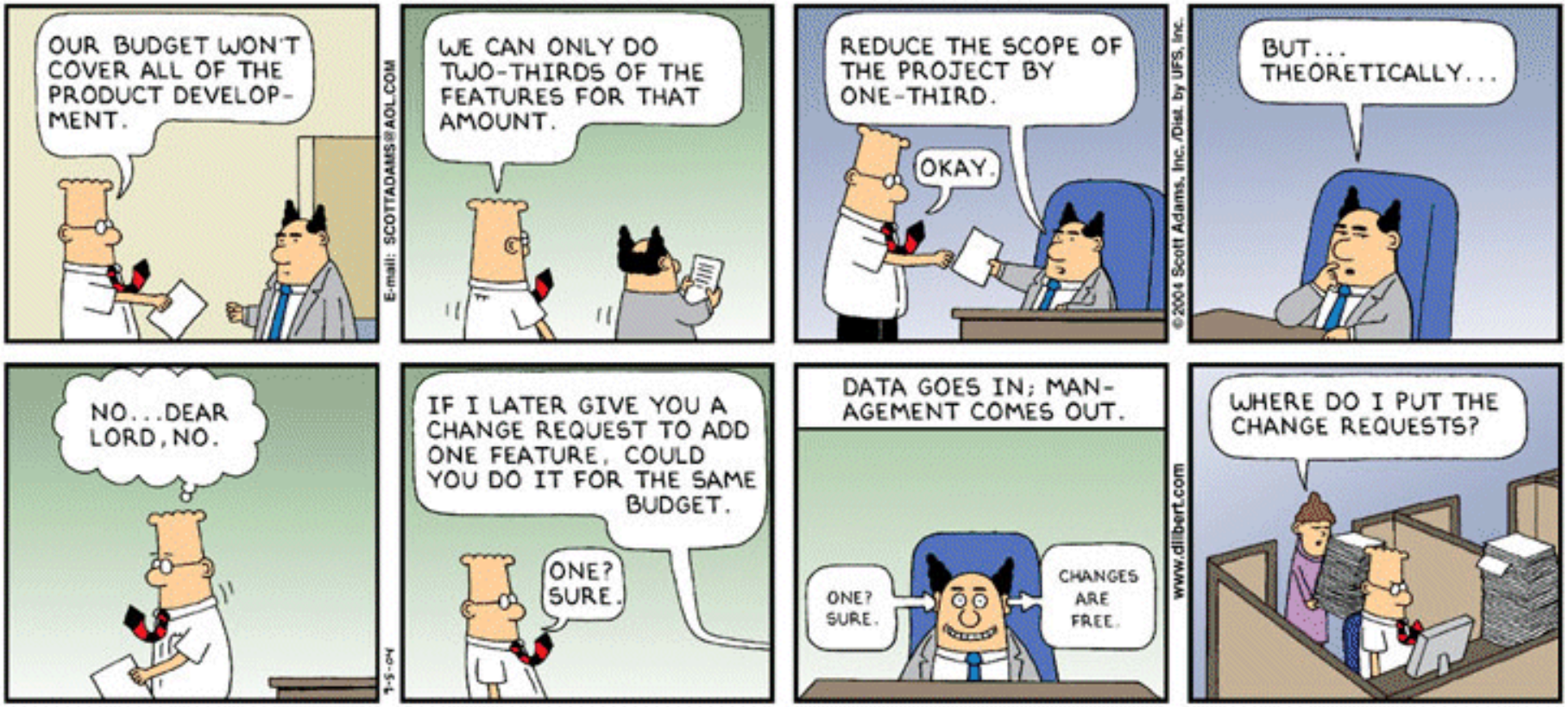
➢ Jedna faza mora biti završena da bi se otpočela naredna, pa izmene zahteva resetuju čitav proces.

➢ Nefliksibilna deoba projekta na disjunktne faze otežava odgovor na promenu korisničkih zahteva.

Waterfall model

- Model vodopada je jedino prihvatljiv kada su zahtevi dobro definisani i usaglašeni na početku projekta sa malom verovatnoćom izmene u toku razvoja.
- Najčeće zahtevi nisu u toj meri jasni na početku razvoja.
- Model vodopada se najčeće koristi kod velikih sistema gde je razvoj razdeljen na nekoliko lokacija.
- U ovim situacijama planom vođena priroda modela vodopada pomaže u koordinisanju posla.

Promena korisničkih zahteva u praksi



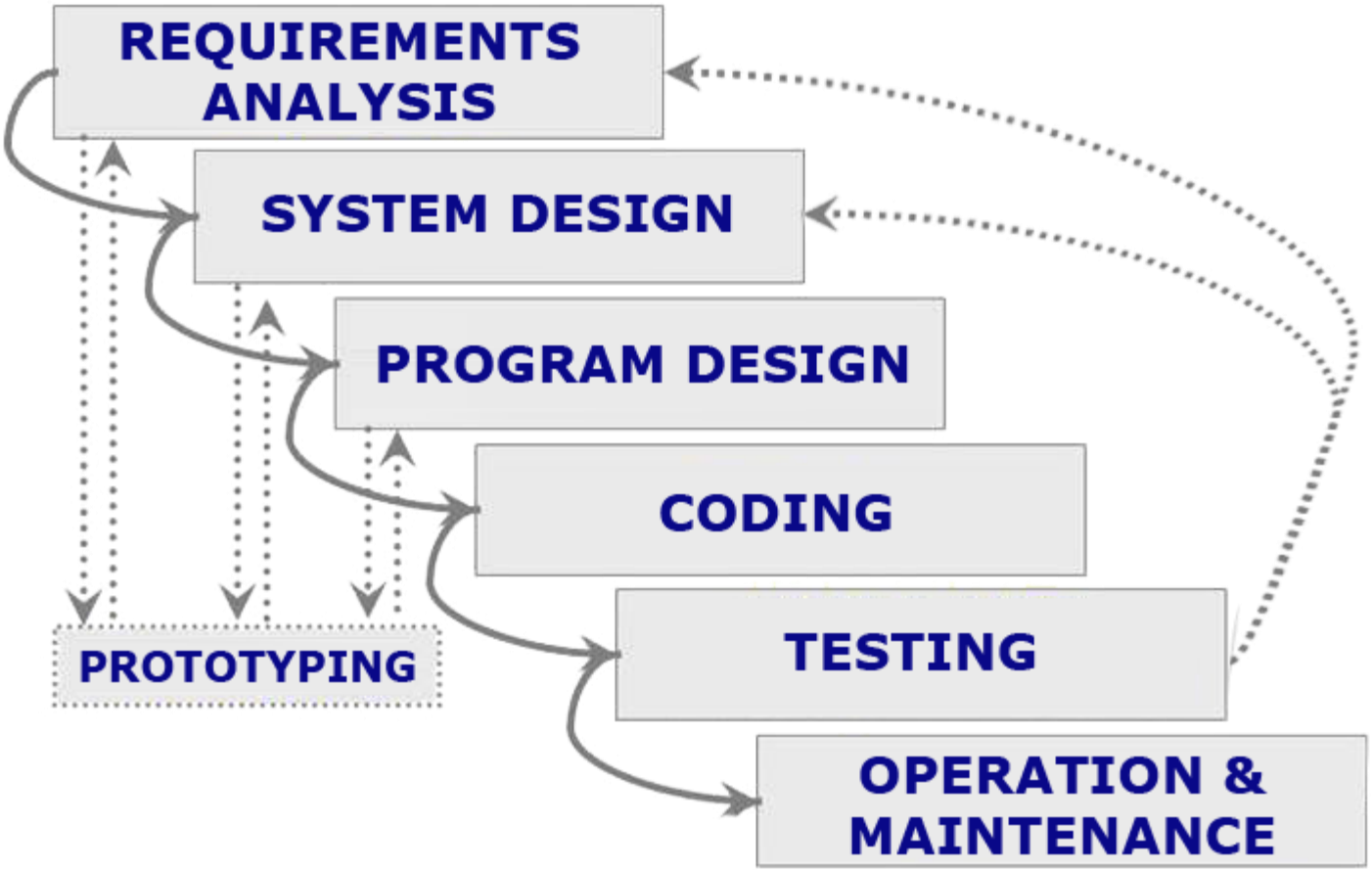
Waterfall model sa prototipom

- Prototip je delimično razvijen proizvod.
- Prototip pomaže dizajnerima da procene alternativne strategije dizajna (prototip dizajna)
- Prototip pomaže korisnicima da razumeju kako će izgledati sistem (prototip korisničkog interfejsa)
- Prototip je koristan za verifikaciju i validaciju

Waterfall model sa prototipom

- Omogućava klijentu da eksperimentiše sa radnom verzijom produkta.
- Razvojni proces se nastavlja samo ako je klijent zadovoljan funkcionisanjem prototipa.
- U toku određenih faza developer utvrđuje specifikaciju klijentovih stvarnih potreba.

Waterfall model sa prototipom



Waterfall model sa prototipom

- Prednosti:

➢ Dizajner softvera i programer mogu dobiti povratne informacije od korisnika u ranim fazama projekta.

➢ Klijent i izvođač radova mogu uporediti da li softver ispunjava specifikacije, prema kojima je softver napravljen.

➢ Omogućava softverskom inženjeru određeni uvid u tačnost inicijalnih projektnih predračuna i da li se mogu ispuniti predloženi vremenski rokovi.

Waterfall model sa prototipom

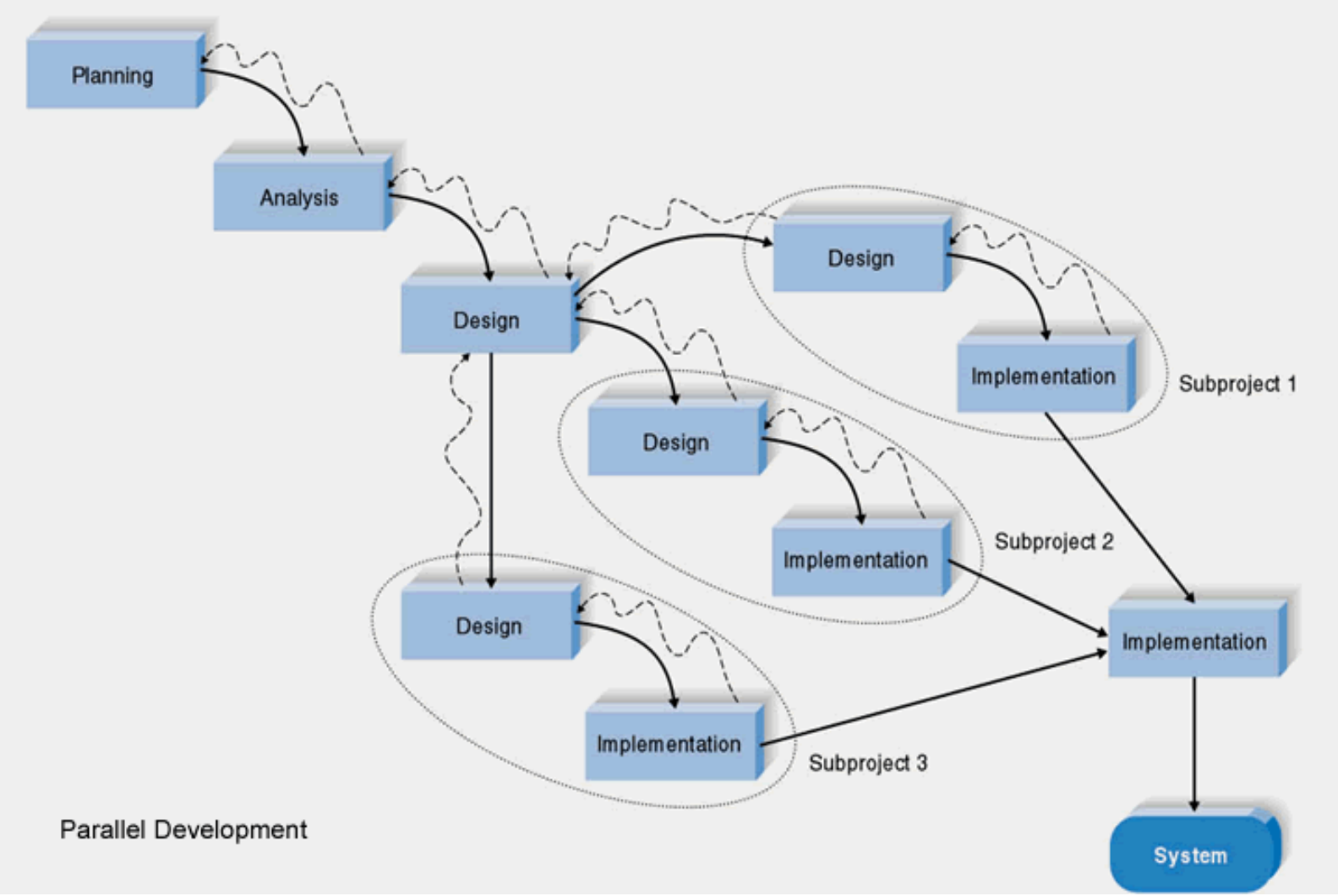
- Nedostaci:

- Klijenti često očekuju da će nekoliko minimalnih izmena na prototipu zadovoljiti njihove potrebe, ne shvatajući da se u žurbi izgradnje prototipa nije vodilo računa o celokupnom kvalitetu softvera .
- Programeri mogu izgubiti fokus o stvarnoj nameni prototipa i kompromitovati kvalitet proizvoda, npr. mogu primeniti neke neefikasne algoritme ili neodgovarajuće programske jezike korišćene u razvoju prototipa (uglavnom posledica lenjosti ili bliskosti sa jednostavnijim metodama)
- Prototip će teško biti prihvaćen na sudu u slučajevima kada se klijent ne slaže da je developer ispunio svoje obaveze.

Metodologija za paralelni razvoj

- Metodologija je slična Waterfall modelu
- Projekat je podeljen na manje projekte i svaka faza se izvršava posebno za svaki od njih.
- Razvojni proces se izvršava konkurentno i odvojeno za svaki od potprojekata.

Metodologija za paralelni razvoj



Metodologija za paralelni razvoj

- Prednosti:
  - Smanjuje vreme razvoja
  - Manje su šanse da su potrebne prepravke
- Nedostaci:
  - Podprojekte je ponekada teško integrisati

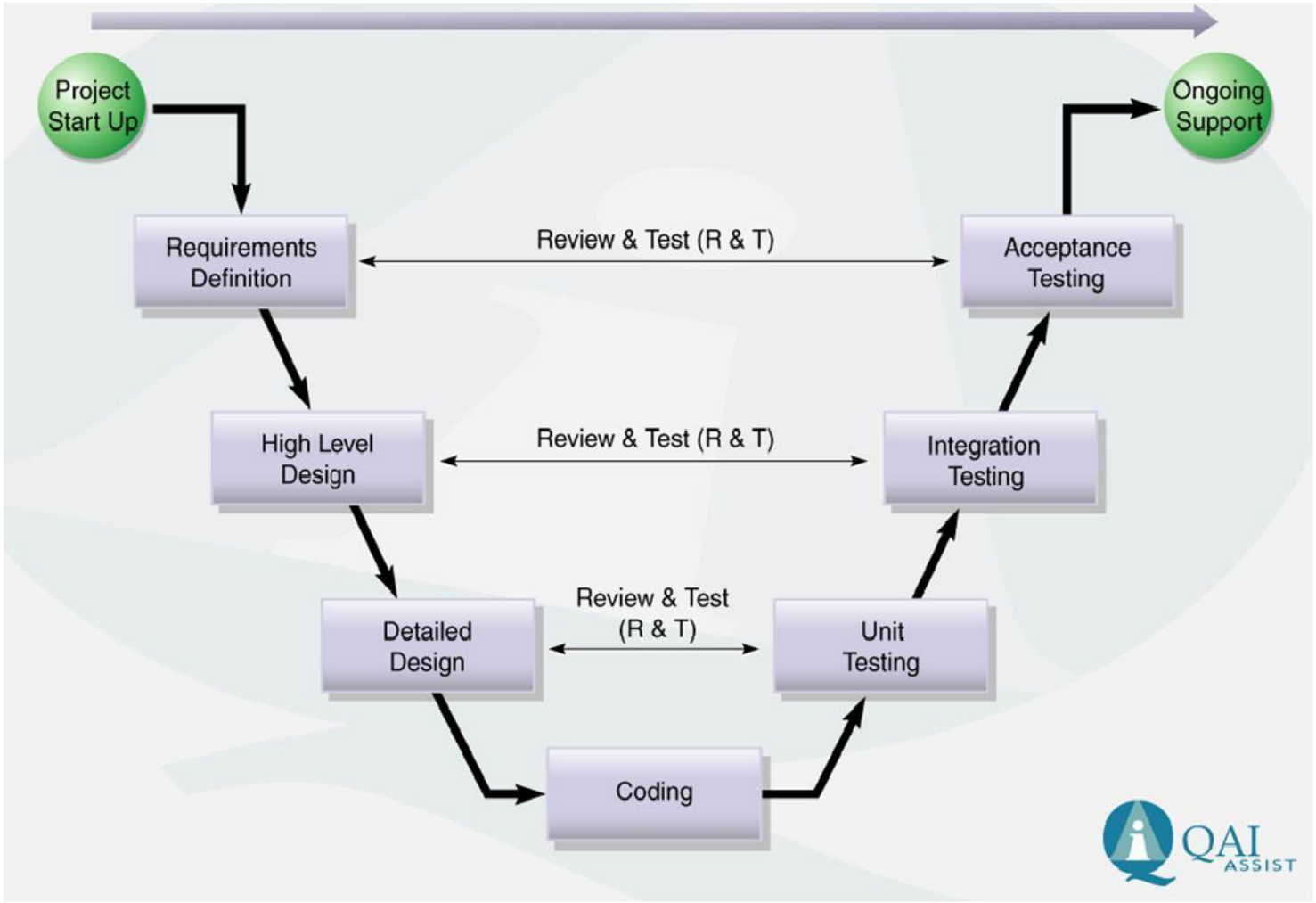
V model

- Mnogo je veći značaj aktivnosti testiranja nego kod Waterfall modela. Procedure i uslovi testiranja se razvijaju u ranim fazama životnog ciklusa pre implementacije.
- Životni ciklus počinje zahtevima kao i kod waterfall modela.
- Pre početka razvoja kreira se plan testiranja prihvatljivosti sistema koji se fokusira na zadovoljenju funkcionalnosti definisanih zahtevima.

V model

- U ranim fazama dizajna definiše se arhitektura sistema i dizajn. Plan testiranja integracija se takode definiše u ovoj fazi kako bi se utvrdilo da svi delovi funkcionišu zajedno.
- U kasnijim fazama dizajna se dizajniraju komponente softvera i testovi pojedinačnih komponenti.
- U fazi implementacije se realizuje kodiranje. Po završetku kodiranja, ranije planirani testovi se izvršavaju.

V model


Project Start Up
Requirements Definition
Review & Test (R & T)
Acceptance Testing
Ongoing Support
High Level Design
Review & Test (R & T)
Integration Testing
Detailed Design
Review & Test (R & T)
Unit Testing
Coding
QAI ASSIST

V model

- Prednosti

- Svaka faza ima specifične produkte.
- Veće su šanse za uspeh od Waterfall modela zbog razvoja planova testova u ranim fazama životnog ciklusa.
- U poređenju sa waterfall modelom vreme realizacije je kraće do 50%.
- Daje dobre rezultate kod projekata čije zahteve je lako razumeti.
- Iskorišćenje resursa je veliko.
- Aktivnosti testiranja koje počinju veoma rano, doprinose smanjenju vremena razvoja i troškova projekta.

V model

- Nedostaci

- Vrlo je nefleksibilan, kao i waterfall model.
- Jako teško i skupo je postići malu fleksibilnost.
- Softver se razvija tokom faze implementacije, pa nema ranih prototipova.
- Model ne obezbeđuje rešenja problema nađenih u fazi testiranja.

V model

- Kada koristiti ovaj model?

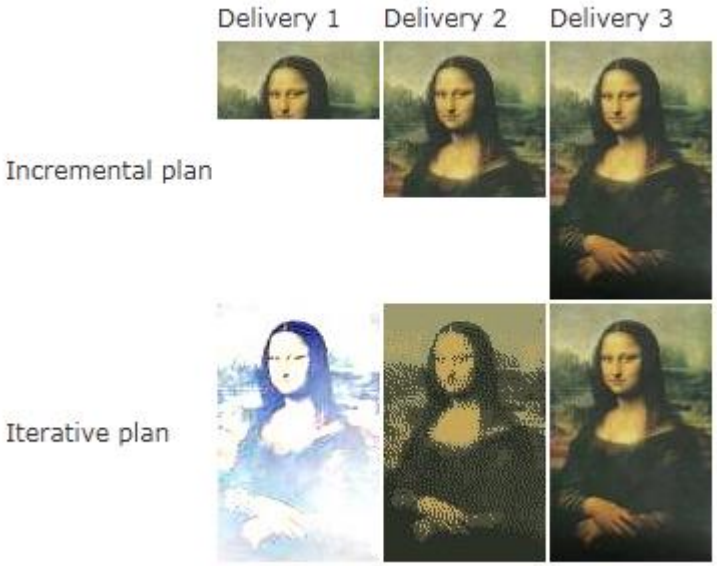
➢ Kada vreme i troškovi predstavljaju ograničenje projekta.

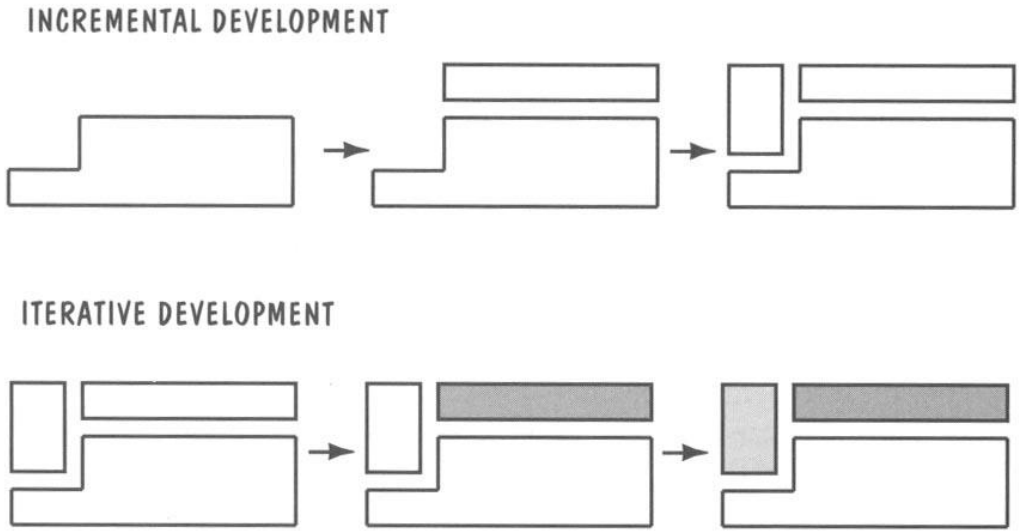
➢ Kada i Waterfall model.

Inkrementalni model

- Kombinuje elemente linearnog sekvencijalnog modela sa iterativnom filozofijom prototipova.
- Svaka linearna sekvenca isporučuje uvećan softver.
- Kod ovog modela, prvi inkrement je često osnovni proizvod, dok se u ostalim inkrementima dodaju dodatne funkcionalnosti.

Inkrementalni vs iterativni





Inkrementalni model

- Prednosti
  - Smanjena cena izmene korisničkih zahteva u toku razvoja.
  - Količina analiza i dokumentacije koju treba ponovno uraditi je mnogo manja u odnosu na model vodopada.
  - Olakšano dobijanje povratne informacije od korisnika u toku razvoja.
  - Korisnici imaju uvid i mogućnost komentarisanja trenutno implementiranih funkcionalnosti, a samim tim i u progres realizacije projekta.
  - Moguća brža isporuka korisnog softvera naručiocu.
  - Korisnici mogu da koriste softver pre nego u slučaju modela vodopada.

Inkrementalni model

- Nedostaci

➢ Proces razvoja nije vidljiv.

- Menadžerima su potrebne redovne isporuke kako bi merili napredovanje. Ako se sistem brzo razvija nije efikasno praviti dokumentaciju za svaku verziju sistema.

➢ Struktura sistema ima tendenciju degradacije sa dodavanjem novih inkremenata.

- Ukoliko se ne ulaže novac u refaktorisanje i unapređenje softvera, periodične izmene imaju za posledicu degradaciju strukure.

Spiralni model

- Sličan je inkrementalnom modelu, sa većim naglaskom na analizi rizika.
- Ima četiri faze: planiranje, analiza rizika, inženjerstvo i evaluacija.
- Projekat više puta prolazi ove faze u iteracijama.

Spiralni model

- Zahtevi se analiziraju u fazi planiranja.
- U fazi analize rizika identifikuju se rizici i alternativna rešenja. Prototip se kreira na kraju faze analize rizika.
- Softver se razvija u inženjerskoj fazi a na kraju faze se realizuje testiranje.
- Faza evaluacije omogućava korisniku da evaluira projekat pre nego što se pređe na narednu spiralu.

Analiza rizika

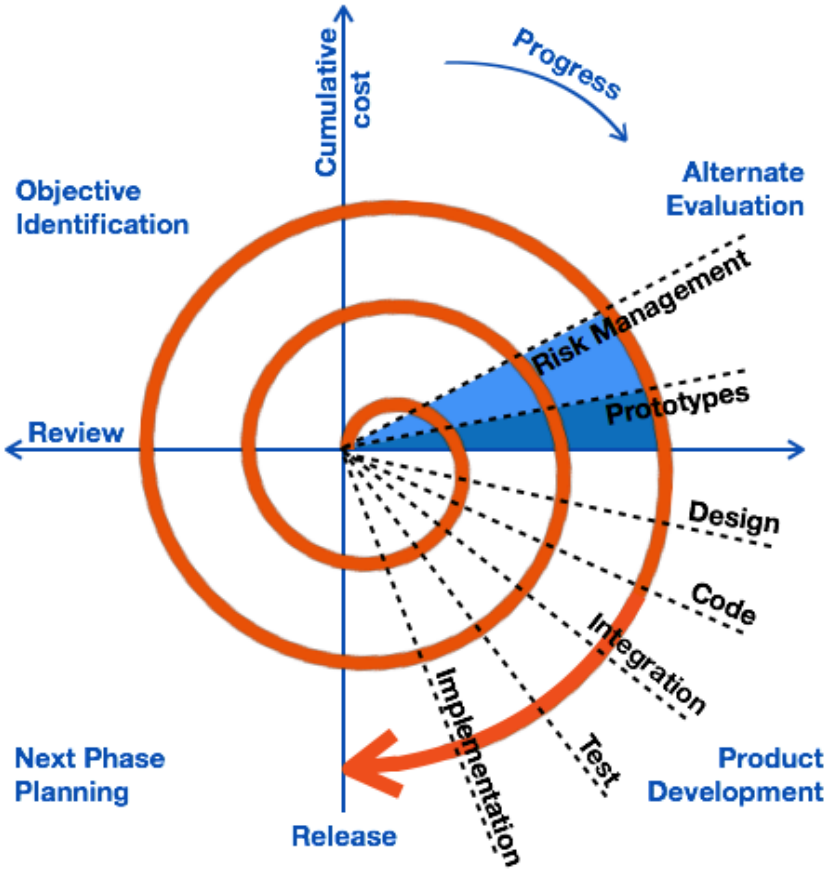
- Rizik posmatramo sa dva aspekta: verovatnoća javljanja problema i stepen ozbiljnosti poblema.

➢ Verovatnoća je mera izvesnosti da će se određeni problem dogoditi.

➢ Stepen ozbiljnosti je mera posledica koje određeni problem može da izazove.

Spiralni model

- U spiralnom modelu ugaona komponenta predstavlja progres, a radijus spirale predstavlja sveukupne troškove.



Spiralni model

- Prednosti:
  - Visok je stepen analize rizika.
  - Dobar je za velike i kritične projekte.
  - Softver se proizvodi u ranim fazama životnog ciklusa.
- Nedostaci:
  - Može biti veoma skup model.
  - Analiza rizika zahteva visok stepen ekspertize.
  - Uspeh projekta je veoma zavistan od faze analize rizika.
  - Nije dobar za male projekte.

Rapid application development (RAD)

- Inkrementalni proces razvoja softvera koji naglašava veoma kratak razvojni ciklus.
- Predstavlja “high-speed” adaptaciju linearnog sekvencijalnog modela u kome je brzi razvoj postignut korišćenjem **konstrukcije zasnovane na komponentama.**
- Sistem se razvija paralelno, od strane više timova.
- Ukoliko su zahtevi jasni omogućava razvoj funkcionalnog sistema u kratkom periodu (npr. 60-90 dana)

Rapid application development (RAD)

- Naglašava sledeće faze:

➢ Poslovno modelovanje – tokovi informacija između poslovnih funkcija.

➢ Modelovanje podataka – skupovi podataka potrebnih za podršku poslovanju.

➢ Modelovanje procesa – skupovi podataka se transformišu da bi ostvarili tokove informacija neophodne za implementaciju poslovnih funkcija.

➢ Generisanje aplikacije – upotreba postojećih programskih komponenti (kada je moguće)

➢ Testiranje – s obzirom da se naglašava upotreba postojećih komponenti, mnoge od njih su već testirane.

Rapid application development (RAD)

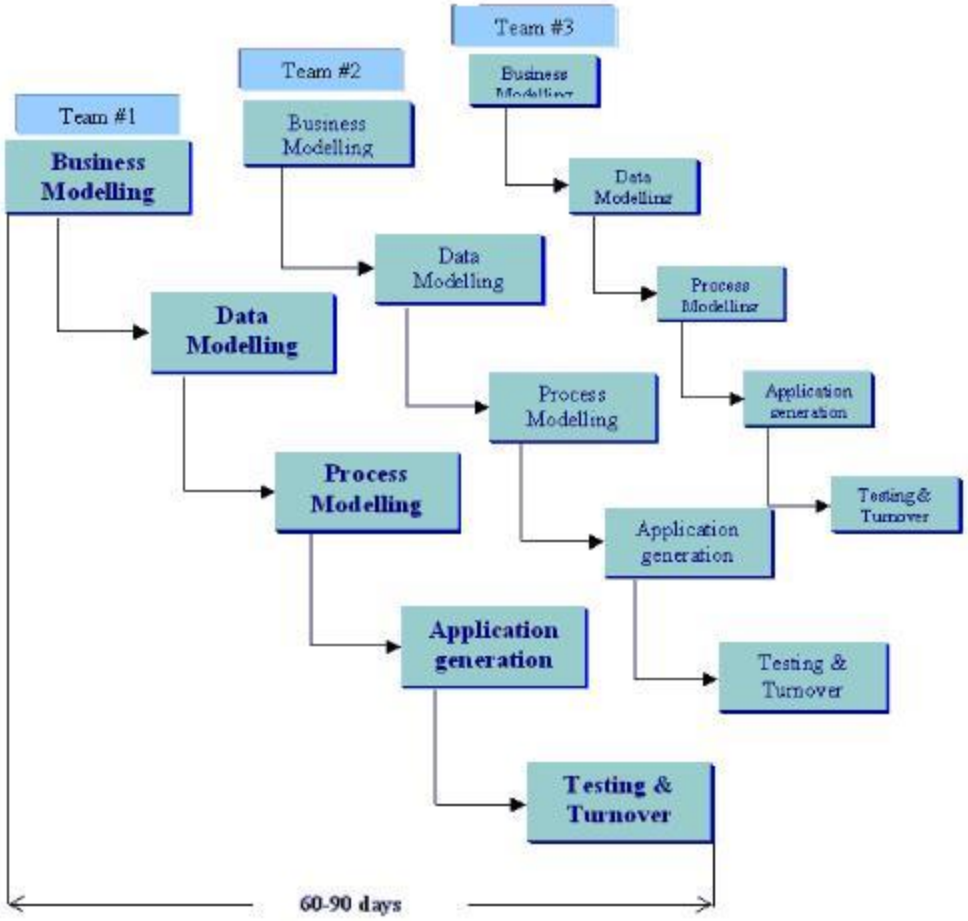


Figure 1.5 – RAD Model

Rapid application development (RAD)

- Nedostaci:

➢ Za velike ali skalabilne projekte, RAD zahteva dovoljno ljudskih resursa koji bi kreirali odgovarajući broj RAD timova.

➢ Nisu sve aplikacije pogodne za RAD. Ako sistem ne može biti modularizovan na odgovarajući način, kreiranje komponenti potrebnih za RAD može biti problematično.