# Skripta Razvoj Softvera 1 Prva Parcijala

## Uvod

Program je set instrukcija

Aplikacija je jedan ili više programa

U softver spadaju sve komponente koje sačinjavaju ili opisuju softver:

* Programske instrukcije
* Strukture podataka
* Dokumentacija
* ...

Inžinjering podrazumijeva primjenu naučnih principa i metoda u razvoju upotrebljivih struktura i komponenti a zastupljen je u gotovo svim područjima ljudskog djelovanja

Inžinjering je primjenjena nauka, a softverski inzinjering je primjenjena racunarska nauka

Racunarska nauka osigurava teorijsku osnovu za dizajn i koristenje savremenih racunarskih komponenti

Softverski inžinjering omogućava efikasnije rješavanje postavljenih problema.

Definira :

* Razumijevanje problema ( potrebe narucioca projekta)
* Definisanje plana za rjesavanje problema
* Izvrsenje plana
* Evaluacija rezultata

Kompleksnost problema zahtijeva analizu ili dekompoziciju (kompleksne probleme razlazemo na manje cjeline – divide and conquer)

Nakon analize (dekompozicije) slijedi sinteza – kreiranje komplektnog softverskog rješenja od manjih cjelina

Prilikom rješavanja problema koriste se različite metode, alati, procedure i paradigme.

**Metoda** je formalni postupak za postizanje nekog rezultata

**Alat** je instrument (automatizovani sistem) koji omogućava da se određena aktivnost obavi na kvalitetniji način

**Procedura** - kombinacija alata i metoda koji u međusobnom skladu proizvode dati proizvod

**Paradigma** je pristup ili filozofija razvoja softver-a (npr. Objektno orjentisano programiranje)

Korištenjem alata tehnika procedura i paradigmi poboljšava se kvalitet softverskog proizvoda.

Procesni modeli :

* Preskriptivni ( vodopad, inkrementalni, spiralni..)
* Agilni ( XP, Scrum, KanBan, Crystal)

**Agilne metode** – zahtjeva se interakcija i kolaboracija sa korisnikom, softver je važniji od detaljne dokumentacije, a prilagođavanje promjenama važnije od krute administracije

Da bi se osigurao kvalitet softvera potrebno je osigurati **efikasno testiranje.**

Osim što softver radi treba imati i sljedeće karakteristike:

* Mora ispuniti potrebe naručioca
* Softver treba biti stabilan
* Softver treba biti razumljiv
* Treba ostaviti mjesta za nadogradje i olakšati održavanje

**Softverski inžinjering je projektovanje i razvoj visoko kvalitetnog softvera.**

**Kvalitet softvera posmatra se kroz :**

* Kvalitet proizvoda(ispunjeni zahtjevi, stabilnost, arhitektura i organizacija koda, mogućnost proširenja..)
* Kvalitet postupka izrade(procesa)
* Kvalitet proizvoda u kontekstu poslovnog okruženja za koji je namijenjen (povrat investicije)

**Aktivnosti u razvoju softvera** jesu analiza i definisanje zahtjeva, projektovanje sistema, pisanje programa, unit testing, integration testing, testiranje sistema, isporuka sistema, odrzavanje.

**U fazama razvoja sudjeluju :**

1. ANALIZA I DEFINISANJE ZAHTJEVA : Analitičar
2. PROJEKTOVANJE SISTEMA : Analitičar i projektant
3. PROJEKTOVANJE PROGRAMA : Projektant i programer
4. IMPLEMENTACIJA PROGRAMA : Programer
5. UNIT TESTING : Programer i tester
6. INTEGRATION TESTING : Tester
7. TESTIRANJE SISTEMA : Tester
8. ISPORUKA SISTEMA : Instruktor
9. ODRŽAVANJE : Analitičar, projektant, programer, tester, instruktor, cuko, maca, pujdo

**Sistem je** skup entiteta, aktivnosti, njihovih odnosa i granica sistema.

Softverski inžinjering kao disciplinu označava:

* Apstrakcija
* Analiza i dizajn
* Prototip korisničkog interfejsa
* Arhitektura softvera(dekompozicija)
* Softverski procesi
* Ponovna iskoristivost
* Mjerenja(uspjeh i kvalitet)
* Alati i integrisana okruženja

## Modeliranje procesa i životnog ciklusa softvera

**Proces** predstavlja niz definisanih koraka koji obuhvataju aktivnosti, ograničenja i resurse u cilju ostvarenja određenog rezultata.

**Svaki proces posjeduje** glavne aktivnosti, resurse, kolekciju povezanih podprocesa i aktivnosti kao sekvence sa jasnim relacijama

**Softverski razvojni proces** se naziva životni ciklus softvera.

Modeli procesa izrade softvera :

* Model vodopada
* V model
* Prototipski model
* Fazni razvoj
* Spiralni model
* RUP

### Model vodopada

Veoma visok nivo apstrakcije razvojnog procesa.

Kraj jedne faze omogućava prelazak na iduću

Nakon što se zahtjevi zabilježe, analiziraju i dokumentiraju u potpunosti tim može nastaviti sa dizajnom sistema.

Svaka aktivnost u sklopu procesa posjeduje **međuproizvod** kao mjeru realizacije projekta.

Nedostaci

* Preobimna dokumentacija
* Nefleksibilnost
* Nemogućnost korisnika da definiše kompletne zahtjeve
* Dug period izrade softvera
* Veći troškovi
* Greške otkrivene u kasnijim fazama mogu biti fatalne

### Model vodopad::Poboljšanja ( Model vodopad s prototipom)

Projektni tim može da izgradi sistem koji implementira jedan dio ključnih zahtjeva. Nad tim izgrađenim prototipom provjerava se dosljednost, izvodljivost i praktične vrijednosti.

Ako se utvrdi ostupanje zahtjevi se revidiraju i time izbjegavaju mnogo skuplje revizije u fazi testiranja

Kad se softver testira vrsi se validacija i verifikacija.

**Validacija** osigurava potpunost implementacije zahtjeva u sklopu sistema. Svaka funkcija sistema može se popratiti i povezati sa određenim zahtjevom definisanim u specifikaciji zahtjeva.

**Verifikacija** osigurava ispravan rad svake od funkcionalnosti. Ona provjerava kvalitet implementacije.

Izrada prototipa korisna je verifikaciju i validaciju ali te aktivnosti mogu da se vrše i u drugim fazama razvoja softvera.

Ako verifikacija ukaže na grešku vraćamo se na fazu projektovanja sistema.  
Ako validacija ukaže na grešku vraćamo se na fazu analize zahtjeva.

### V Model

Nastao je kao modifikacija modela vodopada

Demonstrira odnos testiranja s fazama analize i dizajna.

Naziv je dobio prema V izgledu.

U ishodištu se nalazi proces programiranja.

Na lijevoj strani nalazi se sve vezano uz analizu i dizajn a na desnoj testiranje i odrzavanje.

Uz analizu se veže :

* Analiza zahtjeva
* Projektovanje sistema
* Projektovanje programa

Uz testiranje se veže

* Završno testiranje
* Testiranje sistema
* Testiranje dijelova i testiranje pri integraciji

**Tokom testiranja dijelova i testiranje** prilikom integracije možemo verificirati dizajn programa.

Tokom testiranja dijelova i testiranja pri integrisanju programeri i clanovi tima za testiranje treba da osiguraju da svi aspekti dizajna programa budu ispravno implementirani u kodu.

**Testiranje sistema** verificirat će dizajn sistema.

Programeri i testeri pobrinu se da su svi aspekti analize sistema ispravno implementirani u kodu

**Završno testiranje validirat** **će analizu zahtjeva.**

V model se usmjerava na aktivnosti i ispravan rad sistema u odnosu na dokumentaciju.

### Prototipski model

Model vodopada može da se dopuni dodavanjem aktivnosti izrade prototipa radi boljeg razumijevanja zahtjeva.

Izrada prototipa se pokazala dobrom i može da bude osnov za efikasno modeliranje procesa.

Tako je nastao protipski model.

Omogućava da se cijeli sistem ili dijelovi sistema brzo napravi radi razjašnjenja ili boljeg razumijevanja zahtjeva.

Zahtjevi ili dizajn iniciraju ponovne analize kako bi se osiguralo da projektni tim, korisnik i naručilac postignu zajednički stepen razumijevanja stvarnih potreba i predložene funkcionalnosti.

**Prednost jest smanjenje rizika i neodređenosti prilikom projektovanja**

U početku se sastavi početna lista zahtjeva od starne naručilaca i korisnika. Zatim se istražuju alternative tako što zainteresovane strane analiziraju mogiće ekrane, tabele, izvještaje i druge izlaze iz sistema koji direktno koriste korisnici i kupci.

Kako korisnik i naručilac definišu svoje želje, dolazi do revizije zahtjeva.

Kad se završi sa svim revizijama zahtjeva ide se dalje na dizajn. Istražuju se sve varijante u dizajnu uz konsultacije sa korisnikom i naručiocem.

Moguće se vratiti na listu zahtjeva pa ih ponovno revidirati. Na kraju kad se odredi lista zahtjeva i formira dizajn ide se na kodiranje. Moguće je vratiti se iz faze kodiranja u fazu dizajna ili zahjeva. Nakon toga slijedi testiranje i isporuka softvera.

#### Fazni razvoj

Kod faznog razvoja govori se o dva pristupa :

* Inkrementalni (podsistemi podijeljeni prema funkcionalnostima)
* Iterativni

Kod inkrementalnog pristupa sistem se podijeli na podsisteme prema funkcionalnostima. Sa svakim novim release-om sistema sistem se nadogradjuje sve dok ne dostigne sve funkcionalnosti.

Iterativni razvoj omogućava da se izbaci potpun sistem na početku a zatim vrše modifikacije pod sistema.

Iterativni pristup obicno se kombinuje s inkrementalnim.

Svaki inkrement sastoji se od vise iteracija.

### Spiralni model

Sličan je iterativnom model.

Ideja je da se u aktivnosti razvoja ubaci i upravljanje rizicima radi smanjenja i kontrole rizika.

Sastoji se od četiri iteracije.

Prva iteracija rezultirat ce dokumentom „Princip rada“ – dokument koji kontroliše kompletnost zahtjeva i dosljednost

Druga iteracija rezultirat će zahtjevima

Treća iteracija rezultirati će dizajnom

Četvrta iteracija omogućit će testiranje.

### RUP – Rational Unified Process

Procesni okvir kojeg karakteriše velik stepen fleksibilnosti, odnosno prilagođavanja različitim potrebama realizacije projekta.

Podržava iterativni razvoj

Insistira na vizualnom modeliranju (UML) i konstantoj provjeri kvaliteta

Sastoji se od jasno definiranih faza:

* Početak(Inception)
* Elaboracija
* Konstrukcija
* Tranzicija

Početak – definisanje opsega projekta i zahtjeva, planiranje aktivnosti i razmatranje potencijalnih arhitektura.

Elaboracija – defniiranje i validacija arhitekture, potpuno razumijevanje najkritičnijih dijelova i implementacijskih faza

Konstrukcija - implementacija i testiranje, upravljanje resursima

Tranzicija – planiranje isporuke, isporuka i beta testiranje

## Planiranje i upravlajnje projektom

Projektni rokovi predstavljaju razvojni ciklus nekog softvera u okviru određenog projekta sa fazama svake pojedinačne aktivnosti za koje se procjenjuju potrebni resursi i vrijeme trajanja.

Projekat započinje razgovorom sa naručiocem i potencionalnim korisnicima.

Definiraju se međuproizvodi ( stavke koje naručilac očekuje tokom razvoja projekta)

Razvoj projekta može se prodijeliti na niz faza.

**Svaka faza sastoji se od koraka a svaki korak od niza aktivnosti.**

**Akitvnost** predstavlja dio projekta koji se odigrava tokom nekog vremenskog perioda.

**Work Breakdown Structure (WBS)** - predstavlja proces generisanja strukture poslova u projektu.

Aktivnosti i prekretnice predstavljaju elemente koje naručilac i razvojni tim koriste za praćenej projekta.

Svaka aktivnost opisuje se s 4 parametra :

* Prethodnik – uslovi koji omogućavaju praćenje aktivnosti
* Trajanje – vrijeme potrebno za kompletiranje aktivnosti
* Krajnji rok – datum do kojeg aktivnost mora biti okončana
* Krajnja tačka – obično prekretnica, označava da je aktivnost završena

**Detaljnom razradom aktivnosti** može se dobiti potpuni pregled resursa neophodnih za realizaciju projekta( resursi jesu učesnici projekta i korišteni alati).

Radi opisivanja zavisnosti najčešće se crta graf aktivnosti.

**Čvorofi grafa** predstavljaju **prekretnice projekta**.

**Grane** koje povezuju čvorove predstavljaju sadržane **aktivnosti**.

Analizom putanja među prekretnicama(čvorovima) metodom kritične putanje mogu se odrediti aktivnosti koje su najkritičnije za završetak projekta u predviđenom vremenu.

**Stvarno potrebno vrijeme** – vrijeme potrebno za okončanje aktivnosti

**Raspoloživo** **vrijeme** – vrijeme koje je u okviru vremenskih rokova raspoloživo za okončanje aktivnosti

**Latentno vrijeme** razlika između vremena najkasnijeg i najranijeg započinjanja aktivnosti.

Jako bitan dio projekta jesu sastanci.

Rukovodilac projekta određuje tko treba prisustovati sastanku.

Sastanak treba da ima dnevni red.

Predsjedavajući sastanka treba usmjeravati diskusiju.

**INTUITIVAN** – Priznaje osjecaje

**RACIONALAN** – Odlučuje logično

**INTROVERT** – Pita drugoga

**EXTROVERT** – Saopštava drugima

Struktura organizacije članova tima :

**Glavni programer** – potpuno odgovoran za projektovanje i razvoj sistema, svi drugi članovi tima podnose izvještaj glavnom programeru, glavni programer ima posljednju riječ prilikom donošenja svih odluka

Svaki član tima je potpuno ravnopravan

Struktura tima počiva na demokratskim principima gdje se članovi tima izjašnjavaju o svim odlukama.

Ključna akitvnost u realizaciji projekta je procjena troškova.

Loše procjene najčešće su uzrokovane :

* Čestim izmjenama zahtjeva
* Nerazumijevanjem zahteva od strane korisnika
* Lošom koordinacijom tokom projektovanja, administracije i drugih aktivnosti.
* Nedostatkom adekvatnih metoda za procjenjivanje

Aspekti projekta koji imaju ključnu ulogu za dobru procjenu troškova:

* Složenost predloženog rješenja
* Integraciju sa postojećim rješenjima
* Složenost pojedinačnog programa u sistemu
* Veličina sistema izražena brojem funkcionalnosti ili programa
* Broj članova projektnog tima
* Sposobnost i iskustvo članova projektnog tima
* Očekivana učestalost ili obim mogućih izmjena u zahtjevima

Metode za procjenu potrebnog rada zasnivaju se na stručnoj procjeni.

Neformalne metode se zasnivaju na iskustvu rukovodioca sa sličnim projektnima.

Delfi metod koristi stručnu procjenu na drugačiji način.

Stručnjaci u tajnosti donose pojedinačna predviđanja koristeći postupak po vlastitom izboru. Zatim se računa srednja procjena i ona se predstavlja grupi. Svaki strucnjak ima mogucnost da ako zeli revidira svoju procjenu. Proces se ponavlja sve dok svi strucnjaci ne odustanu od revizija.

Većina modela potvrđuje da je veličina projekta najutjecajniji faktor u jednačinama u kojim se rad izračunava kao

E = (a+bSc )m(X)

S – procjena velicina sistema ( broj linija code-a)

A,b,c – konstante

X – vektor faktor troskova

M – korekcioni faktor

### Upravljanje rizicima

Rizik se razlikuje od drugih događaja na osnovu tri pokazatelja

* Utiicaj rizika – gubitak vremena, kvaliteta, novca, kontrole..
* Vjerovatnoća rizika – vjerovatnoca pojave tog događaja
* Kontrola rizika – skup radnji koje se poduzimaju u cilju minimiziranja ili potpunog eliminisanja rizika

Efekti identifikovanih rizika se mogu kvantifikovati množenjem uticaja rizika s njegovom vjerovatnoćom čime se dobija **izloženost riziku**.

Postoje dva glavna izvora rizika :

* Opšti rizici – zajednički za sve softverske projekte
* Rizici specificni za projekat – prijetne koje su posljedica slabih tačaka konkretnog projekta

**Upravljanje rizicima obuhvata više koraka:**

* Procjena rizika
* Kontrola rizika

**U procjenu rizika** spada identifikovanje rizika, analiza rizika i dodjeljivanje prioriteta svakom riziku.

**U kontrolu rizika** spada umanjenje rizika, planiranje rizika, razješavanje rizika.

**Rizik se može umanjiti** tako što se proba izjeći(izmjenom zahtjeva), prenošenjem rizika( dodjeljivanje rizika drugim sistemima) ili podrazumijevanjem rizika (prihvatanje i kontrola u okviru resursa projekta).

Predstavljanje troškova projekta, rokova, organizacije, te rezultata analize rizika i upravljanje njima vrši se unutar plana projekta.

# DIZAJN SISTEMA

Dizajniranjem sistema definišemo rješenje za postavljene zahtjeve, a predstavlja se kroz konceptualni i tehnički dizajn.

Konceptualni dizajn klijentu opisuje šta bi sistem trebao raditi. Opisuje sve aspekte sistema(odakle ce podaci stizati, sta ce se s podacima dogadjati, kako će sistem izgledati korisnicima...).

Konceptualni dizajn trebao bi :

biti pisan jezikom klijenta, ne sadrzavati previse tehnickih termina, opisivati funkcije sistema, ne zavisiti od implementacije, biti povezan s dokumentima speicifikacije.

Tehnički dizaj omogućava onima koji će razvijati sistem da utvrde koji će hardver i sofftver biti potreban za rješenje postavljenog problema.

Tehnički dizajn sadrži

* Opis glavnih hardverskih komponenti i njihovih funkcija
* Hijerarhiju i funkcije softverskih komponenti
* Strukture i tokove podataka

Osnovni dizajn definiše sljedeće stavke

* Platforma
* Sigurnosni koncepti
* Hardver
* Korisnicki interfejs
* Interfejs izmedju komponenti softvera
* Arhitektura
* Baza podataka
* Izvjestaju
* UML
* Obuka

Definisanje arhitekture ne predstavlja zasebnu fazu u procesu razvoja softvera.

Njom se definise odno pojedinih kompenenti ili dijelova softver.

Arhitektura može biti:

* Monolitska
* Klijent server
* Komponent bazirana – softver podijeljen na komponente a svaka komponenta je u nekoj vrsi interakcije s drugom komponentom
* Servisno orijentisana arhitektura (SOA) – Dijelovi softver aimplementirani kao servis, tj dijelovi koji pruzaju servis ili uslugu klijentskim aplikacijama.
* Arhitektura bazirana na pravilima – izvrsenje zavisi od seta pravila
* Distrubirana arhitektura – dijelovi softvera se izvrsavaju na razlicitim lokacijama
* Kombinovana

Dizajn niskog nivoa (detaljni dizajn) ima zadatak da definise sve nepoznanice izmedju osnovnog dizajna i momenta otpočinjanja programiranja ili pisanja programskog koda.

Odgovaramo na pitanje kako a ne na sta.

Govori se o mnogo konkretnijim implementacijskim odlukama.

# EVIDENTIRANJE ZAHTJEVA

Cilj faze definisanja zahtjeva je razumijevanje problema.

Analitičar radi s naručiocem na izvođenju zahtjeva.

Krajni rezultat razgovora je specifikacija softverskih zahtjeva koja se koristi za komunikaciju sa drugim clanovima razvojnog tima.

Tehnike definisanja zahtjeva :

* Analiza službene dokumentacije
* Intervju
* Anketiranje
* Analiza scenarija
* Posmatranje
* Izrada pretpostavki
* Pravljenje i koristenje video zapisa
* FAST tehnika
* Izrada prototipa

Analiza službene dokumentacije je obavezna tehnika.

FAST – timski rad u definisanju zahtjeva.

Tipovi zahtjeva:

* Funkcionalni zahtjevi
* Nefunkcionalni zahtjevi
* Projektno ograničenje
* Procesno ograničenje

**Funkcionalni zahtjevi** - opisuju obavezno ponašanje softvera u kontekstu neophondih aktivnosti, a neke od njih su reakcija na ulaze, stanje svakog entitteta nakon okončanja aktivnosti i sl.

**Nefunkcionalni zahtjevi** – opisuju neke karakteristike kvaliteta koje softversko rješenje mora da posjeduje a neki od njih su kratko vrijeem odziva, lakoca koristenja, visoka pouzdanost..

**Projektno ograničenje** – predstavlja unaprijed donijetu odluku koja ograničava skup mogućih rješenja razmtranog problema, na primjer : izbor platforme, interfejs komponenti i sl

**Procesno ograničenje** – predstavlja ograničenje koje se odnosi na tehnike ili resurse koji se mogu koristiti u izgradnji sistema.

Postoje tri prioriteta zahtjeva :

* Zahtjevi koji se apsolutno moraju zadovoljiti(sustinski)
* Zahtjevi koji su veoma pozeljni (pozeljni)
* Zahtjevi koji se mogu ispuniti ali se mogu i izostaviti(opcionalnio)

Najčešće se kreiraju dva dokumenta :

* Definicija zahtjeva – namijenjena poslovnom auditoriju
* Specifikacija zahtjeva - namijenjena tehničkom auditoriju

**Definicija zahtjeva** predstavlja kompletan spisak svega što naručilac želi a sadrži sve neophodne zahtjeve, opis svih entiteta, ograničenja entiteta, ograničenja u vezi s nadgledanjem entiteta, ograničenja u vezi s transformacijom entiteta.

Definicija zahtjeva predstavlja ugovor koji sadrži opis funkcionalnosti koju razvojni tim prihvata da isporuči naručiocu.

Specifikacija zahtjeva iskazuje zahtjeve u vidu specifikacije ponašanja predloženog sistema.