P12 | MODELI PROCESA RAZVOJA IS

UVOD

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.

UVOD

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."

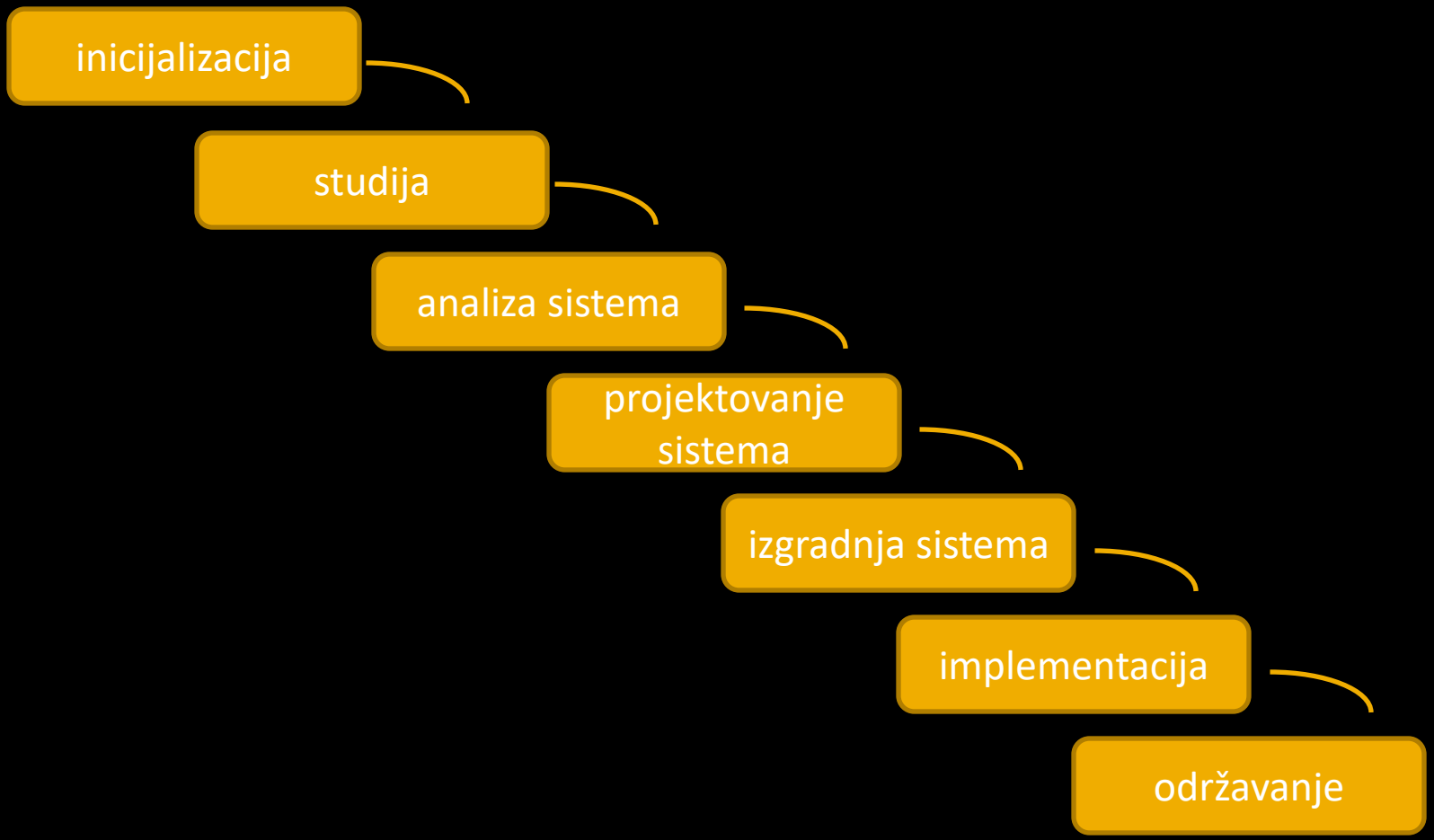
AD-HOC

"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:

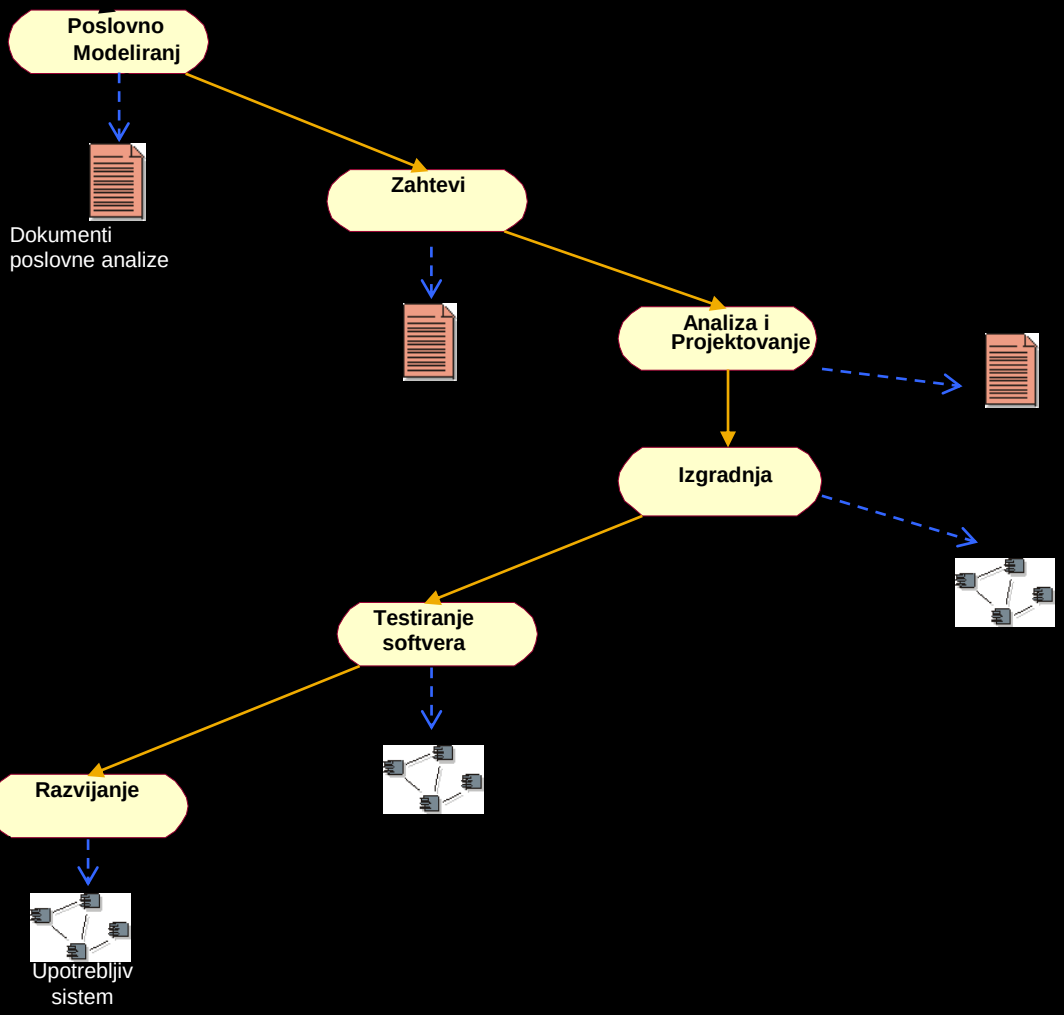
- istinski projekti retko slede proceduru koju model predlaže
- na početku većine projekata često postoji puno neizvesnosti vezanih za zahteve i ciljeve i zbog toga je korisnicima teško da detaljno identifikuju ove kriterijume. Model vodopada takođe ne prolagođava dobro ovu prirodnu neizvesnost.
- razvijanje 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.

V-model


Poslovno Modeliranj
Dokumenti poslovne analize
Zahtevi
Analiza i Projektovanje
Izgradnja
Testiranje softvera
Razvijanje
Upotrebljiv sistem

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).

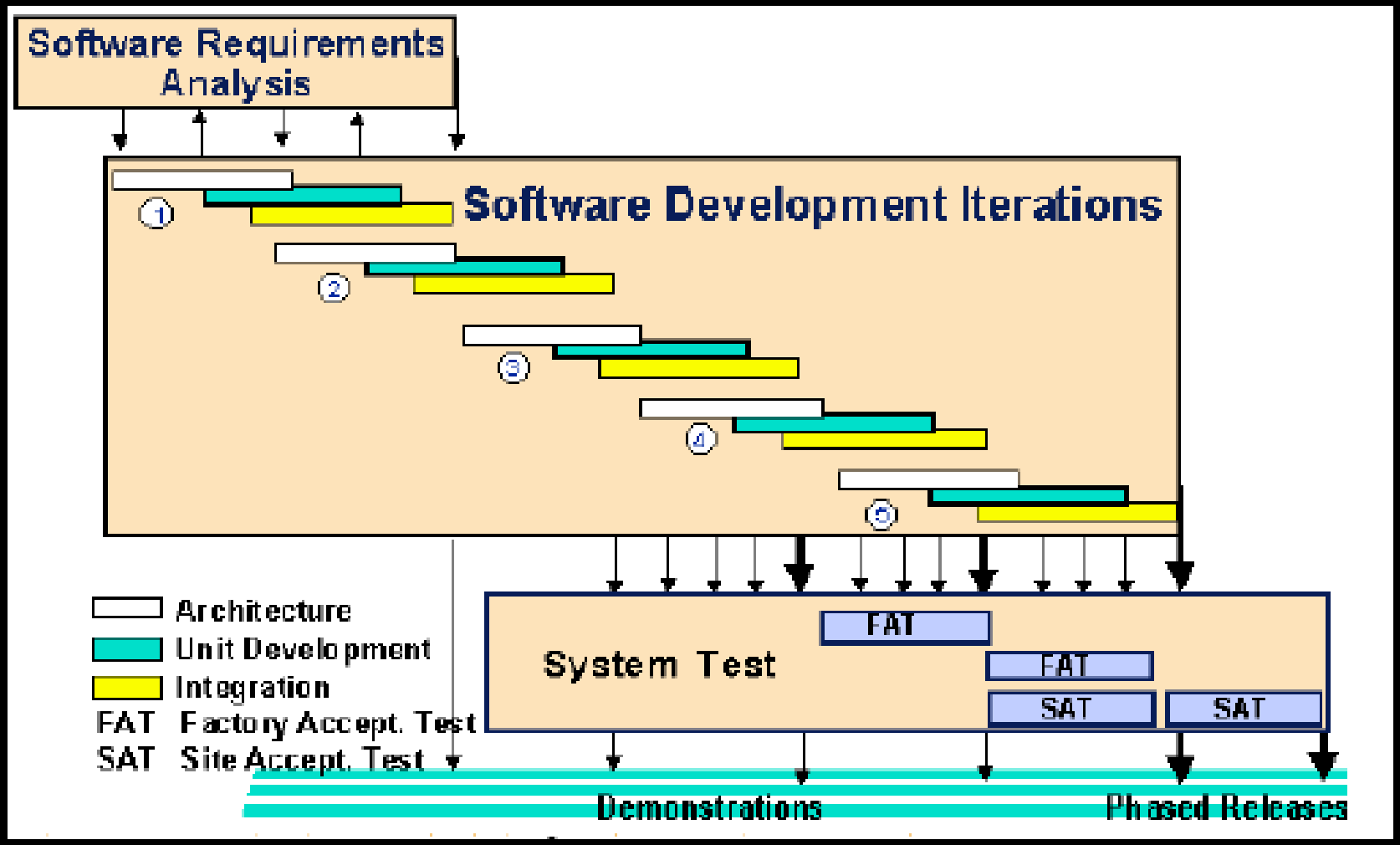
ITERATIVNI RAZVOJ

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

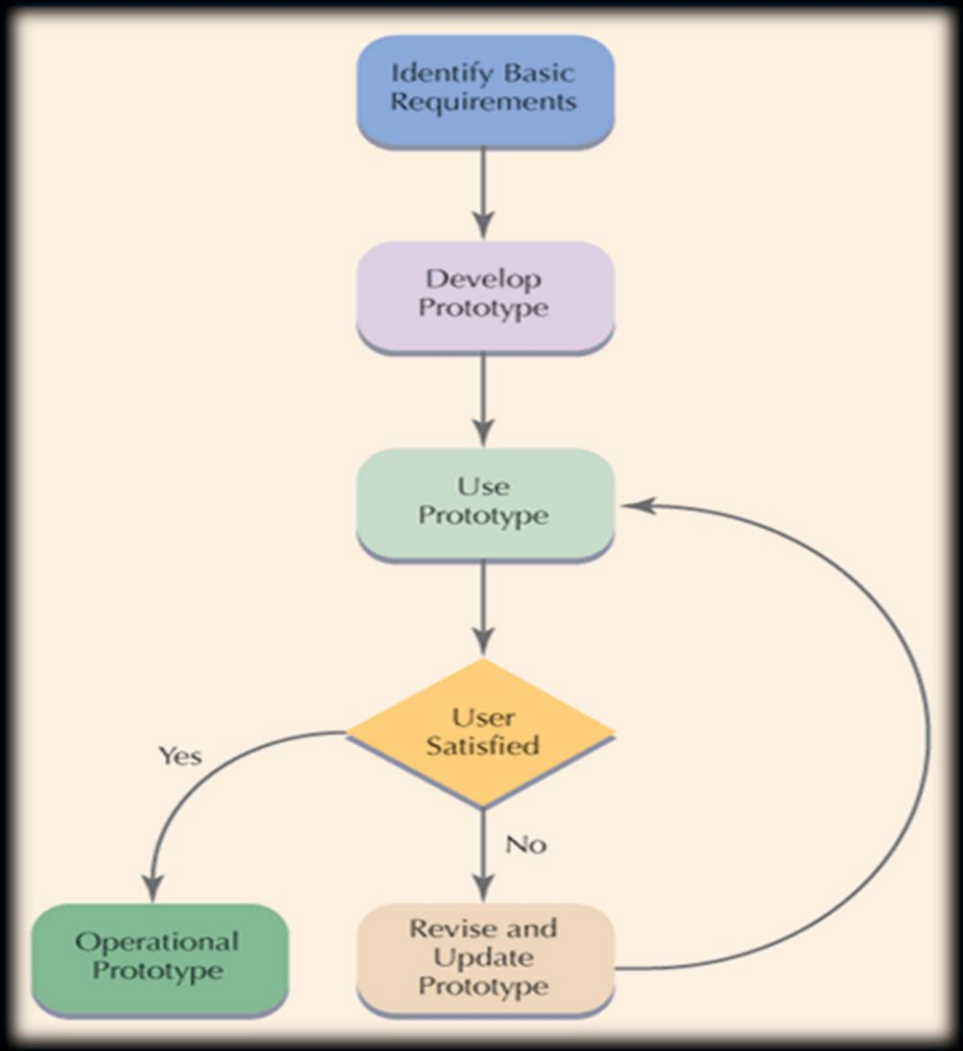
Izvestan broj procesa modela su razvijeni iz iterativnog pristupa. Sve ove metode proizvode neke softverske proizvode koje je moguće pokazati rano u procesu da bi se došlo do vredne povratne informacije od strane korisnika sistema ili drugih članova projektnog tima. Neke od ovih metoda su opisane u daljem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

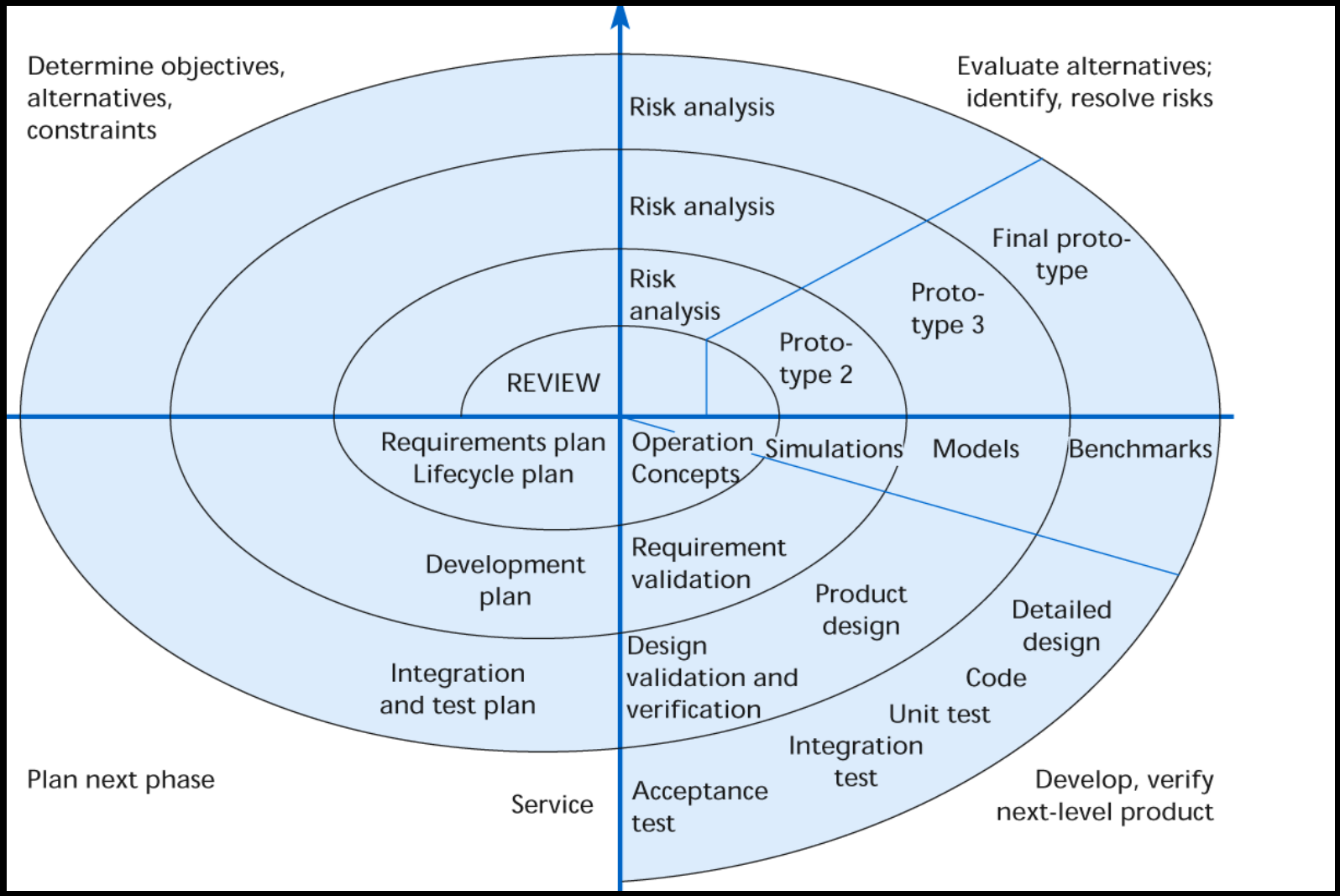
Ciljevi projekta. Ovo je slično fazi konceptualizacije sistema u modelu vodopada. Određuju se ciljevi, identifikuju se moguće prepreke i alternativni pristupi.

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.