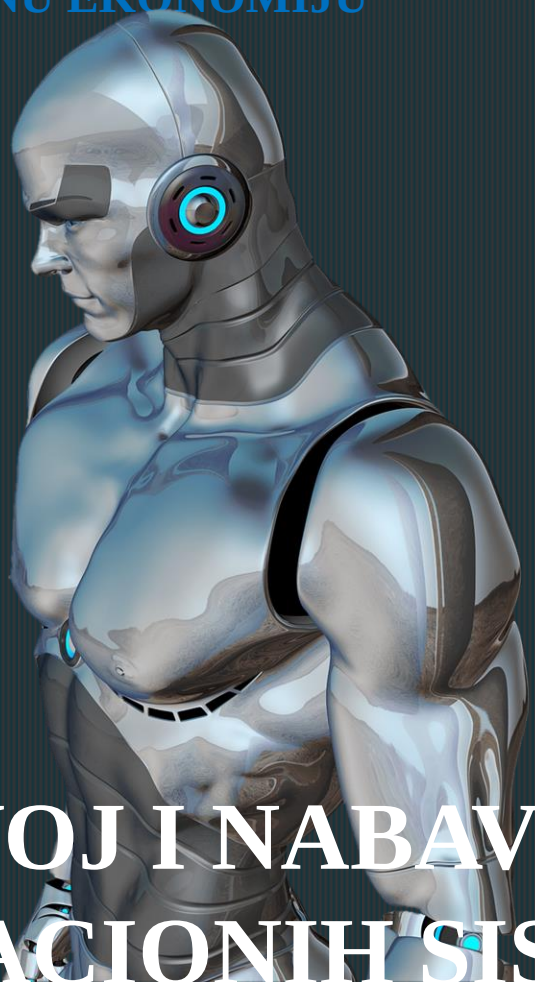


RAZVOJ I NABAVKA INFORMACIONIH SISTEMA

Prof. dr Saša Ivanov
sasa.ivanov@fmz.edu.rs

FAZE ŽIVOTNNOG CIKLUSA IS

Jedan od osnovnih zadataka razvoja i upravljanja projektom razvoja informacionog sistema organizacije, ogleda se u izabiranju kompleksa pravila i ciljeva koji se u razvoju informacionog sistema organizacije žele postići, kao i odgovarajućeg kompleksa akcija čijim se preduzimanjem sa veoma velikom verovatnoćom postižu postavljeni ciljevi.

Za svrhe izgradnje koncepcije razvoja informacionog sistema neke organizacije neophodno je proučiti i shvatiti tu organizaciju celovito, sa stanovišta njene forme, suštine, funkcija, structure, razvoja i vrednosti.

Koristeći se analogijom sa živim organizmima, smatra se da upravljački (menadžerski) informacioni sistemi nastaju, rastu, sazrevaju i nestaju, pa se taj process naziva ″**životni ciklus sistema**″, koji u pojedinim slučajevima može trajati samo nekoliko meseci, a u drugim nekoliko godina.

FAZE ŽIVOTNNOG CIKLUSA IS

Kada se donese strategijska odluka o projektovanju informacionog sistema, kreće se sa njegovom projektovanjem. Tradicionalni metod koji primenjuju organizacije u razvoju informacionog sistema kompanije naziva se **životni ciklus u razvoju informacionog sistema**. Faze koje ćemo koristiti da bi objasnili životni ciklus sistema, se u manjoj ili većoj meri razlikuju od autora do autora, ali generalno možemo prihvatiti činjenicu da je primetno da teoriju i praksu razvoja informacionih sistema, kao i sve oblike ljudske djelatnosti, karakteriše kontinuelni proces traženja rješenja kojima se, uz maksimalnu uštedu vremena i sredstava (odnosno, povećanje produktivnosti) na kvalitetan način prevazilazi definisani problem. Ovaj proces je istovremeno usmjeren ka apsolutnoj formalizaciji i automatizaciji razvoja informacionih sistema. Stoga se u našem slučaju ograničavamo na sedam sekvencijalnih procesa (faza) tokom kojih se razvija informacioni sistem.

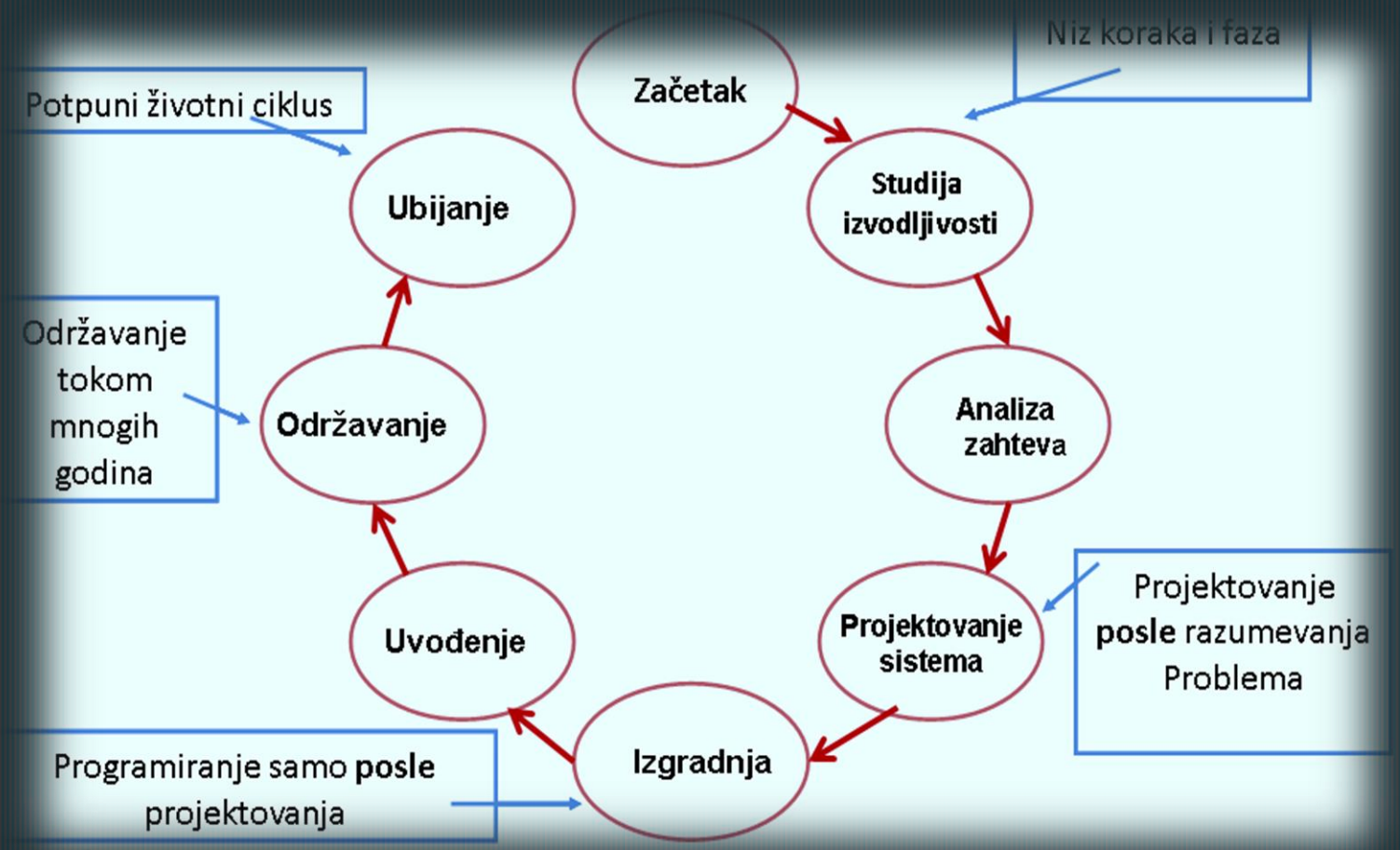
FAZE RAZVOJA IS

Faze životnog ciklusa u razvoju informacionog sistema su:

1. *Inicijalizacija (eng. initiating)*
2. *Studija izvodljivosti (eng. feasibility study)*
3. *Analiza zahteva (eng. requirement analysis)*
4. *Projektovanje sistema (eng. system design)*
5. *Izgradnja sistema: programiranje i testiranje (eng. system build)*
6. *Implementacija (eng. implementation)*
7. *Održavanje (eng. maintenance)*

Svaki proces se sastoji iz tačno definisanih zadataka. Veliki projekti obično zahtevaju izvršavanje svih zadataka, dok manji razvojni projekti obično zahtevaju izvršavanje samo jednog dela zadataka.

FAZE RAZVOJA IS



FAZE RAZVOJA IS

Na razvoju informacionog sistema angažuje se **razvojni tim**. U njega su obično uključeni korisnici, sistem analitičari, programeri i tehnički eksperti. **Korisnici** su zaposleni iz svih funkcionalnih oblasti i nivoa organizacije, koji će koristiti sistem, na direktan ili indirektan način. **Sistem analitičari** su informatički profesionalci, koji su se specijalizovali u analiziranju i projektovanju informacionih sistema. **Programeri** su informatički profesionalci, koji pišu nove ili modifikuju postojeće računarske programe da bi zadovoljili zahteve korisnika. **Tehnički eksperti** su specijalisti iz određene računarske oblasti (na primer baze podataka ili telekomunikacija), koji svojim znanjem učestvuju u razvoju informacionog sistema.

FAZA I – INICIJALIZACIJA SISTEMA

To je početna faza razvoja informacionog sistema (IS), koja je ujedno i ″okidač″ (eng. trigger) koji inicira jedan IS projekat. Profesionalci se slažu u tvrdnji da što se više vremena uloži u sagledavanje poslovnog problema koji treba da se reši izgradnjom informacionog sistema, u razumevanje tehničkih opcija sistema, i u razmatranju problema koji mogu da nastanu tokom razvoja sistema, veće su šanse za uspešno rešavanje problema. Dakle, za početak, *poslovni problem* koji treba da se reši izgradnjom informacionog sistema, *treba detaljno sagledati.*

Razlozi za inicijalizaciju projekta (tzv. *The five ″C″ of Senn (1995)*):

1. Smanjenje troškova (eng. **C**ost reduction): izvršavanje starih poslova brže, efektivnije
2. Mogućnost (eng. **C**apability) izvršavanja novih poslova
3. Komunikacija (eng. **C**ommunication): efektivnije prosleđivanje informacija
4. Korisnički servis (eng. **C**ustimer service): bolji, brži, različiti servisi kupaca
5. Kontrola (eng. **C**ontrol): efikasnije, brže, razumnije korišćenje informacija
6. Konkurentna prednost (eng. **C**ompetitive advantage): raditi sve prethodno navedeno brže i bolje od konkurencije (*šesti C su dodali Bocij i Chaffey*)

Cilj ove faze je da se utvrdi da li je projekat izvodljiv i da se odluči da li dalje da se ulaže.

FAZA II – STUDIJE IZVODLJIVOSTI

Sledeći zadatak u fazi istraživanja sistema je izgradnja **studije izvodljivosti**. Njome se određuje verovatnoća uspeha predloženog projekta, i obezbeđuje gruba procena projektne *tehničke*, *ekonomske*, *organizacione* i *operativne* izvodljivosti. Studija izvodljivosti je krucijano važna u procesu razvoja sistema, zbog toga što, ukoliko se korektno izvede, može da spreči ogromne greške po budući sistem. Bez pravilno izvedene studije izvodljivosti, moguće je napraviti sistem koji ne radi, ili ne radi efikasno, te korisnici ne mogu, ili ne žele da ga koriste.

U ovoj fazi možemo razlikovati četiri podfaze:

1. Tehnička izvodljivost
2. Ekonomska izvodljivost
3. Organizaciona izvodljivost
4. Operativna izvodljivost

FAZA II – STUDIJE IZVODLJIVOSTI

1. **Tehnička izvodljivost.** Tehnička izvodljivost određuje na koji način se mogu uskladiti hardverske, softverske i komunikacione komponente u rešavanju problema. Tehnička izvodljivost takođe određuje i *da li će postojeća tehnologija koju dotična organizacija poseduje biti korišćena u novom projektu ili je treba kreirati.*
2. **Ekonomska izvodljivost**. Ekonomska izvodljivost određuje da li je planirani projekat prihvatljiv finansijski rizik i da li organizacija može da izdrži troškove i vreme potrebno za kompletiranje projekta. Ekonomska izvodljivost postavlja tri ključna pitanja: *Šta su koristi? Šta su troškovi prihvatanja projekta? Da li korist prevazilazi cenu projekta?*
3. **Organizaciona izvodljivost**. *Organizaciona izvodljivost razmatra mogućnost organizacije da prihvati predloženi projekat i ako ovo dovodi do promena, da li se one mogu sprovesti i kakve su posledice za organizaciju?* Ponekad, na primer, organizacija ne može da prihvati finansijski prihvatljiv projekat zbog zakonskih ili drugih ograničenja. U okviru organizacione izvodljivosti, razmatra se politika organizacije, uključujući poslovne odnose i mogućnost unutrašnjih resursa.
4. **Operativna izvodljivost**. *Operativna izvodljivost se odnosi na uticaj projekta na ljude tj. procenjuje ljudski faktor predloženog sistema:*
   - *otpor na promene*
   - *organizaciona politika.*

FAZA II – STUDIJE IZVODLJIVOSTI

Svaki razvojni projekat novog sistema unosi promene u organizaciji, a ljudi se genaralno plaše promena. Otpor zaposlenih može poprimiti oblik omalovažavanja novog sistema, pa čak i sabotiranja (na primer, namernog nekorektnog unošenja podataka).Otpor se obično javlja kada zaposleni svoj posao rade jednostavnije koristeći stare metode. Mnogo pozitivnija briga operativne izvodljivosti je procena potreba za obukom zaposlenih, kako bi ovladali korišćenjem novog informacionog sistema.

FAZA II – STUDIJE IZVODLJIVOSTI

FAZA II – STUDIJE IZVODLJIVOSTI

Jedan od **razloga** za uvođenje ove faze je sigurno razmatranje pitanja:

1. **troškova i dobiti (koristi)** uvođenja novog sistema (eng. **cost-benefit analysis**):

Koji su *troškovi* uvođenja novog sistema:

- očigledni troškovi, odgovaraju na pitanje: Koliko će koštati informacioni sistem? Tu ubrajamo troškove kupovine hardvera i softvera i troškove obuke (trenera i kurseva).
- troškovi samog projekta informacionog sistema: troškovi zaposlenih na razvoju sistema (eng. system development staff costs); troškovi instalacije (instalation costs): kablovi, fizičko pomeranje opreme, nova oprema; troškovi seobe (migration costs): prenošenje informacija sa trenutnog na nov sistem i operativni troškovi (operating costs): troškovi održavanja hardvera, troškovi vezani za zaposlene na održavanju...

Dobiti (koristi) od uvođenja novog sistema

S obzirom da informativni sistem uključuje kompjuterizaciju, osnovna dobit se ogleda u:

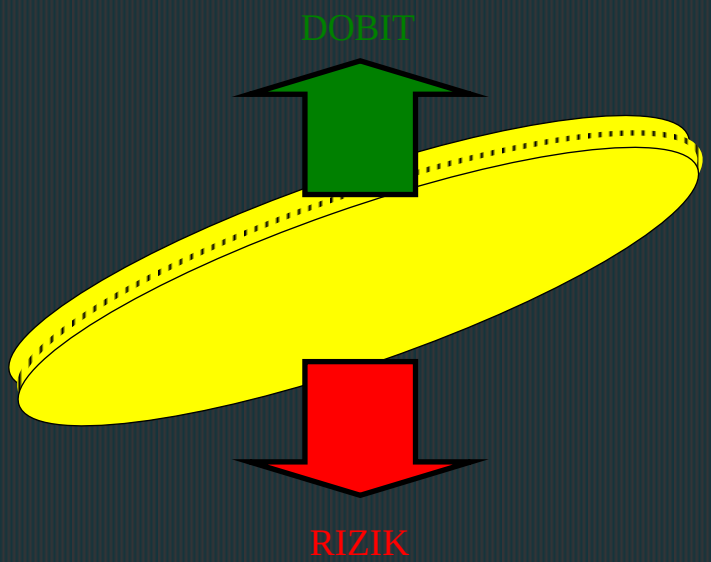
- eliminisanju ljudskog faktora: ″rezanje″ troškova, kontrola (manje zaposlenih, bolje upravljačke odluke...)
- u mogućnosti da nove stvari učini mogućim: sposobnosti, komunikacija, korisnički servis

Stalna dobit zahteva promenu u poslovanju i bez ove promene dobit se neće ostvariti.

FAZA II – STUDIJE IZVODLJIVOSTI

Drugi razlog za uvođenje ove faze je razmatranje pitanja:
2.rizika (eng. risk analysis).

Dobit je uspešan i važan ishod realizacije nekog projekta IS, dok je rizik mogućnost da se iz određenih razloga dobit ne ostvari.



FAZA II – STUDIJE IZVODLJIVOSTI

Faktori rizika:

1. Veličina projekta
2. Kompleksnost projekta
3. Kadrovi, osposobljenost korisnika sistema
4. Kontrola projekta
5. Novine
6. Stabilnost zahteva

Cilj ove faze je da se nakon analize svih nabrojanih izvodljivosti, *donosi se odluka o pokretanju ili napuštanju projekta izgradnje novog informacionog sistema*. Ukoliko se donese odluka o napuštanju projekta, projekt se stavlja na čekanje dok se ne uspostave pogodniji uslovi za njegovu realizaciju, ili se projekat jednostavno odbacuje. Ukoliko se donese odluka o pokretanju projekta, tada započinje sledeća faza, sistemska analiza.

FAZA III – ANALIZA ZAHTEVA SISTEMA (SISTEMSKA ANALIZA)

Kada razvojni projekat dobije neophodnu saglasnost od strane svih učesnika, započinje faza sistemske analize. Ova faza definiše poslovni problem koji organizacija planira da reši informacionim sistemom, identifikuje njegove uzročnike, određuje rešenja i određuje informacione zahteve koje rešenje mora da zadovolji. Shodno rečenom, ova faza se odnosi na prikupljanje poslovnih zahteva i predstavlja jedan proces istraživanja: intervjuisanjem, posmatranjem, modeliranjem, brainstormingom i pregovaranjem, formalizovanjem.

Pitanja koja se u ovoj fazi postavljaju:

1. Kakve osobine i funkcije treba da ima sistem?
2. Šta su okviri predloženog sistema? Koji su delovi organizacije pogođeni? Koji delovi organizacije utiču na sistem?
3. Ko su potencijalni korisnici?
4. Koji nefunkcionalni zahtevi: performanse sistema, operativne mogućnosti, su u skladu sa radnim okruženjem, drugim sistemima, softverom i hardverom.

FAZA III – ANALIZA ZAHTEVA SISTEMA (SISTEMSKA ANALIZA)

Organizacijama stoje na raspolaganju tri solucije da reše poslovni problem datog informacionog sistema:

- Ne raditi ništa i nastaviti sa korišćenjem postojećeg sistema bez promena
- Modifikovati i poboljšati postojeći sistem
- Razviti u potpunosti nov sistem

Glavni **cilj** sistemske analize je u prikupljanju informacija o postojećem sistemu, u cilju donošenja odluke koju od ove tri solucije odabrati.

Uloge i odgovornosti **sistem analitičara** su:

1. istraživačka uloga: sticanje saznanja o načinu funkcionisanja poslovog sistema i okruženja, proučavanje zahteva
2. stvaralačka uloga: predviđanje i definisanje zahteva, novi načini obavljanja određenih stvari
3. uloga u formalizovanju: definisanje prioriteta, usaglašavanje i dokumentovanje zahteva

FAZA III – ANALIZA ZAHTEVA SISTEMA (SISTEMSKA ANALIZA)

Tehnike istraživanja:

1. intervjuisanje budućih korisnika sistema. Intervjui mogu biti struktuirani (sa unapred pripremljenim i napisanim pitanjima) i nestruktuirani (gde analitičar ne poseduje unapred pripremljena pitanja, već koristeći svoja prethodna iskustva u razvoju sistema, već postavlja ad hoc pitanja o određenim delovima funkcionisanja sistema).
2. upitnici (najpraktičniji!)
3. dokumentacija za čitanje
4. posmatranje samog sistema

FAZA III – ANALIZA ZAHTEVA SISTEMA (SISTEMSKA ANALIZA)

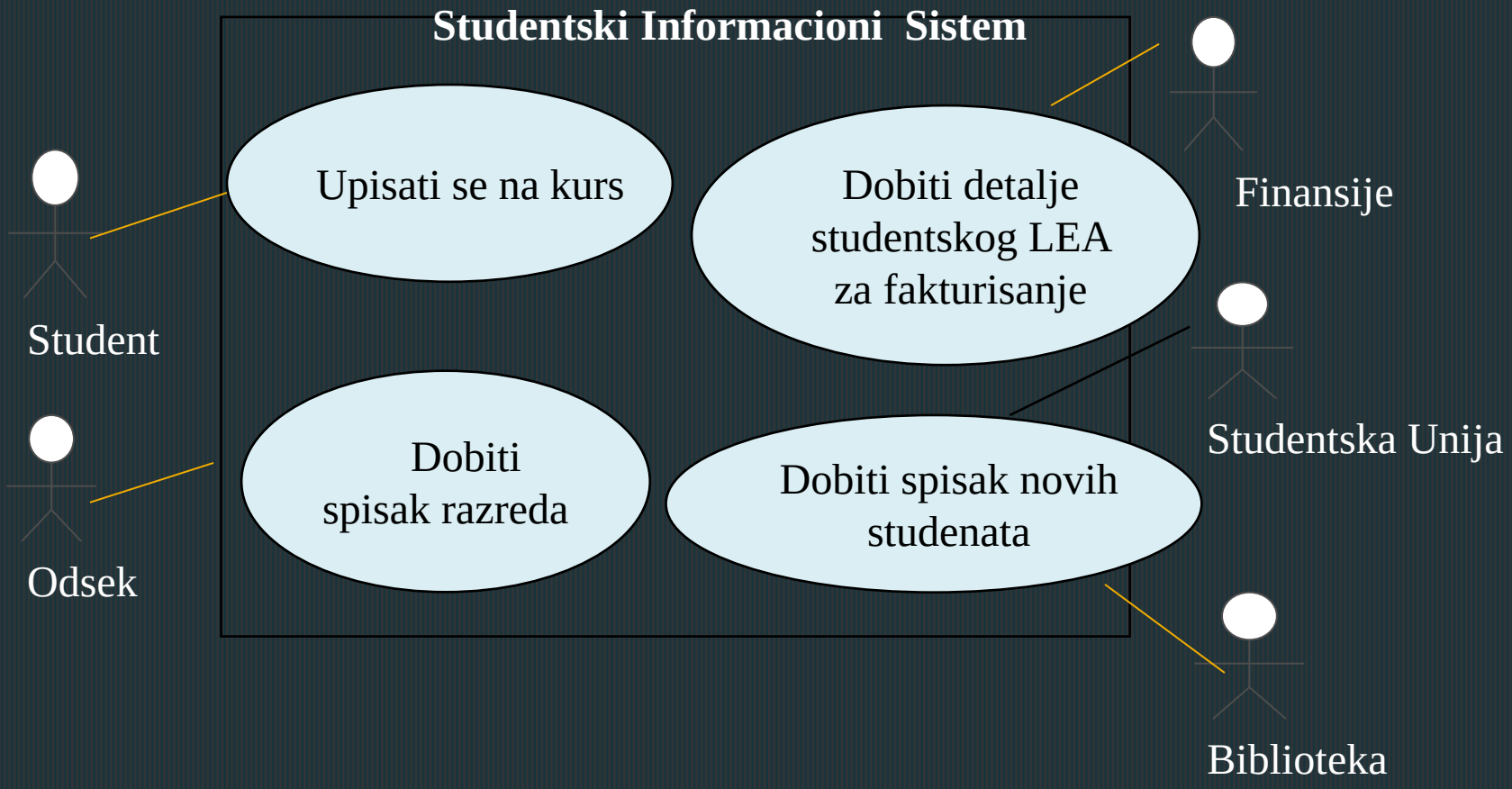
Tehnike modeliranja:

1. **upotreba CASE dijagrama** (eng. use cases diagrams) – ko koristi sistem i zašto
2. **dijagrami toka informacija** (eng. **I**nformation **F**low **D**iagrams) – kako informacija teče *između* sistema
3. **dijagrami toka podataka** (eng. **D**ata **F**low **D**iagrams)- kako podaci teku *unutar* sistema
4. **bogate slike** (eng. Rich Pictures)- šta su važni elementi sistema
5. **dijagrami entitetskih relacija** (eng. **ERD**, **E**ntity **R**elationship **D**iagrams)- ilustruje logičku strukturu baze odnosno kako će baza podataka sistema biti projektovana (dizajnirana)

Krajnji proizvod sistemske analize predstavlja skup *informacionih zahteva* koje sistem mora da zadovolji. Informacioni zahtevi određuju koje informacije su potrebne sistemu, koja količina informacija, od koga se određena informacija dobija, kome se koja informacija prosleđuje i u kom formatu.

FAZA III – ANALIZA ZAHTEVA SISTEMA (SISTEMSKA ANALIZA)

Tehnike modeliranja pomoću CASE dijagrama



FAZA IV – PROJEKTOVANJE SISTEMA

Analizu zahteva sistema možemo poistovetiti sa *”razumevanjem problema”*: šta sistem mora da uradi da bi rešio poslovni problem. Projektovanje sistema se odnosi na ”*razumevanje rešenja”*: *kako* će sistem da izvrši taj zadatak, kao i razmatranje mogućih alternatinih rešenja, njihova evaluacija (troškovi, izvodljivost, implikacije), izbor najboljeg rešenja i opis izabranog rešenja sa dovoljno detalja (da li će on biti kupljen, izgrađen ili sastavljen).

Projektovanje sistema definiše do detalja kako izgraditi sistem. Iz faze analiza zahteva sistema (faza 3), prelazimo na fazu projektovanja sistema (faza 4), u okviru koje imamo projektovanje i logičkog i fizičkog aspekta novog sistema

FAZA IV – PROJEKTOVANJE SISTEMA

FAZA IV – PROJEKTOVANJE SISTEMA

Projektovanje sistema definiše do detalja kako izgraditi sistem. U ovoj fazi se javlja projektovanje i logičkog i fizičkog aspekta novog sistema. U okviru **projektovanja logičkog sistema** određuje se šta sistem treba da radi (funkcionalnost sistema), koristeći apstraktne specifikacije. U okviru **projektovanja fizičkog sistema**, specificiraju se sve aktuelne komponente koje se koriste u implementaciji logičkog projektovanja, koncizno se definiše neophodan hardver, softver, baze podataka, telekomunikacije i procedure. Na primer, u logičkom projektovanju telekomunikacionog sistema kompanije definiše se potreba za uspostavljanjem veze sa glavnim sedištem kompanije.

Fizičko projektovanje telekomunikacionog sistema određuje tip komunikacionog hardvera (tip računara i rutera), softvera (određeni mrežni operativni sistem), i komunikacija (satelitska veza).

FAZA IV – PROJEKTOVANJE SISTEMA

Projektanti treba da razumeju šta je moguće, ekonomično i efektivno i da ponude tehnička rešenja koja bi radela u ciljnoj organizaciji i zadovoljila zahteve. Oni treba da definišu fizički izgled baze podataka (tabele, indekse, relacije...)

Kada se od strane svih učesnika razvojnog tima odobre oba aspekta projektovanja sistema, sistem se «zamrzava». Drugim rečima, kada se jednom uspostavi saglasnost oko specifikacija sistema, one se više ne mogu suštinski menjati. Naime, korisnici obično zahtevaju dodatne opcije u sistemu. Ova pojava se dešava iz više razloga: kao prvo, što korisnici budu jasnije razumevali novi sistem i načine njegovog funkcionisanja, uočavaće dodatne funkcije koje bi oni želeli da sistem uradi.

Takođe, kako vreme prolazi nakon «zamrzavanja» projektnih specifikacija, poslovni uslovi se mogu promeniti, te korisnici i iz tih razloga mogu tražiti povećanu funkcionalnost. Stoga, razvojni tim mora obazrivo razmatrati zahteve korisnika i ne dozvoliti da zbog stalnih izmena projektat premaši planirani budžet ili premaši planirani rok završetka.

FAZA IV – PROJEKTOVANJE SISTEMA

Pitanja koja se javljaju u ovoj fazi su:

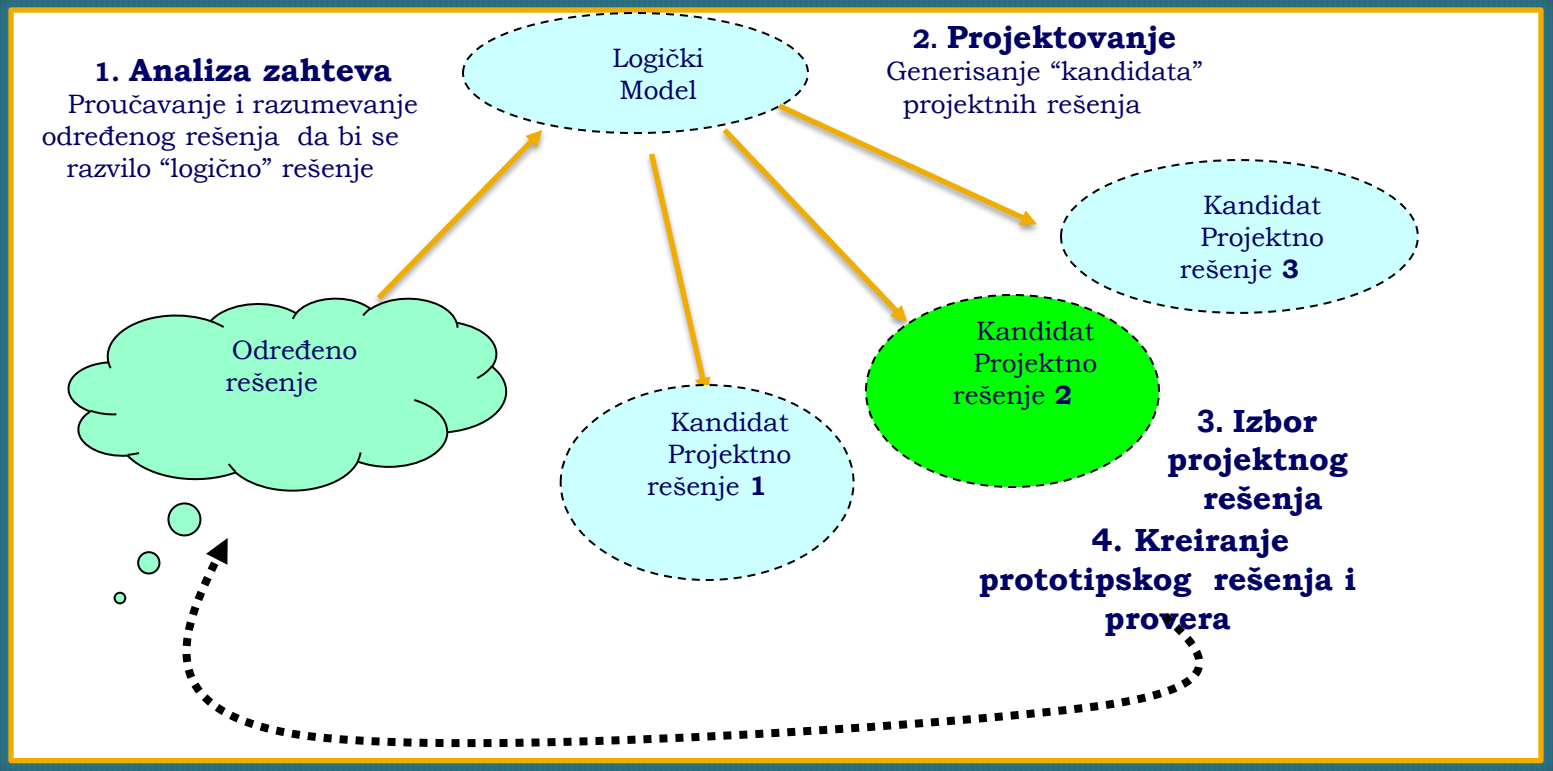
1. Koji hardver, mreže, operativne sisteme koristiti?
2. Koje modele projektovanja baza podataka (distribuirane (svaka lokacija ima deo baze podataka), centralizovane (npr. UNIX mašine- svi podaci su na jednom serveru), klijent-server)?
3. Koji sistem za upravljanje bazama podataka (npr. DBASE, SQL, ORACLE...)? Koji su okviri kontrole i bezbednosti? Šta su ulazi/izlazi?
4. Šta su softverski alati (npr. Visual Basic, Visual C...)? Šta su softverske komponente?
5. Šta se može kupiti, a šta treba posebno razviti? **RAZVITI ili KUPITI**?

Rezultat ove faze je tehničko projektovanje koje definiše:

- sistemske ulaze, izlaze i forme koje će koristi korisnici pri radu sa sistemom
- hardver, sofver, baze podataka, telekomunikacije, kadrove i procedure
- kako će ove komponenete biti integrisane.

FAZA IV – PROJEKTOVANJE SISTEMA

Faze životnog ciklusa u razvoju informacionog sistema su u međusobnoj interakciji, tako da je moguće (u zavisnosti od modela procesa razvoja poslovnog informacionog sistema) da se faze iterativno smenjuju, kao što se može videti na slici:



FAZA V – IZGRADNJA SISTEMA

Izgradnja sistema je kreiranje softvera od strane programera, izgradnja krajnjih verzija softvera i testiranje od strane programera i krajnjih korisnika (programiranje - stari stil), **i** ugradnja postojećih programa i biblioteka (programiranje - novi stil), **i** instalacija i interpretiranje softverskih paketa.

Izgradnja sistema je tehnički proces koji projekte projektanata (faza 4) transformiše u softverske module koji zadovoljavaju zahteve definisane od strane analitičara (faza 3). Uključuje implementiranje projekta kao fizičke baze podataka, izgradnju korisničkog interfejsa, ugradnju poslovne logike u programe, razvijanje i testiranje softverskih komponenti.

FAZA V – IZGRADNJA SISTEMA

Izgradnja sistema je kreiranje softvera od strane programera, izgradnja krajnjih verzija softvera i testiranje od strane programera i krajnjih korisnika (programiranje - stari stil), **i** ugradnja postojećih programa i biblioteka (programiranje - novi stil), **i** instalacija i interpretiranje softverskih paketa.

Izgradnja sistema je tehnički proces koji projekte projektanata (faza 4) transformiše u softverske module koji zadovoljavaju zahteve definisane od strane analitičara (faza 3). Uključuje implementiranje projekta kao fizičke baze podataka, izgradnju korisničkog interfejsa, ugradnju poslovne logike u programe, razvijanje i testiranje softverskih komponenti.

FAZA VI – IMPLEMENTACIJA SISTEMA

Implementacija je proces prelaska sa starog na nov sistem i uključuje i pripreme za ovaj prelazak. Implementacija podrazumeva proveravanje da su hardverske i mrežne infrastrukture novog sistema ″na svom mestu″; testiranje sistema i kadrova kako na najbolji način da se edukuju i osposobe zaposleni koji će koristiti nov sistem ili su pod uticajem novog sistema.

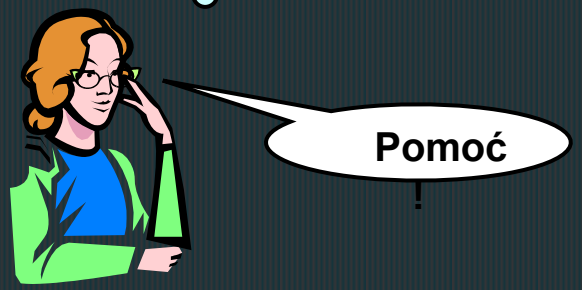
U fazi implementacije treba da se:

Prestane sa korišćenjem starog sistema
Prestane sa starim načinom rada

Prebace podaci (eng. data migration)
Započne sa upotrebom novog sistema
Započne sa novim načinom rada

Upravlja se već započetim poslovi

Kada da stanem, šta da zaustavim?
Kada da stanem, šta da zaustavim?
Ko to radi, kada? Da li će to biti ispravno?
Koja će se obuka sprovesti u novom sistemu?
Koja će se obuka sprovesti u novim procedurama? Da li da koristim stari sistem, stare procedure ili nove?



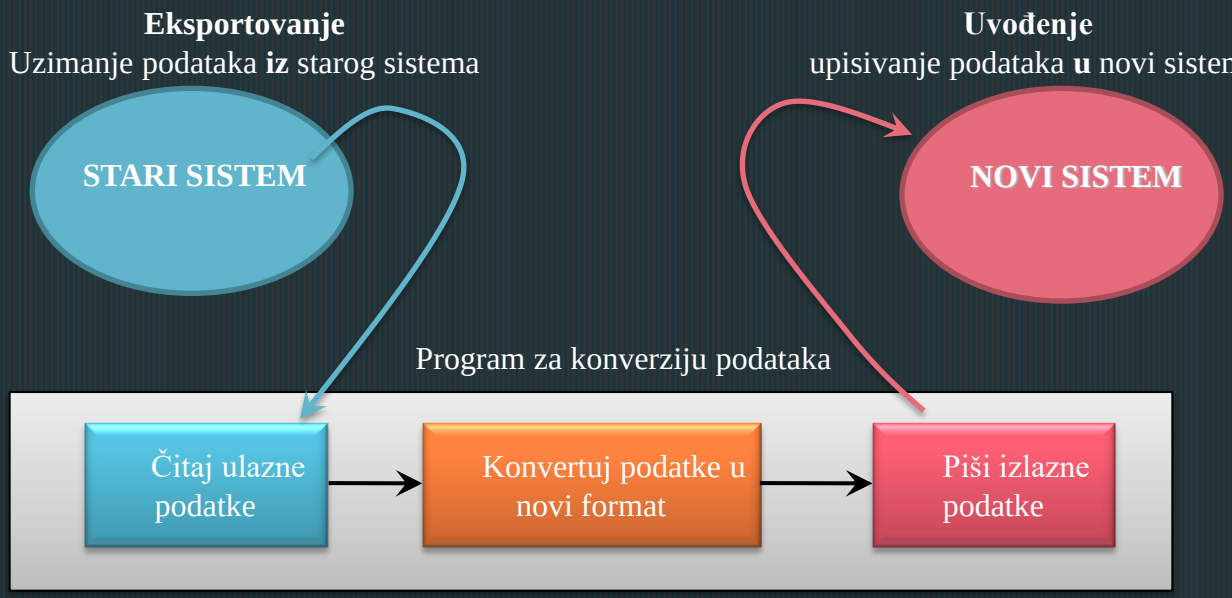
FAZAFAZA VI – IMPLEMENTACIJA SISTEMA

Implementacija se odnosi na:

1. *konverziju i prebacivanje podataka* (eng. Data migration)
2. sve vidove *promena* u organizacionoj strukturi i *obuku* kako korisnika tako i zaposlenih na održavanju sistema i sistemskoj podršci
3. *izbor strategije prelaska*
4. *instalaciju i testiranje tehnologije*

FAZA VI – IMPLEMENTACIJA SISTEMA

Konverzija i prebacivanje (migracija) podataka (eng. Data migration) predstavlja proces u kome se podaci *eksportuju* iz starog sistema, *konvertuju* pomoću adekvatnih programa za konverziju, a onda *uvode* u nov sistem. Postoji veliki broj programa za prebacivanje podataka. Ovaj proces je prilično kompleksan jer nepravilno prebačene podatke je praktično nemoguće ispraviti (čak i detektovati). Javlja se i problem kako izbeći duplikate i kako upravljati promenama ako su oba sistema istovremeno u upotrebi.



FAZA VI – IMPLEMENTACIJA SISTEMA

Prilikom prelaska na nov informacioni sistem, neizbežno je uvesti značajne **promene**:

- među korisnicima koji koriste sistem
- u radnim procedurama
- u organizaciji koja koristi sistem
- među spoljnim korisnicima i u organizacijama koji su direktno ili indirektno pod uticajem novog sistema.

FAZA VI – IMPLEMENTACIJA SISTEMA

Organizacije koriste četiri strategije prelaska: paralelna, direktna, pilot i prelazak po fazama:

• Primenom paralelnog prelaska, stari i novi sistem se neko vreme koriste simultano. Naime, oba sistema obrađuju iste podatke u isto vreme, te se potom upoređuju dobijeni rezultati. Ovaj način prelaska je najskuplji, ali zato najmanje rizičan.

• Primenom direktnog prelaska, stari sistem se isključuje (stavlja van upotrebe), a na određeno vreme se uključuje novi sistem. Ovaj tip prelaska je najjeftiniji, ali i najrizičniji ukoliko novi sistem ne funkcioniše kako je planirano.

• Primenom pilot prelaska novi sistem se predstavlja samo u *pojedinom delu organizacije*. Novi sistem se pokreće na određeni period i potom se vrši ocena njegovog rada. Ukoliko su rezultati povoljni, novi sistem se pokreće i u ostalim delovima organizacije, tj. čim se pilot prelazak pokaže uspešnim sledi pokretanje sistema u drugim delovim organizacije.

• Prelazak po fazama odlikuje uvođenje samo *pojedinih komponenti novog sitema*: od onih jednostavnijih ka složenijima. Svaka komponenta se ocenjuje, i ukoliko su rezultati zadovoljavajući, predstavlja se sledeća komponenta, sve dok se ne predstavi celokupan sistem.

FAZA VI – IMPLEMENTACIJA SISTEMA

Izbor strategije prelaska zavisi od sledećih faktora:

1. *troškovi*: treba ih uzeti u obzir ali je ipak važniji kvalitet novog sistema
2. *vreme*: treba da postoji balans između raspoloživog vremena i željenog kvaliteta sistema koji će se vrednovati
3. *kvalitet novog sistema posle implementacije*: ovo će zavisiti od broja grešaka (bagova) i od njegove pogodnosti za upotrebu
4. *uticaj na kupce*: šta će biti efekat korisničkog servisa ako se implementacija prekorači ili novi sistem sadrži greške (bagove)?
5. *uticaj na zaposlene*: koliko će prekovremenog rada zaposlenih zahtevati implementacija? Da li će oni za to biti plaćeni?
6. *tehnička pitanja*: neke opcije neće biti moguće ako sistem nema modularno projektovanje.

FAZA VI – IMPLEMENTACIJA SISTEMA

Instalaciju i testiranje tehnologije: instalacija sistema na veliki broj mesta u organizaciji koja ima složenu organizacionu topološku strukturu, nije ni malo lak zadatak.

Potrebno je izgraditi globalnu i verovatno više lokalnih računarskih mreža, instalisati enterprise softver, više servera aplikacija, rasporediti i uvezati desetine radnih stanica.

I pre i posle instalacije tehnologije, neophodno je izvršiti određena testiranja: hardvera, sistemaskog softvera, softvera za upravljanje bazama podataka.

FAZA VII – ODRŽAVANJE SISTEMA

Nakon potpunog prelaska na nov sistem, sistem se pušta u rad.
Održavanje sistema uključuje proveru, modifikaciju i unapeđenje sistema da bi ga učinili korisnim i efikasnijim u obezbeđivanju i zadovoljavanju informacionih potreba korisnika i postizanju ciljeva organizacije.

Sistem može zahtevati nekoliko tipova održavanja:

1. prvi tip je **ispravljanje grešaka** (eng. debugging), proces kojim se vraćamo u neku od faza iz razvojnog ciklusa,
2. drugi tip je **ažuriranje sistema** (eng. updating) u cilju prilagođavanja sistema sa određenim novonastalim promenama u poslovanju. Primer ažuriranja sistema bi bio usklađivanje sistema sa novim zakonskim regulativama.
3. treći tip održavanja podrazumeva **dodavanje novih karakteristika** postojećem sistemu, bez ugrožavanja operativnosti sistema.

FAZA VII – ODRŽAVANJE SISTEMA

Održavanje realizuje tim za održavanje. Moguća su dva pristupa: a) tim stručnjaka koji je sistem razvio i ujedno ga i održava i b) formira se specijalni tim koji isključivo održava izgrađeni sistem.

Reći ćemo nešto više o softverskim greškama (bagovima): zajednički naziv za problemske greške i defekte u softveru. Male su i neprijatne. Prouzrokovane su ljudskom greškom u programiranju ili projektovanju.

Troškovi održavanja obuhvataju više od 80% ukupnih troškova jednog informacionog sistema.

ZAKLJUČAK

Spoznaja životnog ciklusa razvoja poslovnih informacionih sistema putem kratkog opisa njegovih faza razvoja, naravno nije dovoljna da bi se sa sigurnošću i izveo proces razvoja. Nakon prikaza životnog ciklusa i njegovih aktivnosti, primarno je upoznati i modele procesa razvoja.

MODELI RAZVOJA IS

Evolucija modela procesa u okviru razvoja sistema ukazuje na promenljive potrebe korisnika kompjutera. Kako korisnici traže brže rezultate, veće uključenje u razvojni proces i korišćenje mera za određivanje rizika i efektivnosti, menjaju se i metode za razvoj sistema. Pritom, softverski i hardverski instrumenti koji se koriste u industriji su se značajno promenili (i menjaju se i dalje). Brže mreže i hardveri podržavali su upotrebu jacih i brzih operativnih sistema koji su napravili mesta za nove jezike i baze podataka, kao i za aplikacije koje su daleko moćnije nego prethodne. Ove brze i brojne promene su istovremeno inicirale razvoj praktičnijih novih modela procesa i krah starijih modela koji se više nisu mogli koristiti.

MODELI RAZVOJA IS

Većina modela procesa za razvoj sistema koji se danas koriste izvedeni su iz tri primarna pristupa: ad-hoc razvoj, model vodopada i iterativni proces. Mi ćemo u ovom delu obraditi sledeće modele procesa razvoja informacionog sistema:

1. **Ad-hoc razvoj**
2. **Model vodopada**
3. **V-model**
4. **Iterativni model**
5. **Model prototipa**
6. **RAD model**
7. **Istraživački model**
8. **Spiralni model**
9. **Model ponovne upotrebe (eng. Reuse model)**
10. **Kreiranje i kombinovanje modela**

AD-HOC

Ranije se razvoj sistema često dešavao na prilično haotičan i neplanski način, oslanjajući se u potpunosti na veštine i iskustvo zaposlenih pojedinaca koji su obavljali taj posao.
Klasičan primer ne postojanja metoda u komjuterskimm sistemima bio je i prvi računar za poslovne primene LEO iz 1951 koga su ″isprogramirali″ matematičari bez ikakvih metoda što je prouzrokovalo razna ograničenja u prepoznavanju poslova, potreba, korisnika informacionog sistema, haotičnim rezultatima i nemogućnosti održavanja takvih sistema.
Danas, mnoge organizacije i dalje praktikuju ad-hoc razvoj ili u potpunosti ili samo za neke podsisteme razvoja (npr. za male projekte).
Institut inženjerskog softvera na Univerzitetu Karnedži Melon (eng. The Software Engineering Institute at Carnegie Mellon University) ističe da sa Ad-hoc modelima procesa “sposobnost procesa se ne može predvideti jer se softverski proces stalno menja ili modifikuje kako rad napreduje. Rasporedi, budžeti, funkcionalnost i kvalitet proizvoda su generalno nekonzistentni. Učinak zavisi od sposobnosti pojedinaca i varira u zavisnosti od urođenih veština, znanja i motivacije. Ima malo stabilnih softverskih procesa koji su dokazani i učinak se pre može predvideti sposobnošću pojedinca nego sposobnošću organizacije.“

AD-HOC

“Međutim, dešava se čak da i u nedisciplinovanim organizacijama neki individualni softverski projekti dovedu do odličnih rezultata. Kada takvi projekti uspeju, to se obično dešava zahvaljujući herojskim naporima posvećenog tima a ne ponavljanjem dokazanih metoda organizacije sa potpuno razvijenim softverskim procesom. U odsustvu softverskog procesa koji se može koristiti u celoj organizaciji, ponavljanje rezultata u potpunosti zavisi od toga da isti pojedinci budu raspoloživi za sledeci projekat. *Uspeh koji se jedino bazira na raspoloživosti oderđenih pojedinaca ne obezbeđuje osnovu za dugoročnu produktivnost i poboljšanje kvaliteta u okviru jedne organizacije*.“

VODOPAD

Model vodopada je najraniji metod razvoja strukturisanih sistema. Pojavio se 1960-tih godina. Iako su ga kritikovali poslednjih godina da je rigidan i nerealističan kada se radi o brzom ispunjavanju potreba korisnika, ovaj model se jos uvek puno koristi. On obezbeđuje teoretsku osnovu za druge modele procesa jer najviše podseca na ‘generički’ model za razvoj softvera. Tradicionalni model vodopada opisuje aktivnosti kao niz nivoa (etapa) koji se dešavaju po unapred definisanom redosledu. Uglavnom se završava jedna faza pre nego što počne sledeća, što može predstavljati problem ako neka kasnija faza zahteva povratak na prethodnu.

VODOPAD

- inicijalizacija
- studija
- analiza sistema
- projektovanje sistema
- izgradnja sistema
- implementacija
- održavanje

VODOPAD

Konceptualizacija sistema (inicijalizacija i studije izvodljivosti). Konceptualizacija sistema odnosi se na razmatranje svih aspekata ciljne poslovne funkcije ili procesa, a sa ciljem određivanja međusobnog odnosa ovih aspekata, kao i toga koji aspekt bi mogao biti inkorporiran u sistem.

Analiza sistema. Ovaj korak se odnosi na sakupljanje zahteva sistema sa ciljem da se odredi kako će se ovi zahtevi prilagoditi u okviru sistema. Ono što je od osnovnog značaja je ekstenzivna komunikacija između kupca i osobe zadužene za razvoj.

Projektovanje sistema. Kada se zahtevi prikupe i analiziraju, neophodno je detaljno identifikovati kako konstruisati sistem da bi on izveo neophodne zadatke. To preciznije znači da se faza projektovanja sistema usredsređuje na zahteve podataka (koje informacije će biti procesirane u sistemu?), konstrukciju softvera (kako ce se aplikacija konstruisati?), i konstrukciju interfejsa (kako ce izgledati sistem? koji standardi ce se primenjivati?).

VODOPAD

Izgradnja sistema.
Kodiranje. Takodje poznat pod nazivom programiranje, ovaj korak podrazumeva stvaranje sistemskog softvera. Zahtevi i specifikacije sistema do kojih se došlo u fazi projektovanja se sada prevode u komjuterski kod koji se može masinski očitati.

Testiranje.
Pošto se softver napravi i doda sistemu, sprovodi se testiranje kako bi se osiguralo da ispravno i efikasno radi. Testiranje se uopšteno fokusira na dve oblasti: unutrašnja efikasnost i spoljašnja efektivnost. Cilj spoljašnje efektivnosti je verifikacija softvera, tj. da softver funkcioniše u skladu sa dizajnom sistema i da sprovodi sve neophodne funkcije i podfunkcije. Cilj unutrašnjeg testiranja je da osigura da je kod kompjutera efikasan, standardizovan i dobro dokumentovan. Testiranje može zahtevati dosta rada zbog svoje iterativne prirode.

VODOPAD

Implementacija. Ovaj korak predstavlja proces prelaska sa starog na nov sistem i uključuje i pripreme za ovaj prelazak.

Održavanje. Održavanje sistema uključuje proveru, modifikaciju i unapeđenje sistema da bi ga učinili korisnim i efikasnijim u obezbeđivanju i zadovoljavanju informacionih potreba korisnika i postizanju ciljeva organizacije.

VODOPAD

Problemi / izazovi vezani za Model vodopada

Iako se ovaj model naveliko godinama koristi u proizvodnji mnogih sistema kvaliteta, to ne znači da se ne javljaju problemi. Poslednjih godina on se kritikuje zbog svoje rigidnosti i nefleksibilne procedure. Ove kritike se mogu svrstati u sledeće kategorije:

- istinski projekti retko slede proceduru koju model predlaže
- na početku većine projekata često postoji puno neizvesnosti vezanih za zahteve i ciljeve i zbog toga je korisnicima teško da detaljno identifikuju ove kriterijume. Model vodopada takođe ne prolagođava dobro ovu prirodnu neizvesnost.
- razvijanje sistema uz korišćenje Modela vodopada može biti dug i bolan proces koji ne daje radnu verziju sistema sve do pred kraj procesa.

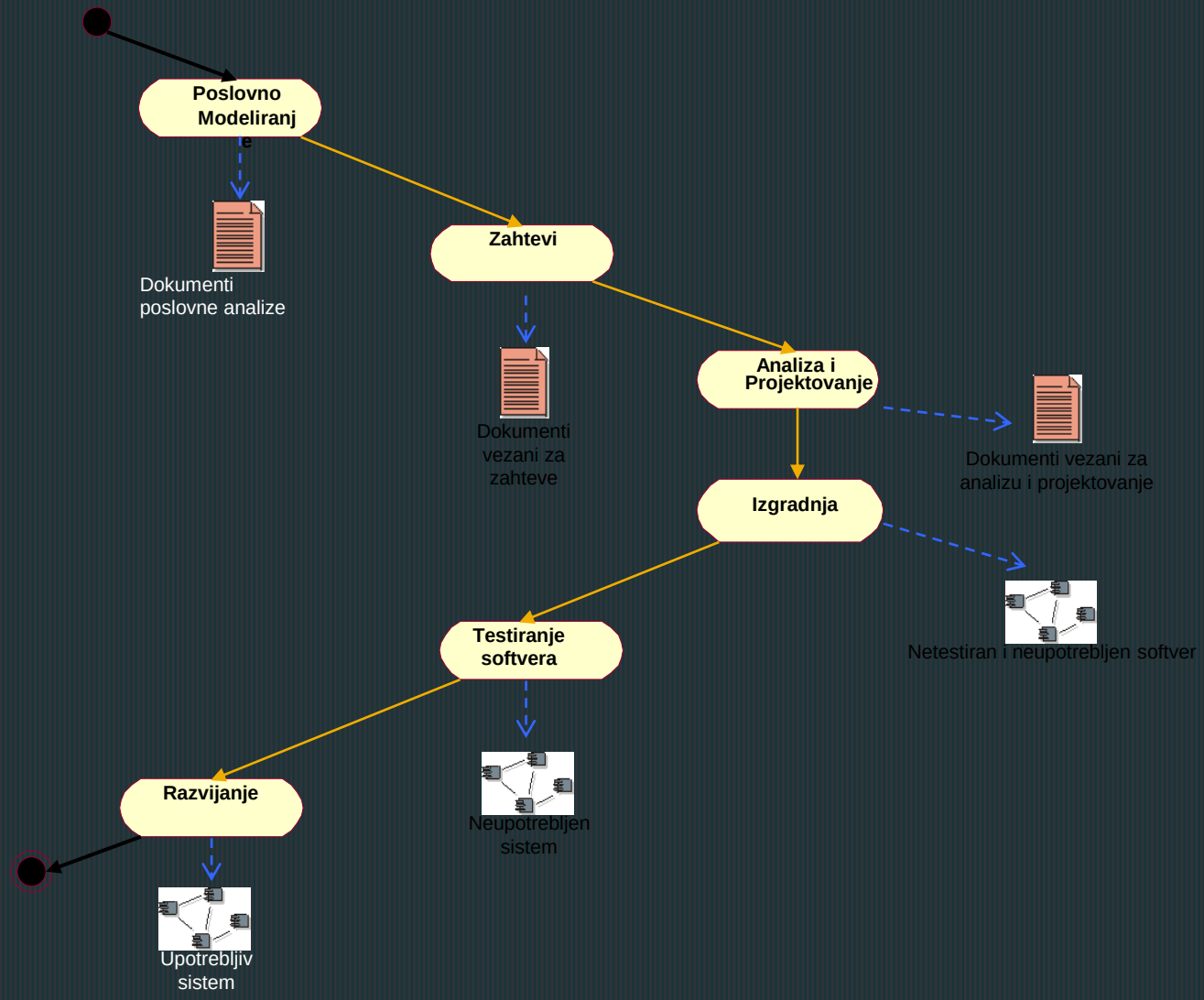
V-model

Ovaj model je razvijen u Nemačkoj za potrebe vlade u domenu projekata odbrane, 1970-tih godina i bazira se na modelu vodopada pri čemu je izlaz iz svakog koraka dokumentacija ili neupotrebljen softver . Predstavljen je u obliku slova **V** da bi označio (prikazao) vezu između aktivnosti. Osnovne postavke ovog modela su: dokumenti imaju vrednost i neupotrebljen softver ima vrednost.

Proces razvoja započinje od gornje leve tačke V-modela udesno i završava se u gornjoj desnoj tački. Sa leve strane naniže razvoj definiše poslovne zahteve, funkcionalne, tehničke i programske specifikacije faze projektovanja sistema . U osnovi V-modela je zapisan kod. Sa desne strane uzlazno, rađeno je testiranje i otklanjanje grešaka (eng. Debugging). Nakon toga sledi testiranje softverskih modula (eng. Unit testing), odozdo-naviše, pa integraciono testiranje (eng. Integration testing) i testiranje prihvatljivosti (testiranje funkcionalnosti) sistema od strane korisnika sistema (eng. User acceptance test). Krajnja gornja tačka prikazuje lansiranje proizvoda.

UVOD U DIGITALNU EKONOMIJU

V-model



V-model

Proces razvoja započinje od gornje leve tačke V-modela udesno i završava se u gornjoj desnoj tački. Sa leve strane naniže razvoj definiše poslovne zahteve, funkcionalne, tehničke i programske specifikacije faze projektovanja sistema . U osnovi V-modela je zapisan kod. Sa desne strane uzlazno, rađeno je testiranje i otklanjanje grešaka (eng. Debugging). Nakon toga sledi testiranje softverskih modula (eng. Unit testing), odozdo-naviše, pa integraciono testiranje (eng. Integration testing) i testiranje prihvatljivosti (testiranje funkcionalnosti) sistema od strane korisnika sistema (eng. User acceptance test). Krajnja gornja tačka prikazuje lansiranje proizvoda.

V-model prikazuje kompleksnost veza između svake faze životnog ciklusa razvoja, pri čemu treba imati u vidu da za svaku fazu razvoja postoji njoj odgovarajuća faza testiranja.

V-model je prihvaćen zbog svoje jednostavnosti i jasnosti. Međutim, neki stručnjaci zaduženi za razvoj sistema smatraju da je V-model previše rigidan (tvrd, čvrst) za razvijanje prirode jednog IT (informacione tehnologije) poslovnog okruženja.

V-model

Problemi sa tradicinalnim razvojem sistema (Model vodopada i V-model)

Tradicionalno izgrađene metodologije imaju tendenciju da isporuče sisteme koji prekasno stižu i stoga više ne mogu da zadovoljavaju njihove originaln zahteve. Problemi koji se tu javljaju:

- jaz u razumevanju između korisnika i stručnjaka zaduženih za razvoj sistema
- težnja stručnjaka zaduženih za razvoj sistema da se izoluju od korisnika
- dugo vreme razvoja
- poslovne potrebe se menjaju tokom procesa razvoja
- ono šta korisnici dobiju nije baš ono šta oni žele
- ignorisanje rizika (vodopad pristup za projekte informacionih sistema omogućavaju da se rizici ignorišu: organizacija može jedino da otkrije da li sistem zaista radi - u fazi implementacije).

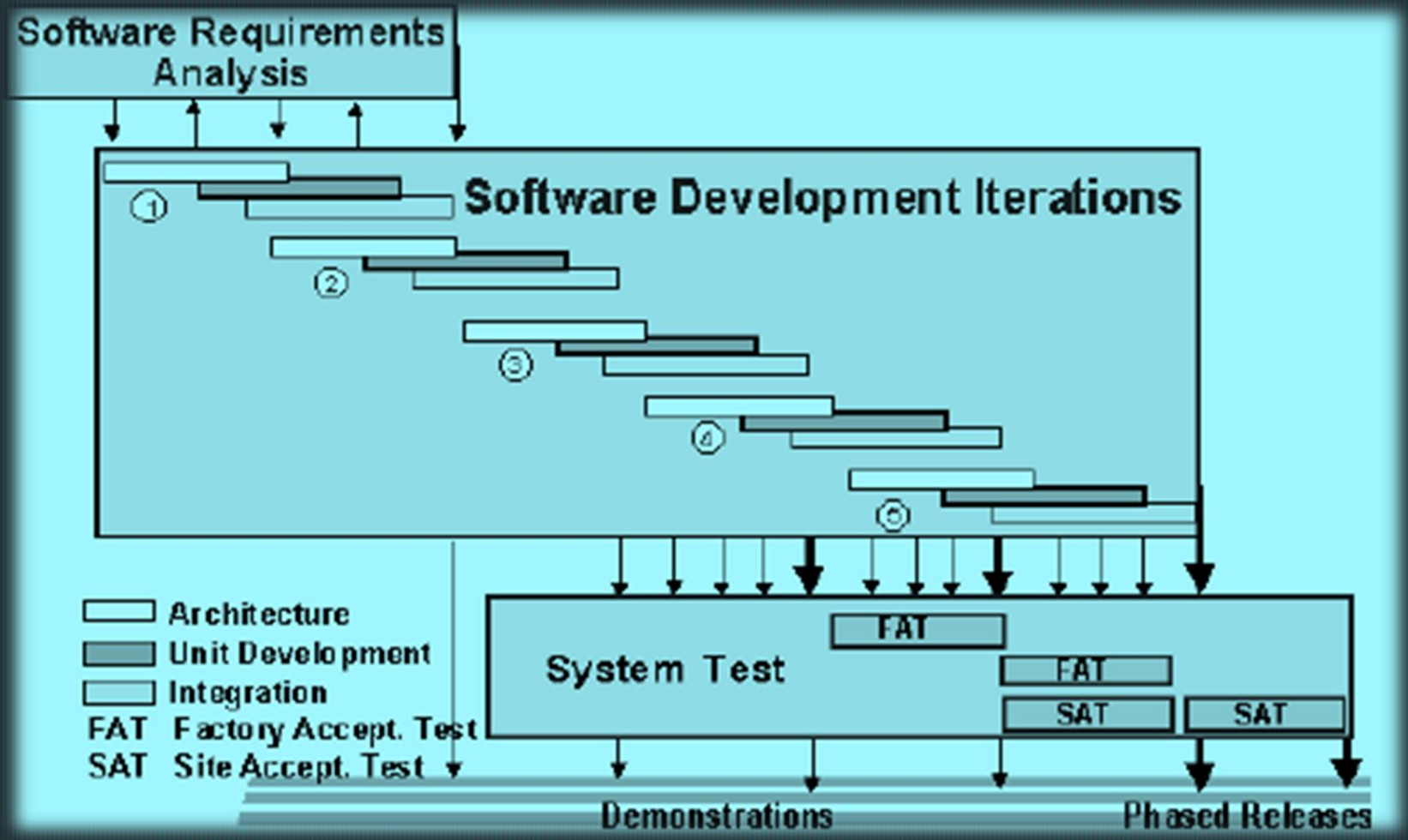
ITERATIVNI RAZVOJ

Problemi koji su se javili u Modelu vodopada doveli su do potrebe za novom metodom razvoja sistema koji bi omogućio brze rezultate, koji bi zahtevao manje početnih informacija i nudio veću fleksibilnost.

Kada se primenjuje iterativni razvoj, projekat se deli na manje delove. Ovo omogućava razvojnom timu da demonstrira rezultate ranije u procesu i da od korisnika sistema dobije vredne povratne informacije. Često, svaka iteracija je zapravo jedan mini-vodopad sa povratnom informacijom iz jedne faze koji pruža informaciju od vitalnog značaja za sledeću fazu.

U varijaciji ovog modela, softverski proizvodi koji su proizvedeni na kraju svakog koraka (ili niza koraka) mogu direktno da pređu u proizvodnju kao postepeni mehanizmi.

ITERATIVNI RAZVOJ



ITERATIVNI RAZVOJ

Problemi / izazovi vezani za iterativni model

Iako iterativni model rešava mnoge probleme vezane za model vodopada, on nosi sa sobom i nove izazove:

- korisnici se moraju aktivno uključiti u toku celog procesa. Ovo je sa jedne strane pozitivno po projekat, ali sa druge strane zahteva puno vremena i može odložiti izvrsenje samog projekta
- centralnu fazu razvoja projekta čine veštine komunikacije i koordinacije
- neformalno, zahtevi za unapređenjem nakon svake faze mogu voditi konfuziji – potrebno je stoga razviti kontrolni mehanizam za rukovanje važnim zahtevima
- iterativni model moze voditi ka “širenju van predviđenog opsega” jer povratne informacije koje se dobijaju od korisnika mogu dovesti do toga da korisnici imaju veće zahteve. Kako korisnici uočavaju razvoj sistema, mogu shvatiti potencijal sposobnosti drugih sistema što će povećati njihov rad.

ITERATIVNI RAZVOJ

Varijacije iterativnog razvoja

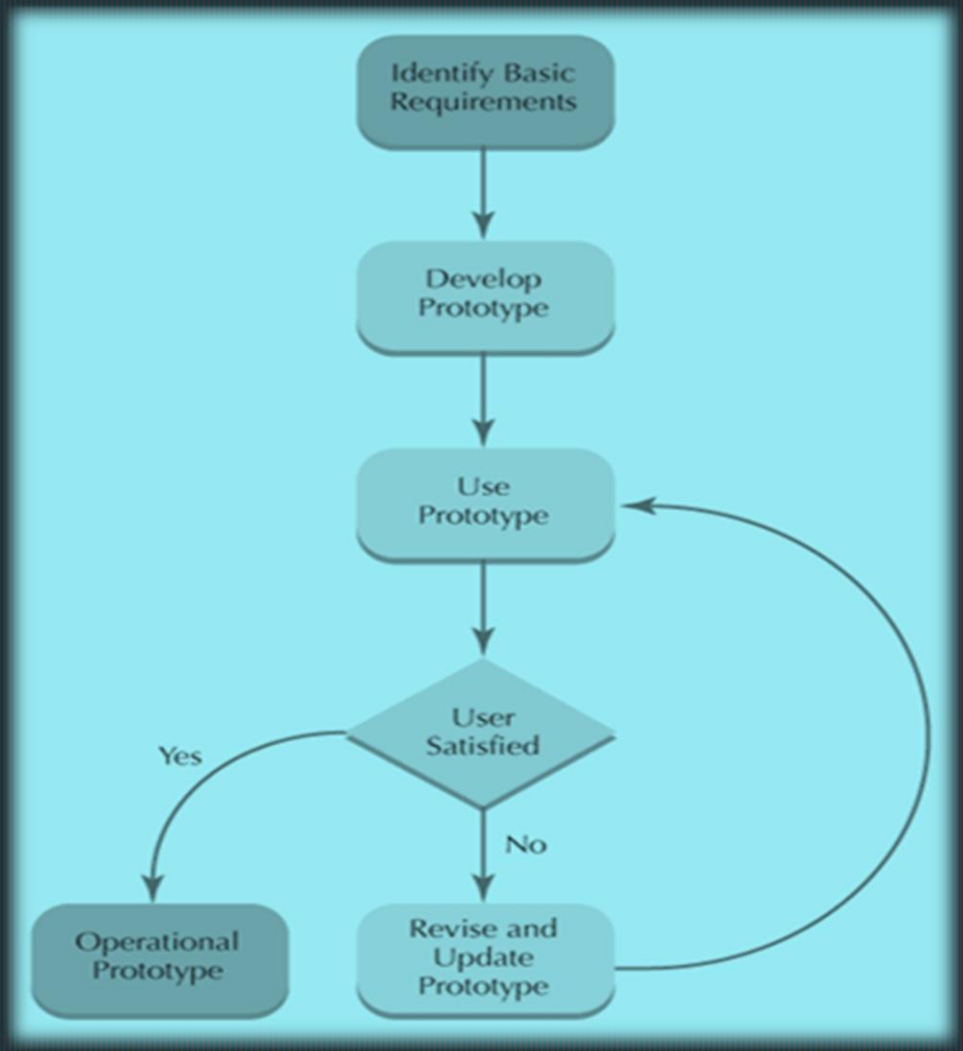
Izvestan broj procesa modela su razvijeni iz iterativnog pristupa. Sve ove metode proizvode neke softverske proizvode koje je moguće pokazati rano u procesu da bi se došlo do vredne povratne informacije od strane korisnika sistema ili drugih članova projektnog tima. Neke od ovih metoda su opisane u daljem tekstu.

MODEL PROTOTIPA

Model prototipa je razvijen na pretpostavci da je često teško znati sve zahteve na početku projekta.Uobičajno je da korisnici znaju mnoge ciljeve koje žele da dostignu ili reše u okviru sistema, ali ne znaju sve podatke niti detalje kada su u pitanju karakteristike i sposobnosti samog sistema. Model pravljenja prototipa omogućava ove uslove i nudi pristup razvoju koji dovodi do rezultata a da ne zahteva prethodno poznavanje svih početnih informacija.

Kada se koristi ovaj model, osoba zadužena za razvoj gradi *uprošćenu verziju* predloženog sistema i prezentuje ga korisniku kako bi ga ovaj razmotrio kao deo razvojnog procesa. Zauzvrat korisnik daje povratnu informaciju osobi za razvoj koji onda poboljšava zahteve sistema kako bi uključio dodatne informacije (slika 33.). Često se dešava da, kada se zahtevi jednom identifikuju, *kod* prototipa se odbacuje i razvijaju se potpuno novi programi.

MODEL PROTOTIPA



MODEL PROTOTIPA

Ima nekoliko različitih **pristupa** koji se mogu primeniti kada se koristi model prototipa:

- stvaranje glavnih korisnih interfejsa bez ikakvih važnih kodiranja u pozadini da bi korisnici razvili ‘osećaj’ za to kakav će sistem biti.
- razvoj skraćene verzije sistema koji može da odradi ograničeni niz funkcija; razvoj sistema na papiru (opis predloženog ekrana, izvestaja, odnosa/veza itd.) ili
- upotreba postojećeg sistema ili komponenata sistema kako bi se demonstrirale neke funkcije koje će biti uključene u krajnji sistem.

MODEL PROTOTIPA

Stvaranje prototipa sadrži sledeće korake:

1. **Implementacija i definisanje/sakupljanje zahteva**. Slično fazi konceptualizacije u modelu vodopada, ali nije toliko sveobuhvatno. Sakupljene informacije su obično limitirane na podskup totalnih zahteva sistema. Ovaj korak sadrži i studiju izvodljivosti i upravljanje promenama i rizicima.
2. **Planiranje projekta.** Kada se prikupe početne informacije ili kada se sakupe nove, one se ubrzano integrišu u novi ili postojeći projekat tako da se mogu primeniti u prototipu.
3. **Stvaranje/modifikovanje prototipa**. Informacije iz prethodne faze se brzo primenjuju na prototip. Ovo može podrazumevati stvaranje/modifikovanje informacija na papiru, novo kodiranje ili modifikovanje do postojećeg koda.
4. **Finalna implementacija sistema**. U većini slučajeva, sistem se prepisuje nakon što se zahtevi razumeju. Podrazumeva testiranje sistema i njegove prihvatljivosti (testiraje funkcionalnosti), konverziju i prebacivanje (migracija) podataka u novi sistem. Ponekad iterativni proces na kraju dovodi do radnog sistema koji je temelj sistema koji u potpunosti funkcioniše.
5. **Održavanje:** praćenje i unapređivanje sistema.

MODEL PROTOTIPA

Problemi / izazovi vezani za model prototipa

Kritike vezane za ovaj model generalno se svstavaju u sledeće kategorije:

- Stvaranje prototipa može dovesti do pogrešnih očekivanja. Stvaranje prototipa često stvara situaciju u kojoj korisnik pogrešno veruje da je sistem ‘završen’ kada on u stvari nije završen. Preciznije, kada koristimo ovaj model, verzije pre same implementacije nisu ništa drugo do jednodimenzionalne strukture. Neophodni ‘pozadinski’ rad kao što je sređivanje baze podataka, prikupljanje dokumentacija, testiranje i ocenjivanje efikasnosti jos nije urađen. To znači da neophodna podrška sistema jos uvek nije obezbeđena.
- Stvaranje prototipa može dovesti do loše dizajniranih sistema. Pošto je primarni cilj ovog modela brži razvoj, dizajn sistema ponekad može da trpi jer se sistem izgrađuje u nizovima ‘nivoa’ a da se ne gleda kako se komponente međusobno slažu. Dok se inicijalni razvoj softvera stvara kao nešto što ce biti ‘odbačeno’, pokušaj da se retroaktivno proizvede solidan dizajn ponekad može biti problematičan.

BRZI RAZVOJ APLIKACIJA - RAD

Brzi razvoj aplikacija (eng. RAD: Rapid Application Development) je model procesa razvoja softvera koji podrazumeva veoma kratak ciklus razvoja (uglavnom 60-90 dana). RAD model, je brza adaptacija modela vodopada, gde je rezultat svakog ciklusa potpuno funkcionalan sistem.

Mali timovi stručnjaka rade ubrzano u blizini korisničkog okruženja izgrađujući *prototipove* sistema i isprobavajući ih. Prototip proizvodi probnu verziju dela ili okvira jednog informacionog sistema koji može biti isproban i proveren od strane krajnjih korisnika. RAD je u osnovi jedan iterativni proces u kome korisnici predlažu modifikacije pre izgradnje daljih prototipova ili pre nego što se izgradi krajnji informacioni sistem. U daljem tektu ćemo koristiti skraćenicu RAD model.

BRZI RAZVOJ APLIKACIJA - RAD

RAD se primarno koristi za aplikacije informacionog sistema. RAD pristup uključuje sledeće **faze**:

1. **Modeliranje poslova (poslovnih funkcija)**

Tok informacija kroz poslovne funkcije je modelovan na način koji daje odgovore na sledeća pitanja:

- Koje informacije nose poslovni proces?
- Koje informacije su generisane?
- Ko ih generiše?
- Gde idu informacije?
- Ko ih obrađuje?

2. **Modeliranje podataka**

Tok informacija definisan kao deo faze modeliranja poslova je prečićen u skup objekata podataka koji su potrebni za podršku poslovima. Definisane su karakteristike (atributi) svakog objekta i njihove međusobne veze.

BRZI RAZVOJ APLIKACIJA - RAD

3. Modeliranje procesa

Objekti podataka definisani u fazi modeliranja podataka se transformišu da bi se postigao tok informacija potreban za implementiranje poslovne funkcije. Opisi obrađivanja (procesiranja) su kreirani za sabiranje, menjanje, brisanje ili obnavljanje objekata podataka.

4. Stvaranje (generisanje) aplikacije

RAD model daje prednost upotrebi četvrte generacije RAD tehnika i oruđa kao što su: Visual Basic, Visual C++, Delphi itd. više nego kreiranje softvera korišćenjem konvencionalnih programskih jezika treće generacije. RAD model koristi, ponovo upotrebljava (eng. Reuse) već postojeće programske komponente ili kreira komponente koje se ponovo mogu upotrebiti (ukoliko je to potrebno). U svakom slučaju automatizovana oruđa se koriste da olakšaju izgradnju softvera.

5. Testiranje

S ozirom da RAD proces naglašava ponovnu upotrebu (eng. Reuse), mnoge programske komponente su već testirane. Ovo minimizira vreme namenjeno testiranju i razvoju. Ako jedna poslovna aplikacija može biti modularizovana (segmentirana, podeljena na više delova) tako da se svaka važnija funkcija može završiti unutar razvojnog ciklusa, ona je ona kandidat za RAD model. U tom slučaju svakom timu može biti dodeljen jedan model koji se posle integriše u celinu.

BRZI RAZVOJ APLIKACIJA - RAD

Nedostaci RAD modela:

· Kada se razvija veliki projekat, tada je neophodan veliki broj izvršioca koji bi se struktuirali u veći broj projektnih RAD timova.

· RAD projekat će propasti ako ne postoji predanost u radu projektanata i korisnika i spremnost za brzom realizacijom pojedinih aktivnosti što je bitno ako bi hteli da se sistem završi u mnogo kraćem vremenskom periodu.

· Ako se sistem ne može adekvatno modularizovati, izgradnja komponenti RAD modela će biti problematična i neizvesna.

· RAD nije pogodan kada je u pitanju visok stepen rizika od novih tehnologija, pa nove aplikacije sistema teško omogućuju njihovu primenu.

ISTRAŽIVAČKI MODEL

Istrživački model je metod razvoja sistema koji se koristi za projektovanje i razvijanje komjuterskog sistema ili proizvoda i u osnovi se sastoji od planiranja i isprobavanja različitih projekata sve dok se jedan ne učini podesnim za dalje razvijanje. Ovaj model najbolje radi u situacijama kada malo ili nijedan od zahteva sistema nisu poznati do detalja na početku projekta.

U nekim situacijama vrlo je teško, čak nemoguće, identifikovati bilo koji zahtev sistema na početku projekta. Teoretske oblasti kao što je veštačka inteligencija su potencijalni kandidati za upotrebu *istraživackog modela* jer se puno istraživanja koja su sprovedena u ovim oblastima baziraju na nagađanju, procenjivanju i hipotezama. U ovim slučajevima, procena se pravi sa ciljem da se vidi da li sistem može da radi a onda se koriste brže iteracije kako bi se predložene promene brzo inkorporirale i kako bi se izgradio upotrebljiv sistem. Značajna karakteristika *istraživačkog modela* je odsustvo preciznih specifikacija. Validnost se bazira na adekvatnosti krajnjeg rezultata a ne u skladu sa zahtevima koji su unapred zamišljeni.

ISTRAŽIVAČKI MODEL

Ovaj model je izuzetno jednostavan po svojoj konstrukciji; sastoji se od sledećih **koraka**:

1. Razvoj početnih specifikacija. Moguće je korisćenje bilo kojih informacija odmah, formira se kratka specifikacija sistema koja omogućava osnovni početni korak.
2. Konstrukcija/modifikacija sistema. Sistem se stvara i/ili modifikuje prema raspoloživim informacijama.
3. Testiranje sistema. Sistem se testira kako bi se moglo videti šta on može da uradi, šta se iz njega može naučiti, kako se on može poboljšati.
4. Implementacija sistema. Višestruka ponavljanja prethodna dva koraka dovode do zadovoljavajućih rezultata i za sistem se kaze da je ‘završen’ i implementiran.

ISTRAŽIVAČKI MODEL

Problemi / izazovi vezani za *istraživački model*

Postoje brojne kritike kada je ovaj model u pitanju:

- njegova upotreba je ogranicena na jezike na visokom nivou koji omogućavaju brzi razvoj kao što je LISP.
- teško je izmeriti ili predvideti koliko bi iznosili najmanji troškovi a da se postigne najveća efektivnost
- kao i kod *modela prototipa*, upotreba *istraživackog modela* obično za rezultat ima nedovoljno efikasne ili nedovoljno precizno dizajnirane sisteme jer nema prethodnog razmatranja kako proizvesti sistem koji bi bio i jednostavan i efikasan.

SPIRALNI MODEL

Ovaj model je tako dizajniran da obuhvati najbolje karakteristike modela vodopada i prototipa i uvodi novu komponentu – procenu rizika. Termin "spirala" koristi se kako bi se opisao proces koji sledi nakon razvoja sistema. Slično modelu prototipa, *razvija se početna verzija sistema a onda se stalno modifikuje na osnovu informacija koje se dobijaju od korisnika*, odnosno vrši se evaluacija korisničih zahteva. Međutim za razliku od modela prototipa, razvoj svake verzije sistema se pažljivo dizajnira koristeći korake iz modela vodopada. Sa svakom iteracijom oko spirale (počinje se u centru i radi se ka spolja) izgradjuju se sve progresivnije verzije sistema.

Procena rizika je jedan od koraka u procesu razvoja kao sredstvo evaluacije svake verzije sistema kako bi se odredilo da li je potrebno nastaviti razvoj ili ne. Ako korisnik odluči da je bilo koji od identifikovanih rizika preveliki, razvoj projekta se stopira. Npr., ako se u toku jedne faze procene rizika identifikuje postojanje značajnog povećanja troškova ili se dovede u pitanje vreme završetka projekta, korisnik ili stručnjak zadužen za razvoj mogu odličiti da nema smisla da se nastavi takav projekat, jer povećanje troškova ili vremena za završetak projekta znači da je taj projekat nepraktičan i neisplativ. Spence L, University of Sutherland, "Software Engineering,":

SPIRALNI MODEL

Model spirale čine sledeći koraci:

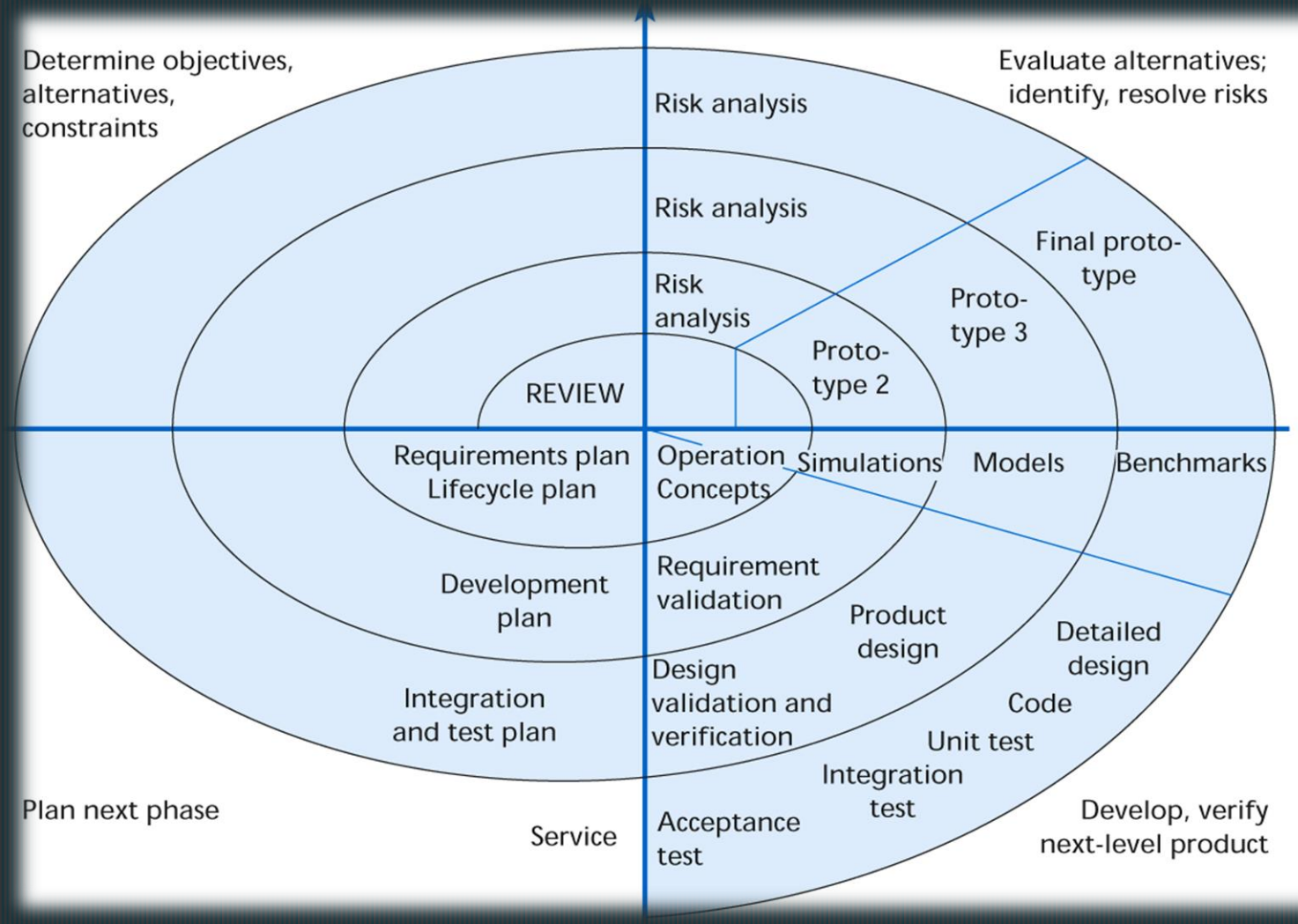
Ciljevi projekta. Ovo je slično fazi konceptualizacije sistema u modelu vodopada. Određuju se ciljevi, identifikuju se moguće prepreke i alternativni pristupi.

Procena rizika. Stručnjak zadužen za razvoj proučava moguće alternative i identifikuju se njima pridruženi rizici/problemi. Procenjuju se i odmeravaju odluke vezane za rizike u cilju daljeg nastavka projekta. Ponekad se koriste prototipi kako bi se pojasnile potrebe.

Inženjering i proizvodnja. Određuju se detaljni zahtevi i razvija se softver.

Planiranje i menadžment (upravljanje). Korisniku se pruža mogućnost da analizira rezultate verzije koja je napravljena u prethodnoj fazi i prilika da da povratnu informaciju stručnjaku zaduženom za razvoj.

SPIRALNI MODEL



SPIRALNI MODEL

Problemi / izazovi vezani za spiralni model

Pošto je ovaj model relativno novijeg datuma – datira iz 1988.godine, teško je proceniti njegove dobre i loše strane. Međutim, jedna komponenta ovog modela – procena rizika - je instrument i za korisnike i za stručnjake zadužene za razvoj, a ovakvu mogućnost raniji modeli procesa nisu posedovali.

Merenje rizika je karakterisitika koja se svakodnevno javlja u realnim životnim situacijama, ali (na žalost) ne toliko često u industriji razvoja sistema. Zahvaljujuci ovoj praktičnoj primeni mogućnosti merenja rizika, spiralni model je realniji model procesa nego sto je to bio slučaj sa modelima koji su mu prethodili.

MODEL PONOVNE UPOTREBE

Osnovna premisa koja stoji iza ovog modela jeste taj da *sisteme treba graditi uz upotrebu postojećih komponenata*, nasuprot običaju da se stalno prave nove komponente. Model ponovne upotrebe je jasno prilagođen kompjuterskoj sredini koja je orijentisana na postizanje ciljeva, što je jedna od prvih tehnologija u današnjoj industriji razvoja sistema.

U okviru modela ponovne upotrebe, postoje biblioteke softverskih modula koje se mogu kopirati i potom koristiti u bilo kom sistemu. Postoje dve vrste ovih komponenata: proceduralni moduli i moduli baze podataka. Kada se gradi novi sistem, osoba zadužena za razvoj ce ‘pozajmiti’ kopiju modula iz biblioteke sistema a onda će je ugraditi u određenu funkciju ili proceduru. Ako željeni modul nije na raspolaganju, stručnjak zadužen za razvoj će je napraviti i ostaviti njenu kopiju u biblioteci za buduću upotrebu. Ako su moduli inženjerski dobro napravljeni, stručnjak koji se bavi razvojem ih može implementirati uz minimalne izmene.

MODEL PONOVNE UPOTREBE

Model ponovne upotrebe se sastoji iz sledećih koraka:

- Određivanje zahteva. Sakupljaju se inicijalni zahtevi i ovi zahtevi su obično podskup svih zahteva sistema.
- Definisanje ciljeva. Identifikuju se ciljevi koji mogu da podrže neophodne komponenete sistema.
- Prikupljanje ciljeva. Vrši se skeniranje biblioteka kako bi se odlučilo da li su željeni ciljevi na raspolaganju ili ne. Nakon toga ‘skidaju’ se (download) kopije potrebnih kopija iz sistema.
- Stvaranje prilagođenih ciljeva. Dolazi do stvaranja onih ciljeva za koje je utvrđeno da su neophodni ali koji nisu na raspolaganju u biblioteci.
- Sakupljanje prototipa. Pravi se verzija prototipa i/ili se modifikuje, uz upotrebu neophodnih ciljeva.
- Evaluacija prototipa. Vrši se evaluacija prototipa kako bi se odredilo da li on na adekvatan način odgovara potrebama i zahtevima korisnika.
- Usavršavanje zahteva. Zahtevi se dalje obradjuju i usavršavaju kao detaljnija verzija stvorenog prototipa.
- Usavršavanje ciljeva. Ciljevi se obradjuju i usavršavaju kako bi se poklapali sa promenama vezanim za zahteve.

MODEL PONOVNE UPOTREBE

Problemi / izazovi vezani za model ponovne upotrebe

Opšta kritika ovog modela se odnosi na to da je on ograničen na na razvojne sredine koje su orijentisane na cilj(eve). Iako ova sredina postaje sve popularnija, ona se trenutno koristi samo u malom broju aplikacija razvoja sistema.

STVARANJE I KOMBINOVANJE MODELA

U mnogim slučajevima, delovi i procedure iz različitih *modela procesa* se integrišu kako bi podržali razvoj sistema. Do ovoga dolazi jer je većina modela dizajnirana tako da pruži okvir za postizanje uspeha samo pod određenim okolnostima. Kada se okolnosti promene van granica modela, rezultati koje je inače bilo moguće predvideti, sada više to ne dozvoljavaju. Kada dođe do takve situacije, ponekad je neophodno promeniti postojeći model kako bi se prilagodio datoj promeni ili je potrebno usvojiti ili kombinovati različite modele da bi se prilagodili novim okolnostima.

Odabir adekvatnih modela procesa u potpunosti zavisi od dva faktora: organizaciona sredina i priroda aplikacije. Frank Land, sa Ekonomskog fakulteta u Londonu, smatra da odgovarajući pristupi analizi sistema, dizajnu, razvoju i implementaciji treba da se baziraju na odnosu izmedju informacionog sistema i njegove organizacione sredine

STVARANJE I KOMBINOVANJE MODELA

Identifikovane su četiri kategorije ovih odnosa:

Nepromenljiva sredina. Zahtevi kada su informacije u pitanju su nepromenljivi u toku životnog ciklusa sistema (npr., oni koji zavise od naučnih algoritama). Zahtevi se mogu dati nedvosmisleno i razumljivo. Visok stepen tačnosti je od suštinskog značaja. U ovakvoj sredini formalne metode (kao sto su *modeli vodopada i spiralni modeli*) omogućavaju kompletnost i preciznost koje sistem zahteva.

Turbulentna sredina. Organizacija podleže stalnim promenama i zahtevi se stalno menjaju. Sistem koji je razvijen na bazi *tradicionalnog modela vodopada* bi delimično bio zastareo do trenutka kada ga treba implementirati. Mnogi poslovni sistemi spadaju u ovu kategoriju. Uspešni metodi bi uključivali one koji podrazumevaju brži razvoj, neke odbačene kodove (kao u *modelu stvaranja prototipa*), maksimalnu upotrebu koda koji se ponovo koristi, i visoko modulirani dizajn.

STVARANJE I KOMBINOVANJE MODELA

Nesigurna sredina. Zahtevi sistema nisu poznati ili nisu sigurni. Nije moguće unapred tačno definisati zahteve jer se radi o novoj situaciji ili o sistemu koji je inovativan. Ovde, metode razvoja moraju da naglase učenje. Najadekvatniji su eksperimentalni *modeli procesa* koji imaju prednost u odnosu na stvaranje prototipa i brži razvoj.

Prilagodljiva sredina. Sredina se može menjati kao odgovor na razvoj sistema i na taj način uvodi promenjeni skup zahteva. Sistemi učenja (predavanja) i ekspertski sistemi spadaju u ovu kategoriju. Ključna stvar za ove sisteme jeste prilagođavanje i metodologija mora biti takva da omogućava direktno uvođenje novih pravila.

NABAVKA IS

Sticanje PIS-a opisuje *metod nabavke* informacionog sistema za poslovanje. Glavni izbor sistema je raspoloživ za kupovinu (upakovan), recimo primene razvijene u odeljenju sopstvenog informacionog sistema ili iz softverske kuće ili razvijeni sistemi za krajnjeg korisnika.

NABAVKA IS


Metodi pribavljanja BIS-a
Skrojeni razvoj
Razvoj za krajnjeg korisnika
Softver sa police
U kući
Spoljnja kompanija
Sklopljen (Tailored)
Standardan

KUPOVINA GOTOVOG SOFTVERA

PREDNOSTI	NEDOSTACI
Standardne funkcije mogu odmah da se koriste	Implementacija je duga i skupa
Ugrađeno iskustvo eksperata	Implementacija traži organizacione promene
Nema programskih grešaka	Ograničene mogućnosti prilagođavanja
Obezbeđeno održavanje	Mogu nastati problemi sa isporučiocem u budućnosti
Usklađeno sa standardima	VISOKA CENA

IZNAJMLJIVANJE SOFTVERA

PREDNOSTI	NEDOSTACI
Povoljna cena	Vrlo ograničene mogućnosti prilagođavanja
Može odmah da se koristi	Zavisnost od isporučioca
Obezbeđeno održavanje i unapređenja	Skupo – na duži rok
Nema programskih grešaka	

KORIŠĆENJE SOFTVERA OTVORENOG KODA

PREDNOSTI	NEDOSTACI
Vrlo povoljna cena	Zahteva sopstveni tim IT stucnjaka
Omogućava veću nezavsnost od isporučioca	Duže vreme implementacije
Daje mogućnost adaptacije “po meri” korisnika	Mogućnost grešaka prilikom adaptacije
	Mogu nastati problemi sa isporučiocem u budućnosti
	Moguci problemi pri održavanju

RAZVOJ SOPSTVENOG SOFTVERA

PREDNOSTI	NEDOSTACI
Daje mogućnost izrade aplikacija “po meri” korisnika	Zahteva sopstveni tim IT stučnjaka (rizik od odlaska kljucnih ljudi iz firme)
Adaptacije moguće u svako doba	Rizik od loše odabrane tehnike
Vlasništvo nad softverom	Nerealni zahtevi menadžmenta
Nezavisnost od isporučioca	Duže vreme implementacije
	Mogućnost grešaka prilikom izrade
	Visoka cena

KUPI GA: SA POLICE [OFF- THE-SHELF]

Kupovina **sa police** (*off-the-shelf*) upakovanog softvera je metod pribavljanja koji podrazumeva direktnu kupovinu unapred napisanih aplikacija koje koriste mnoge kompanije.

Softver sa police [gotov softver] je tako pisan da omogući širu funkcionalnost kako bi se zadovoljio širok opseg pojedinaca, namena i poslova.

Neki softveri mogu se i doslovce kupiti sa police “off-the-shelf” (n.p. PC World).
Drugi “off-the-shelf” softver je: kupljen sa Weba i navučen uz pomoć licenciranog ključa dobijenog od dilera (VAR – Value Added Reseller) koji će mu “dodati vrednost” dajući savete o neophodnom hardveru, pomagajući instaliranje i obučavanjem korisnika.

KUPI GA: SA POLICE [OFF- THE-SHELF]

Poboljšanja

- Cena posedovanja se deli sa drugim kompanijama
- Manje nedostataka

Nedostaci

- Može posedovati suviše mnogo funkcija
- Može zahtevati od poslovanja i zaposlenih da promene način na koji rade
- Manje je lako razlikovati se od konkurencije – svako od njih ima isti sistem.

IZGRADI GA: KROJENJE

Krojenje [Bespoke] – Urediti (dobra, specijalno odeću); trgovci, krojači itd.

Novi PIS razvijen po nacrtu od strane tima profesionalaca za informacione sisteme.

IS profesionalci koji će ili raditi u okviru posla, i u tom slučaju to označavamo ‘**kućni** ’ razvoj nacrta, ili za kuću izvan kao što su softverske kuće, i u tom slučaju kažemo da je razvoj softvera **‘izmešten’**

Prednosti: *Skrojen po egzaktnim zahtevima organizacije*

Nedostaci: *Veliki troškovi*
Zahteva vreme
Kvalitet

IZGRADI GA: RAZVOJ ZA KRAJNJEG KORISNIKA

Softver razvijen za krajnjeg korisnika je softver napisan od strane ljudi koji nisu profesionalci u informacionim sistemima, t.j. Korisnici poslova.

Aplikacije krajnjeg korisnika su mnogo više ograničene po širini – manja širina onemogućuje zadovoljenje lokalnih potreba.

Aplikacije mogu biti vezane za odeljenje ili personal po prirodi i usmereni su ka izlazu i izveštajima a manje prema ulaznim pobudama.

***Senn* (1989)** je procenio da 50 do 75 procenata svih računarskih aplikacija mogu biti klasifikovane kao aplikacije krajnjeg korisnika (za razliku od institucionalnih aplikacija) pa je stoga veći broj ovih sistema razvijen od krajnjih korisnika (t.j. Ne od IT profesionalaca).

IZGRADI GA: RAZVOJ ZA KRAJNJEG KORISNIKA

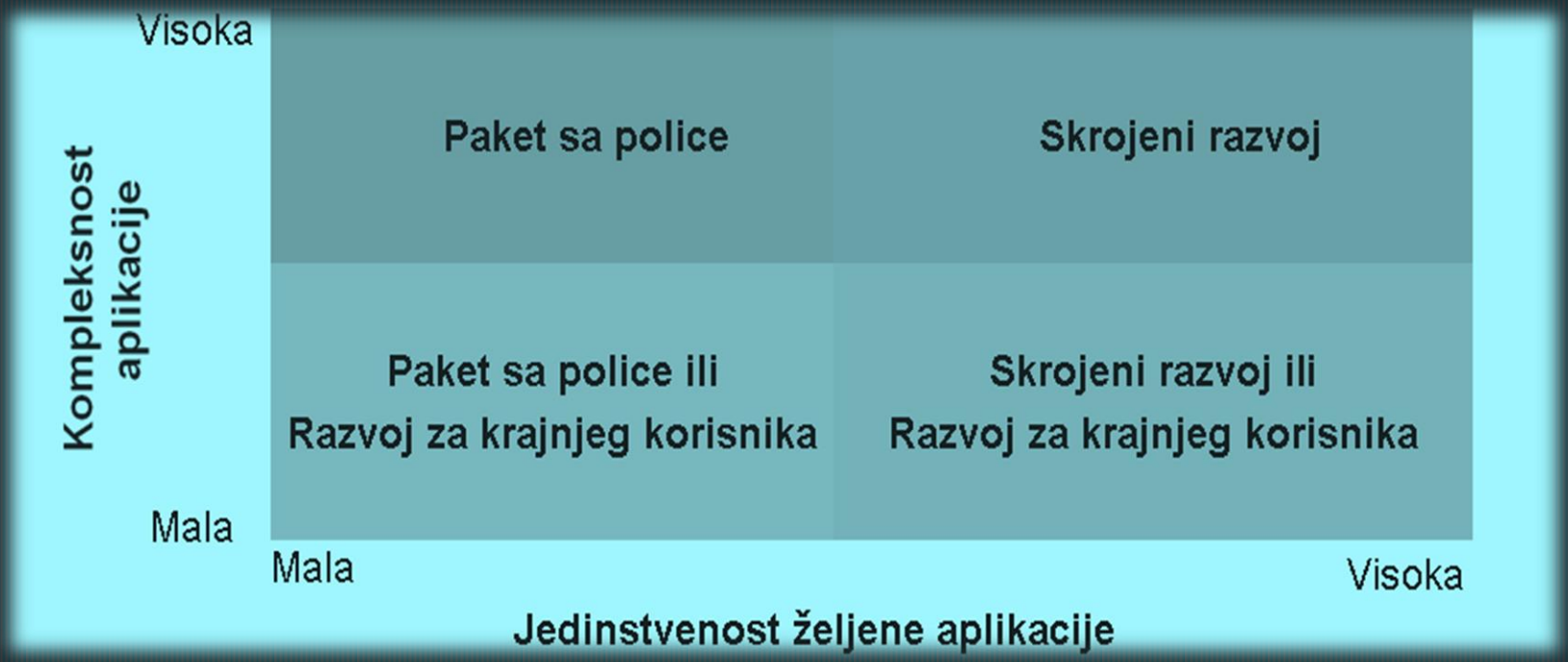
Prednosti:

1. Korisnici dobro prepoznaju svoje potrebe;
2. Alati ponekad nisu potrebni (postojeći softver postoji i drugde).

Nedostaci:

1. Softver se ne može dobro testirati i dokumentovati;
2. Često je zavisan od broja zaposlenih (sistem se ne može održati kada zaposleni napuste firmu).

KAKO IZABRATI

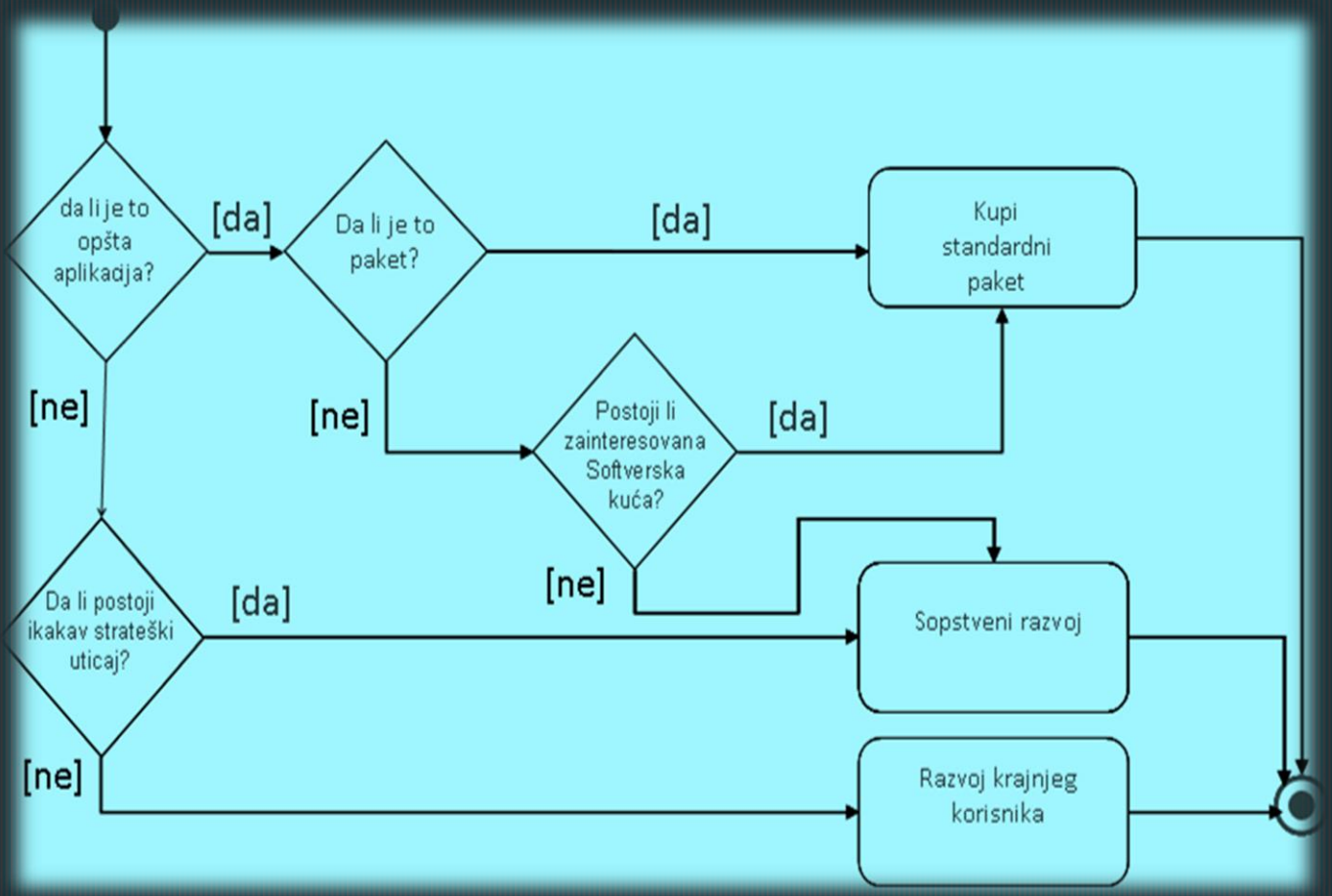


KAKO IZABRATI

Ostali faktori koji utuču na pribavljanje softvera uključuju:

1. Veličinu organizacije
2. Sopstvenu IS/IT ekspertizu.
3. Kompleksnost potrebnog informacionog sistema.
4. Jedinstvenost posla ili oblasti posla koje treba podržati.
5. IS/IT ekspertiza u odnosu na krajnjeg korisnika.
6. Povezanost sa postojećim softverskim aplikacijama.

KAKO IZABRATI



NABAVKA APLIKATIVNOG SOFTVERA

***Posao** može izabrati da izgradi ili kupi aplikativni softver. Boljitak kupovine uključuje:*

- **Jednostavnost korišćenja.** Softver može biti jednostavan za upotrebu, zahtevajući sasvim malo obučavanje pre nego što se paket koristi produktivno.
- ***Kompatibilnost.*** Softver može biti kompatibilan sa paketima koji se kporiste negde drugde u kompaniji , a posebno je korisno ako je kompatibilan sa paketima koji koriste kupci ili dobavljači. U dodatku, softver može biti osposobljen za korišćenje na kompanijskom hardveru.
, ***Konverzija podataka***. IMože biti sasvim jednostavno konvertovati kompaniske postojeće podatke za korišćenje sa softverom.
- ***Podrška proizvodu***.Softver može biti dobro podržan od dobavljača. Kvalitet i nivo ponašenja podrške treba da je primerena zahtevima kompanije.
- ***Podrška***. Softver treba da je podržan sa tri strane. N a primer, obuka i materijal za obuku treba da je široko raspoloživ iz većeg broja izvora.
- ***Cena***. Cena softvera treba da je porediva sa cenom paketa od drugih dobavljača.
- ***Reputacija***. Ako je moguće, kompanija treba da je uverena da kupljeni softver uživa dobru reputaciju sa stanovišta pouzdanosti i efikasnosti.

GENERIČKI BOLJITAK APLIKATIVNOG SOFTVERA

Paketi aplikativnog softvera imaju nekoliko opštih prednosti u odnosu na ručne ili poluručne alternative:

• ***Vremenske uštede***. Na primer, mnogi programi pomažu korisnicima da unose podatke mnogo brže obezbeđujući unos listi ili kompletiranje unosa automatski.

• ***Poboljšana tačnost***. Na primer, mnogi programi pomažu korisniku da izbegne unos invalidnih podataka . Pored toga, neki programi (kao što su računovodstveni paketi) su u stanju da identifikuju greške ili pretivurečnosti u podacima u podacima.

• ***Automatizacija rutinskih*** *poslova*. Na primer, automatsko obračunavanje PDVa po računima ili generisanje automatske listekorisnika koje treba pozvati telefonom.

• ***Kompatibilnost***. Paket aplikativnog softvera treba da je kompatibilan sa drugim aplikativnim softverima koji su nabavljeni od istog isporučioca. Na primer Sage Platni spisak je kompatibilan sa Sage računovodstvom – postoji sobro podržan interfejs između njih i kada se plate isplate u računovodstvu se to automatski beleži.

• ***Interfejsi***. Mnogi softverski paketi podržavaju generički interfejse (većina od njih zasnovanih na XML) i vezuju ih na desktop softver. Na primer računovođa može da izvuče upravljačke informacije iz Sage, kopira ih u Excel i nacrta kružnu raspodelu.

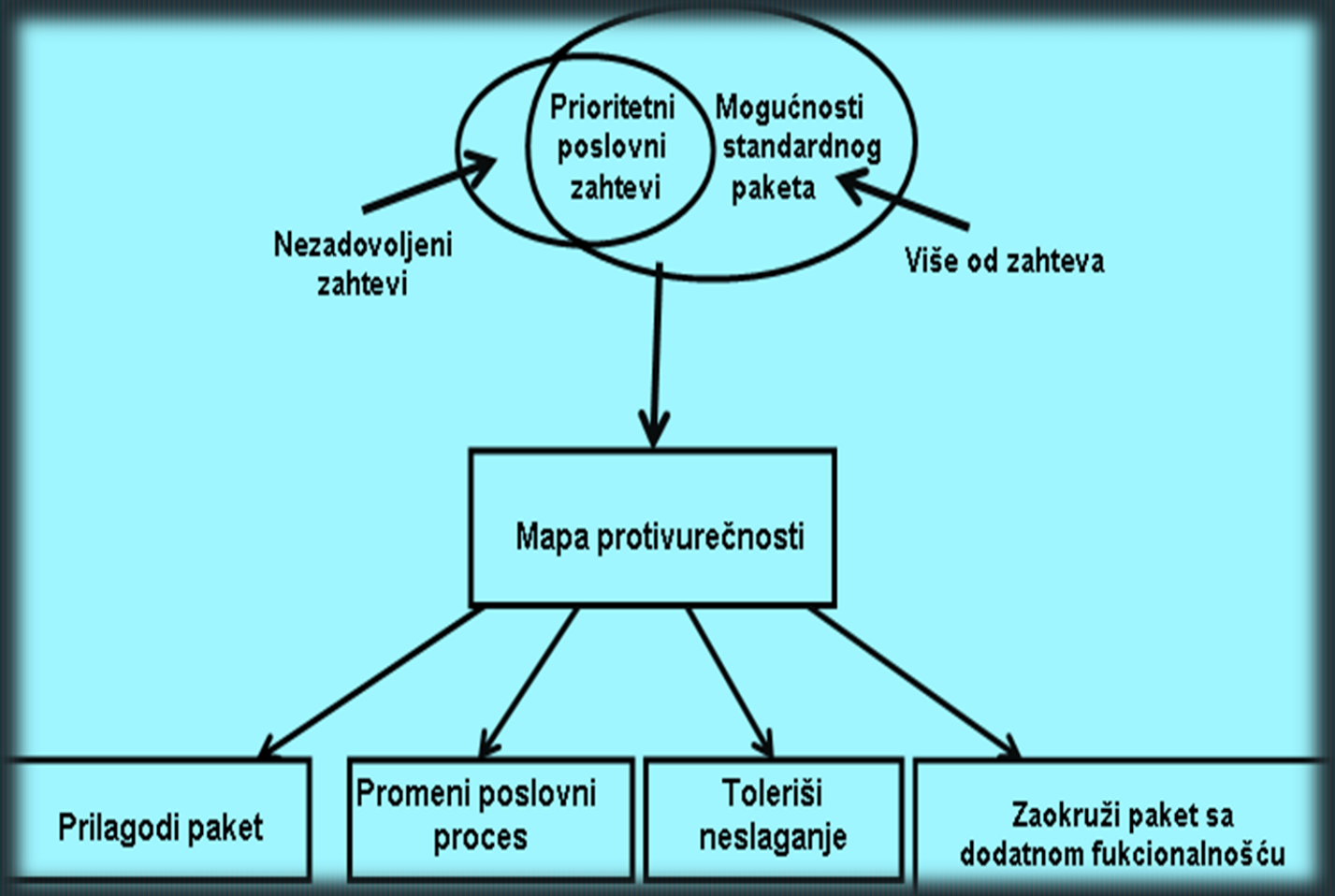
KUPOVINA SOFTVERSKIH PAKETA

Najuspešnije firme se trude da koriste vodeće tržište softverskih paketa.
Ako potrebe preduzeća nisu zadovoljenje koji paket onda to preduzeće može da podrži…

Menjaj Organizaciju a NE Softver!

Organizacije teško donose ovakve odluke.
Takva odluka traži prepoznavanje tekuće prakse kao one koje nije najbolja praksa i da ona zahteva da bude promenjena.
Na sreću raste saznanje među poslovnim svetom da softverskim paketima treba prilagoditi organizaciju

KUPOVINA SOFTVERSKIH PAKETA



KUPOVINA SOFTVERSKIH PAKETA

Razumevanje ukupne cene vlasništva

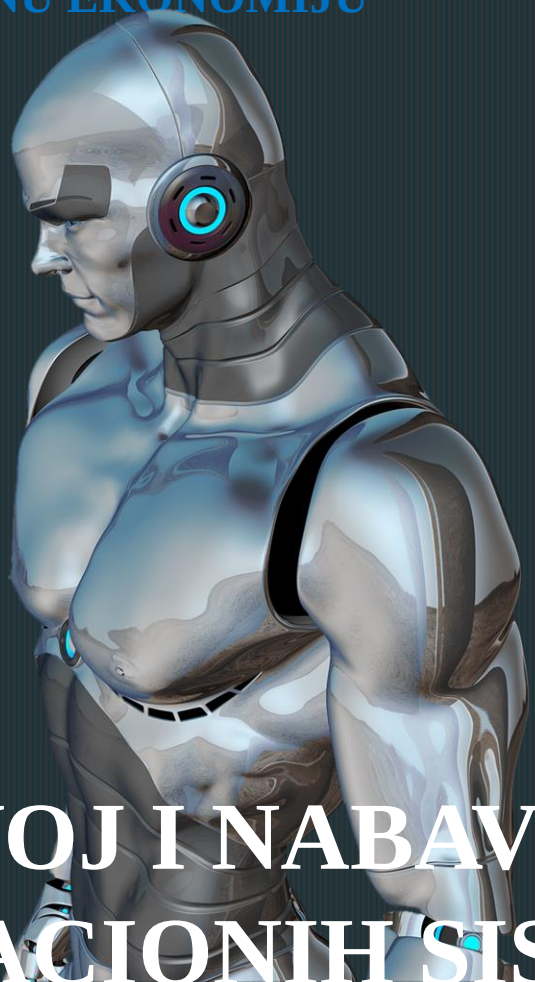
Cena vlasništva softvera uključuje

- Radno vreme zaposlenih koji rešavaju tehničke probleme
- Troškovi projekta instaliranja i usvajanja softvera
- Vežbanje
- Neugodosti i lomovi koji su posledica pogrešnog softvera

Većina poslova će:

a) Sklopiti ugovor sa dobavljačem softvera radi podrške a on uključuje pomoć na radnim mestima, pomoć pri pristupu sajtovima, podrška udaljenom pristupu, besplatna poboljšanja u softveru, a to iznosi 25% nabavne cene po godini.

b) Zaposliti svoj sopstveni IS tim za podršku koji će se baviti manjim izmenama ,privikavanjem na softver,izveštajima i interfejsima, baviće se korisničkim zahtevima i obučavanjem novih korisnika upravljaće nadgradnom , podržavaće hardver, softver itd.



RAZVOJ I NABAVKA INFORMACIONIH SISTEMA

Prof. dr Saša Ivanov
sasa.ivanov@fmz.edu.rs