

Procesi životnog ciklusa programskog proizvoda

Uvodno predavanje

Uvodno ponavljanje osnovnih pojmova iz područja programskog inženjerstva

Programski proizvod

Skup računalnih programa i odgovarajuće dokumentacije koji se isporučuju korisniku

- ◆ Proizvodi namijenjeni nepoznatim kupcima
 - samostalni proizvodi, prodaju se na otvorenom tržištu
- ◆ Proizvodi izrađeni po narudžbi i za jednog kupca
 - razvoj se posebno ugovara

◆ Osnovna svojstva

- Nevidljivost, složenost, promjenljivost, interoperabilnost (sklopovlje, korisnici, domena, drugi dijelovi sustava)
- Osnovna svojstva se ne mijenjaju

◆ Ostala svojstva

- Programski jezik, brzina sklopovlja, memorija
- Arhitektura programa
 - Funkcionalna
 - Objektno orijentirana
- Mogu se mijenjati

- ◆ Korektnost (zadovoljavanje zadanih specifikacija)
- ◆ Pouzdanost (!)
- ◆ Prilagođenost korisniku (usability)
- ◆ Točnost (ulazi, izlazi, performanse)
- ◆ Robustnost (reduciranje utjecaja radnih grešaka, krivih ulaznih podataka i sklopovskih neispravnosti)
- ◆ Pogodnost za održavanje (maintainability)
- ◆ Pogodnost za testiranje (testability)
- ◆ Čitljivost (readability)
- ◆ Efikasnost (zadovoljenje zahtjeva s najboljim iskorištenjem resursa)
- ◆ Proširivost
- ◆ Portabilnost (mogućnost prilagodbe u drugu radnu okolinu)

Definicija prema standardu TL9000

- ◆ procesi, aktivnosti i zadaci uključeni u koncept, definiciju, razvoj, funkcioniranje i održavanje programskog proizvoda
 - sastoji se od faza (npr. definicija i analiza zahtjeva, dizajn sustava, ..., itd.)
- ◆ Često se još koriste i sinonimi
 - generički proces razvoja programskog proizvoda
 - proces razvoja programskog proizvoda
 - razvoj programskog proizvoda
 - engleski nazivi: *software lifecycle*, *(generic) software development (process)*, *software process*

Životni ciklus programskog proizvoda



Definicija i analiza zahtjeva

Dizajn sustava

Dizajn programa

Implementacija programa

Testiranje programa

Integracijsko testiranje

Testiranje sustava

Isporuka

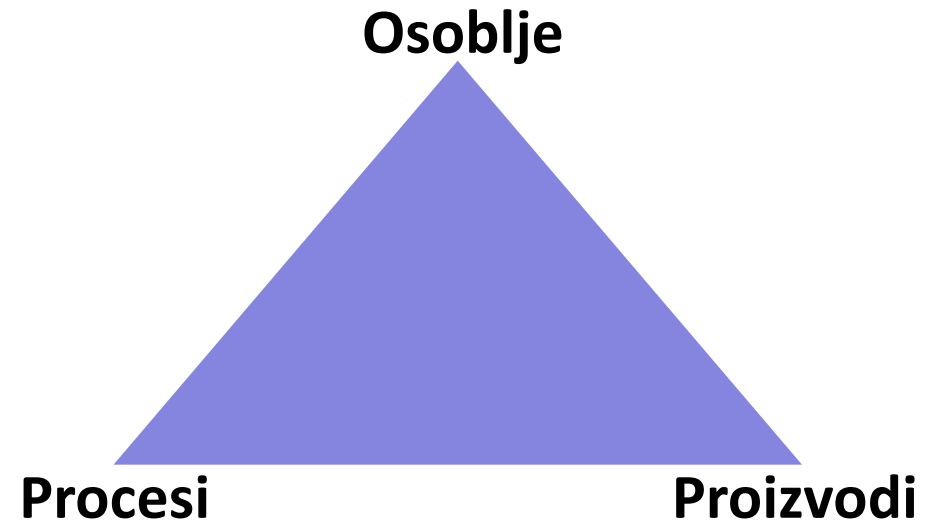
Održavanje

Povlačenje iz uporabe

Ciljevi razvoja programskog proizvoda



- ◆ Zadovoljenje korisnikovih potreba
- ◆ Niža cijena razvoja
- ◆ Visoke performanse
- ◆ Portabilnost
- ◆ Niži troškovi održavanja
- ◆ Visoka pouzdanost
- ◆ Isporuka na vrijeme



- ◆ Sustavni razvoj
- ◆ Računalna podrška razvoju programskih proizvoda (CASE)
- ◆ Detaljna analiza korisničkih zahtjeva i formalna specifikacija
- ◆ Demonstracija ranih verzija sustava (prototip)
- ◆ Više napora uložiti u razvoj koda bez neispravnosti (statičke provjere koda)



◆ Proces

- pretpostavka “dobar proces -> dobar proizvod”
- dobar proces: razumljiv, ponovljiv, predvidiv, mjerljiv

◆ Tehnologija

- metode i pomagala

◆ Osoblje

- programeri, timovi



- ◆ Skup aktivnosti koje vode željenom cilju
 - poslovni proces, programski proces

- ◆ Različite definicije procesa
- ◆ ovise o definiciji cilja
 - način na koji funkcionira kompanija (marketing, ljudski resursi)
 - način na koji se razvija (kodira, testira, ...) softver
 - upravljanje, inženjerstvo
 - uvođenje dodatnih formalizama i napora (fokus na proizvodu)



Procesi općenito (2)



- ◆ Svrha procesa
 - upute za obavljanje posla – podjela i koordinacija
 - osiguranje efikasne komunikacije

- ◆ Koordinacija i komunikacija
 - glavni problemi velikih projekata

Elementi procesa

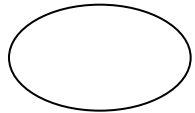


- ◆ Skup aktivnosti
- ◆ Redoslijed aktivnosti
- ◆ Međuovisnosti između aktivnosti

- ◆ Izvršitelji aktivnosti
- ◆ Ulazi aktivnosti
- ◆ Rezultati aktivnosti
- ◆ Trajanje aktivnosti
- ◆ Pomagala

Elementi procesa (2)

Komponente



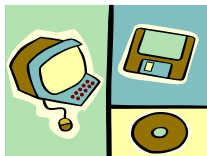
aktivnosti



rezultati aktivnosti
(kod, dokument)



izvršitelji



pomagala

Interakcije

tok informacija

tok artefakata

nadzor

komunikacija

vremenski raspored

ovisnosti

konkurentnost

- ◆ Identične značajkama ostalih procesa
 - Razumljivost
 - Vidljivost
 - Prihvatljivost
 - Pouzdanost
 - Robusnost
 - Brzina
 - Evolucija
 - Kvaliteta

Razine definicije procesa

Poslovna strategija



Prioriteti



Projekti

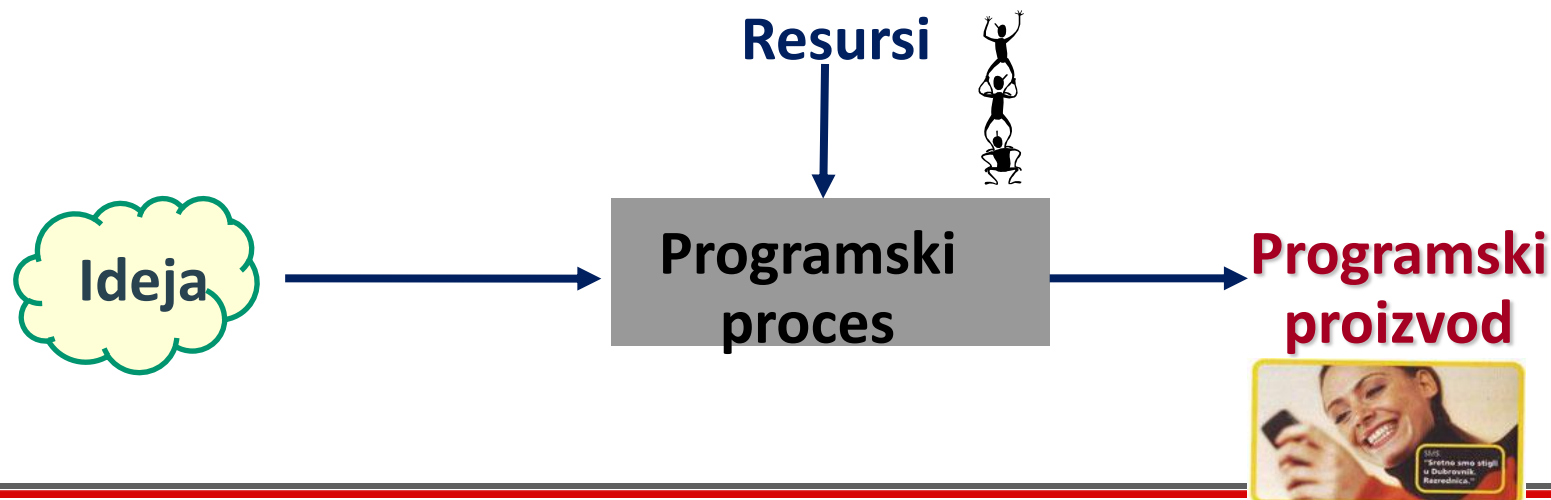
Detaljniji
ciljevi

Veličina kompanije
Resursi
Aplikacijska domena



Programski proces

- ◆ Organizirani skup aktivnosti tijekom kojih se programski proizvod razvija
 - proučavanje prirode i tipova aktivnosti
 - organiziranje aktivnosti u proces
 - koordinacija aktivnosti
 - okolina neophodna za efikasnu podršku



Programski proces (2)



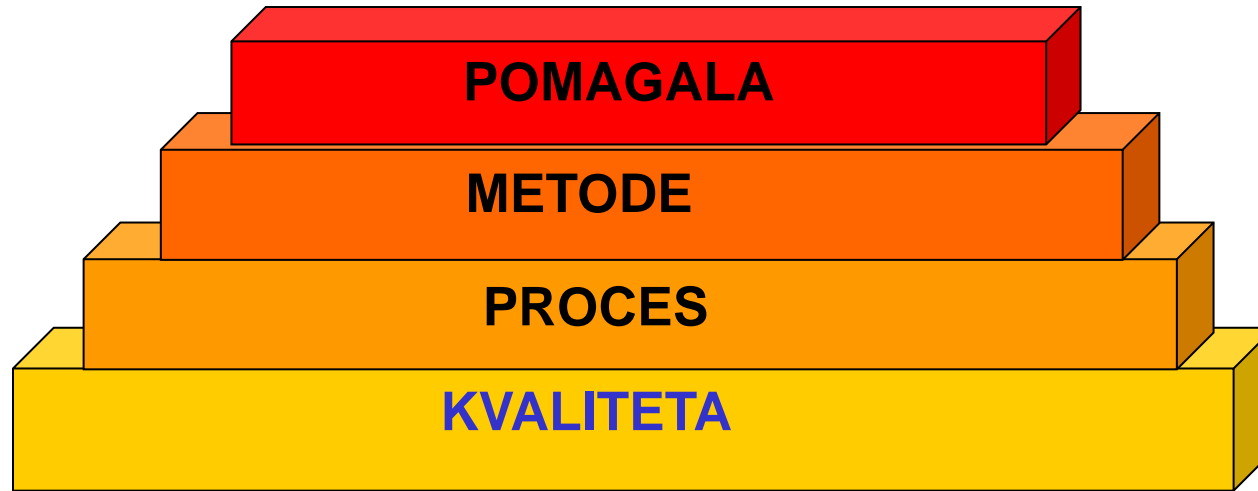
- ◆ Osnovne skupine procesnih aktivnosti zajedničke su svim procesima razvoja bez obzira na primijenjeni model
 - odgovaraju fazama generičkog životnog ciklusa programskog proizvoda (u daljnjem tekstu: PP)
- ◆ Različite organizacije mogu imati međusobno različite procese za razvoj istog tipa PP
- ◆ Ista organizacija može koristiti različite procese za razvoj različitih PP

- ◆ Svijest o važnost procesa unutar programskog inženjerstva
 - intenzivnije fokusiranje početkom 90-ih
 - prepoznat utjecaj procesa na kvalitetu programskog proizvoda / softverskog dijela nekog velikog sustava

- ◆ Akademska zajednica – novi alati i tehnike za podršku procesima
 - empirijska istraživanja procesa i njihovih poboljšanja

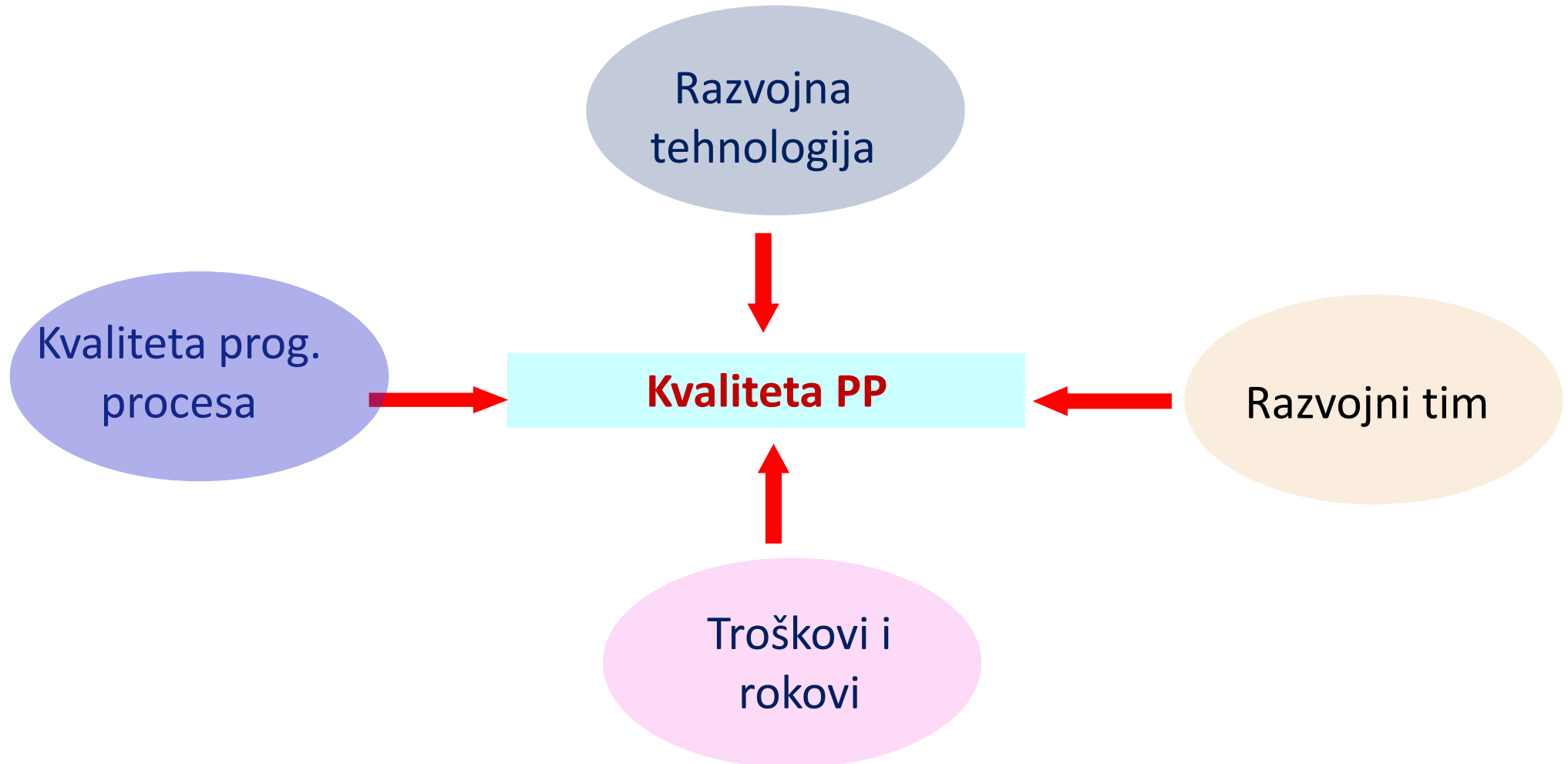
- ◆ Problemi vezani uz programske procese
 - povezani s drugim disciplinama, posebno menadžmentom (razlika ponekad u terminologiji)

Uloga programskih procesa



Glavno područje za poboljšanje kvalitete programskog proizvoda jer sadrži aktivnosti tijekom kojih se programski proizvod ili neki njegov dio stvarno kreira!!!

Uloga programskih procesa (2)



Definicija programskih procesa



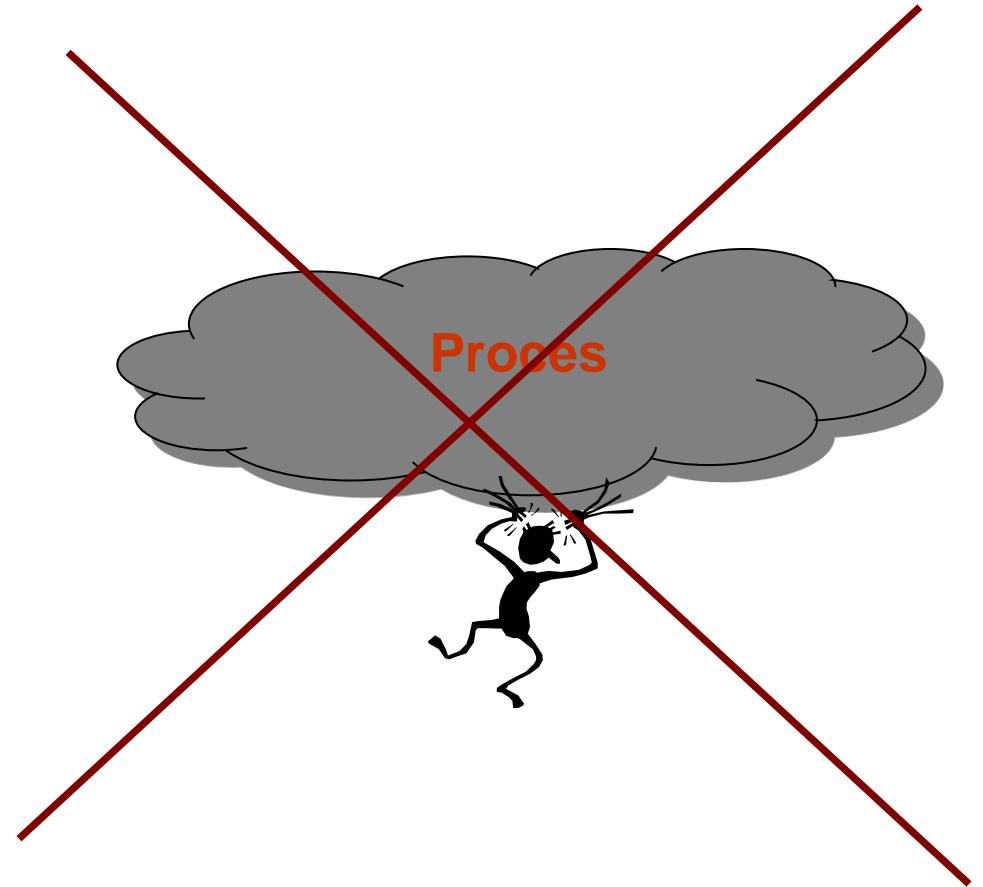
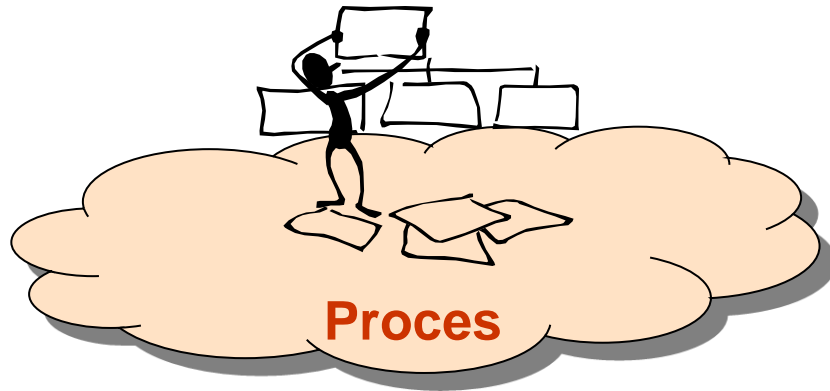
- ◆ Razlozi definicije programskih procesa
 - povećanje kvalitete programskog proizvoda
 - olakšavanje razumijevanja i komunikacije
 - podrška poboljšanju procesa
 - podrška upravljanju procesima
 - automatizacija procesu

- ◆ Način na koji se proces definira ovisi o razlozima definicije

- ◆ Svi predefinirani procesi – čak i standardizirani – prilagođuju se lokalnim potrebama
 - kontekst organizacije
 - veličina projekta
 - regulatorni zahtjevi
 - industrijska praksa
 - organizacijska kultura

- ◆ Alati za podršku
 - podržavaju izvođenje procesa
 - upućuju ljude pri izvođenju pojedinih aktivnosti
 - analiziraju proces – simulatori

Značajke programskih procesa



Značajke programskih procesa (2)

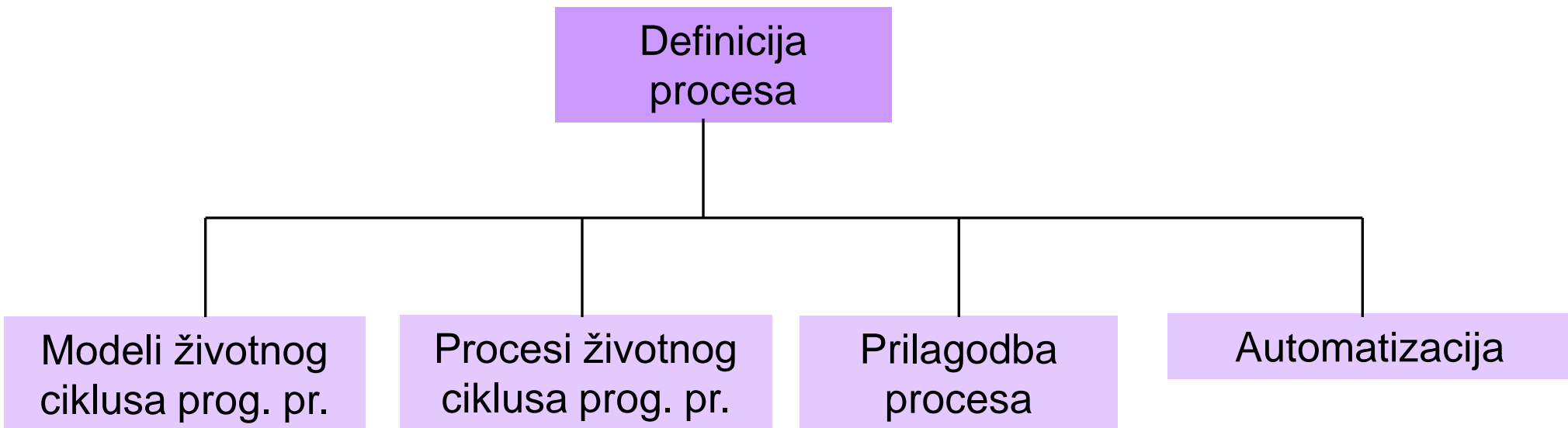


- ◆ Najčešći problemi vezani uz prog. procese
 - često nisu (dobro) definirani
 - aktivnosti vezane uz programske procese jako ovise o individualnim sposobnostima i umijeću sudionika procesa
 - teški za upravljanje
 - kompleksnost (komponente, interakcije, stanja)
 - promjenjivost
 - nevidljivost
 - nemoguće optimizirati sve značajke

Aspekti analize programskih procesa



- ◆ Programski procesi se mogu analizirati s različitih aspekata, kao što je prikazano na slici
- ◆ Svrha svih procesa i njihovih analiza i poboljšanja je isporuka kvalitetnog PP na vrijeme i u okviru budžeta



- odnosi se na definiciju, implementaciju, mjerenja, upravljanje, mijenjanje i poboljšanje programskog procesa
- ◆ Proučavanje programskih procesa
- ◆ Tehničke i menadžerske aktivnosti unutar životnog ciklusa PP
- ◆ Definicija, implementacija, upravljanje, promjena, poboljšanje programskih procesa

Mitovi o procesu razvoja PP-a

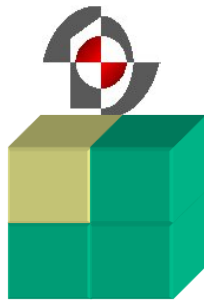


1. Postoji razvoj bez procesa

2. Definiranje procesa
usporava razvoj



Mitovi o procesu razvoja PP-a



3. Laka definicija procesa = lagan razvoj*

Definicija procesa na projektu P2
+
Ad hoc znanje na projektu P2



© Croz

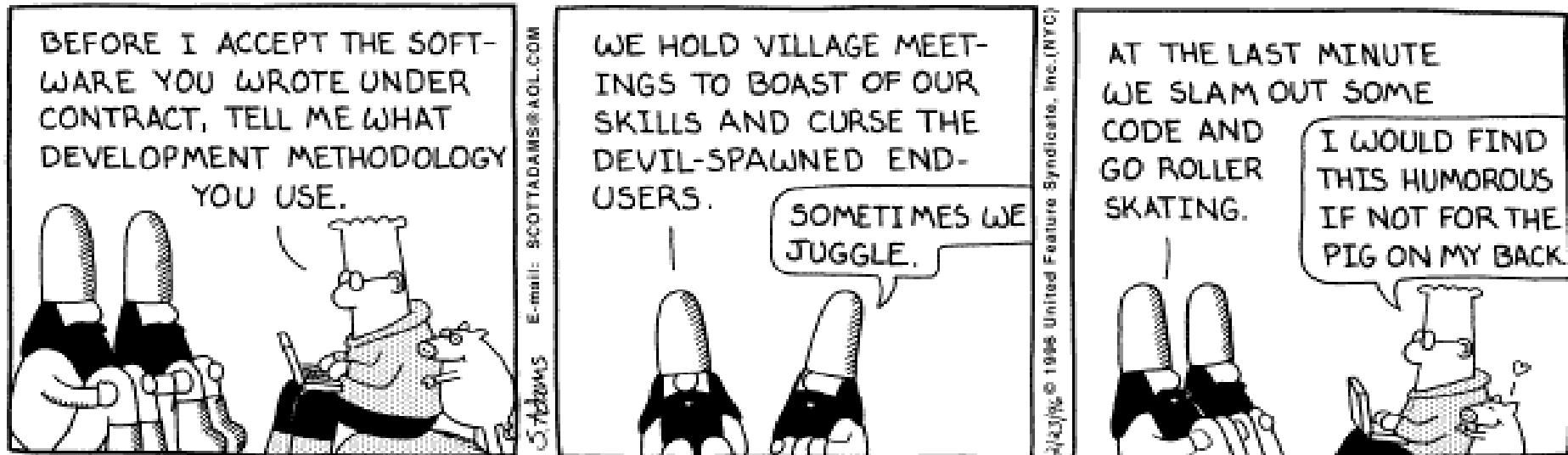
* Ivar Jacobson 2004

- ◆ Modeli životnog ciklusa PP (*SW development models*)
 - definiraju slijed aktivnosti koje se izvode pri razvoju i održavanju
 - na različite načine kombiniraju različite procesne aktivnosti
 - identificiraju redoslijed, ulaze i izlaze iz pojedinih aktivnosti
 - ne osiguravaju veliku razinu detalja, nego prikazuju samo ključne aktivnosti i njihove ovisnosti
- ◆ Definirane su različite norme (*standards*)
 - specificiraju pojedini model (npr. ISO 9000) ili
 - definiraju načine procjene i poboljšanja postojećih modela (npr. ISO 15504)

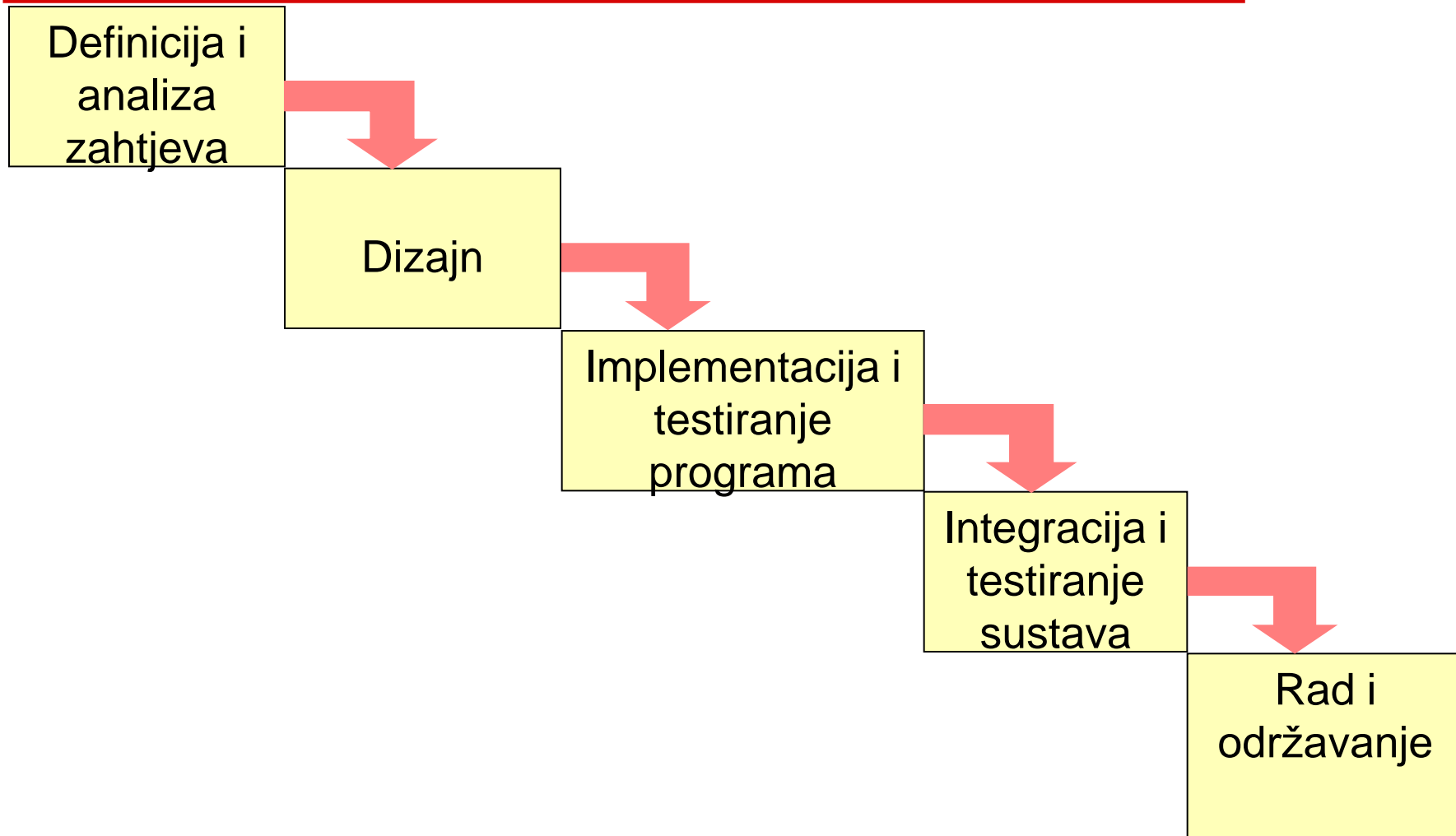
Ad hoc model



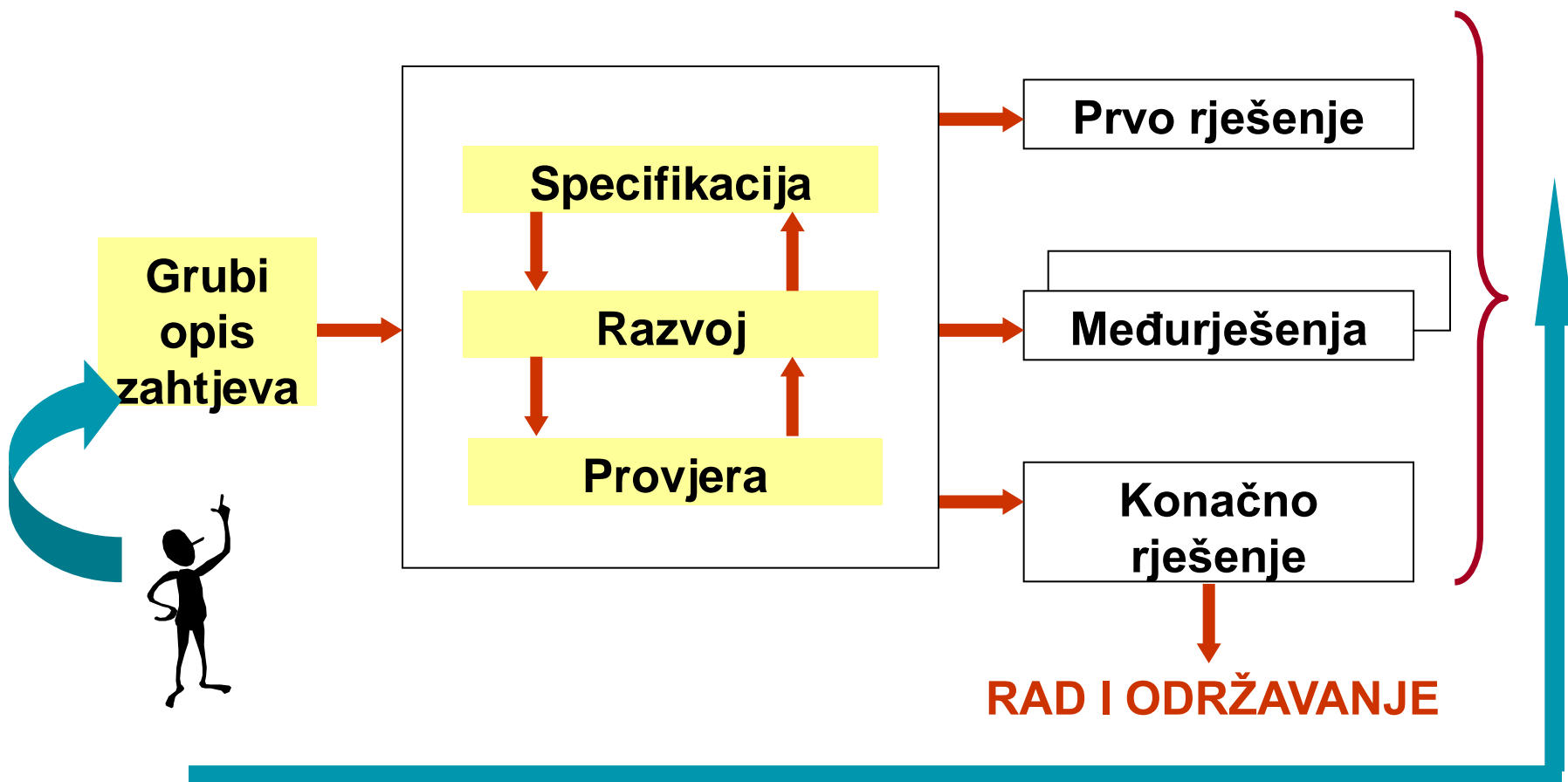
- ◆ Code And Fix
- ◆ Nespecificirani zahtjevi
- ◆ Nedefinirana arhitektura
- ◆ Razvije se prototip proizvoda koji se ponovno i ponovno razvija dok korisnik nije zadovoljan



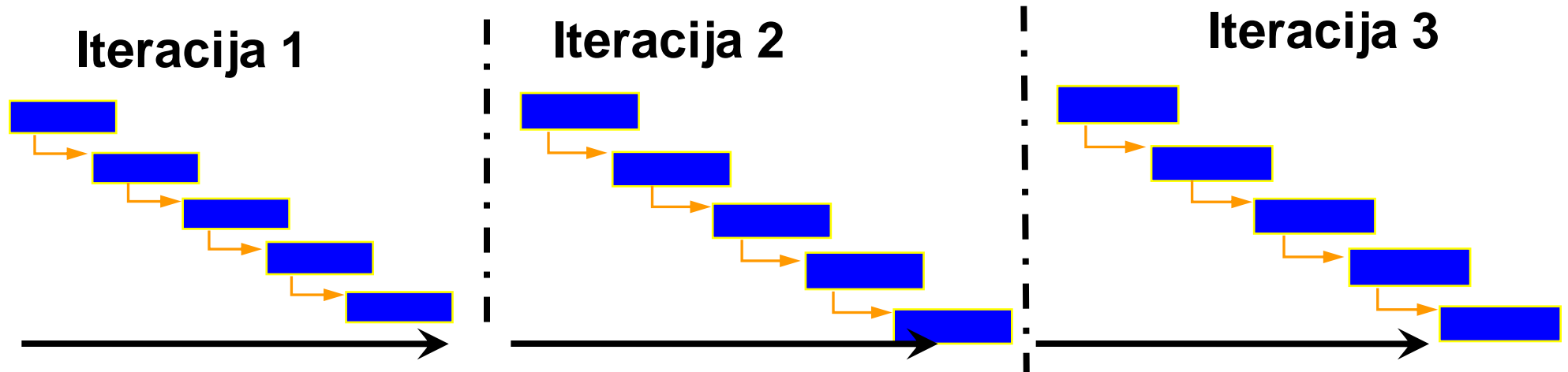
Vodopadni model



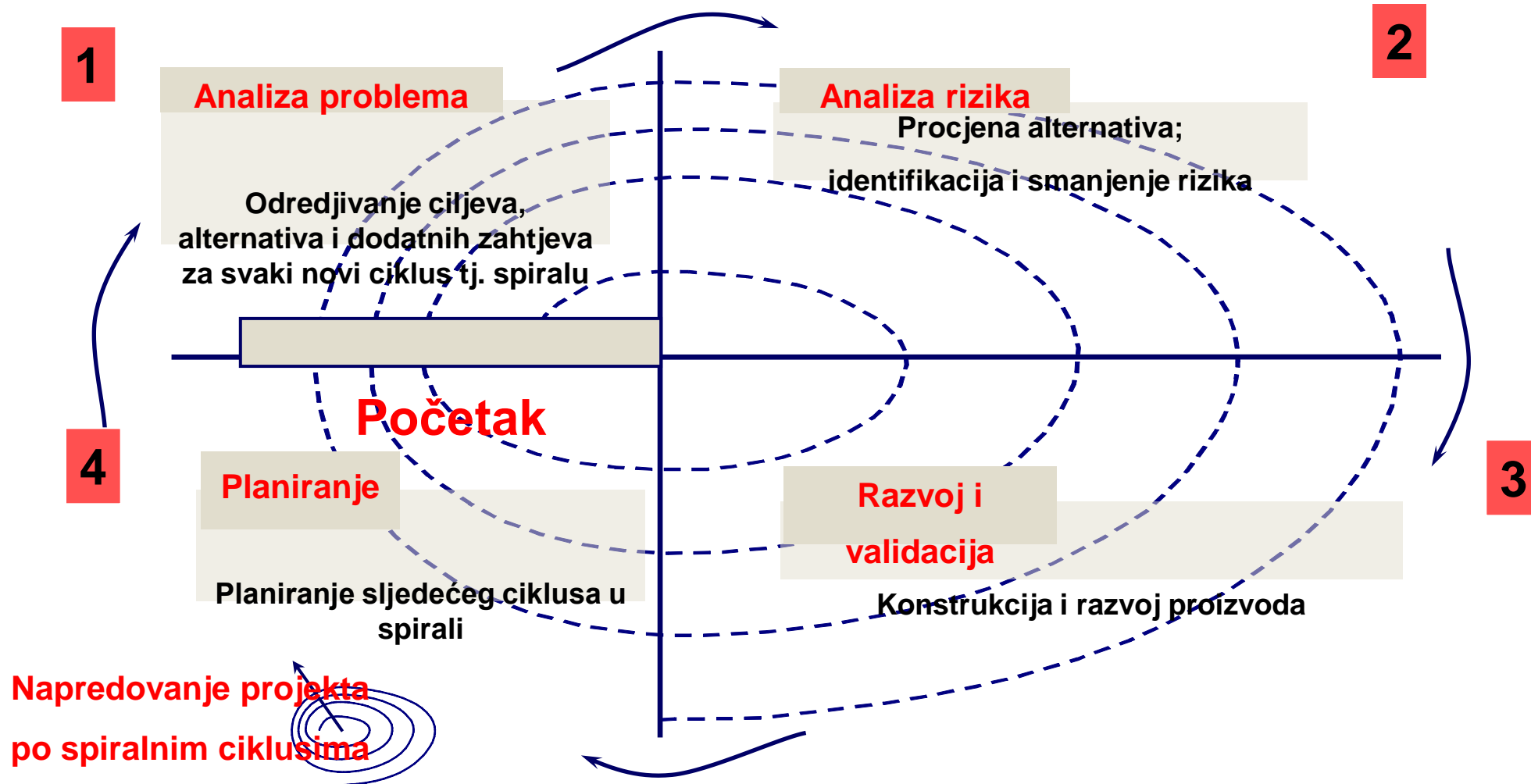
Evolucijski model

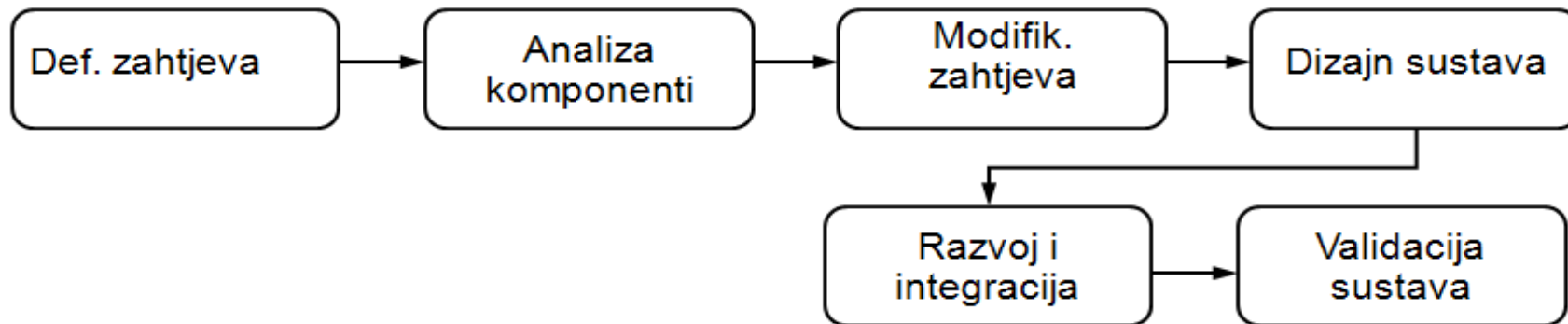


Iterativni model



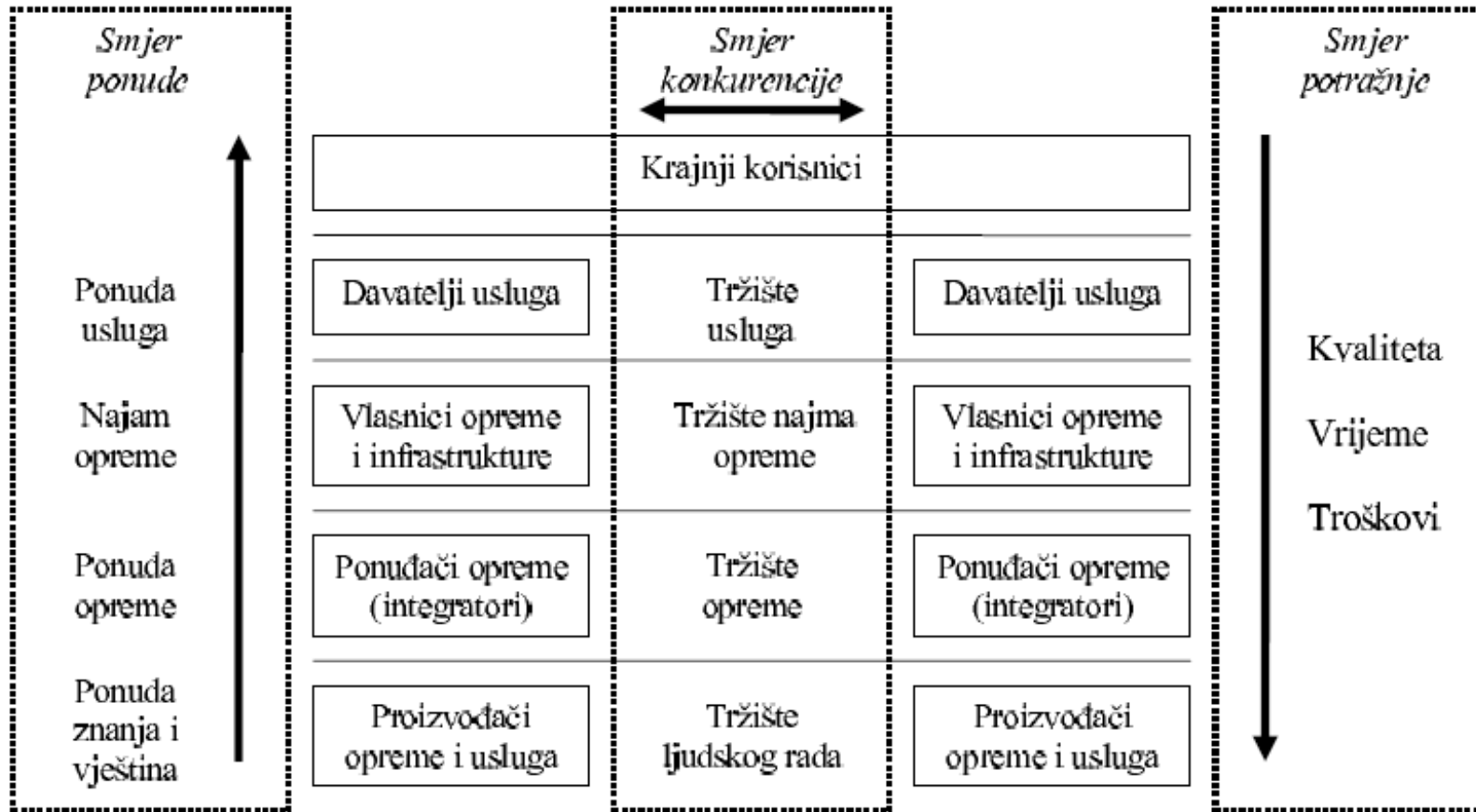
Spiralni model





Konkurentnost u softverskoj industriji i upravljanje kvalitetom

Softverska industrija



Referenca [3]

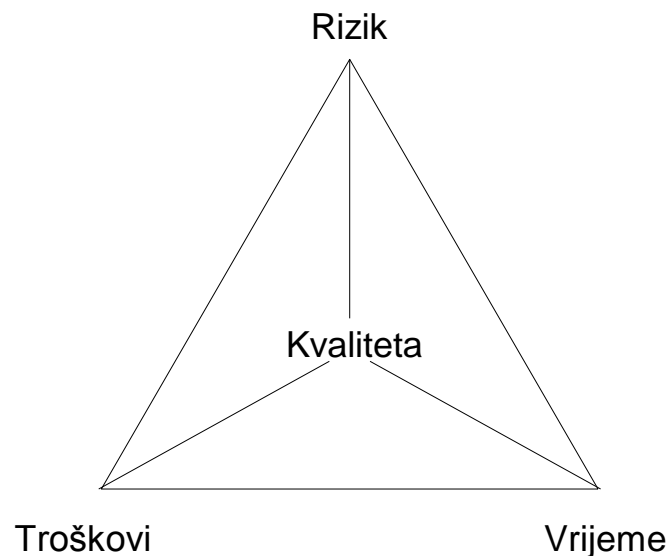
Razvoj konkurencije na četiri razine, ostvaruje se između:

- ◆ davatelja usluga na tržištu usluga
- ◆ vlasnika opreme i infrastrukture na tržištu najma opreme
- ◆ ponuđača opreme na tržištu ponude opremom
- ◆ proizvođača opreme na tržištu rada.

Porast konkurencije:

- ◆ pooštavanje osnovnog skupa zahtjeva na razvoj, izvođenje i održavanje programskog proizvoda

- ◆ posljedica porasta konkurencije
 - stezanje osnovnog trokuta
 - proizvođači se orijentiraju na inovativnost i učinkovitost razvoja
- ◆ inovativnost – nove i privlačne usluge za korisnike
- ◆ učinkovitost razvoja i održavanja – osvajanje konkurencijske prednosti



Na razini vlasnika telekomunikacijske opreme i infrastrukture [3]

- ◆ značajno pooštrevanje zahtjeva na kvalitetu proizvoda.

(Primjerice, svaki kvar na telekomunikacijskoj opremi koji uzrokuje prekid svih poziva u vremenu direktno je povezan s gubitkom ostvarene zarade. Još važnije, učestali prekidi rada uzrokuju gubitak povjerenja korisnika što može imati puno dalekosežnije posljedice.)

- ◆ porast konkurencije na tržištu rada

(Time se unose novi zahtjevi na organizacije proizvođača opreme, a to su povećana produktivnost i učinkovitost kako bi se dobili novi i zadržali postojeći poslovi.)

- ◆ Odnosi se na sve aktivnosti unutar životnog ciklusa programskog proizvoda koje služe za postizanje veće kvalitete programskog proizvoda u eksploataciji

Kvaliteta [ISO/IEC 8402]

- ◆ skup svojstava i značajki proizvoda, procesa ili usluga koje se odnose na mogućnost zadovoljenja utvrđene ili neizravno izražene potrebe.

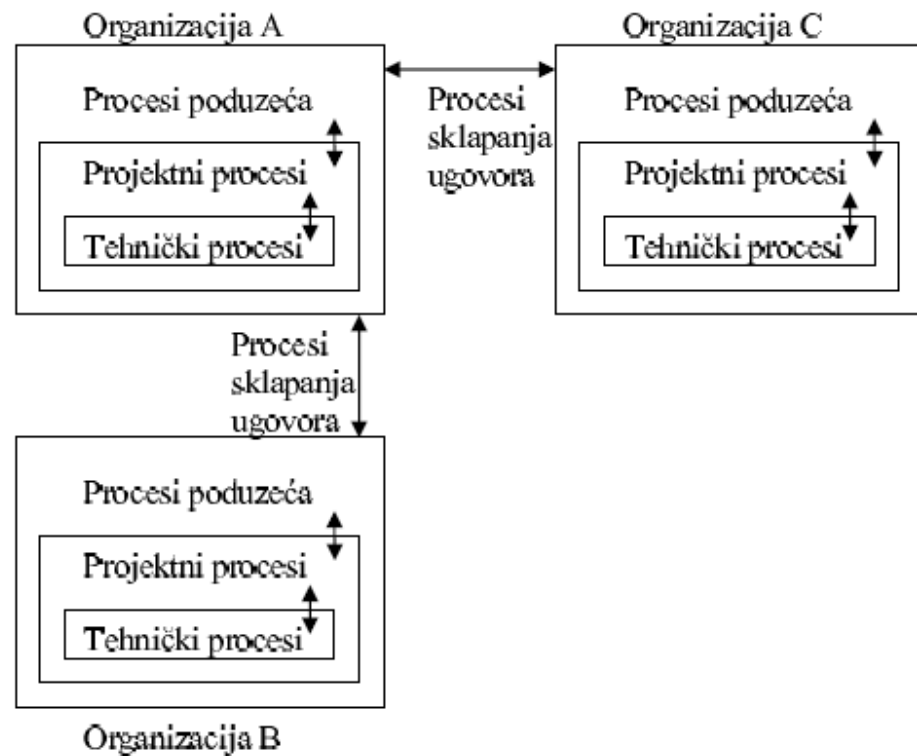
- ◆ Neophodne za efikasno upravljanje kvalitetom.
- ◆ Internacionalne, nacionalne, organizacijske, projektne

Norme se mogu odnositi na:

- ◆ **Prog. proizvode** – definicija ključnih značajki koje svi elementi prog. proizvoda moraju zadovoljavati (npr. zajednički stil programiranja).
- ◆ **Procese** – načini kako se definiraju i poboljšavaju
 - ◆ Enkapsulacija najboljih praksi
 - ◆ Okvir za proces osiguranja kvalitete
 - ◆ Kontinuitet (novi zaposlenici razumijevanje djelovanja organizacije)

Procesi životnog ciklusa sustava grupirani su u sljedeće četiri kategorije:

- ◆ procesi sklapanja ugovora,
- ◆ procesi poduzeća,
- ◆ projektni procesi i
- ◆ tehnički procesi



- ◆ Prva međunarodno prihvaćena norma kvalitete
 - definira osnovne značajke procesa neovisno o vrsti djelatnosti organizacije (proizvodne ili uslužne).
 - omogućeno je certificiranje poduzeća čime se garantira kvaliteta proizvoda ili usluge
 - definira posebne norme za primjenu u razvojnim procesima programskog proizvoda
- ◆ Osnovna svrha modela
 - mogućnost definiranja zahtjeva kvalitete
 - mogućnost procjena kvalitete programskog proizvoda
- ◆ Model je definiran pomoću skupa svojstava i značajki programskog proizvoda kojima se može opisati i procijeniti kvaliteta programskog proizvoda.

ISO 9001

- za organizacije koje dizajniraju, razvijaju i održavaju svoje proizvode (od ideje do realizacije)
- definira generički model procesa kvalitete
- utvrđuje se pojedinačno za svaku organizaciju

ISO 9000 certifikat

- dokumentiranje procedura i standarda (*quality manual*)
- eksterno tijelo ocjenjuje *odgovara li quality manual ISO 9000 standardu*
- korisnici sve više zahtijevaju da dobavljači imaju ISO certifikat

ISO 9000 i upravljanje kvalitetom



Organizacijske norme za proizvode i procese



Proizvodi	Procesi
Formular za pregled dizajna	Način provođenja pregleda dizajna
Standardi imenovanja dokumenata	Dokumenti za upravljanje konfiguracijom
Formati zaglavlja procedura	Proces izdavanja verzija
Format projektnog plana	Proces odobravanja plana projekta
Formulari zahtjeva za promjenom	Proces upravljanja promjenama
Standard programiranja	Proces dokumentirana testiranja

- ◆ Razvijatelji ih smatraju neažurnima i premalo relevantnim.
- ◆ Premala automatiziranost - dosta manualnog posla, popunjavanje formulara.

Rješenja:

- ◆ razvijatelji moraju shvatiti ideju korištenja predloženog standarda, koji su ciljevi i prednosti (razvijatelji trebaju biti aktivno uključeni u proces razvoja)
- ◆ osigurati automatiziranu podršku

Učinak i učinkovitost razvoja

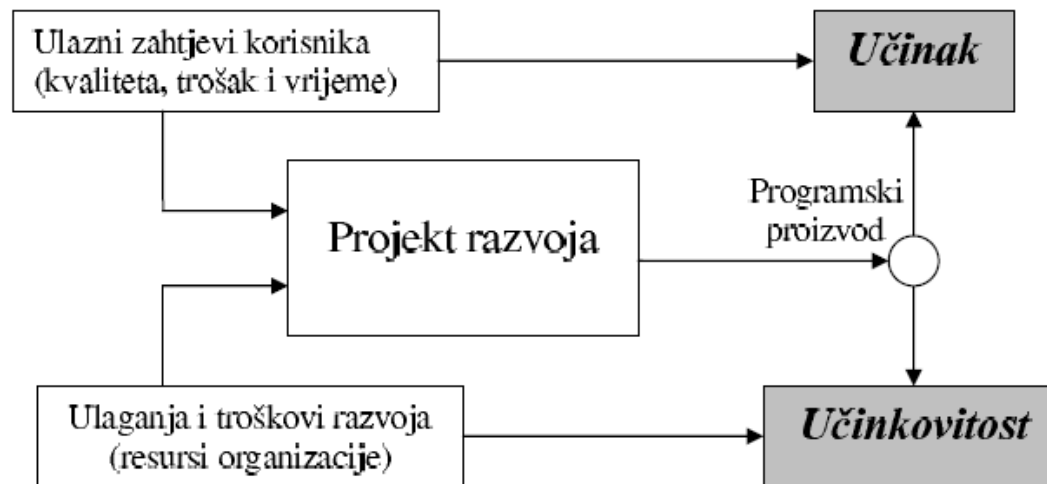


Učinak (eng. *effectiveness*)

- ♦ sposobnost ispunjenja unaprijed zadanih ciljeva

Učinkovitost (eng. *efficiency*)

- ♦ ispunjenost ciljeva uz minimalno korištenje resursa



Učinak i učinkovitost razvoja (2)



- ◆ Za vrijeme trajanja razvoja programskog proizvoda ulaže se u kvalitetu
 - postizanje što veća kvaliteta programskog proizvoda u eksploataciji (smanjenje troškova održavanja uslijed loše kvalitete)

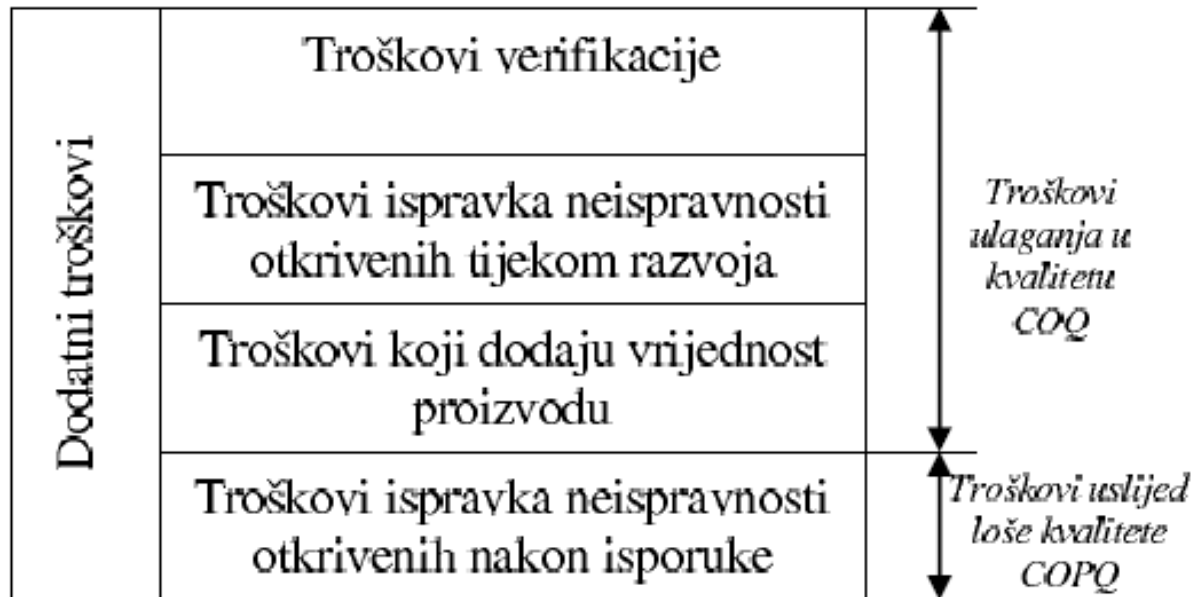
Troškovi ulaganja u kvalitetu (*Cost of Quality, COQ*)

- ◆ svi troškovi (ulaganja) unutar razvojnog i proizvodnog dijela ciklusa proizvoda s ciljem da se postigne veća kvaliteta izlaznog proizvoda.

Troškovi uslijed loše kvalitete proizvoda (*Cost of Poor Quality, COPQ*)

- ◆ svi troškove tijekom eksploatacije prouzročeni lošom kvalitetom izlaznog proizvoda (troškovi uslijed neispravno obavljenog posla po prvi put ili neuspjeha u zadovoljavanju zahtjeva korisnika).

Učinak i učinkovitost razvoja (3)



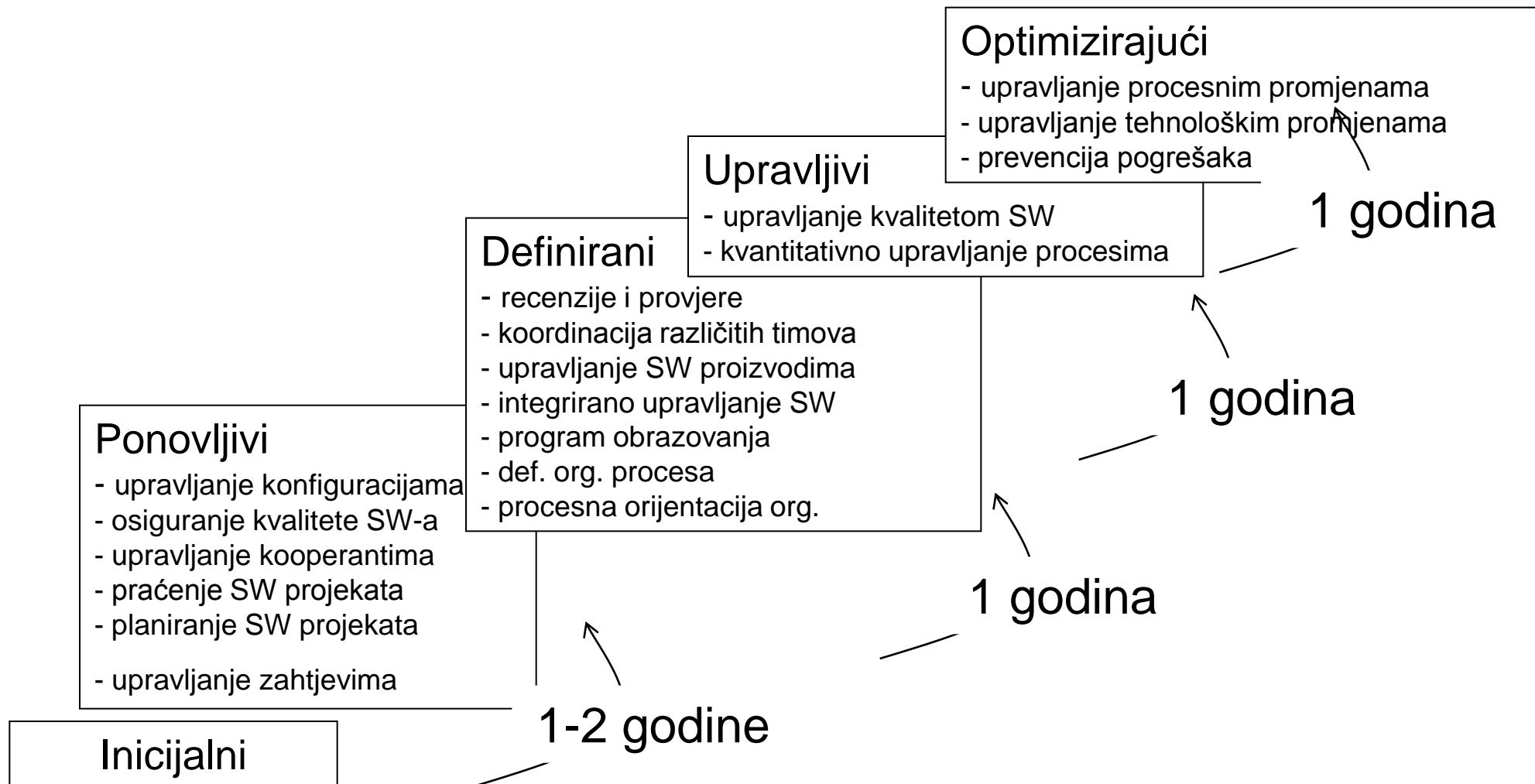
- ◆ Učinkovito upravljanjem kvalitetom PP
 - upravljanje aktivnostima ulaganja u kvalitetu i aktivnostima uslijed loše kvalitete, a uz minimalne ukupne troškove

- ◆ Kvaliteta kao strategija razvoja konkurencijske prednosti i postizanja poslovnog uspjeha
 - prepoznata ranih šezdesetih godina 20 st.
 - začetnici Joseph Juran, Philip B. Crosby i Kaoru Ishikawa
- ◆ Pristup upravljanja organizacijom usredotočen na kvalitetu proizvoda koji se razvija ili proizvodi
 - osnovni cilj programa je ugrađivanje svijesti o kvaliteti u sve organizacijske procese na svim razinama
 - zasnovan na sudjelovanju svih zaposlenika
 - s ciljem postizanja dugoročnog poslovnog uspjeha kroz zadovoljstvo korisnika i zaposlenika
- ◆ Strategija je fokusirana na svjesnost ljudi u organizaciji te njihovom doprinosu kontinuiranoj prilagodbi potrebama korisnika

- ◆ *Software Process Improvement and Capability Evaluation*
 - u ranim devedesetim zajednička radna grupa ISO i IEC (eng. International Electrotechnical Committee, IEC)
 - novi naziv radne grupe *Software Process Improvement and Capability Determination*
- ◆ Rezultat rada - **ISO/IEC 15504** norma koja daje okvir za procjenu procesa životnog ciklusa programskog proizvoda.
 - Prva revizija norme analizirala je isključivo procese razvoja programskog proizvoda
 - Norma se proširila i na sve ostale poslovne procese razvoja programskog proizvoda
- ◆ Specificira okvir mjerenja koji se može koristiti zajedno s različitim referentnim procesnim modelima.

- ◆ Modeli za procjenu zrelosti organizacija
- ◆ **CMM** (*Capability Maturity Model*)
- ◆ **CMMI** (*Capability Maturity Model Integration*)
- ◆ sredinom osamdesetih godina, Software Engineering Institute, (SEI) pri Sveučilištu Carnegie Mellon u Pittsburghu
- ◆ Model za poboljšanje organizacijskih procesa - skup praksi koje su se dokazale u svojoj primjeni u programskoj industriji
- ◆ Model zrelosti organizacije s više razina i kontinuirani model poboljšanja procesa razvoja

- ◆ Organizacije se mogu procjenjivati obzirom na ljestvicu od pet razina zrelosti:
 - inicijalna razina (*initial*)
 - razina ponovljivosti procesa (*repeatable*)
 - razina definiranih procesa (*defined*)
 - razina procesa pod nadzorom (*managed*)
 - razina optimizirajućih procesa (*optimizing*)
- ◆ Kontinuirano poboljšanje procesa razvoja
 - rezultira postepenim penjanjem organizacije po predloženim razinama zrelosti
 - cilj je doseći razinu optimizirajućih procesa na kojoj se dostiže poslovna izvrsnost.

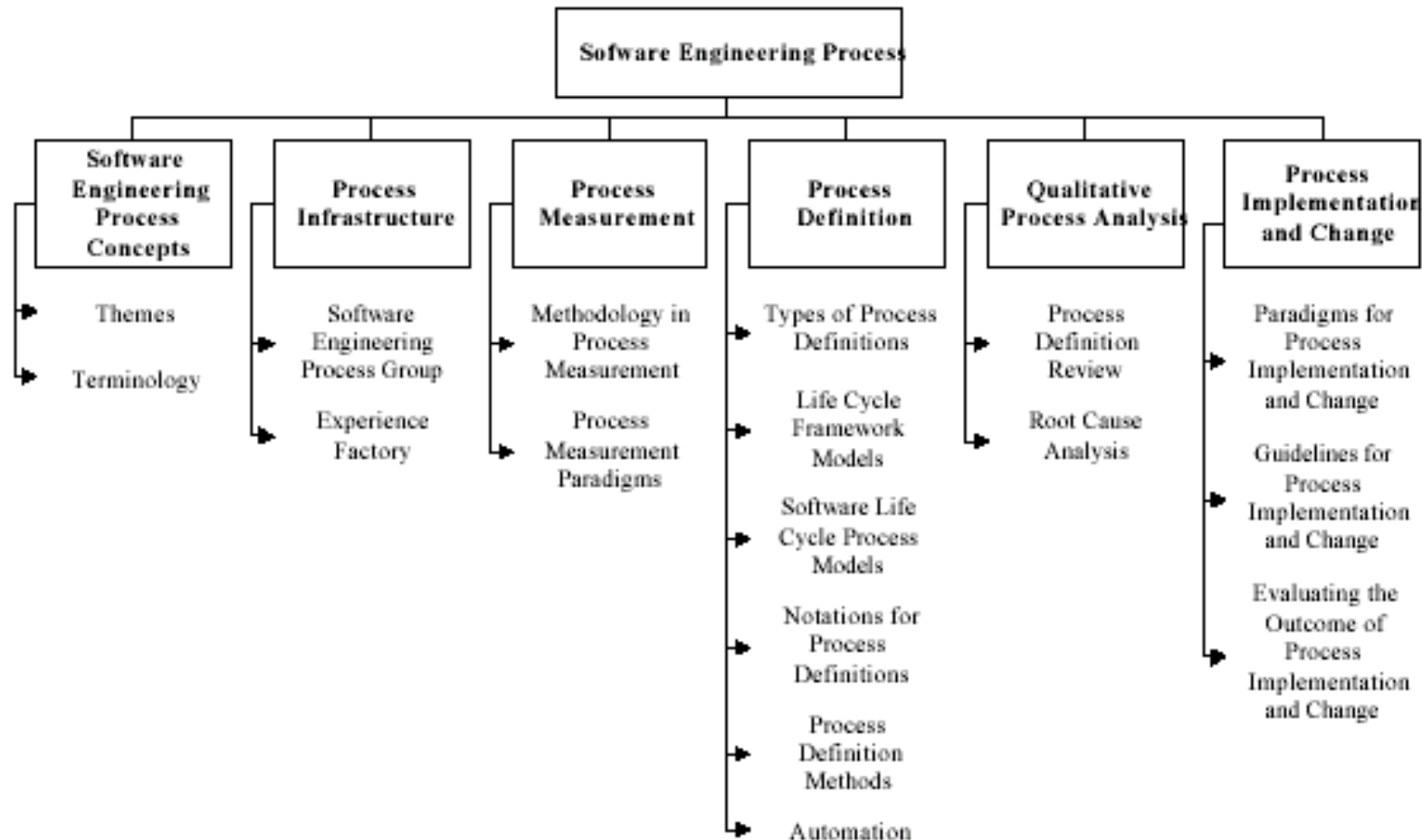


- ◆ CMM integracijski projekt (CMM Integration, CMMI)
 - Pokreće se zbog pojave višestrukih modela organizacijske zrelosti programskih procesa
 - ciljem je integracija aktivnosti poboljšanja procesa razvoja iz različitih disciplina u organizaciji
- ◆ unosi dodatnu zasebnu disciplinu mjerenja temeljenoj na SPICE normi i uključuje dodatno procesno područje Mjerenje i analiza (*Measurement and Analysis*), definiranu za drugu razinu zrelosti
- ◆ definiranje i uspostava mehanizma upravljanja informacijama koje su neophodne za praćenje i kontrolu svakog definiranog procesnog područja na jedinstven način

- ◆ CMM i CMMI postaju sve popularniji
- ◆ Organizacije se mogu samostalno procjenjivati i pozicionirati na ljestvici zrelosti
 - koristeći CMM upitnik za evaluaciju stupnja zrelosti organizacija
 - Osnovna motivacija je analiza vlastitih mogućnosti i snage, te utvrđivanje područja za poboljšanje.
- ◆ Nezavisne procjeniteljske organizacije koje provode analizu zrelosti organizacija u programskoj industriji.
- ◆ Kao rezultat procjenjivanja
 - organizacije dobivaju certifikat koji postaje neizostavan materijal u konkurencijskoj utrci.

Modeliranje i analiza programskih procesa

Aspekti istraživanja procesa ŽCPP



- ◆ meta razina
 - skup općenitih koncepata za analizu i definiranje modela procesa koji će se koristiti za izvedbu projekta razvoja i/ili održavanja programskog proizvoda
- ◆ razina modela procesa
 - dokumentiranje, komuniciranje i poboljšavanje predloženog okvira procesa koji će se koristiti tijekom projekta (razvoja i/ili održavanja) za određeni programski proizvod
- ◆ razina izvođenja

- ◆ **Aktivni elementi** modela su oni elementi koji opisuju različite aktivnosti modela, odnosno koji prikazuju odvijanje procesa u nekom realnom sustavu koji se modelira.
 - nezavisno paralelni
 - kooperativni
 - konkurentni

- ◆ **Pasivni elementi** modela se generiraju tijekom vremena života modela. Trebaju imati mogućnost pohranjivanja proizvoljnih informacija, a generiraju se s obzirom na svrhu zbog koje je model razvijen.

- ◆ *Statički i dinamički elementi* modela
 - definiraju se s obzirom na sposobnost zadržavanja svojih karakteristika i ponašanja tijekom vremena života modela, (odnosno tijekom analize modela)
- ◆ Element nekog modela je *statički* ako on postoji cijelo vrijeme analize. To su na primjer događaji i vrijeme. Ostali statički elementi su varijable ili kompleksne strukture podataka.

- ◆ Za dinamičke elemente modela tijekom izvođenja analize nad modelom razlikuje se nekoliko faza:
 - **Inicijalna faza:** model je u svojem početnom stanju.
 - **Faza izvršavanja:** nad modelom se vrši analiza, pri čemu se događaju promjene elemenata
 - Generiranje elemenata u izvoru.
 - Usmjeravanje, tok, transfer elemenata kroz mrežu.
 - Čekanje elemenata, vrijeme čekanja, rep.
 - Posluživanje elemenata, vrijeme posluživanja, poslužitelj.
 - **Završna faza:** prikaz i procjena promatranih veličina. Rezultati se pohranjuju radi buduće analize.

Postupci pri modeliranju programskog procesa



- ◆ Formulacija problema
- ◆ Definiranje formalnog modela
- ◆ Razvoj računalnog modela
- ◆ Validacija i verifikacija modela
- ◆ Podešavanje modela
- ◆ Izvođenje eksperimenata nad modelom
- ◆ Procjena modela



Mreže repova

- ◆ Osnovne pojave koje se analiziraju pri modeliranju nekog sustava primjenom teorije repova jesu:
- ◆ dolazni proces
- ◆ proces posluživanja
- ◆ disciplina posluživanja
- ◆ broj poslužitelja

Osnovne performanse sustava čekanja i posluživanja



- ◆ Prosječno vrijeme koje korisnik provede u sustavu/repu.
- ◆ Prosječan broj korisnika u sustavu/repu.
- ◆ Vjerojatnost da će korisnik morati čekati na uslugu.
- ◆ Faktor iskoristivosti (opterećenje poslužitelja).

- ◆ *Notacija*
 - Koristi se standardna *Kendall-Lee* notacija $F/H/m$, gdje F označava distribuciju međudolaznih vremena, H označava distribuciju vremena posluživanja, a m predstavlja broj paralelnih poslužitelja u sustavu.

Osnovne performanse sustava čekanja i posluživanja



- ◆ *Međudolazno vrijeme* (razlika vremena nailazaka dva uzastopna korisnika-a): t_{ai}
- ◆ $t_{ai} = t_i - t_{i-1}$

- ◆ *Vrijeme čekanja*: T_w
- ◆ *Vrijeme zadržavanja*: T_q
- ◆ *Vrijeme posluživanja*: T_s

$$T_q = T_w + T_s$$

Osnovne performanse sustava čekanja i posluživanja



- ◆ *Duljina repa: L_w*
- ◆ *Srednji broj korisnika u sustavu: L_q*

- ◆ *Repovi s prioritetima*
- ◆ *Ukupni intenzitet nailazaka:*

$$\lambda = \sum_{i=1}^3 \lambda_i \quad (2.13)$$

Primjena simulacije u analizi programskih procesa

Općenito, simulacija omogućava uvid u kompleksno ponašanje sustava.

- ◆ Simulacija se može primijeniti u mnogim kritičnim područjima:
 - specifikacija zahtjeva
 - poboljšanja procesajer omogućava uvid u dizajn procesa prije investiranja velikih sredstava za njegovu promjenu.
- ◆ Softverski procesi sadrže mnoge povratne veze koje su povezane s ispravljanjem neispravnosti u dizajnu procesa ili kôdu softverskog produkta.
- ◆ Uslijed toga nastaju vremenska kašnjenja koja mogu varirati od nekoliko minuta do nekoliko godina.

Primjena simulacije u menadžmentu i inženjerstvu programskih procesa



- ◆ Podrška
 - odlučivanju
 - menadžment na strategijskoj, taktičnoj i operacijskoj razini
 - planiranju
 - nadzoru projekata
 - razumijevanju procesa
 - od strane sudionika
 - obuka i učenje
 - tehnologija za
 - analizu i smanjenje rizika
 - poboljšanje procesa
 - prilagodbu postojeće i uvođenje nove tehnologije
-

- ◆ Identifikacija svrhe modeliranja i problema koji se žele riješiti je ključna za
 - definiranje opsega i razine apstrakcije modela
 - podataka koje treba prikupiti

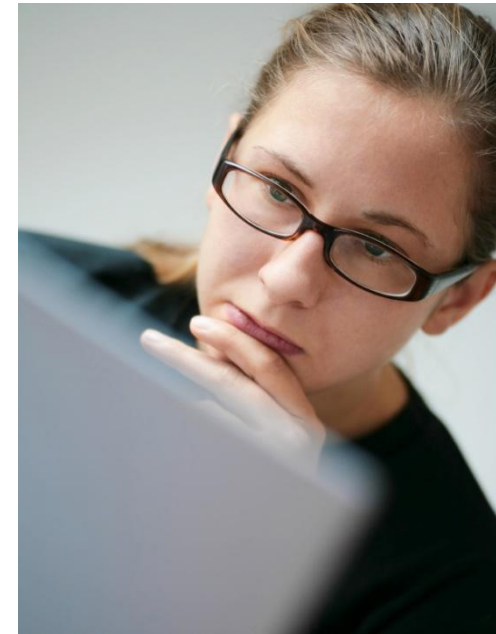
- Treba li posao distribuirati na više lokacija ili centralizirati?
 - Samostalni razvoj ili podugovaratelji (outsourcing)?
 - Primjena COTS (commercial off-the-shelf) komponenti (prilagodba i integracija) ili razvoj komponenti i sustava
 - Razvoj sličnih proizvoda koristeći model proizvodnih linija ili tradiciionalni pojedinačni razvoj svakog pojedinog proizvoda?
 - Dugoročni utjecaj trenutnih ili potencijalnih politika i inicijativa (zapošljavanje, edukacija, poboljšanja procesa?)
-
- ◆ Simulacijski modeli sadrže organizacijske parametre i razvijaju se kako bi odgovorili na pojedina pitanja
 - ◆ Usporedba rezultata simulacije alternativnih scenarija – pomoć pri odlučivanju.

- ◆ predviđanje i analiza rizika
 - troškova / napora, rokova i kvalitete proizvoda
 - potrebnog osoblja tijekom određenog perioda
 - razrješenja problema ograničenja i alokacije resursa
 - razine usluge (npr. za podršku korisnicima / razvoju)
- ◆ Pitanja:
 - Koji procesi će se koristiti u novom projektu, A ili B?
 - Hoće li proces C omogućiti postizanje postavljenih ciljeva?
 - Koji dijelovi procesa D se mogu minimizirati/izostaviti kako bi se smanjili troškovi / vrijeme uz očuvanu kvalitetu?
- ◆ Simuliraju se različiti scenariji procesa i njihove alternative
- ◆ Rezultati se procjenjuju s obzirom na dimenziju promatranja: troškovi, vrijeme kvaliteta

Simulacija i nadzor projekta



- ◆ Simulacija olakšava praćenje i nadzor projekta
- ◆ Ključni projektni parametri (status, napredak, iskorištenje resursa itd.) se snimaju i uspoređuju s planiranim vrijednostima dobivenim simulacijom



- ◆ Pomoć pri donošenju odluka vezanih uz poboljšanja procesa
 - *go / no-go* za pojedini prijedlog ili dodjela prioriteta različitim prijedlozima
- ◆ predviđanje utjecaja uvođenja promjene u proces prije implementacija u stvarnu praksu
 - simulacijski modeli se razvijaju isto kao pri planiranju.
- ◆ Post-evaluacija već implementiranih procesa
 - za kalibraciju procesnog modela (veličina, resursi, ograničenja)

- ◆ Želi se postići bolje razumijevanje procesa od strane korisnika
 - menadžeri, razvijatelji, ljudi koji se brinu za osiguranje kvalitete
 - razumijevanje redoslijeda i ovisnosti aktivnosti, paralelizama, toka posla, informacija i artefakta, itd.)
 - vizualizacija procesa (animirane simulacije)
 - razumijevanje složenih povratnih petlji i kašnjenja inherentnih procesima (primjer testiranje)
 - razumijevanje konzistentnih i prožimajućih značajki procesa razvoja i održavanja
 - nesigurnost u predviđanju procesnih rezultata
 - varijabilnost rezultata

- ◆ Simulacije pri edukaciji o upravljanju projektima
- ◆ Simulirana okolina koja ukazuje na
 - utjecaj donesenih odluka (najčešće neispravnosti – kodiranje prije dobrog planiranja, izbjegavanje inspekcija, skraćivanje testiranja)
 - svijest o nesigurnosti početnih pretpostavki o rezultatima aktivnosti

- ◆ Modeli procesa zasnovani na stanjima
- ◆ Simulacija diskretnih dogadaja
- ◆ Kontinuirana simulacija
- ◆ Jezici zasnovani na pravilima
- ◆ Modeli Petrijevih mreža
- ◆ Modeli mreža repova
- ◆ CPM (*Critical Path Method*) i PERT (*Program Evaluation and Review Technique*)

◆ Simulacija diskretnih događaja

- dobro specificirani procesni zadaci, repovi i vremenski rokovi, te grananja procesa na osnovu atributa entiteta
- sposobnost animacije osigurava važne informacije s obzirom na vrijeme i koordinaciju toka programskih produkata
- pogodna za procesno orijentirane aplikacije
- kritična ograničenja - teško simuliranje faktora promjene procesa kao što su produktivnost, znanje i iskustvo osoblja, psihološki pritisak uslijed ugovorenih rokova itd.

- ◆ **Kontinuirana simulacija (dinamika sustava)**
 - prezentira kako različite karakteristike poput produktivnosti, ugovorenih rokova, motivacije osoblja i slične, utječu na promjene procesa tijekom vremena
 - pogodna za simulaciju na razini procesa u cjelini, ali nije prikladna za njegovu detaljniju analizu
- ◆ **Simulacija na osnovu stanja**
 - koristi se za simulaciju na nivou procesa u cjelini, i karakterizira je mogućnost dobre grafičke prezentacije sustava
 - nije prikladna za matematičku analizu performansi sustava.

Simulacijske paradigme (2)



- ◆ Ne postoji jedinstveni ili najbolji model

- ◆ Kontinuirana simulacija
 - strateške analize, početne aproksimacije, dugoročni trendovi, analiza značajki na visokoj (globalnoj) razini
 - ne omogućuje analizu na detaljnoj procesnoj razini

- ◆ Simulacija diskretnih događaja i zasnovana na stanjima
 - detaljne analize procesa, korištenje resursa, stvaranje repova i sl.

Analiza najčešće korištenih paradigmi



Table 2: Paradigms applied in simulation models

	1998	1999	2000	2002	2003	2004	2005	2006	2007	total
SD	4	5	7	1	4	3	5	2	1	32
DES	1	2	3	1	2	2	4	1	2	18
SBS				1	1				1	3
KBS	1		1	1						3
QSIM				1				1		2
RPG						1	1			2
ABS							1	2		3
DTS						1				1
Stig.							1			1
<i>Hybrid</i>	<i>1</i>	<i>1</i>	<i>2</i>		<i>1</i>	<i>1</i>	<i>3</i>	<i>1</i>		<i>10</i>

SD: system dynamics

DES: discrete-event simulation

SBS: state-based simulation

KBS: knowledge-based simulation

RPG: role-playing game

ABS: agent-based simulation

Stig.: stigmergy

QSIM: qualitative/semi-quantitative simulation

Emrg.: emergent

DTS: discrete-time simulation

Istraživanje provedeno 2008,
preuzeto iz:

“Software Process
Simulation over the Past
Decade: Trends Discovery
from A Systematic Review”

[1]

- ◆ Vezano uz paradigme
 - Kontinuirana i simulacija diskretnih događaja najčešće korištene
 - Uvođenje novih paradigmi

- ◆ Vezano uz granularnost
 - Veća primjena diskretne simulacije u odnosu na kontinuiranu

Analiza najčešće korištenih paradigmi (2)



Table 3: Paradigms applied in primary studies

	1998	1999	2000	2002	2003	2004	2005	2006	2007
SD	●	●	●	●	●	●	●	●	●
DES	●	●	●	●	●	●	●	●	●
SBS	●		●	●	●				●
KBS	●		●	●					
QSIM				●		●		●	●
RPG					●	●	●		
ABS							●	●	
DTS						●			
Stig.							●		
Emrg.			●						
Number	4	2	5	5	4	5	5	4	4

Istraživanje provedeno 2008,
preuzeto iz:

“Software Process
Simulation over the Past
Decade: Trends Discovery
from A Systematic Review”

[1]

- ◆ Ovisi o primijenjenoj paradigmi simulacije.
- ◆ Kontinuirana simulacija
 - proces se modelira kao generalni sustav na razini sustava
 - ponašanje procesa se opisuje skupom vanjskih parametara koji se kontinuirano mijenjaju tijekom vremena
 - istraživanje makro procesa
- ◆ Diskretna simulacije
 - ◆ proces se modelira detaljno na razini pojedinih aktivnosti
 - prati se stanje pojedinih aktivnosti i njihove međuovisnosti, te stanja parametara za opis pojedinih aktivnosti

Analiza paradigmi s obzirom na granularnost



Table 4: Paradigms in support of granularities

	SD	DES	SBS	KBS	QSIM	RPG	ABS	DTS	Stig.	Emrg.
System	●				●	○				
Process		●	●	●		○		●		
Entity						○	●		●	●

● - inherently supported ○ - condition applying

Istraživanje provedeno 2008,
preuzeto iz:

“Software Process
Simulation over the Past
Decade: Trends Discovery
from A Systematic Review”

[1]

Table 5: Granularity levels of simulation studies

	1998	1999	2000	2002	2003	2004	2005	2006	2007	total
System	4	5	7	2	4	3	5	3	1	34
Process	2	2	4	3	3	3	4	1	3	25
Entity							2	2		4

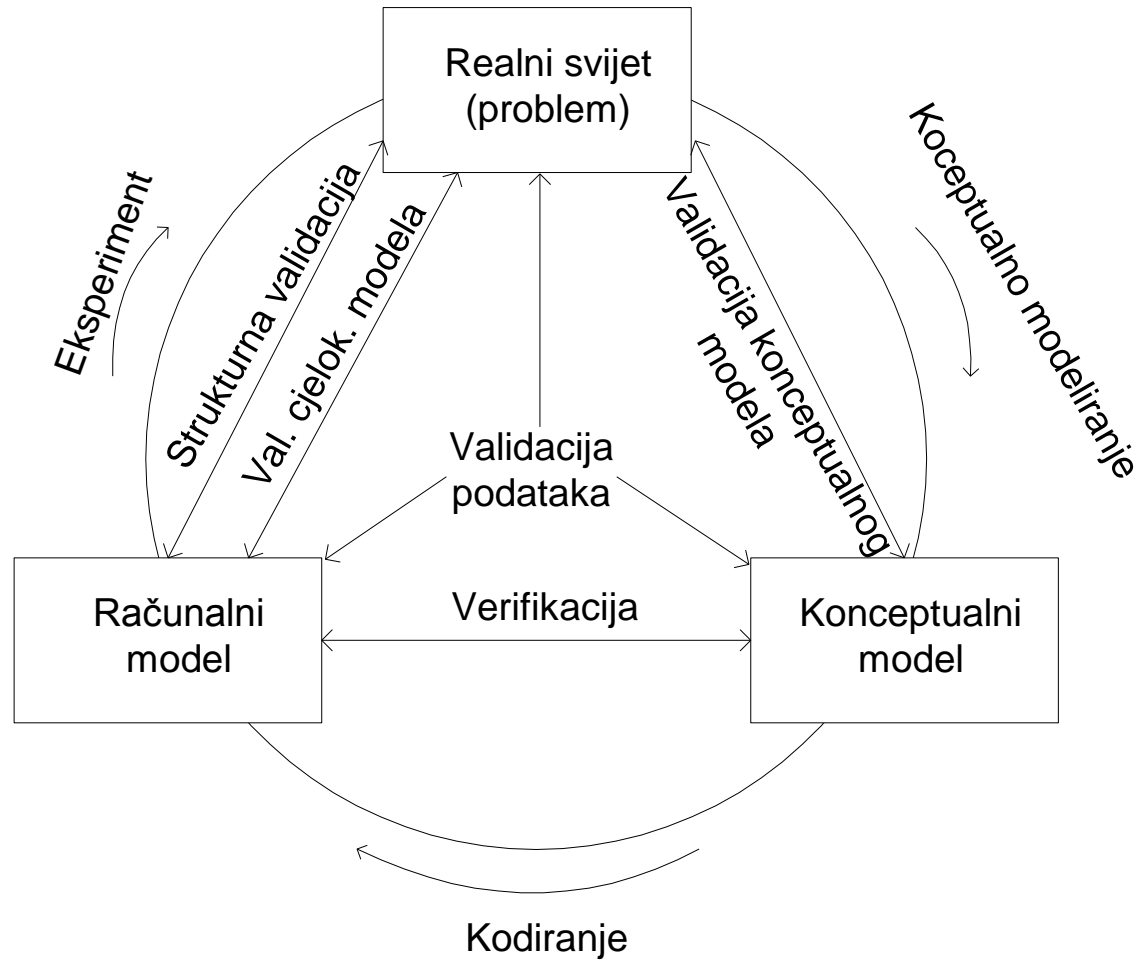
Primjena simulacije u analizi programskih procesa



- ◆ Simulacijom se ne mogu riješiti svi problemi
- ◆ Njezina prediktivnost ovisi prvenstveno o tome je li osnovni konceptualni model nad kojim će se izvršiti simulacija dobro postavljen.
- ◆ Simulacija u programskom inženjerstvu
 - obuhvaća djelovanje ljudskog faktora i ostalih vrijednosti koje nije moguće prikazati numerički (za razliku od drugih tehničkih područja u kojima se modeli osnivaju na dobro utvrđenim fizikalnim zakonima).
 - Teškoće predstavlja prikupljanje podataka ako su izvori sudionici procesa, a reproducibilnost metoda za ocjenu valjanosti simulacijskog modela ne može se standardizirati.

- ◆ *Simulacijski model* je računalni oblik konceptualnog modela, nad kojim se izvodi simulacija primjenom odgovarajućeg softverskog pomagala.
- ◆ *Verifikacija* je proces provjere je li neki konceptualni model ispravno transformiran u računalni model (provjera je li računalni model pomoću kojeg će se realizirati simulacija ispravan u odnosu na konceptualni model).
- ◆ *Ocjena valjanosti ili validacija* je proces provjere je li konceptualni model dovoljno ispravan za potrebe određenog istraživanja

Validacija i verifikacija modela (2)



- ◆ *Validacija konceptualnog modela*
 - jesu li opseg predloženog modela i razina detalja dovoljni s obzirom na svrhu istraživanja,
 - primijenjene pretpostavke ispravne,
 - sadrži li konceptualni model sve neophodne detalje kako bi se ostvarili ciljeve istraživanja.

- ◆ *Validacija podataka*
 - jesu li podaci neophodni za izgradnju modela, njegovu validaciju i eksperimente primjenom simulacije dovoljno precizni.

- ◆ *Strukturna validacija (bijela kutija)*
 - predstavljaju li konstitutivni elementi modela odgovarajuće elemente realnog sustava s dovoljnom ispravnošću.
 - detaljna (mikro) provjera modela na nivou svake pojedine komponente modela.
- ◆ *Validacija cjelokupnog modela (crna kutija)*
 - reprezentira li model u cjelini realni sustav s odgovarajućom ispravnošću
 - predstavlja globalnu provjeru modela.
- ◆ Validacija i verifikacija predstavljaju kontinuirane procese koji se izvode tijekom cijelog životnog ciklusa simulacijskog istraživanja

- ◆ Ne postoji generalna ocjena valjanosti nekog modela.
 - Ocjena valjanosti modela može se izvršiti jedino s obzirom na njegovu svrhu.
 - Model koji je valjan za jednu svrhu nije nužno valjan ako se upotrijebi u druge svrhe.
- ◆ Nepreciznost prikupljenih podataka
 - nepreciznosti podataka o realnom sustavu
 - historijski podaci predstavljaju samo uzorak koji sam po sebi unosi nepreciznost.
 - simulacija za rezultate ima također uzorke - validacija u ovom slučaju proces usporedbe dva različita uzorka (Iako postoje statističke procedure koje mogu ustanoviti jesu li ta dva dva uzorka slična, to su ipak samo vjerojatnosni, a ne definitivni odgovori).

Problemi vezani uz validaciju



- ◆ Ne postoje podaci o pojavama u realnom sustavu
 - nepostojeći sustavi

- ◆ Nemoguće analizirati apsolutnu valjanost nekog modela - analizira se koliko taj model može biti vjerodostojan.
- ◆ Što se više testova izvede, koji ne mogu dokazati da je model neispravan, to se više može imati povjerenja u rezultate koji se dobiju simulacijom nad tim modelom.
- ◆ Cilj verifikacije i validacije povećanje povjerenja u model i rezultate simulacije nad modelom.

- ◆ Validacija konceptualnog modela
 - Ne postoji formalna metoda za validaciju konceptualnog modela.
 - Najčešće se primjenjuje proučavanje projektne specifikacije, pri čemu se analiziraju ciljevi nekog projekta i pristup koji se koristi pri modeliranju
 - Dokument se u vidu izvještaja prosljeđuje svim relevantnim akterima s dobrim poznavanjem sustava i problematike, i od njih se prikupljaju povratne informacije je li pristup i model prikladan, te se mogu odmah identificirati i ispraviti eventualne pogreške.

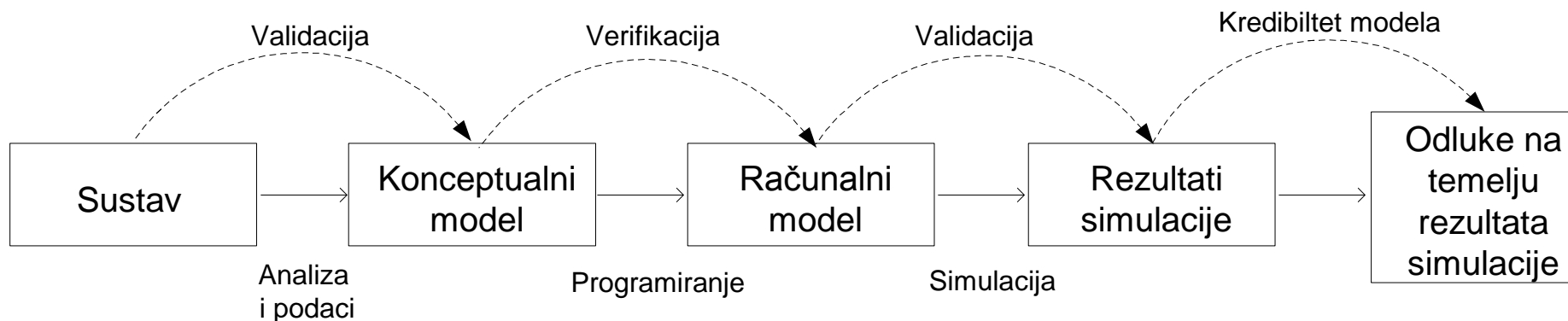
- ◆ Validacija podataka
 - ispitati pouzdanosti i vjerodostojnost svakog izvora podataka (bilo osobe ili baze podataka),
 - provjeriti postoje li neke nekonzistentnosti unutar podataka,
 - ustanoviti koji su podaci u pogrešnom formatu ili neprimjenjivi za analizu, te ustanoviti postupak prikupljanja takvih podataka i njihove transformacije u prikladan format,
 - pohraniti podatke odvojeno od simulacijskog kôda, kako bi se olakšala identifikacija neispravnosti i efikasnije izvršilo ažuriranje novim preciznijim podacima

- ◆ Verifikacija i strukturalna validacija
 - provjera kôda i vizualna provjera modela
 - prolaženje kroz model događaj po događaj;
 - zaustavljanje modela u određenom trenutku, predviđanje što će se sljedeće dogoditi, ponovno pokretanje modela i provjera predviđanja;
 - interaktivno postavljanje uvjeta kako bi se forsiralo izvođenje određenih problematičnih događaja;
 - izoliranje pojedinih dijelova modela kako bi se simulacija izvršavala brže, reducirajući na taj način vrijeme potrebno za strogu verifikaciju i validaciju;
 - praćenje kretanja pojedinog entiteta kroz modela tijekom cijele simulacije
 - provjera izlaznih rezultata simulacije

- ◆ Validacija cjelokupnog modela
 - usporedba s realnim sustavom
 - prikupljaju se historijski podaci o ponašanju sustava
 - potrebno je usporediti:
 - prosječne vrijednosti
 - raspršenje podataka
 - distribucije podataka (može i vizualno)
 - odnose između ulaznih i izlaznih veličina za realni sustav i za model.
 - statistički testovi koji omogućuju takve usporedbe

- Usporedba s drugim modelima
 - usporedba s matematičkim modelom (u svrhu provjere modela, matematički model može dati grubu aproksimaciju rezultata koji se očekuju; matematičke jednadžbe i formule, statička analiza i primjena teorije čekanja i posluživanja.);
 - usporedba s determinističkim modelom (iz modela se uklanjaju svi elementi koji uzrokuju slučajnost, i na taj se način dobije deterministički model čiji se rezultati mogu predvidjeti primjenom matematičkog modela);
 - usporedba s drugim simulacijskim modelima (ako je za realni sustav ili njemu sličan sustav već razvijen model nad kojim je moguće izvršiti simulaciju, ali u neke druge svrhe, i ako je taj model već verificiran i ocijenjen kao valjan, tada se modeli mogu usporediti s obzirom na ulazne podatke, izlazne rezultate i njihove međuodnose).

Metode verifikacije i validacije



Metoda za modeliranje i određivanje performansi procesa održavanja telekomunikacijske programske podrške

Ciljevi metode i područja primjene



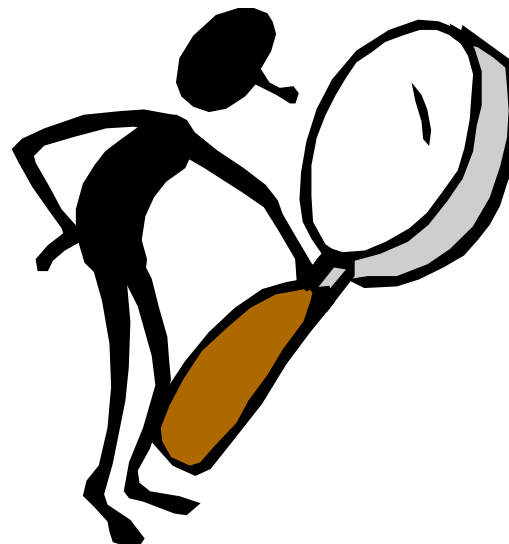
- ◆ Predikcija performansi postojećeg procesa održavanja.
- ◆ Osiguravanje podataka o nedostacima procesa održavanja vezanim uz vrijeme potrebno da se određeni softverski problem riješi.
- ◆ Mogućnost određivanja koju metriku treba dodatno uključiti u automatizirano mjerenje procesa.
- ◆ Podrška donošenju odluka na koji način efikasnije raspodijeliti ili modificirati postojeće resurse procesa održavanja.

Potencijalni procesi na koje se metode može primijeniti su:

- ◆ procesi koji imaju kritične vremenske performanse s obzirom na isporuku svojeg produkta,
- ◆ programski procesi kojima se želi promijeniti dizajn,
- ◆ procesi u kojima se planira primjena novih pomagala, promjena broja sudionika i resursa.

- ◆ Metoda se može koristiti unutar odabranog generičkog modela kao dio organizacijskih napora kako bi dosegla viša razina zrelosti procesa

- ◆ Model mreže repova
- ◆ Prikupljanje kvantitativnih podataka
- ◆ Prikupljanje kvalitativnih podataka
- ◆ Simulacija
- ◆ Statistička analiza rezultata



Programski proizvodi i kontekst metode

- ◆ Zahtjevi za modifikacijom koji se odnose na distribuirani telekomunikacijski programski sustav.
- ◆ Organizacijske jedinice za održavanje PP unutar telekomunikacijske kompanije.



Pretpostavke korištenja metode



- ◆ Proces održavanja unutar softverske organizacije odvija se u skladu s procesnom definicijom i može se formalno opisati.
- ◆ Postoje historijski kvantitativni podaci o vremenima odvijanja aktivnosti u pojedinim elementima procesa, koji se automatski mjere tijekom izvršavanja realnog procesa.
- ◆ Postoje baze podataka svih korisničkih zahtjeva za modifikacijom i vremenskog slijeda njihovog životnog ciklusa.

Pretpostavke korištenja metode (2)



- ◆ Postoje podaci o poduzetim akcijama na temelju pojedinog korisničkog zahtjeva.
- ◆ Postoje jasno definirani i kvantitativno izraženi poslovni ciljevi za uspješno izvođenje i poboljšanje procesa.
- ◆ Pri implementaciji metode neophodna je suradnja s ekspertima i ostalim sudionicima dotičnog procesa

Osnovne faze metode

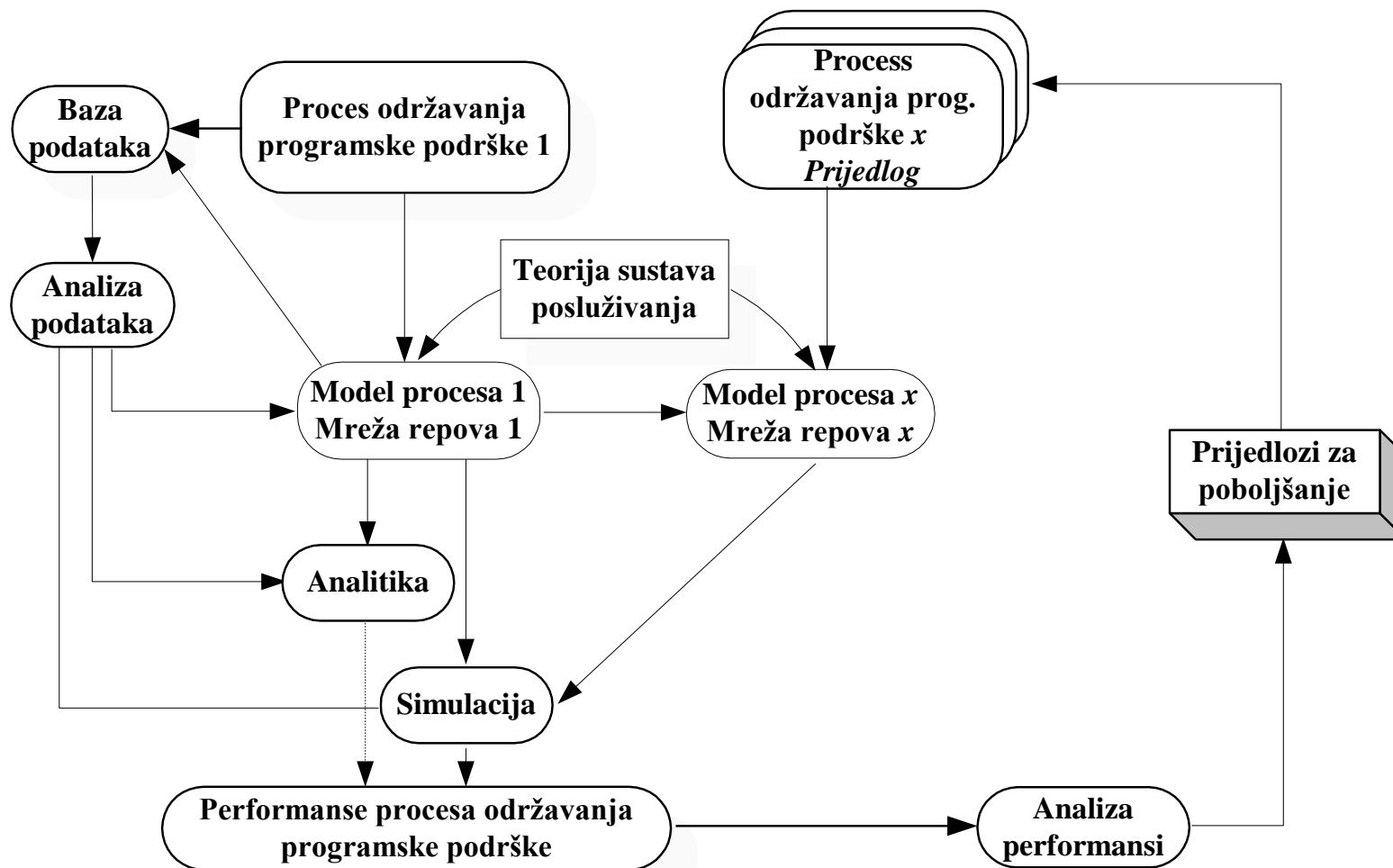
- ◆ Planiranje i priprema
- ◆ Modeliranje procesa
- ◆ Analiza procesa
- ◆ Izvođenje procjene
- ◆ Izvještaj o rezultatima



- ◆ Definiranje projekta
- ◆ Razvoj modela procesa održavanja na temelju mreže repova
- ◆ Prikupljanje i analiza podataka
- ◆ Simulacija nad modelom mreže repova
- ◆ Analiza performansi dobivenih simulacijom
- ◆ Analiza i procjena novog dizajna procesa

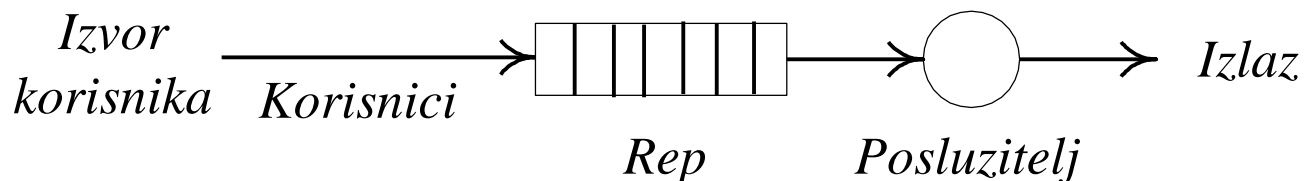
- ◆ Svaka slijedeća aktivnost metode osniva se na prethodno izvedenim aktivnostima.
- ◆ Neke aktivnosti se iterativno ponavljaju, na primjer, prvo se razvije model procesa, zatim se prikupljaju i analiziraju podaci.
- ◆ Na osnovu prikupljenih podataka povećava se razina detalja u modelu.
- ◆ Uočava se koji podaci još nedostaju - ponovno prikupljanje i analiza podataka - dok se ne dostigne dostatna razina detalja modela za rješavanje osnovnih ciljeva istraživanja, i dovoljna opskrbljenost podacima kako bi se mogla izvesti analiza primjenom simulacije.

Aktivnosti metode (2)

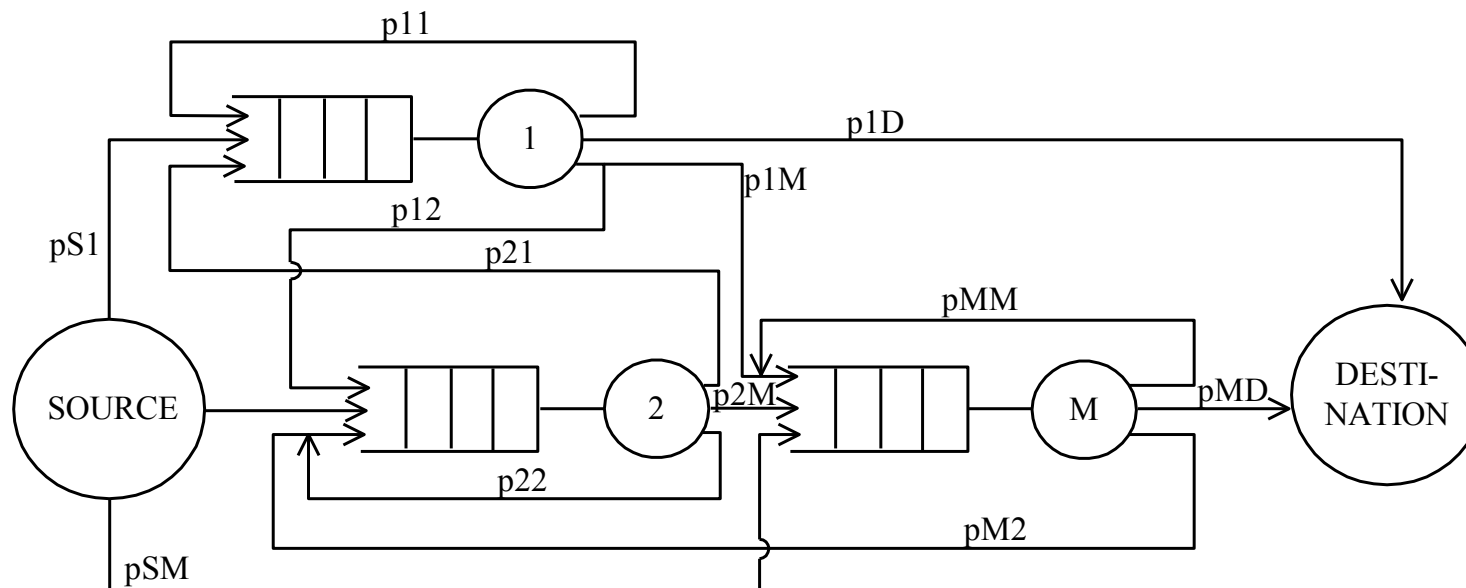


1. Broj zahtjeva za modifikacijom koji dolaze u proces nije ograničen (u modelu to znači da je izvorište generira beskonačan broj korisnika).
2. Dolasci korisnika u proces nezavisni su događaji.
3. Mreža repova je otvorena.
4. Uvjetni prekid posluživanja pri nailasku korisnika višeg prioriteta u rep.

- ◆ *Teorija čekanja i posluživanja* je grana vjerojatnosne teorije
- ◆ *Rep* je fenomen koji se pojavljuje uvijek kada je količina zahtjeva za nekom uslugom veća od kapaciteta pružaoca usluge.
- ◆ Općenito, teorija čekanja i posluživanja obuhvaća situacije kada korisnik ulazi u sustav gdje neko vrijeme provodi čekajući u repu dok poslužitelj obrađuje nekog drugog korisnika.



Otvorena mreža repova



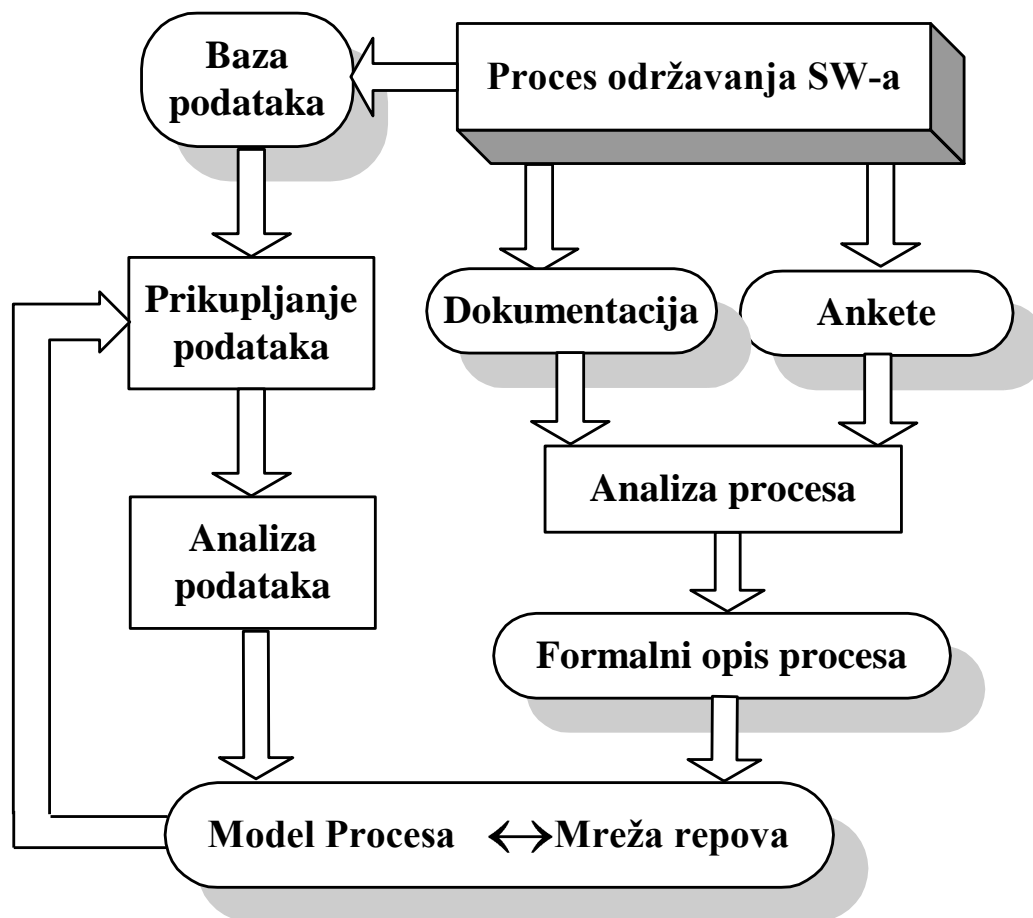
mreža s izvorom beskonačne populacije korisnika, i odredištem ili ponorom koji apsorbira sve korisnike koji napuštaju sustav.

Modeliranje procesa održavanja u formi mreže repova



- ◆ **Korisnici mreže repova** - zahtjevi za modifikacijom upućeni u organizacijsku jedinicu za održavanje.
- ◆ **Izvorišta korisnika** (na primjer, tržište, drugi procesi, sudionici zadanog procesa održavanja).
- ◆ **Dolazni proces korisnika** opisuje se s distribucijom i intenzitetom nailazaka zahtjeva u proces.
- ◆ Identificiraju se **odredišta korisnika** - predstavljaju moguće rezultate procesa (na primjer, isporuka rješenja korisničkog zahtjeva u obliku korekcije, odgovora ili neke druge akcije).
- ◆ **Aktivnosti** koje se obavljaju nad korisničkim zahtjevom za modifikaciju - čvorovi mreže.
- ◆ Svakom čvoru mreže pridružuje se **izvršitelj** pripadne aktivnosti u realnom procesu koji se imenuje poslužiteljem.
- ◆ Svakom čvoru mreže pridružuje se **rep** u kojem zahtjevi za modifikacijom čekaju da budu usluženi od strane poslužitelja.

Prikupljanje i analiza podataka



Prikupljanje kvantitativnih podataka



- ◆ **Vremena nailazaka zahtjeva** kod pojedinog poslužitelja;
- ◆ **Vremena posluživanja** za svaku aktivnost modeliranu čvorom u mreži repova;
- ◆ **Prioritet** pridružen pojedinom zahtjevu za modifikacijom;
- ◆ **Tipovi odgovora** na svaki pojedini zahtjev (koji opisuju kako je zahtjev riješen: korekcijom, tehničkim odgovorom, odbijen itd.);
- ◆ **Stanja i vremena promjene stanja** zahtjeva za modifikacijom

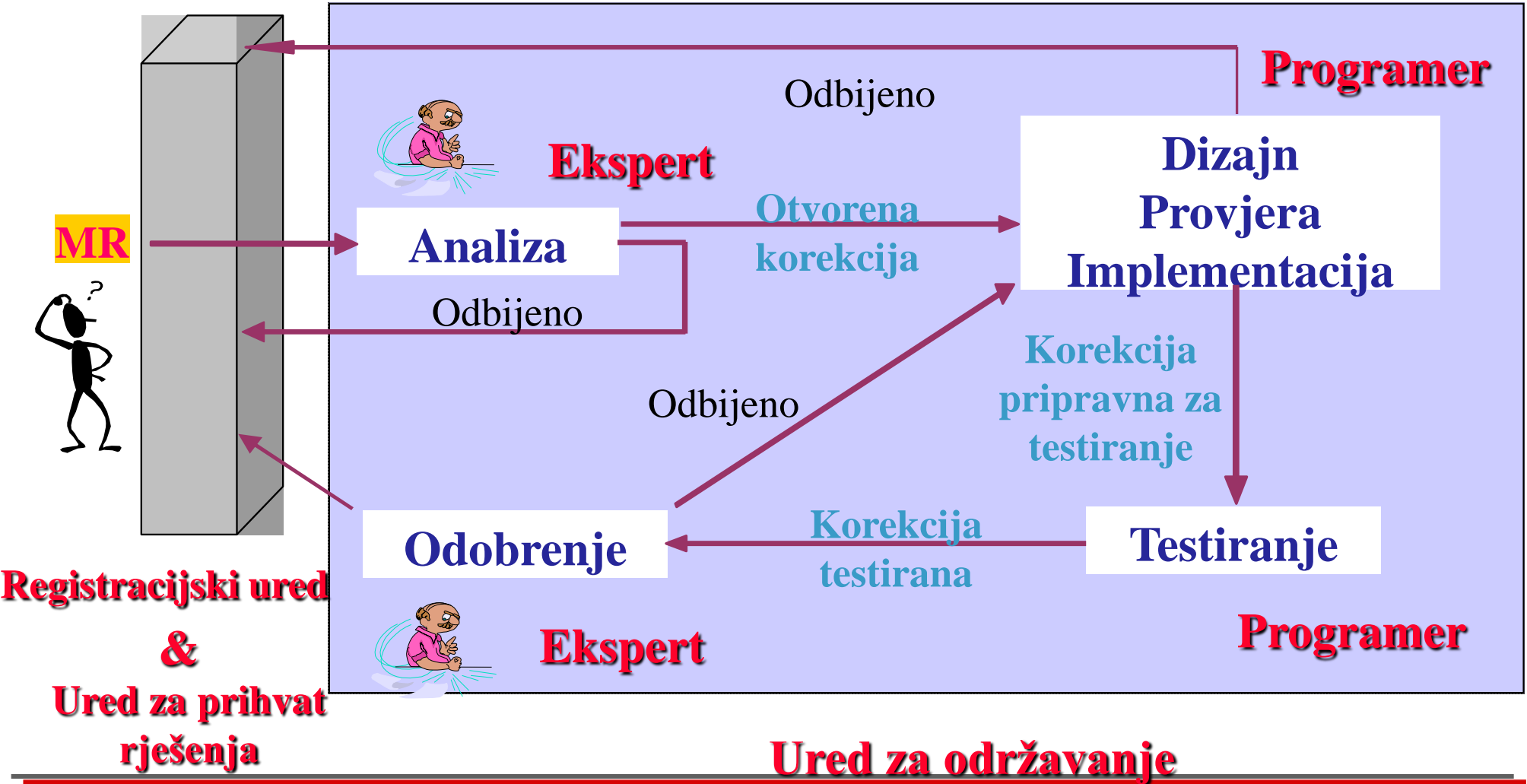
	+2.688
0	+5.000
1	+1.500
0	+1.125
0	+1.062

- ◆ Kvalitativni podaci se prikupljaju:
 - ako nedostaju neki kvantitativni parametri neophodni za izgradnju modela na određenom nivou detalja (odnosno ako se pri izvođenju procesa ta vrijednost automatski ne bilježi),
 - kako bi se bolje objasnili statistički rezultati dobiveni analizom kvantitativnih podataka
 - prikupljaju se putem anketa (usmenih ili pismenih) i opažanjima sudionika procesa.

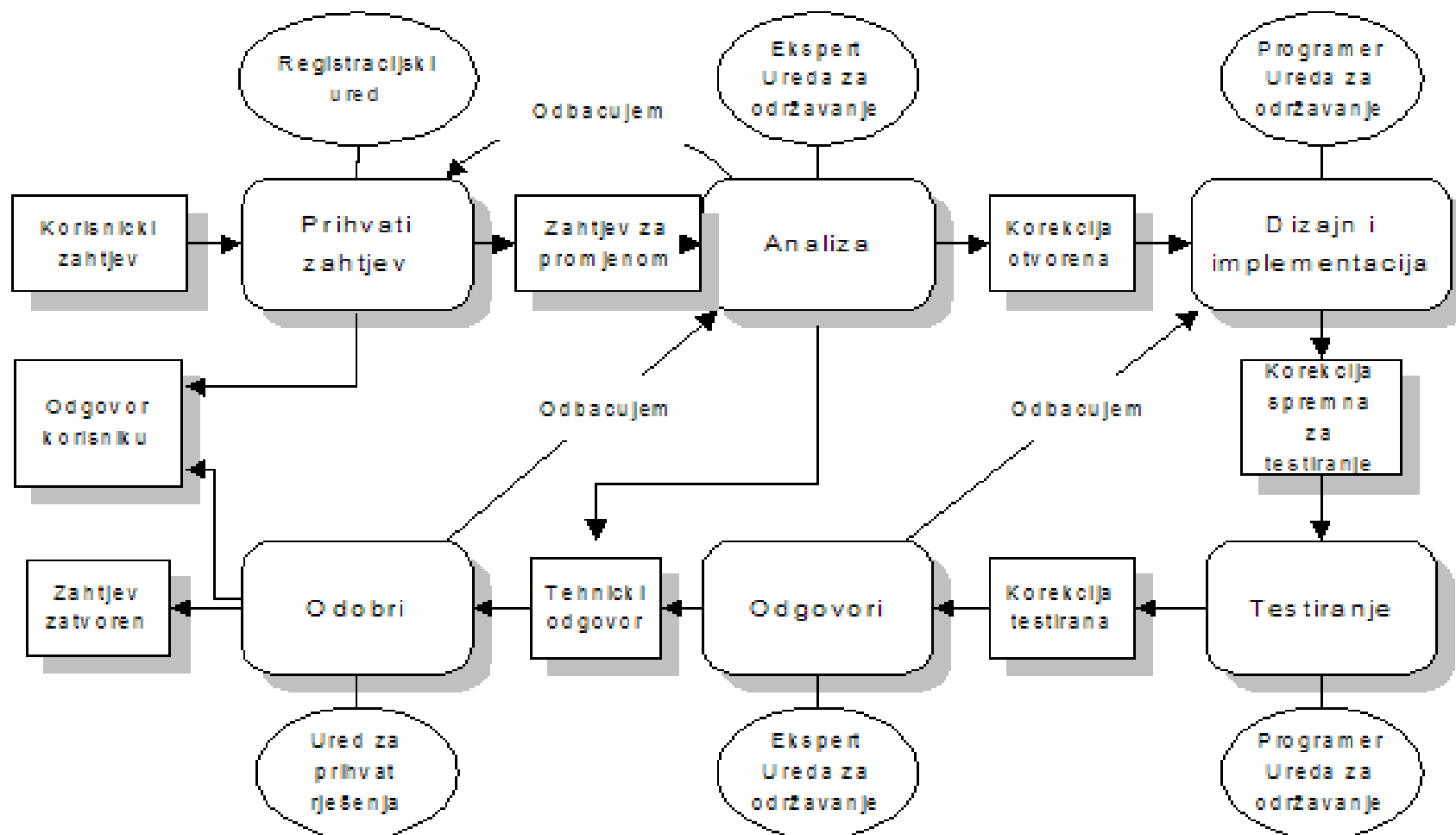


- ◆ Osiguravanje sigurnosti kvalitativnih podataka postiže se primjenom sljedećih tehnika:
 - Kvalitativni podaci se prikupljaju iz najmanje dva nezavisna izvora (dvije neovisne osobe, ili osoba i dokument).
 - Kvalitativni podaci se ne prikupljaju odjednom, nego u najmanje dva navrata.
 - Opažanja trebaju biti potvrđena od strane barem jednog izvora podataka

Studijski primjer procesa održavanja programske podrške



Analiza studijskog primjera



- ◆ Postoje **tri vrste prioriteta** koji se dodjeljuju zahtjevima za modifikacijom:
 - **A** - *vrlo veliki problem*; centrala ne može biti u funkciji ako se greška ne ispravi, postoji vrlo veliki pritisak na održavanje da se pronađe rješenje što prije. Definirani vremenski okvir za rješenje je 14 kalendarskih dana.
 - **B** - *problem srednje veličine*; dio sustava je ispao, ali sustav u cjelini ipak funkcionira. Definirani vremenski okvir za rješenje je 21 kalendarski dan.
 - **C** - *manji problem*; ne uzrokuje nikakvu trenutnu štetu, ovdje spadaju zahtjevi za poboljšanje dokumentacije, prijedlozi poboljšanja softverskog produkta itd. Definirani vremenski okvir za rješenje je 28 kalendarskih dana

- ◆ **MR-ovi koji ne zahtijevaju korekciju** su sljedećeg tipa, i na njih se odgovara **tehničkim odgovorom**:
 - ne postoji problem, lažno prijavljivanje od strane korisnika,
 - ne postoji problem, krivo razumijevanje funkcionalnosti,
 - zahtjev za novom funkcionalnošću,
 - pogreške u dokumentaciji,
 - pogrešno postavljanje osnovnih parametara sustava, nema pogreške u kôdu,
 - produkt se ne održava jer postoji njegova nova verzija,
 - razvoj novog softverskog produkta je u tijeku i on će sadržavati ispravak prijavljene pogreške.

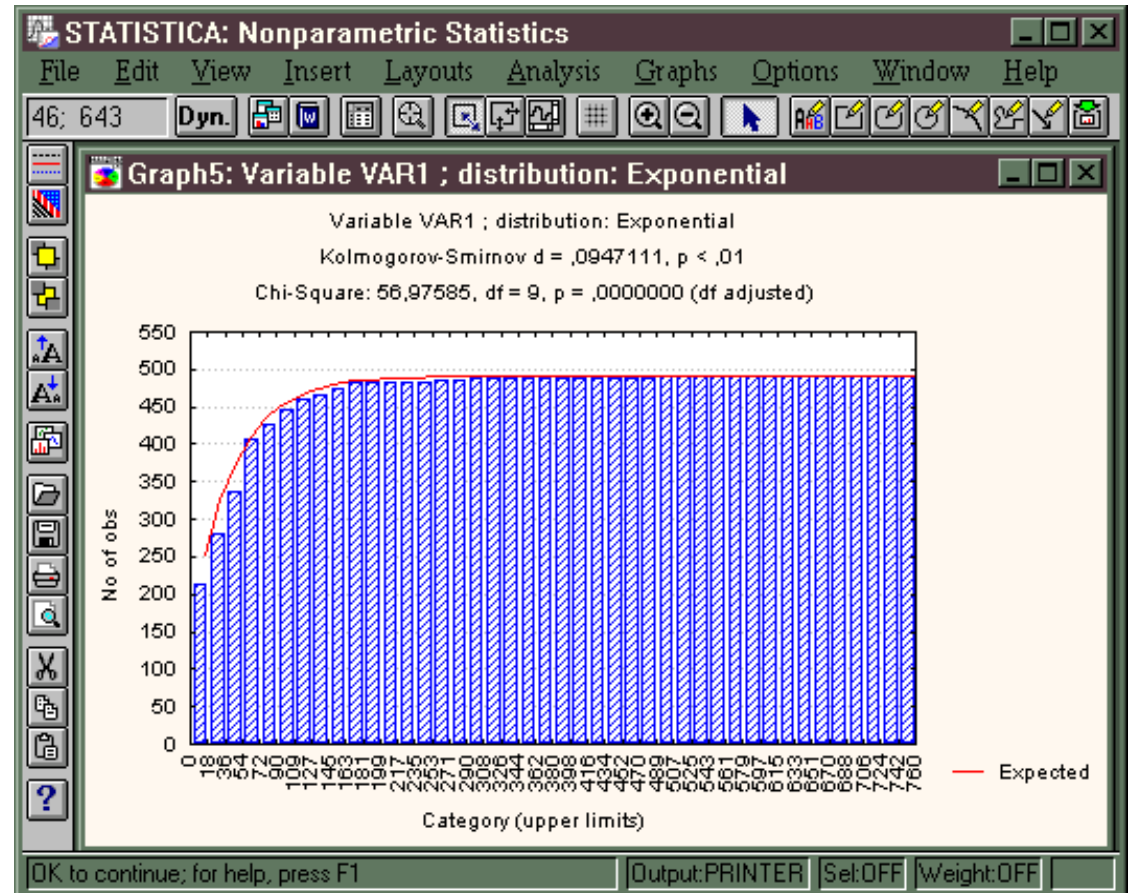
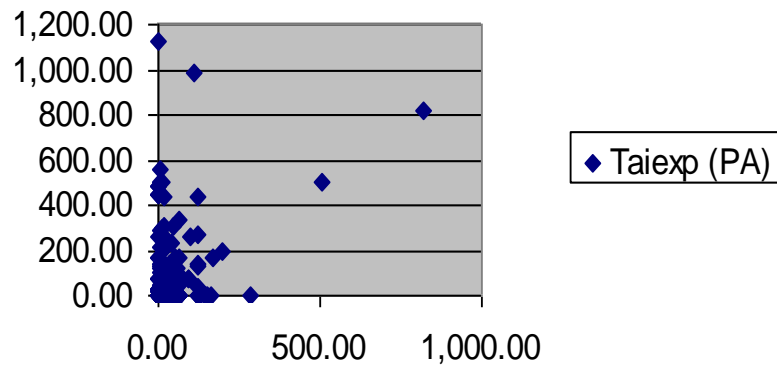
- ◆ Definirani su **sudionici procesa održavanja** i njihove uloge:
 - *Podnositelj Zahtjeva za održavanjem* - korisnik telekomunikacijskog softverskog sustava
 - *Vlasnik Zahtjeva za održavanjem* - *Ured za održavanje*, odgovara za grupu softverskih produkata, kontrolira upravljanje problemom,
 - *Ekspert* - najviši autoritet u grupi odgovornoj za održavanje određenog softverski produkt.
 - *Programer* - izvodi akcije korektivnog održavanja nad softverskim produktom u skladu sa *Zahtjevom za održavanjem*.

- ◆ Definiranje ciljeva, pitanja i opsega projekta
 - određivanje prosječnog vremena zadržavanja zahtjeva za modifikacijom u Uredu za održavanje,
 - pronalaženje odnosa vremena zadržavanja zahtjeva za modifikacijom za pojedine faze procesa održavanja,
 - definiranje točaka procesa u kojima se zahtjev najdulje zadržava,
 - definiranje točaka procesa kojima zahtjev najdulje čeka na obradu od strane nekog sudionika procesa ili resursa

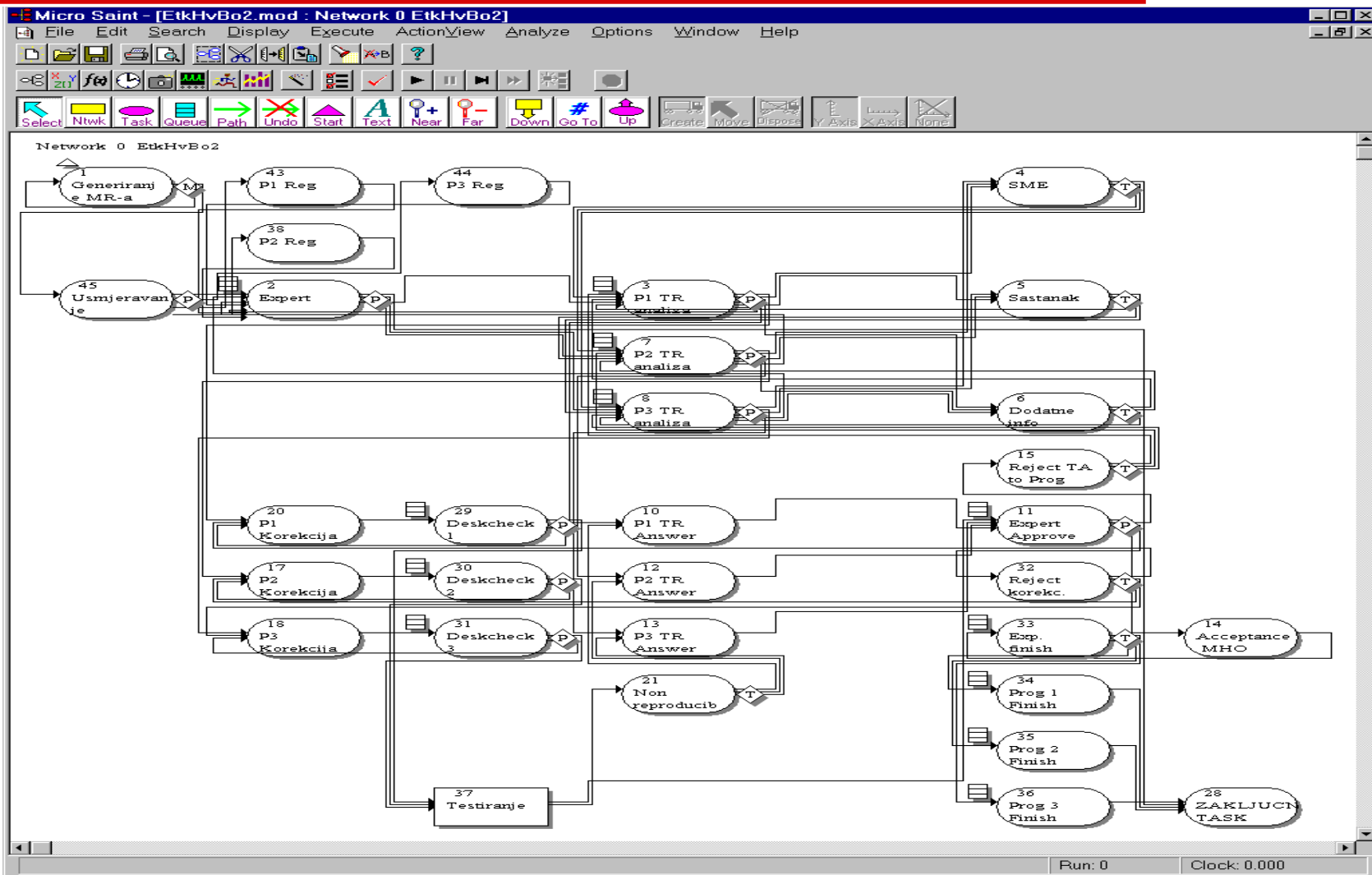
Analiza podataka



Ekspert-medjudolazna vremena



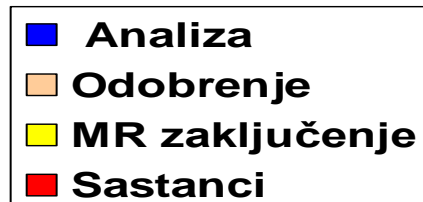
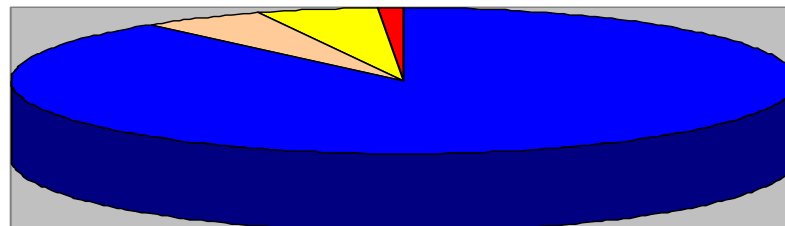
Konačni model mreže repova



- ◆ Korišteni su sljedeći parametri:
 - prosječno vrijeme zadržavanja MR-a u procesu (neovisno o prioritetu) - T_q
 - prosječno vrijeme zadržavanja MR-a u procesu za prioritete A, B, C - T_{qA} , T_{qB} , T_{qC}
 - ukupan broj generiranih zahtjeva tijekom zadanog simulacijskog vremena
 - broj generiranih zahtjeva tijekom zadanog simulacijskog vremena po prioritetima

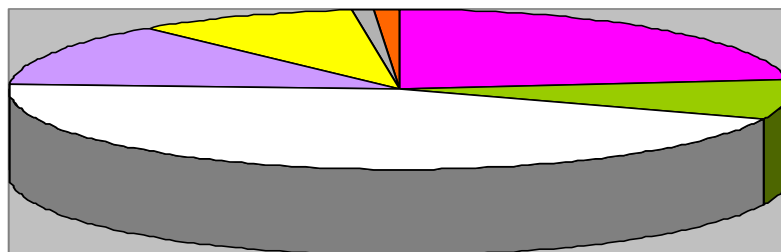
Iskoristivost poslužitelja

Expert



Ukupna iskoristivost = 44 %

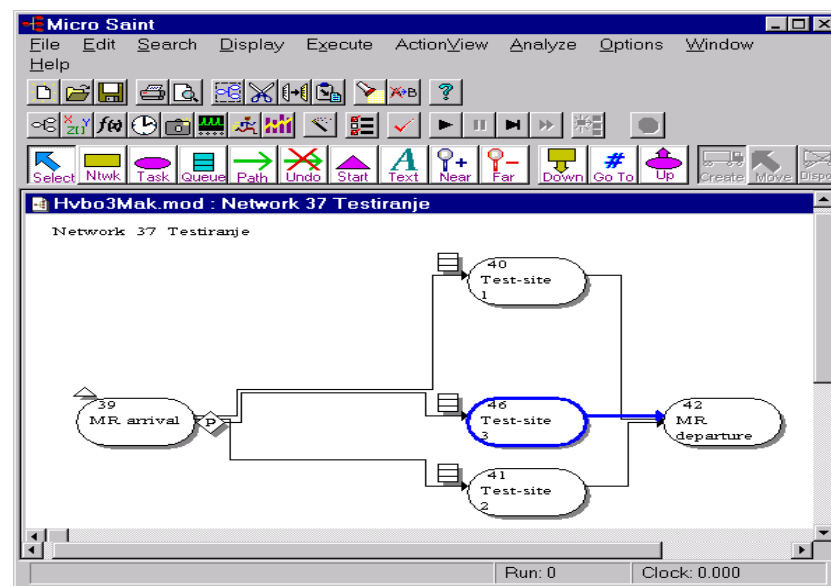
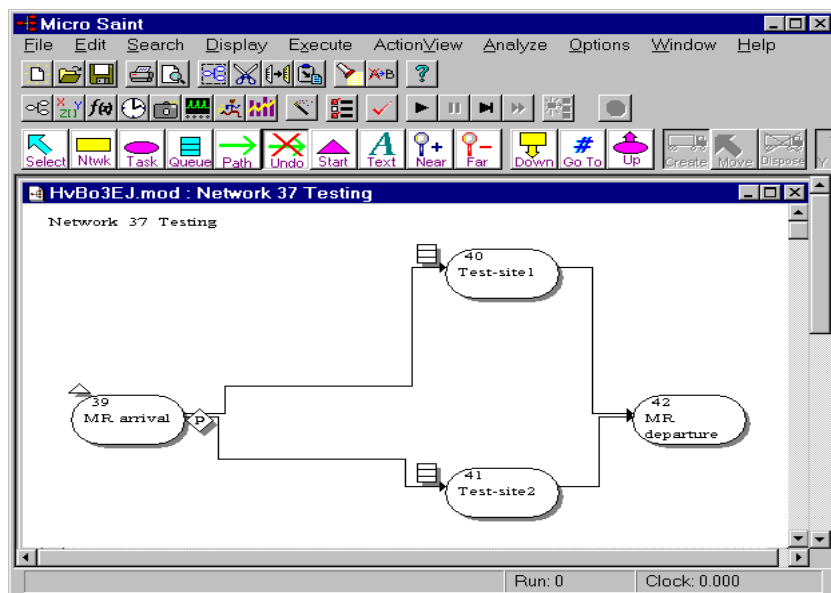
Programer



Ukupna iskoristivost = 75 %

Alternativni dizajn procesa 1

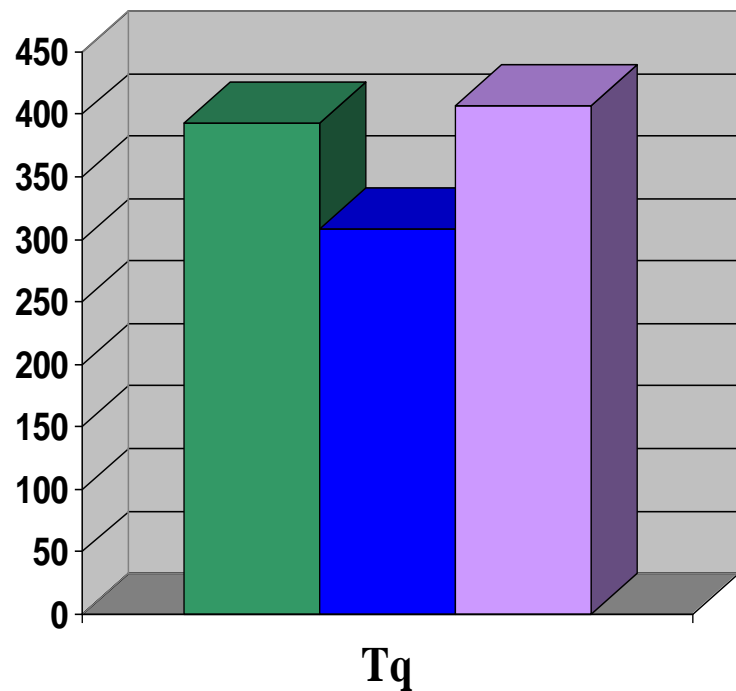
Povećanje broja test maketa



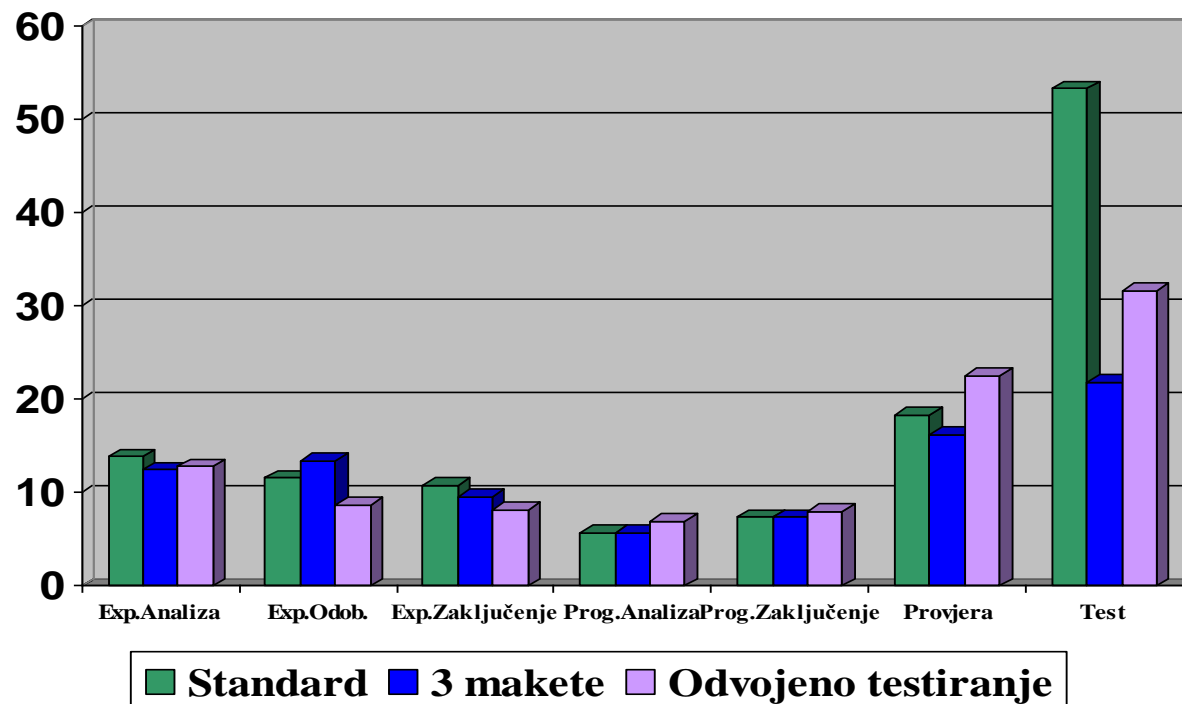
Usporedba performansi novih prijedloga procesa



Vremena zadržavanja



Vremena čekanja u različitim fazama procesa



1. Zhang, He., B. Kitchenham ; D. Pfahl. Software Process Simulation over the Past Decade: Trends Discovery from A Systematic Review. *Proceedings of the ESEM 08*, Kaiserslautern, Germany, 2008.
2. Kellner, M.I., R. J. Madachy, D. M. Raffo. Software process simulation modeling: Why? What? How? *Journal of Systems and Software*, Volume 46, Issues 2-3, 1999.
3. Galinac, T. *Povećanje učinkovitosti razvoja programskog proizvoda pomoću upravljanja procesom rane verifikacije*. Doktorska disertacija, FER, Zagreb, 2009. Mentori: Car, Ž., D. Huljenić.
4. Car, Ž. Modeliranje procesa održavanja telekomunikacijske programske podrške. Doktorska disertacija, FER, Zagreb, 2001. Mentor: Mikac, B.
5. SWEBOK - *Software Engineering Body of Knowledge* , IEEE Computer Society, 2004, dokument dostupan na Internetu.