INFORMACIONI SISTEMI: ANALIZA I PROJEKTOVANJE

1 Informacioni sistemi

Informacioni sistem u najopštijem smislu je svaki sistem razvijen u cilju kreiranja, sakupljanja, čuvanja, prenosa, obrade i interpretacije informacije. Informacijom se ovde smatra znanje ili dopuna znanja, a predstavlja se podacima (uredjene kolekcije simbola - znakova i signala). Informacioni sistemi se mogu bazirati i samo na pisanoj i usmenoj komunikaciji, ali je verovatnije da će zahtevati upotrebu računara za upravljanje sakupljanjem, čuvanjem, obradom i distribucijom informacija. Istu informaciju različiti ljudi mogu da upotrebe na različite načine u različita vremena. Na primer, cena rada specifičnog posla služi i za obračun plata, i za računanje proizvodne i reklamne cene, cene proizvoda, i za odredjivanje potreba za protokom novca. Računari omogućuju efikasnost i mogućnosti informacionih sistema koje ne postoje bez njih - to su **računarski** (zasnovani) **informacioni sistemi (RIS)**. Informacioni sistem je obično jedan od podsistema poslovnog sistema.

Kategorije informacionih sistema:

1. *Sistemi za obradu podataka* - vrše automatsku obradu velikog broja podataka, npr. kontrola inventara, obračuni bankovnih računa;

2. *Upravljački informacioni sistemi* - vrše automatsku analizu podataka i proizvode informacije korisniku za upravljanje poslom, npr. mesečna analiza prodaje, žalbe klijenata;

3. *Sistemi za podršku odlučivanju* - vrše specijalne analize koje se tiču strateških problema odlučivanja u upravljanju, npr. dugoročno planiranje nabavke ili izgradnje fabrike ili izbora snabdevača sirovina sa najboljim balansom kvalitet / cena / pouzdanost snabdevanja.

Poslednjih trideset godina, u opisivanju limita raspoložive tehnologije glavna teškoća je savladavanje kompleksnosti softvera. Uprkos značajnom progresu softver je uvek u krizi. Verovatno je, kako jedan od autora objektne metodologije navodi, da to i nije kriza već permanentno stanje stvari.

Postoji više kategorija softvera, i svaka može da se okarakteriše ograničenjima koja postavlja na proces razvoja:

- Amaterski softver – svaki softver koji nema značajniji ekonomski efekat. Razvijaju ga pojedinci u okviru grupa sa istim interesovanjem. U ovu grupu spada i studentski softver.
- Potrošni softver, kao što su procesori reči ili raširene tabele. Ovo je jeftin softver i njegova zamena drugim ne opterećuje finanijski položoj posla za koji je nabavljen. Ovaj softver je skupo održavati, a teško je održavati i balans izmedju kvaliteta, razvojne cene i distribucije.
- Programi neophodni za poslovanje i teško zamenljivi. Oni se grade za specifični zadatak i obično podrazumevaju značajne investicije. Treba da imaju pouzdano, bezbedno i predvidljivo ponašanje. Koriste se u industrijskom i finansijskom poslovanju. Ovi softverski sistemi su kompleksni kao posledica kompleksnosti domena ali i okruženja u kome se izvršavaju – metoda, jezika, operativnih sistema. Njihovo održavanje postaje sve teže, i zato je u njihovom razvoju poterbno primeniti odgovarajuće metodologije koje će ograničiti softverski haos. Ovoj kategoriji softvera pripadaju i **informacioni sistemi**.
- Sistemi sa kritičnom bezbednošću – oni od kojih zavise ljudski životi – u saobraćaju, odbrani ili medicini. Neispravna operacija se u ovim sistemima ne meri novcem već ljudskim životima.

Zadatak softverskih inženjera je da učine stvari jednostavnim, što je izuzetno teško. Stvari se, zapravo, ne mogu učiniti jednostavnim, već se može učiniti da izgledaju jednostavno, tj. može se enkapsulirati kompleksnost.

Razvoj softvera, i posebno informacionih sistema, dakle, potrebno je podići na nivo nauke, tj. eksplicitno definisane metodologije. On se ne sme oslanjati na imitiranje majstora (softverskih "čarobnjaka") jer se tako ne zasniva dugoročna i ponovljiva industrijska praksa. Dakle, aktivnosti koje se izvode za vreme implementacije softvera moraju da se opišu, objasne i opravdaju. Proces razvoja IS opisuje lanac aktivnosti unutar razvojng tima, on usmerava zadatak svakog pojedinca kao i cele grupe. Proces definiše referentne tačke i kriterijume kontrole i mere proizvoda i projektnih aktivnosti.

Greške i defekti u raznim fazama razvoja softvera nemaju sve iste efekte. Greške u analizi proizvode programe koji ne zadovoljavaju sve korisničke zahteve. Greške u projektovanju generišu programe nezgrapne za manipulaciju, spore ili neefikasne. Greške pri implementaciji rezultuju nepredvidivim ponašanjem.

2 Životni ciklus informacionog sistema

Razvoj informacionog sistema može se podeliti u nekoliko glavnih područja (faza). One se redjaju **sekvencijalno** u okviru **linearnog životnog ciklusa** (tj. životnog ciklusa oblika vodopada), ili su distribuirane medju raznim iteracijama **iterativnog životnog ciklusa**.

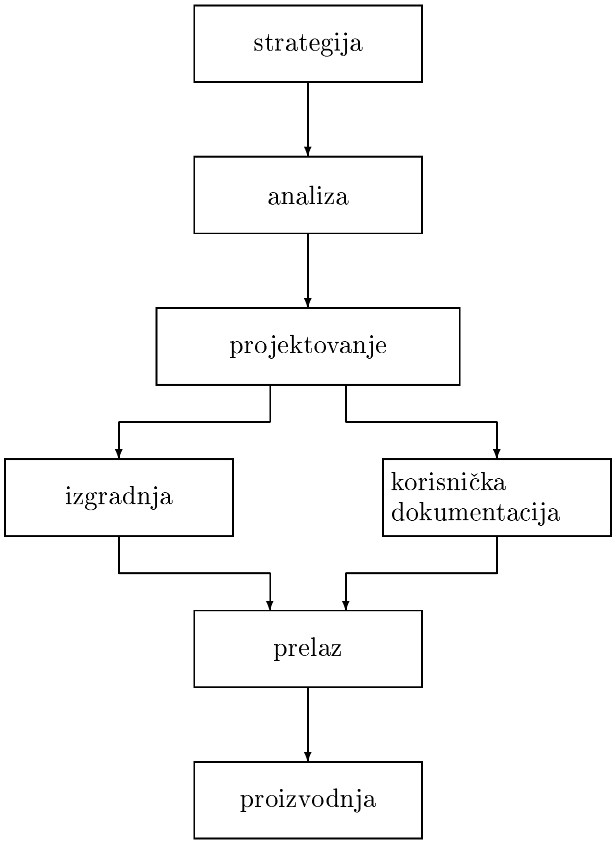
Bez obzira na pristup, linearan ili iterativan, strukturni ili objektno- orijentisani, razvoj aplikacije odgovara na četiri pitanja:

aplikacija = šta uraditi + u kom domenu + kako + kojom veštinom

- **Šta uraditi** – to je pitanje na koje korisnik daje odgovor, opisujući svoja očekivanja od sistema, način interakcije sa sistemom i različite aktere. To je funkcionalni opis, i samo opis – bez detalja implementacije.
- **U kom domenu** – jeste pitanje čiji odgovor treba da sadrži opis domena – okruženja u kome aplikacija treba da egzistira i bitne, po aplikaciju, elemente domena. Odgovor na ovo pitanje pripada prvoj fazi analize – analizi domena, koja je apsolutno nezavisna od razmatranja implementacije. Analiza domena mora biti potpuno razumljiva korisniku.
- **Kako** – dobija odgovor u fazi projektovanja (dizajna). Taj odgovor je rezultat projektantskog iskustva i veštine. Projektovanje treba da učini mogućim korisnikova očekivanja i želje izražene kroz prvo pitanje – **šta uraditi** , uzimajući u obzir domena aplikacije i ograničenja implementacije.
- **Kojom veštinom** jeste pitanje o neophodnim znanjima i veštinama za izgradnju aplikacije – tehničkim, medjuljudskim i organizacionim.

Odgovore na ova pitanja daju faze specifikacije, analize (domena i sistema), projektovanja i implementacije.

Faze razvoja i korišćenja informacionog sistema koje se koncepcijski razlikuju (mada se u implementaciji obično preklapaju) mogu se predstaviti sledećim dijagramom:



Ključne faze u životnom ciklusu informacionog sistema predstavljaju faze *analize* i *projektovanja*. Stoga su i ljudi u timu za razvoj informacionih sistema obično podeljeni u grupe sistem analitičara i projektanata sistema.

Karakteristike procesa analize i projektovanja informacionih sistema najbolje se uočavaju pri uporednom opisu svojstava ovih procesa:

1. Analiza informacionog sistema je proces u kome se opisuje *funkcija* postojećeg ili planiranog informacionog sistema. Projektovanje informacionog sistema je proces u kom se opisuje *struktura* planiranog sistema koja je (naj)pogodnija za automatizovani rad.

2. Analiza informacionog sistema počinje proceduralnim opisom transakcija tog sistema, a zatim se iz tog opisa izvodi opis podataka pojedinih transakcija sistema i globalna funkcija svake od transakcija sistema. Projektovanje informacionog sistema polazi od globalne funkcije transakcija sistema uočenih u analizi sistema, koje zatim širi u praktično primenljivi i optimalni, s gledišta posla, korisnika i cene sistema, opis podataka i procedura koje se implementiraju.

3. Analiza predstavlja opis mogućeg. Projektovanje predstavlja opis praktičnog. Pritom, procedure koje opisuju funkciju sistema, od kojih se polazi u analizi

sistema, u slučaju većih informacionih sistema, po pravilu se ne poklapaju sa procedurama sistema kojma završava proces projektovanja (zbog zahteva koje nalaže efikasnost i praktičnost realizacije).

4. Analiza sistema je pronalazački posao koji zahteva prikupljanje informacija, istraživanje, ispitivanje i uvid u sistem koji treba da se razvije. Projektovanje je kreativni proces koji na osnovu informacije pronadjene u vreme analize, kreira strukturu novog informacionog sistema.

Analiza počinje sa elementima realnog sveta, ali mora neizostavno biti praćena procesom apstrakcije koji omogućuje identifikovanje baznih koncepata iz kojih su generisani, prema odredjenim principima, svi pojavni oblici – elementi – realnog sveta. Na primer, analiza je kao zaključak da su tačka, krug, elipsa, prava, dve prave koje se seku, hiperbola i parabola – svi dobijeni kao ravni konusni preseci. Princip koji generiše ovde je izražen jednačinom

$$ax^2 + by^2 + 2cx + 2dy + e = 0$$

Analiza traži suštinu, tj. generičke i apstraktnije modele.

Paralelno sa analizom okruženja odvija se i analiza postojećeg i ograničenja implementacije.

Kao analiza i projektovanje, tako se i projektovanje i implementacija delimično preklapaju. Da bi se "otkrio" najbolji projekat potrebno je, često, isprobati razne tehnike i proceniti rezultate. Zato su projektovanje i implementacija komplementarni. Projektovanje zahteva eksperiment, dok implementacija zahteva ideju vodilju ("driving principle").

Projektovanje je obično podeljeno u dve faze: logički korak koji je nezavisan od okruženja implementacije, i fizički korak koji se odnosi na resurse, programske jezike ili izvršno okruženje.

Projektovanje počinje sa odredjivanjem **arhitekture softvera**, razvojem statičkih i dinamičkih struktura podataka koje će se koristiti kao skelet za ceo proces razvoja. Arhitektura definiše opštu strukturu aplikacije, ona je posledica strateških odluka koje se donose za vreme projektovanja i taktičkih odluka u vreme implementacije.

Projktovanje je domen kompromisa. Ne postoje crna i bela rešenja – sve su nijanse sivog. Razne taktičke odluke utiču na aktivnosti u razvoju. Jedna od odluka koja mora biti doneta odnosi se na implementaciju komponenti sa mogućnosšću ponovnog korišćenja (u slučaju primene objektne metodologije). Ova implementacija označava dve stvari: programiranje **sa** ponovnim korišćenjem (ukljčenje postojećih komponenti što ubrzava proces razvoja), i programiranje **za** ponovno korišćenje, što predstavlja investiciju kojoj se cena vidi odmah, a dobit stiže tek kasnije.

Rezultat faza analize i projektovanja su analitička i projektantska biblioteka dokumentacije razvoja informacionog sistema, koje se sastoje od dijagrama i opisa procedura i tokova podataka identifikovanih u svakoj od ovih faza. Za realizaciju ovih faza postoji niz metoda od kojih će u narednim odeljcima biti predstavljena metoda strukturne sistem analize (SSA) tj. strukturnog projektovanja, i metoda objektne analize i objektnog projektovanja.

Faza *izgradnje* informacionog sistema odnosi se na kreiranje baza podataka i implementaciju transakcija koje odgovaraju pojedinim jedinicama posla, tj. na izradu aplikativnog softvera.

Izrada *korisničke dokumentacije* je važna faza razvoja informacionog sistema, koja svojim rezultatima omogućuje efikasnu upotrebu i održavanje informacionog sistema.

Prelaz uključuje konverziju podataka iz postojećeg oblika u kreiranu bazu podataka tj. punjenje baze, generisanje test-podataka i upravljanje računarskom konfiguracijom u cilju priprema za eksploataciju informacionog sistema.

Proizvodnja je faza koja odgovara eksploataciji i održavanju informacionog sistema i koja uključuje kontrolu promena, kontrolu verzija, i eventualno redefinisanje i redokumentovanje sistema.

U pojedinim fazama razvoja IS ponavljaju se prethodne faze, da bi se dobio kvalitetniji sistem, ili da bi se udovoljilo izmenjenim zahtevima.

Kao što je to slučaj sa bilo kojom tehnikom modeliranja, ne postoje konačna rešenja i odgovori na zadatke koji se postavljaju u svakoj od faza životnog ciklusa informacionog sistema. Svaka od niza metoda i tehnika kojima se obavljaju pojedine faze predstavlja jedan od mogućih načina modeliranja realnosti, i uz sve prednosti i nedostatke koje poseduju, jedini uslov koji se nameće pri njihovom izboru je da zadovolje postavljene zahteve izrade modela, koji su sadržani u samom opisu funkcionalnosti pojedinih faza.

2.1 Linearni životni ciklus

Linearni životni ciklus odnosi se na pristup razvoju softvera koji je baziran na sukcesiji koraka, od zahteva do implementacije. Postoji nekoliko varijanti ovakvog pristupa.

Model tunela. Ovo je ilustrativni način da se izrazi nepostojanje modela razvoja. U projektima sa tunelskim pristupom nemoguće je znati šta se dešava. Razvoj napreduje, ljudi rade – često vrlo naporno, ali nema pouzdane informacije o tome kako softver napreduje ili kakvog su kvaliteta razvijeni elementi. Ovaj model razvoja odgovara samo malim projektima sa vrlo ograničenim brojem učesnika.

Model vodopada. Ovaj model podržava opšti opis aktivnosti u razvoju softvera. Prema ovom modelu, razvoj softvera je niz faza povezanih u linearno izvršenje, od specifikacije i analize zahteva do isporuke proizvoda. Svaka faza odgovara jednoj aktivnosti. Ovakav razvoj praćen je izradom dokumentacije koja predstavlja podršku za validaciju svake faze.

U odnosu na tunelski model, ovaj model ima veliku prednost u tome što definiše kontrolne tačke, u vidu dokumenata. Status projekta je vidljiviji (mada

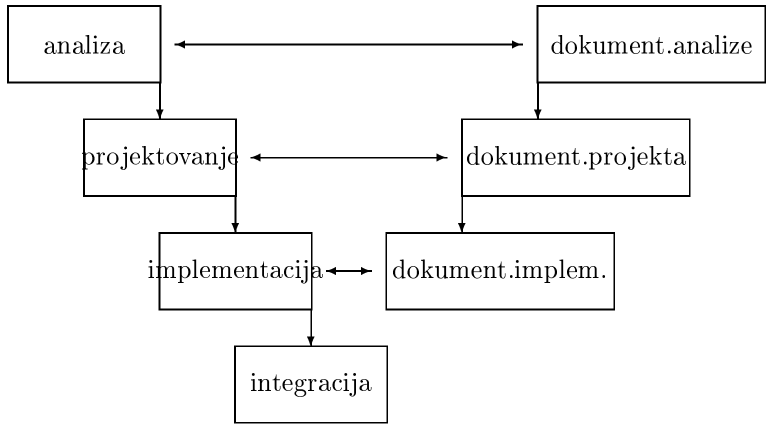


Figure 1: Životni ciklus: model vodopada

samo kroz dokumenta tj. dokumentaciju faza). Grafički se simultanost faza razvoja i izrade odgovarajuće dokumentacije predstavlja slovom "V" (slika 1).

Ograničenja ovog modela su što je dokaz dobrog (ili lošeg) rada sistema dostupan tek kasno u ciklusu – za vreme faze integracije. Sve do te faze, dostupna je za evaluaciju samo dokumentacija a ne i konkretni softver. Da bi model dobro funkcionisao, neophodno je da se faze mogu povezati efikasno bez većih problema, da je poznat na početku kompletan skup zahteva i da analitičari potpuno razumeju problem, da projektanti mogu lako da odrede rešenje, a da se programiranje idealno svodi na automatsko generisanje koda, na osnovu dokumentacije projekta.

Na žalost, projekti nisu uvek tako idealni.

2.2 Iterativni i inkrementalni životni ciklus

Iterativni životni ciklus softvera, pa i informacionog sistema, bazira se na jednostavnoj ideji: kada je sistem suviše složen da se razume, projektuje i implementira u jednom pokušaju, bolje je implementirati ga iterativno, evolucijom. I u prirodi su funkcionalno složeni sistemi uvek evoluirali od jednostavnijih sistema.

Implementacija ove ideje u razvoju softvera nije uvek jednostavna. Softver je veoma osetljiv na promene i modifikacije. Softversko inženjerstvo nas uči da je neophodno, da bi softver bio robustan,

- segmentirati prostor mogućih stanja,
- redukovati kopčanje (v. odeljak Strukturno projektovanje) korišćenjem nivoa apstrakcije, i
- razdvojiti specifikaciju od implementacije.

Na primer, OO pristup se gradi oko koncepata kao što su enkapsulacija i modularnost koji nameću defanzivni programski stil. Po prirodi, OO programi imaju veću otpornost na promene i iznenadjenja. Stoga OO pristup favorizuje iterativni pristup u razvoju programa. S druge strane, nije neophodno koristiti OO pristup da bi se primenio iterativni ciklus razvoja. OO pristup samo nudi udoban okvir za razvoj na evolutivni način, korišćenjem iterativnog pristupa.

Karakteristike iterativnog životnog ciklusa. Ovaj pristup bazira se na evoluciji merljivih i izvršnih prototipova, pa prema tome i na evoluciji konkretnih elemenata (nasuprot modelu vodopada koji se bazira na razvoju dokumenata). Neophodno proizvodjenje konkretnih rezultata eliminiše sindrom "90% uradjeno, još 90% treba da se uradi". Regularno izvršavanje iteracija omogućuje da se problemi uzmu u obzir i da se promene ugrade u buduće iteracije, a ne da se remeti i prekida aktivnost koja je u toku.

Za vreme razvoja, neki prototipovi se pokazuju korisnicima i kupcima, što ima višestruke prednosti, od mogućnosti za korisnika da bolje razume i struktuira svoje zahteve, da postane član tima sa sopstvenom odgovornošću, preko jače motivacije za razvojni tim, do mogućnosti postupne integracije softverskih komponenti (bez "big bang" efekta) i mogućnosti merenja napretka programima koji se mogu demonstrirati, a ne dokumentima ili procenama.

Mini vodopad. U iterativnom životnom ciklusu, svaka iteracija predstavlja model vodopada na manjoj skali. Ciljevi naredne iteracije se baziraju na evaluaciji prethodnih iteracija. Aktivnosti su stoga povezane u mini-vodopad čiji je opseg definisan ciljevima iteracije (slika 2).

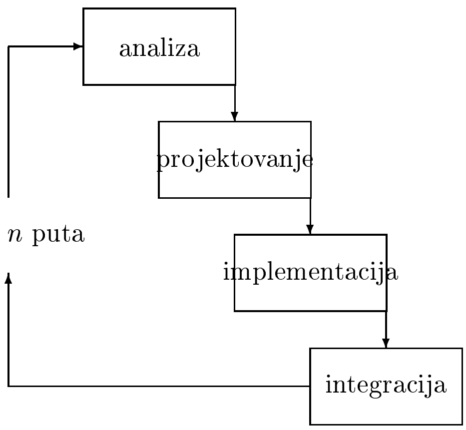


Figure 2: Iterativni životni ciklus: model minivodopada

Svaka iteracija podrazumeva sledeće aktivnosti:

- planiranje iteracije na osnovama rezultata proučavanja rizika (komercijalnog – u odnosu na konkurenciju, finansijskog – u odnosu na mogućnosti firme, tehničkog – u odnosu na kvalitet tehnološke baze, razvojnog – u odnosu na iskustvo tima).
- analizu elemenata planiranih za tu iteraciju
- projektovanje se fokusira na taktičke izbore neophodne da se implementiraju mehanizmi dodeljeni iteraciji; ako je potrebno, primenjuju se modifikacije na arhitekturu i model projektovanja se ažurira.
- programiranje i testiranje; integracija novog koda sa postojećim kodom (iz prethodnih iteracija) implementira se postupno za vreme konstrukcije. Procedure testiranja elemenata i integracije se primenjuju na prototip.
- Evaluacija za isporuku – ispitivanje rezultata testa koristi se kao osnova za evaluaciju prototipa.
- Priprema isporuke – prototip se zamrzava i ceo skup elemenata od kojih se sastoji smešta se u kontrolne biblioteke. Završavaju se razni dokumenti koji se odnose na opis i instalaciju prototipa.

U raznim varijantama ove vrste životnog ciklusa, razne faze se mogu iterirati (ne moraju sve). Na primer, kod srednje velikih aplikacija kod kojih se implementacija sprovodi u okviru već poznate arhitekture, može se iterirati samo konstrukcija (i integracija). U slučaju da zahtevi nisu potpuno poznati od početka, da bi se uzele u obzir modifikacije arhitekture, u ciklus se unose i faze projektovanja, tj. analize. Za velike projekte karakterističan je ciklus koji uključuje i analizu i projektovanje u svoje iteracije (tzv. životni ciklus O-oblika (slika 2).

Planiranje iteracija. Ne postoje gotova rešenja vezana za planiranje iteracija. Za projekat koji traje 18 meseci, biće izmedju tri i šest iteracija. Iteracije imaju približno isto trajanje. Prvu iteracije je najteže sprovesti, jer ona zahteva postavku celog okruženja razvoja, kao i tima. Ne treba očekivati suviše od prve iteracije, inače se rokovi probijaju, broj iteracija smanjuje i gube prednosti iterativnog pristupa.