Predavanje 2

Osnovni koncepti i metodologije razvoja informacionih sistema

Projektovanje informacionih sistema

Prilikom projektovanja informacionih sistema je potreban metodološki pristup. Informacioni sistem koji nastaje treba da korisniku pruži informacije koje su:

- **Relevantne** – Pre nego što se krene sa razvojem informacionog sistema treba izvršiti analizu procesa upotrebe informacija i doneti odluku koje su informacije relevantne za korisnika, a koje to nisu. Informacija je relevantna u toku procesa donošenja odluke, ako njen sadržaj može da utiče na samu odluku.
- **Tačne** – Informacija treba da bude onoliko tačna koliko je to potrebno. Ponekad je prihvatljivo da informacija bude u određenoj meri netačna, posebno ako je tačnost informacije u direktnoj vezi sa poskupljenjem dobijanja informacije, bilo u pogledu vremena bilo u pogledu novca.
- **Došle na vreme** – Informacija mora da do korisnika stigne u definisanom vremenskom intervalu, u kome će biti od koristi. Zakasnele informacije nisu korisne. Ponekad je bolje brže dobiti informaciju, nego čekati da ona bude apsolutno tačna.
- **Usmerena** – Informacija mora da stigne do pravog korisnika.
- **U pravom obliku** – Način na koji se informacija prikazuje korisniku utiče na njegovu efikasnost. Umesto da se prave tradicionalni tekstualni izveštaji, treba težiti grafičkom prikazu informacija.
- **Interaktivne prirode** – Korisnik treba da dobije onoliko informacija koliko mu je potrebno. Tek kada zatraži dodatnu informaciju treba mu je pružiti. Ne treba ga unapred bombardovati svim informacijama.
- **Konstrolisane** – Neke informacije su osetljive i mogu biti od koristi konkurenciji. U skladu sa time moraju se preduzeti odgovarajući koraci koji će obezbediti sigurnost informacija.

Pristupi projektovanju informacionih sistema

Prilikom projektovanja informacionih sistema mogu da postoje različiti pristupi. Neki od njih su:

- **Kupovina softverskog proizvoda** – Ovo je pristup koji se koristio na početku priče o informacionim sistemima. Težište kod ovakvog pristupa je razvoj računarskog sistema koji će zameniti svu papirologiju, koja je ranije obavljana ručno. Kod ovakvog pristupa obično nema velike analize zahteva sistema. Informacije koje ovakav sistem daje su obično u obliku izveštaja, sa puno podataka, u kojima je teško snaći se i u kojima je teško izdvojiti pravu informaciju.
- **Iz početka** – Ovo je pristup nastao kao odgovor na nedostatke prethodnog pristupa. Kao što i ime pokazuje kod ovog pristupa se razvoju informacionog sistema pristupa od nule.
- **Ključne informacije** – Kod ovog pristupa se pretpostavlja da su neke informacije od ključnog značaja za uspešno funkcionisanje sistema. To na primer, mogu biti ukupna

gotovina kojom preduzeće raspolaže, nivo pražnjenja zaliha ili odnos dobijenog i uloženog. Razvoj informacionog sistema se odvija tako što se pronađu te ključne informacije, a onda se informacioni sistem pravi tako da se te informacije mogu uvek dobiti.

- **Kompletna studija** – Ovaj pristup se zasniva na kompletnom proučavanju procesa rada i poređenju trenutne situacije sa onom koja će nastati posle uvođenja informacionog sistema. Postupak se obično sastoji u razgovoru sa svima koji učestvuju u procesu i određivanju informacija koje su za njih bitne. Ovakav način rada može biti razumljiv i koristan u otkrivanju nedostataka postojećih sistema. Sa druge strane, može biti jako skupo intervjuisati sve ljude, a sve to može dovesti do prikupljanja velike količine podataka, koje nije lako analizirati.

Na osnovu onog što je izloženo može se zaključiti da kod projektovanja informacionih sistema ne postoji pristup koji je univerzalno prihvaćen. Ni pristupi koji su ovde pomenuti nisu jedini niti najbolji. Postoji puno različitih metoda, odkojih svaki ima svoje prednosti i nedostatke. Ono što se može zaključiti je da su tehnička pitanja u ovom slučaju od sekundarnog značaja. Primarni cilj kod projektovanja je definisati koje su informacije potrebne i na osnovu toga definisati zahteve. Ako to niste dobro uradili ni informacioni sistem ne može biti dobar.

Strategija i planiranje informacionih sistema

Iz prethodne priče se može zaključiti da uvođenje informacionog sistema zahteva:

- Angažovanje velikih resursa
- Prepoznavanje činjenice da će, kao rezultat uvođenja informacionog sistema, aktivnosti preduzeća da se promene
- Nameru da se rad organizacije poboljša ili da se poveća profit preduzeća

Ovako važne odluke se ne mogu doneti bez odgovarajućeg strateškog planiranja. Osnovno stvar koju ovde treba imati na umu je da strategija razvoja informacionog sistema proizilazi iz strategije razvoja preduzeća, a ne iz tehnologije koja se koristi.

Strateško planiranje preduzeća

Ni u ovoj oblasti ne postoji opšte prihvaćen put koji treba slediti. Sa druge strane, ipak postoje određeni koraci koje treba slediti prilikom definisanja strategije razvoja preduzeća.

Većina velikih preduzeća već ima formulisanu strategiju razvoja. Ove strategije se obično odnose na period od pet godina. Ovo je optimalan vremenski period. Ako bi bio duži onda bi neizvesnost koju nosi budućnost dovela do toga da strateški plan nema neku veliku vrednost. Ako bi period bio kraći ne preduzeće bi ni stiglo da isplanira sve što je potrebno. U određenim situacijama vreme za koje se planira ima smisla produžiti. To je slučaj sa odbranom zemlje ili nuklearnom industrijom na primer. U okruženjima koja su dinamička i koja se brzo menjaju se taj period može i smanjiti (na primer visoka tehnologija).

Kod definisanja strategije razvoja prvo treba odlučiti koja je misija i cilj preduzeća. Na primer, za preduzeće iz oblasti hemijske industrije, misija bi mogla biti „da postane glavni snabdevač agrohemijskim proizvodima za poljoprivrednike kroz razvoj novih efikasnijih sredstava“.

Ciljevi preduzeća treba da podrže njegovu misiju. Svaki cilj treba da bude merljiv, na osnovu čega se određuje da li je preduzeće ispunilo cilj. Na primer, cilj bi mogao biti povećanje prodaje za 35% u roku od naredne tri godine.

Na osnovu ciljeva treba odrediti koje su buduće performanse odnosno potrebni kapaciteti za ispunjenje ciljeva. Ako postoji raskorak, onda strategija treba da predvidi i kako popuniti tu prazninu.

Na kraju treba formirati strategiju razvoja, koja u suštini predstavlja niz planova koje treba izvršiti da bi s ostvarili projektovani ciljevi. Ti planovi mogu da se odnose na nove projekte ili na nastavak rada na postojećim aktivnostima.

Strategija razvoja informacionih sistema

Strategija razvoja informacionog sistema treba da definiše koji informacioni sistem treba da postoji da bi se ostvarili projektovani ciljevi. Pri tome se treba koncentrisati na to da se odredi koje su informacije potrebne i da se osigura da strategija razvoja informacionog sistema bude u skladu sa strategijom razvoja preduzeća.

Pored strategije razvoja informacionog sistema postoji i strategija upotrebe informacionih tehnologija. Ova strategija treba da definiše koja je tehnologija potrebna da bi se realizovala strategija razvoja informacionih sistema. Ovde se treba usresrediti na to kako da se obezbedi informacija, a ne na to koje su informacije potrebne. Ova strategija treba da definiše i kako će se upravljati informacijama i informacionim sistemom. Postavlja se pitanje koji je najbolji način za definisanje strategije razvoja informacionog sistema. Prilikom definisanja te strategije treba uzeti u obzir brzinu kojom se informacione tehnologije menjaju.

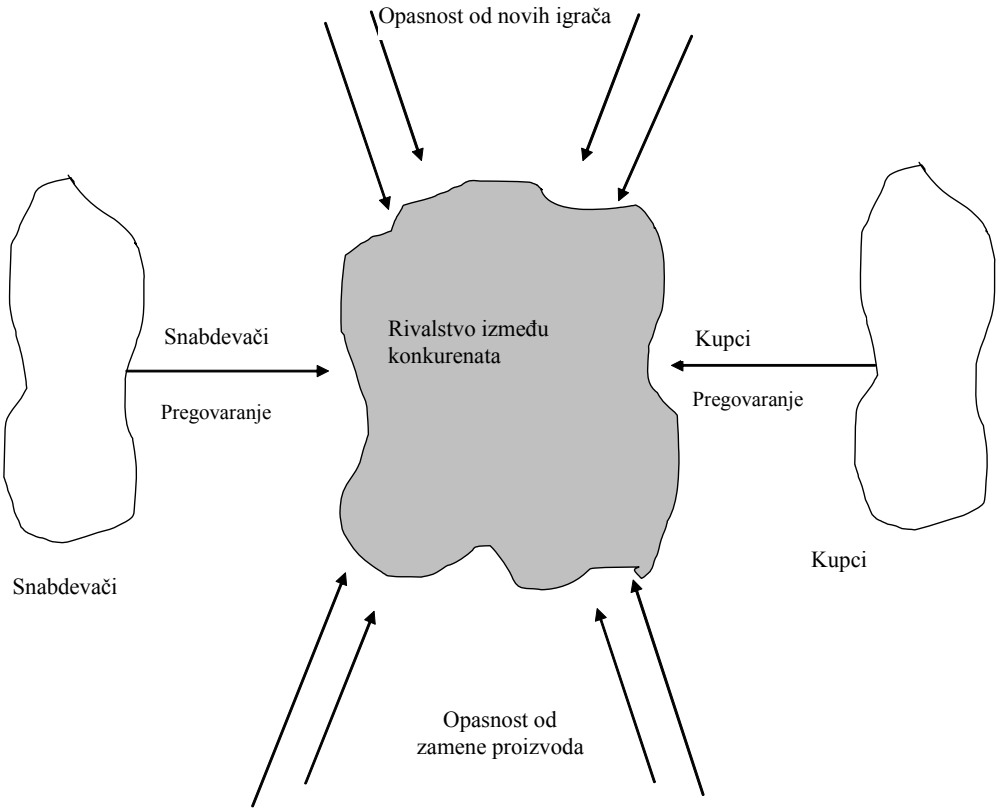
Kao pomoć prilikom definisanja stategije razvoja informacionog sistema koriste se različiti modeli. Ovi modeli koriste različite pristupe i ciljeve i odnose se na različite aspekte strategije. Neki modeli se više odnose na strategiju razvoja informacionog sistema, dok se drugi više odnose na strategiju upotrebe informacionih tehnologija. U okviru modela koji se koriste kod definisanja strategije razvoja informacionog sistema, postoji podela na modele koji se bave unutrašnjim aspektima preduzeća i one koji se bave spoljašnjim aspektima odnosno okruženjem.

Među modelima koji u obzir uzimaju spoljašnje uticaje na preduzeće je verovatno najpoznatiji model pet faktora uspešnosti, koji su predložili Porter i Millar. Po ovom modelu na uspešnost preduzeća najviše utiče način na koji se preduzeće bori sa pretnjama koje dolaze iz njegove okoline. Faktori su prikazani na sledećoj slici.

Snabdevači su ti koji treba da obezbede sirovine i komponente koje učestvuju u proizvodnom procesu. Oni mogu da utiču na rad preduzeća time što će povećati cenu svojih proizvoda ili usluga. To dolazi kao rezultat mogućnosti da se to proda konkurenciji. U interesu preduzeća je da učini što više da snabdevači ne požele da prodaju konkurenciji.

Jedan od načina da se to učini je održavanje dobrih veza sa snabdevačima, pomoću na primer, elektronske razmene podataka (EDI standard). EDI zahteva da postoji elektronska veza između snabdevača i preduzeća. Prednost ovakvog rada je u smanjenju vremena isporuke, smanjenoj papirologiji i tačnijim informacijama.

Kupci na preduzeće mogu da utiču time što će proizvod kupiti od konkurencije. Ova opasnost je veća ako ima malo kupaca i mnogo proizvođača. Jedan od načina da se na to utiče je da se kupac dovede u situaciju da ga prelaz na konkurentski proizvod košta, bilo u novcu bilo na neki drugi način. Zbog toga sa kupcima treba razvijati veze koje će dovesti do njihovog ostanka. To se može postići na primer, omogućavanjem kupovine preko računarskog sistema. Istraživanja u bankama su pokazala da se kupci nakon što se naviknu na neki sistem, relativno teško prelaze na neki drugi.



Slika. Pet faktora uspešnosti preduzeća

Proizvodi – zamene su oni proizvodi su slični, ali sa sobom donose i nešto novo. Uvek postoji opasnost da se izgube kupci ako se konkurencija pojavi sa proizvodom koji će bolje odgovoriti zahtevima kupaca. Informacione tehnologije mogu da pomognu da se spreči odlazak kupaca time što bi se uvela pomenuta cena prelaska na drugi proizvod, kao i upotrebom CAD/CAMM sistema da se brzo pređe na proizvodnju novih, poboljšanih proizvoda.

Novi igrači su nova preduzeća koja se javljaju na tržištu. Uvek postoji opasnost da se pojavi novo preduzeće koje će privući deo kupaca. Tradicionalan odgovor na ovakve izazove je postavljanje barijera novim preduzećima. Te barijere mogu biti zakonske (na primer, patenti), ili ekonomske (na primer, granska povezanost i sl.).

U uspostavljanju barijera mogu da pomognu i informacione tehnologije. Informacioni sistem može da poveća produktivnost u preduzeću, tako da i nova preduzeća moraju da ulažu u informacione sisteme, ako žele da uđu na tržište. To može da ih obeshrabri, ako je cena velika. Isto važi i za slučaj da je potreban skup CAD/CAMM softver.

Konkurencija. Osim u slučaju monopola svako prezeće ima konkurenciju. To je verovatno i najveća pretnja uspešnosti preduzeća. U skladu sa onim što je ranije pomenuto i ovde mogu da pomognu informacione tehnologije, bilo da je reč o uspostavljanju veza sa snabdevačima i kupcima, bilo investicijama u informacione sisteme koji povećavaju produktivnost.

Pored spoljašnjih faktora, koji se u obzir mogu uzeti i preko drugih modela (PEST analiza) na razvoj strategije razvoja informacionog sistema mogu da utiču i unutrašnji faktori. Unutrašnji faktori najčešće proizilaze iz faze rasta u kojoj se preduzeće nalazi. Tako na primer, po Nolanu postoji šest faza rasta preduzeća.

Početna faza se karakteriše upotrebom računara za obradu transakcija. Tu ima mnogo obrade podataka, a malo planiranja informacionog sistema. Korisnici ne brinu o tehnologiji koja se koristi. Nove aplikacije se razvijaju u tradicionalnim programskim jezicima (COBOL i sl.)

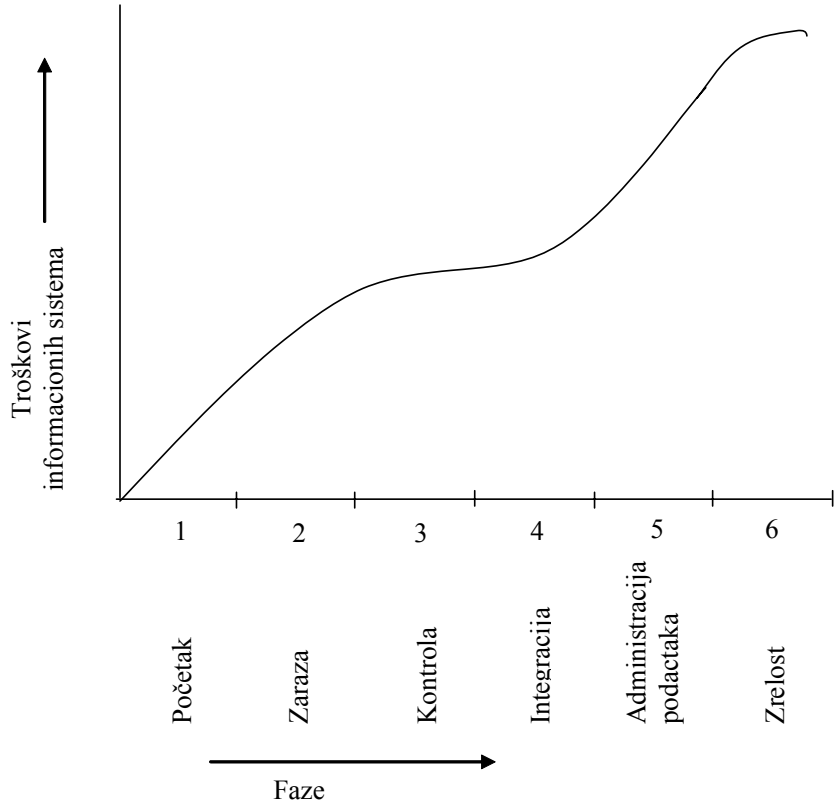
Druga faza je faza zaraze kada korisnici postaju svesni mogućnosti IT tehnologija, ali ne razumeju u potpunosti ograničenja i prednosti koje one donose. IT se generalno tretiraju kao trošak i nema provere kod korisnika da li su potrebne nove aplikacije. Javljaju se tehnički problemi vezani za razvoj. Mnogo se troši na održavanje sistema.

Treća faza je faza kontrole. Ovde se obično reorganizuje odeljenje za obradu podataka. Nekontrolisani rast projekata vezanih za informacione tehnologije se kontroliše ispostavom računa onima koji to traže.

Četvrta faza je faza integracije. Nakon konsolidacije u trećoj fazi obrada podataka kreće u novom pravcu, odnosno više se orijentiše na pružanje informacija. Uvode se tehnologije baza podataka, a u svim odeljenjima se primećuje napredak u korišćenju informacionih tehnologija. Dolazi do napretka u razvoju novih aplikacija. Sada se javlja problem duplikacije podataka.

Peta faza je faza administriranja podataka. Odgovor na fazu 4 je uvođenje kontrole nad podacima. Sada se sagledava da podaci predstavljaju kapital preduzeća. Razvija se integrisana baza podataka koja opslužuje celu organizaciju. Razvijaju se aplikacije koje kao osnovu koriste tu jedinstvenu bazu podataka.

Poslednja faza je zrelost preduzeća. Informacioni sistem je sastavni deo preduzeća. Aplikacije odslikavaju organizacione aktivnosti. Strukture pdoataka su postale model pdoataka. Sada je jasno da su informacije od strateškog značaja. Menadžer informacionog sistema dobija svoje mesto u hijerarhiji i to mesto je u istom rangu sa mestom finansijskog direktora.



Slika. Nolanov model sa šest faza rasta preduzeća

Kao što se vidi iz prethodnog na planiranje strategije razvoja informacionog sistema utiču i unutrašnji i spoljašnji faktori. Rezultat treba da bude strategija koja uzima u obzir sve ove faktore i koja će na kraju preduzeću doneti najviše koristi.

Međusobni odnos strategije razvoja preduzeća i strategije razvoja informacionog sistema

U početku se smatralo da je strategija razvoja preduzeća prethodi strategiji razvoja informacionog sistema. Pokretačkom snagom razvoja je smarana poslovna strategija, dok je strategija razvoja informacionog sistema bila tu da pomogne da se ostvari poslovna strategija. Vremenom su informacioni sistemi postali sastavni deo rada preduzeća, tako da je postalo jasno da strategija razvoja informacionog sistema sastavna i integrisana komponenta celokupne strategije razvoja preduzeća.

Razvoj interneta i elektronskog poslovanja su uneli nove promene. Danas je postalo jasno da informacione tehnologije mogu da budu pokretačka snaga celog preduzeća. Pojavom interneta su se javile nove opasnosti, ali i nove mogućnosti. U određenim situacijama preduzeća su bila prinuđena da revidiraju svoju celokupnu strategiju i da razmotre nove zahteve iz oblasti informacionih sistema.

U oblasti razvoja informacionih sistema su se u poslednje vreme iskristalistali određeni pravci. Ovde će se pomenuti ERP sistemi i tehnika nazvana BPR (reinženjering poslovnog procesa).

ERP (Enterprise resource planning – planiranje resursa preduzeća)

U početku se razvoj informacionih sistema uglavnom svodio na različite oblasti rada, često segmentirano i neusaglašeno. Poslednjih godina se više pažnje posvećuje integrisanim strategijama. Jedna od najpopularnijih strategija jeste upotreba ERP softvera. Ovakvi sistemi pokušavaju da modeliraju kompletno preduzeće. Smatra se da to vodi efikasnijoj deobi informacija i samim tim efikasnijem radu celog sistema.

Danas postoje paketi koji se koriste za različite poslovne funkcije, kao što su proizvodnja, distribucija, marketing, prodaja i sl. Prednosti kupovine ovakvih programa, odnosno kupovine jedinstvenog sistema su:

- Radi se samo sa jednim proizvođačem softvera
- Manja je cena održavanja
- Celo preduzeće koristi jedinstvenu strategiju

Pored prednosti uvođenje ERP sistema ima i svojih nedostataka.

- Visoka početna cena i visoka cena održavanja
- Ponekad je potrebno promeniti rad preduzeća, da bi se uskladilo sa ERP sistemom.
- Sistemi nisu fleksibilni, ako dođe do promena.

Reinženjering poslovnog procesa

Ova strategija traži fundamentalno preispitivanje poslovnog procesa. Ona može da dovede i do dramatičnih poboljšanja, time što će se promeniti osnovne aktivnosti.

Jedna od razloga pojave ove strategije je neuspeh preduzeća sa strogo definisanom funkcionalnom strukturom. U takvim okruženjima se inovacije i fleksibilnost često guše. Reinženjering poslovnog procesa obuhvata preispitivanje osnovnih poslovnih procesa i njihovu analizu od početka do

kraja. Tom prilikom se zanemaruje postojeća struktura i pokušava se identifikacija ključnih karakteristika poslovnog procesa.

Na primer, ako se razmatra prodaja, ona može da teče od inicijalnog zahteva kupca, preko narudžbe, isporuke i održavanja. U celom procesu mogu da učestvuju različita odeljenja i osobe. Analizom tog procesa može se na primer, zaključiti zašto narudžbe kasne, zašto se greši sa isporukom ili zašto održavanje ne zadovoljava.

I ova strategija ima svoje prednosti i nedostatke. Prednost je svakako da ukoliko se uradi dobro, može da preduzeću donese značajne uštede i poboljšanja. Nedostatak je u tome što je praksa pokazala da se ovaj reinženjering često ne radi dobro, tako može dovesti i do propasti perduzeća. Ispitivanja su pokazala da je samo 30% svih procesa uspešno redizajnirano.

Modeliranje informacionih sistema

Razvoj informacionih sistema se odvija na osnovu različitih modela.

Realni procesi i objekti su obično suviše složeni da bi ljudski um mogao da ih bez problema analizira i prihvata. Zbog toga se kreiraju apstrakcije, koje u suštini predstavljaju idealizovane predstave realnih procesa i objekata. Apstrakcija zanemaruje sve ono što nije od primarnog značaja za odvijanje procesa ili opis samog objekta.

U računarskoj praksi se za opis različitih pojava, procesa i sistema koriste modeli. Osnovna namena modela je da se realni objekti i događaji pojednostave u dovoljnoj meri, tako da se osnovni objekat, koji se modelira, može bolje razumeti.

Proces nastanka i upotrebe modela se naziva modeliranjem. Ako se govori o razvoju informacionih sistema, onda se mogu definisati modeli razvoja tih sistema. Osnovna namena tih modela je da se omogući definisanje strukture i ponašanja sistema, da se definiše obrazac po kome se može raditi na razvoju sistema, da se omogući lakše dokumentovanje svih odluka koje se donose u procesu razvoja i da se omogući lakše vizuelno predstavljanje sistema.

Proces modeliranja se najčešće odvija tako što se na osnovu opisa funkcija sistema najpre pravi logički model sistema. U okviru logičkog modela se definišu svi bitni objekti i funkcije sistema, ali se pri tome ne uzimaju u obzir karakteristike softvera i hardvera koji se koriste ili će se koristiti.

Na osnovu logičkog modela se pravi fizički model sistema koji opisuje konkretne programe koji će se koristiti, kao i hardversku opremu na kojoj će ti programi raditi.

Softversko inženjerstvo

Razvoj novih programa se ubrzao krajem 60-ih godina prošlog veka. Osnovna pokretačka snaga je bila sve veća primena u američkoj vojsci i pojava novih generacija računara. U to vreme nisu postojale jasne smernice za pisanje softvera, definisanje zahteva i uspešno vođenje složenih projekata. Sistemi koji su u to vreme razvijani su generalno bili vrlo jednostavni.

Kreiranje sistema bez ikakvih ograničenja i pisanje programa po principu „samo neka radi“ doveli su do neuspeha velikog broja softverskih projekata. Pojavila se softverska kriza. Na konferenciji NATO iz 1968 su pokušali da nađu rešenje za nastale probleme, pa se tada prvi put javio termin softversko inženjerstvo. Rešenje za haos koji se javio u projektima vezanim za razvoj softvera se tražilo u primeni dokazanih metoda i principa iz oblasti mašinstva i građevine. Tu se odmah postavlja pitanje da li se razvoj softvera može upoređivati sa drugim inženjerskim disciplinama. Razlog je što se stanje softvera ne može naučno verifikovati ili izvesti na osnovu neke jednačine.

Zaključak je da ne postoje čvrsti principi za razvoj softvera. Najviše što se može dobiti su praktične preporuke (best practices). Praktične preporuke predstavljaju skup metoda, praksi, alata i procesa za koje se generalno smatra da su industrijski dokazani i optimalni.

Metodologije razvoja softvera

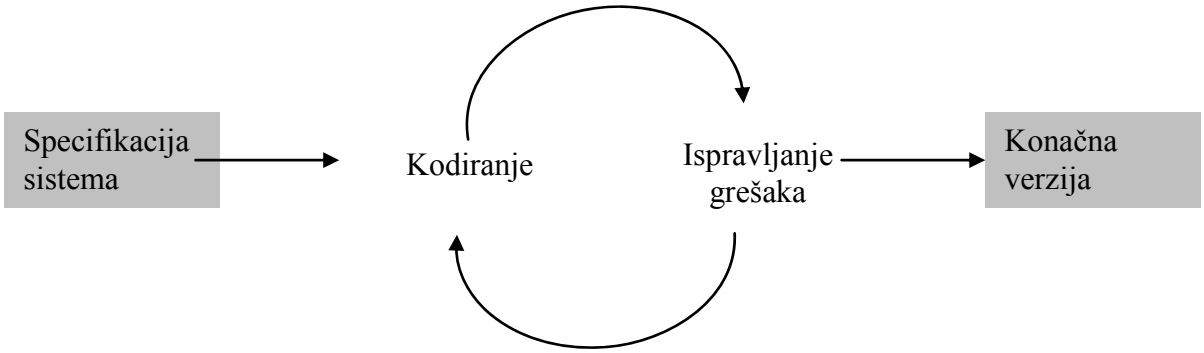
Metodlogija razvoja softvera predstavlja skup procedura, principa i alata koji pomažu programerima da naprave računarske sisteme.

U razvoju softvera postoje različite metode. Svaka od tih metoda ima svoje prednosti i nedostatke. U daljem tekstu će se opisati neke od najpoznatijih metoda za razvoj softvera.

Kodiraj pa popravljaj

Ovaj pristup i nije metodologija u pravom smislu reči, ali ga pominjemo pošto neki i danas rade na taj način. Ovakav pristup ponovo uspostavlja princip „samo neka radi".

Inicijalno, kupac može da navede šta je za njega najvažnije, ali to nije najvažnije kod razvoja. Ti njegovi navodi mogu biti u formi skice, elektronske poruke ili neke druge vrste slabe specifikacije. Razvoj se može zasnivati i na programerovom lokalnom ili ekspertskom znanju o poslu kupca i načinu njegovog rada. S vremena na vreme programer svoju aplikaciju demonstrira kupcima i dobija povratnu informaciju, nakon čega nastavlja rad. Sa sledeće slike se vidi da programeri najviše svog vremena provode u programiranju i ispravljanju grešaka.



Ovakav pristup ima negativne efekte, pa ga ne bi trebalo koristiti. Negativne strane su:

- Kvalitet proizvoda je mali
- Sistem često završava sa nekoordinisanom, zamrešnom zbrkom od koda.
- Sistemi koji se ovako razvijaju se teško proširuju i održavaju.
- Previše komplikovani sistemi obično imaju malu fleksibilnost.

Kaskadni pristup (Waterfall model)

Kaskadni model je primer modela kod kojeg se razvoj softvera zasniva na ideji o životnom ciklusu softvera. Ideja sa životnim ciklusom softvera je preuzeta od inženjera, koji su mnogo ranije uvideli da svi proizvodi imaju konačan vek upotrebe, koji počinje kreiranjem koncepta, nakon čega slede specifikacija, projektovanje, implementacija, održavanje i zastarelost. U softverskom inženjerstvu životni ciklus predstavlja niz različitih zadataka u kojima se mogu primeniti inženjerski metodi.

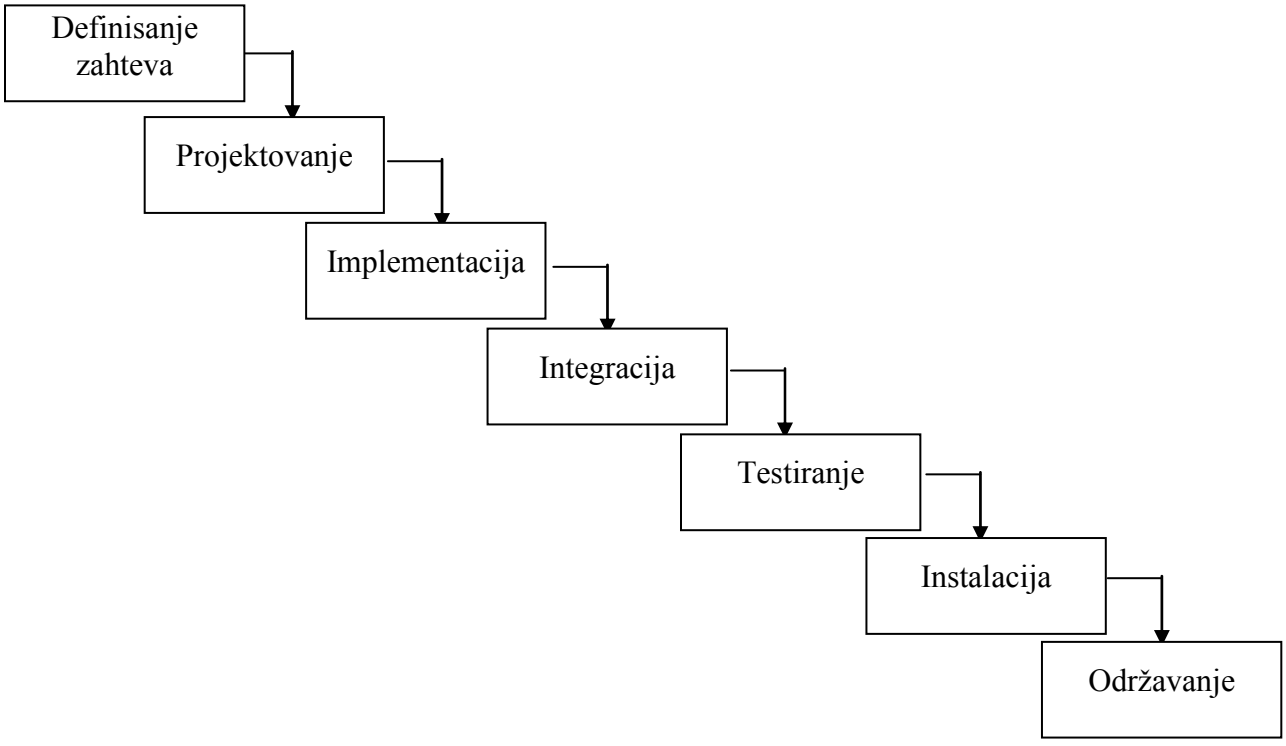
Kaskasdni model je sekvencijalni model razvoja softvera kod kojeg razvoj postojano teče naniže (kao vodopad – waterfall). Smatra se da ovaj termin proizilazi iz članka Rojsa, iz 1970-e godine, kojeg

svi navode i kao tvorca ovog modela, mada je on primer ovakvog modela dao kao nešto što ne treba koristiti.

Originalan kaskadni model je doživeo mnoge izmene i poboljšanja, ali ćemo ovde predstaviti kako je izgledao na početku. U originalnom modelu u razvoju softvera postoje sledeće faze.

1. Definisanje zahetva
2. Projektovanje
3. Implementacija
4. Integracija
5. Testiranje
6. Instalacija
7. Održavanje

U čistom kaskadnom modelu faze se odvijaju potpuno sekvencijalno. To znači da, dok se ne završi jedna faza, sledeća ne može da počne. Sve počinje definisanjem zahteva. Tom prilikom se zahtevi koji se predstavljaju pred novi informacioni sistem čvrsto definišu, tako da kasnije više nisu moguće promene. Tek kad se zahtevi kompletiraju, prelazi se na projektovanje. Projektuju se odgovori na sva pitanja koja se postavljaju pred programere i pravi se uputstvo koje oni treba da slede. Projekat je plan za implementaciju zahteva. Nakon što se završi projektovanje, programeri počinju sa implementacijom, odnosno pisanjem programskog koda. Nakon programiranja, se vrši integracija različitih softverskih komponente, koje su napravili različiti delovi tima. Tek nakon implementacije i integracije se vrši testiranje softvera. Ovde se ispravljaju sve greške koje su proistekle iz prethodnih faza. Nakon toga se program instalira i kasnije održava (u smislu dodavanja novih funkcija i popravke grešaka uočenih tokom eksploatacije).



Slika: Kaskadni model razvoja informacionih sistema

Kao što se vidi sa slike kaskadni model prelazi na sledeću fazu tek ako je prethodna faza u potpunosti završena i dovedena do savršenstva. Faze su ovde diskretne i nema preklapanja između njih, niti povratka unazad.

Prednosti kaskadnog modela

Vreme koje se potroši u početnoj fazi kreiranja softvera može kasnije da donese velike uštede. Mnogo puta je dokazano je mnogo jeftinije ako se greška pronađe u početnim fazama (definisanje zahteva i projektovanje). Uštede se odnose kako na novac, tako i na vreme i rad koji treba uložiti u ispravljanje grešaka. Neke procene pokazuju da greške u specifikaciji koje nisu primećene i koje se otkriju tek kod održavanja koštaju od 50 do 200 puta više, nego ako se otkriju u fazi definisanja zahteva. Ako na primer, postane nemoguće implementirati program zbog grešaka u projektu, mnogo je jeftinije popraviti projekat, nego da mesecima kasnije, nakon uloženog velikog rada i potrošenog novca, prilikom integracije komponenti ispravljamo ono što nije dobro.

Ovo je centralna ideja koja stoji iza kaskadnog modela i modela Projektovanja unapred. Vreme koje se provede na tome da se osigura da su zahtevi i projekat apsolutno tačni kasnije donosi uštede u vremenu i radu.

Još jedan argument koji se ističe kao dobra strana kaskadnog modela je njegovo insistiranje na dokumentaciji (dokumenti sa specifikacijom zahteva i dokumenti projekta), kao i na izvornom, programskom kodu. Većina „agilnih“ metodologija favorizuje kodiranje u odnosu na pisanje dokumentacije. Argument protiv ovakvog pristupa, pa samim tim i u korist kaskadnog modela jeste i da je znanje o projektu smešteno u u glavama članova tima. Ako neko iz tima ode, njihovo znanje je izgubljeno, tako da se može desiti je vrlo teško nastaviti sa projektom. Ako postoji potpuni dokument sa opisom onog što se radi, onda bi teoretski, novi članovi tima trebalo da budu u stanju da se uključe u projekat, čitanjem tih dokumenata. Agilne metodologije pokušavaju da se sa tim izbore na različite načine. Na primer, ekstremno programiranje propagira rotaciju članova tima, unutar projekta, tako da svi o svemu znaju sve.

Neki ljudi vole kaskadni model zbog njegovog disciplinovanog i jednostavnog pristupa. Umesto haosa koji je vladao pre ovog modela, sada se koristi struktuisani pristup. Model napreduje po fazama, svaka faza je jasna i dokumentovana, tako da se lako mogu definisati tačke provere u procesu razvoja. Verovatno je i to razlog što se kaskadni model koristi u uvodnim predavanjima na mnogim kursevima koji se odnose na softversko inženjerstvo.

Kaskadni model, generalno, može biti primenljiv za softverske projekte koji su stabilni (posebno kod projekata kod kojih se zahtevi ne menjaju), kao i tamo gde je moguće i verovatno da projektanti u potpunosti predvide rad sistema i naprave ispravan projekat, pre nego što se počne sa implementacijom.

Nedostaci kaskadnog modela

Za kaskadni model se tvrdi da nije dobra ideja uglavnom zato što se misli da, za bilo koji ozbiljniji projekat, nije moguće u potpunosti završiti jednu fazu projekta, pre nego što se pređe na sledeću. Na primer, možda klijenti nisu u potpunosti svesni toga šta tačno žele, pre nego što vide radni prototip, koji mogu da komentarišu. Može se desiti da oni stalno menjaju svoje zahteve i nad time programeri i projektanti imaju malo ili nimalo kontrole. Ako klijent svoje zahteve promeni posle završetka faze projektovanja, onda se i projekat mora prilagoditi tome. Na taj način vreme provedeno u prthodnom projektovanju predstavlja uludo potrošeno vreme. To je i razlog što Agilni metodi propagiraju manje oslanjanje na fiksne, statičke dokumente.

Projektanti nisu (i ne mogu biti) svesni svih teškoća koje se mogu javiti u budućnosti kod implementacija. Možda tek u trenutku implementacije postane jasno da je neku funkciju programa

izuzetno teško ostvariti. U takvim slučajevima je bolje promeniti projekat, nego tvrdoglavno nastaviti sa prethodnim projektom, u pokušaju da se geške u projektu reše u fazi implementacije.

Osim u specijalnim slučajevima kada su oni koji definišu zahteve i projektuju informacioni sistem visoko kompetentni, vrlo je teško unapre sve što je potrebno za svaku fazu, pre nego što se izvesno vreme provede u samoj fazi. To znači da je potrebna povratna informacija iz narednih faza, da bi se uspešno završila prethodna. Projektant verovatno treba da proveri u praksi svoje koncepte. Kod kaskadnog modela se tvrdi da projektanti treba da imaju prethodno iskustvo u radu na drugim sličnim projektima, tako da su u stanju da tačno predvide problem.

Da bi se proverile faze, od projektovanja, preko implementacije do verifikacije, potrebno je testiranje. Stalno je potrebno imati neki prototip, da bi se osiguralo da zahtevi nisu kontradiktorni i da ih je moguće ostvariti. Da bi se osiguralo da implementacija ostaje na pravom putu, potrebno je stalno vršiti integraciju i provere. Oni koji zagovaraju upotrebu kaskadnog modela će odgovoriti da su stalni implementacija i testiranje potrebni samo ako se javljaju bagovi u programu. Ako programeri slede disciplinovan proces, onda bagova neće ni biti.

Vrlo često je potrebno da se inkrementalno softver postavlja kod klijenta, da bi se osiguralo da se programeri i klijenti razumeju.

Vrlo je teško u napred proceniti vreme i cenu svake faze procesa razvoja softvera, bez „prljanja ruku“, osim ako su oni koji prave te procene izuzetno iskusni.

Samo pojedini članovi tima su u potpunosti kvalifikovani za pojedine faze. Ako u timu imate ljude koji samo kodiraju i ne koriste se kod projektovanja i obrnuto, onda to predstavlja čist gubitak, jer programeri sede dok projektanti rade i obrnuto.

Usled nedostataka koji su uočeni kod kaskadnog modela, predložena su različita proširenja ovog modela, koja pokušavaju da prevaziđu nedostatke, a istovremeno zadrže dobre strane. Jedan od takvih modela razvoja softvera je tzv. V model.

V model

V model je koji predstavlja proširenje kaskadnog modela. Umesto da se naniže, u nove faze, ide linearno, pojedine faze teku tako da daju oblik slova V.

Za V model se može reći da je nastao kao rezultat evolucije u testiranju softvera. Definisane su različite tehnike testiranja, koje su jasno odvojene jedna od druge, što zajedno sa kaskadnim modelom vodi ka V modelu.

Prva faza kod V modela je **analiza zahteva**. U ovoj fazi se definiše kako bi trebalo da izgleda idealan sistem. U ovoj fazi se ne definiše kako softver treba da se napravi. Obično se rade intervjui korisnika i prave se odgovarajući dokumenti. Ovi dokumenti opisuju rad sistema sa aspekta korisnika. Korisnici treba da pažljivo prouče ove dokumente, jer će oni služiti kao vodič za projektante. U ovoj fazi se definišu i testovi za prihvatanje softvera od strane korisnika.

Druga faza je **projektovanje sistema**. U ovoj fazi projektanti treba da izvrše analizu funkcija predloženog sistema, na osnovu zahteva iz prethodne faze. Istražuju se mogućnosti i tehnike putem kojih će se implementirati zahtevi korisnika. Ako neki od zahteva nije izvodljiv, o tome treba informisati korisnika. Pronalazi se odgovarajuće rešenje i u skladu sa tim se ažuriraju dokumenti. U ovoj fazi se pravi dokument sa specifikacijm softvera, koji služi kao vodič kasnije prilikom kodiranja. Ovaj dokument sadrži organizaciju sistema, strukturu menija, strukturu podataka itd. U ovoj fazi se pripremaju i dokumenti za testiranje sistema.

Projektovanje arhitekture sistema je sledeća faza. Ova faza predstavlja projektovanje na najvišem nivou. Tu treba izabrati arhitekturu koja je u stanju da ostvari zahteve u zadatom roku, sa zadatom cenom i resursima. Obično se bira između arhitekture u dva nivoa, tri nivoa ili više nivoa. Ovi modeli se obično sastoje od baze podataka, korisničkog interfejsa i poslovne logike. Detaljno se opisuju moduli i komponente koji čine svaki nivo. Opisuju se njihove veze, podsistemi, operativno okruženje i interfejsi. Izlaz iz ove faze je dokument koji sadrži listu modula, njihove veze, tabele u bazi podataka, detalje o tehnologiji itd. Ovde se definiše i način testiranja integracije modula.

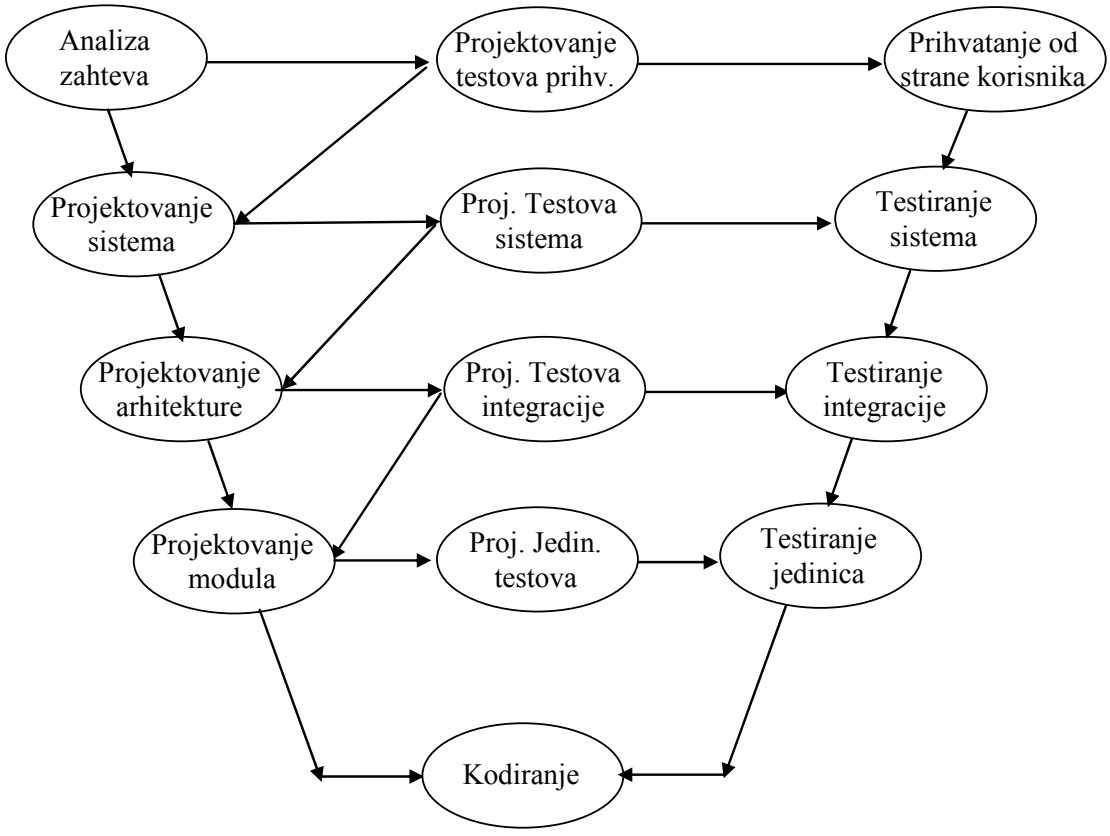
Sledeća faza je **projektovanje modula**. Ovo je projektovanje na nižem nivou. Sada se projektovani sistem deli na manje jedinice i svaka od njih se detaljno objašnjava tako da programeri mogu da počnu da kodiraju. U dokumentima koji ovde nastaju je detaljno opisan način rada svakog modula, opisane su tabele u bazi, kao i interfejs i svi detalji vezani za API. U ovoj fazi se definišu jedinični testovi.

Testiranje počinje **testiranjem pojedinih softverskih jedinica**. Greške koje se ovde otkriju se mnogo jeftinije ispravljaju nego ako se otkriju u toku upotrebe sistema. Ova faza obuhvata analizu napisanog koda i otklanjanje grešaka. Tu se proverava i da li je kod efikasan i da li je u skladu sa prihvaćenim standardima kodiranja. Testiranje se radi na osnovu testova koji su pripremljeni u fazi projektovanja modula. Testove izvršavaju posebni ljudi ili programeri.

U fazi testiranja integracije se proverava interfejs pojedinih modula i komunikacija između pojedinih komponenti. Ovo se radi na osnovu testova pripremljenih u fazi projektovanja arhitekture.

U fazi testiranja sistema se vrši poređenje specifikacije sistema i realizovanog sistema.

Na kraju se vrši i testiranje od strane korisnika. Ovde se testiranje vrši sa realnim podacima, rade ga ljudi koji će kasnije i koristiti sistem. Izvršavaju se testovi koji su napravljeni ranije. U ovoj fazi se pregleda i korisnička dokumentacija.



Iterativni model

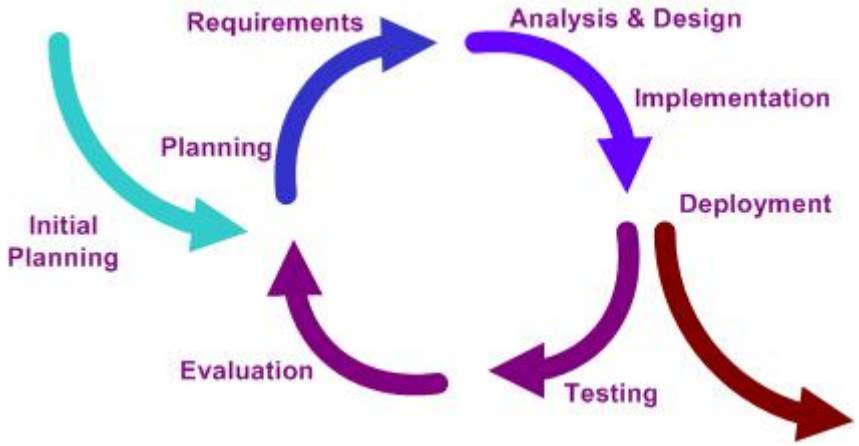
Iterativni i inkrementalni razvoj je ciklični model razvoja softvera, nastao kao odgovor na slabosti kaskadnog modela. On predstavlja sastavni deo RUP (Rational Unified Process) metodologije, kao i Agilnih metodologija.

Incrementalni razvoj je strategija definisanja termina i faza kod koje se različiti delovi sistema razvijaju u različito vreme i integrišu se nakon završetka. Ovakva strategija nije vezana ni za iterativni ni za kaskadni (redni) razvoj. Alternativa incrementalnom razvoju je da se ceo sistem razvije odjednom, po metodu „velikog praska“.

Iterativni razvoj je je strategija prepravki kod koje se odvaja posebno vreme za pregled i poboljšanje delova sistema. Nije obavezno da se iterativni razvoj koristi zajedno sa inkrementalnim razvojem, ali se najčešće koriste zajedno. Osnovna razlika je u tome da se izlaz iz inkrementa šalje korisniku, dok se izlaz iz iteracije ispituje radi poboljšanja. Izlaz iz inkrementa može biti nova funkcija, novi korisnički interfejs, novi algoritam ili tehnologija. Vrlo je verovatno da taj izlaz treba pregledati, proveriti i sa tim rezultatom praviti revizije.

Osnovna ideja koja stoji iza iterativnog razvoja je da se softver razvoja malo pomalo, tako da programeri mogu da iskoriste ono što su naučili tokom razvoja prethodne verzije. Informacije dolaze kako iz procesa razvoja, tako i iz upotrebe sistema, ako je to moguće. Osnovni koraci su početak projekta sa jednostavnom dokumentacijom i samo podskupom zahteva, nakon čega se to malo pomalo proširuje, sve dok se ne implementira ceo sistem. U svakoj iteraciji se menja i dopunjuje projekat, odnosno dodaju se nove funkcije.

Procedura se sastoji od inicijalizacije, iteracije i kontrolne liste projekta. U fazi inicijalizacije se pravi osnovna verzija sistema. Cilj ove faze je da se napravi proizvod koji korisnik može da koristi i na koji može da daje primedbe. Tu treba da budu primeri kako će se rešavati ključni aspekti problema. Rešenje treba da bude jednostavno i razumljivo za korisnike. Za proces iteracije je osnovna kontrolna lista projekta koja sadrži listu svih zadataka koje treba obaviti. Tu su nove stvari koje treba ubaciti, ali i oblasti iz prethodnih iteracija koje treba popraviti.

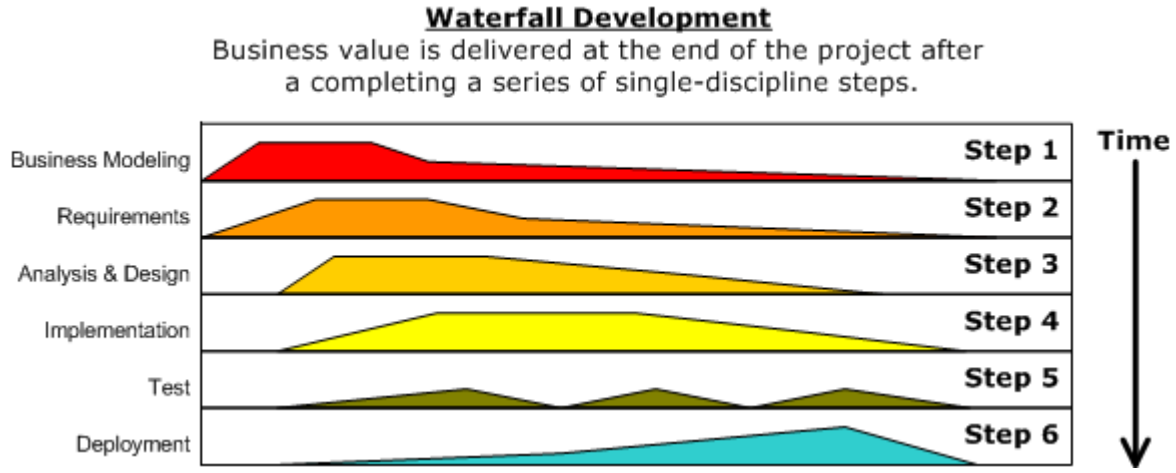


Prilikom projektovanja sistema na umu treba imati sledeće:

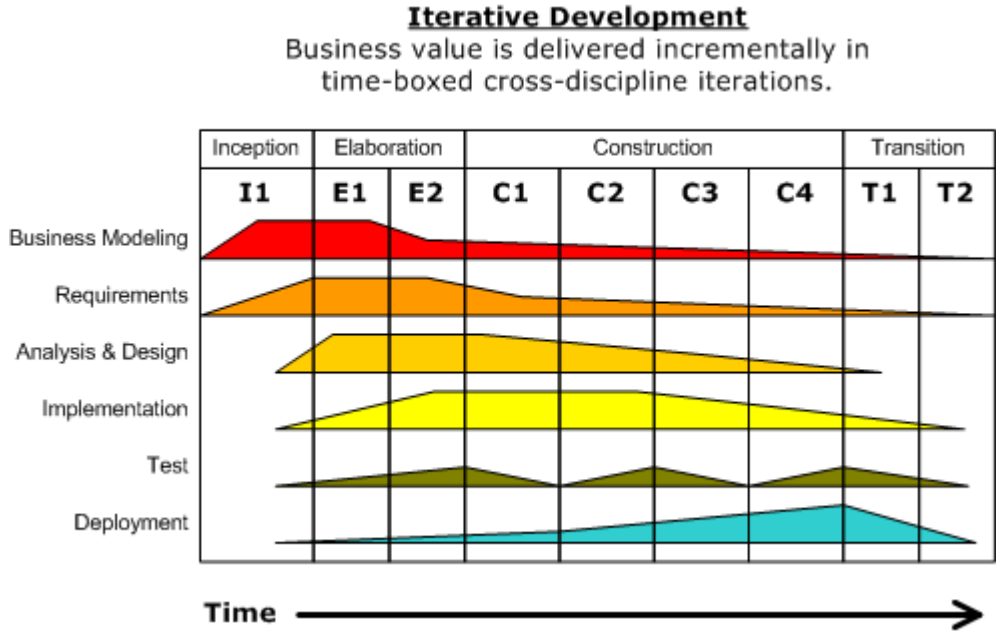
- Sve poteškoće u projektovanju, kodiranju i testiranju signaliziraju da je potrebno popravljati projekat.
- Sve promene treba da se lako ubacuju u module koji su odvojeni i koje je lako pronaći. Ako to nije slučaj, potrebno je menjati projekat.

- Promene u tabelama u bazi podataka treba da je posebno lako napraviti. Ako to nije slučaj, treba menjati bazu.
- Promene treba da budu sve lakše kako proces iteracije napreduje.
- Zakrpe su dozvoljene samo za jednu ili dve iteracije. Može biti da su one neophodne da bi se tokom implementacije izbeglo ponovno projektovanje.
- Postojeća implementacija treba da se stalno analizira da bi se odredilo koliko dobro zadovoljava ciljeve projekta.
- Treba analizirati i reakciju korisnika na trenutnu imlpementaciju.

Upoređenje kaskadnog i iterativnog pristupa razvoju softvera



Kod kaskadnog pristupa se svaka faza završava u potpunosti, pa se tek onda prelazi na sledeću. Isporuka se vrši odjednom, vrlo blizu kraju celog projekta.



Kod iterativnog razvoja se funkcije sistema koje se isporučuju dele u iteracije. U svakoj iteraciji se isporučuje po neki deo funkcija. Kod unificiranog procesa se iteracije grupišu u faze početka,

razrade, konstruisanja i prelaza. U početnoj fazi se definiše domen projekta, rizici i zahtevi. Ovo se radi na vrlo visokom nivou, ali ipak dovoljno detaljnom da može da se vrše procene. U fazi razrade se isporučuje radna verzija arhitekture koja ublažava glavne rizike i ispunjava nefukncionalne zahteve. U fazi konstrukcije se malo pomalo ubacuje kod, a tu se rade i analiza i projektovanje, implementacija i testiranje. U završnoj fazi prelaza se sistem postavlja u radno okruženje. Svaka faza se može podeliti na jednu ili više iteracija. Arhitekte sistema i analitičari rade jednu iteraciju ispred programera.

Agilne metode za razvoj softvera

Agilne metode predstavljaju koncept u razvoju softvera koji propagira iteracije u razvoju. Postoji više agilnih metoda. Zajedničko za sve njih je da se smanjuje rizik kod razvoja softvera, time što se skraćuje vreme razvoja. Ovde se pod iteracijom podrazumeva softver proizveden u jednom vremenskom intervalu. Vremenski interval može biti od jedne do četiri nedelje. Svaka iteracija predstavlja celokupan softverski projekat, što znači da sadrži planiranje, analizu zahteva, projektovanje, kodiranje, testiranje i dokumentovanje. Jedna iteracija možnda neće dati dovoljno funkcija da se proizvod iznese na tržište, ali je cilj da se na kraju svake iteracije napravi nešto što može da se implementira kod korisnika (bez bagova). Na kraju svake iteracije tim ponovo definiše prioritete u projektu.

Agilne metode daju prednost komunikaciji sa korisnicima, nad pisanim dokumentima. Većina timova se nalaze u jednoj kancelariji. Tu se nalaze i programeri i njihovu „kupci“ (kupci su ti koji definišu proizvod i mogu biti menadžeri, poslovni analitičari ili klijenti). U kancelariji se mogu još nalaziti oni koji testiraju programe, ljudi zaduženi za integraciju itd.

Agilne metode takođe ističu da je osnovna mera napretka u projektu softver koji se koristi. U kombinaciji sa komunikacijom lice u lice, agilne metode proizvode vrlo malo pisane dokumentacije, u odnosu na druge metode. Iz ovog i proizilaze kritike agilnih metoda da tu nema discipline.

Savremena definicija agilnih metoda je nastala 1990-ih godina, kao odgovor na tadašnje metode razvoja softvera (kaskadni model). Proces je nastao kao reakcija na probleme sa kaskadnim modelom i činjenicu da taj model nije pratio način na koji softverski inženjeri zaista rade.

2001-e godine su se sastale vodeće ličnosti u ovoj oblasti i napravile manifet agilnih metodologija. U ovom manifestu su definisani osnovni principi razvoja na ovaj način. Neki od najvažnijih principa su:

- Kupac mora biti zadovoljan. To se obezbeđuje time što se softver isporučuje brzo i stalno.
- Često se isporučuje softver koji radi (nedelje su u pitanju, a ne meseci)
- Softver koji radi je mera napretka.
- Naknadne promene u zahtevima su čak i dobrodšle.
- Svakodnevna, bliska saradnja između ljudi iz preduzeća i programera.
- Najbolji oblik komunikacija je licem u lice.
- U projekte su uključeni motivisani ljudi, kojima treba verovati.
- Stalna pažnja se obraća na dobar dizajn i upotrebu najnovijih tehnologija.
- Jednostavnost
- Timovi koji se sami organizuju

Većina agilnih metoda su slične sa drugim iterativnim metodama u tome da se akcenat stavlja na isporuku softvera koji se može koristiti u kraćim vremenskim intervalima. Agilne metode se razlikuju od ostalih po tome što se taj period meri u nedeljama, a ne u mesecima, kao i u tome što se posao obavlja uz visoku meru saradnje. Mnoge agilne metode vremenski period tretiraju kao fiksni vremenski okvir.

Održivost i upotrebljivost agilnih metodlogija se može posmatrati iz različitih perspektiva. Iz perspektive proizvoda, agilni metodi su pogodniji kada su zahtevi hitni i kada se brzo menjaju. Manje su pogodni za sisteme koji imaju kritične, pouzdane zahteve. Iz organizacione perspektive održivost se može posmatrati kroz tri ključne dimenzije jedne organizacije. To su kultura, ljudi i komunikacija. U vezi sa ovim dimenzijama se mogu definisati neki faktori uspešnosti:

- Kultura organizacije mora biti takva da podržava pregovaranje
- Ljudima se mora verovati
- Manje ljudi, ali kompetentniji
- Organizacije moraju da đživeti sa odlukama koje donesu programeri.
- Organizacija mora da ima okruženje koje olakšava komunikaciju između članova tima.

Faktor koji na sve ovo najviše utiče je verovatno veličina projekta. Kako projekat raste, komunikacija licem u lice postaje sve teža. To znači da su agilne metode pogodnije kod projekata sa malim timovima, manje od 20. Veliki softverski projekti koji koriste ovakve metodologije su još uvek u fazi istraživanja.

Najpoznatije metode iz ovog domena su Extremno programiranje, SCRUM itd.