Informacioni sistemi

Ajzenhamer Nikola
Bukurov Anja
Stanković Vojislav

Ajzenhamer Nikola
Bukurov Anja
Stanković Vojislav

Informacioni sistemi

8. januar 2017.

Sadržaj

Predgovor **vii**

1 Pojam informacionih sistema **1**
1.1 Informacija . . . 1
1.2 Sistem . . . 2
1.3 Pojam informacionog sistema . . . 3
1.3.1 Opšte o poslovnim aspektima informacionih sistema . . . 3
1.3.2 Opšte o strukturnim aspektima informacionih sistema . . . 4
1.3.3 Opšte o funkcionalnim aspektima informacionih sistema . . . 4
1.4 Socijalna dimenzija informacionih sistema . . . 5
1.5 Formalizacija informacionih sistema . . . 5
1.6 Poslovni aspekti informacionih sistema . . . 6
1.6.1 Poslovni informacioni sistemi . . . 7
1.7 Funkcionalni aspekti informacionih sistema . . . 8
1.8 Strukturni aspekti informacionih sistema . . . 9
1.9 Pitanja . . . 11
1.10 Literatura . . . 12

2 Razvojne metodologije **13**
2.1 Uloga razvojnih metodologija . . . 13
2.2 Životni ciklusi informacionih sistema . . . 15
2.2.1 Model vodopada . . . 15
2.2.2 Drugi oblici životnih ciklusa . . . 18
2.3 Pitanja . . . 20
2.4 Literatura . . . 21

3 Tehnike **23**
3.1 Opšte o tehnikama . . . 23
3.2 Tehnike opisivanja procesa . . . 23
3.2.1 Tehnike za opisivanje poslovne logike . . . 24
3.2.2 Tehnike za praćenje projekta . . . 27
3.2.3 Tehnike za opisivanje strukture procesa . . . 28
3.3 Tehnike modeliranja podataka . . . 38
3.3.1 Dijagrami entiteta i odnosa . . . 38

3.3.2 Dijagrami klasa podataka . . . 40
3.3.3 Izbor tehnike . . . 41
3.4 Objektno-orijentisane tehnike . . . 41
3.4.1 Osnovni pojmovi . . . 42
3.4.2 Osnovni koncepti . . . 43
3.4.3 UML . . . 44
3.4.4 Vrste UML dijagrama . . . 44
3.4.5 UML opis slučajeva upotrebe . . . 62
3.5 Druge tehnike . . . 64
3.5.1 Holističke tehnike . . . 64
3.5.2 Tehnike vođenja projekata . . . 65
3.6 Pitanja . . . 68
3.7 Literatura . . . 71

4 Strukturne metodologije razvoja 73
4.1 Procesno-orijentisane metodologije . . . 73
4.1.1 STRADIS . . . 73
4.1.2 YSM . . . 74
4.2 Kombinovane metodologije . . . 75
4.2.1 SSADM . . . 75
4.3 Pitanja . . . 77
4.4 Literatura . . . 77

5 Problemi pri razvoju IS i njihovo rešavanje 79
5.1 Problemi i neuspeh . . . 79
5.2 Neupotrebljiv rezultat . . . 79
5.2.1 Najčešći problemi . . . 80
5.3 Upravljanje rizicima . . . 81
5.3.1 SCERT . . . 81
5.3.2 Upravljanje rizicima (*Boehm*, 1991) . . . 81
5.3.3 Procena izloženosti riziku . . . 82
5.3.4 Međusobna zavisnost poslova . . . 83
5.4 Pitanja . . . 85
5.5 Literatura . . . 85

6 Objektno-orijentisane metodologije 87
6.1 Savremeni koncepti razvoja informacionih sistema . . . 87
6.2 Razvoj objektno-orijentisanih metodologija . . . 89
6.2.1 Osnovni koncepti . . . 90
6.2.2 Pregled objektno-orijentisanih metodologija . . . 91
6.3 Objektno-orijentisani analiza i dizajn (OOAD) . . . 92
6.4 Pitanja . . . 93
6.5 Literatura . . . 94

7 Rational Unified Process (RUP) 95
7.1 Pojmovi, postulati, i principi . . . 95
7.2 Životni ciklus projekta . . . 98
7.2.1 Početak (Inception) . . . 99
7.2.2 Razrada (Elaboration) . . . 99
7.2.3 Konstrukcija (Construction) . . . 99
7.2.4 Prelazak (Transition) . . . 100
7.2.5 Odnos sa pristupom vodopada . . . 101
7.3 Strategije planiranja . . . 101
7.3.1 Inkrementalna strategija . . . 102
7.3.2 Evoluciona strategija . . . 102
7.3.3 Strategija inkrementalno isporučivanje . . . 103
7.3.4 Strategija "Veliki projekat" . . . 104
7.3.5 Hibridne strategije . . . 105
7.4 Discipline . . . 105
7.4.1 Poslovno modeliranje . . . 105
7.4.2 Zahtevi . . . 107
7.4.3 Analiza i projektovanje . . . 109
7.4.4 Konfigurisanje i upravljanje izmenama . . . 112
7.4.5 Okruženje . . . 113
7.4.6 Implementacija . . . 114
7.4.7 Upravljanje projektom . . . 116
7.4.8 Testiranje . . . 118
7.5 Uloge . . . 120
7.6 Rezultat rada . . . 122
7.7 Pitanja . . . 126
7.8 Literatura . . . 128

8 Bezbednost 129
8.1 Predmeti ugrožavanja . . . 129
8.2 Napadi . . . 130
8.3 Slojevi sistema i zaštita . . . 131
8.4 Opasnosti i problemi . . . 131
8.4.1 Uobičajene opasnosti . . . 132
8.4.2 Uzroci problema bezbednosti . . . 135
8.5 Identifikacija, autentikacija i autorizacija . . . 136
8.5.1 Identifikacija . . . 136
8.5.2 Autentikacija . . . 137
8.6 Razmatranja bezbednosti . . . 139
8.6.1 Praćenje aktivnosti . . . 139
8.6.2 Nadgledanje . . . 140
8.6.3 Procenjivanje . . . 140
8.6.4 Kriptografija . . . 140
8.6.5 Fizičke opasnosti . . . 140

8.7 Slabosti i prevazilaženja . . . 141
8.8 Pitanja . . . 143
8.9 Literatura . . . 144

9 Arhitekture orijentisane prema servisima 145
9.1 Uvod . . . 145
9.1.1 Uopštavanje komunikacije . . . 146
9.1.2 *SOA* i distribuirani objekti . . . 146
9.2 Istorijat arhitektura . . . 146
9.3 Uglovi posmatranja na *SOA* . . . 147
9.4 Servisi u kontekstu *SOA* . . . 148
9.5 UDDI . . . 148
9.6 Servisi i arhitektura . . . 149
9.6.1 Vrste servisa . . . 149
9.6.2 Servis kao fasada . . . 149
9.6.3 Slojevi arhitekture *SOA* . . . 149
9.6.4 Referentna arhitektura . . . 150
9.6.5 Životni ciklus . . . 151
9.7 Uloge u razvoju . . . 151
9.8 Implementacija i alternative . . . 151
9.9 Pitanja . . . 152
9.10 Literatura . . . 153

10 Virtualizacija 155
10.1 Pojam virtualizacije . . . 155
10.2 Virtualne mašine . . . 156
10.3 Motivacija . . . 158
10.4 Istorija virtualizacije računara . . . 159
10.5 Vrste virtualnih mašina . . . 159
10.5.1 Sistemske virtualne mašine . . . 160
10.6 Implementaciona pitanja . . . 160
10.6.1 Režim rada . . . 161
10.6.2 Rad sa memorijom . . . 163
10.7 Virtualizacija i informacioni sistemi . . . 163
10.7.1 Infrastrukturni problemi informacionih sistema . . . 163
10.7.2 Doprinosi virtualizacije . . . 164
10.8 Dodatna pitanja i tehnike . . . 165
10.8.1 Ograničenja virtualizacije servera . . . 165
10.8.2 Virtualno računarstvo . . . 165
10.8.3 Računarstvo u oblaku . . . 165
10.9 Pitanja . . . 166
10.10 Literatura . . . 167

11 Računarstvo u oblaku **169**
11.1 Pojam računarstva u oblaku . . . 169
11.2 Prethodnici računarstva u oblaku . . . 169
11.2.1 Klasteri . . . 169
11.2.2 Grid . . . 170
11.2.3 "Korisno" računarstvo . . . 170
11.3 Modeli i vrste oblaka . . . 170
11.4 Bezbednost u oblaku . . . 171
11.5 Pitanja . . . 172
11.6 Literatura . . . 172

12 Pokretno računarstvo **173**
12.1 Uvod . . . 173
12.2 Odnos sa informacionim sistemima . . . 176
12.2.1 Izazovi . . . 176
12.3 Distribuiranost. Podaci. Mobilni informacioni sistemi . . . 177
12.3.1 Primer arhitekture MIS . . . 177
12.3.2 Kategorizacija podataka . . . 179
12.4 Transakcije . . . 180
12.4.1 Uticaj pokretljivosti na atomičnost . . . 180
12.4.2 Uticaj pokretljivosti na konzistentnost . . . 180
12.4.3 Uticaj pokretljivosti na izolovanost . . . 181
12.4.4 Uticaj pokretljivosti na trajnost . . . 181
12.4.5 Uticaj pokretljivosti na potvrđivanje transakcija . . . 181
12.5 Povezanost pokretne jedinice . . . 182
12.5.1 Problemi usled nepovezanosti . . . 182
12.6 Pokretni SUBP (PSUBP) . . . 182
12.6.1 Transakcije u PSUBP-u . . . 185
12.7 Pitanja . . . 188
12.8 Literatura . . . 188

Dodaci **189**

Varijante modela vodopada **191**

Literatura **196**

Predgovor

Ovaj tekst predstavlja skriptu iz predmeta "Informacioni sistemi", na Matematičkom fakultetu Univerziteta u Beogradu, zasnovanu na materijalima prof. dr Saše Malkova iz istog predmeta. Skripta je prateći materijal pre svega studentima koji ovaj kurs slušaju u okviru svojih studija, ali i svima Vama koji biste želeli da se upoznate sa ovom tematikom.

Ovaj materijal ne može zameniti pohađanje nastave niti drugu preporučenu literaturu.

Ukoliko ste pažljivi čitalac ove skripte, i ukoliko uočite bilo kakvu grešku ili propust, molimo Vas da nam to javite na **mi13050@alas.matf.bg.ac.rs**. Svi komentari, sugestije, kritike, ali i pohvale vezane za ovaj materijal su dobrodošli.

Autori

Glava 1

Pojam informacionih sistema

1.1 Informacija

Pojam informacije interesuje informatičare, matematičare i filozofe. Ne postoji jednostavna, a univerzalna definicija. Najčešće se povezuje sa pojmovima *podatak*, *znanje* i *mudrost*. Za podatak možemo reći da je to najjednostavnija reprezentacija činjenice, događaja ili ideje u određenom zapisu, a za informaciju da je to podatak povezan kontekstom. Pogrešno je poistovetiti podatak i informaciju.

Podaci (engl. *data*) su prosti zapisi koji imaju smisao samo u kontekstu u kome su zapisani. Primeri podataka su: "1052/2014", "25. oktobar 1992", "Petar Petrović", "M", itd. *Informacije* (engl. *information*) su podaci koji su obrađeni i zapisani u složenom kontekstu u kome imaju određeni složeniji smisao. Primeri informacija su: "Petar Petrović je 25. oktobra 1992. zaposlen u firmi M i 2014. godine je 1052. po rangu primanja", ili "Petar Petrović je rođen 25. oktobra 1992. i upisao je master studije na Matematičkom fakultetu sa brojem indeksa 1052/2014".

Da bi informacije imale vrednost, one moraju biti: tačne i precizne, potpune, ekonomične, fleksibilne, pouzdane, relevantne, jednostavne, blagovremene, proverljive, pristupačne, i bezbedne. Tačnost se odnosi na to da informacije nisu lažne. Preciznost se može narušiti ukoliko je informacija nedovoljno precizna (na primer, ako za rangiranje studenata prema proseku koristimo informaciju da je "prosečna ocena veća od 8") ili ako je preterano precizna (ako koristimo informaciju da je "prosečna ocena jednaka 9.13254021"). Tačnost i preciznost uslovljavaju upotrebljivost. Informacije moraju biti potpune u kontekstu kome odgovaraju. Informacije treba da budu ekonomične za proizvođenje, prenos i održavanje. Fleksibilnost se odnosi na upotrebljivost u više različitih konteksta (za koje nisu izvorno namenjene). Informacija je pouzdana samo ako na njoj mogu počivati i kritične operacije. Nerelevantne informacije, tj. one koje nisu značajne u nekom kontekstu, predstavljaju samo dodatno opterećenje. Suvišna složenost predstavlja značajno

preopterećenje. Informacija mora da se odnosi na odgovarajući vremenski trenutak ili period. Za svaku informaciju mora da bude poznato poreklo, kako bi se mogla proveriti. Konačno, informacije moraju biti lako dostupne autorizovanim korisnicima, a zaštićene od pristupa neautorizovanih korisnika.

Informacija nije potpuna i proverljiva ako nije poznato njeno poreklo i prostor važenja. Ulazni podaci se dopunjavaju podacima o kontekstu, kao što su: izvor (odakle potiče informacija), vrsta i tip (različite kategorije informacije), vreme (od kada potiče informacija), povezanost sa drugim podacima ili informacijama, i domen (odnosno, na šta se informacija odnosi).

Svako poslovanje počiva na informacijama o: subjektima koji učestvuju u poslovanju, predmetima poslovanja, načinu obavljanja određenih postupaka, prethodno obavljenim poslovima, planiranim poslovima, stanju poslova koji su u toku, kao i na drugim potrebnim informacijama.

1.2 Sistem

Pojam sistema možemo posmatrati na različite načine. Na primer, sistem možemo posmatrati kao kombinaciju sastavnih delova koja predstavlja složenu ili jedinstvenu celinu. Takođe, možemo ga posmatrati kao funkcionalnu celinu sastavljenu od više komponenti, itd. Formalne definicije nisu uvek prikladne zato što: granice sistema često nisu sasvim jasne, praktično svaki sistem predstavlja sastavni deo nekog šireg sistema, praktično svaki sistem obuhvata neke podsisteme ili tesno sarađuje sa drugim sistema, i dr.

Umesto definicije, sistem je mnogo pogodnije opisati njegovim odlikama. Neke od odlika sistema su: postoje u nekom okruženju, imaju (nekakve) granice (kojima su odvojeni od svog okruženja), imaju ulaze ili izlaze (odnosno, komuniciraju sa okruženjem), imaju interfejs(e) (odnosno, komuniciraju sa drugim sistemima), obuhvataju neke procese, mogu imati podsisteme, mogu imati kontrolne mehanizme, i imaju neka svojstva koja ne potiču od komponenti nego su karakteristika celine (sistem je više od prostog skupa povezanih komponenti).

Neke karakteristike sistema su:

- jednostavnost/složenost, koja zavisi od broja komponenti i njihovih odnosa. Jasno je da želimo da naš sistem bude jednostavan, ali, naravno, nekada to ne možemo da izbegnemo. Složenost sistema u velikoj meri zavisi i od problema kojim se bavimo.

- otvorenost/zatvorenost, koja zavisi od obima interakcija sa okolinom. Ako imamo sistem koji treba da bude visoko bezbedan, on će biti zatvoren. Ako

imamo sistem koji želimo da lako razvijamo, dodajemo nove komponente i slično, on će biti otvoren.

- stabilnost/nestabilnost, koja predstavlja stepen promenljivosti tokom vremena. Stabilnost znači da se sistem nalazi u okruženju koje se slabo menja, i da on sâm ne trpi velike izmene. Idealno je kada imamo zadatak da napravimo stabilan sistem. Međutim, u praksi to obično nije slučaj.

- prilagodljivost/neprilagodljivost, koja predstavlja sposobnost menjanja u zavisnosti od promena u okruženju. Na primer, napravili smo informacioni sistem u jednoj firmi; da li možemo da ga iskoristimo u drugoj? Ukoliko je napravljen dovoljno apstraktno, sigurno će moći, samo je pitanje koliko.

- trajnost/privremenost, koja predstavlja trajanje sistema kao celine. Na primer, informacioni sistem olimpijade, nakon što se događaj održi, neće imati funkciju (pri čemu prikupljeni podaci mogu biti iskorišćeni nadalje, naravno). Sa druge strane, sistem opštinske uprave bi bio primer sistema koji je trajan.

1.3 Pojam informacionog sistema

Koncept informacionog sistema se javlja oko 1960. godine, i iako se smatra dobro uspostavljenim konceptom, i dalje se teško precizno definiše. Ova teškoća ima delom korene u činjenici da se informacioni sistem može analizirati iz tri različita ugla:

1. iz poslovnog ugla (kakav doprinos čini, tj. zašto se informacioni sistem pravi),

2. iz ugla strukture i ponašanja, i

3. iz funkcionalnog ugla (odnosno, koje su to funkcije koje informacioni sistem i svaki njegov deo vrši).

Ova tri ugla se nazivaju *aspekti informacionih sistema*.

1.3.1 Opšte o poslovnim aspektima informacionih sistema

Posmatranje informacionih sistema iz poslovnog ugla je uobičajeno u rukovodećim strukturama organizacije, posebno pri planiranju i investiranju u izgradnju i održavanje informacionih sistema. Od interesa su, prvenstveno, doprinos informacionih sistema širem kontekstu poslovanja, i cena i rokovi izgradnje i održavanja informacionih sistema. Informacioni sistem se posmatra kao sastavni deo opšteg poslovnog sistema. Svaka poslovna organizacija koja upotrebljava neke informacije predstavlja (ili čini ili obuhvata) jedan informacioni sistem.

Iz ovog aspekta, neke od definicija informacionog sistema bi mogle biti:

- *Informacioni sistem* je sistem razvijen da bi podržavao procese, upravljanje i odlučivanje u organizaciji.
- *Informacioni sistem* je sistem koji posreduje u komunikaciji između svojih korisnika.

Shvatanje informacionih sistema iz ovog ugla je veoma značajno (i često presudno) u svim faza nastajanja i upotrebe informacionih sistema. Takođe je veoma daleko od sagledavanja iz tehničkog ugla.

1.3.2 Opšte o strukturnim aspektima informacionih sistema

Posmatranje informacionih sistema iz strukturnog ugla je primereno razvoju i održavanju. Prvenstveno se posmatraju struktura informacionih sistema (unutrašnja i spoljašnja), ponašanje pojedinačnih komponenti, i ponašanje informacionog sistema kao celine. Informacioni sistem se može posmatrati kao proizvod koji mora da zadovoljava određene uslove.

Iz ovog aspekta, definicija informacionog sistema bi mogla biti:

- *Informacioni sistem* je sistem koji čine ljudi, podaci, procesi i interfejsi.

Strukturni aspekt nije primeren početnim fazama konceptualnog modeliranja.

1.3.3 Opšte o funkcionalnim aspektima informacionih sistema

Posmatranje informacionih sistema iz funkcionalnog ugla je primereno konceptualnom modeliranju. Od interesa su, pre svega, funkcije koje informacioni sistem ispunjava. Na ovom nivou apstrakcije je nebitno zašto ili kako se nešto implementira.

Iz ovog aspekta, definicija informacionog sistema bi mogla biti:

- *Informacioni sistem* je sistem koji prikuplja, skladišti, obrađuje i isporučuje informacije.

Dobro je napomenuti da se operacije prikupljanja i skladištenja, najčešće, razmatraju kao jedno.

Rezimirajmo ova tri pogleda njihovim razlikama. Poslovni aspekt nam govori šta je cilj informacionih sistema i zašto se oni prave, strukturni aspekt nam govori od čega se informacioni sistem sastoji i kako se informacioni sistem implementira, a funkcionalni aspekt nam govori koje zadatke informacioni sistemi (kao i svi njegovi delovi) moraju da izvršavaju. Važno je napomenuti da je za izradu jednog informacionog sistema nephodno razmatrati sve aspekte.

1.4 Socijalna dimenzija informacionih sistema

Svaki informacioni sistem ima dve podjednako značajne dimenzije. To su socijalna i tehnička dimenzija. *Socijalna dimenzija* obuhvata sve ljude koji učestvuju u radu informacionog sistema, a *tehnička dimenzija* obuhvata sve ono što je napravljeno u okviru informacionog sistema. Informacioni sistem dobija celovitost i pun smisao tek u socijalnom kontekstu kome pripada. Značaj socijalne dimenzije se posebno uočava u ranim fazama razvoja informacionih sistema. Neki ciljevi i funkcije nisu dostižni bez razumevanja socijalne dimenzije problema.

Prema Landu (*Land*, 1992), informacioni sistem je sistem koji obuhvata četiri komponente. To su: korisnici informacija, domen informacionog sistema, formalni sistem, i neformalni sistem. Prema Vinteru (i ostalima; *Winter et.al*, 1995), informacioni sistem obuhvata dva povezana sistema. To su: sistem koji pruža usluge i sistem kome se pružaju usluge. Suštinski su ova dva pristupa jednaka, s obzirom da je sistem koji pruža usluge zapravo formalni sistem, a sistem kome se pružaju usluge zapravo čine korisnici informacija, domen informacionog sistema i neformalni sistem.

Sistem kome se pružaju usluge se mora konceptualizovati tokom prikupljanja zahteva da bi bilo moguće dobro konceptualizovati i projektovati sistem koji pruža usluge. Konceptualizovanje ljudskih aktivnosti u okviru informacionog sistema se uspešnije odvija ako se informacioni sistem posmatra u punoj širini, a ne ograničeno na tehničku dimenziju.

1.5 Formalizacija informacionih sistema

Formalizovan sistem predstavlja plansko prikupljanje i izdavanje informacija. Čine ga predefinisana pravila za rukovanje informacijama, pri čemu je razmena informacija formalizovana. Obično se za neke ili sve operacije rukovanja informacijama upotrebljavaju savremene informacione tehnologije. Za razliku od toga, *neformalizovan sistem* je predstavljen odsustvom planskog prikupljanja i izdavanja informacija i intuitivnim rukovanjem informacijama. On je primenljiv samo u okruženjima sa malom količinom informacija (kao i malim brojem subjekata, predmeta i postupaka). Razmena informacija je neformalna i nema upotrebe savremenih informacionih tehnologija.

Termin *informacione tehnologije* je savremeni izraz koji označava kombinaciju računarskih i telekomunikacionih tehnologija koje se upotrebljavaju u prikupljanju, obradi, skladištenju, razmeni i izdavanju informacija. Zavisno od konteksta, u informacije tehnologije se mogu ubrajati i druge tehnologije za rukovanje informacijama, na primer, tehnologije za proizvodnju ili skladištenje papirnih dokumenata, mikrofilmova i sl. Izraz *savremen informacione tehnologije* se najčešće

odnosi samo na računarske i telekomunikacione informacione tehnologije. U daljem tekstu ćemo pod pojmom "informacione tehnologije" podrazumevati samo "savremene informacione tehnologije".

Što se tiče odnosa informacionih sistema i informacionih tehnologija, informacioni sistemi ne moraju koristiti savremene informacione tehnologije. Informacione tehnologije (i posebno računarske tehnologije) ne predstavljaju osnovni aspekt informacionih sistema. Da bi informacioni sistem koristio informacione tehnologije, mora biti formalizovan, i to bar u meri u kojoj koristi informacione tehnologije. Jedan od najvećih problema pri uvođenju informacionih sistema je upravo formalizacija postupaka u poslovnom okruženju.

Sada ćemo dati tri definicije informacionog sistema.

- *Informacioni sistem* je sistem koji prikuplja, skladišti, obrađuje i isporučuje informacije relevantne za poslovnu organizaciju, tako da su informacije dostupne i upotrebljive svima koji ih koriste ([4]).
- *Informacioni sistem* je uređeni sistem koji prikuplja, skladišti, obrađuje i isporučuje informacije o stanju domena ([3]).
- *Informacioni sistem* je uređeni sistem koji čine ljudi, podaci, procesi i interfejsi koji međusobno sarađuju na podršci i unapređivanju svakodnevnih operacija u poslovanju i omogućavaju rešavanje problema i donošenje odluka upravi i korisnicima ([2]).

1.6 Poslovni aspekti informacionih sistema

Neki od ciljeva primene informacionih tehnologija u informacionom sistemu su: povećanje profita, unapređenje kontrole, pojednostavljivanje poslova službenicima, ušteda vremena, ušteda materijalnih resursa, kvalitetnije informacije za donošenje odluka, i proširivanje broja učesnika u poslovanju.

Uloga informacionih sistema se obično posmatra iz poslovnog ugla. Uloga informacionih sistema u poslovnoj organizaciji može predstavljati osnovu za klasifikovanje informacionih sistema. Jedan informacioni sistem obično ima više uloga, bilo kao celina ili preko različitih podsistema. Neke od najčešćih uloga (vrsta) informacionih sistema su: upravljanje resursima, obrada transakcija, podrška proizvodnji, podrška odlučivanju, podrška planiranju, informisanje (interno i eksterno), upravljanje dokumentima i administrativnim poslovima i druge uloge.

U zavisnosti od konteksta i uloge, informacioni sistem može obuhvatiti više različitih vrsta aplikacija. Neke od osnovnih vrsta aplikacija informacionih sistema su:

aplikacije za obradu transakcija (aplikacije za unos podataka i baratanje transakcijama), analitičke aplikacije (aplikacije koje rade zbirnu obradu podataka ili izračunavaju statistike, pa samim tim pomažu u raznim odlučivanjima), ekspertski sistemi (aplikacije koje pokušavaju da same donose odluke na osnovu rezultata nekih drugih aplikacija), automatizacija poslovanja (aplikacije koje pokušavaju same da odrade neki posao), pretraživači informacija, i druge aplikacije. Informacioni sistemi se obično klasifikuju prema najvažnijim vrstama aplikacija.

Značaj pojedinačnih uloga informacionih sistema zavisi od domena poslovne organizacije. Prema domenima poslovnih organizacija (i značaju pojedinačnih uloga) razlikuju se vrste informacionih sistema. Neke od osnovnih vrsta informacionih sistema su: poslovni, proizvodni, naučni, zdravstveni, obrazovni, geografski, veb-zasnovani.

1.6.1 Poslovni informacioni sistemi

Poslovni informacioni sistemi su oni informacioni sistemi koji su prvenstveno namenjeni komunikaciji između različitih delova firme i omogućavaju osnovno upravljanje transakcijama, pregledanje različitih situacija, davanje informacija itd.

Prema vidu primene, među poslovnim informacionim sistemima razlikujemo:

- elektronsko poslovanje,
- sistem za obradu transakcija,
- upravljački informacioni sistem (uz podvrstu sistemi za upravljanje resursima),
- sistem za podršku odlučivanju, i
- sisteme posebne namene.

Elektronsko poslovanje obuhvata četiri aspekta: posao – posao (kada informacioni sistem jedne firme komunicira sa informacionim sistemom druge firme bez nečijeg učešća), posao – klijent (omogućavanje direktne komunikacije između poslovnog sistema i klijenta), posao – uprava (isti sistem proizvodi razne analitičke obrade i isporučuje ih upravi), uprava – klijent (kada sistem omogućava komunikaciju uprave i klijenta, na primer, naručivanje proizvoda za veliku firmu).

Transakcija predstavlja razmenu nekih dobara. *Sistemi za obradu transakcija* su oni kojima je primarna delatnost podrška velikog broja transakcija (potencijalno raznorodnih, i na različitim bazama). Pod ovakvim sistemima podrazumevamo naredne procese: poslovne razmene (podataka, novca, proizvoda), održavanje podataka, obavljanje jednostavnih ali kritičnih poslova, i obavljanje jednostavnih izračunavanja. Najočigledniji primer ovakvih sistema je bankarski informacioni

sistem. Ovi sistemi su ranije bili jedini ili najvažniji sistemi, a danas predstavljaju jedne od najvažijih sistema.

Upravljački informacioni sistemi su oni koji, pre svega, služe za obaveštavanje upravljačkih struktura o dobrim i relevantnim informacijama. Oni podrazumevaju naredne procese: održavanje redovnih informacija o redovnim aktivnostima, kontrola i efikasnost poslovanja, korišćenje transakcionih podataka kao osnovni izvor informacija, i pravljenje izveštaja (koji mogu biti redovni, zahtevani, i izuzetni).

Sistemi za podršku odlučivanju omogućavaju interaktivnu podršku za rešavanje problema i donošenje odluka po pitanjima koja nisu redovna i svakodnevna. Po pravilu su mnogo fleksibilniji i moćniji nego uobičajeni upravljački sistemi, što podrazumeva: više mogućnosti, povećanu slobodu u definisanju zahteva, i to da krajnji korisnici moraju da budu bolje poslovno i tehnički obučeni.

Pod *sistemima posebne namene* spadaju: sistemi veštačke inteligencije, sistemi virtualne realnosti, ekspertski sistemi, kao i drugi specifični poslovni informacioni sistemi. Primer sistema posebne namene je *Google Maps*.

1.7 Funkcionalni aspekti informacionih sistema

Posmatranje informacionih sistema iz funkcionalnog ugla stavlja u centar pažnje tri *funkcije informacionih sistema*. Svaki informacioni sistem ima tri osnovne funkcije:

- memorijsku funkciju,
- informativnu funkciju, i
- aktivnu funkciju.

Memorijska funkcija informacionih sistema podrazumeva održavanje slike stanja domena. Odvija se na dva osnovna načina: po zahtevu, i autonomno. Po zahtevu se vrši eksplicitno, odnosno, kada se stanje domena promeni, korisnik na odgovarajući način informiše sistem o nastaloj promeni. Autonomno se vrši implicitno, odnosno, sistem bez eksplicitnih akcija korisnika prepoznaje i evidentira nastale promene stanja domena.

Informativna funkcija se odnosi na pružanje informacija o stanju domena. I ona se odvija po zahtevu, i autonomno. Po zahtevu, kada je korisniku potrebna neka informacija, on je eksplicitno zahteva. Autonomno, kada sistem bez eksplicitnih akcija korisnika obaveštava odgovarajuće korisnike o trenutnom stanju domena ili nastalim promenama. Dodatno, informativna funkcija može biti: osnovna, kada se

vrši isporučivanje osnovnih podataka o stanju, i složena, kada se vrši isporučivanje rezultata analiza podataka.

Aktivna funkcija se odnosi na izvođenje akcija koje menjaju stanje domena. I ona se odvija po zahtevu, i autonomno. Po zahtevu, korisnik eksplicitno nalaže sistemu da izvede potrebnu akciju. Autonomno, sistem bez eksplicitnih akcija korisnika, a na osnovu odgovarajućih promena stanja domena, izvodi određene akcije.

1.8 Strukturni aspekti informacionih sistema

Posmatranjem informacionih sistema iz strukturnog ugla sagledavaju se strukturni elementi koji čine informacioni sistem. Elementi se prepoznaju na različite načine: u zavisnosti od ugla posmatranja problema, i u zavisnosti od širine posmatranja problema.

Možemo reći da su *osnovni elementi* informacionih sistema: subjekti (ljudi), podaci, procesi, i interfejsi.

Subjekt je bilo koja osoba koja ima interest od informacionog sistema. Vrste subjekata informacionih sistema su: vlasnici, korisnici, projektanti, implementatori, analitičari sistema, i proizvođači informacionih tehnologija i konsultanti.

- *Vlasnici* su sponzori i glavni zagovornici informacionog sistema. Oni su odgovorni za finansiranje razvoja i održavanja informacionog sistema, i oni su ti koji postavljaju ciljeve.

- *Korisnici* su osobe koje redovno neposredno koriste informacioni sistem (iz ugla razvoja informacionog sistema, koriste se termini "klijenti" ili "mušterije"). Mogu biti: interni korisnici, mobilni i udaljeni korisnici, i spoljašnji korisnici. Interni korisnici su zaposleni u poslovnoj organizaciji. To su: osnovni korisnici informacionog sistema, tehničko i stručno osoblje informacionog sistema, rukovaoci "znanjem", i uprava. Internet je omogućio pomeranje granica tradicionalnih informacionih sistema tako da uključe druge organizacije ili direktne klijente kao spoljašnje korisnike. To su: kupci, dostavljači, partneri, i zaposleni koji rade od kuće ili na putovanju. Udaljeni korisnik je korisnik koji nije fizički lociran uz organizaciju koja koristi informacioni sistem, ali koji zahteva pristup tom sistemu. Mobilni korisnik je korisnik čija se lokacija stalno menja, ali koji zahteva pristup informacionom sistemu sa bilo koje lokacije.

- *Projektanti* su osobe koje prevode poslovne zahteve i uslove sistema u tehnička rešenja. Oni projektuju baze podataka, ulazne i izlazne procese, aplikativne i korisničke interfejse, računarske mreže, programe i ostale potrebne vidove informacionih tehnologija.

- *Implementatori* su osobe koje proizvode komponente informacionih sistema na osnovu projektnih specifikacija. Često su projektanti i implementatori isti subjekti.

- *Analitičari sistema* su osobe koje upravljaju razvojem i održavanjem informacionih sistema kroz održavanje komunikacije sa svim ostalim subjektima. *Analiza sistema* je istraživanje u domenu poslovnih problema radi prepoznavanja i pedlaganja konkretnih poslovnih zahteva za uvođenje unapređenja u postojeći informacioni sistem. Analiza sistema mora da se odvija u kontekstu tehničkih mogućnosti i da daje rezultate koji su tehnički ostvarivi u konkretnom poslovnom okruženju. Poslovni analitičari se razlikuju od sistemskih analitičara po tome što analize poslovnih problema obavljaju nezavisno od tehničkih aspekata (poslovna analiza).

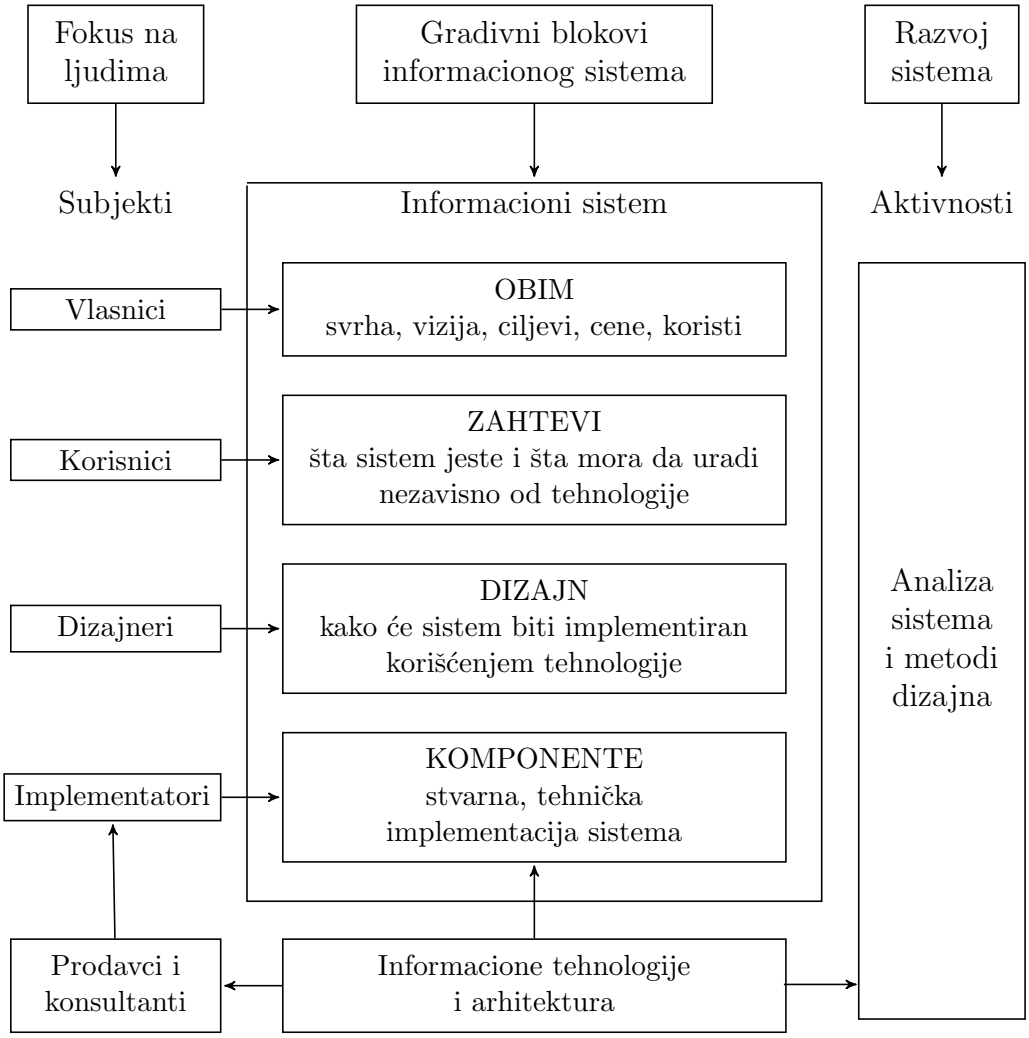
- *Proizvođači informacionih tehnologija i konsultanti* su osobe koje pružaju savetodavne, proizvodne i druge usluge u odnosu na specifične informacione tehnologije. Primena tzv. "visokih" informacionih tehnologija je često nemoguća bez neposrednog učešća eksperata za konkretne informacione tehnologije.

Na slici 1.1 dat je grafički prikaz vrsta subjekata u informacionom sistemu.

Podaci informacionog sistema obuhvataju: podatke koji su predmet poslovanja, podatke o procesu poslovanja, podatke o korisnicima sistema, podatke o implementaciji. Podacima se posvećuje velika pažnja pri razvoju informacionih sistema. Podaci imaju trajnost i promene strukture podataka su mnogo složenije i neugodnije od promena softvera. Podaci su u centru pažnje metodologija orijentisanih prema podacima.

Procese informacionog sistema predstavljaju svi poslovi i postupci koje informacioni sistem implementira ili čije odvijanje podržava. Procesi predstavljaju jednu od osnovnih strukturnih komponenti informacionih sistema. Za značajan broj razvojnih metodologija procesi su u centru pažnje pri razvoju. Jedan od osnovnih ciljeva je podrška poslovnih procesa na način koji je što intuitivniji i prirodniji.

Informacioni sistemi sadrže dve vrste *interfejsa*. To su: korisnički interfejsi, i aplikativni interfejsi. *Korisnički interfejsi* predstavljaju osnovne korisničke alate za upotrebu informacionih sistema. Veoma su važni za uspešnu implementaciju i odvijanje poslovnih procesa. *Aplikativni interfejsi* su osnova za povezivanje informacionog sistema sa drugim informacionim sistemima (B2B). Značaj aplikativnih interfejsa u savremenom računarstvu je sve veći.



Slika 1.1: Vrste subjekata informacionih sistema.

1.9 Pitanja

1. Objasniti razliku između informacije i podataka. Navesti poželjne karakteristike informacija.
2. Šta je sistem?
3. Koji su osnovni aspekti informacionih sistema?
4. Objasniti poslovni aspekt informacionih sistema.
5. Objasniti strukturni aspekt informacionih sistema.
6. Objasniti funkcionalni aspekt informacionih sistema.
7. Objasniti socijalnu dimenziju informacionih sistema.

8. Šta su formalizovani i neformalizovani elementi informacionog sistema?
9. Šta su informacione tehnologije?
10. Navesti i objasniti bar jednu definiciju pojma “informacioni sistem”.
11. Navesti bar pet najčešćih uloga informacionih sistema?
12. Navesti bar pet vrsta informacionih sistema?
13. Navesti i objasniti osnovne funkcije informacionih sistema?
14. Navesti i objasniti osnovne strukturne elemente informacionih sistema?
15. Navesti i objasniti vrste subjekata informacionih sistema?

1.10 Literatura

Literatura za ovu oblast su poglavlja 1 i 2 iz [2], kao i poglavlje 1 iz [3].

Glava 2

Razvojne metodologije

2.1 Uloga razvojnih metodologija

Razvoj informacionih sistema je veoma složen i osetljiv proces iz više razloga. Postoji veliki broj uključneih subjekata, njihova različitost, kao i različitost njihovih ciljeva. Zatim, veličina i složenost sistema koji se modelira, različite funkcije sistema, dinamičnost i nestalnost sistema, tehnički i poslovni aspekti.

Rane faze razvoja softvera (do 1960-ih godina) odlikovale su se nepostojanjem širokih metodologija. Vremenom su uočene neke suštinske potrebe, kao što su: potreba za analizom i projektovanjem sistema, kao vidom prethodnice programiranja, a samim tim i za novim subjektima – analitičarima sistema, zatim, potreba za prelaskom sa koncepta slabo povezanih kolekcija pojedinačnih rešenja na koncept jače integrisanih informacionih sistema, i potreba za usvajanjem široke i detaljne metodologije razvoja koja bi predstavljala okvir za sve faze razvoja.

Osnovni ciljevi uvođenja metodologija su: bolji krajnji rezultat, bolji razvojni proces, i standardizovan razvojni proces.

Da bismo uveli pojam metodologije, pogledajmo prvo šta sve jedna metodologija treba da obuhvati. Metodologija razvoja informacionih sistema je skup procedura, tehnika, alata i dokumentacionih tehnika koji pomažu razvijaocima sistema u njihovim naporima da izgrade nov informacioni sistem. Metodologija opisuje tok razvoja kroz faze, podfaze i korake razvoja informacionih sistema i predlaže tehnike i alate koje se koriste u različitim fazama i koracima razvoja. Pored samog razvoja, metodologija predlaže i načine planiranja, praćenja i procenjivanja uspešnosti procesa razvoja.

Primeri definicija metodologije su:

- *Metodologija* je preporučen skup filozofija, faza, procedura, pravila, tehnika, alata, dokumentacije, upravljanja i obuke za razvojaoce informacionih sistema (Medison, 1983).

- *Metodologija* je skup principa i metoda, koji se u konkretnoj situaciji svodi na metode koji su najbolje prolagođeni toj konkretnoj situaciji (Čeklend, 1981).

Metodologije nam pružaju preporučene odgovore na neka pitanja, kao što su: "Koji poslovi se obavljaju u tom koraku?", "Koje rezultate je potrebno napraviti?", "Kada i pod kojim uslovima se rade koji poslovi?", "Koji uslovi se moraju ispoštovati?", "Koji ljudi se moraju uključiti?", "Kako se rukovodi razvojem?", "Kako se ostvaruje kontrola kvaliteta?", i "Koji alati se upotrebljavaju?". U skladu sa time, svaka metodologija se odlikuje svojim elementima. Ti elementi su:

- *Ideja vodilja* (ili, *filozofija*). Ona predstavlja osnovnu ideju kojom se rukovodi metodologija, odnosno, metodologija se gradi u cilju ostvarivanja ili praćenja osnovne ideje. Neke metodologije se više usmeravaju socijalnom aspektu, neke pak strukturnom aspektu, i dr. Neki uprošćeni primeri filozofija su: "dobro rešenje je sistem koji pruža najveću iskorišćenost računara", "dobro rešenje je sistem koji obezbeđuje najbolju dokumentaciju", "dobro rešenje je sistem koji je najjevtiniji u eksploataciji", i "dobro rešenje je sistem koji se dopada vlasnicima".

- *Ciljevi.* Različitost filozofija ima za posledicu da metodologije mogu da imaju i različite ciljeve. Cilj metodologije usmerava pažnju razvojnog tima na odgovarajuće aspekte sistema i utiče na izbor tehnika i alata.

- *Faze životnog ciklusa* (ili, *faze razvoja*). Svaka metodologija deli proces razvoja u neke faze kroz koje mora proći razvoj informacionih sistema. Sadržaj i oblik faza, kao i njihov propisan tok, predstavljaju posledicu filozofije i cilja metodologije. Faze razvoja značajno utiču na primenjivost metodologija u rešavanju konkretnih problema.

- *Tehnike koje se primenjuju u određenim fazama.* Tehnika je način obavljanja nekog posla. Često se isti posao može uraditi primenom različitih tehnika. Različite metodologije predlažu različite tehnike. Na primer, tabla se može okačiti na zid tehnikom lepljenja ili tehnikom zakucavanja. Tehnike obuhvataju: načine istraživanja (intervjui, upitnici, itd.), načine predstavljanja (dijagrami, dokumentacija, itd.), načine projektovanja (tranzicija jednog oblika predstavljanja u drugi, povezivanje različitih dijagrama, itd.), i dr.[1]

[1] Više reči o tehnikama će biti u poglavlju 3.

- *Alati koji se upotrebljavaju u tehnikama.* Alat je sredstvo za primenu neke tehnike. Različite tehnike zahtevaju različite alate. Na primer, lepljenje zahteva lepak i ravnalo, a zakucavanje zahteva eksere i čekić. Neki alati podržavaju pojedinačne tehnike, a neki alati podržavaju čitave metodologije. Upotreba dobrih alata olakšava dinamične procese analiziranja i projektovanja.

- *Proizvodi* (ili *artefakti*). Proizvod je svaki rezultat rada koji nastane tokom razvoja informacionog sistema. Svaka primena neke tehnike u nekom razvojnom koraku ima za cilj stvaranje odgovarajućeg proizvoda. Proizvodi mogu predstavljati deo informacionog sistema ili neophodan preduslov za naredne korake. Primeri proizvoda mogu biti: baza podataka, formular, uputstvo za upotrebu, konceptualni model sistema, dijagram toka podataka, dijagram slučajeva upotrebe, itd.

- *Uloge subjekata u razvoju.*

2.2 Životni ciklusi informacionih sistema

Životni ciklus informacionog sistema predstavlja skup razvojnih faza, njihov sadržaj i redosled odvijanja. Životni ciklusi su posebno značajni u strukturnim metodologijama (strogo opisuju redosled aktivnosti). U daljem tekstu ćemo upoznati nekoliko različitih životnih ciklusa.

2.2.1 Model vodopada

Životni ciklus razvoja sistema (engl. *System Development Life Cycle – SDLC*) jedan je konkretan pristup konceptu životnih ciklusa. Prvi put je predložen kasnih 1960-ih godina. Poznat je i po imena *kovencionalna analiza sistema* i *model vodopada*, pri čemu ćemo u daljem tekstu koristiti drugi naziv. Takođe, postoji više varijanti onog pristupa, a mi ćemo obraditi jednu od njih.

Životni ciklus po modelu vodopada se sastoji od faza kroz koje informacioni sistem prolazi u predefinisanom redosledu, pri čemu je ime dobio po jednosmernom kretanju kroz faze. Sada ćemo se detaljno upoznati sa svakom od faza[2].

- *Studija izvodljivosti* (engl. *feasibility study*). Uloga ove faze je sagledavanje okvira (odnosno, granica) u kojima se razmatra unapređenje postojećeg

[2]Napomenimo dve stvari. Prvo, u različitih literaturama, ove faze se nazivaju različito, pri čemu, u suštini, svaki odabir naziva ima za zadatak da ukratko objasni celu fazu. Odabir naziva koji je predstavljen u ovom tekstu upravo to i čini. Drugo, u različitim literaturama broj faza može biti drugačiji. Na primer, u nekim literaturama se (ovde razdvojene) faze *projektovanje sistema* i *implementacija* sažimaju u jednu fazu. Cilj svakog izlaganja jeste da se prepoznaju glavni elementi ovog pristupa, a konkretna podela na faze nije u velikoj meri značajna.

ili pravljenje novog sistema. Ishod ove faze je predlog rešenja. Ono što sledi nakon ove faze je odluka upravljačkih struktura da li se prihvata predloženo rešenje, zamisao o promeni odbacuje, ili zahteva nova studija izvodljivosti. Studija izvodljivosti obuhvata osnovnu analizu postojećeg stanja (pre svega, pažnja se usmerava na manjkavosti), željenog novog stanja (pre svega, unapređenja), i različitih aspekata (kao što su pravni, organizacioni, socijalni, tehnički, i ekonomski aspekti). Kao rezultat studije se oblikuje zaključak u obliku predloženog idejnog rešenja. Zaključak bi trebalo da pokaže da li je izvođenje moguće iz svih posmatranih aspekata.

- *Istraživanje sistema* (engl. *systems investigation*). Uloga ove faze je detaljno sagledavanje postojećeg stanja i potrebnih promena. Ishod ove faze je iscrpna slika stanja. Istraživanje sistema obuhvata iscrpno posmatranje sistema, što se odnosi na funkcionalne zahteve postavljene pred postojećim sistemom (analiza ispunjenosti i problemi), funkcionalne zahteve koji se postavljaju pred novim sistemom, dodatne zahteve i uslove, vrste i obim obuhvaćenih podataka, i posebne slučajeve. Metodi istraživanja sistema su: posmatranje (uslova rada, uskih grla, metoda, i sl.), intervjui (mogu biti individualni i grupni; služe radi prikupljanja informacija o sistemu, radi provere tačnosti prikupljenih informacija, i radi sagledavanja prostora za promene i stvaranje istog), upitnici (koriste se ako više ljudi radi isto posao ili ako su dislocirani), pregledanje postojeće dokumentacije, i posmatranje uzoraka materijala na kojima se radi.

- *Analiza sistema* (engl. *system analysis*). Uloge ove faze su: razumevanje postojećeg sistema, razumevanje problema, razumevanje prostora za unapređenja, i oblikovanje zahteva. Ishod ove faze su postavljeni zahtevi. Analiza sistema obuhvata analizu prikupljenih podataka o sistemu. Ovi podaci se odnose na pitanje postojanja problema, pitanje primenjivanja baš postojećih postupaka (posebno, da li postoje alternative i zašto nisu primenjene?), i pitanje načina menjanja obima podataka u vremenu.

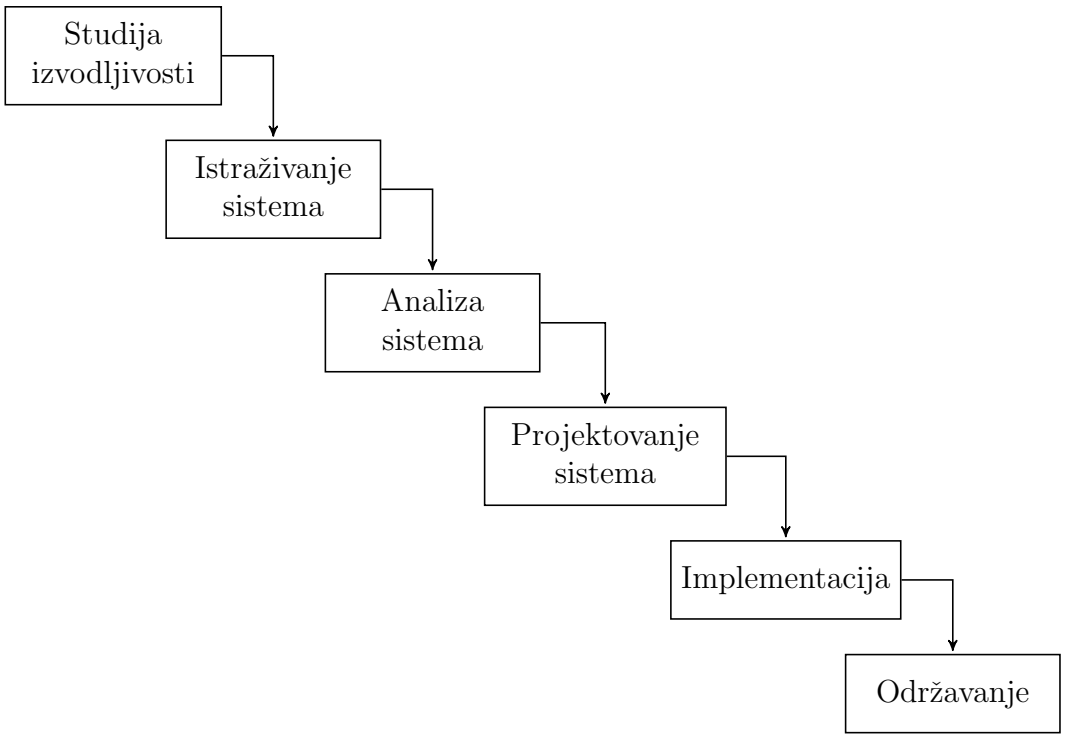
- *Projektovanje sistema* (engl. *system design*). Uloga ove faze je priprema za implementaciju. Ishod ove faze je iscrpna dokumentacija o svim aspektima predstojeće implementacije sistema. Projektovanje sistema obuhvata iscrpno dokumentovanje: ulaznih podataka, izlaznih podataka, toka procesa u sistemu, strukture i organizacije podataka (na primer, projekat baze podataka, specifikacije dokumenata, i sl.), sigurnosnih aspekata i procedura, plana implementacije, plana testiranja, i plana tranzicije. Naglasimo da se projekat može značajno razlikovati od onog koji je nastao u okviru studije izvodljivosti.

- *Implementacija* (engl. *implementation*). Uloga ove faze je izgradnja novog (unapređenog) sistema. Ishod ove faze je napravljen nov sistem u skladu sa zahtevima. Implementacija obuhvata: pravljenje svih elemenata novog

sistema (što se odnosi na softver, hardver, ali i na organizacione uslove za primenu), kontrolu kvaliteta (odnosno, testiranje i stalne provere korektnosti), edukaciju budućih korisnika, pripremu dokumentacije (korisničke i razvojne), implementaciju sigurnosnih mehanizama, i tranziciju sa starog na novi sistem.

- *Održavanje* (engl. *maintenance*). Uloga ove faze je očuvanje saglasnosti sistema sa okruženjem. Ishod ove faze je sistem prilagođen okruženju. Održavanje obuhvata: stalno sagledavanje saglasnosti organizacionih aspekata okruženja i postojećeg informacionog sistema, usklađivanje informacionog sistema sa promenama u okruženju, i uočavanje i otklanjanje naknadno prepoznatih nedostataka.

Na slici 2.1 prikazana je jedna varijanta modela vodopada. Ova varijanta predstavlja grafički prikaz prethodno pomenutih faza. Svaka faza je reprezentovana pravougaonikom sa nazivom faze u okviru njega, a svaka usmerena strelica predstavlja moguće prelaske između faza. Posledica prikazane varijante jeste da jednom kada je faza završena, zabranjuje se ponovno vraćanje na nju. U dodatku 12.8, grafički su prikazane različite varijante modela vodopada, uz dodatne (kratke) napomene.



Slika 2.1: Grafički prikaz jedne varijante modela vodopada.

Dobre strane modela vodopada su sistematičnost (posebno ukoliko uzmemo u obzir kontekst vremena u kome je nastao), proveren pristup, standardizovanost, planski

pristup problemu razvoja, i činjenica da specijalizacija po fazama olakšava upravljanje.

Ipak, model vodopada nije savršen. Štaviše, on sadrži veći broj potencijalnih slabosti. Iz činjenice da se potrebe vlasnika neposredno razmatraju samo u prvoj fazi razvoja, javlja se mogućnost njihovog "promašivanja". S obzirom da se najviše pažnje u prvim fazama posvećuje procesima, a procesi su promenljivi, to je ovaj model nestabilan. On ima i odliku nefleksibilnosti, odnosno, mnogo toga se zacrtava u ranim fazama i teško se naknadno menja. Nezadovoljstvo korisnika se može javiti zbog složene upotrebe sistema (uzrokovane ostalim slabostima), mnogo specijalnih slučajeva, itd. S obzirom da dokumentacija počiva na ranim fazama, a rezultat se kasnije menja, dolazi do neusklađenosti dokumentacije. Zbog velike raslojenosti je veoma teško unapred proceniti obim pojedinih faza što predstavlja nedostatak kontrole. Pojedini specijalni slučajevi se često ne razmotre na vreme, što dovodi do nekompletnosti sistema. Dugo se čeka na gotov sistem. Usled dugog trajanja razvojnog procesa, sve promene uslova nastale u međuvremenu se moraju ispratiti u fazi održavanja, odnosno, obično mnogo posla ostaje za fazu održavanja. Ovaj model može dovesti do pojave idealizacije. Idealizacija u projektovanju često ima za posledicu preterano komplikovanje i odlaganje suštinski važnih stvari zbog manje značajnih sitnica. Akcenat je na ozbiljnom razmišljanju o činjenicama, a one su nestalne. Konačno, model se vodi pretpostavkom "belog papira", odnosno, koncept je prilagođen razvoju novih sistema, ali najčešće se sistemi ne prave "od nule", već se menja i unapređuje neko postojeće okruženje.

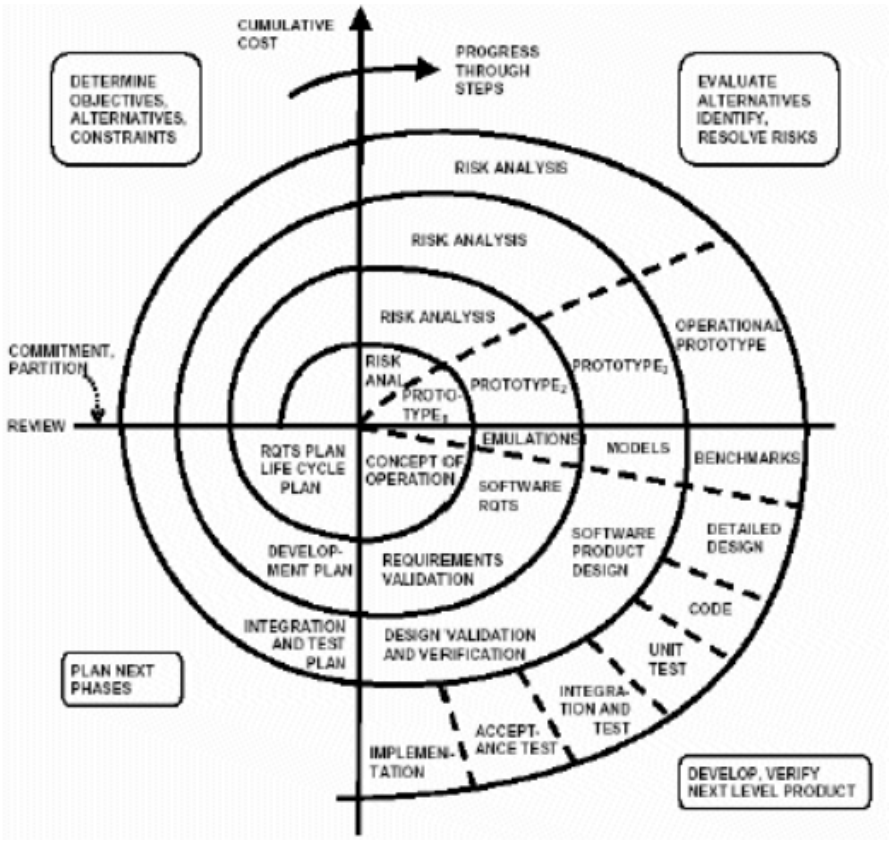
2.2.2 Drugi oblici životnih ciklusa

Slabosti modela vodopada se pokušavaju rešiti drugim oblicima životnih ciklusa. Postoje dve osnovne vrste takvih ciklusa: *spiralni životni ciklusi*, i *ciklični životni ciklusi*. U ciklične životne cikluse spadaju: *iterativni*, i *evolutivni*. Nekim primerima ćemo se baviti detaljnije u daljem tekstu, a na ovom mestu ćemo ukratko objasniti razlike između ovih pojmova i grafički prikazati varijante spiralnog modela.

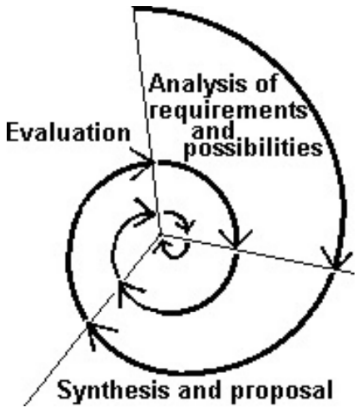
Razlika između spiralnog i cikličnog modela je u tome što se u spiralnom modelu menja ciklus, odnosno, ne ostaje baš isti "spisak" faza u svakom ciklusu kroz koji prolazimo. U cikličnom modelu ipak, svaki put prolazimo kroz iste faze. Naravno, sâm projekat neće prolaziti kroz iste faze.

Postoje različite vrste cikličnog modela, odnosno, postoje vrste definisane na osnovu dva smera. Neki modeli od njih polaze iz centra, tj. ideja je da polazimo ni od čega, pa polako povećavamo ono što smo napravili. Jedan takav model je prikazan na slici 2.2. Ono što ostaje "u krugu", to je napravljeno. Možemo primetiti da se neke od faza ponavljaju (na primer, analiza rizika), ali to ne važi za sve (na primer, u prvom koraku se na jednom mestu prolazi kroz fazu emulacije, pa u

drugom koraku na istom mestu kroz fazu izrade modela, zatim u trećem koraku kroz fazu testiranja brzine, itd.).



Slika 2.2: Primer modela spiralnog životnog ciklusa u smeru “ka spolja”.



Slika 2.3: Primer modela spiralnog životnog ciklusa u smeru “ka unutra”.

Kod nekih drugih modela, kao što je prikazano na slici 2.3, imamo smer kruženja prema centru. Ideja je da pravimo sve manje krugove dok sami ne odlučimo da

smo "stigli u centar", čime je posao završen. U ovakvom modelu se obično ponavljaju skoro iste faze, uz neke manje izmene.

Iterativni i evolutivni model su danas veoma izraženi u praktičnoj upotrebi, i to u različitim metodologijama, i u različitim oblicima. Za razliku od cikličnih, spiralni model se vrlo ređe koristi.

2.3 Pitanja

1. Šta je razvojna metodologija?
2. Navesti primere definicija razvojnih metodologija.
3. Šta čini metodologiju razvoja informacionih sistema?
4. Šta predstavlja filozofija metodologije razvoja informacionih sistema?
5. Šta predstavljaju ciljevi metodologija razvoja informacionih sistema?
6. Šta predstavljaju faze razvoja jedne metodologije razvoja informacionih sistema?
7. Šta predstavljaju i čemu služe tehnike metodologije razvoja informacionih sistema?
8. Šta su proizvodi metodologije razvoja informacionih sistema?
9. Šta je životni ciklus informacionih sistema?
10. Objasniti životni ciklus informacionih sistema po modelu vodopada. Navesti osnovne faze.
11. Navesti ulogu, ishod i sadržaj faze životnog ciklusa informacionih sistema "studija izvodljivosti"?
12. Navesti ulogu, ishod i sadržaj faze životnog ciklusa informacionih sistema "istraživanje sistema"?
13. Navesti ulogu, ishod i sadržaj faze životnog ciklusa informacionih sistema "analiza sistema"?
14. Navesti ulogu, ishod i sadržaj faze životnog ciklusa informacionih sistema "projektovanje sistema"?
15. Navesti ulogu, ishod i sadržaj faze životnog ciklusa informacionih sistema "implementacija"?

16. Navesti ulogu, ishod i sadržaj faze životnog ciklusa informacionih sistema "održavanje"?
17. Objasniti dobre strane životnog ciklusa informacionih sistema po modelu vodopada.
18. Objasniti slabosti životnog ciklusa informacionih sistema po modelu vodopada.
19. Kada nije rizično primeniti u praksi životni ciklus informacionih sistema po modelu vodopada?
20. Koje su karakteristike spiralnih životnih ciklusa?

2.4 Literatura

Literatura za ovu oblast su odabrana poglavlja iz [1].

Glava 3

Tehnike

3.1 Opšte o tehnikama

Informacioni sistem se pravi radi podržavanja nekog posla (ili više njih) u okviru nekog preduzeća. Posao je, na primer, upis studenta na fakultet. To je u suštini jedan složen proces koji obuhvata prijavljivanje studenta, održavanje prijemnog ispita, unošenje osvojenih poena, rangiranje, i drugih (možemo reći jednostavnijih) procesa.

U ranim fazama razvoja informacionih sistema, osnovna prepoznatljiva celina u informacionom sistemu je bio proces. Kada je postojao neki sistem u kojem već postoji nekakva baza podataka ili kolekcija fajlova, zadatak programera je bio da napravi skup programa (na primer, koristeći programski jezik COBOL) koji obrađuju te podatke. Time se zapravo opisuju neki od poslovnih procesa, odnosno, izvršavaju se konkretni poslovi na modelu podataka. Dakle, rane metodologije su prvenstveno bile orijentisale ka procesima i strukturi. U tom metodologijama je opisivanje procesa bilo najvažnija stvar.

Već smo rekli da je tehnika način obavljanja nekog posla. Prema predmetu bavljenja i konceptu na kome počivaju, tehnike se dele na: tehnike opisivanja procesa, tehnike opisivanja podataka (imaju za cilj način opisivanja modela podataka, odnosno, nad čime radimo, koji su to podaci koji dobro modeliraju neku stvar), objektno-orijentisane tehnike (opisuju podatke i procese odjedanput, korišćenjem objektnog modela), holističke tehnike (kojima je cilj da prikažu što više različitih stvari na jednom modelu), tehnike upravljanja projektom, organizacione tehnike, i kadrovske tehnike.

3.2 Tehnike opisivanja procesa

Tehnike opisivanja procesa imaju veliki značaj u razumevanju i modeliranju sistema. Procesi predstavljaju jednu od osnovnih komponenti sistema. Modeliranje

procesa omogućuje da se bolje razumeju odnosi subjekata i objekata u sistemu. Ipak, procesi nisu stabilni i kroz životni vek sistema trpe relativno česte promene.

Postoje veliki broj tehnika za opisivanje procesa. Oni se dele u tri osnovne kategorije: tehnike za opisivanje poslovne logike (pravila odlučivanja), tehnike za praćenje projekta, i tehnike za opisivanje dekompozicije (strukture).

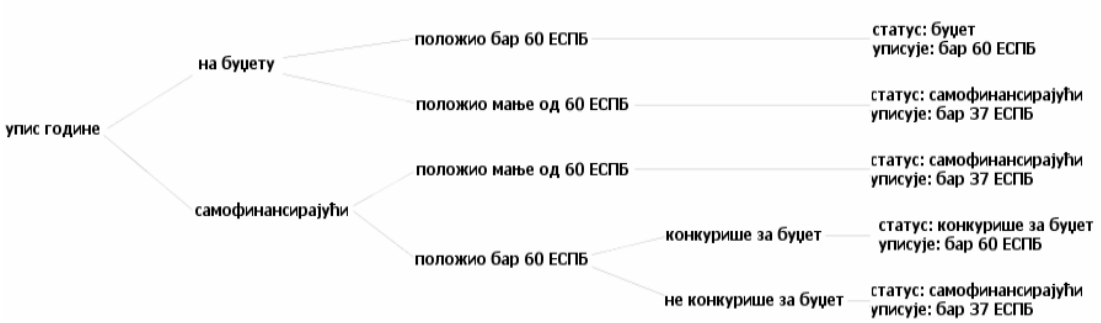
3.2.1 Tehnike za opisivanje poslovne logike

Ove tehnike opisuju poslovna pravila i objašnjavaju način donošenja odluka. Najčešće se odnose na pojedinačne procese, ali mogu da se odnose na više procesa. U njih spadaju: drvo odlučivanja, tabele odlučivanja, struktuiran tekst, i dijagrami akcija.

3.2.1.1 Drvo odlučivanja

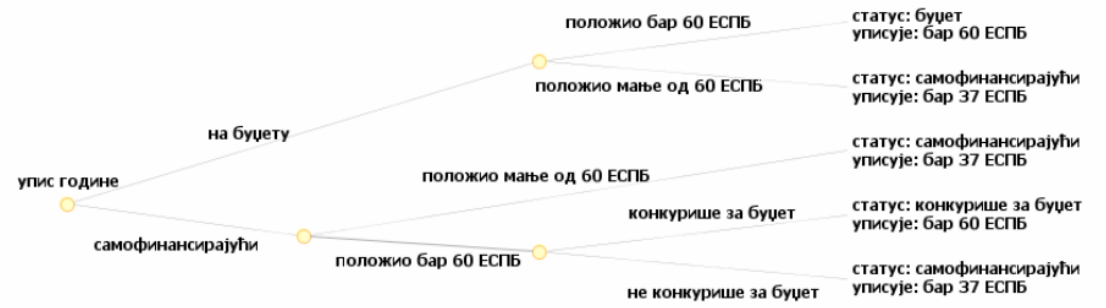
Tehnika se upotrebljava za modeliranje načina odlučivanja, i to relativno uopšteno, ali pre svega pri izboru procesa koji se primenjuje.

Način odlučivanja se predstavlja u obliku drveta. U korenu drveta je opšti slučaj. Svakim granjanjem se opštiji slučaj deli na više specifičnih. U listovima su poznate *odluke*, tj. opisi procesa koji se primenjuju. Obično se grananje predstavlja udesno (mada može i nadole). Na slici 3.1 prikazan je primer drveta odlučivanja za slučaj "upis godine".



Slika 3.1: Primer drveta odlučivanja za slučaj "upis godine".

Postoje različite grafičke reprezentacije drveta odlučivanja, pri čemu je za sve njih zajedničko da se jasno mogu razumeti pri njihovom čitanju, odnosno, sagledavanju. Na primer, na slici 3.2 dat je primer drveta u kojem se stanja ne nalaze u čvorovima (kao na slici 3.1), već su čvorovi predstavljeni kružićima, a stanja se nalaze na granama. Ovim se dobija nešto (prostorno) sažetiji prikaz.



Slika 3.2: Alternativna grafička reprezentacija drveta odlučivanja za slučaj "upis godine".

U objektno-orijentisanim metodologijama se kao alternativa u nekim slučajevima upotrebljavaju dijagrami aktivnosti ili dijagrami stanja.

3.2.1.2 Tabele odlučivanja

Slično drvetu odlučivanja, tabele odlučivanja se upotrebljavaju za modeliranje načina odlučivanja. Nadopunjuje se sa drvetom odlučivanja. Međtuim, u nekim slučajevima neka od ovih tehnika nije pogodna.

Odlučivanje se predstavlja tabelom, podeljenom na dva dela: gornji i donji. U gornjem delu se predstavljaju uslovi, tako da vrste odgovaraju pojedinačnim uslovima. U donjem delu se predstavljaju akcije, tako da vrste odgovaraju akcijama. Kolone tabele odgovaraju skupovima uslova. Naime, u preseku kolone i vrste koja odgovvara uslovu se označava da li je uslov ispunjem, a u preseku kolone i vrste koja odgovara akciji se označava da li se akcija primenjuje. Na slici 3.3 dat je primer tabele odlučivanja za slučaj "upis godine" (analogan primerima sa slika 3.1 i 3.2).

Status je budžet	DA	DA	ne	ne	ne
Status je samofinansiranje	ne	ne	DA	DA	DA
Položeno bar 60 ESPB	DA	ne	DA	DA	ne
Položeno manje od 60 ESPB	ne	DA	ne	ne	DA
Konkuriše za budžet	–	–	DA	ne	–
Upisuje budžet	X				
Upisuje "konkuriše za budžet"			X		
Upisuje samofinansiranje		X		X	X
Mora da upiše bar 60 ESPB	X		X		
Mora da upiše bar 37 ESPB		X		X	X

Slika 3.3: Primer tabele odlučivanja za slučaj "upis godine".

Tabela odlučivanja bi trebalo da obuhvati sve kombinacije uslova koje imaju smisla. Akcije mogu odgovarati procesima. Istim uslovima može odgovarati više različitih akcija, ali i ista akcija može odgovarati različitim uslovima.

3.2.1.3 Struktuiran tekst

Opisivanje pomoću struktuiranog teksta namenjeno je za modeliranje odlučivanja, procesa, ali i drugih stvari. Struktuirani tekst ima sličnosti sa meta-programskim jezikom. Na primer, sada već dobro poznat slučaj "upis godine" može se pomoću struktuiranog teksta opisati na sledeći način:

```
ako je student na budžetu
    i položio je bar 60 ESPB
    onda upisuje godinu na budžetu, sa bar 60 ESPB
inače, ako je položio manje od 60 ESPB
    ili iz bilo kog razloga ne konkuriše za budžet
    onda upisuje godinu kao SF, sa bar 37 ESPB
inače dobija status ``konkuriše za budžet'' i
    upisuje bar 60 ESPB
```

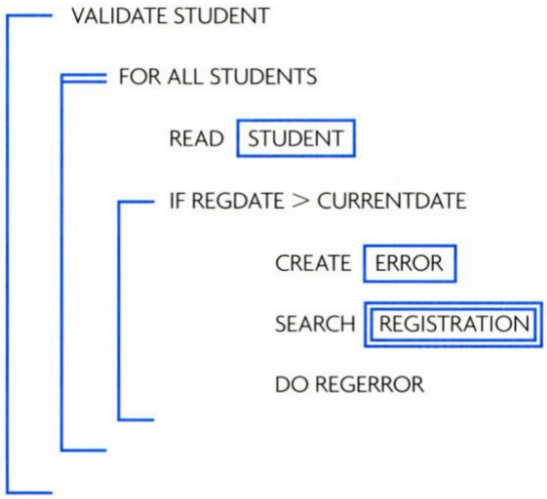
3.2.1.4 Dijagrami akcija

Dijagrami akcija imaju sličnosti sa struktuiranim tekstom. Mogu se koristiti na više nivoa: za pregledno predstavljanje, i za detaljno predstavljanje.

Osnovni element dijagrama je *blok*. Predstavlja se kao zagrada sa leve strane struktuiranog teksta, i odgovara bloku u programskom jeziku ili strukturnom tekstu. Postoje i posebna označavanja:

- Ponavljanja se označavaju dvostrukom gornjom linijom.
- Paralelizacije se označavaju povezivanjem paralelnih blokova lučnom linijom.
- Čitave akcije ulaza i izlaza se zaokružuju pravougaonikom. Ulazi se navode iznad, a izlazi ispod.
- Operacije koje koriste bazu podataka se zaokružuju pravougaonikom. Ako rade nad više podataka, zaokružuju se dvostrukim pravougaonikom.

Na slici 3.4 dat je primer dijagrama akcije slučaja "validacija studenta", na kojem su predstavljeni blokovi od kojih neki imaju posebnia označavanjima.



Slika 3.4: Primer dijagrama akcija slučaja "validacija studenta".

3.2.2 Tehnike za praćenje projekta

Tehnike za praćenje projekta se bave procesom razvoja informacionog sistema. One opisuju sistematično praćenje toka razvoja. Postiču kritičko sagledavanje rada. Doprinose blagovremenom uočavanju problema.

Najvažnija tehnika za praćenje projekta je strukturni pregled kojom ćemo se nešto detaljnije pozabaviti u nastavku.

3.2.2.1 Strukturni pregled

Strukturni pregledi (engl. *structured walkthrough*) niz su formalnih pregleda sistema, koji se odvijaju u različitim fazama životnog ciklusa. Potičku iz strukturnih metodologija. Osnovna ideja je da se redovno sprovode kritičke analize svih značajnih odluka, i to: u vidu sastanaka i diskusija, u vidu komisijskih analiza, ali i na bilo koji drugi način. Osnovni cilj je da se potencijalni problemi uoče što je moguće ranije, kako bi se umanjile negativne posledice.

Dobre strane ove tehnike su: unapređenje ukupnog kvaliteta sistema, omogućava se rano uočavanje problema, svi članovi tima imaju priliku da se upoznaju sa različitim aspektima sistema, tehničke informacije se razmenjuju u vidu diskusija i izveštaja i omogućava se kritičko sagledavanje problema i odluka, pregledi predstavljaju dobre tačke preseka (engl. *milestones*), mogu da pomognu u razvoju timskog duha, i podstiču stručni razvoj članova tima.

Jedan od najboljih pristupa je da se odluke preispituju kroz neformalne diskusije članova tima. Time se održava duh zajedništva, dobija se veća efikasnost, i lakše

izražavanje sumnje u neku odluku nego u formalnim uslovima.

Strukturni pregledi se obično odvijaju na kraju faze životnog ciklusa ili nekog značajnog koraka u okviru faze. Na primer, posle: izrade studije izvodljivosti, istarživanja sistema, analize sistema, projektovanja sistema, određivanja specifikacije programa, projektovanja logike programa, implementacije, izrade plana testiranja, izrade plana implementacije, izrade plana izvođenja radova, pisanja korisničkih uputstava za rad, i ciklusa revizije i održavanja.

U kontekstu proveravanja ispravnosti programa postoje dva dijametralno različita pristupa: čitanje koda, i izvršavanje programa.

3.2.3 Tehnike za opisivanje strukture procesa

Tehnike za opisivanje strukture procesa se bave strukturom procesa, i to: dekompozicijom procesa na jednostavnije potprocese, analizom sastavnih delova procesa (obuhvaćenih funkcija, povezanih podataka, itd.), i vremenskim međuzavisnostima. One nam pružaju informacije za planiranje i projektovanje.

U njih spadaju: dijagram toka podataka, matrice, dijagrami strukture, životni ciklus entiteta, i dijagrami BPMN.

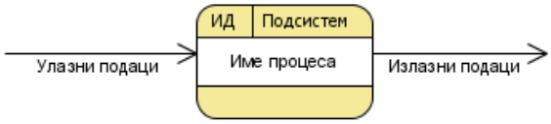
3.2.3.1 Dijagram toka podataka

Dijagram toka podataka predstavlja jednu od osnovnih tehnika strukturnih metodologija (ali se odnosi na opisivanje procesa). Opisuje na koji način se kreću podaci kroz sistem. Time se prepoznaju i glavne komponente sistema.

Omogućava: modeliranje poslovne strukture okruženja, pravljenje funkcionalne dekompozicije sistema, modularno predstavljanje okruženja, i koncizno grafičko predstavljanje (koje je razumljivo svim subjektima). Osnovna slabost je da se iz ovog dijagrama ne vidi eksplicitno vremenski tok događaja.

Osnovni elementi dijagrama su: tok podataka, procesi, skladišta podataka, izvori i ishodi. Grafički izgled se razlikuje, zavisno od izvora i metodologije.

Procesi se predstavljaju kao zaobljeni pravougaonici, što je prikazano na slici 3.5. ID predstavlja numeričku identifikaciju procesa. Podsistem predstavlja naziv (ili oznaku) podsistema u kome se proces odvija, i on može biti zapisan i u dnu, a ne iznad. Procesi se vezuju strelicama koje nazivamo *tokovi podataka*. Znači, neki podaci teku iz jednog procesa u drugi. Osim između procesa, tokovi podataka mogu da se kreću i između skladišta i procesa, ili procesa i izvora (odnosno, ishoda). Svaki proces mora da ima bar jedan ulazni i bar jedan izlazni tok.



Slika 3.5: Elementi procesa u dijagramu toka podataka.

Skladišta podataka se predstavljaju pravougaonicima otvorenim na desnoj strani. Uobičajeno je da se vrsta skladišta označava u kvadratu na levoj strani pravougaonika, kao što je prikazano na slici 3.6. Skladišta mogu biti: datoteke, baza podataka, fizička skladišta, i drugo.

Slika 3.6: Primer skladišta podataka u dijagramu toka podataka.

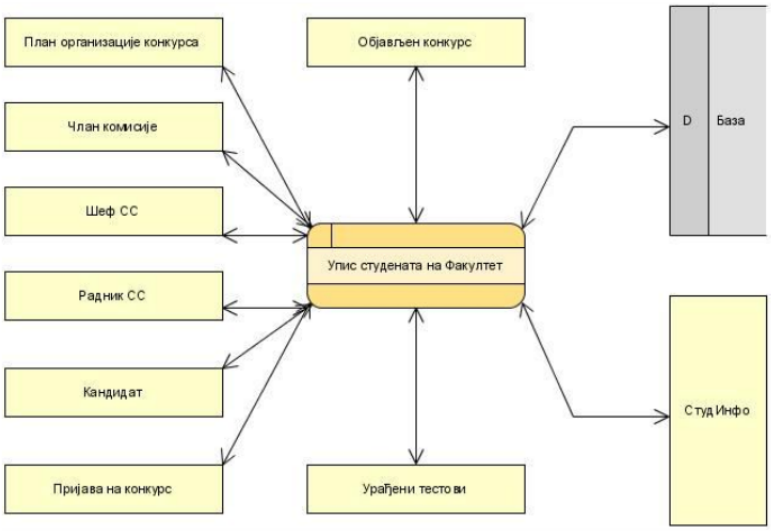
Izvor ili *ishod* je nešto što nije skladište podataka, a može da predstavlja nekakav cilj ili nekakvo poreklo ulaznih podataka (dakle, spoljni entiteti i subjekti). Na primer, to može biti korisnički terminal. Ako čekamo da korisnik upiše neke podatke, onda je u pitanju izvor. Ako želimo da nešto odštampamo na korisnički terminal, onda je u pitanju izhod. Izvori i ishodi se predstavljaju kao pravougaonici. Na slici 3.7 prikazan je sistem koji sadrži proces "naručivanje nedostajućih proizvoda" koji, pored izvoza rezultata u bazu podataka (skladište podataka), izvozi rezultate rada i snabdevaču (ishod).

Slika 3.7: Primer ishoda u dijagramu toka podataka.

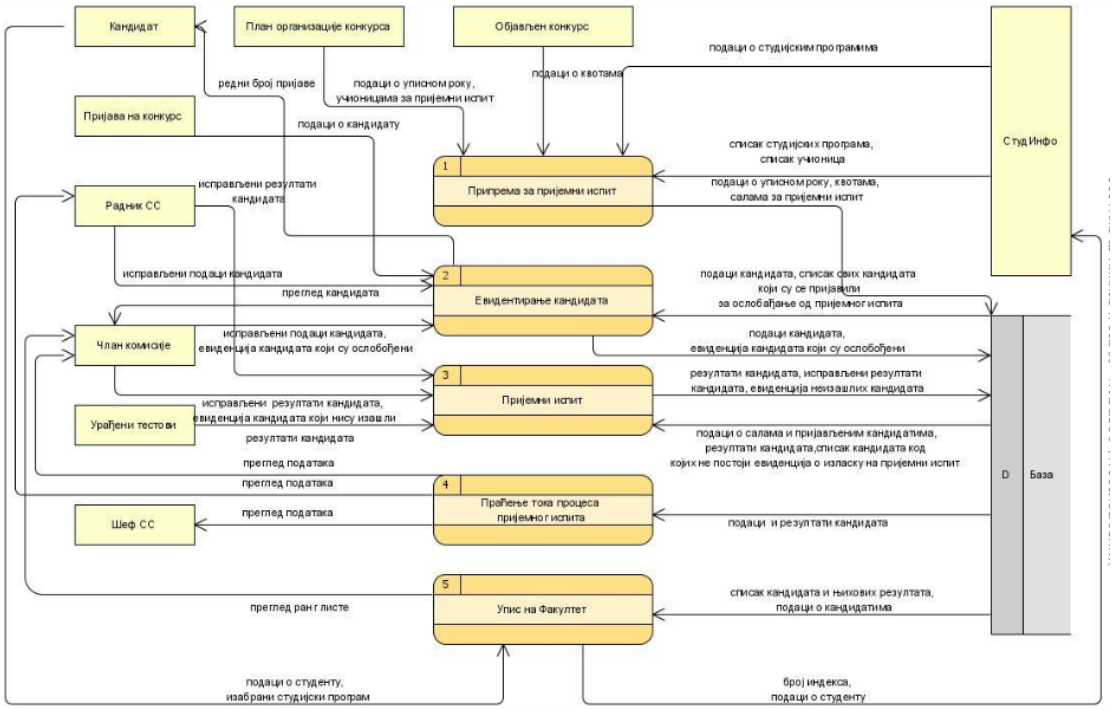
Prema Masonu (*Mason*, 1994), pravila oblikovanja dijagrama toka podataka su: detaljno upoznavanje sa procesom, prepoznavanje izvora i ishoda, prepoznavanje skladišta podataka, i prepoznavanje procesa.

Razlikujemo nivoe dijagrama toka podataka. Dijagram konteksta predstavlja čitav sistem (podsistem) kao jedan proces. U dijagramu toka podataka nivoa nula,

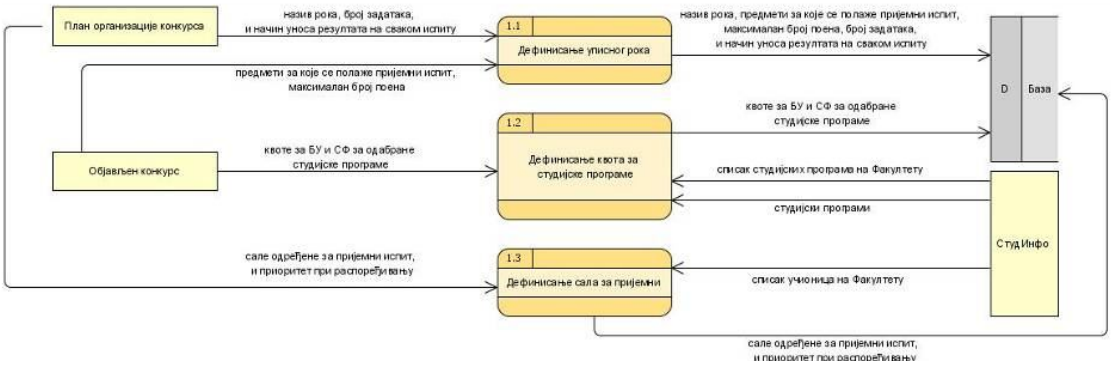
razlikuju se glavni procesi u čitavom sistemu (podsistemu). U dijagramu toka podataka nivoa jedan, za svaki proces iz dijagrama toka podataka nivoa nula se pravi poseban dijagram toka podataka. Po potrebi se prave i dijagrami nižih nivoa. Na slikama 3.8, 3.9, i 3.10 prikazani su različiti nivoi dijagrama toka podataka procesa "upis studenata na fakultet".



Slika 3.8: Dijagram konteksta.



Slika 3.9: Dijagram nivoa 0, hijerarhijska dekompozicija sistema.



Slika 3.10: Dijagram nivoa 1, priprema za prijemni ispit.

Rezimirajmo: dijagrami toka podataka opisuju dobro strukturu složenih procesa, kao i funkcionalnu dekompoziciju. Međutim, oni ne opisuju dobro vremensku međuzavisnost procesa.

3.2.3.2 Matrice

Matrice se upotrebljavaju za modeliranje i predstavljanje odnosa u veoma različitim slučajevima. Neke od primena obuhvataju predstavljanje: odnosa poslova i funkcija, odnosa entiteta (podataka) i funkcija, odnosa funkcija i subjekata, odnosa entiteta (podataka) i subjekata, i drugo.

Što se tiče predstavljanja odnosa poslova i funkcija, namene matrica su: dokumentovanje odnosa poslova i funkcija, prepoznavanje funkcija sa lokalnim i širim značajem, i prepoznavanje atomičnih funkcija.

Kolone predstavljaju poslove, a vrste funkcije (ili obratno). Poslovi obuhvataju krupnije postupke ili događaje koji se dešavaju u okviru domena. Funkcije predstavljaju pojedinačne (potencijalno složene) operacije. Poželjno je da funkcije budu atomične (tj. da se svaka funkcija upotrebljava samo kao celina, a nikada delimično). Svakom poslu mora da odgovara bar jedna funkcija. Broj i veličina tablica zavisi od veličine i složenosti sistema. Idealno je da se svaki posao i svaka funkcija pojavljuju u tačno po jednoj tablici, u suprotnom su moguće greške usled redundantnosti. Primer matrice odnosa poslova i funkcija dat je slikom 3.11.

Posao / Funkcija	Upis na fakultet	Upis godine	Ispisivanje sa fakulteta
Prijavljivanje kandidata	X		
Upisivanje na fakultet	X		
Promena statusa		X	
Izbor predmeta	X	X	
Upisivanje godine	X	X	
Određivanje školarine	X	X	
Proveravanje dugovanja		X	X
Evidentiranje uplate	X	X	X

Slika 3.11: Primer matrice posao–funkcija.

Što se tiče predstavljanja odnosa entiteta i funkcija, namene matrica su: dokumentovanje odnosa entiteta i funkcija, i prepoznavanje entiteta sa lokalnim i širim značajem.

Kolone predstavljaju entitete, a vrste funkcije (ili obratno). Svaka funkcija mora da upotrebljava bar jedan entitet. Funkcija može upotrebljavati entitet na različite načine (PČMB, engl. *CRUD*), i oni su: pravljenje (funkcija pravi novi entitet, engl. *create*), čitanje (funkcija čita postojeći entitet, engl. *read*), menjanje (funkcija (potencijalno) menja postojeći entitet, engl. *update*), i brisanje (funkcija briše postojeći entitet, engl. *delete*). Broj tablica zavisi od veličine i složenosti sistema. Primer matrice odnosa entiteta i funkcija dat je slikom 3.12.

Entitet / Funkcija	Prij. na konkurs	Indeks	Prijava za upis godine	Upisana godine	Upisan kurs	Fin. obaveza
Prijavljivanje kandidata	P					
Upisivanje na fakultet	Č	P				
Promena statusa		ČM				
Izbor predmeta			P			
Upisivanje godine			ČM	P	P	
Određivanje školarine		Č		ČM	Č	
Proveravanje dugovanja		Č				Č
Evidentiranje uplate		Č				ČM

Slika 3.12: Primer matrice entitet–funkcija.

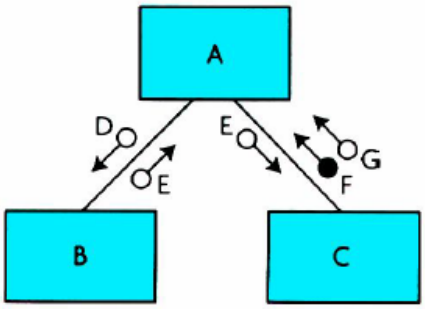
Što se tiče predstavljanja odnosa procesa i subjekata, namena matrice je sagledavanje koje funkcije se upotrebljavaju u kojim organizacionim jedinicama.

Što se tiče predstavljanja odnosa entiteta i subjekata, namena matrice je sagledavanje koji podaci se upotrebljavaju u kojim organizacionim jedinicama. Upotrebljava se pretežno za: označavanje entiteta koji se ad-hok održavaju (na primer, neki podaci o službenicima u kadrovskoj službi), i označavanje organizacionih jedinica koje čitaju podatke (na primer, uprava čita podatke radi pravljenja izveštaja).

3.2.3.3 Dijagrami strukture

Dijagrami strukture predstavljaju funkcionalnu dekompoziciju problema (hijerarhijska organizacija). Struktura se predstavlja grafički: pravougaonicima se predstavljaju procesi ili moduli, linijama se povezuju procesi (odnosno, moduli), i uz linije se navode podaci koji se prenose.

Na primer, na slici 3.13 dat je primer dijagrama strukture iz kojeg se vidi da modul A može da koristi (poziva) module B i C. Međutim, redosled poziva se ne vidi iz ovog dijagrama. Pri pozivanju se prosleđuju naznačeni podaci. Popunjen kružić označava *zastavicu* koja utiče na dalji tok operacija.



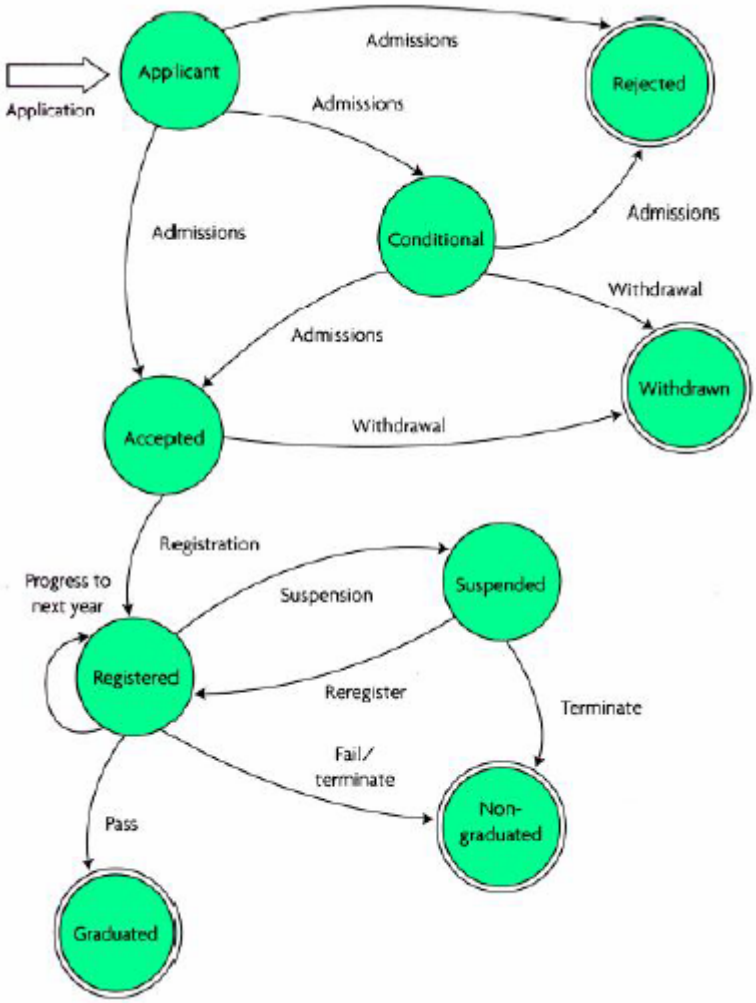
Slika 3.13: Primer dijagrama strukture.

Dijagrami strukture se koriste u različitim metodologijama. Teži se minimalnoj međuzavisnosti modula (spregnutost) i višestrukoj upotrebi modula (kohezija).

3.2.3.4 Životni ciklus entiteta

Životni ciklus entiteta se upotrebljava za opisivanje stanja kroz koja prolazi neki entitet. Odgovara dijagramu stanja UML-a[3]. Na slici 3.14 dat je primer dijagrama životnog ciklusa entiteta.

[3]Dijagrami stanja biće obrađeni detaljnije u okviru objektno-orijentisanih tehnika (3.4).



Slika 3.14: Primer životnog ciklusa entiteta.

Jedna jednostavna notacija podrazumeva sledeće. Svako stanje je označeno krugom u koji se smešta naziv stanja. Početno stanje je stanje nastajanja entiteta (stanje u koje ulazi posebno označena strelica; na slici 3.14 to je strelica sa oznakom "Application"). Završno stanje (može ih biti više) označeno je dvostrukom kružnicom. Strelice označavaju procese, odnosno, poslove koje prevode entitet iz jednog stanja u drugo.

3.2.3.5 Modeliranje poslovnih procesa – BPMN dijagrami

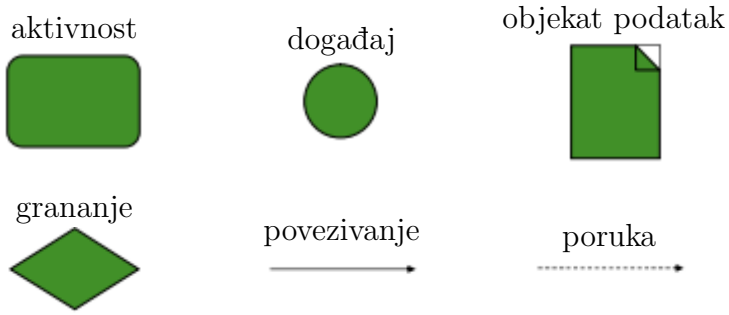
Većina tehnika koje smo do sada upoznali su "stare" tehnike, u smislu da su prethodile objektno-orijentisanim tehnikama. Neke od njih su nastale tokom 1970-ih godina, većina njih tokom 1980-ih godina, ali se neke od njih koriste i danas.

Sa druge strane, *notacija za modeliranje poslovnih procesa* (engl. *Business Process Modeling Notation (BPMN)*) predstavlja napredniju notaciju za predstavljanje po-

slovnih procesa. Iako pod tehnikama koje su "novije" obično podrazumevamo objektno-orijentisane tehnike, na primer UML, BPMN je tehnika koja je novija i od UML-a. Razvijena je od strane *OMG* u periodu od 2001. do 2006. godine (verzija 1).

Naime, kada je izvršena tranzicija na objektno-orijentisane tehnike, zbog svog načina pristupa, cela priča o procesima je bila potpuno zapostavljena. U sistemu, poslovni procesi su veoma promenljivi jer na njih često utiče ogroman broj parametara. Ako procese posmatramo kao fiksne tačke u našem sistemu, odnosno, kao početna polazišta, postavlja se pitanje da li je to dobro. Mi time baziramo ceo sistem da počiva na nekoliko procesa koje smo posmatrali, a koji su veoma promenljivi. Objektno-orijentisane tehnike zbog toga u centar stavljaju objekte, koji nisu promenljivi na način na koji su procesi promenljivi. Međutim, ukoliko se potpuno zanemari proces, onda dolazi do toga da nije lako sagledati jedan veliki sistem. Zbog toga se na početku koristio dijagram toka podataka, kao sredstvo da se idejni (ili završni) projekat predstavi nekome tako da taj neko može da razume šta tu piše od samog pov cetka, a ne tek kad dođe do kraja. Zbog navedenih slabosti dijagrama toka podataka, nastala je nova tehnika, BPMN.

Osnovne vrste ovih dijagrama su: dijagram procesa, i dijagram saradnje. Oni su vrlo slični. Razlika je u tome što se dijagram procesa usmerava na predstavljanje procesa i njihovih odnosa, bez interesa o tome ko koji proces obavlja, dok je u dijagramu saradnje oslonac na grupisanju procesa radi pregledanja koji subjekat obavlja koji proces. Notacija obuhvata različite oblike zapisivanja nefunkcionalnih zahteva, kao što su: načini iniciranja, čekanje, slanje poruka, i drugo.



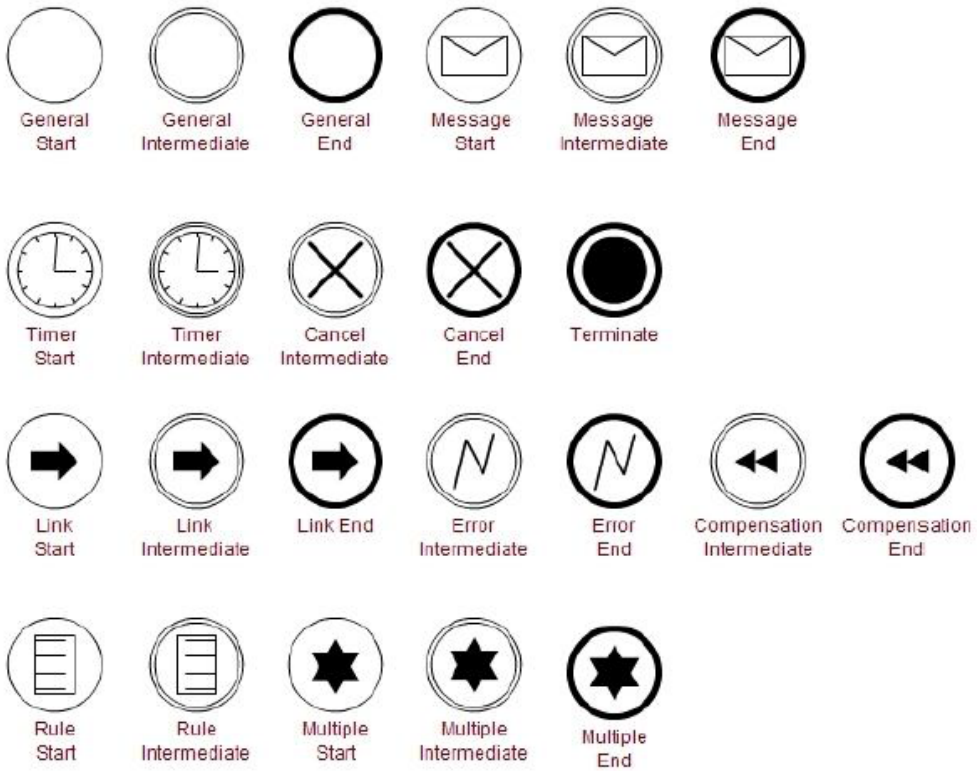
Slika 3.15: Osnovni elementi BPMN dijagrama.

Osnovni elementi BPMN dijagrama su: aktivnost, događaj, objekat podatak, grananje, povezivanje, i poruka (slika 3.15).

BPMN dijagrami koji su izgrađeni od *aktivnosti*, *grananja*, i *povezivanja* veoma liče na UML dijagrame aktivnosti. Ono što ih razlikuje su novi elementi, poput događaja. *Događaji* su razni faktori koji mogu da utiču na to da se neki proces

inicira, kao što i neki proces može da inicira nekakav događaj. Događaj može biti: tajmer (na primer, ako svakog dana u određeno vreme treba da se obavi neki posao), inicijator odgovora (na primer, ako jedan proces pošalje poruku koja inicira neku drugi proces), i drugo. Događaji nam pomažu da vidimo vremenski sled u sistemu. Kao što smo napomenuli, proces može da šalje drugom procesu *poruku*. Takođe, proces može da pošalje *objekat podatak* drugom procesu, ili da ga prihvati.

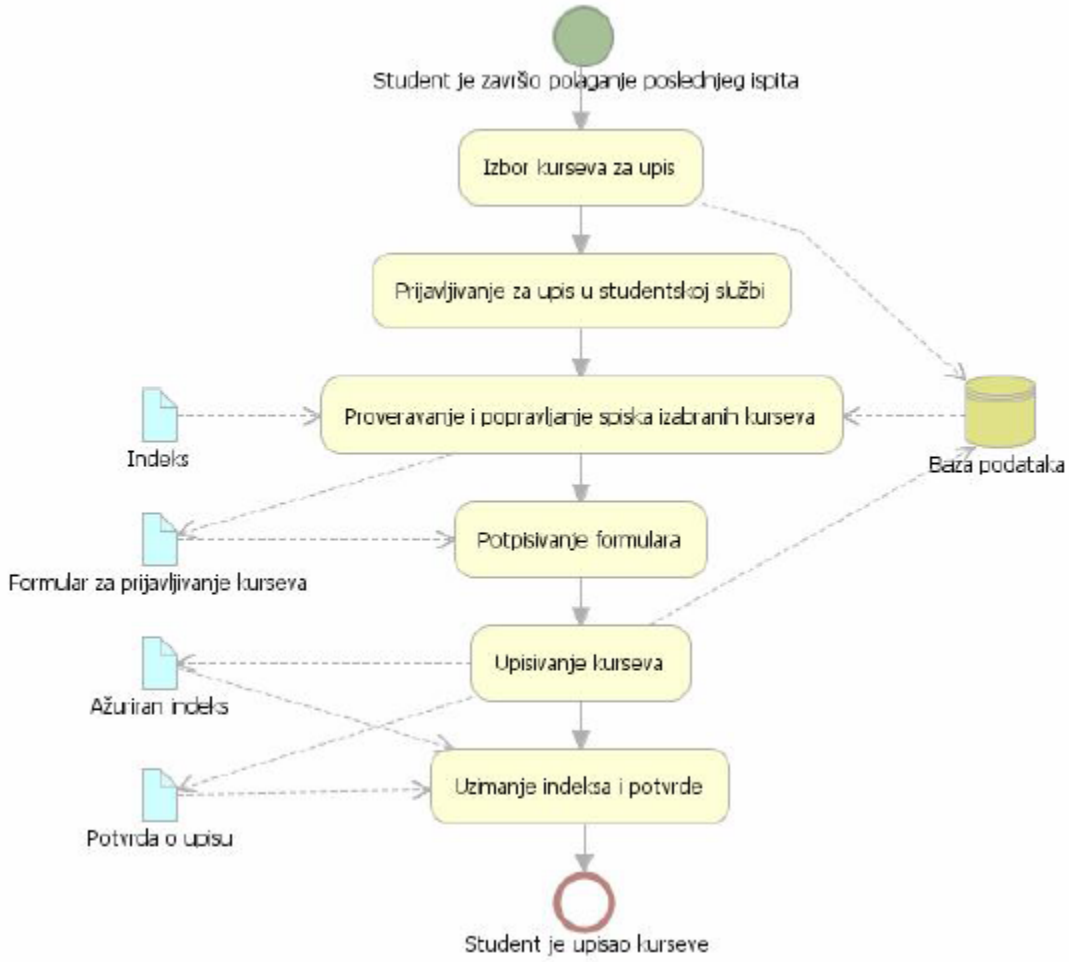
Ono što nedostaje u BPMN dijagramima, a postoji u dijagramu toka podataka jesu podaci. Nije nam vidljivo na koji način teku podaci koje koristimo i koje proizvodimo. Ovde na raspolaganju imamo samo objekte podatke koji se koriste između procesa.



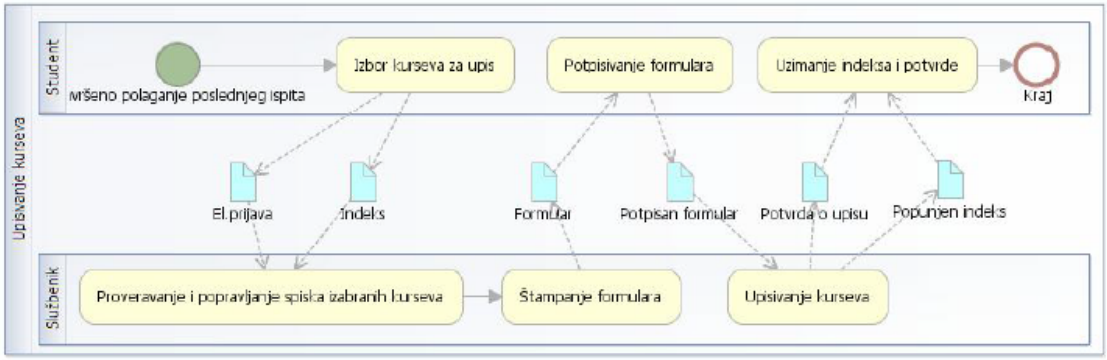
Slika 3.16: Dodatni simboli događaja.

Pored osnovnih, BPMN dijagrami imaju veliki broj dodatnih simbola. Neki od dodatnih simbola događaja prikazani su na slici 3.16. Možemo prepoznati tri osnovne vrste događaja: običan krug označava iniciranje procesa, dvostruki krug označava nekog događaja posrednika, i podebljani krug označava da time završavamo neki proces.

U nastavku su dati primeri nekih BPMN dijagrama.



Slika 3.17: BPMN dijagram procesa – primer.



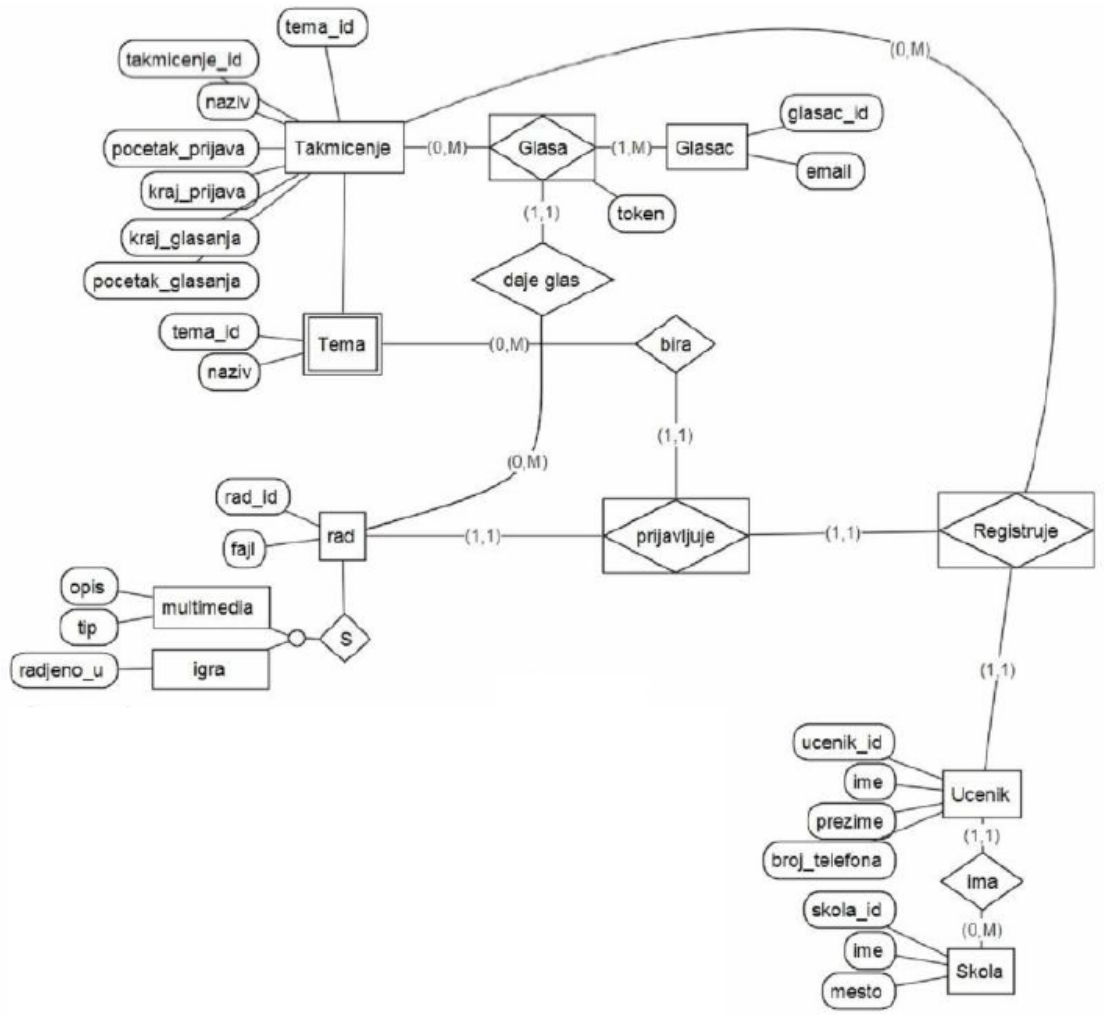
Slika 3.18: BPMN dijagram saradnje – primer.

3.3 Tehnike modeliranja podataka

Tehnike modeliranja podataka modeliraju strukturu podataka, i međusobne odnose podataka. Najvažnije tehnike su: dijagrami entiteta i odnosa, i dijagrami klasa podataka. Ako se dobro primenjuju, skoro su ekvivalentne.

3.3.1 Dijagrami entiteta i odnosa

Dijagrami entiteta i odnosa (engl. *Entity-Relationship (ER) diagrams*) su veoma značajna tehnika modeliranja i opisivanja podataka.



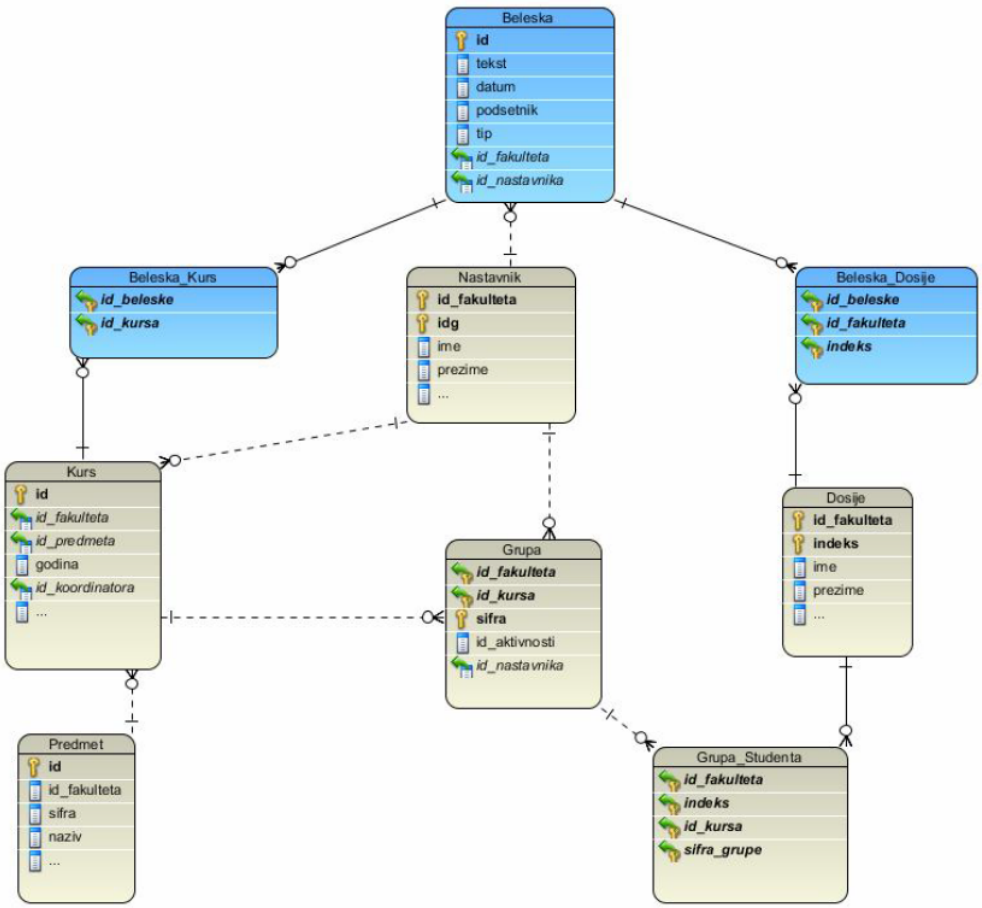
Slika 3.19: Dijagrami entiteta i odnosa – primer 1.

Osnovni koncepti ER modeliranja su entiteti i odnosi. *Entitet* je osnovni pojam koji se ne definiše, a čije je značenje stvar, biće, pojava ili događaj od značaja koji se može jednoznačno identifikovati. *Odnos* je neko međusobno pridruživanje skupova entiteta (na primer, otac–sin, radnik–preduzeće). Uloga entiteta u odnosu

je funkcija koju on obavlja u tom odnosu. Informacije o entitetima i odnosima se izražavaju skupom parova atribut–vrednost [10].

ER dijagrami su sastavni deo ER modela. Oni predstavljaju osnovni način zapisivanja semantičkih znanja o informacijama. Tipovi entiteta se predstavljaju pravougaonicima, atributi se predstavljaju elipsama, a primenjivost atributa na tip entiteta se predstavlja linijom. Odnosi se predstavljaju rombovima i mogu da imaju opisne atribute. Odnosi mogu da uključuju više tipova entiteta [10].

Na slici 3.19 dat je primer dijagrama entiteta i odnosa. Na slici 3.20 dat je, zapravo, dijagram koji se naziva *dijagram tabela*, međutim, u literaturi se često i takvi dijagrami smatraju pod dijagramima entiteta i odnosa. U daljem tekstu, kada kažemo "dijagram entiteta i odnosa" mislićemo prvenstveno na "dijagram tabela".

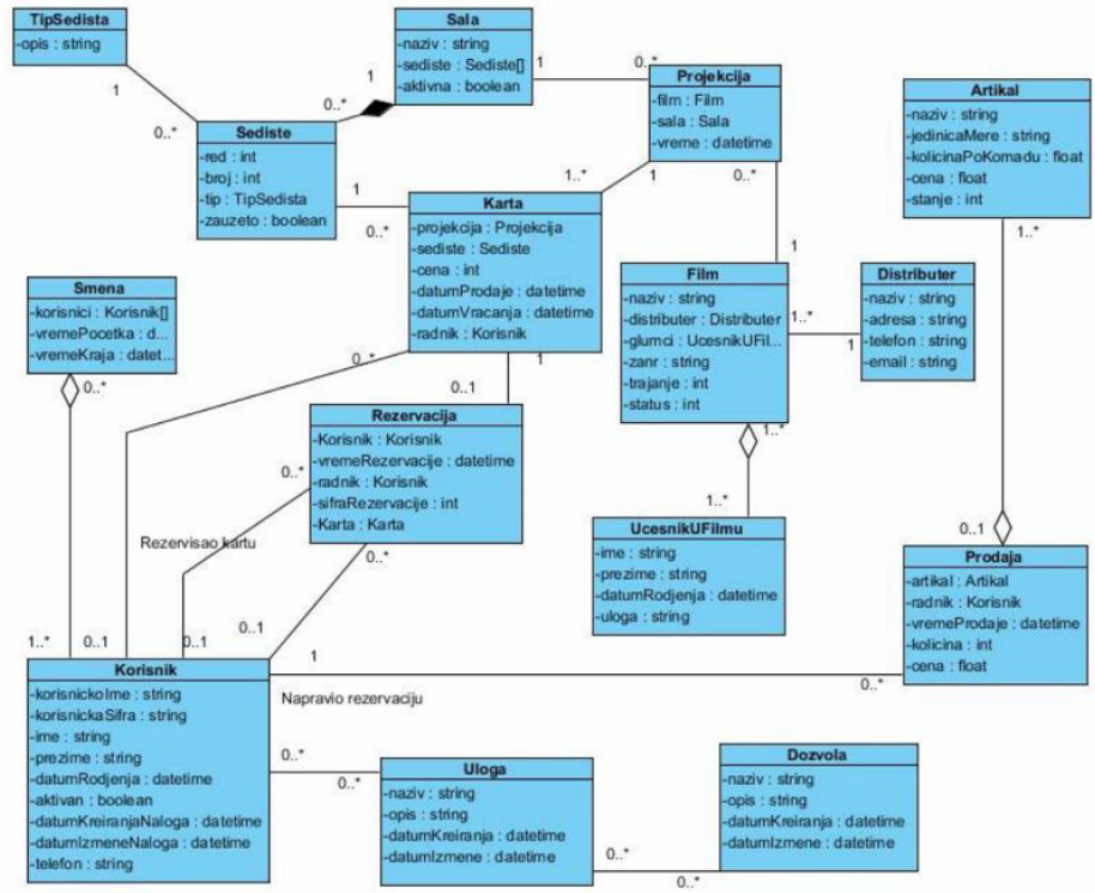


Slika 3.20: Dijagrami entiteta i odnosa – primer 2.

3.3.2 Dijagrami klasa podataka

Dijagrami klasa podataka se danas često upotrebljavaju umesto dijagrama entiteta i odnosa. Razlozi za to su sledeći: predstavljaju širu tehniku od dijagrama entiteta i odnosa, omogućavaju modeliranje nekih aspekata ponašanja, omogućavaju modeliranje hijerarhija entiteta, i modeliranje podataka je bliže aplikaciji nego bazi podataka.

Predstavljaju specifičan vid dijagrama klasa, tzv. *dijagrami klasa podataka* ili *dijagrami klasa entiteta*. Oni čine samo podskup koji se odnosi na strukturu i odnose, uz izostavljanje ponašanja klasa[4].



Slika 3.21: Dijagrami klasa entiteta – primer.

Dijagrami klasa podataka predstavljaju dobar način za opisivanje strukture na konceptualnom nivou. Neko se može zapitati zašto na konceptualnom nivou. Na primer, ne mora svakoj klasi odgovarati jedna tabela u relacionoj bazi podataka. Klasa je domen, odnosno, opis tipa podataka koje koristimo. Tabela u relacionoj

[4]Dijagrami klasa će biti detaljnije obrađeni kao objektno-orijentisana tehnika (3.4).

bazi podataka je skup. U jednom domenu može biti više skupova. Dakle, jednoj klasi može odgovarati veći broj kolekcija podataka tog tipa.

Takođe, hijerarhija klasa može biti implementirana na više načina. Prvi način je da jedna tabela može imati atribute iz cele hijerarhije uz stavljanje nedostajućih vrednosti (NULL) tamo gde vrednost atributa nema smisla. Drugi način je da se jedna tabela pravi za baznu klasu i jedna tabela se pravi za izvedenu klasu, uz postojanje stranog ključa. Treći način je pravljenje tabela samo za klase listove. Takve tabele obuhvataju atribute svih natklasa.

Kada se bavimo konceptom podataka, na ove navedene stavke ne treba da obraćamo pažnju. Ono što jeste bitno da prepoznamo je koje podatke treba čuvati, i kakvi su njihovi odnosi.

3.3.3 Izbor tehnike

Dijagrami entiteta i odnosa se neposrednije preslikavaju u implementaciju relacione baze podataka, što obuhvata: relacije, primarne i strane ključeve, implementaciju hijerarhija entiteta, bliže fizičkoj implementaciji.

Dijagrami klasa su bliži implementaciji aplikacije: eksplicitno sadrže hijerarhije, mogu da sadrže aspekte ponašanja (u smislu semantike odnosa), mogu da sadrže odnose klasa podataka sa klasama implementacije, apstraktniji su.

Iz dijagrama klasa se skoro jednoznačno može izvesti dijagram entiteta i odnosa (prepoznavanje tabela). Može da se bira između više načina implementiranja nasleđivanja (o čemu smo diskutovali). Obrnuto ne važi, zato što se različiti odnosi dijagrama klasa prevode u iste odnose na dijagramu entiteta i odnosa.

U fazi modeliranja sistema (kada se bavimo konceptom) obično je bolje da se upotrebljava dijagram klasa (ili izvorni dijagram entiteta i odnosa). U fazi logičkog ili fizičkog projektovanja baze podataka je zgodniji dijagram entiteta i odnosa (odnosno, dijagram tabela).

3.4 Objektno-orijentisane tehnike

Uvođenjem dijagrama klasa podataka, donekle smo zašli u domen objektno-orijentisanog modeliranja, preciznije, u domen objektno-orijentisanih tehnika. Ove tehnike prirodno prate objektno-orijentisane metodologije. Naravno, treba imati na umu da u različitim metodologijama je moguće korišćenje različitih tehnika. Dobro je znati što više različitih tehnika radi odabira onih tehnika koje su najzgodnije za neki posao.

Objektno-orijentisane tehnike imaju dva osnovna koncepta. Prvi je da u žižu stavljaju objekte, a ne procese. Ideja je da će objekti i njihovi međusobni odnosi biti stabilniji nego procesi. Drugi je postojanje različitih dijagramskih tehnika za modeliranje ponašavanja, strukture, odnosa, i životnog ciklusa objekata.

3.4.1 Osnovni pojmovi

Pojam objekta se definiše na mnogobrojne načine, a mi ćemo ovde navesti nekoliko njih.

- *Objekat* je apstrakcija nečeg konkretnog u domenu problema, koja opisuje sposobnost sistema da o tome čuva informaciju, interaguje sa time ili oboje. (*Coad*, *Yourdon*, 1990)
- *Objekat* je koncept, apstrakcija ili nešto sa jasnim granicam i smislom u odnosu na konkretan problem. Objekat ima dve svrhe: da pomogne razumevanju stvarnog saveta, i pruži praktičnu osnovu za računarsku implementaciju. (*Rumbaugh et al*, 1991)
- *Objekti* se opisuju odgovorima koje mogu da daju na pitanja: "Ko sam ja?", "Šta mogu da uradim?", i "Šta znam?". (*Wirf-Brock*, 1990)
- *Objekti* imaju stanje, ponašanje i identitet. (*Booch*, 1994)

Sve ovo se može rezimirati na sledeći način. Objekat je apstrakcija nečeg konkretnog u domenu problema. Objekat mora da ima stanje, ponašanje, ili znanje (ne obavezno sve). Objekat ima životni ciklus koji se sastoji od tri faze: nastajanje, postojanje, i nestajanje.

Klasa je apstrakcija skupa objekata koji imaju "dovoljno sličnosti". Kao kriterijum sličnosti se prevashodno uzima u obzir ponašanje. Stanje i znanje objekta ne utiče značajno na njihovu klasifikaciju. Objekat prestavlja konkretan primerak klase.

Grupa *OMG* daje sledeće definicije pojmova klasa i objekat.

- *Klasa* je opis skupa objekata koji dele iste atribute, operacije, metode, odnose i sematniku. Svrha klase je da deklariše kolekciju metoda, operacija i atributa koja u potpunosti opisuje strukturu i ponašanje tih objekata.
- *Objekat* je primerak koji potiče iz klase, strukturiran je i ponaša se u skladu sa svojom klasom.

U skladu sa ovim, možemo reći da su *atributi* sredstvo za opisivanje stanja i znanja, kao i da su *metodi* sredstvo za opisivanje ponašanja. Klase se formalno definišu kao skupovi svih objekata koji: imaju propisane atribute odgovarajućeg imena i

tipa, mogu da primene propisane metode odgovarajuće imena i tipa, i zadovoljavaju odgovarajuće formalne uslove semantika atributa i metoda.

U većini savremenih programskih jezika se raspoznavanje da li neki objekat pripada nekoj klasi odvija pragmatično, bez primene ad-hok pravila. Pripradnost objekta klasi se, po pravilu, određuje u trenutku pravljenja objekta i ne može se menjati tokom života objekta. Međutim, postoje jezici kod kojih se pripadnost klasi uspostavlja dinamički, proveravanjem semantičkih pravila.

3.4.2 Osnovni koncepti

Osnovni koncepti na kojima počivaju tehnike objektno orijentisanih metodologija su:

- *Enkapsulacija.* Struktura objekta je njihova interna stvar, odnosno, ona se ne sme izlagati spoljašnjem svetu. Atributima se sme pristupati samo posredno. Postoje slučajevi kada neki atributi predstavljaju osnovu ponašanja objekata ili klasa. Iako im se može omogućiti javni pristup, i tada je ipak preporučljivo enkapsuliranje. Enkapsulacijom se vrši apstrahovanje strukture metodima, i dobija se viši nivo međusobne nezavisnosti klase od modula u kojima se upotrebljava.

- *Interfejs.* Objekat (klasa) pruža spoljašnjim korisnicima samo skup metoda putem kojih mogu komunicirati sa njim. Takav skup metoda se naziva javni interfejs objekta (klase). Interfejs se oblikuje tako da omogući obavljanje jednog celovitog posla. Suština objekta je u njihovom ponašanju. Interfejs omogućava to ponašanje. Sužavanjem interfejsa na suštinu funkcionisanja objekta prikriva se sva složenost implementacije i pruža viši nivo nezavisnosti objekta od okruženja. Međutim, ako objekat ima više interfejsa, to znači da ima više funkcija, pa je potrebno razmotriti njegovo razlaganje na više objekata.

- *Polimorfizam.* Polimorfizam podrazumeva da se jednom napisan kod može upotrebljavati za različite vrste objekata. Postoje tri osnovne vrste polimorfizma: hijerarhijski, parametarski i implicitni[5]. Hijerarhijski polimorfizam koristi ispostavljene odnose "nadtip" i "podtip" među klasama kao sredstvo ostvarivanja polimorfizma. Parametarski polimorfizam počiva na upotrebi tzv. tipskih promenljivih. One se koriste za simboličko označavanje tipova vrednosti, promenljivih, i izraza. Implicitni polimorfizam podrazumeva da se pri pisanju koda uopšte ne navode tipovi vrednosti i izraza. Implicitni polimorfizam predstavlja uopštenje parametarskog polimorfizma. Polimorfizam omogućava pisanje apstraktnijeg koda, koji ima visoku upotrebljivost.

[5]Nekada se kao poseban oblik polimorfizma ubraja i tzv. ad-hok polimorfizam. On počiva na osobini nekih programskih jezika da dopuštaju višeznačnost imena funkcija i operatora (engl. *overloading*).

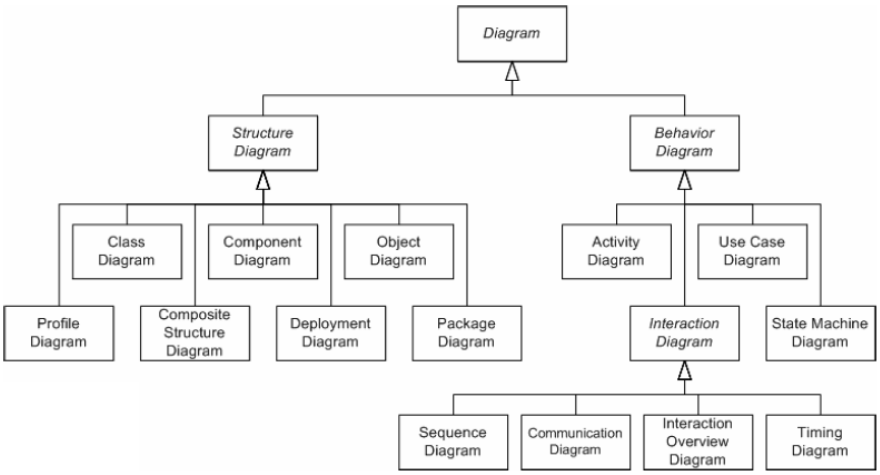
- *Nasleđivanje.* Nasleđivanje klase je ekvivalentno uvođenju jednosmerne parcijalno uređene relacije "jeste" između klasa. Kažemo da klasa A "jeste" klasa B akko svaki objekat klase A ima sve osobine koje imaju i objekti klase B. Ako A "jeste" klasa B kaže se i da je A "izvedena" klasa iz B ili A "je potomak" klase B, kao i da je B "osnovna" (ili "bazna") klasa za A ili B "je predak" klase A. Relacija "jeste" je parcijalna relacija poretka (refleksivna je i tranzitivna). Predstavlja osnovu za građenje hijerarhija klasa (bazne i izvedene klase formiraju hijerarhiju klasa). Nasleđivanje se koristi za eksplicitno označavanje sličnosti među klasama (objektima). Nasleđivanje predstavlja osnovu za hijerarhijski polimorfizam. Ako se nešto može uraditi sa svakim objektom klase B, onda se to može uraditi i sa svakim objektom klase A koja je izvedena iz B. Nasleđivanje se posmatra u dva smera, kao specijalizacija ili generalizacija: klasa A je poseban (specijalan) slučaj klase B, odnosno, klasa B je opštiji (generalan) slučaj klase A.

3.4.3 UML

UML je objedinjeni jezik za modeliranje (engl. *Unified Modeling Language*). Nastao je objedinjavanjem više različitih pristupa objektno-orijentisanom projektovanju i odgovarajućih jezika modeliranja. Sam jezik ne predstavlja metodologiju, ali je oblikovan prema paralelno razvijanoj metodologiji "objedinjeni pristup" (engl. *Unified Approach*). Sam jezik je razvijen, publikovan i šire prihvaćen pre odgovarajuće metodologije. Danas se koristi u praktično svim objektno-orijentisanim razvojnim metodologijama, pa i šire.

3.4.4 Vrste UML dijagrama

UML dijagrami se dele u tri grupe: dijagrami ponašanja, dijagrami interakcije, i strukturni dijagrami. Na slici 3.22 dat je nešto drugačiji prikaz podele UML dijagrama. U toj podeli, dijagrami interakcije se smatraju podvrstom dijagrama ponašanja.



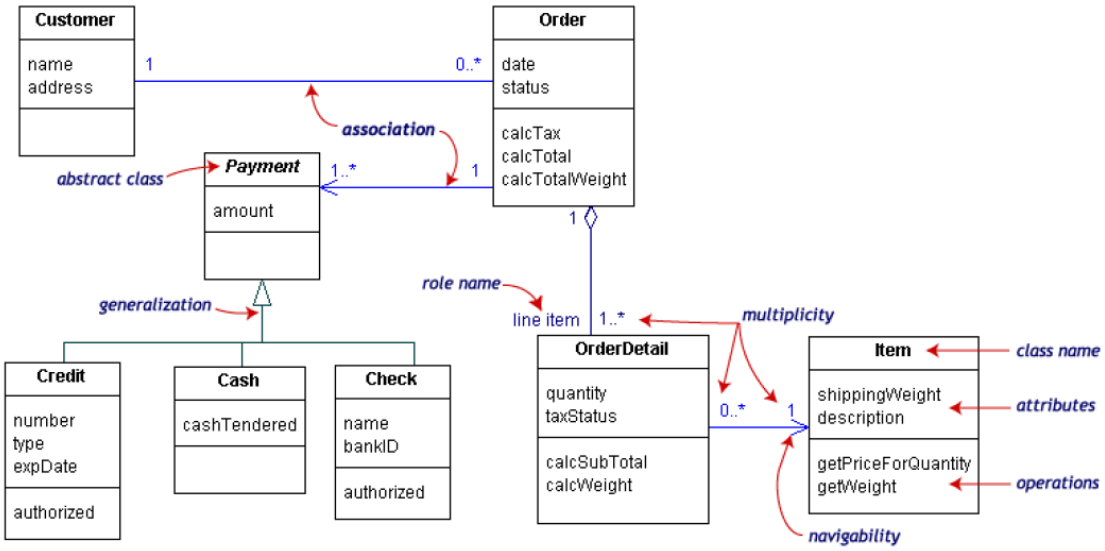
Slika 3.22: Vrste UML dijagrama.

3.4.4.1 Strukturni dijagrami

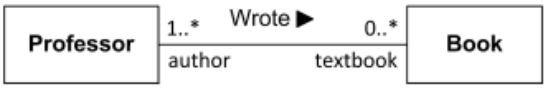
U strukturne UML dijagrame ubrajamo:

- Dijagram klasa (engl. *class diagram*). On ilustruje elemente statičkog modela, kao što su klase, njihov sadržaj, i međusobne odnose. Sadrži: nazive klasa, atribute klasa, modele klasa, specijalizaciju i generalizaciju, i odnose (među kojima su najvažniji: asocijacija, agregacija, i kompozicija). Slika 3.23 prikazuje primer dijagrama klasa.

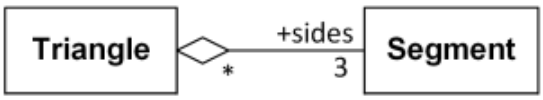
  Asocijacija se koristi radi prikazivanja načina na koji su dve klase povezane. Predstavlja se punom linijom. Na slici 3.24 prikazan je primer asocijacije. Agregacija je binarna asocijacija koja predstavlja odnos ceo–deo nečega između kompozita i nekog njegovog dela. Predstavlja se punom linijom koja na jednom kraju je prazna (predstavlja deo), a na drugom kraju ima prazan romb (označava kompozit pripada deo). Na slici 3.25 prikazan je primer agregacije. Kompozicija je sličan odnos agregaciji, s razlikom da je odnos identifikujući. To znači da deo celine može da se nađe u najviše jednom kompozitu. Razlika u predstavi između kompozicije i agregacije je da je romb u kompoziciji obojen crnom bojom. Na slici 3.26 prikazan je primer kompozicije.



Slika 3.23: Dijagram klasa – primer.



Slika 3.24: Primer asocijacije: “profesor” koji ima ulogu “autora” je asociran sa “udžbenikom” tipa “knjiga”.

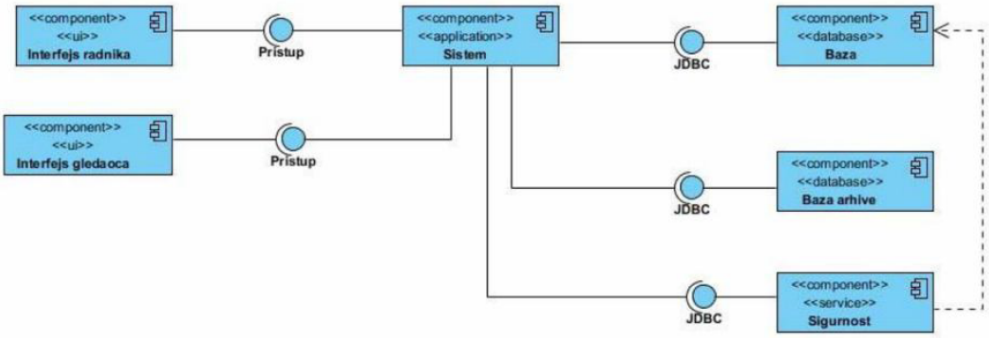


Slika 3.25: Primer agregacije: “trougao” ima kolekciju “stranica” od 3 “duži”. Jedna “duž” može biti deo nijednog ili više “trouglova”.

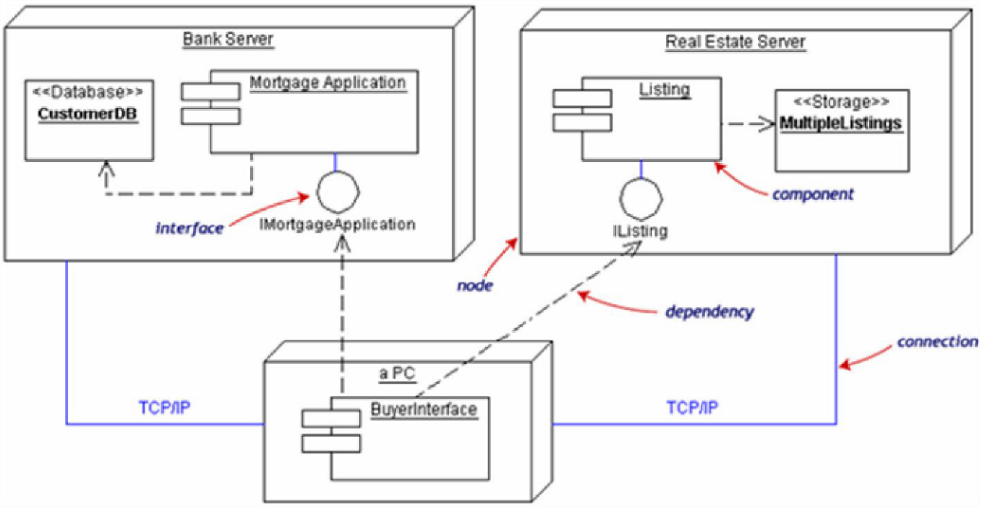
Slika 3.26: Primer kompozicije: “direktorijum” može da sadrži nijedan ili više “fajlova”, pri čemu svaki “fajl” ima tačno jednog roditelja (“direktorijum”). Ukoliko je “direktorijum” obrisan, svi “fajlovi” u njemu su takođe obrisani.

Dijagram klasa može da opisuje sistem, ali pre će opisivati neki njegov deo. Naime, jedan iole ozbiljniji sistem će imati preko stotinu (ili pre hiljadu) klasa. Veoma retke su situacije u kojima se pravi jedan ogroman dijagram klasa koji opisuje ceo sistem.

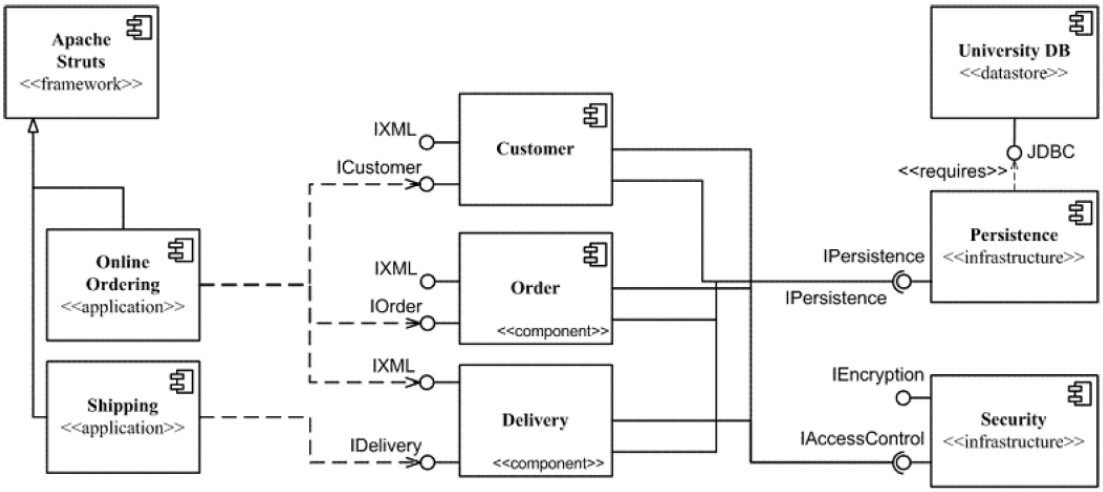
- Dijagram komponenti (engl. *component diagram*). On ilustruje komponente (osnovne celine) koje čine aplikaciju, sistem ili organizaciju. Sadrži: nazive komponenti, njihove međusobne odnose, i javne interfejse. Slike 3.27, 3.28, i 3.29 prikazuju primere dijagrama komponenti. Postoje različite oznake za komponente, a najčešće su to pravougaonici sa oznakom slagalice u uglu, ili 3D kvadri. Interfejs je sastavni deo komponente i označava se linijom i krugom.



Slika 3.27: Dijagram komponenti – primer 1.



Slika 3.28: Dijagram komponenti – primer 2.



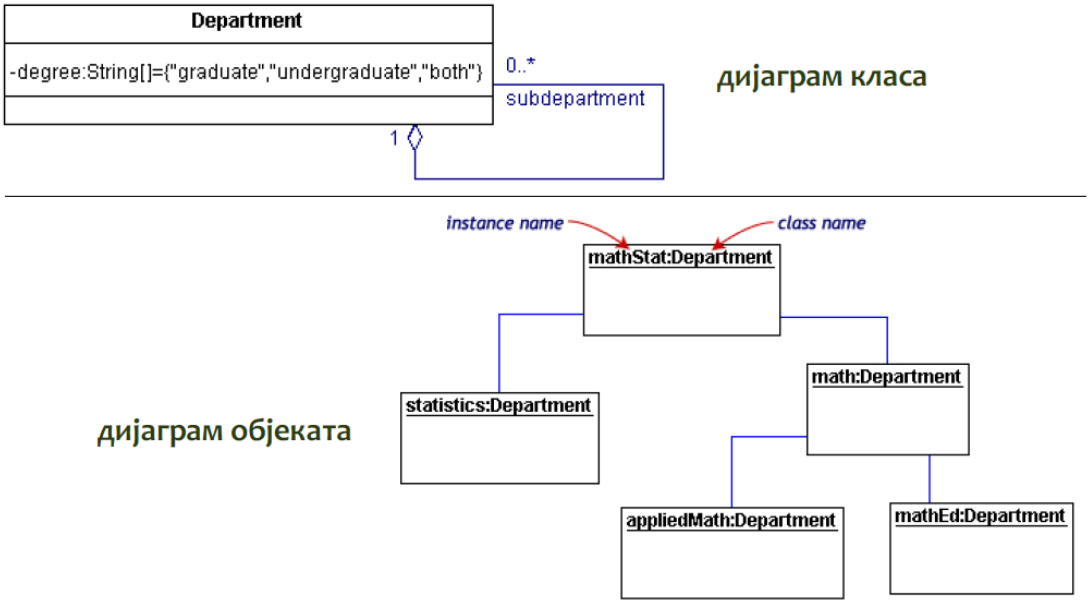
Slika 3.29: Dijagram komponenti – primer 3.

Suština komponenti je da nas njihova struktura praktično uopšte ne zanima. Interfejs je opis skupa usluga koje neka komponenta pruža i koje moramo da obezbedimo u sistemu. Javnim interfejsom opisujemo ponašanje komponenti. To je onaj deo komponenti koji je vidljiv ostatku sistema ili korisnicima.

Jedna komponenta koristi druge komponente tako što koristi njihove interfejse. Ono što je bitno napomenuti je da komponente služe da implementiraju interfejse, a ne da pomoću interfejsa koristimo komponente. Na primer, komponenta "arhiva" pruža "interfejs za upotrebu arhive".

Jedan sistem je moguće podeliti na komponente na više načina. Štaviše, nekada ume biti veoma teško unapred reći koja podela je bolja od druge. Da bismo mogli da prepoznamo veličinu jedne podele na komponente, moramo da izvršimo detaljnu analizu, što obuhvata: upoznavanje strukture sistema, upoznavanje strukture informacija, poslova, entitetima, subjektima, i drugo. Upravo zbog toga je dijagram komponenti bitan. On pokazuje da je potrebno da se izvrši baš takva dekompozicija, pa uz dijagram komponenti obično ide neka vrsta obrazloženja: razlozi odabira određene dekompozicije, specifikacije interfejsa, funkcionalnosti komponenti, spisak poslova za svaku komponentu koje će ona raditi i funkcija koje će obavljati (recimo, iz matrice poslova i funkcija).

- Dijagram objekata (engl. *object diagram*). On predstavlja objekte i njihove odnose u jednom trenutku vremena. Koristi se kao dopuna dijagrama klasa i komunikacije za opisivanje dinamičkih sistema. Sadrži: nazive klasa, nazive objekata, imena i vrednosti atributa, i odnose. Slika 3.30 prikazuje primer dijagrama objekata. Pre svega služi da dopuni dijagram klasa.



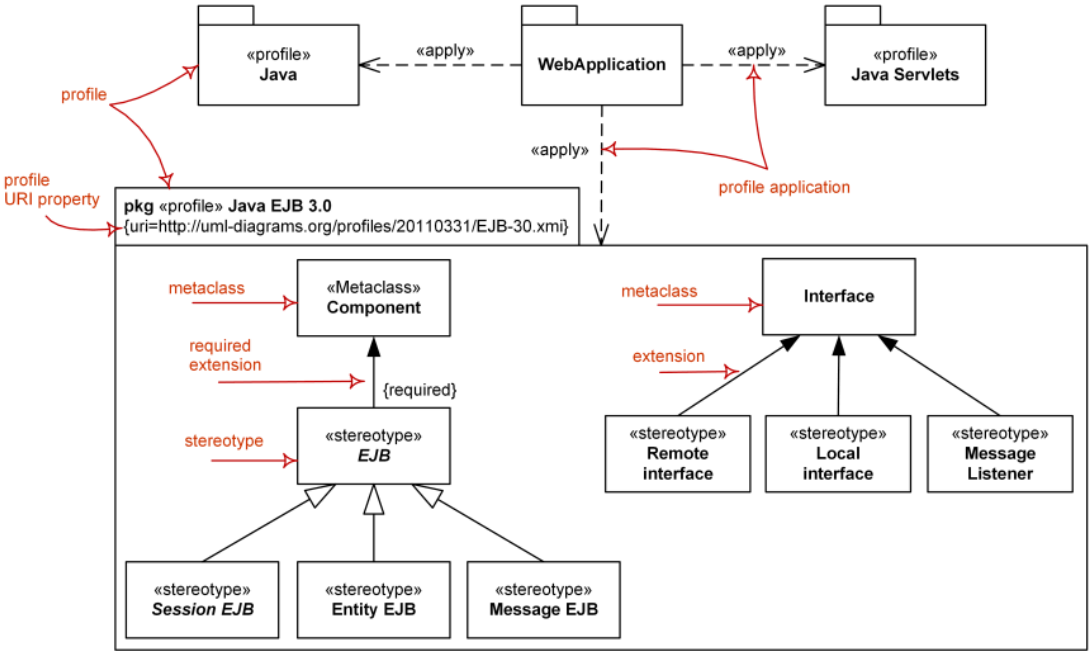
Slika 3.30: Dijagram objekata – primer.

- Dijagram profila (engl. *profile diagram*)[6]. On opisuje mehanizam lake ekstenzije ka UML-u tako što definiše tzv. stereotipove (klase profila koji definišu kako postojeća metaklasa može biti proširena kao deo profila), označene vrednosti (kada je stereotip primenjen na model, vrednosti njegovih svojstva mogu se referisati kao označene vrednosti), i ograničenja. Sadrži: profile, metaklase, stereotipove, produženja, referencu, i aplikaciju profila. Slika 3.31 prikazuje primer dijagrama profila.

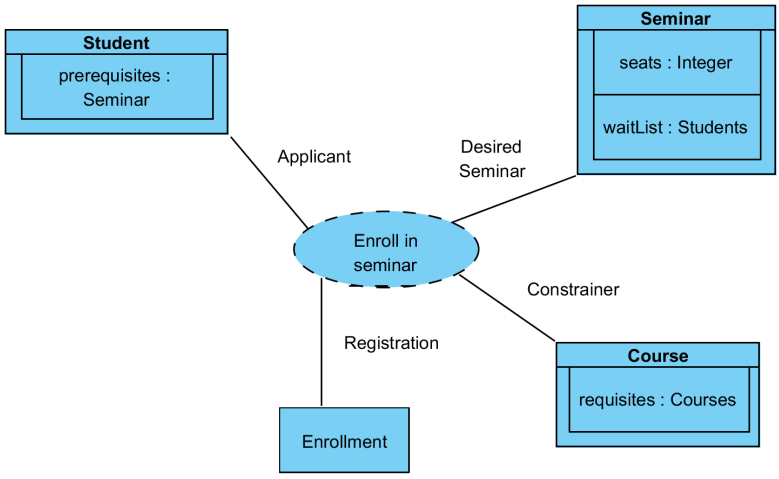
- Dijagram složene strukture (engl. *composite structure diagram*). On predstavlja internu strukturu klase, objekta, komponente, ili slučaja upotrebe. Sadrži: složene komponente i njihove elemente, i tačke interakcije sa drugim elementima sistema. Slika 3.32 prikazuje primer dijagrama složene strukture.

  Dijagrami složene strukture se koriste i u prikazivanju elemenata koje učestvuju u nekom slučaju upotrebe i to onda kada je slučaj upotrebe dovoljno složen da je tekst opisa takav da nije naizgled jasno o čemu se radi.

[6]Ovaj dijagram nije toliko uobičajen, pa samim tim i ne treba obraćati preveliku pažnju na njega.



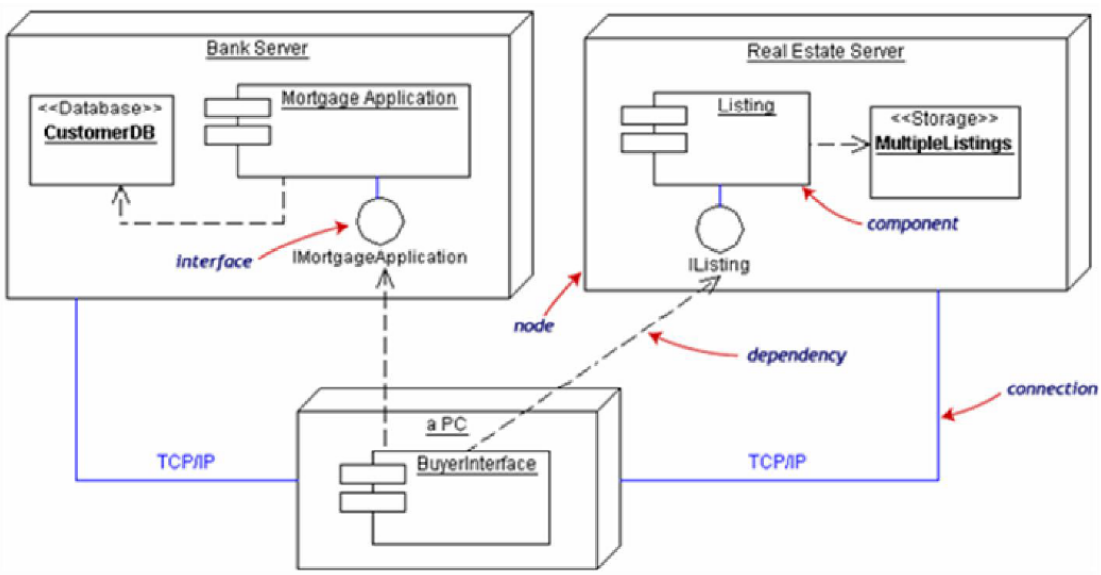
Slika 3.31: Dijagram profila – primer.



Slika 3.32: Dijagram složene strukture – primer.

- Dijagram isporučivanja (engl. *deployment diagram*). On predstavlja elemente fizičke arhitekture sistema. Za njega se kaže da predstavlja dijagram komponenti najvišeg nivoa. Sadrži: čvorove (ili servere; to su hardverske jedinice koje izvršavaju nekakve softvere), softverske ili hardverske podsisteme, međusobne veze između podsistema ili između čvorova, i može da ilustruje i zastupljenost komponenti u podsistemima (odnosno, koja komponenta će raditi na kojoj hardverskoj jedinici sistema). Slika 3.33 prikazuje

primer dijagrama isporučivanja. Vrlo često se ovi dijagrami kombinuju sa dijagramima komponenti.

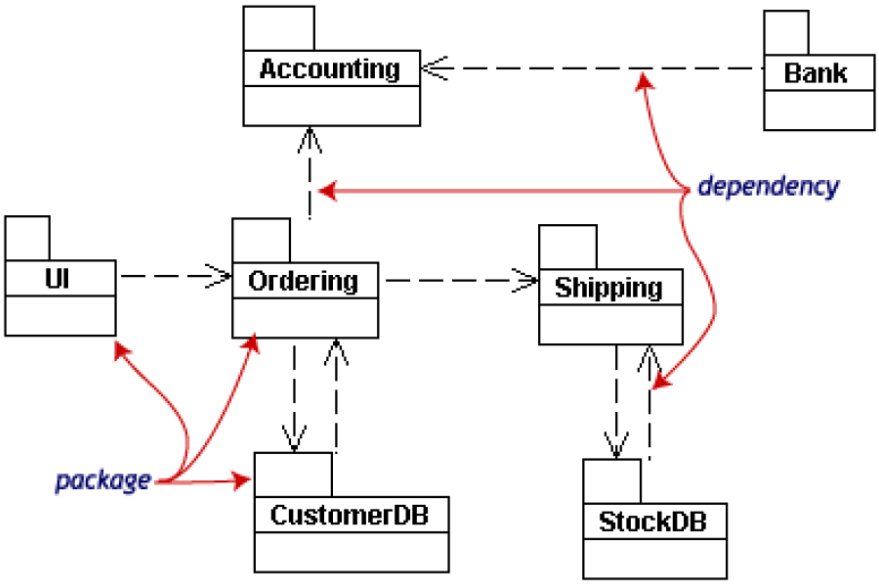


Slika 3.33: Dijagram isporučivanja – primer.

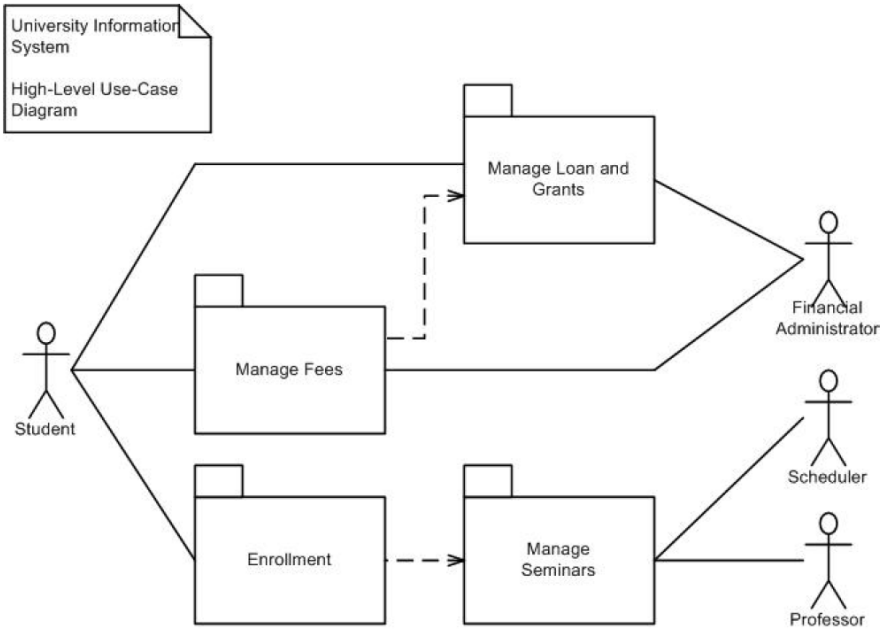
Najčešća notacija je da se čvorovi predstavljaju 3D kvadrima, podsistemi pravougaonicima, i uz to možemo predstaviti konkretne komponente i interfejse na svakoj (ili nekoj) jedinici. Međusobne veze su predstavljene linijama (usmerenim ili neusmerenim). Možemo predstaviti specifične čvorove nešto drugačije, na primer, serveri baza podataka se obično predstavljaju valjcima ili čvorovi koji predstavljaju klijentske mašine se predstavljaju PC oznakom (monitor i tastatura).

- Dijagram paketa (engl. *package diagram*). On ilustruje kako su elementi logičkog modela organizovani u pakete, kao i međuzavisnosti paketa. Sadrži: nazive i granice paketa, klase u paketima, međusobne odnose klasa, međusobne zavisnosti paketa. Može se koristiti i u domenu slučajeva upotrebe. Slike 3.34 i 3.35 prikazuju primere dijagrama paketa.

Dijagram paketa i dijagram komponenti imaju sličnosti. Naime, oni pokušavaju da u sebe grupišu više klasa, komponenti, itd. i da to zaokruže u jednu celinu. Međutim, razlika je u tome koja vrsta dekompozicije je u pitanju. Kod dijagrama komponenti, radi se o funkcionalnoj dekompoziciji, odnosno, o dekompoziciji na celine koje rade neki celovit posao. Kod dijagrama paketa se radi o logičkoj međuzavisnosti.



Slika 3.34: Dijagram paketa – primer 1.



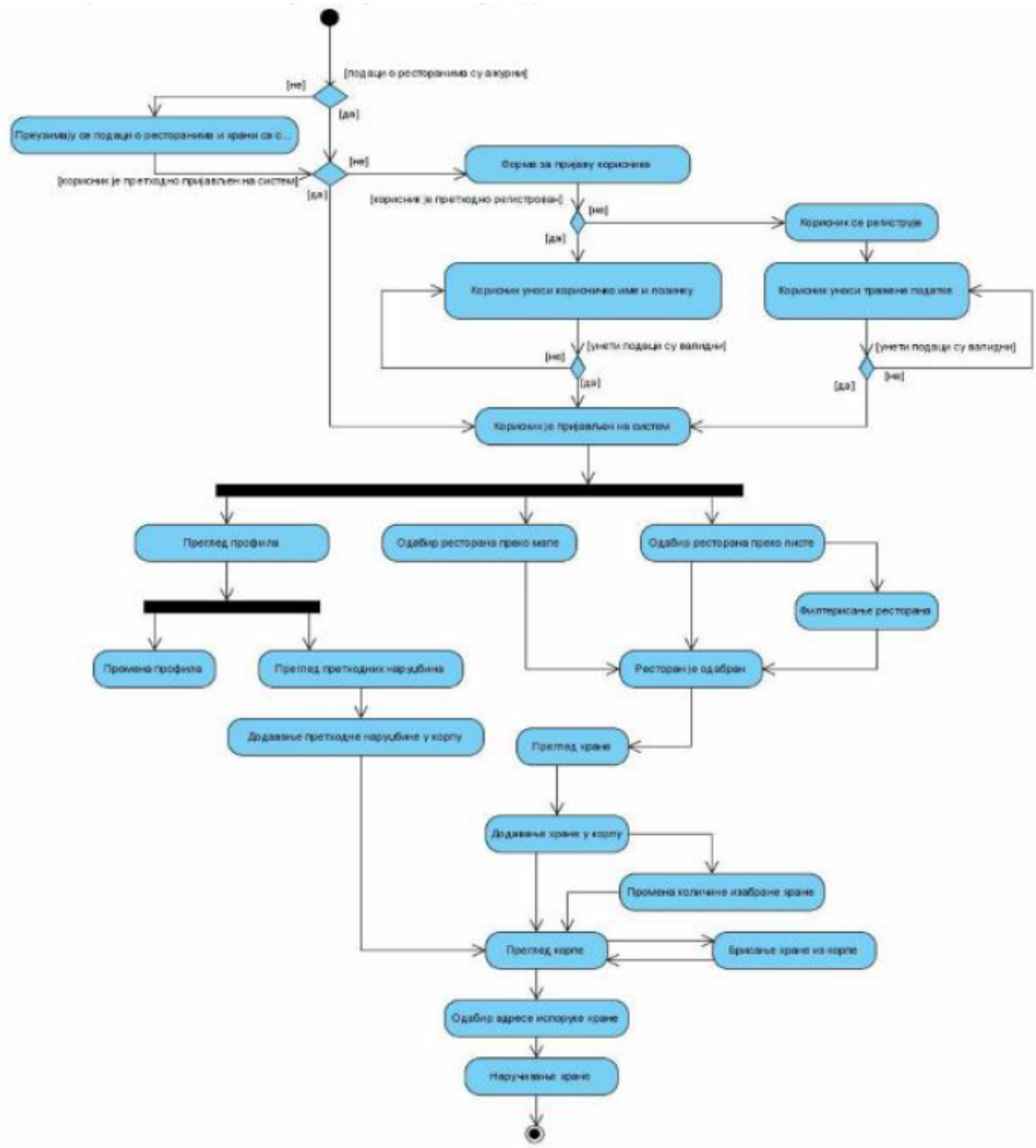
Slika 3.35: Dijagram paketa – primer 2.

3.4.4.2 Dijagrami ponašanja

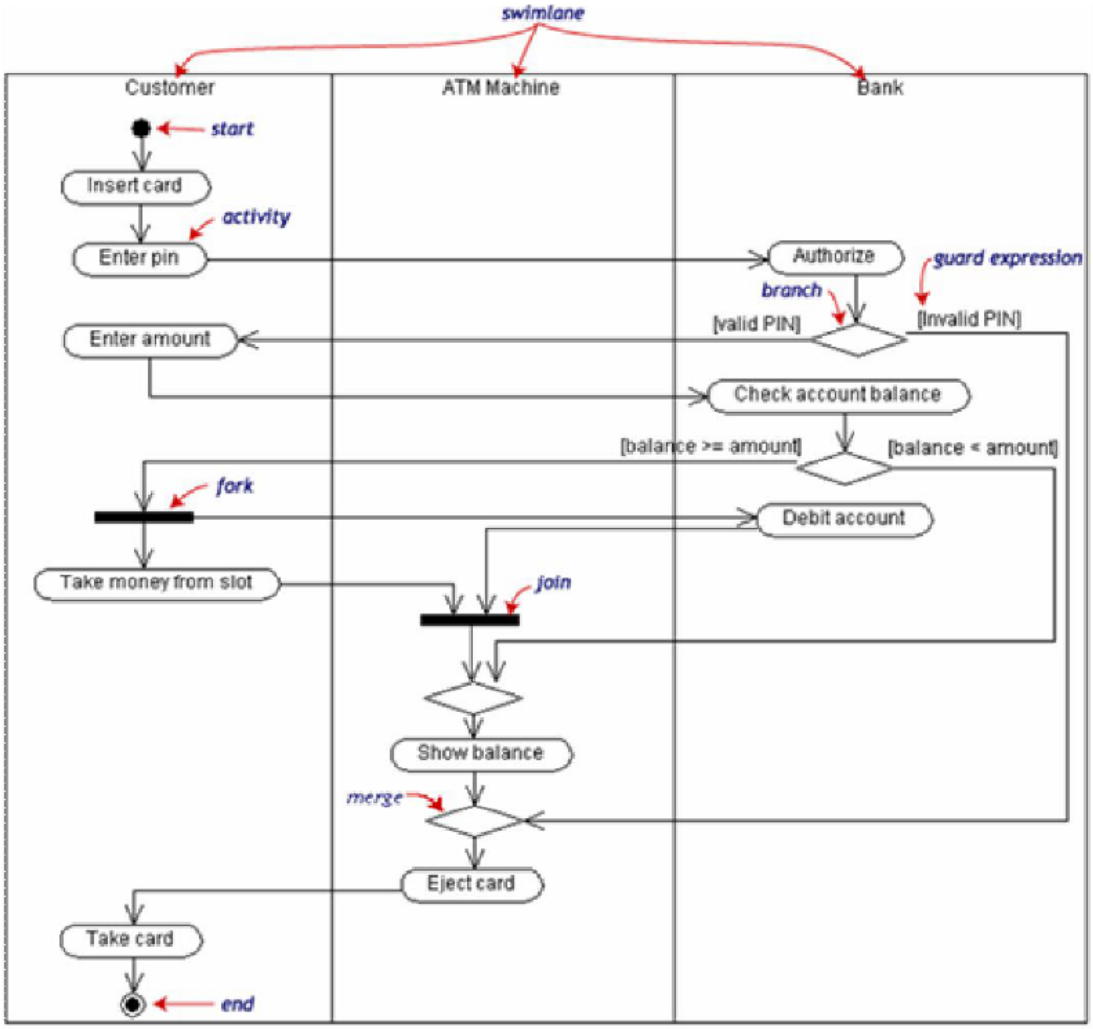
U UML dijagrame ponašavanja ubrajamo:

- Dijagram aktivnosti (engl. *activity diagram*). On predstavlja poslovne procese višeg nivoa, tokove podataka, i eventualno složene logičke elemente sistema. Sadrži: procese, tokove podataka, čvorišra i grananja, uslovne tačke,

početne i završne tačke, a može da sadrži i "linije autora". Slike 3.36 i 3.37 prikazuju primere dijagrama aktivnosti. Za razliku od BPMN dijagrama, ovi dijagrami nemaju događaje i poruke.



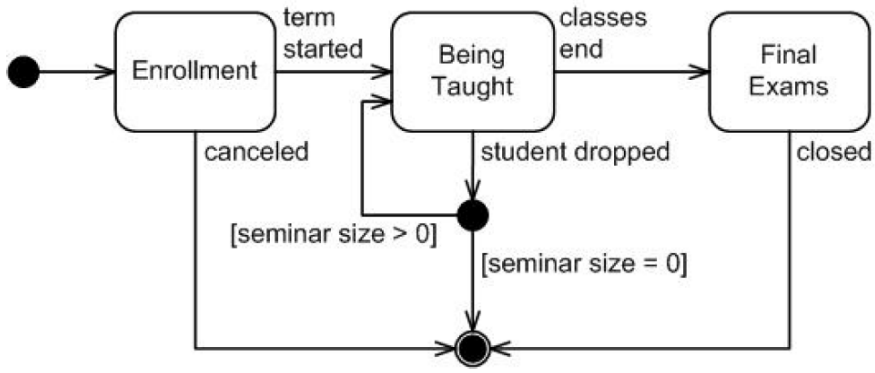
Slika 3.36: Dijagram aktivnosti – primer 1.



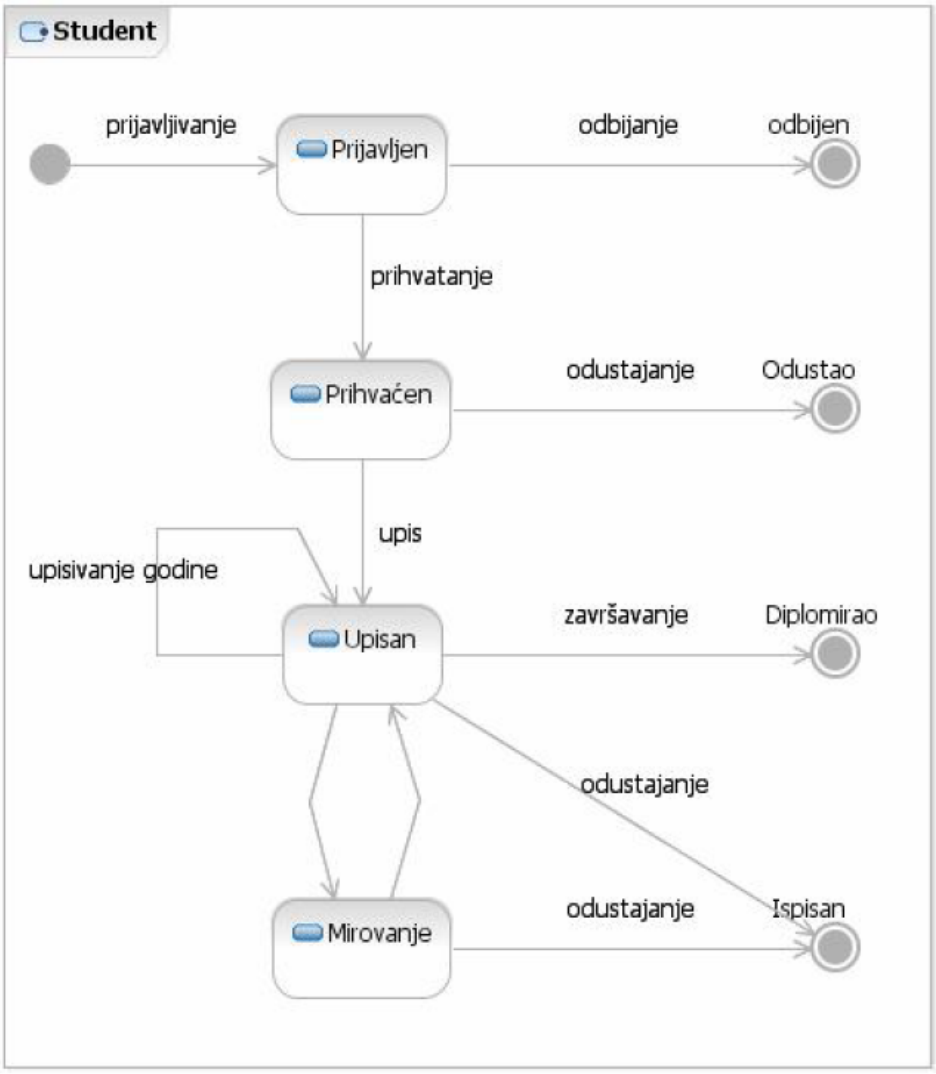
Slika 3.37: Dijagram aktivnosti – primer 2.

- Dijagram stanja (engl. *state machine diagram*). On opisuje kako se stanja objekta menjaju u zavisnosti od interakcija u koje objekat ulazi. Sadrži: početna stanja, aktivna stanja, završna stanja, i promene stanja. Aktivnosti prevode objekat iz jednog stanja u drugo stanje. Slike 3.38 i 3.39 prikazuju primere dijagrama stanja.

  Ovaj dijagram veoma liči na dijagram životnog ciklusa entiteta, sa veoma malim razlikama. Za razliku od dijagrama aktivnosti koji se ograničava na jedan slučaj upotrebe, dijagrami stanja često mogu da opišu promena stanja objekta tokom života objekta (što je obično više od jednog slučaja upotrebe).

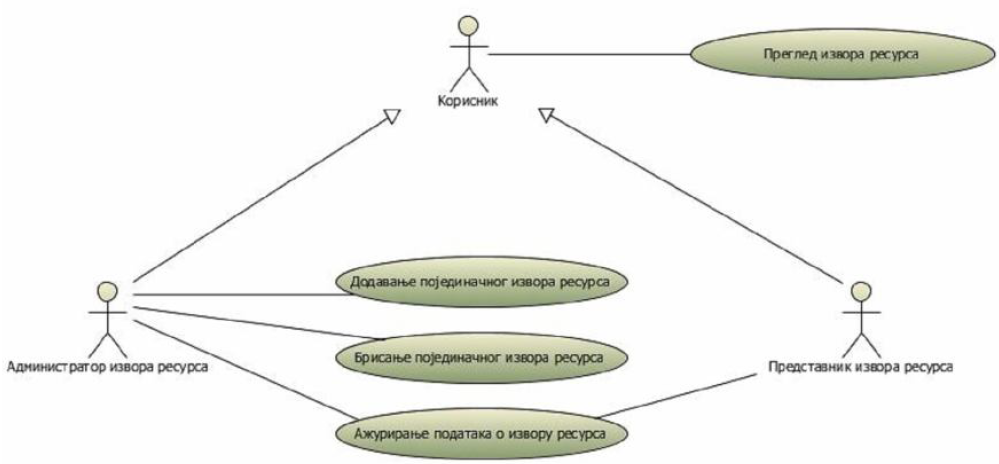


Slika 3.38: Dijagram stanja – primer 1.

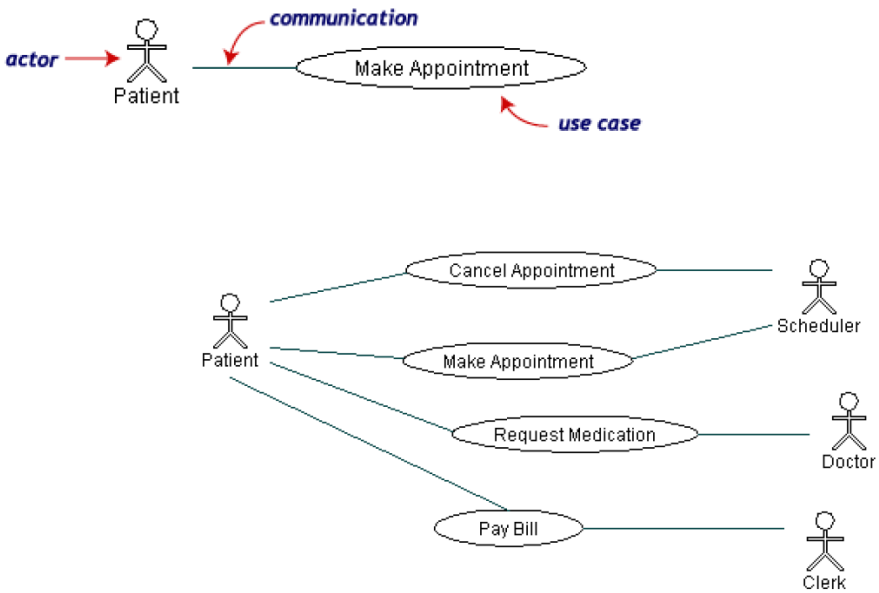


Slika 3.39: Dijagram stanja – primer 2.

- Dijagram slučajeva upotrebe (engl. *use case diagram*). On predstavlja slučajeve upotrebe, aktere, i njihove međusobne odnose. Opisuje sve poslove koji čine jednu celinu. Sadrži: slučajeve upotrebe (poslovi), aktere (subjekti koji učesvuju u poslovima), pakete (služe za grupisanje poslova), podsisteme (spoljni ili unutrašnji, ako ih koristimo), i međusobne odnose. Slike 3.40 i 3.41 prikazuju primere dijagrama slučajeva upotrebe.



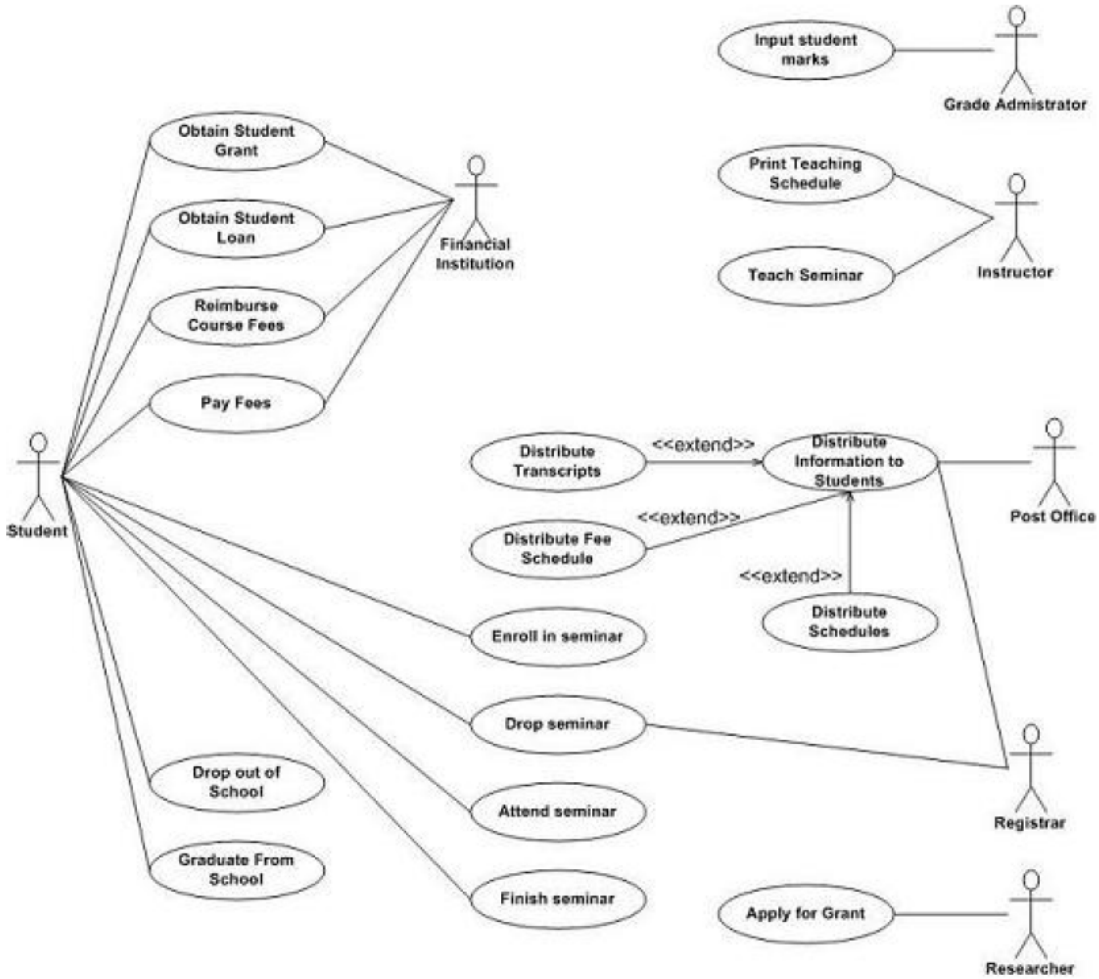
Slika 3.40: Dijagram slučajeva upotrebe – primer 1.



Slika 3.41: Dijagram slučajeva upotrebe – primer 2.

Slučajevi upotrebe mogu imati više različitih aktera. To ima dva tumačenja. Jedno tumačenje je da više aktera "zajedno" rade da bi posao bio celovit i

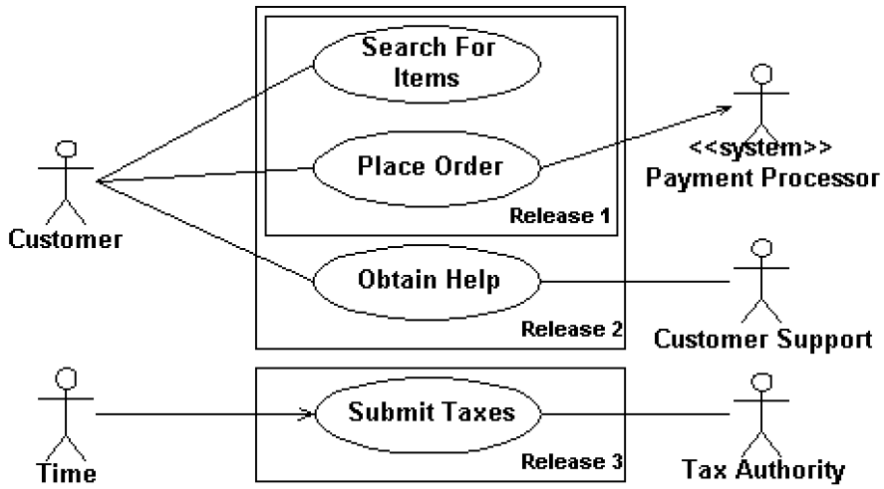
obavljen. Na primer, na slici 3.41 "pacijent" i "doktor" zajedno učestvuju u slučaju upotrebe "zahtevanje leka". Drugo tumačenje je da svi akteri rade isti posao, ali, na primer, sa drugačijim nivoom autorizacije. Na primer, na slici 3.40, "administrator izvora resursa" i "predstavnik izvora resursa" rade isti posao – "ažuriranje podataka o izvoru resursa".



Slika 3.42: Dijagram slučajeva upotrebe – primer 3.

Između slučajeva upotrebe postoje dve osnovne vrste relacija: proširivanje, i uključivanje. *Proširivanje* (engl. *extend*) predstavlja vrstu specijalizacije između slučajeva upotrebe. Na primer, na slici 3.42, "distribuiranje informacija studentima" ima specifične slučajeve: "distribuiranje transkripta ocena", "distribuiranje računa", i "distribuiranje rasporeda". *Uključivanje* (engl. *include*) predstavlja odnos u kojem jedan slučaj upotrebe uključuje drugi. Na primer, ako postoji poseban slučaj upotrebe "prijavljivanje na sistem" i postoji, na primer, "prijavljivanje ispita", onda možemo reći da drugi slučaj upotrebe uključuje prvi.

Slučajeve upotrebe možemo grupisati u pakete, s time da to grupisanje nije vršeno i funkcionalno, ni fizički, ni logički, nego samo radi, na primer, redosleda implementacije (što je predstavljeno na slici 3.43).



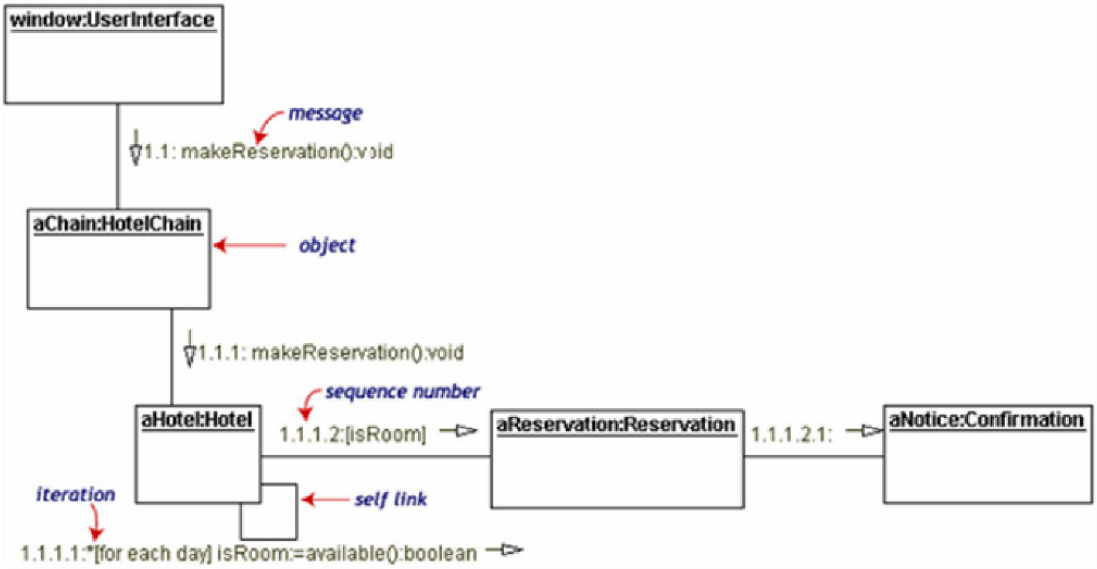
Slika 3.43: Dijagram slučajeva upotrebe – primer 4.

3.4.4.3 Dijagrami interakcije

Već smo napomenuli da se u dijagrame ponašanja mogu ubrojati i dijagrami interakcije. Osnovna ideja ovih dijagrama jeste opisivanje načina na koji se obavlja komunikacija između više objekata. U UML dijagrame interakcije ubrajamo:

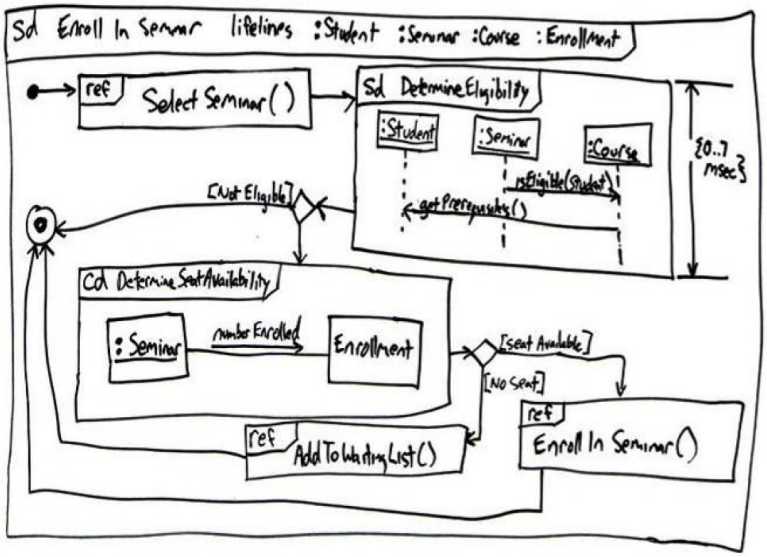
- Dijagram komunikacije (engl. *communication diagram*; raniji naziv dijagram saradnje (engl. *collaboration diagram*)). On predstavlja objekte, njihove međusobne odnose, i poruke koje razmenjuju. Pažnja se po pravilu posvećuje strukturnoj organizaciji objekta koji učestvuju u razmeni poruka. Njihovo korišćenje ima smisla kada je niz aktivnosti nejasan (ili slabo jasan). Sadrži: objekte (može i aktere), poruke, i komentare i napomene. Slika 3.44 prikazuje primer dijagrama komunikacije.

  Na slici 3.44, prikazano je sledeće: posao počinje tako što "prozor" pošalje poruku "napravi rezervaciju" objektu "lanac hotela". Oznaka 1.1 označava da je u poslu 1 ovo prva poruka koja se šalje. "Lanac hotela" na to odgovara tako što šalje odgovarajuću poruku "hotelu". Oznaka 1.1.1 označava da se poruka šalje u okviru reakcije na prethodnu poruku. Nakon primanja poruke, "hotel" radi dve stvari. Prvo (oznaka 1.1.1.1), za svaki dan, proverava da li je soba slobodna za taj dan. Drugo (oznaka 1.1.1.2), ako je ta soba slobodna, vrši rezervaciju. Nadalje se vrši potvrda na sličan način.



Slika 3.44: Dijagram komunikacije – primer.

- Dijagram interakcija (engl. *interaction diagram* ili *interaction overview diagram*). On predstavlja varijantu dijagrama aktivnosti u kojoj je akcenat na upravljanju procesima ili sistemom. Svaki čvor (odnosno, aktivnost) u dijagramu može da predstavlja neki drugi dijagram interakcija ili aktivnosti. Sadrži: objekte, manje dijagrame aktivnosti ili interakcija, slučajeve upotrebe, tok odvijanja procesa (protoka podataka), grananja i spajanja, i početak i kraj. Slika 3.45 prikazuje primer dijagrama interakcija. Zbog upotrebe BPMN dijagrama, ovaj dijagram nije toliko često upotrebljavan.

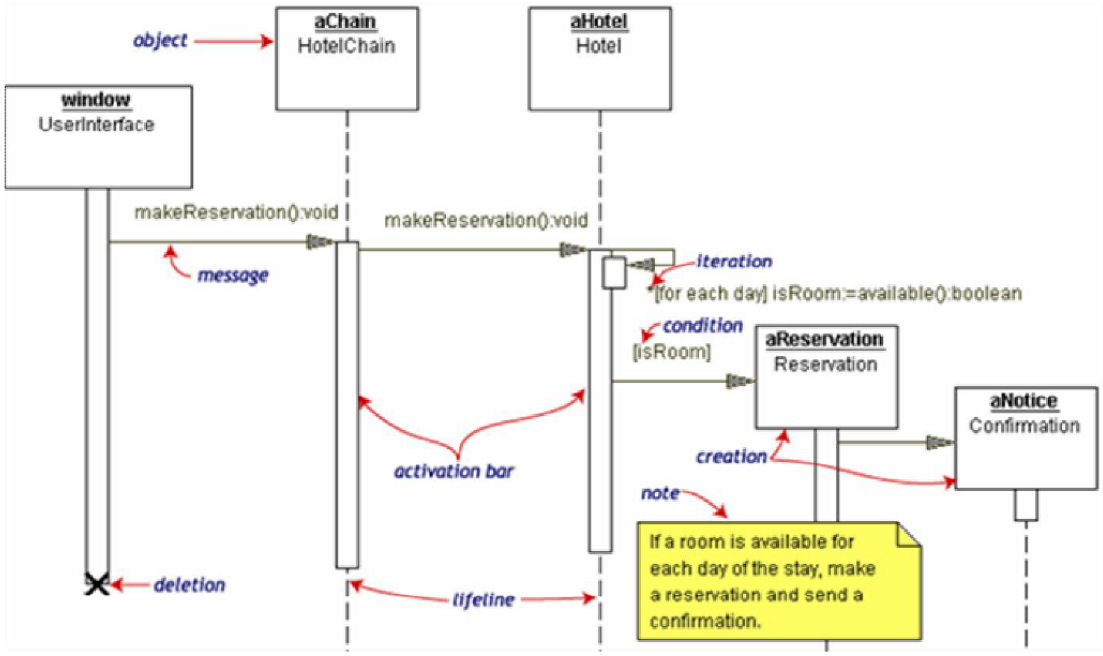


Slika 3.45: Dijagram interakcija – primer.

- Dijagram sekvence (engl. *sequence diagram*). On predstavlja redosled razmenjivanja poruka tokom odvijanja procesa. Sadrži: objekte (ili učesnike), i poruke.

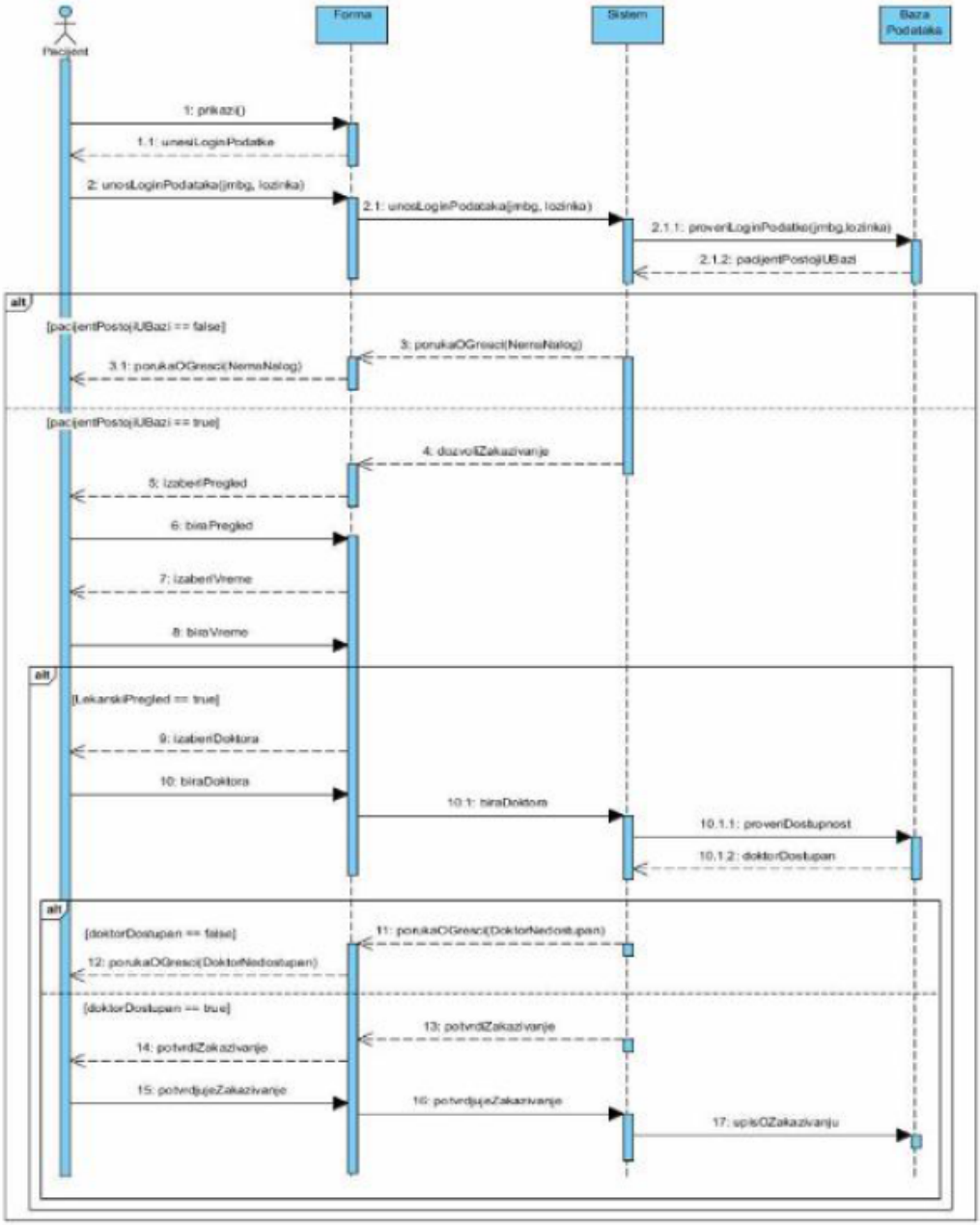
  Objekti su pravougaone oznake objekata u vrhu dijagrama. Obično su imenovani, ali privremeni objekti mogu biti i neimenovani. Za njih se vezuju vertikalne *linije "života"* koje pokazuju period aktivnosti objekata u posmatranom procesu. *Poruke* su horizontalne linije koje povezuju linije života objekata. Punim strelicama su označene sinhrone, a svetlim strelica su označene asinhrone poruke (nakon slanja poruke, nije nam bitno kada će objekat koji je primio poruku završiti posao; UML, verzija 2). Isprekidanim linijama su označene poruke koje nose rezultate (nakon slanja poruke objekat čeka da dobije odgovor; UML, verzija 2).

  Slike 3.46 i 3.47 prikazuju primere dijagrama sekvence. Dijagram sekvence je danas najviše korišćen dijagram iz ove grupe.



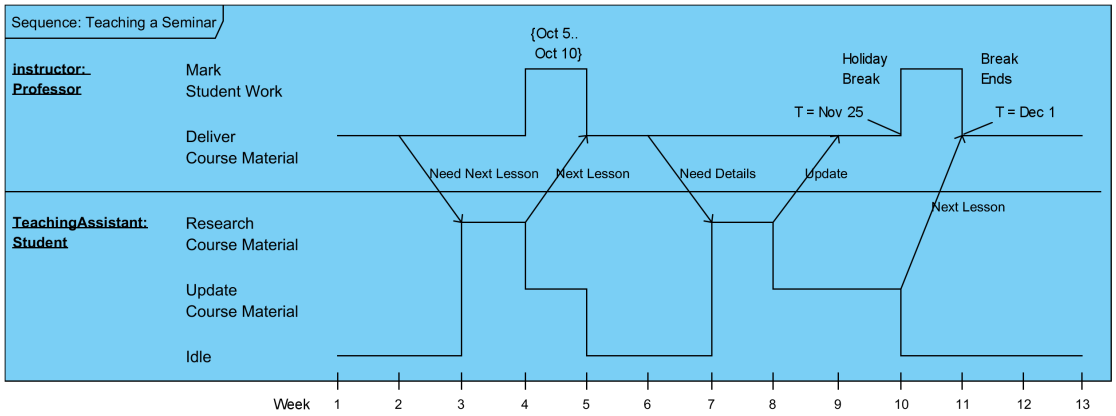
Slika 3.46: Dijagram sekvence – primer 1.

Na slici 3.47 prikazan je nešto veći dijagram sekvence. Na ovom dijagramu su prikazane celine koje su zaokružene. Ona može da označava alternativu ili jednu celinu posla koju na dijagramu možemo da predstavimo jednom porukom. Odgovore obično možemo da predstavimo i isprekidanom linijom, dok obične (prave) poruke predstavljamo punom linijom.



Slika 3.47: Dijagram sekvence – primer 2.

- Dijagram vremena (engl. *timing diagram*). On predstavlja ili promene stanja i uslova objekata ili događaji koji se dešavaju (y-osa) tokom vremena (x-osa). Uobičajeno se upotrebljava za predstavljanje promena stanja u zavisnosti od spoljnih događaja. Sadrži: protok vremena, spoljašnje događaje, i promene stanja. Slika 3.48 prikazuje primer dijagrama vremena.



Slika 3.48: Dijagram vremena – primer.

3.4.5 UML opis slučajeva upotrebe

Slučaj upotrebe je celina posla koja daje nekakav smisleni rezultat, a dovoljno je mala da može da se jednoznačno opiše i implementira. Opis slučaja upotrebe bi trebalo da bude dovoljno dugačak, tj. da obuhvati sve ono što je potrebno da bi neko mogao na osnovu tog opisa da napravi tačnu implementaciju. Svaki slučaj upotrebe mora da ima svoj opis.

Većina UML dijagrama se zasniva prvenstveno na grafičkom obliku. Dijagram slučajeva upotrebe predstavlja odnose slučajeva upotrebe i aktera. Pojedinosti sličajeva upotrebe se opisuju tekstualno. Dobar opis slučaja upotrebe sadrži dovoljno informacija da se može sagledati kada, kako, ko i sa kojim podacima učestvuje u slučaju upotrebe.

Opis slučaja upotrebe može sadržati različite elemente, ali obično ga čine: naziv (jednoznačno identifikuje slučaj upotrebe), akteri (za svakog aktera mora da bude jasno koju ulogu ima), kratak opis (veoma kratak tekst koji objašnjava šta je slučaj upotrebe i koja je njegova uloga u sistemu, bez detaljisanja o redosledu koraka i sl.), preduslovi (šta mora da bude ispunjeno da bi posao mogao da se izvrši), postuslovi (opisuje stanje sistema nakon završenog posla), opis toka slučaja upotrebe (numerisana lista koraka), opis posebnih slučajeva (numerisana lista koraka svakog od posebnih slučajeva do kojih može doći, a nisu u kontradikciji sa preduslovima), i dijagrami koji tačnije opisuju slučaj upotrebe (dijagrami aktivnosti, dijagrami interakcije, itd.).

Pored toga, opis slučaja upotrebe mogu činiti i: klasifikacija (po nivou, vrsti, itd.), dodatni zahtevi (na primer, ciljne performanse), najvažniji rizici u izvođenju (potencijalno otvorena pitanja, alternative, moguća unapređenja), objašnjenje cilja koji se ostvaruje slučajem (ako nije očigledno), i različiti prilozi (na primer, formulari, štampani izveštaji, itd.).

U nastavku dajemo primer opisa slučaja upotrebe "prijavljivanje na konkurs" (za upis novog studenta na fakultet).

Naziv: Prijavljivanje na konkurs

Akter: Kandidat koji se prijavljuje na konkurs za upis na fakultet.

Kratak opis: Kandidat podnosi neophodna dokumenta i formulare na konkurs za upis na fakultet.

Osnovni tok događaja: Kandidat donosi popunjene obrasce P1, P2, i po potrebi P3, i sledeća dokumenta:

- fotokopiju izvoda iz matične knjige rođenih (original na uvid),
- fotokopije svedočanstva sva četiri razreda srednje škole (originali na uvid),
- fotokopiju svedočanstva o završenoj srednjoj školi (original na uvid),
- potvrdu o uplati odgovarajućeg iznosa za prijavu.

Alternativni tokovi događaja: Kandidat donosi i:

- za pripadnike srpskog naroda iz susednih država – pisanu izjavu o nacionalnoj pripadnosti overenu u opštinskom organu mesta stalnog prebivališta; ako kandidat nema takvu izjavu, potpisuje na licu mesta potvrdu sa odgovarajućim sadržajem i upućuje se od strane službenika da do termina upisa pribavi odgovarajuću izjavu u opštinskom organu mesta prebivališta,
- za druge strane državljane – nostrifikovana školska dokumenta ili potvrda o započetoj nostrifikaciji,
- ako je kandidat položio prijemni ispit na drugom fakultetu – potvrdu o položenom prijemnom ispitu na drugom fakultetu.

Preduslovi: Nema.

Postuslovi: Nema.

Prilozi: Sadržaj i predložen izgled formulara P1, P2 i P3

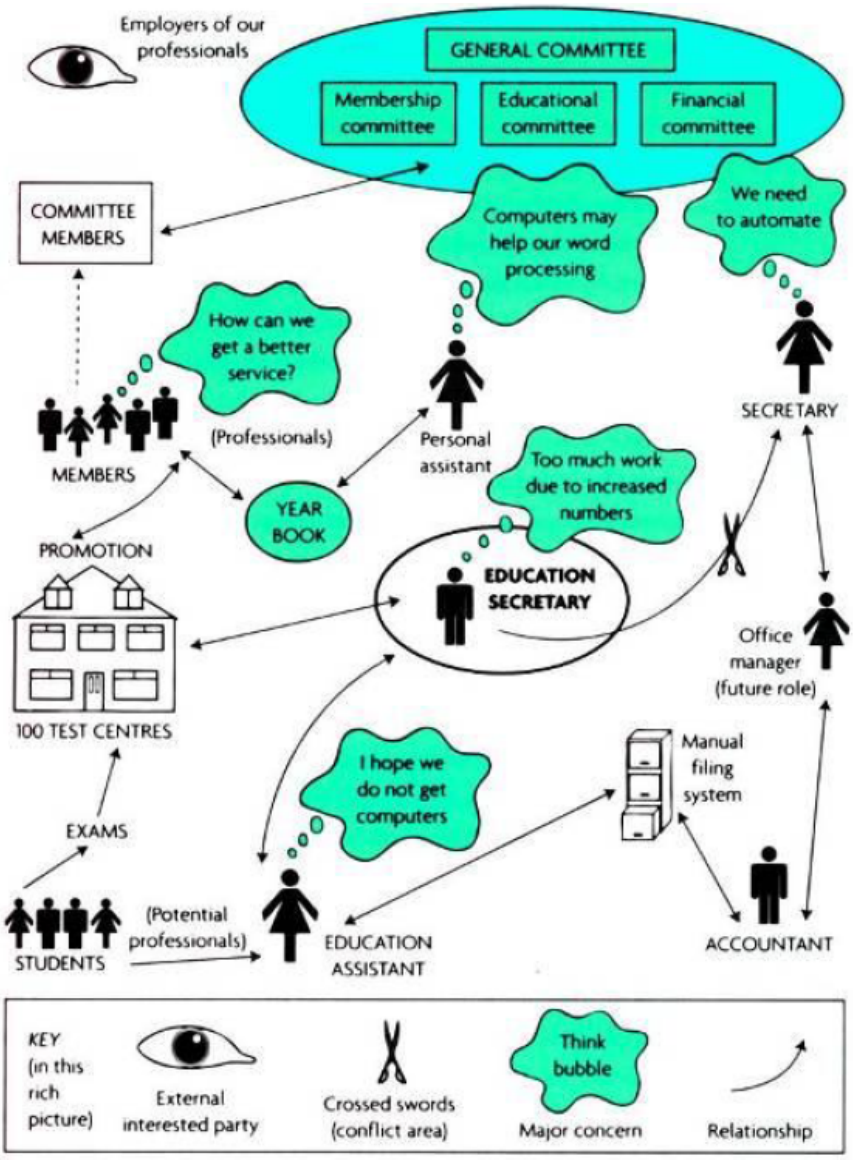
- P1 – Prijavni list sa ličnim podacima kandidata, podacima o mestu rođenja, državljanstvu, roditeljima, prebivalištu, prethodno završenoj instituciji i željama kandidata.
- P2 – Identifikacioni list sa imenom, prezimenom, JMBGom i brojem prijave.
- P3 – Formular za diplome koje kandidat prijavljuje i na osnovu kojih želi da bude oslobođen od prijemnog ispita.

3.5 Druge tehnike

Pored tehnika za opisivanje podataka, procesa i objekata postoje i druge tehnike. To su holističke tehnike, i upravljačke tehnike. Upravljačke tehnike su: organizacione tehnike, tehnike vođenja projekata, i tehnike rada sa ljudima.

3.5.1 Holističke tehnike

Holističke tehnike su one koje pokušavaju da sagledaju čitav sistem (ili podsistem) kao celinu, bez posebnog odvajanja procesa od funkcija ili strukture. Namenjene su, pre svega, nestručnim licima.



Slika 3.49: Holističke tehnike – primer velike slike.

Primer jedne takve tehnike je "velika slika" (engl. *rich pictures*). Naime, na relativno neformalan način se predstavljaju najvažniji elementi sistema i povezuju strelicama. Broj i detaljnost elemenata zavise od raspoloživog prostora. Ravnopravno se kombinuju različiti elementi (procesi, entiteti, subjekti, itd.). Na slici 3.49 prikazan je primer velike slike.

Na ovom primeru možemo diskutovati o tome koliko su ove tehnike informativne za nas. Neku načelnu informaciju o nekakvim odnosima ili subjektima možemo sagledati, kao i to šta neko očekuje od našeg sistema.

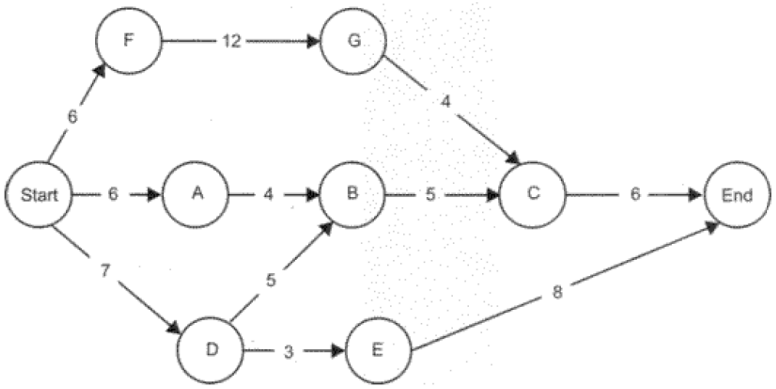
Holističke tehnike ovog tipa imaju smisla samo u ranim fazama razvoja. Kada pokušavamo da sagledamo neki sistem, odnosno, da probamo da ga grafički predstavimo, na ovaj način možemo da vidimo da li smo dobro razumeli ono što smo saznali o sistemu. Nakon pravljenja neke predstave sistema, očekujemo da nam klijent (ili neko ko poznaje domen) to odobri ili odbaci.

3.5.2 Tehnike vođenja projekata

Tehnike vođenja projekta spadaju u već pomenute upravljačke tehnike. *Upravljačke tehnike* omogućavaju bolje praćenje razvoja.

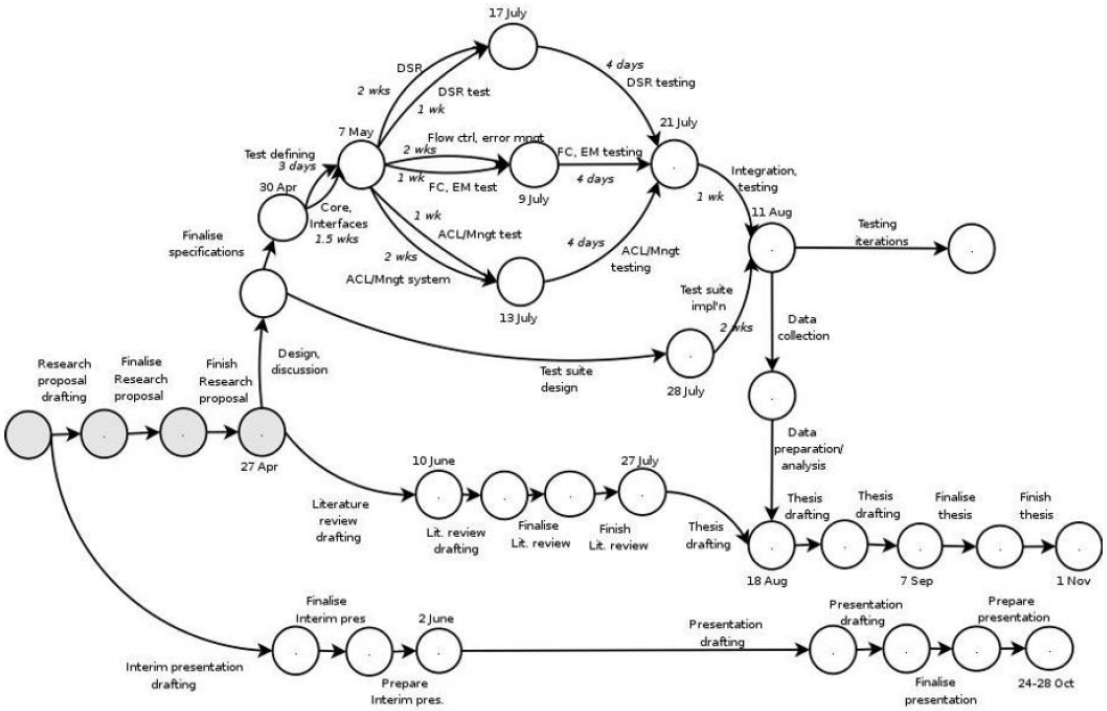
Tehnike vođenja projekata služe da se omogući lakše sagledavanje toka projekta i međuzavisnosti pojedinačnih koraka. Primeri tehnika vođenja projekta su: PERT dijagrami, i GANT dijagrami.

U PERT dijagramima, krugovima se označavaju određena stanja projekta ili određeni poslovi koji su obavljeni (može bili ili neobavljeni, ali obično krug predstavlja obavljen posao), na primer, napravljena skica korisničkog interfejsa. Stanja su povezana usmerenim strelicama, pri čemu oznaka (broj) na strelici predstavlja broj sati, radnih dana, ili neke druge vremenske odrednice, koji je bio potreban da se posao završi.



Slika 3.50: Tehnike vođenja projekta – primer PERT dijagrama 1.

Bitno je napomenuti da se u PERT dijagramu ne posmatra sistem koji se pravi, nego proces razvoja tog sistema. Na slikama 3.50 i 3.51 dati su primeri PERT dijagrama. Na slici 3.51 opisano je koji posao treba uraditi da bi se napravio prelaz iz jednog stanja u drugo. Neke značajne tačke su obeležene datumima što predstavlja da poslovi do tada moraju biti obavljeni. Dobra strana ovog pristupa je poboljšana vidljivost međuzavisnosti. Mana ovog pristupa je potencijalna nepreglednost, kao i to da nije dovoljno jasno dokle se stiglo u razvoju.

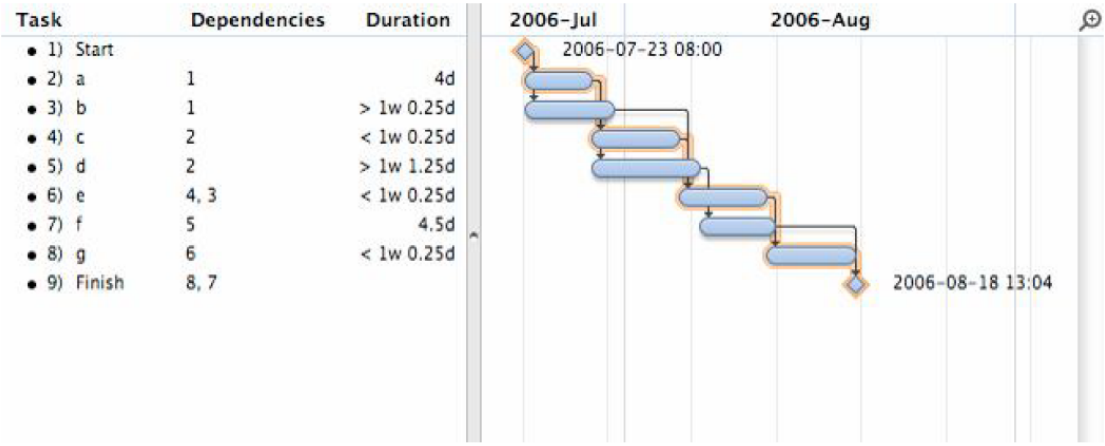


Slika 3.51: Tehnike vođenja projekta – primer PERT dijagrama 2.

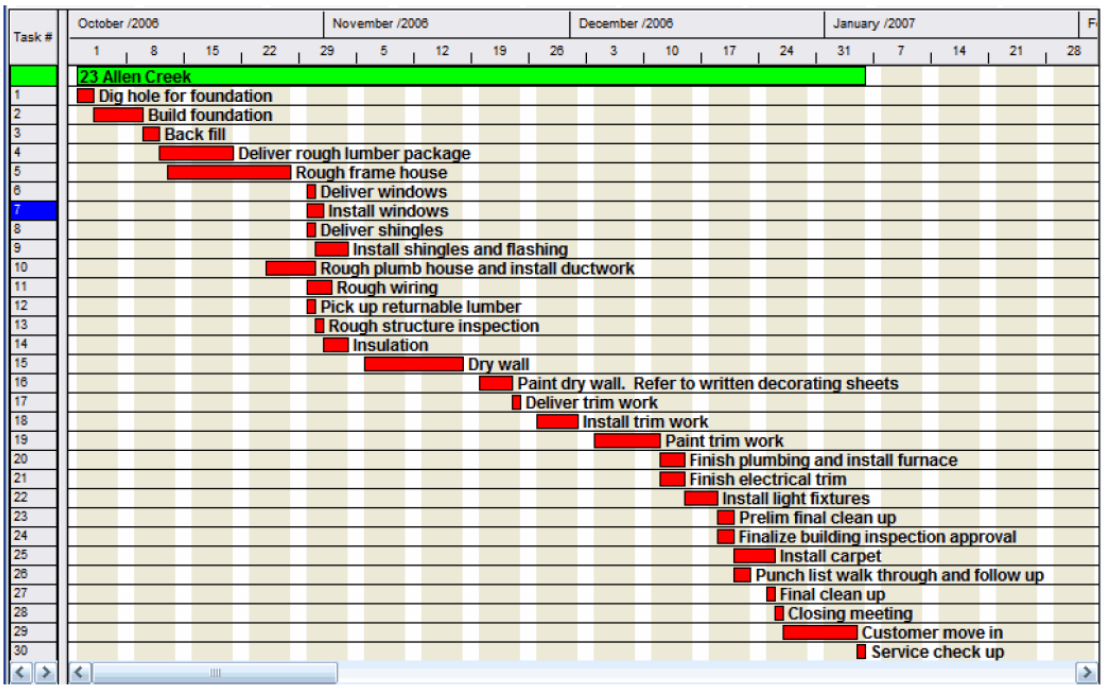
GANT dijagrami predstavljaju alternativu PERT dijagramima. Suštinska razlika je da se stanja, koja su predstavljana krugovima u PERT dijagramu, u GANT dijagramu predstavljaju strelicama. Takođe, obavljanje posla, što je predstavljano strelicama u PERT dijagramu, u GANT dijagramu se predstavlja pravougaonicima.

Cilj korišćenja GANT dijagrama je, u suštini, isti kao cilj korišćenja PERT dijagrama, a to je opisivanje međuzavisnosti poslova. Međutim, u GANT dijagramu se opisuje koji posao u kom trenutku vremena planiramo da započnemo i koliko dugo očekujemo da se radi. U tom smislu, GANT dijagrami su danas (praktično) nezamenljivo sredstvo u bilo kom složenom projektu.

Na slikama 3.52 i 3.53 dati su primeri GANT dijagrama. Drugi primer dijagrama ne oključuje međuzavisnosti poslova.



Slika 3.52: Tehnike vođenja projekta – primer GANT dijagrama 1.

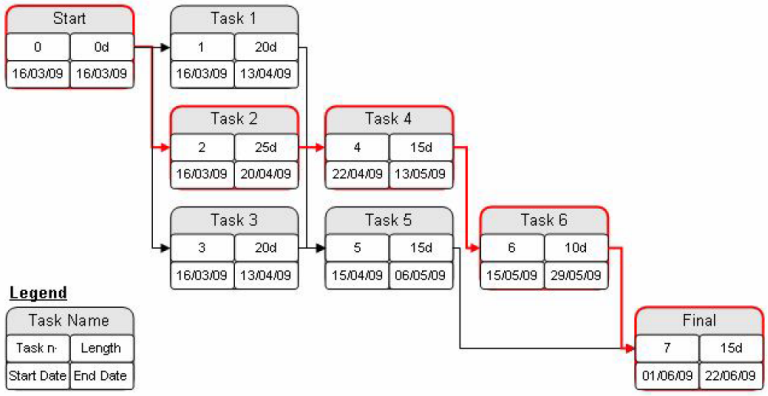


Slika 3.53: Tehnike vođenja projekta – primer GANT dijagrama 2.

Zanimljivo je napomenuti da danas postoji veliki izbor onlajn alata za praćenje projekata koji, nakon unosa zadataka i njihovih međuzavisnosti, automatski generišu pomenute dijagrame iz tih podataka. Dobro napravljeni dijagrami mogu da se iskoriste i za izračunavanje očekivanog perioda do završetka posla, očekivanog obima posla, i drugo.

Postoje i hibridini pristupi koji kombinuju oba dijagrama. Jedan takav je dat na slici 3.54. Svaki čvor predstavlja pojedinačan posao, za koji postoji naziv,

identifikacija, obim posla, početak, i kraj.



Slika 3.54: Tehnike vođenja projekta – hibrid PERT i GANT dijagrama.

3.6 Pitanja

1. Koje vrste tehnika postoje u okviru razvojnih metodologija informacionih sistema?
2. Objasniti ulogu tehnika za opisivanje procesa. Navesti bar 5.
3. Navesti bar 7 tehnika za opisivanje procesa.
4. Objasniti funkciju i osnovne elemente dijagrama toka podataka?
5. Navesti, nacrtati i objasniti osnovne elemente dijagrama toka podataka.
6. Nacrtati primer dijagrama toka podataka u kome se pojavljuju svi osnovni elementi ove vrste dijagrama.
7. Šta je dijagram konteksta? Koji su njegovi osnovni elementi?
8. Šta obuhvata dijagram konteksta?
9. Nacrtati primer dijagrama konteksta.
10. Šta je drvo odlučivanja? Kada se koristi i čemu služi?
11. Napraviti primer drveta odlučivanja.
12. Šta je tabela odlučivanja? Kada se koristi i čemu služi?
13. Napraviti primer tabele odlučivanja.
14. Šta je strukturiran tekst? Kada se koristi i čemu služi?

15. Napraviti primer strukturiranog teksta.
16. Šta je dijagram akcija? Kada se koristi i čemu služi?
17. Napraviti primer dijagrama akcija.
18. Navesti bar 3 vrste matrica koje se upotrebljavaju pri opisivanju procesa.
19. Šta je matrica poslova i funkcija? Kada se koristi i čemu služi?
20. Napraviti primer matrice poslova i funkcija.
21. Šta je matrica entiteta i funkcija? Kada se koristi i čemu služi?
22. Napraviti primer matrice entiteta i funkcija.
23. Šta je dijagram strukture? Kada se koristi i čemu služi?
24. Napraviti primer dijagrama strukture.
25. Šta je životni ciklus entiteta? Kada se koristi i čemu služi?
26. Napraviti primer životnog ciklusa entiteta.
27. Šta je BPMN? Kada je nastala i gde se koristi?
28. Koje vrste dijagrama poslovnih procesa postoje u okviru BPMN? Po čemu se razlikuju?
29. Koji su osnovni elementi dijagrama procesa BPMN? Nacrtati ih i objasniti.
30. Nacrtati primer dijagrama procesa BPMN.
31. Koji su osnovni elementi dijagrama saradnje BPMN? Nacrtati ih i objasniti.
32. Nacrtati primer dijagrama saradnje BPMN.
33. Objasniti ulogu tehnika za opisivanje podataka. Navesti osnovne.
34. Šta je dijagram entiteta i odnosa? Koji su njegovi osnovni elementi?
35. Nacrtati primer dijagrama entiteta i odnosa.
36. Šta je dijagram klasa entiteta?
37. Nacrtati primer dijagrama klasa entiteta.
38. Objasniti razliku između dijagrama entiteta i odnosa i dijagrama klasa entiteta.
39. Šta je UML? Kada se upotrebljava? Čemu služi?

40. Koje vrste dijagrama postoje u UML-u? Kako se klasifikuju?
41. Navesti bar 5 vrsta strukturnih dijagrama UML-a i sasvim kratko objasniti kada se koji upotrebljava.
42. Šta je dijagram klasa? Navesti i objasniti osnovne elemente dijagrama klasa (UML).
43. Navesti primer dijagrama klasa koji sadrži sve glavne elemente (UML).
44. Navesti vrste asocijacija (dijagram klasa), objasniti i ilustrovati primerom svaku od vrsta (UML).
45. Kako se označava kardinalnost na dijagramu klasa (UML)? Nacrtati primer.
46. Koje elemente ima jedna klasa na dijagramu klasa (UML)?
47. Šta je, čemu služi i kada se upotrebljava dijagram komponenti (UML)?
48. Koji su osnovni elementi dijagrama komponenti (UML)?
49. Koje su sličnosti i razlike između dijagrama klasa i dijagrama komponenti (UML)?
50. Šta je, čemu služi i kada se upotrebljava dijagram objekata (UML)?
51. Koji su osnovni elementi dijagrama objekata (UML)?
52. Koje su razlike između dijagrama klasa i dijagrama objekata (UML)?
53. Šta je, čemu služi i kada se upotrebljava dijagram isporučivanja (UML)?
54. Šta je, čemu služi i koji su osnovni elementi dijagrama isporučivanja (UML)?
55. Navesti primer dijagrama isporučivanja (UML).
56. Šta je, čemu služi i koji su osnovni elementi dijagrama paketa (UML)?
57. Navesti primer dijagrama paketa (UML).
58. Navesti dijagrame ponašanja UML-a.
59. Šta je dijagram aktivnosti? Navesti i objasniti osnovne elemente dijagrama aktivnosti (UML).
60. Koje su sličnosti i razlike između dijagrama aktivnosti (UML) i dijagrama procesa (BPMN)?
61. Navesti primer dijagrama aktivnosti (UML) koji sadrži sve glavne elemente.

62. Šta je dijagram stanja? Navesti i objasniti osnovne elemente dijagrama stanja (UML).
63. Nacrtati primer dijagrama stanja (UML) koji sadrži sve glavne elemente.
64. Šta je slučaj upotrebe? Kada se upotrebljava i čemu služi? Kako se opisuje?
65. Šta je dijagram slučajeva upotrebe (UML)? Koji su njegovi osnovni elementi?
66. Šta obuhvata dokumentacija jednog slučaja upotrebe?
67. Koji su mogući odnosi između aktera (UML)? Kako se predstavljaju? Nacrtati primere.
68. Koji su mogući odnosi između aktera (UML)? Kako se predstavljaju? Nacrtati primere.
69. Koji dijagrami spadaju u dijagrame interakcije (UML)? Navesti i kratko objasniti namenu.
70. Šta je dijagram komunikacije (UML)? Čemu služi i kada se koristi?
71. Koji su osnovni elementi dijagrama komunikacije (UML)? Nacrtati primer.
72. Šta je dijagram interakcije (UML)? Čemu služi i kada se koristi?
73. Koji su osnovni elementi dijagrama interakcije (UML)? Nacrtati primer.
74. Šta je dijagram sekvence (UML)? Čemu služi i kada se koristi?
75. Koji su osnovni elementi dijagrama sekvence (UML)? Nacrtati primer.
76. Objasniti šta su holističke tehnike u okviru razvoja IS.
77. Šta su PERT dijagrami? Kada se, zašto i kako koriste?
78. Šta su GANT dijagrami? Kada se, zašto i kako koriste?

3.7 Literatura

Literatura za ovu temu su odabrana poglavlja iz [1] i [9], kao i članci Agile Modeling, *Introduction to the Diagrams of UML 2.X*, UML, i Randy Miller, *Practical UML^{TM}: A Hands-On Introduction for Developers*.

Glava 4

Strukturne metodologije razvoja

Razvojne metodologije se prema orijentaciji dele na: praćenje životnog ciklusa, procesno-orijentisane, kombinovane, objektno-orijentisane, brze razvojne metodologije (RAD), i organizaciono-orijentisane.

4.1 Procesno-orijentisane metodologije

U procesno-orijentisane metodologije spadaju: Strukturirana analiza, projektovanje i implementacija informacionih sistema (engl. *Structured Analysis, Design and Implementation of Information System (STRADIS)*), Jurdonov sistemski metod (engl. *Yourdon Systems Method (YSM)*), Džeksonov razvoj sistema (engl. *Jackson Systems Development (JSD)*), i druge metodologije.

4.1.1 STRADIS

Ceo razvoj je podeljen na određen broj koraka. Koraci nam služe da dovedemo sistem do određenog rezultata. Koraci u STRADIS razvojnoj metodologiji su:

- *Inicijalno izučavanje* (engl. *initial study*). Osnovni kriterijum je procena troškova i dobiti po pojedinačnim predloženim zahtevima. Obično se prave dijagrami toka podataka i interfejsa u postojećem sistemu. Procenjuju se vreme i troškovi koje bi zahtevala detaljnija analiza. Traje od 2 dana do 3–4 nedelje. Po okončanju se izveštaj podnosi nadležnim upravnim strukturama, da bi doneli odluku da li se ide u detaljnu analizu. Delimično odgovara studiji izvodljivosti, međutim, ne obuhvata alternativne slučajeve, i nije toliko iscrpna i obimna faza kao studija izvodljivosti.

- *Detaljno izučavanje* (engl. *detailed study*). U ovom koraku se vrši prepoznavanje korisnika na tri nivoa: stariji rukovodioci, srednji rukovodioci, i krajnji korisnici. Pravi se logički model, odnosno, prave se dijagrami toka podataka postojećeg sistema, ali u širem kontekstu, i prepoznaju se granice postojećeg sistema i granice potrebnog rešenja u odnosu na tokove podataka i procese.

Preciznije se određuju procene rokova i troškova. Takođe se vrše procene opterećenja po procesima/transakcijama/podacima.

- *Određivanje i projektovanje alternativnih rešenja.* Pronalaze se alternative postojećim rešenjima. Jasno se izražavaju ciljevi. Prave se dijagrami toka podataka novog sistema. U ovom koraku se susreću analitičari i projektanti. Rezultat ovog koraka obuhvata: tačne DTP postojećeg sistema, prepoznata ograničenja postojećeg sistema, logički model DTP novog sistema, i, za svaku od alternativa, označene delove DTP koje je potrebno implementirati, korisničke interfejse, procenu troškova i dobiti, okvirni plan i rokove implementacije, i rizike.

- *Fizičko projektovanje.* Obuhvata: dokumentovanje svih pojedinosti DTP-a (upotpunjavanje rečnika podataka, greške, posebni slučajevi, itd.), fizički projekat baze podataka i/ili sistema datoteka (uključujući normalizaciju, optimizaciju, itd.), izvođenje hijerarhije funkcionalnih celina (modula) na osnovu DTP-a, prepoznavanje poslova u kojima učestvuje osoblje, i procenu troškova implementacije i upotrebe (radno vreme razvojnog tima na razvoju, potrebni računarski resursi, troškovi razmene podataka, vreme razvojnog tima na pripremi dokumentacije, vreme korisnika pri upotrebi, i vreme na održavanju).

Dalje faze nisu eksplicitno određene, tj. ova metodologija je usmerena prvenstveno na analizu.

4.1.2 YSM

YSM i STRADIS su u osnovi slični. Obe metode vrše funkcionalnu dekompoziciju sistema i podražavaju analizu i projektovanje odozgo naniže. YSM ipak ima svoje specifičnosti. Kod YSM-a zapravo se radi o smeru analize "od unutra prema spolja". Naime, umesto da se prikupe sve informacije o sistemu, pa onda da se započne "slaganje" slike modela, u YSM razvojnoj metodologiji se isprva grubo posmatra sistem i u što kraćem vremenskom periodu, a zatim da se krene za detaljnim razrađivanjem. Počinje se od dijagrama konteksta relativno visokog nivoa, sa jasnim granicama sistema. Zatim se, na osnovu prikupljenih informacija (intervjui, upitnici, itd.) prepoznaju slučajevi koji se moraju modelirati. Na osnovu toka se od preliminarnog dijagrama prave kako opštiji modularni dijagrami tako i detaljniji dijagrami po modulima.

Koraci u YSM razvojnoj metodologiji su:

- *Studija izvodljivosti.* Uloga ovog koraka je sagledavanje postojećeg stanja, okruženja, i problema. Cilj je uopšteno razumevanje i sagledavanje postojećeg sistema. Metodi koji se koriste su pre svega intervjui, ali ima i drugih. Trajanje koraka je uobičajeno od 1 do 4 sedmice.

- *Suštinsko modeliranje* (engl. *essential modelling*). Ovaj korak predstavlja najznačajniji korak analize, i obuhvata poslovni model i model informacionog sistema. Modeli se prave uporedo sa prikupljanjem informacija. Ciljevi koraka su: model okruženja, model ponašanja, i opciono, model podataka. Model okruženja predstavlja poslovni model, i obuhvata granice postojećeg sistema, i granice budućeg sistema. Obično je to dijagram konteksta nivoa 0 (sistem se opisuje kao celina koja ima ulaze i izlaze, uključujući sve korisnike i događaje, druge sisteme, podsisteme, itd.). Model ponašanja podrazumeva kako se mora ponašati sistem da bi to bilo u skladu sa poslovnim okruženjem. Prave se DTP gde svakom događaju u poslovnom okruženju odgovara po jedan proces. Model podataka je obično opisan ER dijagramom.

- *Modeliranje implementacije.* U ovom koraku se započinje projektovanje sistema. Posmatranjem DTP i dijagrama stanja se prepoznaju granice dela sistema koji se podržava informacionim tehnologijama, tj. koji se implementira.

4.2 Kombinovane metodologije

U kombinovanje metodologije spadaju: Metod strukturne analize i projektovanja sistema (engl. *Structured Systems Analysis and Design Method (SSADM)*), *Merise*, *Information Engineering (IE)*, *Welti ERP Development*, i druge metodologije.

4.2.1 SSADM

SSADM predstavlja skup vrlo preciznih pravila i principa za rad, kao i standardizovano dokumentovanje. Slično kao i STRADIS, pokriva životni ciklus bez implementacije i održavanja, s tim da STRADIS obuhvata i implementaciju, ali bez detaljnog opisivanja koraka. Sagrađen je od celina, koje su dalje podeljene u nivoe:

- *Studija izvodljivosti.* U okviru prve celine nalazi se nivo 0:
    - *Izvodljivost.* Obuhvata: pripremu za studiju izvodljivosti, definisanje problema, pregled i izbor mogućih rešenja (dakle, više njih i detaljnije, za razliku od STRADIS-a), i pripremu izveštaja o izvodljivosti (što naručioca posla najviše zanima). Na osnovu pripremljenog izveštaja o izvodljivosti se ili ide dalje u razvoj ili se odustaje od razvoja ili se traži nova studija izvodljivosti.

- *Analiziranje zahteva.* U okviru druge celine nalaze se nivoi 1 i 2:
    - *Istraživanje postojećeg okruženja.* Ovaj nivo podrazumeva: uspostavljanje okvira za analizu, istraživanje i definisanje zahteva, istraživanje postojećih procesa, istraživanje postojećih podataka, pravljenje logičkog

pogleda na postojeće usluge (podatke i procese), i prikupljanje rezultata istraživanja i pravljenje izveštaja.

  - *Moguća rešenja u poslovnom sistemu.* Ovaj nivo podrazumeva: prepoznavanje mogućih rešenja u poslovnom sistemu, izbor mogućih rešenja u poslovnom sistemu, i definisanje zahteva. Na ovom nivou se predlažu arhitekture rešenja.

- *Specifikacija zahteva.* U okviru treće celine nalazi se nivo 3:

  - *Definisanje zahteva.* Obuhvata: definisanje zahtevanih sistemskih procesa, razvoj neophodnog modela podataka, izvođenje funkcija sistema, unapređivanje modela podataka, izvođenje prototipa specifikacije, izvođenje specifikacije procesa, potvrđivanje sistemskih ciljeva, i komponovanje specifikacija zahteva i pravljenje izveštaja. Dakle, rezultat je sve ono što opisuje sistem, ali bez modela.

- *Logička specifikacija sistema.* Predstavlja dekompoziciju, prepoznavanje komponenti, odnosno, u terminologiji strukturnog modeliranja, module, i oblikuje na koji način će izgledati rešenje. Dakle, opisano je i šta će biti implementirano. U okviru četvrte celine nalaze se nivoi 4 i 5:

  - *Tehničke mogućnosti sistema.* Ovaj nivo obuhvata: prepoznavanje mogućih tehničkih rešenja, izbor tehničkih rešenja, i definisanje celina (modula) fizičkog projekta.
  - *Logički projekat.* Podrazumeva: definisanje korisničkog interfejsa (formulara), definisanje procesa menjanja podataka, definisanje procesa čitanja podataka, i komponovanje logičkog projekta.

- *Fizičko projektovanje.* Opisuje, ali u detalje, odnosno, na niskom nivou, šta će biti implementirano. U okviru pete celine nalazi se nivo 6:

  - *Fizički projekat.* Ovaj nivo obuhvata: pripremu za fizičko projektovanje, fizičko projektovanje podataka, definisanje plana implementacije, optimizacija fizičkog projekta, puna specifikacija sistema, konsolidacija interfejsa procesa, i komponovanje fizičkog projekta.

Slabost ovakvih metodologija jeste što su celine posmatrane strogo kao celine, gde se naručiocima posla sugerisala podela na celine, tako da jedna firma dobije jednu celinu, druga firma dobije drugu celinu, itd. jer "ako više firmi to radi, projekat će biti kvalitetniji". U praksi je to ipak drugačije. Neki od problema koji se javljaju su: nerazumevanje između firmi, nerazumevanje specifikacije, ponavljanje celina, i drugo. Ukoliko jedan tim radi korišćenjem strukturnih metodologija, onda to može imati više smisla. U narednom poglavlju ćemo se pozabaviti problemima do kojih može doći prilikom izrade informacionih sistema.

4.3 Pitanja

1. Kako se razvojne metodologije dele prema orijentaciji?
2. Navesti neke procesno orijentisane metodologije.
3. Šta je STRADIS? Navesti osnovne korake (bez opisivanja)?
4. Objasniti korak "Inicijalno izučavanje" (STRADIS).
5. Objasniti korak "Detaljno izučavanje" (STRADIS).
6. Objasniti korak "Određivanje i projektovanje alternativnih rešenja" (STRADIS).
7. Objasniti korak "Fizičko projektovanje" (STRADIS)?
8. Šta je Jurdonov sistemski metod? Navesti osnovne karakteristike.
9. Navesti i ukratko objasniti osnovne korake Jurdonovog sistemskog metoda.
10. Navesti neke kombinovane metodologije.
11. Šta je SSADM? Navesti osnovne karakteristike.
12. Navesti celine i nivoe SSADM.
13. Objasniti celinu "Studija izvodljivosti" (SSADM).
14. Objasniti celinu "Analiziranje zahteva" (SSADM).
15. Objasniti celinu "Specifikacija zahteva" (SSADM).
16. Objasniti celinu "Logička specifikacija zahteva" (SSADM).
17. Objasniti celinu "Fizičko projektovanje" (SSADM).

4.4 Literatura

Literatura za ovu temu su odabrana poglavlja iz [1] i [9].

Glava 5

Problemi pri razvoju IS i njihovo rešavanje

5.1 Problemi i neuspeh

Problem je sve ono što negativno utiče na proces i rezultat razvoja informacionog sistema. Osnovno merilo neuspeha je izgubljena materijalna vrednost: plaćena cena neuspeha, uložena sredstva, izgubljeno vreme, posledice po čitav poslovni sistem.

Vrste neuspeha su: prekoračenje troškova, prekoračenje vremenskih rokova (što proizvodi dodatne troškove zbog produženog razvoja i kašnjenja puštanja u rad), rezultat nije upotrebljiv (na primer, nije napravljen u planiranim razmerama, ili je sistem implementiran u skladu sa zahtevima, ali ne odgovara stvarnim potrebama, ili nisu ispunjeni nefunkcionalni zahtevi), i na kraju, odustajanje od projekta usled nekog od prethodnih razloga (ili više njih).

5.2 Neupotrebljiv rezultat

Čak i kada postoji planiranje i kada se projekat implementira po planu, rezultat može da bude neupotrebljiv ili da predstavlja neuspeh.

Aspekti "neupotrebljivosti" su: neupotrebljiv korisnički interfejs, i procedura korišćenja nije ostvariva ili dobra u realnim uslovima. Neupotrebljiv korisnički interfejs se ogleda kroz: loše rešene ergonomske aspekte, nepostojanje fizičkog (hardverskog) odziva, neintuitivan izgled korisničkog interfejsa, problematične kontrole interfejsa, spor odziv, nepouzdanost, itd. Procedura korišćenja nije ostvariva ili dobra u realnim uslovima se može ogledati na različite načine. Na primer, uz kupljenu knjigu se dobija besplatna sveska, ali to mora da se zavede kao prodaja. Ako sistem ne omogućava manuelnu promenu cene, to neće biti moguće. Ako sistem ne dopušta da se cena manuelno postavi na 0, takođe neće biti moguće. Drugi primer je

informacioni sistem koji omogućava upravi da tačno vidi i kritikuje magacionere zbog neracionalnog zauzeća prostora, a ne omogućava automatsku podršku u vidu saveta. Magacioneri počinju da troše nesrazmerno mnogo vremena na planiranje prostora. Iako rezultat jeste bolja iskorišćenost prostora, gubi se na većem utrošku radnog vremena. Još jedan primer bio bi pretraživanje knjiga (ili sličnog kataloga) koje podrazumeva unošenje tačnog naslova.

Mogući uzroci koji indukuju neupotrebljiv rezultat su: nerealni ili nejasni ciljevi projekta, neprecizna procena potrebnih resursa, loše definisani zahtevi, i slaba komunikacija između klijenta, razvijaoca i korisnika.

5.2.1 Najčešći problemi

Problemi mogu biti na strani klijenta ili na strani razvijaoca, ali mogu biti i višetrani. U probleme na strani klijenata svrstavamo nerealne ili nejasne ciljeve projekta, neusklađenost ciljeva i strategije, politiku ulagača, komercijalni pritisak, otpor korisnika prema primeni IS. Problemi na strani razvijaoca su slabo vođenje projekta, neprecizna procena potrebnih resursa, slabo izveštavanje o stanju projekta, neupravljani rizici, upotreba "nezrelih" tehnologija, nesposobnost da se iznese složenost projekta, tok praktičnog razvoja bez čvrstih principa i pravila. Višestrani problemi nastaju zbog slabe komunikacije između klijenta, razvijaoca i korisnika, nepoverenja između klijenta i razvijaoca i loše definisani zahtevi. Naravno, obično iza neuspeha stoji više razloga. Svaki od problema na strani razvojnog tima ima korene među rukovodiocima razvoja, i ostali problemi se često mogu na vreme prepoznati i otkloniti od strane rukovodioca razvoja.

Loše planiranje je jedan od najčešćih uzroka neuspeha. Ono ima dva osnovna oblika: nedovoljno dobro planiranje, i preterano planiranje. Nedovoljno planiranje se obično ogleda kroz nedovoljnu analizu problema i loše ili neprecizno definisane zahteve. Najčešće posledice nedovoljnog planiranja su: nesrazmerno veliki broj naknadnih korekcija zahteva, slaba upotrebljivost rešenja, i probijanje rokova. Možda izgleda paradoksalno, ali veoma čest uzrok problema je vid preteranog planiranja. Preterano planiranje se obično ogleda kroz preširoko i nekoncentrisano ulaženje u projekat, suviše duboka i obimna analiza sa ranim detaljnim projektovanjem, i preširoko ulaženje u implementaciju i preambicioznost, nesrazmerno realnim mogućnostima. Najčešće posledice su kasno uočavanje propusta, otežana tranzicija i probijanje rokova.

Nedovoljna interna kontrola takođe može izazvati neuspehe. Informacioni sistemi se mnogo češće unapređuju nego što se prave iznova. Poznata je parola "ako radi, ne diraj". Međutim, to često može biti uzrok problema jer su stare komponente razvijane na pretpostavkama koje možda više ne važe, tehnološki uslovi primene su možda izmenjeni, pa više nisu zadovoljeni, ili su vrlo verovatno izmenjene okolnosti i specifičnosti sredine u kojoj se komponente koriste. Nove tehnologije i savremeni

tokovi sasvim sigurno donose nove važne faktore i vrlo često su bezbednosne okolnosti izmenjene.

5.3 Upravljanje rizicima

Prema [2], definicija pojma upravljanje rizika glasi:

- *Upravljanje rizicima* je proces prepoznavanja, procenjivanja i kontrolisanja svega onoga što bi u projektu moglo da krene naopako pre nego što postane pretnja uspešnom dovršavanju projekta ili imlementacije informacionog sistema.

U upotrebi su i termini "analiza rizika" i "inženjering rizika". Upravljanje rizicima u informacionim sistemima nije vezano samo za razvoj informacionog sistema već i za svaki drugi složen razvojni ili graditeljski proces. U slučaju informacionih sistema najveći značaj ima dobro upravljanje rizicima u početnim fazama, ali je otežano u slučaju dugačkih razvojnih ciklusa.

5.3.1 SCERT

SCERT (engl. *Synergistic Contingency Evaluation and Review Technique*) sastoji se od četiri koraka: prepoznavanje značajnih aspekata, odgovarajućih rizika i mogućih odgovora, strukturiranje rizika i odgovora, prepoznavanje parametara po kojima se prepoznaje ishod, odgovarajućih scenarija i verovatnoća (parametri uključuju bar vreme, novac i bezbednost), i interpretacija parametara i rukovanje rizicima. Strukturiranje rizika i odgovora podrazumeva prepoznavanje opštih i specifičnih odgovora, prepoznavanje pravila odlučivanja i rezultat je lista rizika i odgovora. Interpretacija parametara i rukovanje rizicima procena nastupanja rizika tokom samog procesa i reagovanje na ostvarene rizike.

5.3.2 Upravljanje rizicima (*Boehm*, 1991)

Prema Boumu (*Boehm*, 1991), upravljanje rizicima sastoji se od procenjivanja rizika i njegove kontrole.

Procenjivanje rizika obuhvata tri koraka. Prvi, prepoznavanje rizika, podrazumeva pravljenje lista provera, ispitivanje pravila i pokretača odlučivanja, poređenje sa iskustvom (analiza pretpostavki) i dekompoziciju. Drugi, analiza rizika, sastoji se iz procena verovatnoće i iznosa rizika, modela performansi i troškova, mrežne analize (uslovljenosti i uticaji), statističke analize odlučivanja i analiza faktora kvaliteta. Treći, dodeljivanje prioriteta rizicima, obuhvata analizu izloženosti, analizu uticaja i redukciju povezanosti rizika.

Kontrola rizika, takođe, obuhvata tri koraka. Prvi, planiranje upravljanja rizicima, sadrži pripremu za suočavanje sa rizicima, kupovinu informacija, izbegavanje rizika, prebacivanje rizika, umanjenje rizika, pojedinačno planiranje rizika, integracija u opšti plan rizika. Drugi, razrešavanje rizika podrazumeva otklanjanje ili drugačije, prototipove, simulacije, testiranje, analiziranje i upravljanje kadrovima. Treći, praćenje rizika obuhvata praćenje progresa po tačkama (engl. *milestones*), detaljno praćenje nekoliko najvažnijih rizika ("*top 10*"), ponavljanje procena rizika, popravljanje uslova i eventualnih posledica.

Boum daje listu "*top 10*" rizičnih činilaca i tehnike upravljanja:

- manjak osoblja; tehnike upravljanja: popuna kadrova, odabir kadrova, unakrsna obuka, izgradnja morala, preraspoređivanje ključnih ljudi,
- nerealni rokovi i budžet; tehnike upravljanja: detaljno procenjivanje troškova iz više izvora, projektovanje prema troškovima/inkrementalni razvoj, pročišćavanje zahteva,
- razvoj pogrešnih funkcija; tehnike upravljanja: analiza organizacije i ciljeva, izrada prototipova, rana korisnička uputstva,
- razvoj pogrešnog interfejsa; tehnike upravljanja: izrada scenarija i prototipova, analiza zadataka,
- preterivanje ("pozlaćivanje"); tehnike upravljanja: pročišćavanje zahteva, izrada prototipova, analiza isplativosti, projektovanje prema troškovima,
- neprekidni niz izmena u zahtevima; tehnike upravljanja: postavljanje praga izmena, skrivanje informacija, inkrementalni razvoj,
- slabosti eksterno realizovanih poslova; tehnike upravljanja: provera referenci izvođača, ugovori sa kaznenim merama, raspisivanje nadmetanja za posao,
- slabosti u eksterno nabavljenim komponentama; tehnike upravljanja: referentni testovi, inspekcije, provera referenci, analiza kompatibilnosti, rano testiranje,
- slabe performanse u realnom radu; tehnike upravljanja: referentni testovi, simulacija, pravljenje prototipova, upravljanje i korigovanje postupaka,
- rad na granicama računarskih nauka; tehnike upravljanja: tehnička analiza,analiza isplativosti, provera referenci, pravljenje prototipova.

5.3.3 Procena izloženosti riziku

Najčešći rizici su: dodatni troškovi usled prekoračenja rokova, dodatni troškovi usled prekoračenja budžeta, i neispunjavanje nekog od zahteva. Neispunjavanje

funkcionalnog zahteva donosi rizike: pogrešna implementacija, pogrešan projekat, pogrešan zahtev (na primer, međusobna nekompatibilnost zahteva), itd. Rizici koji dolaze sa neispunjavanjem nefunkcionalanih zahteva su: loše performanse proizvedenog sistema, slaba primenjivost, kritične greške (bagovi), itd.

Neki od uzroka problema se mogu otkloniti izmenjenim pristupom problemu razvoja informacionih sistema. Savremene razvojne metodologije počivaju na konceptima koji omogućavaju smanjivanje rizika. Neki od najvažnijih savremenih kocepata razvoja koji značajno smanjuju rizike su: inkrementalni razvoj, određivanje koraka prema rokovima, pojačana komunikacija među subjektima, pristup "u žiži su objekti, a ne procesi", i pravljenje prototipova.

Svaki ozbiljan razvojni tim vrši procenu izloženosti rizicima. Za svaki nezadovoljavajući ishod NI_i se određuje verovatnoća njegovog nastupanja $P(NI_i)$ i gubitak usled nastupanja $G(NI_i)$. Izloženost riziku se izračunava kao

$$\sum_i P(NI_i) * G(NI_i).$$

Za različite načine rešavanja problema se prave procene izloženosti riziku. Procenjuje se odnos rizika i procenjene vrednosti obavljanja posla primenom odgovarajuće grane.

Na slikama 5.1 i 5.2 data su dva primera procena izloženosti riziku, pri izradi informacionog sistema korišćenjem dve varijante.

Rešenje	Rizik i ver. nastupanja	Razrešavanje	Ver. nastup.	Vrednost	Procenjena vrednost
Varijanta 1					**120.000**
	Rizik 1 (0,30)	Prepoznavanje u ranoj fazi i brzo otklanjanje (0,20)	0,06 $(= 0,30 \cdot 0,20)$	10.000	600
		Prep. u ranoj fazi i sporo otklanjanje (0,40)	0,12	20.000	2.400
		Prep. i otk. u kasnoj f. (0,35)	0,015	30.000	3.150
		Neprepoznavanje (0,05)	0,015	80.000	1.200
		Izostanak	0,70	0	0
		Procena:			**7.350**
	Rizik 2 (0.25)	Prep. i otk. (0,85)	0,2125	10.000	2.125
		Neprepoznavanje (0,15)	0,0375	90.000	3.375
		Izostanak	0,75	0	0
		Procena:			**5.500**
	Rizici:				**12.850**
	Ukupno:				**132.850**

Slika 5.1: Primer procene izloženosti riziku – varijanta 1.

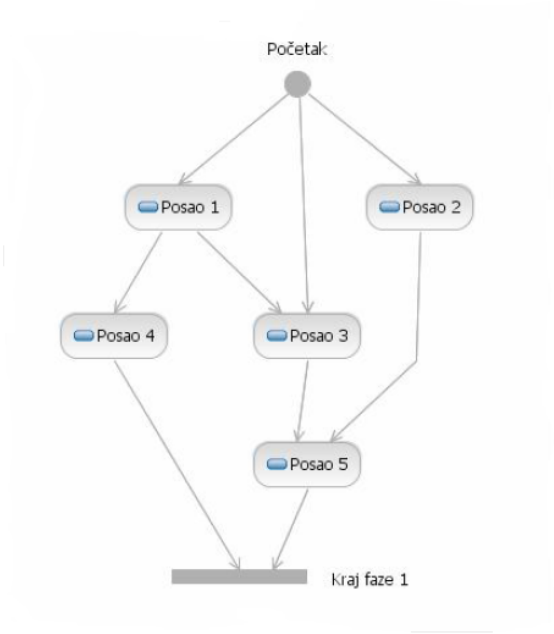
5.3.4 Međusobna zavisnost poslova

Tokom razvoja postupak se deli na manje poslove. Za svaki od njih se pravi posebna procena rizika. Neuspeh jednog od poslova može da se preslikava na

druge u vidu kašnjenja, odlaganja, odustajanja ili promene zahteva. Zbog toga se prave dijagrami zavisnosti poslova. Za svaki posao se vodi evidencija rizika, rokova i troškova. Na taj način se može lakše sagledati i izračunati kako rizici u nekom od poslova utiču na rizike u nekom drugom, pa i u čitavom projektu (slično dijagramima PERT i GANT). Primer takvog dijagram dat je na slici 5.3.

Rešenje	Rizik i ver. nastupanja	Razrešavanje	Ver. nastup.	Vrednost	Procenjena vrednost
Varijanta 2					**100.000**
	Rizik 3 (0,20)	Prep. u ranoj f. i brzo otklanjanje (0,20)	0,04	15.000	600
		Prep. i otk. u kasnoj f. (0,75)	0,15	45.000	6.750
		Neprepoznavanje (0,05)	0,05	380.000	19.000
		Izostanak	0,80	0	0
		Procena:			**26.350**
	Rizik 4 (0,35)	Prep. i otk. u ranoj fazi (0,75)	0,2125	8.000	1.700
		Prep. i otk. u fazi 2 (0,10)	0,12	16.000	1.920
		Prep. i otk. u kasnoj f. (0,05)	0,105	25.000	2.625
		Neprepoznavanje (0,10)	0,0375	140.000	5.250
		Izostanak	0,65	0	0
		Procena:			**11.495**
	Rizici:				**37.845**
	Ukupno:				**137.845**

Slika 5.2: Primer procene izloženosti riziku – varijanta 2.



Slika 5.3: Primer dijagrama međusobne zavisnosti poslova.

5.4 Pitanja

1. Šta znači pojam "problem" u kontekstu razvoja IS?
2. Šta znači pojam "neuspeh" u kontekstu razvoja IS?
3. Šta je merilo neuspeha u razvoju IS?
4. Koje su osnovne vrste neuspeha u razvoju IS?
5. Šta znači pojam "neupotrebljiv rezultat"? Kakvi vidovi neupotrebljivosti postoje?
6. Koji su najčešći problemi na strani naručioca IS?
7. Koji su najčešći problemi na strani razvijalaca IS?
8. Koje su vrste i odlike neuspeha usled "lošeg planiranja"?
9. Šta je upravljanje rizicima? Šta je cilj i čime se bavi?
10. Šta je SCERT?
11. Koji su osnovni elementi upravljanja rizicima?
12. Koje aktivnosti obuhvata prepoznavanje rizika?
13. Koje aktivnosti obuhvata kontrola rizika?
14. Navesti bar 8 od 10 najčešćih izvora rizika u razvoju IS?
15. Kako se izračunava procena izloženosti riziku? Navesti primer sa neka dva rizika.
16. Šta je dijagram zavisnosti poslova? Kada se pravi i čemu služi?
17. Nacrtati primer dijagrama zavisnosti poslova. Objasniti.
18. Kojim pristupima se utiče na smanjenje rizika u savremenim razvojnim metodologijama?

5.5 Literatura

Literatura za ovu oblast su odabrana poglavlja iz [1], [9], i [11], kao i članci Robert N. Charette, *Why Software Fails*, IEEE Spectrum, September 2005, i Barry W. Boehm, *Software Risk Management: Principles and Practices*, IEEE Software 8(1), 1991.

Glava 6

Objektno-orijentisane metodologije

6.1 Savremeni koncepti razvoja informacionih sistema

Neki od uzroka problema se mogu otkloniti izmenjenim pristupom problemu razvoja informacionih sistema. Ovo dovodi do prilagođavanja razvojnih metodologija. Savremene razvojne metodologije počivaju na konceptima koji omogućavaju smanjivanje rizika. Neki od najvažnijih savremenih koncepata razvoja informacionih sistema koji značajno smanjuju rizike su:

- *Inkrementalni razvoj.* Princip inkrementalnog razvoja je umesto da se razvoju pristupa kao velikom celovitom poslu, sa složenim projektom i implementacijom, posao se deli u niz inkrementalnih faza posvećenih manjim celinama posla. Postoje dve osnovne dobiti:

  - Svaka od faza je značajno manja i jednostavnija nego čitav posao, čime se pojednostavljuju planiranje i implementacija.
  - Veća je tačnost planiranja troškova i rokova i tačnije izveštavanje o toku razvoja.

  Postoje tri osnovna rizika:

  - Prvi je da ako se u prvim koracima potpuno zanemare predstojeći, postoji rizik da će u narednim koracima biti potrebne veće izmene.
  - Drugi je da ako se u prvim koracima uzmu u obzir svi predstojeći, postoji rizik da se njihova složenost približi složenosti čitavog sistema ("naduvavanje koraka"), čime ovakav pristup gubi smisao.
  - Treći je da često nije moguće sagledati unapred broj, cenu i ukupno trajanje svih koraka.

- *Određivanje koraka prema rokovima.* Princip je da se za svaki inkrementalni korak prvo odrede budžet i rokovi, a tek posle toga poslovi koji će korakom biti obuhvaćeni. Osnovne dobiti su:
  - Drastično smanjivanje rizika od prekoračivanja budžeta ili rokova.
  - Uspostavljanje ritma redovnog isporučivanja novih verzija sistema.
  - Redovnija kontrola kvaliteta i viši stepen poverenja između subjekata.
  - Postepeno prilagođavanje korisnika novim elementima sistema.

  Osnovni rizici su:
  - Uspostavljanje tesnih okvira inkrementalnog koraka može otežati pojedine korake i podići ukupnu cenu i trajanje razvoja.
  - Neki elementi sistema se ne mogu prirodno podeliti u različite korake.
  - Svaki korak zahteva troškove isporučivanja što u zbiru može da postane velika stavka u slučaju velikog broja malih koraka.
  - U prvim koracima se obično biraju poslovi koji donose veću dobit. Pri kraju razvoja postoji rizik da se ne implementiraju poslovi "čija je cena veća od dobiti" iako su značajni za sistem kao celinu.
- *Pojačana komunikacija među subjektima informacionog sistema.* Princip je da se u planiranju informacionog sistema, i svakog pojedinačnog koraka, uključuju u razmatranje sve vrste subjekata, u što već em broju. Osnovne dobiti su:
  - Dobija se tačnija slika o potrebnim ciljevima iz različitih uglova.
  - Smanjuje se rizik razvoja neupotrebljivog rešenja.
  - Subjekti se kroz proces razvoja pripremaju za upotrebu.

  Osnovni rizici su:
  - Previše informacija može dovesti do preteranog planiranja, a time i do "naduvavanja" pojedinačnih koraka.
  - Prevelikim razmatranjem mišljenja subjekata koji su navikli na postojeće procese i ne sagledavaju planirane izmene može se smanjiti obim suštinskih funkcionalnih izmena u okviru koraka, a time i nepotrebno povećati broj koraka da se dođe do "konačnog" rešenja.
- *U žiži su objekti, a ne procesi.* Princip je da se u središte pažnje se pri razvoju stavljaju objekti, a ne procesi. Osnovne dobiti su:
  - Objekti su obično stabilniji od procesa (tj. izmene u načinu poslovanja se manje odražavaju na objekte nego na procese).

  - Takav pristup je prilagođeniji inkrementalnom razvoju.
  - Smanjuje se rizik od pogrešnih odluka u ranim fazama razvoja (u ranim koracima se više pažnje posvećuje objektima, a u kasnijim koracima se više pažnje posvećuje procesima).

  Osnovni rizici su:

  - Potpuno zanemarivanje procesa u ranim fazama razvoja može voditi pogrešnoj arhitekturi sistema, što se kasnije veoma teško (skupo) menja.
  - Sasvim detaljno razmatranje procea u ranim koracima preti da dovede do "naduvavanja" prvih koraka.

- *Pravljenje prototipova.* Princip je da se u okviru analize problema pravi prototip koji odražava način funckionisanja informacionog sistema. Osnovne dobiti su:

  - Olakšava se netehničkim subjektima da u ranim fazama razvoja uoče određene nedostatke.
  - Smanjuje se rizik od pogrešnih odluka u ranim fazama razvoja.

  Osnovni rizici su:

  - Prototipovi obično odražavaju funkcionalne aspekte i elemente korisničkog interfejsa, ali ne i unutrašnju strukturu informacionog sistema. Posledica je da su samo delimično odgovarajući objektno-orijentisanim metodologijama.
  - Nedovoljno široko napravljen prototip i nedovoljno široko razmatranje prototipa mogu da prikriju nedostatke u drugim aspektima informacionog sistema.
  - Previše pažnje posvećene prototipu preti da "naduva" fazu njegove izrade.

6.2 Razvoj objektno-orijentisanih metodologija

Okolnosti pod kojima je nastupio razvoj objektno-orijentisanih metodologija su mnogobrojne. One obuhvataju: postojanje metodologija koje struturirano pristupaju analizi i opisivanju procesa, potreba za strukturiranim opisivanjem podataka, potreba za opisivanjem entiteta koji menjaju stanja, podizanje nivoa apstraktnosti posmatranja elemenata sistema, programiranje upravljano događajima, vizuelni korisnički interfejsi, povećana modularnost softvera, skraćivanje razvojnog ciklusa, tranzicija modela, višestruka upotrebljivost softvera, i drugo.

Za razliku od strukturnih, koje se u središte pažnje stavljale uređivanje procesa i algoritama, objektno-orijentisane metodologije u središte pažnje stavljaju uređivanje

objekata kojima se opisuje sistem. Razvoj objektno orijentisanih metodologija je počeo u vreme kada su slabosti prethodnih metodologija bile uglavnom poznate. U njih su ugrađeni neki od predstavljenih savremenih koncepata razvoja.

6.2.1 Osnovni koncepti

Sve objektno-orijentisane metodologije u žižu stavljaju objekte, a ne procese. Takođe, one se odlikuju skraćivanjem trajanja razvojnih ciklusa. Na primer, RUP (i druge) propisuju inkrementalni razvoj, dok agilne metodologije propisuju određivanje koraka prema rokovima i troškovima. Sve one podržavaju i pojačanu komunikacija među subjektima u svim fazama razvoja.

Osnovni koncepti objektno-orijentisanih metodologija su:

- *Enkapsulacija.* Struktura objekta je njihova interna stvar, odnosno, ona se ne sme izlagati spoljašnjem svetu. Atributima se sme pristupati samo posredno. Postoje slučajevi kada neki atributi predstavljaju osnovu ponašanja objekata ili klasa. Iako im se može omogućiti javni pristup, i tada je ipak preporučljivo enkapsuliranje. Enkapsulacijom se vrši apstrahovanje strukture metodima, i dobija se viši nivo međusobne nezavisnosti klase od modula u kojima se upotrebljava.

- *Interfejs.* Objekat (klasa) pruža spoljašnjim korisnicima samo skup metoda putem kojih mogu komunicirati sa njim. Takav skup metoda se naziva javni interfejs objekta (klase). Interfejs se oblikuje tako da omogući obavljanje jednog celovitog posla. Suština objekta je u njihovom ponašanju. Interfejs omogućava to ponašanje. Sužavanjem interfejsa na suštinu funkcionisanja objekta prikriva se sva složenost implementacije i pruža viši nivo nezavisnosti objekta od okruženja. Međutim, ako objekat ima više interfejsa, to znači da ima više funkcija, pa je potrebno razmotriti njegovo razlaganje na više objekata.

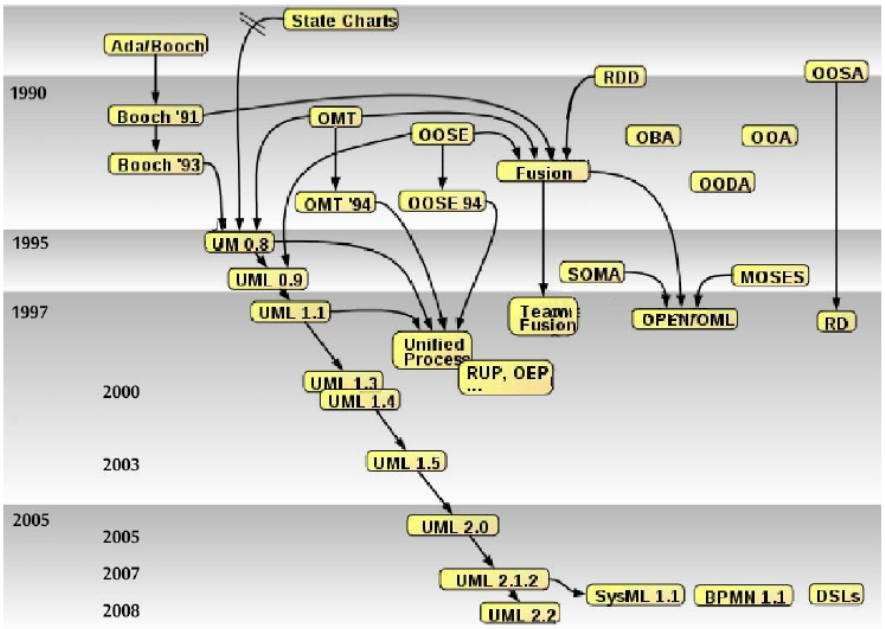
- *Polimorfizam.* Polimorfizam podrazumeva da se jednom napisan kod može upotrebljavati za različite vrste objekata. Postoje tri osnovne vrste polimorfizma: hijerarhijski, parametarski i implicitni[7]. Hijerarhijski polimorfizam koristi ispostavljene odnose "nadtip" i "podtip" među klasama kao sredstvo ostvarivanja polimorfizma. Parametarski polimorfizam počiva na upotrebi tzv. tipskih promenljivih. One se koriste za simboličko označavanje tipova vrednosti, promenljivih, i izraza. Implicitni polimorfizam podrazumeva da se pri pisanju koda uopšte ne navode tipovi vrednosti i izraza. Implicitni polimorfizam predstavlja uopštenje parametarskog polimorfizma. Polimorfizam omogućava pisanje apstraktnijeg koda, koji ima visoku upotrebljivost.

[7]Nekada se kao poseban oblik polimorfizma ubraja i tzv. ad-hok polimorfizam. On počiva na osobini nekih programskih jezika da dopuštaju višeznačnost imena funkcija i operatora (engl. *overloading*).

- *Nasleđivanje.* Nasleđivanje klase je ekvivalentno uvođenju jednosmerne parcijalno uređene relacije "jeste" između klasa. Kažemo da klasa A "jeste" klasa B akko svaki objekat klase A ima sve osobine koje imaju i objekti klase B. Ako A "jeste" klasa B kaže se i da je A "izvedena" klasa iz B ili A "je potomak" klase B, kao i da je B "osnovna" (ili "bazna") klasa za A ili B "je predak" klase A. Relacija "jeste" je parcijalna relacija poretka (refleksivna je i tranzitivna). Predstavlja osnovu za građenje hijerarhija klasa (bazne i izvedene klase formiraju hijerarhiju klasa). Nasleđivanje se koristi za eksplicitno označavanje sličnosti među klasama (objektima). Nasleđivanje predstavlja osnovu za hijerarhijski polimorfizam. Ako se nešto može uraditi sa svakim objektom klase B, onda se to može uraditi i sa svakim objektom klase A koja je izvedena iz B. Nasleđivanje se posmatra u dva smera, kao specijalizacija ili generalizacija: klasa A je poseban (specijalan) slučaj klase B, odnosno, klasa B je opštiji (generalan) slučaj klase A.

6.2.2 Pregled objektno-orijentisanih metodologija

Na slici 6.1 dat je grafički prikaz razvoja objektno-orijentisanih metodologija.



Slika 6.1: Pregled ogjektno-orijentisanih metodologija.

Razvoj se može podeliti u tri velike faze:

- *Početni koraci.* Ova faza traje do 1997. godine. Odlikuju je veliki broj različitih notacija i metodologija, pri čemu nijedna kompletna i dovoljno široka. Akcenat je na razvoju metoda i tehnika. Neke od najznačajnijih metodologija nastalih u ovom periodu su: (1) Booch, 1991, (2) Coad, Yourdon, 1991. i (3) Martin, Odell, 1992.

- *Oblikovanje UML-a.* Ova faza traje od 1995. do 2005. godine. Odlikuje je nekoliko dubljih metodologija koncentrisanih na različite faze razvoja, kao i pokušaji objedinjavanja metoda i tehnika i ujednačavanja notacije. Akcenat je na notaciji.

- *Post-UML koraci.* Treću fazu razvoja, koja traje od 2000. godine, odlikuje ujednačena notacija, široko shvatanje procesa razvoja, i potpuno posvećivanje metodologijama. Metologija koja je značajna je RUP, ali postoje i druge savremene metodologije.

6.3 Objektno-orijentisani analiza i dizajn (OOAD)

Objektno-orijentisani analiza i dizajn (OOAD) je metologija koja u centar pažnje stavlja objekte koji figurišu u sistemu. Ti objekti su: entiteti (sve vrste podataka koji postoje u sistemu), subjekti (različite vrste korisnika sistema), servisi (usluge koje sistem pruža korisnicima), i interfejsi (podsistemi putem kojih subjekti koriste usluge).

OOAD se načelno ne razlikuje mnogo za informacioni sistem i manji softver. Osnovni postupci (propisani od strane Coad-a i Yourdon-a, 1991.) jesu:

- *Pronalaženje klasa i objekata.* Analizira se domen i prepoznaju se stabilne klase i objekti. Oni će predstavljati jezgro budućeg sistema i aplikacija. Pri tome, posmatraju se i traže:

  - strukture,
  - drugi sistemi sa kojima posmatrani sistem sarađuje ili razmenjuje informacije,
  - uređaji koji se koriste u sistemu,
  - događaji koji se moraju pamtiti ili zapisivati, i
  - fizičke i geografska lociranost, organizacione jedinice i sastav timova.

  Značajni aspekti pri oblikovanju klasa su:

  - sve što je potrebno pamtiti čini potencijalne atribute,
  - potrebno ponašanje čini potencijalne metode,
  - objekti obično imaju više od jednog atributa, a ako postoji samo jedan, onda se objekat dovodi u pitanje,
  - klase obično imaju više od jednog objekta, a ako postoji samo jedan, onda se klasa dovodi u pitanje,
  - atributi moraju da se odnose na sve objekte klase,
  - ponašanje mora da se odnosi na sve objekte klase,

  - svi aspekti atributa i ponašanja moraju da potiču iz domena problema, a ne iz domena implementacije, i
  - izbegavanje redundantnosti i izvedenih atributa.

- *Prepoznavanje struktura.* Na osnovu analiza klasa i objekata utvrđuju se njihovi međusobni odnosi. Odnosi koji se prepoznaju su:
  - Generalizacija i specijalizacija. Jedan od osnovnih odnosa u OO metodologiji čine odnosi generalizacije i specijalizacije. Oni su osnova za građenje hijerarhija klasa, a osnovu za utvrđivanje ovih odnosa predstavlja ponašanje klasa. Potklasa je specijalan slučaj natklase, za koji važe sva pravila ponašanja natklase, ali postoje i neka nova pravila.
  - Asocijacija.
  - Agregacija i kompozicija.

- *Prepoznavanje subjekata.* Cilj prepoznavanja subjekata je u segmentaciji sistema na podsisteme po kriterijumu korisnika (cilj ima sličnosti sa DTP). Prepoznavanje subjekata je posebno značajno u velikim sistemima sa velikim brojem usluga ili klasa podataka.

- *Određivanje atributa.* Određivanje atributa klasa i objekata se ne razlikuje mnogo od odgovarajućeg postupka u tehnici modeliranja entiteta i odnosa. Nazivi atributa moraju biti u skladu sa rečnikom podataka. Za svaki atribut se određuje tip i dodaje opis.

- *Određivanje usluga.* Za razliku od strukturnih metodologija, u objektno-orijentisanim metodologijama se puna pažnja uslugama (postupcima, procesima) posvećuje "na kraju". Servisi odgovaraju "atomičnim" celinama poslova, koji bi trebalo da se u praksi odvijaju uvek u potpunosti. Obično se prepoznaju dve vrste usluga:
  - računske usluge, i
  - usluge praćenja i reagovanja na događaje (mogu biti automatske ili manuelne).

6.4 Pitanja

1. Kojim savremenim konceptima razvoja se pokušava smanjiti rizik nastupanja neuspeha?

2. Šta su osnovne odlike koncepta inkrementalnog razvoja? Kvaliteti i slabosti?

3. Šta su osnovne odlike koncepta određivanja koraka prema rokovima? Kvaliteti i slabosti?

4. Šta su osnovne odlike koncepta pojačane komunikacije među subjektima informacionog sistema? Kvaliteti i slabosti?
5. Šta su osnovne odlike koncepta stavljanja objekata u središte pažnje umesto procesa? Kvaliteti i slabosti?
6. Šta su osnovne odlike koncepta pravljenja prototipova? Kvaliteti i slabosti?
7. U kojim okolnostima se razvijaju objektno-orijentisane metodologije razvoja informacionih sistema?
8. Objasniti osnovne karakteristike objektno-orijentisane razvojnih metodologija.
9. Koje su osnovne karakteristike objektno-orijentisanih tehnika?
10. Šta je objekat? Navesti dve definicije.
11. Šta je klasa? Šta čini klasu?
12. Koji su osnovni koncepti objektno-orijentisanih metodologija?
13. Objasniti koncept enkapsulacije kod objektno-orijentisanih metodologija.
14. Objasniti koncept interfejsa kod objektno-orijentisanih metodologija.
15. Objasniti koncept polimorfizma kod objektno-orijentisanih metodologija.
16. Objasniti koncept nasleđivanja kod objektno-orijentisanih metodologija.
17. Koji se periodi mogu prepoznati u razvoju objektno-orijentisanih metodologija? Objasniti.
18. Šta znači skraćenica OOAD?
19. Koji je aspekt informacionog sistema primaran u slučaju OOAD?
20. Koji su osnovni postupci u OOAD (prema Coad-u i Yourdon-u)?
21. Kako se pronalaze klase u OOAD?
22. Šta su elementi strukture u OOAD? Kako se ustanovljavaju?
23. Kako se prepoznaju subjekti (OOAD)?
24. Kako se prepoznaju atributi (OOAD)?
25. Kako se određuju usluge (OOAD)?

6.5 Literatura

Literatura za ovu temu su odabrana poglavlja iz [1] i [9].

Glava 7

Rational Unified Process (RUP)

RUP je pristup razvoju softvera koji počiva na: iterativnom ciklusu, stavljanju arhitekture u središte procesa, i upravljanju prema slučajevima upotrebe.

To je "dobro" definisan i strukturiran proces razvoja softvera, ali i metodološki proizvod sa prilagodljivim okruženjem. RUP izučavamo zbog mnogobrojnih razloga. RUP predstavlja dobar primer velike savremene metodologije. On je objedinio većinu značajnih karakteristika starijih objektno-orijentisanih razvojnih metodologija. I druge savremene metodologije uglavnom počivaju na istim tim karakteristikama (to su pojmovi, postulati, principi, tehnike). Jedna od najznačajnijih osobina jeste što RUP podržava fleksibilan životni ciklus.

Osnovni elementi RUP-a su: skup postulata i principa na kojima se zasniva razvoj, razvojno okruženje i skup metoda, i jezik za opisivanje procesa i modela.

7.1 Pojmovi, postulati, i principi

Pojmovi u okviru RUP-a su:

- **Postulat** (ili **filozofija**). Predstavlja konceptualnu zamisao i okosnicu na kojoj počiva RUP.
- **Princip**. To je pravilo ili skup pravila čije poštovanje se pretpostavlja u okviru RUP-a.
- **Posao** (engl. *task*). To je vrsta aktivnosti koju obavlja pojedinac ili tim u okviru RUP-a.
- **Disciplina**. Predstavlja skup srodnih poslova koje može da obavlja pojedinac ili tim odgovarajuće stručnosti.
- **Rezultat rada** (ili **artefakt**). To je pojedinačni rezultat rada koji nastaje kao posledica obavljanja jednog ili više poslova u okviru metodologije.

Osnovni postulati na kojima RUP počiva su:

- **Prilagođavanje procesa**. *Koncepti* postulata su: složenost procesa mora odgovarati složenosti projekta i karakteristikama razvojnog tima, i formalnost elemenata procesa mora odgovarati konkretnim fazama. *Dobiti* postulata su: efikasan razvojni ciklus, povećana agilnost projekta, i realno planiranje i kompetentne procene.

- **Usklađivanje sukobljenih prioriteta**. *Koncepti* postulata su: razumevanje poslovnih prioriteta, rdređivanje prioriteta projekta i zahteva, uparivanje potreba sa mogućnostima softvera, razumevanje kojim se rezultatima može upravljati, i usklađivanje ponovne upotrebe rezultata sa potrebama. *Dobiti* postulata su: usklađivanje aplikacija sa poslovnim i korisničkim potrebama, smanjivanje zastupljenosti specifičnog razvoja, i optimizacija poslovne vrednosti.

- **Saradnja celokupnog razvojnog tima**. *Koncepti* postulata su: motivisanje osoblja za puno angažovanje, pravljenje samostalnih timova, podsticanje mešovitih timova (na primer, u jednom timu može da se nalazi analitičar, programer, tester, i drugi), obezbeđivanje efikasnog okruženja za saradnju, upravljanje proizvodima rada (artefaktima) u cilju podsticanja saradnje, i integrisanje poslovnih, softverskih i izvršnih timova. *Dobiti* postulata su: produktivnost timova, i veće slaganje poslovnih potreba i procesa razvoja.

- **Iterativno predstavljanje rezultata**. *Koncepti* postulata su: omogućavanje povratnih informacija inkrementalnim isporučivanjem novih rezultata korisnicima, prilagođavanje planova razvoja inkrementalnom procesu i dobijenim odgovorima korisnika, uvažavanje izmena i upravljanje izmenama, i suočavanje sa najvećim tehničkim, poslovnim i operativnim rizicima u ranim fazama. *Dobiti* postulata su: rano smanjenje rizika, visoka predvidivost, i poverenje među zainteresovanim stranama.

- **Podizanje nivoa apstrakcije**. *Koncepti* postulata su: ponovljena upotreba postojećih rezultata, upotreba alata i jezika višeg nivoa, radi smanjivanja obima i složenosti dokumentacije, fokusiranje na arhitekturu, i projekovanje sa akcentom na kvalitet, pouzdanost, razumevanje i upravljanje složenošću. *Dobiti* postulata su: produktivnost, i smanjena složenost.

- **Stalno posvećivanje kvalitetu**. *Koncepti* postulata su: puna kontrola tima nad kvalitetom, rano i kontinualno testiranje, paralelno sa razvojem, integrisanjem i isporučivanjem, i inkrementalno građenje projekta sa automatizovanim testiranjem. *Dobiti* postulata su: viši kvalitet, i raniji uvid u tok razvoja i kvalitet.

Osnovni principi procesa u RUP-u su:

- **Vizija**. Oblikovanje vizije podrazumeva jasno sagledavanje potreba i očekivanja od projekta. U to spadaju: pojmovi, problem, ulagači, karakteristike informacionog sistema, funkcionalni zahtevi (slučajevi upotrebe), nefunkcionalni zahtevi, i druge pretpostavke za ostvarivanje projekta. *Posledica odsustva* vizije je promašena tema ili cilj razvoja, tako da rezultat ne zadovoljava očekivanja.

- **Plan**. Pravljenje plana i upravljanja razvojem presudno utiče na rezultate projekta. Pravljenje plana obuhvata: planiranje iterativnog razvoja, planiranje sastava i dinamike angažovanja timova, planiranje protokola komunikacije, planiranje ujednačavanja dokumentacije, itd. *Posledica odsustva* plana je slabo praćenje toka razvoja i nedovoljno i nepouzdano ostvarivanje ostalih principa.

- **Rizici**. Prepoznavanje rizika i suočavanje sa najvažnijim rizicima presudno utiču na ritam i pouzdanost procesa. *Posledica odsustva* prepoznavanja rizika je fokusiranje na pogrešne probleme i mogućnost pojave neočekivanih problema.

- **Poslovni slučajevi**. Osnovna svrha poslovnog slučaja je omogućavanje ekonomskog i tehničkog planiranja ostvarivanja vizije. Koristi se za precizno procenjivanje profita od projekta i predstavlja opravdanje za projekat. Ne bavi se tehničkim karakteristikama problema već jasnim i konciznim obrazlaganjem zašto je projekat potreban. *Posledica odsustva* poslovnih slučajeva je rizik od besciljnog razvoja, sa nepotrebnim utroškom vremena i novca i rizikovanjem odustajanja od projekta.

- **Arhitektura**. Arhitektura softverskog sistema predstavlja organizaciju (strukturu) najvažnijih sistemskih komponenti, njihovih interakcija i interfejsa, sa konceptima uklapanja u šire okruženje. *Posledica odsustva* arhitekture je slaba mogućnost prilagođavanja, skaliranja, ponovne upotrebe, itd.

- **Prototip**. Prototip je sredstvo za omogućavanje ulagačima bez tehničkog obrazovanja da ostvare rani uvid u buduće rezultate projekta. *Posledica odsustva* prototipa je odlaganje uočavanja problema za kasnije faze.

- **Procenjivanje**. Redovno sagledavanje ostvarenih rezultata i procenjivanje njihove saglasnosti sa planiranom dinamikom razvoja je presudno za uspešno praćenje i planiranje. Svaka iteracija mora biti propraćena analizom uspešnosti i procenjivanjem kvaliteta i saglasnosti sa planovima. *Posledica odsustva* procenjivanja je zakasnelo uočavanje problema i neispravno planiranje iteracija.

- **Izmena zahteva**. Izmene zahteva u toku razvoja su neminovna realnost i razvoj ih mora uvažavati i prihvatati, ali ih i kontrolisati. Svako predstavljanje nekog rezultata, prototipa ili iteracije korisniku ima za posledicu talas izmenjenih ili novih zahteva. Formalizovanjem zahteva za promene se uvodi red u ovaj inače veoma rizičan prostor. *Posledica odsustva* izmena zahteva je neispravno planiranje iteracija i postepeno udaljavanje od stvarnih potreba.

- **Podrška korisnicima**. Svrha razvoja je izrada upotrebljivog proizvoda. Upotrebljivost podrazumeva podršku korisnicima. Vidovi podrške su: uputstvo za upotrebu, podsistem za pomoć, kratka i jasna uputstva na svakom od formulara, tehnička podrška, obuka, itd. *Posledica odsustva* podrške korisnicima je nezadovoljstvo korisnika, slab odziv i manjak kvalitetnih povratnih informacija, a na kraju i neupotrebljiv softver.

- **Prilagođavanje procesa**. Proces se mora prilagođavati promenama u projektu i okruženju u kome se projekat razvija. Prilagođavanje se može odlikovati: jačim upravljanjem u slučaju slabih rezultata, promenama u načinu odvijanja procesa radi uštede resursa ili podizanja kvaliteta, itd. *Posledica odsustva* prilagođavanja procesa je loša komunikacija, neusklađenost timova, neujednačenost dokumentacije, suvišan utrošak resursa, nizak nivo kvaliteta rezultata, itd.

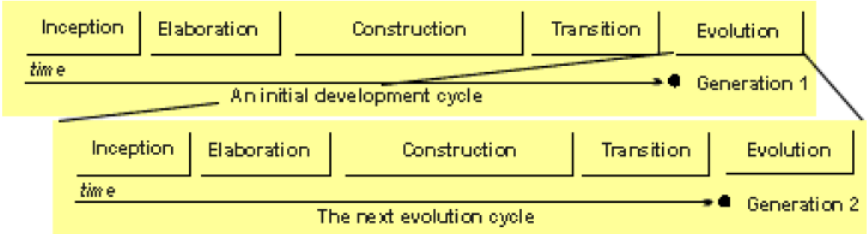
7.2 Životni ciklus projekta

Osnovu životnog ciklusa projekta čine četiri *perioda razvoja*:

- *Početak* (engl. *Inception*),
- *Razrada* (engl. *Elaboration*),
- *Konstrukcija* (engl. *Construction*)
- *Prelazak* (engl. *Transition*)

Za svaki od perioda razvoja važi da: počinje po završavanju prethodnog, završava jasno određenim rezultatom, na kraju obuhvata procenjivanje uspešnosti i eventualno ponavljanje nekih postupaka, i može biti iterativan. Čitav razvojni ciklus može da se ponavlja u istom ili izmenjenom obliku. Pojmovi koji se javljaju su:

- **Razvojni ciklus**. Predstavlja jedan prolazak kroz četiri perioda.
- **Generacija**. Predstavlja rezultat jednog razvojnog ciklusa.
- **Evolucioni ciklusi**. Predstavlja sve razvojne cikluse osim prvog. Obično važi da su im kraći periodi Početak i Razrada.



Slika 7.1: Životni ciklus projekta.

7.2.1 Početak (Inception)

Prvi period razvoja karakteriše određivanje šta se pravi. Alternativni nazivi perioda su *Uvod* i *Usvajanje*. Rezultati perioda su:

- vizija,
- osnovni poslovni slučaj,
- projektni zahtevi visokog nivoa. Akcenat je na inicijalnim slučajevima upotrebe (koji treba da budu od 10 do 20% dovršeni), i
- upravljanje okvirima projekta.

7.2.2 Razrada (Elaboration)

Drugi period razvoja karakteriše određivanje kako se pravi. Rezultati perioda su:

- detaljni zahtevi (80% dovršeni),
- definisanje stabilne i izvršne arhitekture. Implementiraju se neke ključne komponente. Očekuje se da je oko 10% koda implementirano, i
- zasnivanje arhitekture na ključnim slučajevima upotrebe. Oko 20% slučajeva upotrebe određuje 80% arhitekture. Ovde spada projektovanje, implementacija i testiranje nekih ključnih slučajeva upotrebe.

7.2.3 Konstrukcija (Construction)

Treći period razvoja karakteriše pravljenje proizvoda. Rezultati perioda su:

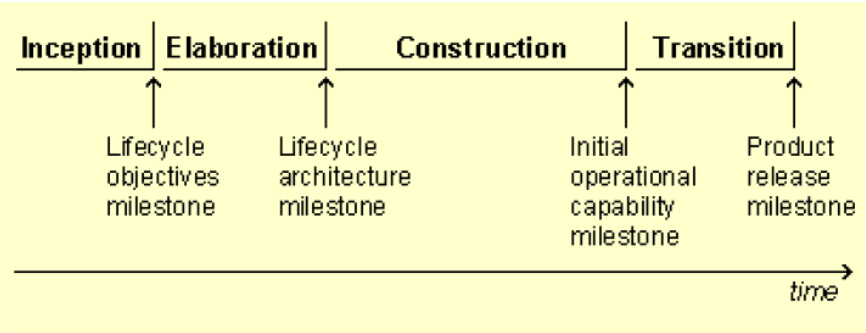
- dovršavanje zahteva i projektnog modela,
- projektovanje, implementacija i testiranje svake od komponenti. Vrši se upotreba prototipova i uključivanje krajnjih korisnika u proces, kao i inkrementalna evolucija izvršnog sistema do gotovog sistema,
- građenje dnevnih ili nedeljnih verzija sa automatizovanim procesom građenja,

- testiranje pri svakom građenju, i
- isporučivanje potpuno funkcionalnog softvera (tzv. *beta* verzije).

7.2.4 Prelazak (Transition)

Četvrti period razvoja karakteriše isporučivanje proizvoda krajnjim korisnicima. Rezultati perioda su:

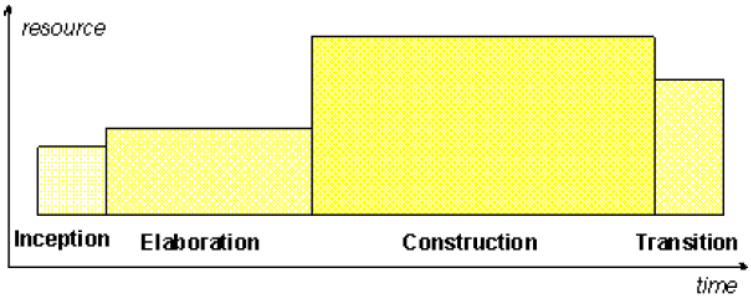
- pravljenje inkrementalnih izdanja radi otklanjanja neispravnosti, item ažuriranje korisničke i instalacione dokumentacije,
- ažuriranje opisa izdanja, i
- završne analize.



Slika 7.2: Životni ciklus projekta (2).

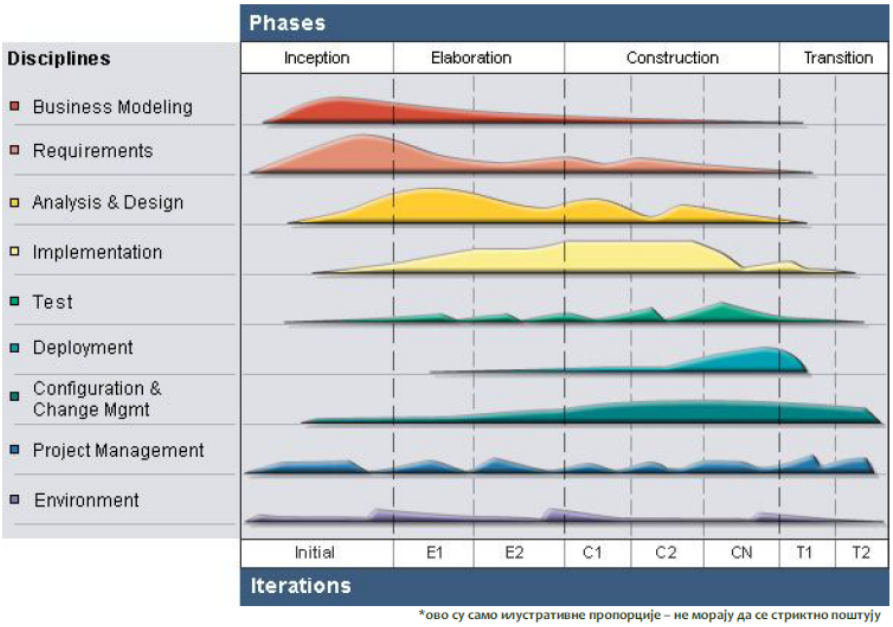
	Почетак	Разрада	Конструкција	Прелазак
Напор	5%	20%	65%	10%
Трајање	10%	30%	50%	10%

*ово су само илустративне пропорције – не морају да се стриктно поштују



Slika 7.3: Životni ciklus projekta (3).

Na slici 7.4 prikazan je jedan mogući grafik zastupljenosti disciplina po periodima. Iako većinu disciplina koristimo u svim periodima, ipak su neke discipline zastupljenije u određenim periodima (i razvojnim ciklusima).



Slika 7.4: Discipline u periodima razvoja.

7.2.5 Odnos sa pristupom vodopada

Životni ciklus projekta u RUP-u se umnogome razlikuje u odnosu na model vodopada. Osnovne razlike RUP modela u odnosu na model vodopada su:

- periodi razvoja se ne oblikuju prema postupcima nego prema ciljevima i očekivanim rezultatima,
- svaki period razvoja može da obuhvati aktivnosti iz različitih disciplina, od prikupljanja i analize zahteva, pa sve do implementacije i isporuke verzije, i
- rezultati prethodnih perioda se mogu menjati, unapređivati i dorađivati u narednim periodima razvoja.

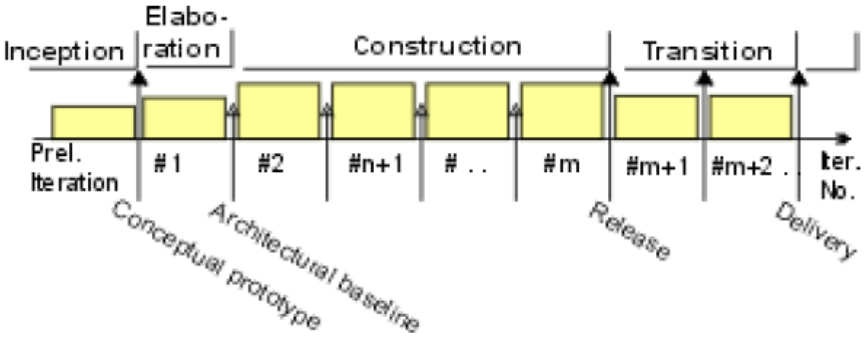
7.3 Strategije planiranja

Složenost i obim pojedinačnih perioda razvoja nisu strogo propisani već se mogu prilagođavati prirodi projekta i stvarnim potrebama. Šabloni planiranja životnog ciklusa se nazivaju *strategije*. Primeri strategija životnog ciklusa su: inkrementalna, evoluciona, inkrementalno isporučivanje, "Veliki projekat", i hibridne strategije.

7.3.1 Inkrementalna strategija

Osnovna ideja inkrementalne strategije je da se određuju korisničke potrebe, definišu se sistemski zahtevi, a zatim se ostatak razvoja odvija kroz sekvencijalnu izgradnju.

Inkrementalna strategija je odgovarajuća ako je: dobro poznat domen problema, dobro razumevanje rizika, i razvojni tim iskusan.



Slika 7.5: Inkrementalna strategija.

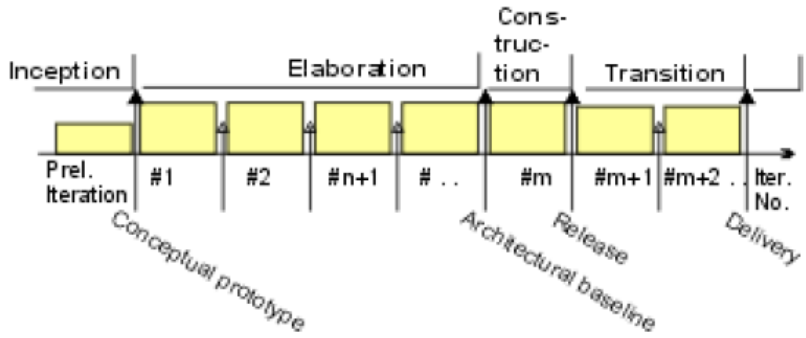
Karakteristične iteracije su:

- kratka iteracija Početak – definisanje okvira i vizije projekta i poslovnog slučaja,
- jedna iteracija Razrade – definisanje zahteva i uspostavljanje arhitekture,
- više iteracija Konstrukcije – slučajevi upotrebe se postepeno implementiraju i arhitektura se ostvaruje, i
- više iteracija Prelazaka – migriranje rezultata u korisničko okruženje.

7.3.2 Evoluciona strategija

Osnovna ideja evolucione strategije je da se pretpostavlja da se ne mogu unapred u potpunosti razumeti korisničke potrebe. Ne definišu se unapred svi slučajevi upotrebe, već se unapređuju u svakom koraku izgradnje.

Evoluciona strategija je odgovarajuća ako je domen problema nov ili nepoznati, i ako je razvojni tim neiskusan.



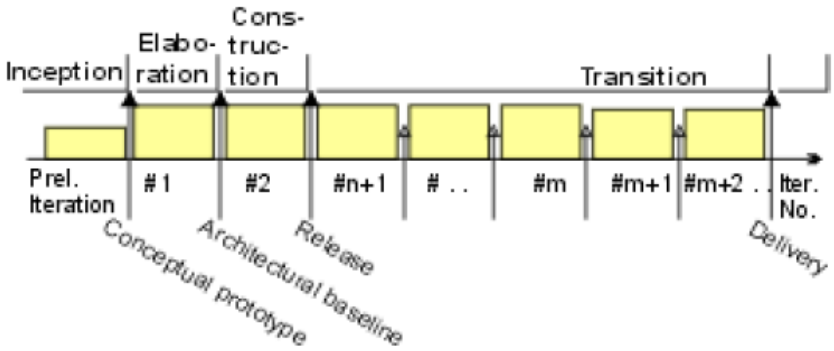
Slika 7.6: Evoluciona strategija.

Karakteristične iteracije su:

- kratka iteracija Početak – definisanje okvira i vizije projekta i poslovnog slučaja,
- više iteracija Razrade – postepeno definisanje i unapređivanje zahteva,
- jedna iteracija Konstrukcije – slučajevi upotrebe se implementiraju i arhitektura se proširuje, i
- više iteracija Prelazaka – migriranje rezultata u korisničko okruženje.

7.3.3 Strategija inkrementalno isporučivanje

Osnovna ideja strategije inkrementalno isporučivanje jeeste da je u nekim slučajevima važno rano isporučiti prve rezultate i tada se veći deo posla obavlja u iteracijama Prelaska. Prelazak započinje rano i ima najviše iteracija. Ova strategija zahteva veoma stabilnu arhitekturu, što je obično teško ostvariti u malom broju uvodnih iteracija.



Slika 7.7: Strategija inkrementalno isporučivanje.

Strategija inkrementalno isporučivanje je odgovarajuća ako je domen problema dobro poznat, arhitektura i zahtevi mogu da se stabilizuju rano u razvojnom ciklusu,

nizak nivo novina u problemu, tim je iskusan, i inkrementalno isporučivanje ima visoku vrednost za klijenta.

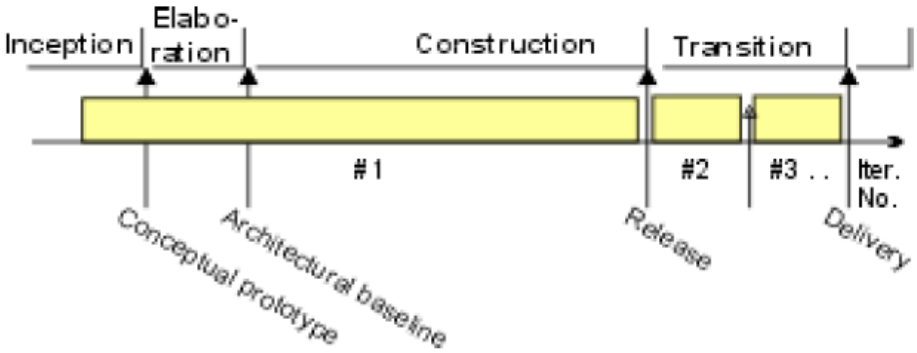
Karakteristične iteracije su:

- kratka iteracija Početak – definisanje okvira i vizije projekta i poslovnog slučaja,
- jedna iteracija Razrade – zasnivanje stabilne arhitekture,
- jedna iteracija Konstrukcije – slučajevi upotrebe se implementiraju i arhitektura se ostvaruje, i
- više iteracija Prelazaka – migriranje rezultata u korisničko okruženje.

7.3.4 Strategija "Veliki projekat"

Osnovna ideja strategije "Veliki projekat" je da tradicionalni pristup vodopada predstavlja izobličen slučaj sa samo jednom iteracijom u fazi Konstrukcije. Obično je posledica veliki broj iteracija u fazi Prelaska.

Strategija "Veliki projekat" je odgovarajuća ako se dodaje mali broj dobro definisanih funkcionalnih izmena proizvodu koji je veoma stabilan, nove funkcionalnosti su dobro definisane i shvaćene, i tim je iskusan i u domenu problema i sa postojećim proizvodom.



Slika 7.8: Strategija "Veliki projekat".

Karakteristične iteracije su:

- kratka iteracija Početak – definisanje okvira i vizije projekta i poslovnog slučaja,
- jedna veoma dugačka iteracija Konstrukcije – slučajevi upotrebe se implementiraju i arhitektura se ostvaruje, i
- više iteracija Prelazaka – migriranje rezultata u korisničko okruženje.

7.3.5 Hibridne strategije

Osnovna ideja hibridne strategije je da u praksi retko može da se prati neka tipična strategija. Zbog toga se obično dolazi do neke hibridne strategije, koja predstavlja kombinaciju različitih elemenata pomenutih strategija. Upravo je mogućnost oblikovanja i primenjivanja hibridnih strategija jedna od najvažnijih osobina faznoiterativnog razvoja koji propagira RUP.

Neki od pristupa hibridnim strategijama su sledeće:

- Za složene probleme ili nepoznate domene problema, gde je neophodno više istraživanja, povećava se broj iteracija u Razradi.
- Za razvojno složene probleme, gde se složenost prevodi u dizajn koda, povećavaju se broj i veličina iteracija u Konstrukciji.
- Ako je potrebno isporučivati proizvod inkrementalno, povećava se broj iteracija Prelaska.

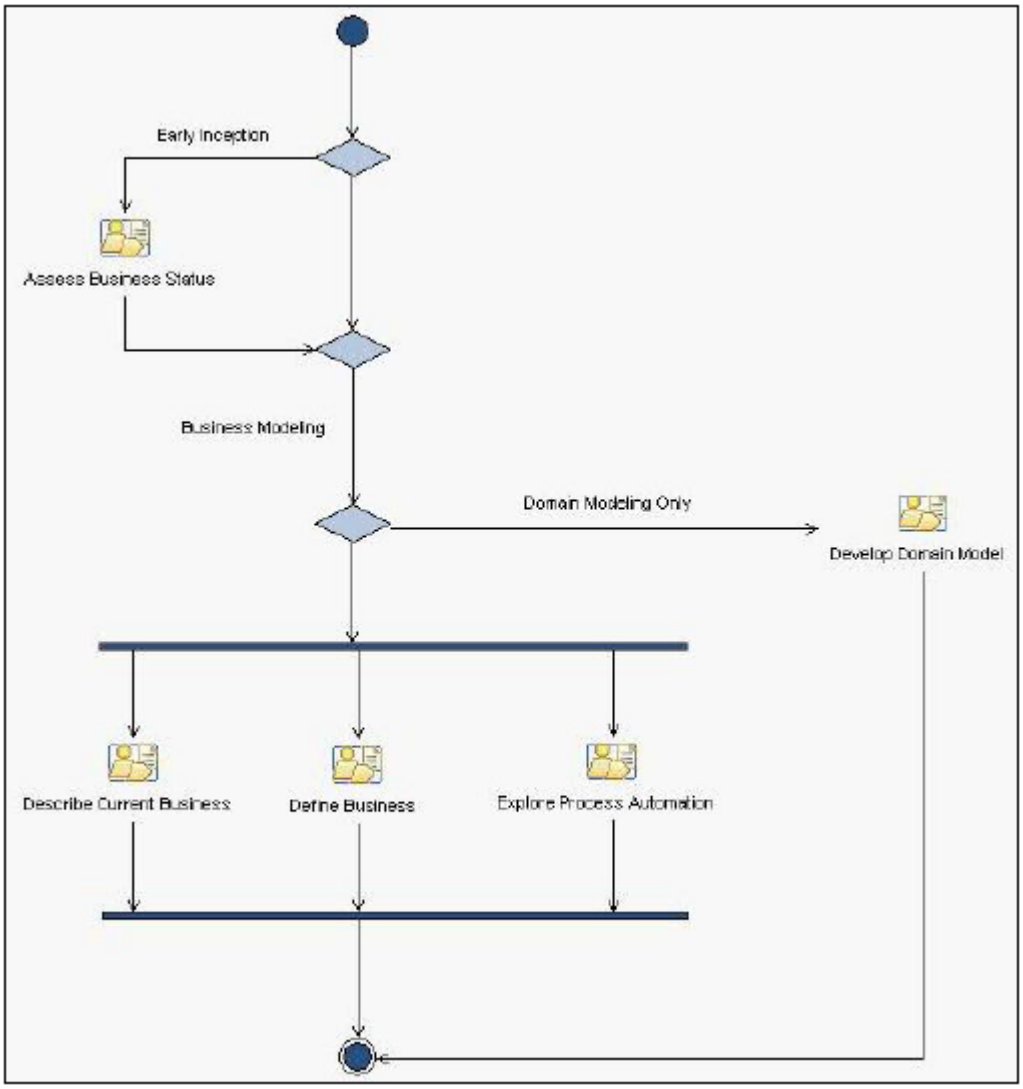
7.4 Discipline

Discipline koje se izučavaju u okviru RUP-a su: poslovno modeliranje, zahtevi, analiza i projektovanje, konfigurisanje i upravljanje izmenama, okruženje, implementacija, upravljanje projektom, i testiranje.

7.4.1 Poslovno modeliranje

Disciplina "Poslovno modeliranje" je posvećena: razumevanju postojećeg stanja, sagledavanju mogućih unapređenja, i stvaranju vizije budućeg sistema.

Svrha je: razumevanje postojećeg stanja i problema u poslovnoj organizaciji, prepoznavanje potencijala za unapređivanje, procenjivanje uticaja promena u organizaciji, staranje da svi ulagači na isti način razumeju organizaciju, izvođenje zahteva za softverski sistem, i razumevanje kako bi se budući sistem uklopio u organizaciju.



Slika 7.9: Tok poslova.

Na slici 7.9 prikazano je kako poslovi teku u disciplini poslovno modeliranje. Poslovi su grupisani u veće poslove na sledeći način (primetimo da se neki poslovi ponavljaju):

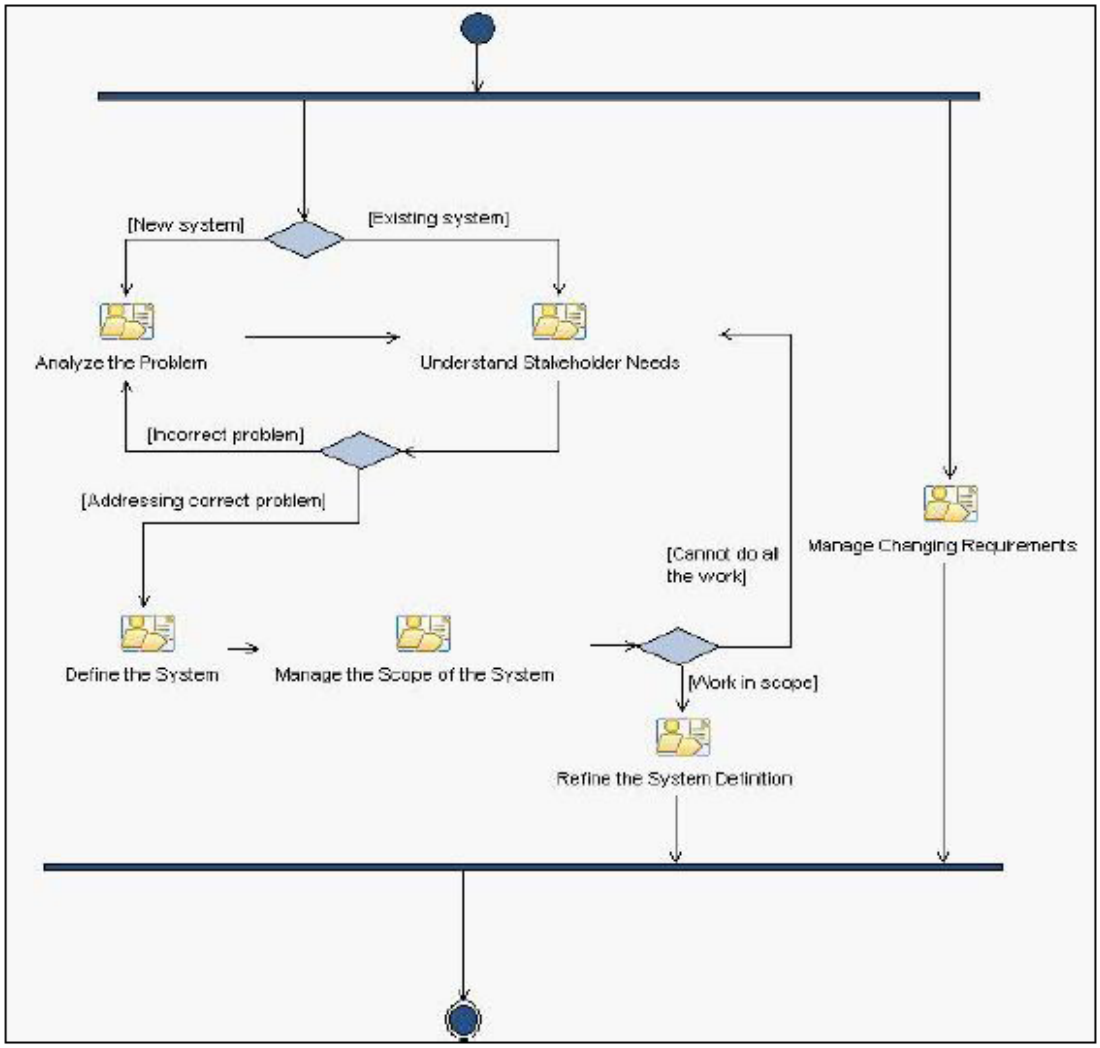
- Procenjivanje stanja poslovanja
  - Procenjivanje ciljnog stanja
  - Postavljanje i prilagođavanje ciljeva
  - Prepoznavanje poslovnih ciljeva i ključnih indikatora performansi
  - Analiza poslovne arhitekture
  - Pravljenje rečnika uobičajenih poslovnih termina
  - Uređivanje poslovnih pravila

- Opisivanje trenutnog stanja
  - Procenjivanje ciljnog stanja
  - Postavljanje i prilagođavanje ciljeva
  - Prepoznavanje poslovnih ciljeva i ključnih indikatora performansi
  - Pronalaženje poslovnih aktera i slučajeva upotrebe
  - Analiza poslovne arhitekture
  - Pravljenje rečnika uobičajenih poslovnih termina
  - Uređivanje poslovnih pravila
  - Analiza funkcionalnih celina
- Definisanje poslovanja
  - Prepoznavanje poslovnih prosecaž
  - Doterivanje definicija poslovnih procesa
  - Definisanje ostvarivanja poslovnih procesa (posl. slučajevi up.)
  - Definisanje poslovnih operacija
  - Doterivanje uloga i odgovornosti
- Istraživanje automatizacije procesa
  - Postavljanje i prilagođavanje ciljeva
  - Definisanje zahteva za automatizaciju
  - Izvođenje "dokaza" validnosti poslovne arhitekture
- Razvoj modela domena
  - Pravljenje rečnika uobičajenih poslovnih termina
  - Uređivanje poslovnih pravila
  - Analiza poslovne arhitekture
  - Detaljno opisivanje poslovnih entiteta
  - Proveravanje modela dobijenog poslovnom analizom

7.4.2 Zahtevi

Disciplina zahtevi posvećena je: prikupljanju zahteva ulagača, i njihovom transformisanju u skup rezultata rada zahteva. Ti rezultati određuju: okvire sistema koji se gradi, i detaljne zahteve u odnosu na ono što sistem mora da radi.

Svrha je: obezbeđivanje i održavanje dogovora sa ulagačima u vezi toga šta bi sistem trebalo da radi, pružanje boljeg razumevanja zahteva razvijaocima, definisanje granica sistema, pružanje osnove za planiranje tehničkog sadržaja iteracija, pružanje osnove za procenjivanje vremena i troškova razvoja sistema, i definisanje korisničkog interfejsa sistema, sa akcentom na potrebe i ciljeve korisnika.



Slika 7.10: Tok poslova.

Na slici 7.10 prikazano je kako poslovi teku u disciplini "Zahtevi". Poslovi su grupisani u veće poslove na sledeći način:

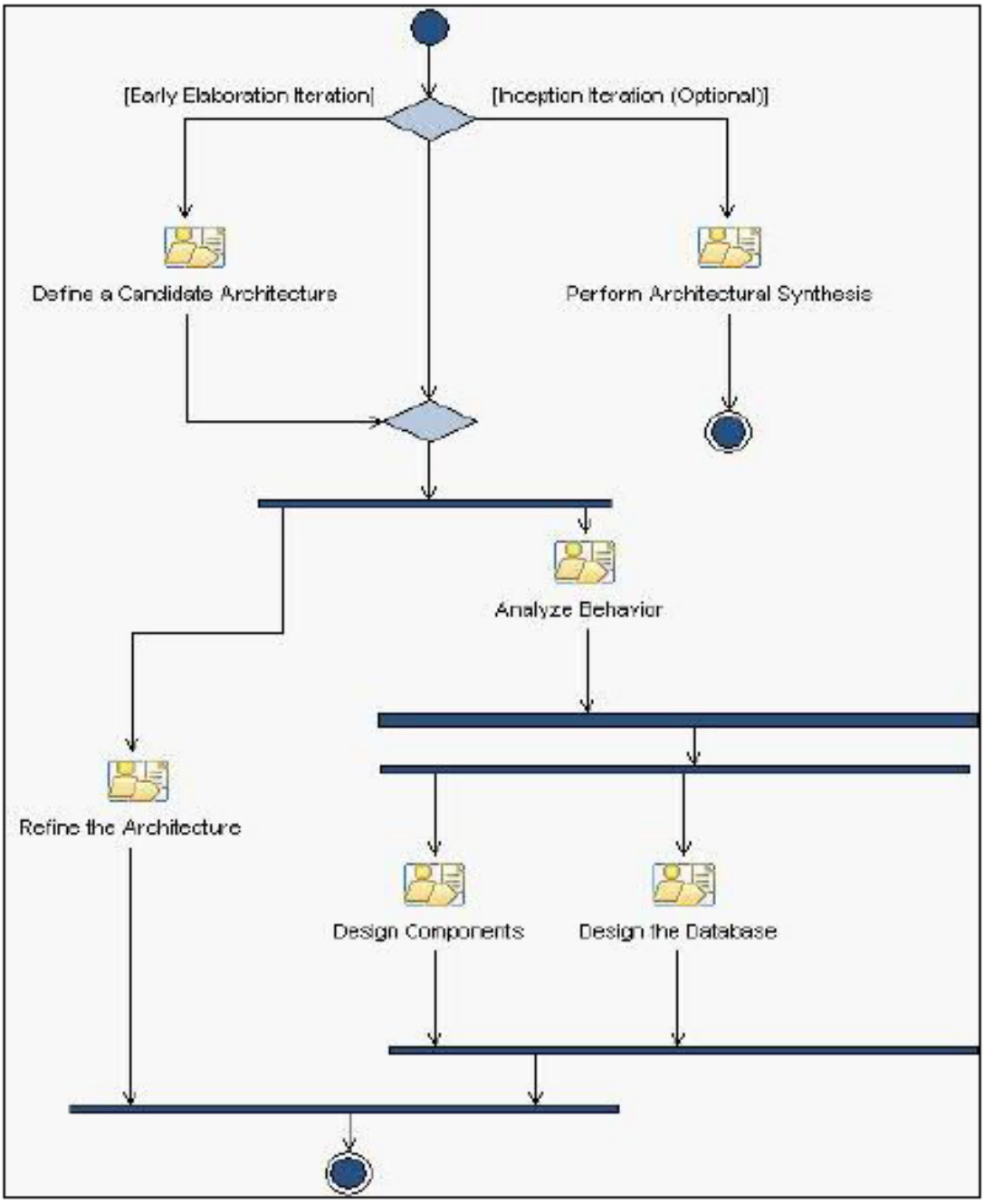
- Analiza problema
  - Prikupljanje zajedničkog rečnika
  - Pronalaženje aktera i slučajeva upotrebe
  - Razvijanje vizije sistema

- Razumevanje potreba ulagača
  - Prikupljanje zajedničkog rečnika
  - Prikupljanje zahteva ulagača
  - Razvijanje vizije sistema
  - Pronalaženje aktera i slučajeva upotrebe
  - Razvijanje pratećih specifikacija
- Definisanje sistema
  - Razvijanje vizije sistema
  - Prikupljanje zajedničkog rečnika
  - Pronalaženje aktera i slučajeva upotrebe
  - Razvijanje pratećih specifikacija
- Upravljanje okvirima sistema
  - Razvijanje vizije sistema
  - Određivanje prioriteta slučajeva upotrebe
- Unapređivanje definicije sistema
  - Detaljno dokumentovanje slučajeva upotrebe
  - Razvijanje pratećih specifikacija
  - Detaljno dokumentovanje softverskih zahteva
- Upravljanje zahtevima za izmene
  - Strukturiranje modela slučajeva upotrebe
  - Revidiranje zahteva

7.4.3 Analiza i projektovanje

Disciplina "Analiza i projektovanje" posvećena je: transformisanju rezultata rada u disciplini "Zahtevi" u rezultate rada koji određuju projekat softvera koji će se razvijati.

Svrha je: transformisanje zahteva u projekat budućeg sistema, postepeno razvijanje robusne arhitekture sistema, i prilagođavanje projekta implementacionom okruženju i postizanju punih performansi.



Slika 7.11: Tok poslova.

Na slici 7.11 prikazano je kako poslovi teku u disciplini "Analiza i projektovanje". Poslovi su grupisani u veće poslove na sledeći način:

- Sinteza arhitekture
  - Arhitekturna analiza
  - Izvođenje "dokaza" validnosti arhitekture

    - Procenjivanje ispravnosti arhitekture

- Definisanje arhitekture-kandidata
    - Arhitekturna analiza
    - Analiza slučajeva upotrebe

- Unapređivanje arhitekture
    - Identifikovanje projektnih mehanizama
    - Identifikovanje elemenata projekta
    - Ugrađivanje postojećih elemenata projekta
    - Strukturiranje implementacionog modela
    - Opisivanje arhitekture procesa izvršnog sistema
    - Opisivanje distribucije
    - Revidiranje arhitekture

- Analiziranje ponašanja
    - Identifikovanje elemenata projekta
    - Analiza slučajeva upotrebe
    - Projektovanje korisničkog interfejsa
    - Pravljenje prototipa korisničkog interfejsa
    - Revidiranje projekta

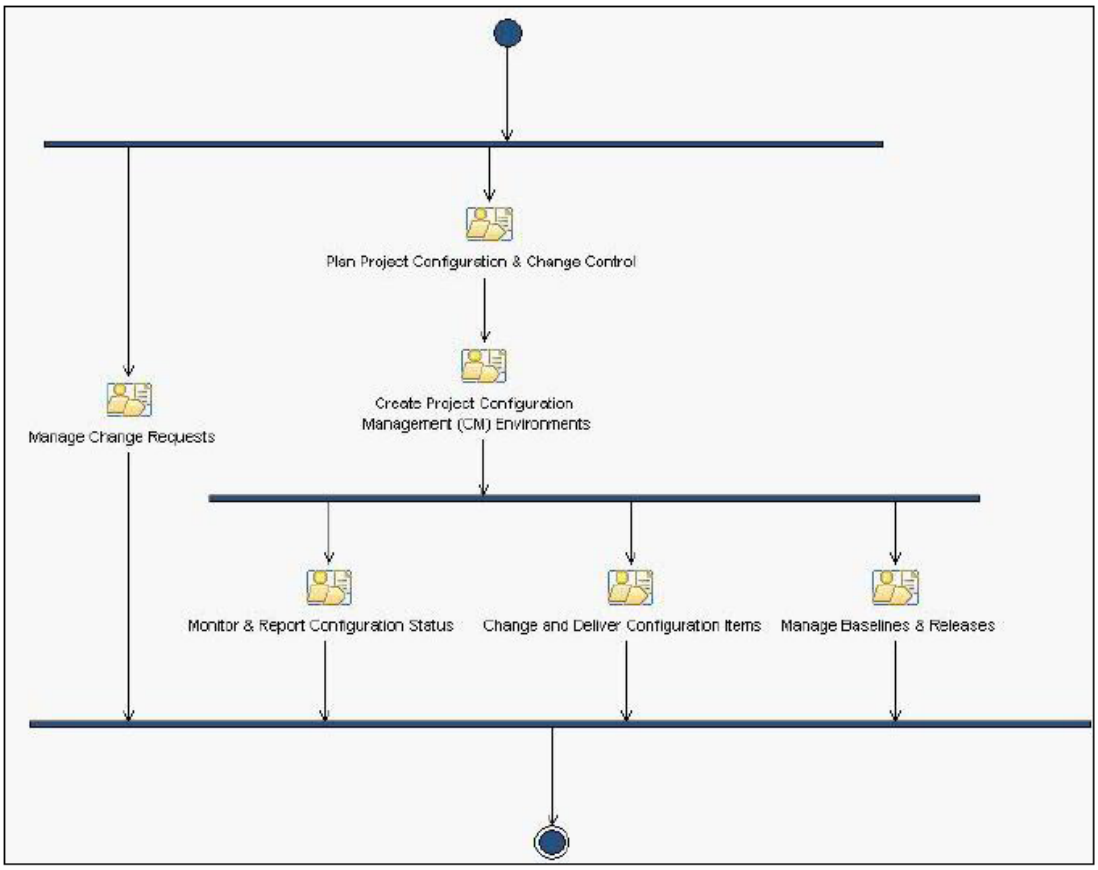
- Projektovanje komponenti
    - Projektovanje slučajeva upotrebe
    - Projektovanje podsistema
    - Projektovanje klasa
    - Revidiranje projekta

- Projektovanje baze podataka
    - Projektovanje klasa
    - Projektovanje baze podataka
    - Revidiranje projekta

7.4.4 Konfigurisanje i upravljanje izmenama

Disciplina "Konfigurisanje i upravljanje izmenama" posvećena je: kontrolisanju i sinhronizovanju rezultata rada nastalih tokom iteracija, i komponovanju pojedinačnih rezultata rada u softverski sistem.

Svrha je: podrška razvojnom metodu, održavanje integriteta proizvoda, obezbeđivanje kompletnosti i korektnosti konfigurisanog proizvoda, pružanje stabilnog okruženja za razvijanje proizvoda, ograničavanje izmena rezultata rada u skladu sa politikom projekta, i omogućava praćenje razloga, vremena i aktera izmena na rezultatima rada.



Slika 7.12: Tok poslova.

Na slici 7.12 prikazano je kako poslovi teku u disciplini "Konfigurisanje i upravljanje izmenama". Poslovi su grupisani u veće poslove na sledeći način:

- Upravljanje zahtevima za izmene
  - Podnošenje zahteva za izmene
  - Revidiranje zahteva za izmene

  - Potvrđivanje duplikata ili odbijenih zahteva za izmene
  - Raspoređivanje i dodeljivanje poslova
- Planiranje konfigurisanja projekta i kontrola izmena
- Pravljenje okruženja za upravljanje konfigurisanjem sistema
  - Podešavanje okruženja za upravljanje konfigurisanjem sistema
- Menjanje i postavljanje konfiguracije
- Upravljanje ključnim tačkama projekta i izdanjima
- Praćenje i izveštavanje o stanju konfiguracije

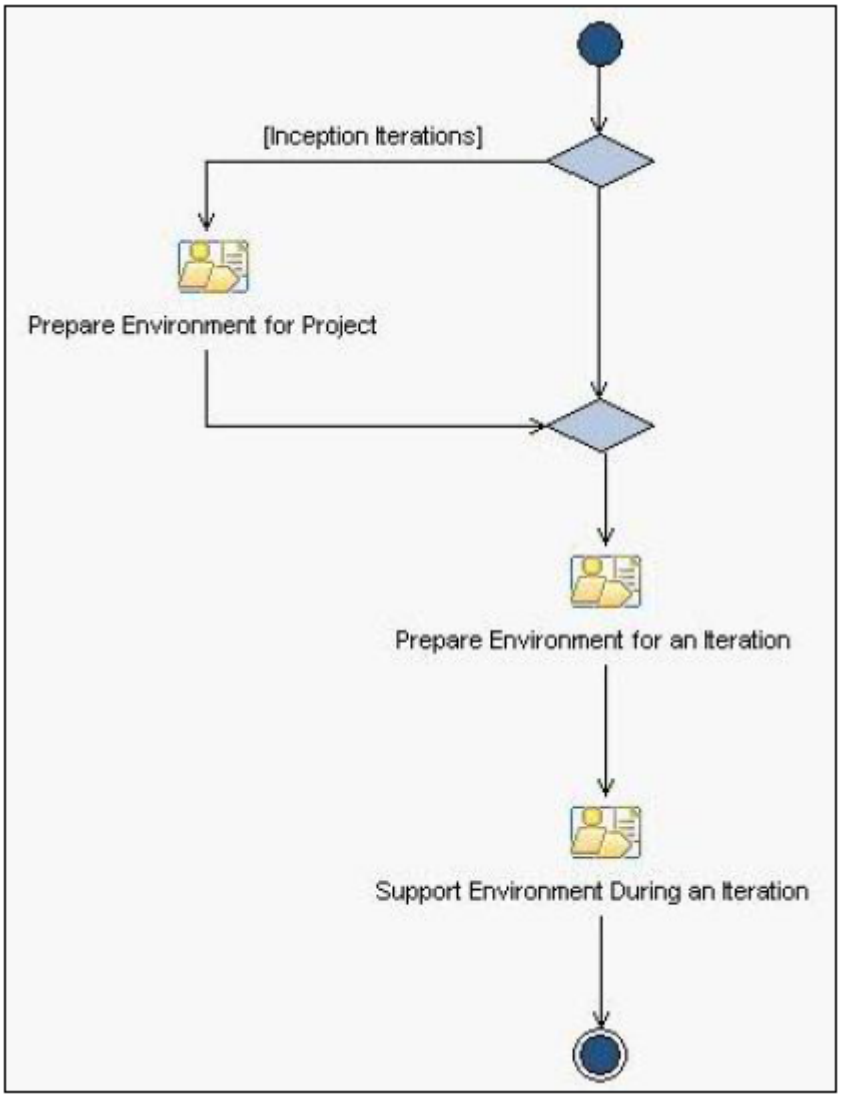
7.4.5 Okruženje

Disciplina "Okruženje" posvećena je: organizaciji elemenata koji čine okruženje za razvoj softvera, tehničkoj i komunikacionoj podršci razvojnom timu, i uključuje i procese i alate.

Svrha je: obezbeđivanje da organizacija za razvoj softvera (timovi i pojedinci) imaju odgovarajuće okruženje za rad, koje podržava njihov rad na razvoju softvera.

Na slici 7.13 prikazano je kako poslovi teku u disciplini "Okruženje". Poslovi su grupisani u veće poslove na sledeći način:

- Priprema okruženja za projekat
  - Povezivanje razvojnog procesa sa projektom
- Priprema okruženja za iteraciju
  - Pokretanje razvojnog procesa
- Podržavanje okruženja tokom iteracije
  - Podrška razvoju

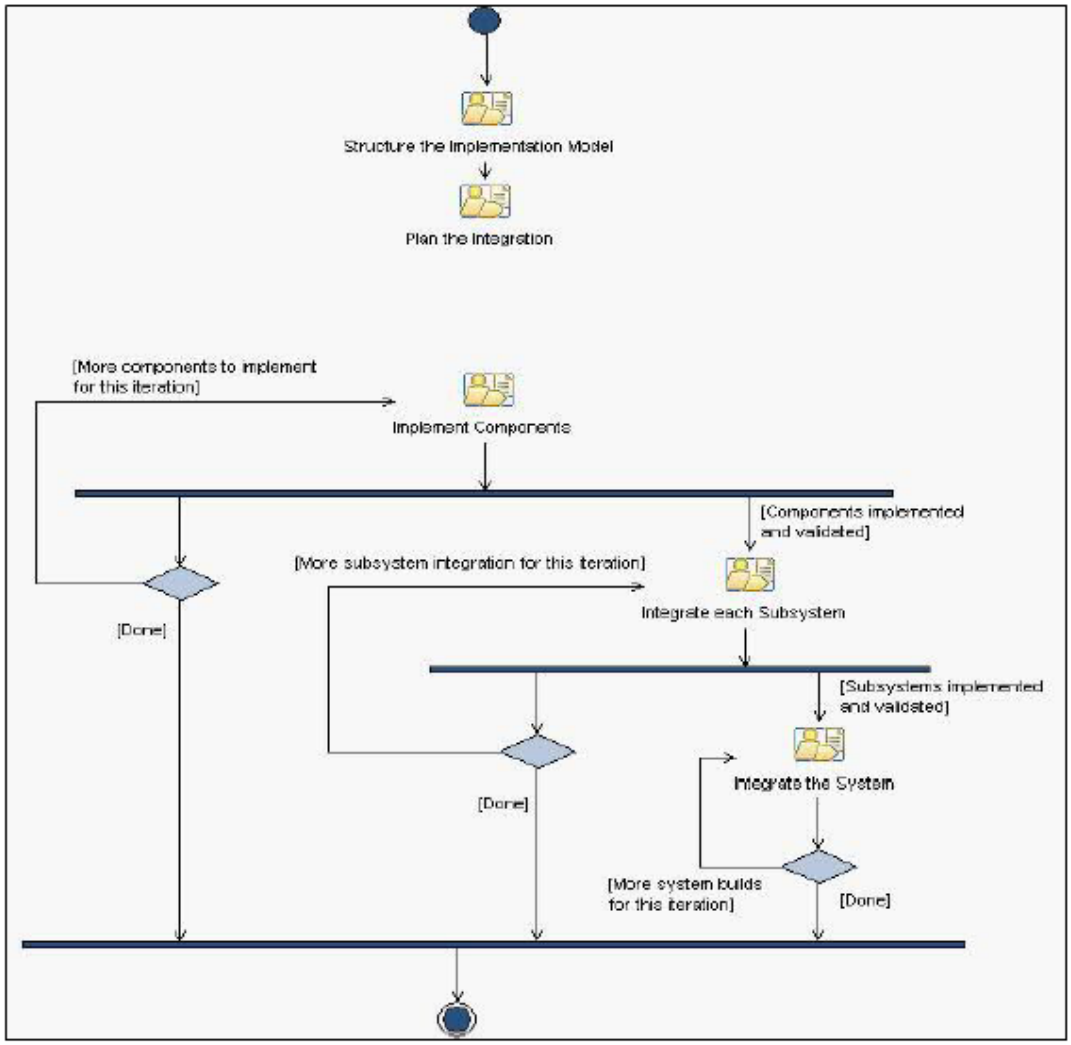


Slika 7.13: Tok poslova.

7.4.6 Implementacija

Disciplina "Implementacije" posvećena je: razvoju, organizaciji, testiranju i integrisanju komponenti na osnovu projektne dokumentacije.

Svrha je: određivanje organizacije koda kroz implementaciju podsistema organizovanih po slojevima, implementiranje elemenata projekta u vidu elemenata implementacije (izvorni kod, izvršni programi i drugo), testiranje razvijenih komponenti kao jedinica, i integrisanje pojedinačnih rezultata (pojedinaca ili timova) u izvršni sistem.



Slika 7.14: Tok poslova.

Na slici 7.14 prikazano je kako poslovi teku u disciplini "Implementacija". Poslovi su grupisani u veće poslove na sledeći način:

- Strukturiranje implementacionog modela
- Planiranje integracije sistema
- Implementacija komponenti
  - Planiranje integracije podsistema
  - Implementacija elemenata projekta
  - Analiza ponašanja pri izvršavanju
  - Implementacija razvojnih testova
  - Izvođenje razvojnih testova

  - Revidiranje koda

- Integrisanje svakog podsistema
  - Implementacija razvojnih testova
  - Izvođenje razvojnih testova
  - Integrisanje podsistema

- Integrisanje sistema

7.4.7 Upravljanje projektom

Disciplina "Upravljanje projektom" posvećena je: svim različitim aktivnostima koje se tiču upravljanja projektom.

Svrha je: obezbeđivanje okruženja za upravljanje projektima razvoja softvera, pružanje praktičnih vodilja za planiranje, rad sa kadrovima, vođenje i praćenje projekata, i pružanje okvira za upravljanje rizicima.

Važno je napomenuti da se ova disciplina ne bavi: kadrovskim poslovima (na primer, zapošljavanje, obuka, itd.), budžetskim poslovima (na primer, definisanje, obezbeđivanje, itd.), i upravljanjem ugovorima sa snabdevačima, klijentima, itd. Na slici 7.15 prikazano je kako poslovi teku u disciplini "Upravljanje projektom". Poslovi su grupisani u veće poslove na sledeći način:

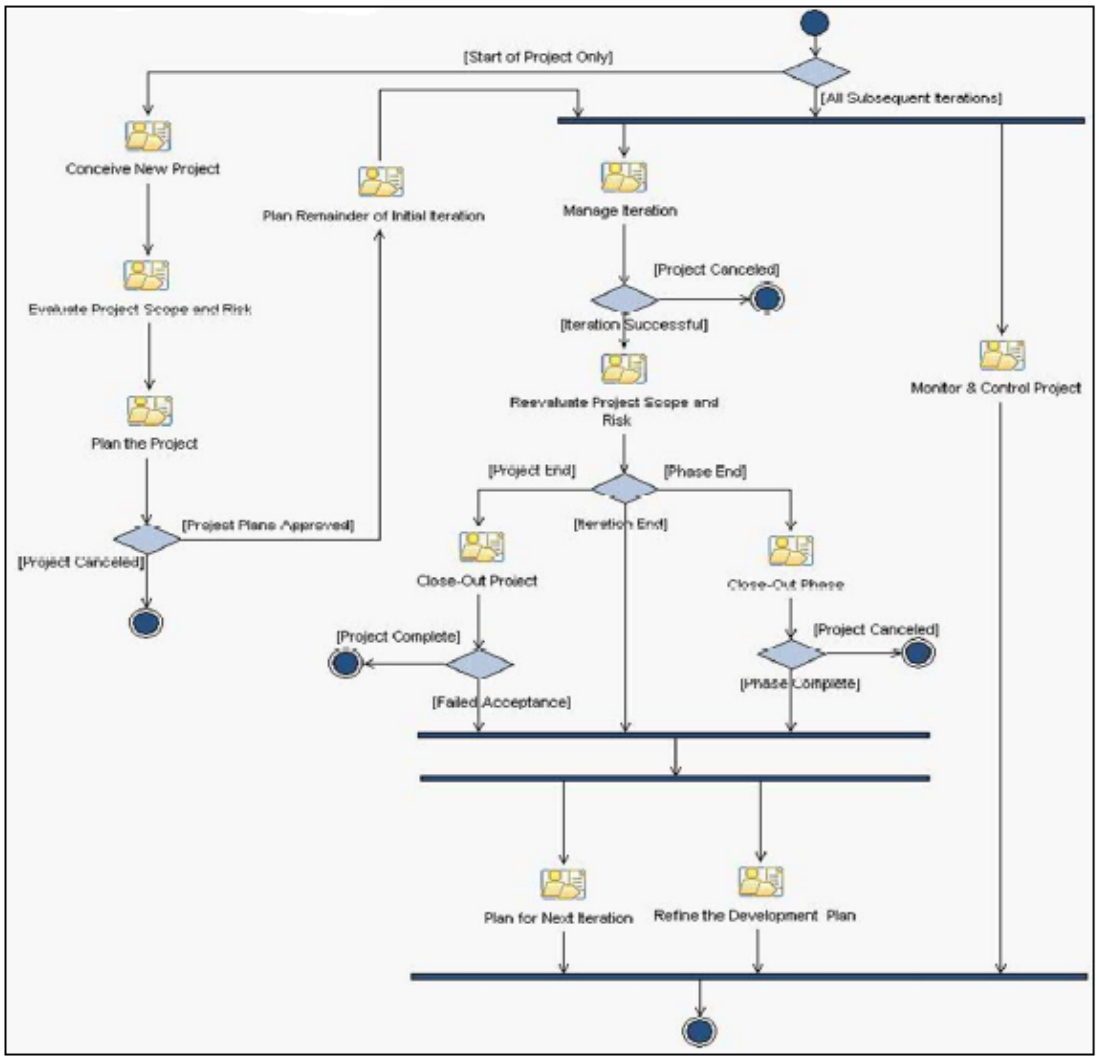
- Oformljivanje novog projekta
  - Prepoznavanje i proceljivanje rizika
  - Razvoj poslovnog slučaja
  - Započinjanje projekta
  - Revizija pri odobravanju projekta

- Procenjivanje okvira i rizika
  - Prepoznavanje i procenjivanje rizika
  - Razvoj poslovnog slučaja

- Planiranje projekta
  - Definisanje organizacije i osoblja
  - Planiranje faza i iteracija
  - Revizija plana projekta

- Planiranje ostatka inicijalne iteracije

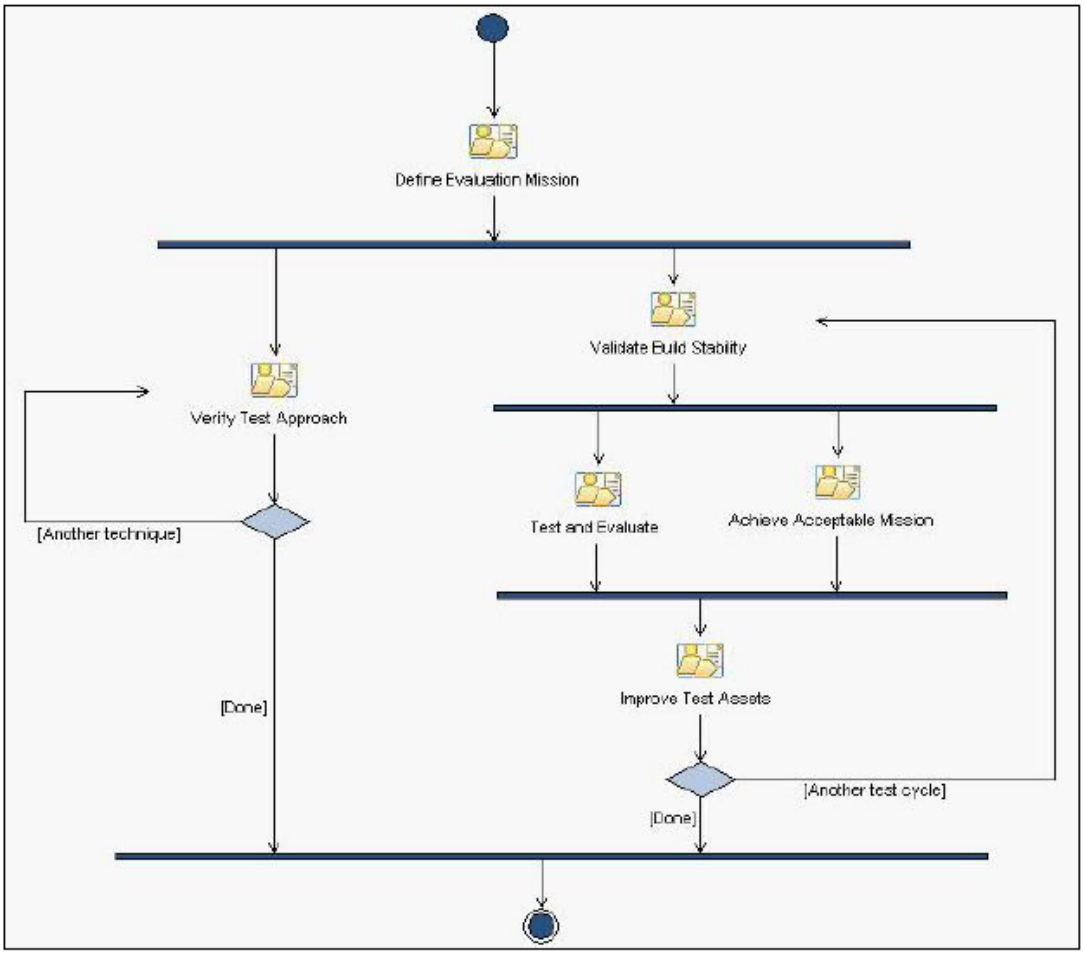
  - Razvoj plana iteracije
  - Razvoj poslovnog slučaja
  - Revizija plana iteracije
- Praćenje i kontrola projekta
  - Raspoređivanje i dodeljivanje poslova
  - Izveštavanje o stanju
  - Organizacija revizije
  - Vođenje revizije
- Upravljanje iteracijama
  - Obezbeđivanje osoblja
  - Započinjanje iteracije
  - Revizija kriterijuma evaluacije iteracije
  - Prepoznavanje i procenjivanje rizika
  - Procenjivanje vrednosti iteracije
- Ponovljeno procenjivanje okvira i rizika
  - Prepoznavanje i procenjivanje rizika
  - Razvoj poslovnog slučaja
- Planiranje naredne iteracije
  - Razvoj plana iteracije
  - Razvoj poslovnog slučaja
  - Revizija plana iteracije
- Unapređivanje plana razvoja
  - Definisanje organizacije i osoblja
  - Planiranje faza i iteracija
  - Revizija plana projekta
- Završna faza
- Zatvaranje projekta



Slika 7.15: Tok poslova.

7.4.8 Testiranje

Disciplina "Testiranje" posvećena je: procenjivanju i proveravanju kvaliteta proizvoda. Najčešće se ostvaruje kroz: traženje i dokumentovanje propusta u kvalitetu softvera, određivanje preporuka na osnovu prepoznatog kvaliteta softvera, proveravanje i dokazivanje pretpostavki koje potiču iz projekta i specifikacije zahteva, proveravanje da li softverski proizvod radi kako je projektovano, i proveravanje da su zahtevi implementirani na odgovarajući način.



Slika 7.16: Tok poslova.

Na slici 7.16 prikazano je kako poslovi teku u disciplini "Testiranje". Poslovi su grupisani u veće poslove na sledeći način:

- Određivanje cilja procenjivanja
  - Identifikovanje ideje testiranja
  - Određivanja pristupa testiranju
- Provera pristupa testiranju
  - Implementiranje skupova testova
- Validacija stabilnosti izgradnje
  - Izvođenje skupova testova
  - Određivanje rezultata testiranja
  - Procenjivanje i argumentovanje kvaliteta

- Testiranje i procenjivanje
  - Implementiranje skupova testova
  - Izvođenje skupova testova
  - Strukturiranje implementacija testova
  - Identifikovanje ideje testiranja
  - Određivanje rezultata testiranja
- Misija dostizanja prihvatljivog
  - Procenjivanje i argumentovanje kvaliteta
  - Određivanje rezultata testiranja
- Unapređivanje procena testova
  - Određivanja pristupa testiranju
  - Strukturiranje implementacija testova
  - Identifikovanje ideje testiranja
  - Implementiranje skupova testova

7.5 Uloge

Skupovi uloga se mogu podeliti na sledeće:

- Uopšteno
  - Ulagač, Koordinator pregledanja, Pregledač, Tehnički pregledač, Bilo ko
- Rukovodioci
  - Rukovodilac projekta
  - Rukovodilac kontrole izmena
  - Rukovodilac testiranja
  - Rukovodilac konfigurasanja
  - Pregledač iz rukovodstva
  - Administrator sistema
  - Rukovodilac isporučivanja
- Analitičari

  - Ulagač
  - Određivač zahteva
  - Analitičar sistema

  - Analitičar poslovnih procesa
  - Poslovni arhitekta
  - Poslovni projektant
- Razvijaoci
  - Softverski arhitekta
  - Projektant
  - Projektant baze podataka
  - Projektant korisničkog interfejsa
  - Implementator
  - Integrator

  - Sigurnosni arhitekta
  - Projektant kapsula
- Testeri
  - Upravljač testova
  - Analitičar testova
  - Projektant testova
  - Tester
- Primena i podrška
  - Inženjer procesa
  - Administrator sistema

  - Razvijalac obuke
  - Grafički urednik
  - Autor tehničke dokumentacije
  - Specijalisti za alate

7.6 Rezultat rada

Rezultat rada je sve ono što nastaje kao rezultat jednog ili više poslova tokom rada na analizi i razvoju sistema. Klasifikuju se prema disciplinama i prema vrsti.

Vrste rezultata rada su:

- Procene
  - Poslovni slučaj
  - Procena iteracije
  - Procena stanja
  - Zaključak testiranja
  - Rezultati analize konfiguracije
  - Procena organizacije razvoja
  - Procena ciljne organizacije
- Koncepti
  - Referentna arhitektura
  - "Dokaz" validnosti arhitekture
  - Prototip korisničkog interfejsa
  - Strategija testiranja
  - Spisak predloženih testova
  - "Dokaz" validnosti poslovne arhitekture
- Infrastruktura
  - Razvojna infrastruktura
  - Repozitorijum projekta
  - Radni prostor
  - Alati
  - Konfiguracija okruženja za testiranje
- Modeli
  - Model slučajeva upotrebe
  - Analitički model
  - Model isporučivanja

  - Projektni model
  - Model podataka
  - Implementacioni model
  - Karta navigacije (kroz kor. int.)

  - Modeli poslovanja
    - analitički
    - model isporučivanja
    - projektni
    - slučajeva upotrebe
  - Model usluga
  - Model analize opterećenja

- Elementi modela
  - Akter
  - Slučaj upotrebe
  - Razrada slučaja upotrebe
  - Paket slučajeva upotrebe
  - Događaj
  - Interfejs
  - Signal
  - Klasa (analitičkog modela)
  - Klasa (projektnog modela)
  - Paket (projektnog modela)
  - Podsistem (projektnog modela)
  - Podsistem (implementacionog modela)
  - Poslovni sistem
  - Radno mesto
  - Poslovni domen
  - Poslovni cilj
  - Poslovna operacija
  - Ostvarivanje poslovne operacije
  - Poslovni događaj
  - Poslovni akter

  - Poslovni slučaj upotrebe
  - Ostvarivanje poslovnog slučaja upotrebe
  - Poslovno pravilo
  - Poslovni entitet
  - Kapsula
  - Poruka
  - Protokol
  - Operacija
  - Ostvarivanje operacije
  - Usluga
  - Kanal usluga
  - Komponenta usluge
  - Ugovor usluge
  - Prolaz do usluge (gateway)
  - Particija usluge
  - Pružalac usluge
  - Specifikacija usluge
  - Projekat testa
  - Klasa za testiranje

- Planovi
  - Plan razvoja softvera
  - Plan iteracije
  - Plan integracije

  - Plan upravljanja zahtevima
  - Plan obezbeđivanja kvaliteta
  - Plan ustanovljavanja problema
  - Plan upravljanja konfiguracijom
  - Plan upravljanja rizicima
  - Plan merenja
  - Plan testiranja
  - Plan isporučivanja
  - Plan usvajanja proizvoda

- Procesi
  - Proces razvoja
  - Slučaj upotrebe razvoja
  - Stil pisanja dokumentacije
  - Smernice specifične za projekat
  - Šabloni specifični za projekat
  - Podaci projekta
  - Redosled rada
  - Spisak rizika
  - Zahtev za izmene
  - Zapis o pregledanju
  - Rezultat testiranja
  - Dnevnik testiranja
  - Svojstva zahteva
  - Spisak tačaka (problema, nedovršenosti,...)
  - Rezultati merenja projekta
- Rešenja
  - Element implementacije (izvorni kod,...)
  - Razvojni test
  - Paket testova
  - Izgrađen projekat ("izvršna verzija")
  - Proizvod
  - Beleške o verziji
  - Jedinica isporučivanja
  - Element instalacije
  - Umetnički elementi proizvoda
  - Spisak upotrebljenog materijala
  - Test podaci, skriptovi, komponente
  - Materijali za obuku i podršku korisnika

- Specifikacije
  - Vizija
  - Rečnik
  - Scenario upotrebe sistema
  - Zahtevi ulagača
  - Softverski zahtevi
  - Specifikacija softverskih zahteva
  - Dodatne specifikacije
  - Dokumentovana arhitektura softvera

  - Poslovna vizija
  - Poslovni rečnik
  - Poslovni cilj
  - Poslovno pravilo
  - Dodatne poslovne specifikacije
  - Dokumentovana poslovna arhitektura
  - Specifikacija miracije podataka
  - Arhitektura automatizacije testova
  - Slučaj testiranja
  - Specifikacija interfejsa za testiranje

7.7 Pitanja

1. Šta je RUP?
2. Koji postulati ilustruju filozofiju RUP-a? Navesti bar Objasniti?
3. Šta podrazumeva postulat prilagođavanja procesa RUP-a?
4. Šta podrazumeva postulat usklađivanja sukobljenih interesa RUP-a?
5. Šta podrazumeva postulat saradnje celokupnog razvojnog tima RUP-a?
6. Šta podrazumeva postulat iterativnog predstavljanja rezultata RUP-a?
7. Šta podrazumeva postulat podizanja nivoa apstrakcije RUP-a?
8. Šta podrazumeva postulat stalnog posvećivanja kvalitetu RUP-a?
9. Koji su osnovni principi RUP-a? Navesti bar

10. Šta je vizija (RUP)?
11. Kako izgleda životni ciklus IS po metodologiji RUP?
12. Navesti i kratko objasniti periode razvoja po RUP-u?
13. Kakav je međusoban odnos perioda razvoja u okviru životnog ciklusa po metodologiji RUP?
14. Čime se odlikuje period životnog ciklusa usvajanje (RUP)?
15. Čime se odlikuje period životnog ciklusa razrada (RUP)?
16. Čime se odlikuje period životnog ciklusa konstrukcija (RUP)?
17. Čime se odlikuje period životnog ciklusa prelazak (RUP)?
18. Šta je razvojni ciklus (RUP)?
19. Šta je generacija (RUP)?
20. Šta je evolucioni ciklus (RUP)?
21. Navesti okvirnu procenu trajanja svakog od perioda životnog ciklusa RUP.
22. Navesti okvirnu procenu obima poslova za svaki od perioda životnog ciklusa RUP.
23. U čemu je razlika između životnog ciklusa RUP i spiralnog životnog ciklusa IS?
24. Šta je strategija planiranja životnog ciklusa? Navesti (samo nazive) bar 3 strategije planiranja životnog ciklusa (RUP)?
25. Po čemu se međusobno razlikuju stragije životnog ciklusa RUP? Objasniti na primeru dve strategije.
26. Čime se odlikuje inkrementalna strategija životnog ciklusa RUP? Kada se primenjuje?
27. Čime se odlikuje evoluciona strategija životnog ciklusa RUP? Kada se primenjuje?
28. Čime se odlikuje strategija inkrementalnog isporučivanja životnog ciklusa RUP? Kada se primenjuje?
29. Čime se odlikuje strategija veliki projekat životnog ciklusa RUP? Kada se primenjuje?

30. Čime se odlikuju hibridne strategije životnog ciklusa RUP? Kada se primenjuju?
31. Šta su discipline (RUP)? Navesti bar 6 disciplina.
32. Šta je disciplina poslovno modeliranje (RUP)? Čime se bavi? Svrha? Koje poslove obuhvata (bar 5)?
33. Šta je disciplina zahtevi (RUP)? Koja je svrha ove discipline? Koje poslove obuhvata (bar 7)?
34. Šta je disciplina analiza i projektovanje (RUP)? Čime se bavi? Svrha? Koje poslove obuhvata (bar 5)?
35. Šta je disciplina konfigurisanje i upravljanje izmenama (RUP)? Čime se bavi? Svrha? Koje poslove obuhvata (bar 5)?
36. Šta je disciplina okruženje (RUP)? Čime se bavi? Svrha? Koje poslove obuhvata (bar 5)?
37. Šta je disciplina implementacija (RUP)? Čime se bavi? Svrha? Koje poslove obuhvata (bar 5)?
38. Šta je disciplina upravljanje projektom (RUP)? Čime se bavi? Svrha? Koje poslove obuhvata (bar 5)?
39. Šta je disciplina testiranje (RUP)? Čime se bavi? Svrha? Koje poslove obuhvata (bar 5)?
40. Navesti skupove uloga subjekata u okviru metodologije RUP. Objasniti po jednom rečenicom.
41. Navesti i objasniti analitičarske uloge subjekata u metodologiji RUP.
42. Navesti i objasniti razvijačke uloge subjekata u metodologiji RUP.
43. Šta je rezultat rada (RUP)?
44. Navesti bar 7 vrsta rezultata rada (RUP).
45. Navesti i objasniti vrste modela u metodologiji RUP.
46. Navesti vrste planova u metodologiji RUP.

7.8 Literatura

Literatura za ovu temu su odabrana poglavlja iz [1] i [6].

Glava 8

Bezbednost

Opšta definicija bezbednosti glasi:

- *Bezbednost* je stepen oslobođenosti od opasnosti.

Bezbednost informacija je stepen u kome je onemogućeno ugrožavanje informacija. Pod ovime možemo posmatrati zaštitu informacija i informacionog sistema od neautorizovanog pristupa, upotrebe, objavljivanja, menjanja, uništavanja, kao i od sprečavanja raspoloživosti. Jedno komično shvatanje bezbednog sistema nam daje A. Hickey na sledeći način: "Jedini stvarno bezbedan sistem je sistem koji je isključen, stavljen u blok betona, zapečaćen u olovom obloženoj sobi sa naoružanim stražarima... pa čak i tada nisam baš uveren..."

8.1 Predmeti ugrožavanja

Osnovni predmeti ugrožavanja su:

- **Poverljivost informacija** (engl. *Confidentiality*). Ovo je koncept sličan privatnosti (ali ne i isti), odnosno, predstavlja neophodan sastavni deo privatnosti. To je sposobnost da se sopstveni podaci zaštite od neovlašćenog pristupa.

- **Integritet informacija** (engl. *Integrity*). To je sposobnost da se spreči neovlašćeno ili nepoželjno menjanje podataka.

- **Raspoloživost informacija** (engl. *Availability*). To je sposobnost da se pristupa podacima kada su potrebni.

Ova tri predmeta se nazivaju "bezbednosna trojka" – CIA, a često namerno okrenuto, kao CAI. Takodje, koristi se često obrnuta notacija, iz ugla postupaka – DAD: **odavanje** (engl. *disclosure*), **menjanje** (engl. *alteration*), i *ometanje* (engl. **denial**).

Drugi predmeti ugrožavanja su:

- **Posedovanje ili upravljanje**. To je sposobnost punog upravljanja podacima. Obuhvata posedovanje fizičkih nosilaca informacija.
- **Autentičnost**. To je sposobnost da se utvrdi originalnost informacija. Obično su to različiti vidovi el. potpisa.
- **Upotreba**. Predstavlja stepen upotrebljivosti podatka. Zavisi od oblika, količine, i sl. To je jedina nebinarna ugroženost.

8.2 Napadi

Osnovne vrste napada su:

- **Upad** (engl. *interception*). Upad je svaki čin neovlašćenog pristupanja informacijama. Upadi ugrožavaju poverljivost.
- **Onesposobljavanje** (engl. *interruption*). To je napad koji ima za posledicu neki vid pune ili delimične neraspoloživosti sistema. Ugrožava raspoloživost, a posredno i integritet.
- **Menjanje informacija** (engl. *modification*). To je napad kojim se menjaju postojeći podaci. Ugrožavanje integriteta podataka se može vršiti na dva načina: menjanje radi prikrivanja, i menjanje radi uništavanja. Ugrožava integritet, a posredno i raspoloživost.
- **Pravljenje informacija** (engl. *fabrication*). To je napad kojim se proizvode novi podaci u sistemu. Ugrožava integritet, a posredno i raspoloživost. Pravljenje velike količine informacija može da neposredno ugrozi raspoloživost.

Važni pojmovi koji se pojavljaju u terminologiji vezanoj za bezbednost su sledeći:

- **Opasnost**. To je svaki događaj koji potencijalno ugrožava bezbednost.
- **Slabost**. To su elementi sistema i njihove karakteristike koje izlažu sistem opasnostima.
- **Rizik**. To je verovatnoća da se dogodi nešto loše.
- **Uticaj**. To je predmet, obim, ili dubina mogućih posledica neke opasnosti.

8.3 Slojevi sistema i zaštita

Postoji nekoliko slojeva sistema i njihovih zaštita. To su:

- Podaci – kriptovanje, kontrola pristupa, rezervne kopije, proveravanje upada, i analiza slabosti.
- Aplikacija – sistemi za objedinjenu proveru autentičnosti, filtriranje sadržaja, provera ispravnosti podataka, praćenje i pravljenje dnevnika aktivnosti, provera upada, i analiza slabosti.
- Server (čvor) – provera autentičnosti, zaštita od virusa, zaštitni zid, sistem za prepoznavanje upada (IDS), sistem za prevenciju upada (IPS), kriptovanje lozinki, praćenje i pravljenje dnevnika aktivnosti, provera upada, i analiza slabosti.
- Interna mreža – sistem za prepoznavanje upada (IDS), sistem za prevenciju upada (IPS), praćenje i pravljenje dnevnika aktivnosti, provera upada, analiza slabosti.
- Granica mreže – zaštitni zid, proksi, nadzor paketa, praćenje i pravljenje dnevnika aktivnosti, provera upada, i analiza slabosti.
- Spoljašnja mreža – izolovanje mreže (DMZ), virtualno privatno umrežavanje (VPN), praćenje i pravljenje dnevnika aktivnosti, provera upada, i analiza slabosti.

8.4 Opasnosti i problemi

Opasnosti se dele na:

- **Unutrašnje**, koje mogu biti otvorene, zatvorene, ili nenamerne.
- **Spoljašnje**, koje mogu biti otvorene, zatvorene, ili nenamerne.
- **Nezgode**.

Podela problema prema uzroku se vrši u odnosu na podelu opasnosti:

- **Unutrašnje opasnosti**. To su opasnosti čiji su nosioci subjekti zaposleni u organizaciji.
- **Spoljašnje opasnosti**. To su opasnosti čiji su nosioci subjekti van organizacije.
- **Nezgode**. To su opasnosti koje nastaju bez eksplicitne ljudske aktivnosti. Na primer, kvarovi, elementarne nepogode, itd.

Podela problema prema načinu ispoljavanja je:

- **Otvoreni problemi**. To su problemi koji se lako uočavaju. Obično je uništavanje integriteta i raspoloživosti. Uočljivost problema predstavlja cilj aktivnosti.
- **Prikriveni problemi**. To su problemi koji se teško uočavaju. Obično je ugrožavanje poverljivosti. Cilj je obično ugrožavanje bezbednosti.
- **Nenamerni problemi**. Oni imaju različito ispoljavanje. Obično nastaju usled nepažnje unutrašnjeg korisnika. Spoljašnji korisnici retko mogu nepažnjom da ugroze bezbednost.

8.4.1 Uobičajene opasnosti

Neke od uobičajenih opasnosti predstavljaju: greške i propusti, prevare i krađe, unutrašnja sabotaža, infrastrukturni i fizički problemi, zlonamerni hakeri, industrijska špijunaža, zlonamerni kôd, špijunaža stranih država, i pretnje ličnoj privatnosti.

8.4.1.1 Greške i propusti

Greške i propusti se javljaju:

- Pri unošenju podataka – neotpornost programa na neispravne ulazne podatke, nedostatak robusnosti
- Pri obradi podataka – bagovi, neispravnost programa
- Pri instalaciji i konfigurisanju softvera – neobučenost administratora i korisnika

8.4.1.2 Prevare i krađe

Prevare i krađe mogu biti:

- Unutrašnje prevare – većina prevara spada u unutrašnje probleme
- Prevare u javnom prostoru – na Internetu, poštom, itd.
- Krađe podataka
- Krađe hardvera

8.4.1.3 Unutrašnja sabotaža

Različiti uzroci sabotaže su:

- osveta (za maltretiranje, izrabljivanje, umor...)
- "viši" ciljevi (protiv "zle" organizacije)
- dosada i zabava

Vidovi sabotaže su:

- uništavanje opreme
- postavljanje logičkih bombi
- neispravan unos podataka
- brisanje podataka
- menjanje podataka
- obaranje sistema
- uzimanje podataka za taoce

Procenjuje se da je manje slučajeva sabotaže nego krađe. Ipak, posledice sabotaže su obično skuplje.

8.4.1.4 Infrastrukturni i fizički problemi

U infrastrukturne i fizičke probleme spadaju:

- Problemi sa napajanjem (prekidi, udari, nestanak)
- Prekid komunikacije (Internet)
- Curenja vode ili nestanak vode
- Problemi sa kanalizacijom
- Prekidi u transportnim uslugama
- Požari, poplave
- Građanski nemiri, štrajkovi

8.4.1.5 Zlonamerni hakeri

Svi zlonamerni neovlašćeni upadi u računarski sistem mogu buti spoljašnji i unutrašnji. Uglavnom se ciljaju značajni društveni ili ekonomski ciljevi. Mnogi sistemi se napadaju svakodnevno.

Često im se pridaje veći značaj nego što realno zaslužuju. O drugim vrstama opasnosti oduvek postoji briga. Namera nije jasna, pa su i ciljevi i sredstva nepredvidivi. Napadi hakera čine da se ljudi osećaju nesigurno.

8.4.1.6 Industrijska špijunaža

Industrijska špijunaža je čin prikupljanja tuđih podataka, i to:

- radi ostvarivanja sopstvene koristi
- radi nanošenja štete drugome

Industrijska špijunaža je u stalnom porastu.

8.4.1.7 Zlonamerni kod

Zlonamerni kod se može javiti u različitim oblicima, a najčešći su: virusi, crvi, trojanski konji, logičke bombe i dr. Broj virusa raste eksponencijalno, ali ne i broj incidenata (1993).

8.4.1.8 Špijunaža stranih država

Pored industrijske (ekonomske), postoji i politička (vojna) špijunaža. Postoji oduvek, samo menja oblike.

8.4.1.9 Pretnje ličnoj privatnosti

Ogromne količine ličnih podataka nalaze se u različitim računarskim sistemima. Porast upotrebe računarstva u oblaku ima za rezultat porast informacija koje delimo u oblaku. Takođe, zaposleni u državnoj službi često imaju neograničen pristup ličnim podacima, što može dovesti do:

- neovlašćenog pregledanja,
- krađe, i
- prodaje informacija trećim licima.

8.4.1.10 Zastupljenost opasnosti

Što se tiče zastupljenosti opasnosti, prema *Computer Systems Security and Privacy Advisory Board, 1991 Annual Report*, pregled gubitaka je sledeći:

- 65% greške i propusti
- 13% nepošteni zaposleni
- 6% razočarani zaposleni
- 8% infrastrukturni problemi (struja, komunikacija, voda, kanalizacija, prevoz, požar, poplava, građanski nemiri, štrajkovi)
- 5% voda, nevezano za poplave i požare
- 3% spoljašnji uticaji, uključujući viruse, špijunažu, disidente i zlonamerne sadržaje raznih vrsta, kao i bivši zaposleni, nakon više od 6 nedelja odsustvovanja

8.4.2 Uzroci problema bezbednosti

Uzroci problema bezbednosti su:

- **Brzina i nekritičnost računara**. Naime, računari rade veoma brzo. Mogu da opslužuju veliki broj korisnika. Oni nisu u stanju da kritički posmatraju dobijene zadatke. Usled toga su predmet (žrtve) računarskog kriminala.
- **Nesrazmernost složenosti pravljenja i prevazilaženja zaštite**. Lakše je probiti računarsku zaštitu nego je napraviti. Za upad je dovoljno pronaći jednu slabost sistema zaštite. Bezbednosni radnik mora da pronađe (i otkloni) sve slabosti.
- **Složenost operativnih sistema**. Savremeni OS imaju mnogo zaduženja. Veoma su složeni i teško se sagledavaju svi mogući problemi. Štaviše, dodatne aplikacije dodatno ugrožavaju ukupnu bezbednost jer povečavaju složenost.
- **Komunikacioni protokoli se teško menjaju**. Imaju mnogo korisnika. Uočene slabosti se moraju krpiti u implementacijama, a ne u standardu.

Prema *MS Technet04*, 10 opštih bezbednosnih pravila su:

1. Ako vas neko ubedi da na svom računaru izvršite njegov program, to više nije vaš računar.
2. Ako neko može da menja OS na vašem računaru, to više nije vaš računar.
3. Ako neko ima neograničen fizički pristup vašem računaru, to više nije vaš računar.

4. Ako neko sme da postavlja programe na vašu veb lokaciju, to više nije vaša veb lokacija.
5. Slabe lozinke pobeđuju jaku bezbednost.
6. Računar je bezbedan samo onoliko koliko je njegov vlasnik (korisnik) pouzdan.
7. Kriptovani podaci su samo onoliko bezbedni koliko i ključ za dekriptovanje.
8. Neažurna antivirusna zaštita je samo malo bolja nego nepostojeća.
9. Potpuna anonimnost nije praktična ni u životu ni na vebu.
10. Tehnologija nije lek za sve probleme.

8.5 Identifikacija, autentikacija i autorizacija

Uvedimo naredne definicije:

- **Identifikacija** je provera identiteta.
- **Autentikacija** je provera autentičnosti identiteta.
- **Autorizacija** je provera prava specifičnog subjekta.

Identifikacija i autorizacija obično idu zajedno. Koriste se pri omogućavanju pristupa nekom sistemu. Obavljaju se u paru, radi "pouzdanog" utvrđivanja identiteta korisnika. Autorizacija je drugi nivo provere, pri čemu se pretpostavlja da je autentikacija uspešno izvršena.

8.5.1 Identifikacija

Identifikacija je pretpostavljanje identiteta subjekta. Uobičajeno se odvija na osnovu: imena osobe, korisničkog imena, ikonice korisnika, broja računa, ID kartice, otiska prsta, i DNK.

Sama za sebe predstavlja veoma slab koncept. Uglavnom nije ili ne mora da bude jedinstvena. Može se lažirati, na primer, kada se neko predstavi, kako ćemo znati da nije slagao?

Provera identiteta (ili **verifikacija identiteta**) korak je nakon identifikacije, ali pre autentikacije. Na primer, kada se od osobe traži da pokaže lični dokument kao potvrdu identiteta. Međutim, to još uvek nije pouzdana potvrda – dokument može da bude falsifikovan.

Provera može da ide i dalje, uz određenu proveru pouzdanosti. Na primer, može da se izvrši poređenje dokumenta sa bazom podataka, ali čak ni to nije dovoljno. U mnogim slučajevima potvrda identiteta potpuno izostaje. Na primer, u elektronskoj pošti (oko 90% svih poruka je spam).

Svaki metod identifikacije može da bude predmet **falsifikovanja**. Neki metodi se teže, a neki lakše falsifikuju. Krađa identiteta počiva na uspešnom falsifikovanju identiteta. Procena gubitaka za 2009. godinu u SAD je oko 54 milijardi dolara. Uglavnom uspeva zbog odsustva autentikacije.

8.5.2 Autentikacija

Autentikacija je skup metoda za ustanovljavanje istinitosti saopštenog identiteta. **Faktori autentikacije** su elementi koji učestvuju u proveri. Što je više faktora, to je provera pouzdanija. Uobičajeni faktori su ono što osoba može da zapamti ili nosi sa sobom.

Faktori autentikacije su:

- Memorijski faktori – lozinka, PIN, fraze za propuštanje, neka sekvenca postupaka, ili bilo šta drugo što osoba može da zapamti.
- Fizički faktori – ID kartica, i token.
- Biometrijski faktori – visina, težina, boja kose ili očiju, otisak prsta, uzorak irisa ili retine, karakteristične mere lica, i rukopis.
- Logički faktori – neka pravila reagovanja na različite uslove.
- Geografski faktori – lokacija korisnika

Radi povećavanja pouzdanosti, autentikacija često koristi više faktora. Na primer, pri podizanju novca sa automata fizički faktor je kartica, a memorijski faktor je PIN.

Na primer, neka se pri autentikaciji koristi DNK. To je nepraktično za svaku proveru. U zavisnosti od nivoa pristupa dodaju se manje pouzdani, ali i manje invazivni faktori.

8.5.2.1 Uzajamna autentikacija

U uobičajenom procesu autentikacije samo jedna strana proverava identitet druge. **Uzajamna autentikacija** je postupak u kome učesnici u komunikaciju uzajamno vrše autentikaciju.

Ako se ne koristi uzajamna autentikacija, postoji mogućnost tzv. napada "čovek između". Neki subjekat može da se postavi između sistema i korisnika. On dalje može da preuzme autentikaciju korisnika i izvrši operacije u ime korisnika, a može i da zapamti autentikaciju korisnika i kasnije je upotrebi.

Primer uzajamne autentikacije je HTTPS.

8.5.2.2 Lozinke

Lozinke su uobičajeni faktor autentikacije. Upotreba lozinke uz korisničko ime je najčešći način identifikacije i autentikacije. Pouzdanost počiva na "jačini" lozinke. Jačinu lozinke odlikuje dužina, i složenost azbuke.

Lozinke ipak imaju neke slabosti. Jake lozinke se teže pamte, pa stoga korisnici često biraju slabije lozinke. Često se koriste programi koji pamte lozinke, i oni predstavljaju dodatnu slabost u sistemu. Često se koristi ista lozinka na više mesta, pa upad u jedan sistem se lako širi na druge sisteme.

Broj kombinacija lozinki predstavljeno je sledećom tabelom:

Azbuka \ Dužina lozinke	2 znaka	4 znaka	6 znakova
Mala slova engl. abecede	676	456 976	308 915 776
Mala i velika slova engl. a.	2704	7 311 616	19 770 609 664
Mala i velika slova i cifre	3844	14 776 336	56 800 235 584
Svi znaci ASCII	8836	78 074 896	689 869 781 056

Tabela 8.1: Broj kombinacija lozinki.

8.5.2.3 Biometrijski faktori

Biometrijski faktori mogu se koristiti na dva načina:

- za potvrđivanje identiteta – na primer, slika u ličnoj karti, ili
- za ustanovljavanje identiteta – na primer, kriminološka upotreba otisaka prstiju.

Neki se mogu falsifikovati lakše, neki teže. Isključiva upotreba biometrijskih faktora često spušta autentikaciju na nivo verifikacije identiteta.

8.5.2.4 Hardverski tokeni

Uobičajeno na osnovu vremena (ugrađenog časovnika) i PIN-a izdaju specifičan izračunat broj koji se koristi za dodatnu autentikaciju. Jednostavnije verzije (bez časovnika) se svode na običan fizički faktor.

8.6 Razmatranja bezbednosti

8.6.1 Praćenje aktivnosti

Praćenje aktivnosti se može vrširi na sledeće način:

- **Obračunavanje** (engl. *accountability*). Svaka obavljena operacija ostavlja trag koji omogućava da se naknadno proveri koji subjekat ju je izvršio, i to:
  - radi naplate, ili
  - radi provere zloupotrebe.
- **Sagledavanje** (engl. *auditing*). Predstavlja sagledavanje značajnih informacija o aktivnosti sistema, pre svega radi provere ugrožavanja bezbednosti, ali i iz drugih razloga. Posmatraju se:
  - tok operacija
  - greške u postupcima
  - lozinke
  - promene u podacima
  - različite informacije o radu sistema

  Razlozi posmatranja su:
  - radi provere ugrožavanja bezbednosti,
  - radi sagledavanja načina upotrebe,
  - radi sagledavanja efikasnosti rada,
  - radi ostvarivanja obračunavanja,
  - radi sagledavanja poštovanja licenci, i
  - radi uočavanja bagova i kvarova.
- **Pravljenje dnevnika** (engl. *logging*). Predstavlja fizičko zapisivanje informacija o operacijama u sistemu. Tehnička pretpostavka za ostvarivanje nadzora. Pravljenje dnevnika je osnovni način prikupljanja informacija za nadzor. **Dnevnici** predstavljaju trajne zapise o aktivnostima sistema. Za dnevnike važe sledeća svojstva:
  - *Veoma su poverljivi.* Obično sadrže poverljive informacije iz samog sistema. Mogu da sadrže i lozinke, i podatke lične prirode o korisnicima. Uobičajeno im mogu pristupati isključivo administratori sistema.
  - *Mogu se konfigurisati.* Konfigurisanje se vrši u smislu: vrsta događaja koji se beleže, preciznosti beleženja, i učestalosti beleženja stanja sistema.

8.6.2 Nadgledanje

Nadgledanje (engl. *Monitoring*) predstavlja budno praćenje sistema. Može se posmatrati kao vid sagledavanja koji se odvija "u realnom vremenu". Često uključuje i mehanizme za automatizaciju reagovanja na neke vanredne slučajeve.

8.6.3 Procenjivanje

Na osnovu prikupljenih informacija se mogu vršiti različita **procenjivanja** (engl. *Assesment*) o saglasnosti ponašanja sistema i njegovih korisnika sa nekim skupom pravila. Postoje:

- procenjivanje zakonitosti,
- procenjivanje efikasnosti upotrebe,
- procenjivanje osetljivosti ili ugroženosti sistema, i
- prepoznavanje napada.

8.6.4 Kriptografija

Kriptografija (engl. *cryptography*) jeste tehnika predstavljanja informacija na izmenjeni način (tzv. **kriptovanje** (engl. *encrypting*)), tako da ih može ispravno tumačiti samo subjekat koji poznaje postupak vraćanja informacija u originalni oblik (tzv. **dekriptovanje** (engl. *decrypting*)).

Kriptografija je veoma važno sredstvo dodatnog obezbeđivanja informacija. Omogućava zaštitu podataka u određenoj meri čak i u slučaju da informacije postanu fizički raspoložive neovlašćenom licu. Stepen zaštite zavisi od snage kriptografkse tehnike.

Savremene kriptografske tehnike počivaju na Kirhofovom principu koji glasi:

- Pri pravljenju kriptografskog metoda potrebno je pretpostaviti da će protivnik da upozna metod i stoga sva snaga metoda mora da počiva u ključu.

8.6.5 Fizičke opasnosti

Fizičke opasnosti su:

- Fizičke infrastrukturne opasnosti – rešenje je fizička zaštita podataka pravljenjem rezervnih kopija.
- Magnetno polje i elektricitet – neoprezno rukovanje magnetnim poljem ili elektricitetom u blizini osetljivih uređaja može da bude uzrok ozbiljnih kvarova.

- Krađa uređaja – krađom uređaja se dolazi u fizički dodir sa informacijama na uređaju.
- Provlačenje – napadač stoji ili se kreće pored vrata, kao da je sve normalno; kada neko autorizovan otključa vrata i prođe kroz njih, on se provuče odmah iza njega. Rešenje je upotreba dvostrukih vrata sa međuprostorom za samo jednu osobu.
- Praćenje korisnika – ako korisnik radi neki drugi posao paralelno sa upotrebom računara, često se dešava da računar sa otvorenim nalogom ostane bez fizičkog nadzora.
- Fizička krađa podataka – krađa medija sa podacima, obično sa rezervnim kopijama.
- Kopiranje podataka – informacije se mogu kopirati u papirnom ili digitalnom obliku. Rešenje je strogo rukovanje fizičkim nosiocima informacija. Ne smeju se ostavljati bez nadzora i moraju se uništiti nakon upotrebe.
- Špijuniranje – može biti: vizualno, audio, elektronsko, i fizičko.

8.7 Slabosti i prevazilaženja

Prevencija i odbrana se vrši kroz dva osnovna koraka:

- *Razumevanje slabosti.* Virusi, kao i hakeri, mogu da iskoriste neke slabosti. Potrebno je sagledati i razumeti slabosti da bi se mogla pripremiti odbrana.
- *Uvođenje odbrambenih mehanizama.*

Uobičajene slabosti su:

- *Apatija korisnika.* Iako su svesni mogućnosti napada, korisnici uobičajeno imaju stav "to se neće dogoditi meni". Ležerno se deli softver (programi i podaci).
- *Nedovoljna bezbednosna kontrola.* Mnogi računari, posebno lični, nemaju zaštitu protiv virusa i drugih bezbednosnih opasnosti.
- *Loša upotreba postojećih bezbednosnih alata.* Antivirusni programi moraju biti ažurni. Ležerno dodeljivanje viših privilegija nego što je neophodno. Omogućavanje slabih lozinki.
- *Slabosti operativnog sistema.* Širina i veličina operativnih sistema i drugog sistemskog softvera zahteva veliki broj programera. Dovoljno je da jedan tim nije dorastao poslu i ostaće neki sigurnosni propust.

- *Neovlašćena upotreba.* Postoje ljudi kojima je izazov upadanje u tuđe sisteme. Kada jednom upadne u sistem, izazov da načini štetu je veliki.
- *Mrežna anonimnost.* Zlonamerna osoba na mreži je anonimna, što otežava otkrivanje i olakšava zlonamerno postupanje.

Osnovna sredstva za prevazilaženje slabosti su:

- *Treninzi.* Bezbednosni službenici moraju da budu u toku sa aktuelnim problemima i rešenjima.
- *Bezbednosna analiza pri nabavci softvera.* Pre stavljanja u rad novog softvera potrebna je odgovarajuća bezbednosna analiza radi sagledavanja i prevazilaženja eventualnih bezbednosnih rizika.
- *Nadgledanje.* Nadgledanje može biti: nadgledanje korisnika, nadgledanje sistema, i nadgledanje mreže.
- *Plan za slučaj opasnosti.* Pri prepoznavanju vanrednog slučaja se mora reagovati brzo. Svaki sekund utrošen na razmišljanje potencijalno uvećava štetu. Moraju postojati definisane precizne procedure postupanja u vanrednim slučajevima.
- *Ograničavanje deljenja.* Svi deljeni resursi (podaci, programi, uređaji, mreža) moraju biti strogo upravljani. Zabrana spoljašnjih uređaja za prenos podataka. Periodično čišćenje i sklanjanje podataka koji su bili deljeni (pretpostavka je da uvek nešto mora da se deli, da bi posao mogao da se obavlja).
- *Samoizolacija u slučaju napada.* U slučaju prepoznavanja da je napad u toku, obično je najočiglednija mera potpuni prekid komunikacije interne mreže sa spoljnom mrežom. To je izvesno temeljno suprotstavljanje napadu. Međutim, ako je poslovanje tesno zavisno od povezanosti sa spoljašnjom mrežom, to nije dobro rešenje. Potrebno je da plan za slučaj opasnosti obuhvati proceduru kojom se isključuje što je moguće više komunikacije, a da se održi neophodan nivo funkcionalnosti sistema.
- *Sagledavanje stvari* (engl. *audit*). Sagledavanje omogućava uočavanje neuobičajenih aktivnosti i podizanja nivoa pažnje.
- *Rezervne kopije.* Neophodno je imati definisanu preciznu proceduru pravljenja rezervnih kopija i izvođenja postupka oporavka podataka (i programa).
- *Temeljna zaštita od zlonamernog softvera.* Ovo se postiže pomoću: antivirus programa, antišpijunskih programa, i zaštitnog zida.

Principi upravljanja bezbednošću su:

1. Službenik bi trebalo da zna samo ono što je neposredno potrebno za dobro obavljanje posla za koji je zadužen. Informacije koje nema, zaposleni ne može odati, ni namerno ni nenamerno.

2. Potrebno je periodično rotirati službenike i radna mesta. Službenik kome je dosadno je rizičan službenik.

3. Svaki osetljiv posao mora da obavi jedan službenik i zatim da proveri drugi službenik. Tako se usporava posao, ali se podiže bezbednost.

8.8 Pitanja

1. Šta je bezbednost informacija?

2. Navesti i ukratko objasniti najvažnije predmete ugrožavanja bezbednosti informacija.

3. Šta je poverljivost informacija i kako se može ugroziti?

4. Šta je integritet informacija i kako se može ugroziti?

5. Šta je raspoloživost informacija i kako se može ugroziti?

6. Navesti i ukratko objasniti najvažnije vrste napada na informacione sisteme.

7. Opisati ukratko vrstu napada "upad"? Šta je predmet ugrožavanja?

8. Opisati ukratko vrstu napada "onesposobljavanje"? Šta je predmet ugrožavanja?

9. Opisati ukratko vrstu napada "menjanje informacija"? Šta je predmet ugrožavanja?

10. Opisati ukratko vrstu napada "pravljenje informacija"? Šta je predmet ugrožavanja?

11. Koji slojevi sistema se prepoznaju pri planiranju zaštite?

12. Na koji način se štite podaci informacionog sistema?

13. Na koji način se štite aplikacije informacionog sistema?

14. Na koji način se štite serveri (čvorovi) informacionog sistema?

15. Na koji način se štiti interna mreža informacionog sistema?

16. Na koji način se štiti granica mreže informacionog sistema?

17. Na koji način se štiti spoljašnja mreža informacionog sistema?

18. Kako se bezbednosne opasnosti / problemi dele prema uzroku? Objasniti.
19. Kako se bezbednosne opasnosti / problemi dele prema načinu ispoljavanja? Objasniti.
20. Navesti uobičajene opasnosti po bezbednost informacionog sistema.
21. Objasniti opasnosti "greške i propusti", "prevare i krađe" i "infrastrukturni i fizički problemi".
22. Objasniti opasnosti "unutrašnja sabotaža", "industrijska špijunaža" i "međudržavna špijunaža".
23. Objasniti opasnosti "zlonamerni hakeri", "zlonamerni kod" i "pretnje ličnoj privatnosti".
24. U čemu se kriju uzroci postojanja problema bezbednosti?
25. Objasniti pojmove identifikacija, autentikacija i autorizacija.
26. Šta je i kako se vrši provera identiteta? Šta je falsifikovanje identiteta?
27. Objasniti najvažnije faktore autentikacije.
28. Objasniti praćenje aktivnosti kao deo sistema bezbednosti informacionog sistema.
29. Navesti uobičajene slabosti informacionog sistema u pogledu bezbednosti.
30. Navesti osnovna sredstva za prevazičaženje bezbednosnih slabosti informacionog sistema.

8.9 Literatura

Literatura za ovu temu su odabrana poglavlja iz [12], kao i [13] i [14].

Glava 9

Arhitekture orijentisane prema servisima

9.1 Uvod

Osnovni zadatak softverske arhitekture je da pruži rešenje problema, performanse, fleksibilnost i proširivost. Česta greška je zadržavanje na prva dva aspekta, bez dovoljnog razmatranja preostalih.

Fleksibilnost arhitekture predstavlja sposobnost da se čitava arhitektura prilagođava promenama u okruženju i primenjuje u novim okolnostima, i da se elementi arhitekture prilagođavaju promenama u okruženju, primenjuju u novim okolnostima i u drugom novim projektima. Fleksibilnost je bliska ponovnoj upotrebljivosti. **Proširivost arhitekture** je svojstvo arhitekture da može da se proširuje i da se proširivanje ostvaruje primarno dodavanjem novih elemenata, bez značajnog menjanja postojećih elemenata. Fleksibilnost i proširivost omogućavaju pojednostavljivanje životnog ciklusa sistema za sve učesnike, što se odnosi na: modularnost, konfigurabilnost, robusnost, mogućnost brze reakcije na promene u poslovnim zahtevima i drugo.

Arhitektura orijentisana prema servisima rezultat je evolucije softverske industrije sa ciljem postizanja maksimalne fleksibilnosti i proširivosti. Modularizacijom se celine sistema oblikuju u komponente koje mogu "samostalno" pružaju određene "celovite" usluge. Takve komponente se nazivaju **servisi**.

Pojam arhitekture orijentisane prema servisima definišemo na sledeći način:

- **Arhitektura orijentisana prema servisima** (engl. *Service-Oriented Architecture* (*SOA*)) jeste arhitekturalni stil koji promoviše korišćenje labavo povezanih servisa radi obezbeđivanja visoke fleksibilnosti na interoperabilan i tehnološki nezavisan način.

SOA je arhitekuralna paradigma koja se primarno odnosi na dizajn softvera (a ne na tehnologiju). Primena koncepta *SOA* uvodi nov koncept aplikacije.

- **Aplikacija** predstavlja kompoziciju i grupisanje labavo povezanih, usko definisanih i na standardima utemeljenih servisa u veće i složenije celine.

9.1.1 Uopštavanje komunikacije

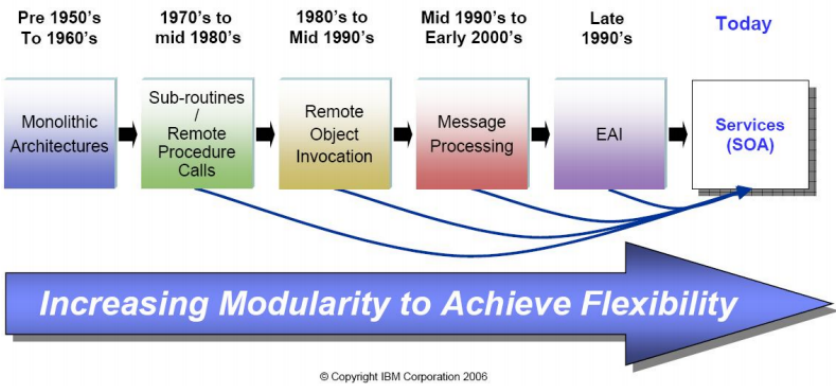
Da bi servisi bili ponovo primenjivi, potrebno je da imaju relativno standardizovane načine povezivanja i ostvarivanja komunikacije. Na nivou *SOA* neophodno je da logika bude razdvojena od komunikacije, da svi servisi jednog sistema koriste iste protokole komunikacije i povezivanja, da protokole komunikacije i povezivanja implementira posebna komponenta. Na nivou arhitekture (a ne tehnologije) se postiže visok nivo uopštavanja i fleksibilnosti. Komponenta za povezivanje obuhvata tehnološki zavisan kod a ostatak koda je lako prenosiv. U slučaju promene načina komunikacije, menja se ili zamenjuje samo komponenta za komunikaciju.

9.1.2 *SOA* i distribuirani objekti

Sistemi za distribuiranje objekata (*CORBA*, *COM*, itd.) kao osnovne jedinice imaju entitete. *SOA* kao osnovne jedinice ima *usluge* (*servise*). Servis odgovara jednom zaokruženom elementu poslovanja kao automatizovan ili slabo interaktivan slučaj upotrebe.

9.2 Istorijat arhitektura

Savremeni poslovni procesi su dinamični i distribuirani. Oni se odlikuju jasnom podelom odgovornosti među učesnicima i delovima procesa. Oni fluktuiraju, pri čemu su promene najčešće lokalnog karaktera – odnose se na pojedine delove ili učesnike. Softverska implementacija takvih procesa mora da prati njihovu logiku. *SOA* to čini vrlo dobro.



Slika 9.1: Istorijat arhitektura.

U istorijatu arhitektura (slika 9.1) razlikujemo sledeće:

- **Monolitna arhitektura**. Sve (podaci, korisnički interfejs, obrada) se nalazi u okviru jednog sistema.
- **Potprogrami i udaljene procedure**. Klijent ("glavni" program) šalje zahtev poznatom izvršiocu (potprogramu, udaljenom serveru) sa datim parametrima. Zatim klijent dobija odgovor i nastavlja sa radom.
- **Povezivanje udaljenih objekata**. Koncept poslovnih objekata (delova koji uključuju i ponašanje koje može da se menja u zavisnosti od konteksta), sistem distribuiranih objekata.
- **Obrada poruka**. Arhitekture distribuiranih objekata su uglavnom jako spregnute, arhitektura obrade poruka apstrahuje interakciju konceptom razmene poruka radi spuštanja nivoa spregnutosti.
- **Integracija poslovnih aplikacija** (engl. *Enterprise Application Integration* (*EAI*)). Unapređivanje i spajanje prethodnih arhitektura: udaljeni objekti (*CORBA*, *COM*, itd.), redovi poruka, razmena podataka preko XML-a, višeslojne arhitekture i dr.
- ***SOA***. Softverski elementi kao "samostalni" servisi. Otvoreni standard povezivanja. Projekat arhitekture ne ulazi u detalje implementacije već se bavi "sastavljanjem". Objektno-orijentisane metodologije razvoja. Savremeni principi razvoja (modularnost, enkapsulacija, niska spregnutost, ponovna upotrebljivost, implementacija kao sastavljanje, itd.).

Tabelom 9.1 prikazan je odnos između *SOA* i višeslojne arhitekture.

SOA	Višeslojna arhitektura
Heterogena	Homogena
Nezavisna od jezika	Zavisna od jezika
Masivno distribuirani servisi	Centralizovani nivoi aplikacije
Izdavač/pretplatnik	Vođenje zahtevima/odgovorima
Internet aplikacije bogate AJAX-om	HTML stranice

Tabela 9.1: *SOA* i višeslojna arhitektura.

9.3 Uglovi posmatranja na *SOA*

SOA se može posmatrati iz tri ugla:

- Iz *poslovnog* ugla, *SOA* je skup poslovnih servisa.

- Iz ugla *softverske arhitekture*, servisi su deo složene arhitekture koju čine *pružalac servisa* (engl. *service provider*), *korisnik servisa* (engl. *consumer*) i *opis interfejsa servisa* (engl. *interface-based service description*).
- Iz ugla *implementacije*, servisi se implementiraju primenom otvorenih standarda i protokola kao što su, na primer, veb servisi.

9.4 Servisi u kontekstu *SOA*

Servis je izdvojena (diskretna) jedinica poslovne (tehničke) funkcije, koja je ponovno upotrebljiva. Servisi su slabo spregnuti što znači da su definisani interfejsima nezavisnim od implementacije, a korisnik zavisi samo od interfejsa. Ovime se promoviše fleksibilnost pri izmenama implementacije. Takođe, servisi su povezivi, transparentni u odnosu na lokaciju, odnosno, ne koriste druge servise na osnovu lokacije, i međusobno sarađuju (interoperabilni su). Dalje, servisi su raspoloživi nezavisno od implementacije ili protokola prenosa (otvoreni standardi). Korisnici i pružaoci servisa mogu biti implementirani na različitim platformama, a komunikacioni protokoli servisa treba da budu kompatibilni sa različitim platformama.

Servise nudi pružaoc servisa (engl. *Service Provider*). Oni objavljuju interfejse i politike. Servise koriste korisnici servisa (engl. *Service Consumer*). Oni koriste interfejse primenom pouzdanih i bezbednih protokola. Takođe, u posredstvu su medijatori servisa (engl. *Service mediators*). Oni obezbeđuju otkrivanje, izbor, praćenje rada i dr.

Servisi se opisuju primenom standardizovanih postupaka za opisivanje interfejsa, tzv. **metapodaci servisa**. Određuju se servis, njegove operacije, ulazni i izlazni parametri, kako se servisu pristupa, kao i koja je njegova lokacija (engl. *service endpoint*).

Registar servisa prestavlja vid kataloga putem koga mogu da se obavljaju (engl. *publish*) i pronalaze (engl. *discover*) servisi. Servise objavljuju organizacije koje pružaju servise organizacije ili radne jedinice u organizaciji. Registar servisa je sredstvo za dodatno spuštanje nivoa spregnutosti. Korisnik ne mora da zna ko pruža servis, gde se fizički nalazi implementacija servisa, a u nekim slučajevima čak ni sve detalje interfejsa (operacija).

9.5 UDDI

Universal Discovery, Description, Integration (*UDDI*) jeste okruženje koje je nezavisno od platforme. Služi za: opisivanje servisa, otvaranje poslovanja, i integrisanje poslovnih servisa.

Čine ga: katalog sa informacijama o veb servisima, podrška za otkrivanje servisa, i sredstva za uspostavljanje odnosa između pružaoca i korisnika servisa.

9.6 Servisi i arhitektura

9.6.1 Vrste servisa

Osnovne vrste servisa su:

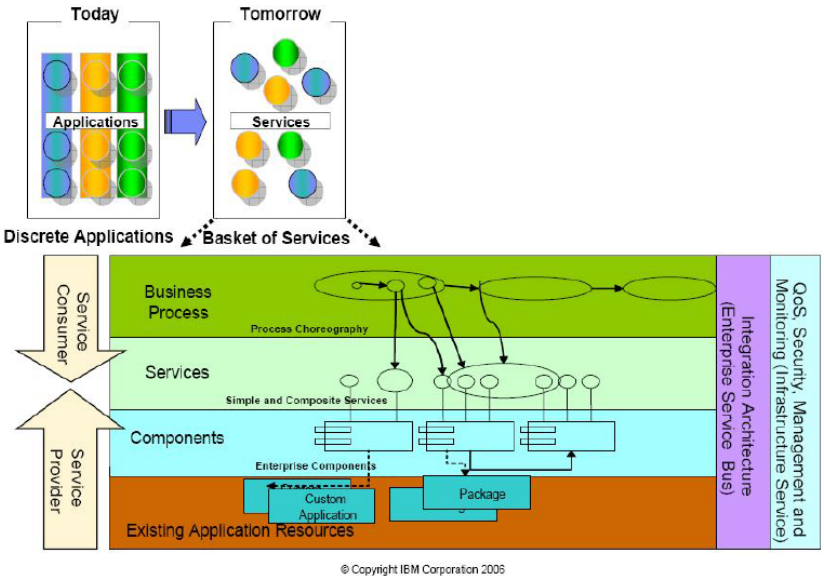
- **Servisi poslovnih procesa** (engl. *business process services*). Poslovni procesi su sekvence aktivnosti koje ispunjavaju jedan poslovni cilj. Jedan proces se može predstaviti kao servis. Na primer, proces prijave za dobijanje kredita. Servisi poslovnih procesa mogu da se sastoje od drugih servisa (tj. mogu ih koristiti).
- **Servisi poslovnih transakcija** (engl. *business transaction services*). Predstavljaju poslovne funkcije koje menjaju neko stanje poslovanja. Na primer, nabavka materijala, prenos sredstava, ili svako menjanje poslovnih podataka.
- **Servisi poslovnih funkcija** (engl. *business function services*). Predstavljaju funkcije koje ne menjaju stanje poslovnih procesa. Oni ili isporučuju podatke ili izvršavaju jednostavnija izračunavanja. Na primer, konverzija valuta, ili izveštaji (u formi podataka).
- **Servisi tehničih funkcija** (engl. *technical function services*). Predstavljaju ponovno upotrebljive servise koji ne pružaju poslovne funkcije, već obezbeđuju tehničke ili infrastrukturne funkcije. Na primer, praćenje događaja, provera autorizacija, i provera stanja sistema, raspoloživosti servisa i sl.

9.6.2 Servis kao fasada

Za postojeću aplikaciju se mogu napraviti i njoj dodeliti servisi "fasade". Oni izlažu, kroz interfejs servisa, funkcije aplikacije, a takođe i sakrivaju implementaciju.

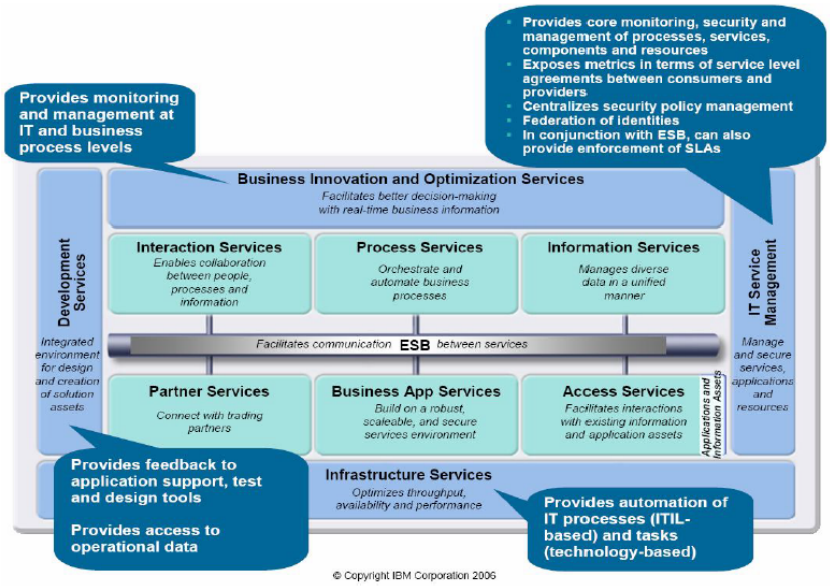
9.6.3 Slojevi arhitekture *SOA*

SOA arhitektura ima više slojeva. Jezgro čine: servisi, komponente koje ih implementiraju, i tokovi servisa.



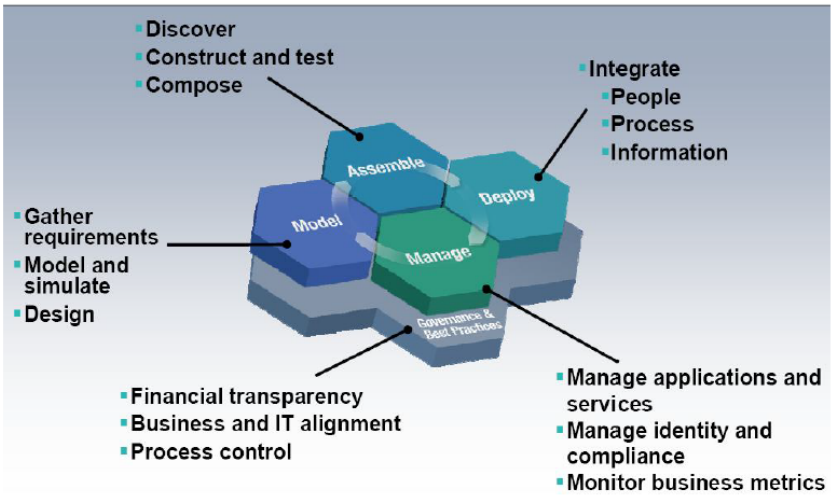
Slika 9.2: Slojevi arhitekture *SOA*.

9.6.4 Referentna arhitektura



Slika 9.3: Referentna arhitektura.

9.6.5 Životni ciklus



Slika 9.4: Životni ciklus.

9.7 Uloge u razvoju

Uloge u razvoju *SOA* su:

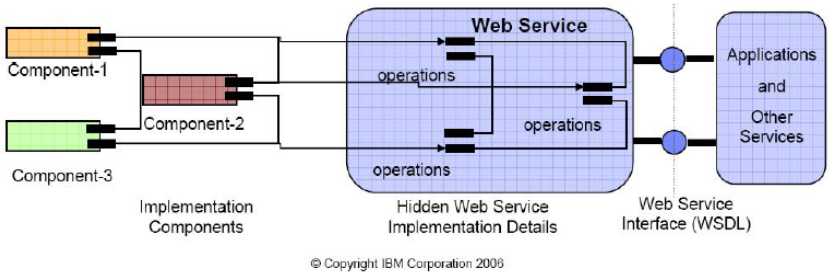
- **Analitičar poslovanja** (engl. *Business Analyst*). On prepoznaje i analizira poslovne procese. Odgovoran je za poznavanje poslovnih procesa i njihovo modeliranje.
- **Integrator** (engl. *Integration Developer*). On integriše funkcionalne komponente organizovanjem u bezbednu, složenu aplikaciju sa arhitekturom *SOA*.
- **Softverski arhitekta** (engl. *Software Architect*). On prepoznaje i projektuje servise i arhitekturu *SOA*.
- **Razvijalac aplikacija** (engl. *Application Developer*). On implementira komponente servisa koje su propisane projektom. Prepoznaje scenarija ponovne upotrebe i razvija komponente koje će se koristiti pri tome.
- **Administrator sistema** (*System Administrator*). On postavlja, konfiguriše, usklađuje, i upravlja složenim aplikacijama. Postavlja gotove aplikacije u proizvodno okruženje, i nadgleda izvršavanje.

9.8 Implementacija i alternative

SOA se može implementirati primenom veb servisa ili drugih tehnologija. Kod svake implementacije *SOA*, osnovna autonomna jedinica je servis. Ako to ne važi,

nije u pitanju *SOA*.

Primer veb servisa je *Web Services Description Language* (*WSDL*). To je jezik za opisivanje veb servisa, ali i standard koji se koristi za opisivanje Veb servisa. Počiva na XML-u.



Slika 9.5: Primer veb servisa.

Postoje brojne alternative *SOA*. One se često deklarišu kao *SOA* ili kao "vrsta Veb servisa", ali to nisu. Međutim, mogu biti sasvim dobro rešenje za neke slučajeve. Primeri alternativa *SOA* su:

- Sistemi distribuiranih objekata (*CORBA*, *COM*, itd.), kod kojih je autonomna jedinica *objekat*.
- REST, kod koje je autonomna jedinica *resurs*.
- Sistemi za obradu poruka, kod kojih je autonomna jedinica *red poruka*.

9.9 Pitanja

1. Šta je fleksibilnost arhitekture softvera?
2. Šta je proširivost arhitekture softvera?
3. U čemu je značaj fleksibilnosti i proširivosti softvera?
4. Šta je *SOA*?
5. Šta predstavlja aplikacija u okviru arhitekture orijentisane prema servisima?
6. Koje su osnovne karakteristike *SOA*?
7. Objasniti pojam servisa u kontekstu *SOA*.
8. Kako se opisuju servisi (*SOA*)?
9. Šta je registar servisa (*SOA*)?

10. Koje su osnovne vrste servisa u *SOA*?
11. Šta su servisi poslovnih procesa (*SOA*)?
12. Šta su servisi poslovnih transakcija (*SOA*)?
13. Šta su servisi poslovnih funkcija (*SOA*)?
14. Šta su servisi tehničkih funkcija (*SOA*)?
15. Šta su Veb servisi? Objasniti osnove Veb servisa.

9.10 Literatura

Literatura za ovu temu su odabrana poglavlja iz [1].

Glava 10

Virtualizacija

10.1 Pojam virtualizacije

Virtualno (engl. *virtual*) u svom izvornom značenju znači "skoro isto kao nešto", ali ne u potpunosti, ili ne u skladu sa strogom definicijom. U srpskom jeziku je poput reči "praktično". Uobičajeno u računarstvu znači nešto što ne postoji fizički, ali se softverski stvara utisak postojanja.

U skladu sa time, pojam virtualizacije možemo definisati na sledeći način:

- **Virtualizacija** je postupak stvaranja virtualne verzije nečega.

U kontekstu računarstva, sredstvo ostvarivanja virtualizacije je softver. Predmet virtualizacije može biti: realan fizički svet (virtualna stvarnost), i elementi računarskog sistema (virtualni računar, sistem, uređaj).

Kontrola računarskog sistema se može vršiti na različite načine:

- Prvobitna rešenja
  - Aplikativni softver – neposredno koristi hardver.
- Jednostavni monoprogramski sistemi
  - *BIOS* (engl. *Basic Input Output System*) – programi za neposrednu kontrolu hardvera.
  - Aplikativni softver – koristi hardver posredstvom *BIOS*-a, i samostalno upravlja resursima računara.
- Operativni sistemi
  - *BIOS* (engl. *Basic Input Output System*) – programi za neposrednu kontrolu hardvera.

  - Operativni sistem – programi za upravljanje resursima računarskog sistema.
  - Aplikativni softver – koristi sve računarske resurse (i hardver) posredstvom operativnog sistema.

Različiti koncepti/podsistemi OS apstrahuju različite elemente hardvera (apstrakcija elemenata hardvera). To su:

- Sistem datoteka, koji predstavlja apstrakciju diskova.
- Procesi, koji predstavljaju apstrakciju adresnog prostora i izolovane memorije.
- Niti, koji predstavljaju apstrakciju procesora.

Savremeni OS su veoma složeni i obavljaju veliki broj različitih funckija. Nivo apstrakcije OS je daleko iznad hardvera. Štaviše, isti OS često rade na potpuno različitom hardveru.

Savremeni OS imaju sloj za apstrakciju hardvera, koji se neposredno oslanja na (potencijalno specifičan) hardver i ostatku OS-a pruža ujednačenu apstrahovanu sliku. Aplikativni programi, kao i većina komponenti OS-a, rade koristeći usluge sloja apstrakcije hardvera, praktično potpuno nezavisno od konkretnih uređaja.

10.2 Virtualne mašine

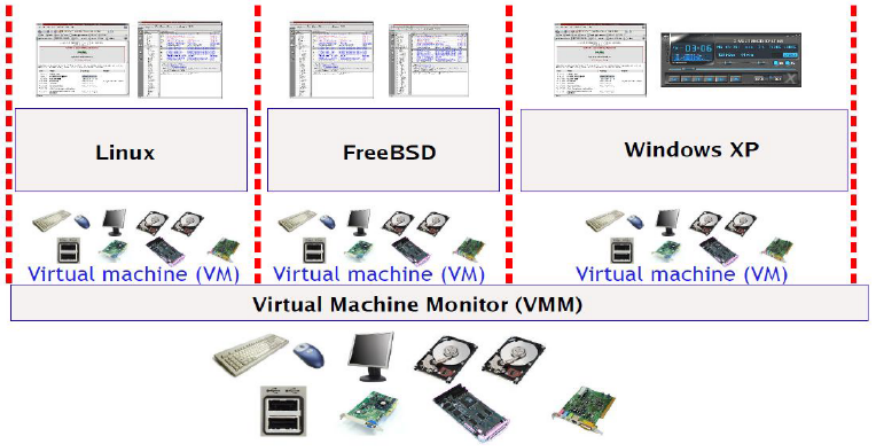
Koncept virtualne mašine predstavlja naredni nivo apstrakcije. Naime, sloj apstrakcije hardvera se izdvaja iz OS-a i predstavlja nezavisnu celinu koja se naziva **Nadzornik virtualnih mašina** (NVM, engl. *Virtual Machine Monitor* (VMM), ili *hypervisor*).

NVM se neposredno oslanja na hardver. Jedan NVM može istovremeno da opslužuje više operativnih sistema na istom računaru. Iz ugla svakog OS-a izgleda kao da je računar pod njegovom potpunom kontrolom. Iz ugla aplikativnog softvera nije vidljivo da li je OS neposredno nad hardverom ili se između nalazi NVM.

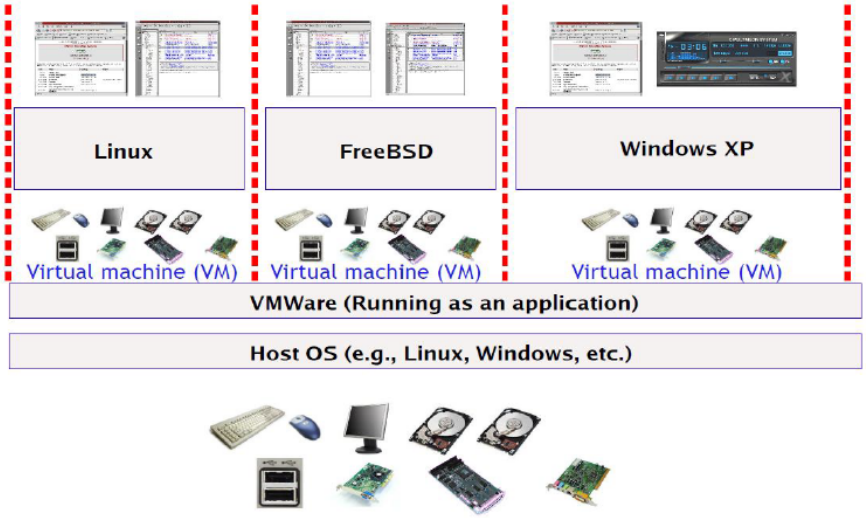
Termini koji se koriste su:

- **Nadzornik virtualnih mašina**. Sloj softvera koji se neposredno oslanja na hardver i podržava rad OS-a,
- **Virtualna mašina**. Apstrahovan računarski sistem koji radi pod NVM-om, a ne na fizičkom hardveru.

- **Virtualni uređaj** (engl. *Virtual Appliance*). Virtualna mašina sa instaliranim i konfigurisanim softverom, potpuno spremna za rad. Dovoljno ju je iskopirati na drugi računar i uključiti.
- **Mašina domaćin** (engl. *Host Machine*). Fizički računar i softver (OS ili NVM) koji radi na njemu.
- **Mašina gost** (engl. *Guest Machine*). Virtualni računar i softver koji radi na njemu, VM.



Slika 10.1: Primer arhitekture VM – Samostalan NVM, bez OS-a.



Slika 10.2: Primer arhitekture VM – NVM koji radi OS-om mašine domaćina.

10.3 Motivacija

Razlikuju se motivi za virtualizaciju radnih stanica, i servera.

Motivacija za virtualizaciju radnih stanica je:

- *Povećana produktivnost.* Neki programi ne postoje za sve vrste OS. Takođe, različiti OS nisu jednako pogodni za sve vrste poslova. Jednostavnije je imati više VM na jednom računaru nego održavati više fizičkih računara na jednom radnom mestu.
- *Izolovanje osetljivih sistema.* Ovim se dobija zaštita od virusa i drugih napasti. Pogodno je za obuku administratora, isprobavanje osetljivih postupaka, kao i obučavanje u razvoju sistemskog softvera.
- *Razvojne radne stanice.* Realno ponašanje mrežnog okruženja na jednoj radnoj stanici. Izgradnja na različitim razvojnim okruženjima, i testiranje.
- *Virtualni uređaji.* Pojednostavljena distribucija pripremljenih složenih okruženja.

Motivacija za virtualizaciju servera je:

- *Konsolidacija servera.* Više manjih servera se virtualizuje na jednom fizičkom računaru. Dobija se razdvajanje odgovornosti servera, a time i olakšano održavanje. Takođe, veća je iskorišćenost hardvera, a samim tim i ušteda na hardveru.
- *Napredna serverska okruženja.* Olakšana je replikacija. Dobija se jednostavnija implementacija visoke raspoloživosti. Prebacivanje aktivnih mašina sa jednog na drugi računar.

Ciljevi virtualizacije su:

- *Stvaranje iluzije kompletnih fizičkih mašina.* Odnosi se na: procesore, memoriju, nivoe zaštite, sistem prekida, ulazno-izlazne uređaje, itd.
- *Potpuna kompatibilnost sa postojećim fizičkim hardverom.* OS i aplikacije u VM moraju raditi u neizmenjenom obliku.
- *Potpuna međusobna izolovanost virtualnih mašina.* Softver jedne VM ne sme biti u stanju da stupi u kontakt sa virtualnim hardverom druge VM.
- *Omogućavanje istovremenog rada više OS-a.* Potreban je visok nivo skalabilnosti.
- *Visoke performanse.* Ovo se posebno odnosi na procesor i memoriju. Takođe, omogućava se dinamičko prilagođavanje opterećenju.
- *Fleksibilnost.* Pojednostavljeno je upravljanje raspoloživim resursima tokom rada VM, i pojednostavljen je transfer aktivne VM na drugi fizički računar.

10.4 Istorija virtualizacije računara

- *IBM Mainframe System M44* (1965). Postojala je potreba da se veliki sistem particioniše na više manjih sistema prema potrebama korisnika. Razvijen je koncept virtualizacije kao rešenje.
- Dijkstra (1967). On predlaže različite nivoe apsrakcije virtualnih mašina.
- IBM S/370, S/390, i dr. (od 1970. godine, pa nadalje). U svojim sistemima IBM koristi virtualne mašine kao sredstvo za upravljanje resursima. Razvijen je *IBM model* koji se odlikuje sledećim: fizička mašina se deli na virtualne replike koje su međusobno identične u svemu osim u količini raspoložive radne memorije. Memorija je potpuno razdvojena, čime je ostvarena potpuna bezbednost. Hardver sistema je prilagođen virtualizaciji. Većina operacija radi neposredno na hardveru, čime su ostvarene visoke performanse.
- Programski jezik *Oak* (1990). Oblikovana je prva verzija programskog jezika Oak, koja je počivala na primeni aplikativnih virtualnih mašina. Ovaj jezik je 1995. godine dobio novo ime – Java. Razvijen je *Java model*, koji se odlikuje sledećim: virtualne mašine su simulirane i prave se za svaki program posebno. U okviru VM se ne izvršava OS nego samo jedan program (zato što radi samo jedan program). Osnovna namena je prenosivost programa.
- *VMWare* i druga PC rešenja (1999). Razvijeno je prvo komercijalno rešenje pune virtualizacije za platformu x86. Za razliku od IBM-ovih rešenja, ovde hardver nije prilagođen. Potrebno je simulirati veći broj operacija, što je praćeno složenijom implementacijom i nižim performansama.

10.5 Vrste virtualnih mašina

Vrste virtualnih mašina su:

- **Sistemske virtualne mašine** (ili **hardverske virtualne mašine**). Pružaju apstrakciju čitavog računara. Primeri su: *KVM*, *VMWare*, *VirtualBox*, *XEN*, i dr.
- **Procesne (aplikativne) virtualne mašine**. Pružaju delimičnu apstrakciju računara, dovoljnu za izvršavanje jednog procesa, a mogu da podrže više programa (**aplikativni serveri**). U okviru VM se ne izvršava OS. Primeri su: *JVM*, *CLR*, i dr.
- **Virtualizacija na nivou OS-a**. Predstavlja kombinaciju, odnosno, u okviru jednog OS-a pružaju se parcijalne instance istog OS-a. Primeri su: *FreeBSD Jail*, *LXC*, *OpenVZ*, i dr.

10.5.1 Sistemske virtualne mašine

U sistemskim virtualnim mašinama razlikujemo sledeće nivoe virtualizacije:

- **Puna virtualizacija**. VM simulira sve elemente hardvera potrebne za rad neizmenjenog operativnog sistema VM. Prvi predstavnik je bio *IBM CP-40*. Primeri su: *VMware* (*Workstation*, *Server*), *Oracle VirtualBox*, i dr.

- **Hardverski podržana virtualizacija**. Hardver pruža mehanizme za efikasan rad nadzornika, simuliranje hardvera i izolovanje VM. Prvi predstavnik je bio *IBM VM/370* (1972.). Primeri su: *Intel/AMD x86* (2005.), *IBM Power Architecture*, *VMware* (*Workstation*, *Fusion*), *Oracle VirtualBox*, *Xen*, i dr.

- **Parcijalna virtualizacija**. VM simulira većinu (ali ne sve) elemenata hardvera. U okviru VM ne može da radi potpun OS, već može jedna ili više aplikacija. Može se reći na parcijalna virtualizacija predstavlja nešto između sistemske i procesne virtualizacije.

- **Paravirtualizacija**. VM ne simulira hardver već pruža alternativni API koji zamenjuje određene komponente gostujućeg OS-a. Naime, zamenjuju se one komponente OS-a koje neposredno pristupaju hardveru, a osetljive instrukcije se kodiraju u skladu sa potrebama VM. Ovim se dobija veća efikasnost, ali manja fleksibilnost. Uzrok toga je što sve VM moraju da izvršavaju isti OS ili NVM mora da pruži interfejs API-ja jednog OS za drugi. Primeri su: *KVM*, *Xen*, i dr. Neki sistemi pune virtualizacije (*Vmware* i dr.) koriste elemente paravirtualizacije za podizanje performansi (*VMwareTools*).

10.6 Implementaciona pitanja

Problemi u implementaciji koji se javljaju su:

- Gostujući OS pokušava da poziva privilegovane instrukcije. To ne bi smeo da čini, zato što to sme samo nadzornik (ili domaćinski OS).

- Gostujući OS pokušava da upravlja stranicama virtualne memorije. To proizvodi potencijalne probleme zato što je tesno povezano sa upravljanjem fizičkom memorijom.

- Gostujući OS mora da veruje da radi na fizičkoj mašini. VM mora da podržava sve instrukcije procesora, a neke narušavaju izolovanost. VM mora da podržava sve komponente rada sa virtualnom memorijom (straničenje, segmentacija, i sl.). Takođe, VM mora da podrži ulazno-izlazne uređaje.

10.6.1 Režim rada

OS po pravilu ima komponente koje rade u povlašćenom režimu, što omogućava punu kontrolu hardvera. VM po pravilu ne bi smela da radi u povlašćenom režimu zato što ne sme neposredno da pristupa fizičkom hardveru.

Posledica ovog zapažanja je da OS u VM mora da radi u korisničkom, a ne u povlašćenom režimu. Neki procesori (i x86) imaju i treći "među"-režim rada.

10.6.1.1 Režim rada – interpretacija

Jedno rešenje je interpretacija. VM ne izvršava instrukcije neposredno već se rad procesora simulira programom koji interpretira instrukcije. Ovim može da se ostvari puna bezbednost i izolovanost.

Problemi koji se javljaju su:

- *Složenost.* Potrebno je implementirati čitavu arhitekturu procesora.
- *Sporost.* Interpretiranje instrukcija procesora je za red veličine sporije od njihovog neposrednog izvršavanja.

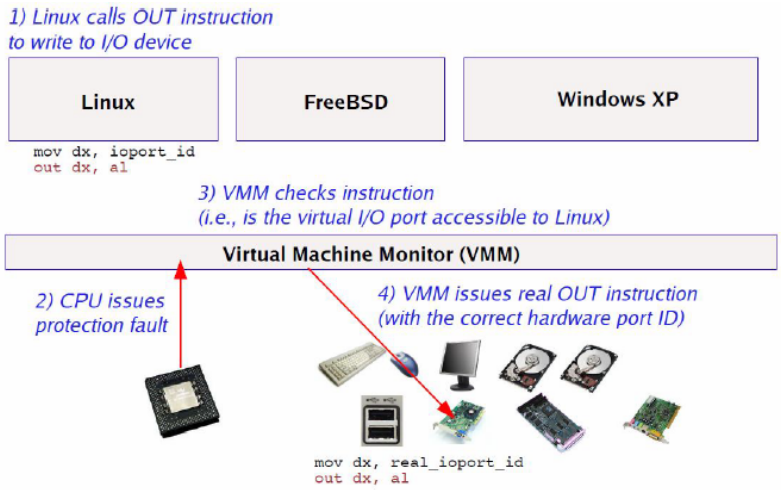
10.6.1.2 Režim rada – izvršavanje

Drugo rešenje je neposredno izvršavanje. VM neposredno izvršava instrukcije procesora, ali u korisničkom režimu. Ovakav način rada je visoko efikasan.

Pokušaji izvršavanja zaštićenih instrukcija izazivaju *zamke* i predaju kontrolu nadzorniku. Nadzornik proverava instrukciju i odlučuje šta dalje. Može da bezbedno emulira operaciju i nastavi rad VM, ili može da zabrani operaciju i isključi VM.

Problemi koji se javljaju su:

- *Umerena složenost.* Potrebno je implementirati sve zaštićene instrukcije procesora.
- *Umereno usporenje.* Interpretiraju se samo zaštićene instrukcije procesora.
- Neke instrukcije rade ispravno samo u zaštićenom režimu, ali ne proizvode greške inače.



Slika 10.3: Režim rada – izvršavanje.

Uobičajeno je da OS i aplikacije ne rade na istom nivou privilegija. Ako rade na istom, može doći do problema usled bezbednosti ili ispravnosti rada. Zbog toga, mogu da se koriste dodatni nivoi zaštite procesora (dovodi do pojave tzv. **prstena**), koji inače nisu iskorišćeni:

- prsten 0: NVM,
- prsten 1: VM OS,
- prsten 3: VM aplikacije.

Ovo se naziva izolovanje OS.

10.6.1.3 Režim rada – prevođenje

Treće rešenje je prevođenje koda pri učitavanju. Neposredno pre izvršavanja programa (ili čak tokom izvršavanja) nadzornik "protrči" kroz kôd i prevede sve problematične instrukcije. Na primer, x86 instrukcija `POPF` bi trebalo da postavi novo stanje zastavica, ali u korisničkom režimu ne može da postavi zastavice prekida. Prave se kopije stranica koda u memoriji i odgovarajuće tablice preslikavanja.

Problem koji se javlja je:

- *Smanjena efikasnost.*

Paravirtualizacija je vid ovakvog rada, s tim da su svi delovi OS-a koji koriste privilegovane instrukcije unapred prevedeni i prilagođeni radu u VM. Ovakav način rada je relativno jednostavan i uglavnom efikasan.

10.6.2 Rad sa memorijom

Ne sme svaka VM da implementira sopstvenu virtualnu memoriju, zato što postoji samo jedna fizička memorija. Zbog ovoga se uvodi dodatni nivo indirekcije. VM smeju samo da čitaju tablice stranica, a tablice održava nadzornik.

10.7 Virtualizacija i informacioni sistemi

U razmatraju odnosa virtualizacije i informacionih sistema, prirodno se javljaju dva pitanja. Prvo je pitanje infrastrukturnih problema informacionih sistema, a drugo je koje od njih može da olakša upotreba virtualizacije.

10.7.1 Infrastrukturni problemi informacionih sistema

Neki infrastrukturni problemi informacionih sistema su:

- **Migracija**. Prebacivanje sistema ili dela sistema sa jednog na drugi računar najčešće nije jednostavno. Preduzima se u dva slučaja: kada dođe do kvara računara, i kada se unapređuje hardverska ili softverska platforma.
  Migracija predstavlja često veoma zahtevan posao, a može da zahteva period neaktivnosti sistema.

- **Udvajanje servisa**. Predstavlja pravljenje kopije postojećeg sistema. Preduzima se radi: obezbeđivanja redundantnosti (prevencija kvarova), i pravljenja test verzije sistema (ili dela sistema). *Obezbeđivanje redundantnosti* je neophodno da bi u slučaju kvara jednog servera neki drugi preuzeo njegov posao. Svaki server se mora posebno instalirati. Replike često stoje ne radeći ništa ili radeći vrlo malo. *Pravljenje test verzija (dela) sistema* se ogleda kroz pravljenje posebne razvojne, probne, i produkcione verzije.
  Udvajanje nije jednostavno. Pre svega, potreban je dodatni hardver. Svaki put je potrebno instalirati OS i sav potreban softver. Idealan slučaj je da sistemi imaju isti hardver, inače je još teže.

- **Skaliranje**. Predstavlja prilagođavanje sistema uslovima povećanog opterećenja. Može se ostvariti dodavanjem memorije, procesora, diskova itd, međutim, postoje granice. Može da zahteva migraciju ili unapređenje softvera. Često se izbegava inicijalnim predimenzioniranjem hardvera.

- **Niska iskorišćenost servera**. Često se teži da jedan server obavlja samo jednu funkciju, radi lakšeg konfigurisanja, lakšeg održavanja pojedinačnih servisa, i lakšeg skaliranja.
  Međutim, često nijedan od njih nije efikasno iskorišćen (uobičajeno 5-15%). Takođe, javljaju se veliki troškovi zbog nabavke, održavanja, i administriranja hardvera.

- **Heterogenost**. Predstavlja raznovrsnost hardvera u okviru informacione infrastrukture.
  Heterogenost otežava mnoge poslove, a pre svega administraciju, udvajanje servisa, i migraciju.

10.7.2 Doprinosi virtualizacije

Osnovni kriterijumi su: spuštanje cene (hardvera, i održavanja), podizanje fleksibilnosti, i skraćivanje perioda neaktivnosti sistema.

Posebno razmatramo: serversku virtualizaciju, i virtualno računarstvo.

Doprinosi virtualizacije servera su:

- **Konsolidacija**. Postavljanjem više malih VM na jedan fizički server, dobijaju se sledeće pogodnosti: štedi se prostor, smanjuje se potrošnja struje, i smanjuje se učestalost hardverskih kvarova (doduše, kvarovi imaju potencijalno veće posledice).

- **Udvajanje virtualnih mašina**. Udvajanje VM je mnogo jednostavnije. Umesto ponavljanja instalacije, VM se samo iskopira. Pravljenje kopija VM na različitim serverima je jednako sigurno kao držanje fizički različitih servera. Ako rezervna VM ne radi ništa na nekom serveru, znači da ima prostora za još drugih VM koje mogu nešto da rade. U slučaju kvarova, kada rezervne replike počnu da rade, može da dođe do pada performansi, ako nije dobro isplanirano.

- **Dinamična test okruženja**. Pojednostavljeno udvajanje VM ima za posledicu lakše odlučivanje za pravljenje višestrukih test okruženja. Dobijaju se sledeće pogodnosti: lakše pravljenje i održavanje, niža cena, povećana bezbednost, i uspešnije i blagovremeno uočavanje i rešavanje problema.

- **Migracija virtualnih mašina**. Prebacivanje VM sa jednog na drugi server je sasvim jednostavno. Na novu mašinu se instalira samo NVM, ili OS i NVM, a zatim se iskopiraju potrebne VM. Dodatnu pogodnost nude savremena komercijalna rešenja koja dopuštaju da se VM prebacuju sa jedne na drugu fizičku mašinu bez prekida rada. Migracija VM omogućava savremen metod upravljanja opterećenjem, kroz raspoređivanje VM po fizičkim serverima.

- **Skaliranje**. Skaliranje sistema je lakše u virtualizovanom okruženju. Kada se doda novi fizički server, na njega se prebace izabrane VM i time se rasterete zagušeni serveri. Sve to je mnogo brže i jednostavnije uz skraćivanje perioda neaktivnosti sistema.

10.8 Dodatna pitanja i tehnike

10.8.1 Ograničenja virtualizacije servera

Virtualizacija servera nije efikasna ako je server podložan preterano visokom opterećenju nekog hardverskog resursa, kao što su procesor, memorija, diskovi, mreža, itd. (ili više njih). Deljenjem snage na više VM se smanjuje snaga svake od VM. Ako na serveru postoji samo jedna VM i ona radi manje efikasno nego što bi radila da nije VM. Čak i tada je lakše migrirati VM na novi računar nego u slučaju fizičkih mašina. U većini slučajeva migracija VM je ograničena na platforme sa istim tipom procesora.

10.8.2 Virtualno računarstvo

Virtualno računarstvo (engl. *Virtual Computing*) jeste tehnika koja obezbeđuje da se jedan računar ponaša kao da je neki drugi. To u praksi znači da je se jedan računar ponaša kao terminal, a drugi mu pruža odgovarajuću uslugu i potreban hardver.

Doprinosi virtualnog računarstva su:

- *Puna kontrola nad udaljenim fizičkim računarom.* Korisnik fizički radi na slabom računaru, a zapravo se programi izvršavaju na udaljenom snažnom računaru. Korisnik može izdaleka da upravlja radom servera.

- *Puna kontrola nad udaljenim virtualnim računarom.* Ovo se najviše odnosi na repliciranu infrastrukturu preduzeća. Korisnici sa svojih ličnih stonih ili prenosnih računara imaju terminalski pristup virtualnim računarima na serverima preduzeća. Infrastruktura je unifikovana. Lako je održavanje virtualnih radnih stanica. Sve VM su međusobno identične replike. Softver se jedanput instalira, a zatim se kopiraju VM. Lako se upravlja pravima i bezbednošću. Ako se korisnikov lični računar zarazi virusima ili drugim napastima, njegova poslovna VM je izolovana i bezbedna.Takođe, na lokalnim računarima se uglavnom ne čuvaju podaci od poslovnog značaja, pa je pojednostavljeno pravljenje rez. kopija.

10.8.3 Računarstvo u oblaku

Naredni korak u virtualizaciji je **računarstvo u oblaku**. Čitava računarska mreža se apstrahuje jednim velikim virtualnim računarskim sistemom. Prividno mnogo programa radi na jednom velikom sistemu, a ustvari se izvršavaju pojedinačno, ili čak po delovima, na različitim računarima te mreže.

10.8.3.1 Oblici računarstva u oblaku

Osnovni oblici računarstva u oblaku su:

- **Virtualne mašine u oblaku**. Cilj virtualnih mašina u oblaku je: potpuno apstrahovanje (i sakrivanje od korisnika) fizičke platforme, univerzalna primena, kao i u slučaju pravih računara, i olakšano skaliranje, restauracija, održavanje, i dr.
- **Virtualne usluge (servisi) u oblaku**. Cilj virtualnih usluga u oblaku je: dodatno apstrahovanje usluge, da se korisniku pruži puna usluga, kao da je instalirao softver na svom računaru, da se korisnik u potpunosti oslobodi staranja o računaru na kome se usluga izvršava i fokusira na samu uslugu, o operativnom sistemu, o softveru koji pruža uslugu, i o drugom, i da se korisnik fokusira na konkretnu uslugu koju koristi.

10.8.3.2 Pružaoci usluga u oblaku

Danas veliki broj kompanija pruža korisnicima virtualne mašine i računarstvo u oblaku. Postoje različiti nivoi pružavanja računarstva u oblaku: širok opseg usluga i VM, samo VM, i samo uži opseg usluga.

Neki od većih pružalaca usluga računarstva u oblaku su: *Amazon Web Services*, *Bluelock Virtual Datacenters and cloud services*, *Dell Cloud Dedicated Service*, *Google Cloud Engine*, *IBM SmartCloud*, *Joyent Manta Storage Service*, *Microsoft Windows Azure Infrastructure Services*, *ProfitBricks*, *Rackspace Hybrid Cloud*, *Stratogent Corp*, i *Verizon Terremark Enterprise Cloud*.

10.9 Pitanja

1. Objasniti pojmove virtualno i virtualizacija.
2. Šta je virtualna mašina? Navesti osnovne komponente koje učestvuju u virtualizaciji sistema.
3. Objasniti motivaciju za virtualizaciju računara.
4. Ukratko navesti istoriju virtualizacije računara.
5. Koje su osnovne vrste virtualnih mašina? Objasniti specifičnosti i razlike.
6. Šta su sistemske virtualne mašine? Na koje se sve načine implementiraju?
7. Šta je puna virtualizacija? Šta je hardverski podržana virtualizacija?
8. Šta je parcijalna virtualizacija? Šta je paravirtualizacija?

9. Šta se primarno analizira pri razmatranju primene virtualnih mašina u nekom IS? Koji su mogući doprinosi virtualizacije servera u IS?
10. Objasniti konsolidaciju, kao doprinos virtualizacije? Objasniti.
11. Objasniti redundantnost, kao doprinos virtualizacije? Objasniti.
12. Objasniti migraciju i skaliranje, kao doprinose virtualizacije? Objasniti.
13. Koja su najvažnija ograničenja virtualizacije računara? Kada virtualizacija nije dobro rešenje?

10.10 Literatura

Literatura za ovu temu su odabrana poglavlja iz [15].

Glava 11

Računarstvo u oblaku

11.1 Pojam računarstva u oblaku

Definicija pojma računarstvo u oblaku glasi:

- **Računarstvo u oblaku** je skup hardverski zasnovanih usluga, koje nude računske, mrežne i skladišne resurse uz visoko apstrahovano upravljanje hardverom.

Računarstvo u oblaku obezbeđuje dinamičku skalabilnost, virtualizuje resurse, i pruža se kao usluga na Internetu.

Počiva na: virtualnim mašinama, virtualizovanom skladišnom prostoru, i pružanju različitih infrastrukturnih servisa. O virtualnim mašinama je bilo reči u prethodnom poglavlju, tako da ćemo sada obratiti pažnju na druga dva.

Virtulizovan skladišni prostor se ostvaruje kroz: **distribuirane sisteme datoteka**, kao što su *Google File System* (*GFS*), i *Hadoop Distributed File System* (*HDFS*), i **klasterizovane sisteme datoteka**, kao što su *VMware vStorage VMFS*, i *XenServer Storage Pool*.

Servisi mogu biti: Veb servisi (Veb servisi u užem smislu, formalno definisani), REST servisi, ali i druge vrste servisa slične funkcionalnosti.

11.2 Prethodnici računarstva u oblaku

Prethodnici računarstva u oblaku su: klasteri, grid, i "korisno" računarstvo.

11.2.1 Klasteri

Klasteri su povezani sistemi koji sarađuju radi obezbeđivanja visokih performansi.

Često korisnik ima utisak da koristi jedan veliki računar. Uobičajeno su povezani lokalnom mrežom. Oni nude visoku skalabilnost uz predstavu jednog servera. Koriste se kao redundantna visoko raspoloživa rešenja i omogućavaju raspoređivanje opterećenja i deljeno izračunavanje.

11.2.2 Grid

Grid (ili **rešetka**, **mreža**, **matrica**) udružen je skup autonomnih distribuiranih resursa.

Često su rezultat udruživanja više organizacija. Resursi su slabo spregnuti i često veoma udaljeni. Koriste se u slučajevima kada se izračunavanja (ili druga usluga koja se pruža) mogu podeliti na veći broj nisko međuzavisnih zadataka. Upotreba ima nisku fleksibilnost.

11.2.3 "Korisno" računarstvo

"Korisno" računarstvo (engl. *utility computing*) model je računanja kod koga korisnici prave poslove i određuju im "korisnost".

"Korisnost" se određuje prema važnosti, rokovima, složenosti, komercijalnom značaju, itd. Pružaoci usluga pokušavaju da maksimizuju sopstvenu korisnost u odnosu na komercijalni profit. Davanjem prioriteta poslovima koji sa visokom korisnošću u odnosu na zauzeće resursa sistem se ponaša kao otvorena berza poslova i usluga. Može se posmatrati kao nadgradnja grida.

11.3 Modeli i vrste oblaka

Modeli oblaka su:

- **Privatni oblaci**. To su oblaci koji se koriste za sopstvene potrebe.
- **Javni oblaci**. To su oblaci koji se koriste za pružanje usluga drugima.
- **Hibridni oblaci**. To su oblaci koji kombinuju prethodna dva pristupa.

Vrste oblaka su:

- **Infrastruktura kao usluga**. Poslužilac nudi sve elemente računarske infrastrukture u virtualizovanom obliku, što obuhvata: virtualnu mašinu (procesore), skladišni prostor (diskove), memoriju, mrežnu infrastrukturu, itd. -Softver se implemetira i pušta u rad praktično isto kao u stvarnim uslovima.

- **Platforma kao usluga**. Poslužilac apstrahuje infrastrukturu i nudi platformu za izvršavanje aplikacija, što podrazumeva: aplikativne servere, baze podataka, drugi vidovi skladišnog prostora, itd. Softver se implementira i pušta u rad na poseban način da bi radio na specifičnoj platformi.

- **Softver kao usluga**. Poslužilac nudi pojedinačne softverske usluge, koje mogu da se koriste i kao celina i pojedinačno. To su: usluga elektronske pošte, usluga plaćanja putem kreditne kartice, usluga skladišnog prostora, i usluga stalno raspoložive veb lokacije. Softver se implementira i pušta u rad skoro isto kao u klasičnim uslovima, s tim da se podešava komunikacija sa uslugama.

11.4 Bezbednost u oblaku

Aspekti bezbednosti u oblacima su: upravljanje identitetima, fizička i lična bezbednost, dostupnost podataka, bezbednost aplikacija, privatnost podataka, i pravni aspekti.

Mesta zaštite podataka su: zaštita podataka u transportu, zaštita podataka na mestu čuvanja, i zaštita podataka na mestu upotrebe (praktično ovde spada i zaštita podataka na mestu nastajanja).

Dostupnost podataka se odnosi na: raspoloživost podataka, rezervne kopije i oporavak, i dnevnike i tragove izvršavanja.

Privatnost podataka predstavlja ozbiljan problem u javnim oblacima, posebno ako se segmenti istih oblaka koriste na više strana. Na primer, *Google* (ili *Facebook*, *Yahoo*, *MS*, i dr.) vrše autentikaciju u različitim Veb aplikacijama. Ovaj problem je manje prisutan problem privatnim oblacima (naime, postoji, ali se ne razlikuje u odnosu na klasične informacione sisteme).

Karakteristični napadi oblaka su:

- **Uvijeni napad** (engl. *Wrapping*). Predstavlja vid napada na podatke u transportu. Odvija se na nekom od transportnih slojeva (na primer, *SOAP*) i uobičajeno na klijentskoj strani. Presreću se najpre zahtevi, a kasnije i odgovori, i uvijaju se u nove pakete zahteva i odgovora.

- **Zlonamerni kôd** (engl. *Malware Injection*). Predstavlja podmetanje zlonamernog koda maskiranog u ispravan servis oblaka.

- **Zatrpavanje** (ili **poplavljivanje**) (engl. *Flooding*). Predstavlja zatrpavanje oblaka zahtevima sa neispravim elementima. Predmeti ostvarivanja napada su autentikacija, i formati podataka.

- **Krađa podataka** (engl. *Data Stealing*). Predstavlja krađu podataka o korisnicima na neki od načina uobičajenih za Veb. Ukradeni podaci se zatim koriste za dalje ugrožavanje sistema.
- **Napad na obračunavanje** (engl. *Accountability checking*). Ideja je da se u ime ispravnog korisnika šalju različiti zahtevi. Cilj je veštačko naduvavanje računa korišćenja oblaka. Ovim se dolazi do narušavanje odnosa oblaka i klijenta.

11.5 Pitanja

1. Objasniti pojam "računarstvo u oblaku"?
2. Koje su osnovne vrste oblaka? Objasniti razlike.
3. Objasniti vrstu oblaka "Infrastruktura kao usluga".
4. Objasniti vrstu oblaka "Platforma kao usluga".
5. Objasniti vrstu oblaka "Softver kao usluga".
6. Objasniti karakteristične aspekte bezbednosti računarstva u oblaku.

11.6 Literatura

Literatura za ovu temu su odabrana poglavlja iz [16], [17] i [14].

Glava 12

Pokretno računarstvo

12.1 Uvod

Definicija pojma pokretno računarstvo glasi:

- **Pokretno računarstvo** (ili **mobilno računarstvo**) jeste svaki računarski sistem kod koga je neki od značajnih elemenata pokretan.

Elementi koji mogu biti pokretni su:

- pokretni fizički elementi, a to su: računari, i bazne stanice, i
- pokretni logički elementi, a to su: instance aplikacije, i softverske komponente.

Savremeni pokretni uređaji imaju veću računarsku snagu nego stoni računari od pre svega nekoliko godina. Iz ugla računarske snage, pokretni i nepokretni korisnički uređaji su danas praktično izjednačeni.

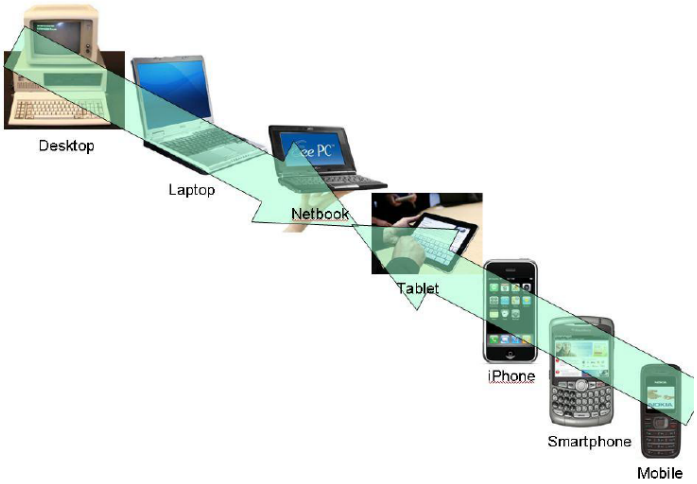
Kada je razlika u snazi u pitanju, možemo primetiti sledeće. Pokretni uređaji su dovoljno snažni da obavljaju većinu uobičajenih kancelarijskih poslova, kao što su: upotreba Veba i Interneta, uobičajen kacelarijski softver, uobičajen softver za zabavu, ali i mnogo više od toga. Takođe, poslovi radnih stanica se danas mogu preneti na pokretne uređaje, ali samo u prilično uprošćenom obliku. Na primer, iOS omogućava montažu filmova snimljenih telefonom. Ipak, pokretni uređaji još neko vreme neće moći da preuzmu ulogu servera. Međutim, to i ne spada u sagledive masovno poželjne ciljeve.

Kada je razlika u upotrebljivosti u pitanju, pokretni uređaji ne nude dovoljno fleksibilan korisnički interfejs za efikasno obavljanje poslova. To se pre svega ogleda kroz ograničenu veličinu uređaja (mali ekran, i mali prostor za upravljanje), i činjenicu da novi načini upotrebe još nisu sasvim sazreli (dodir, pokret, i glas). Njihov razvoj praktično još traje, a nije jasno koliko daleko može da ide.

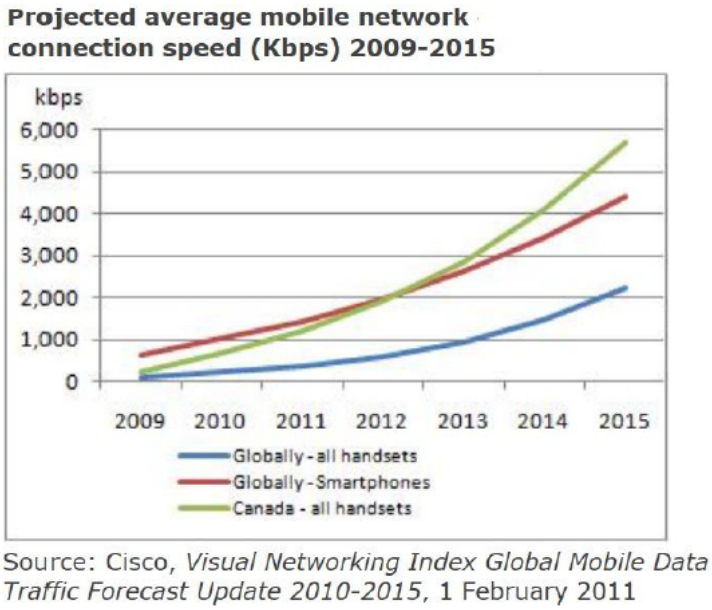
U razvoju pokretnog računarstva, obraća se pažnja na dva cilja:

- *Visoka prenosivost*, odnosno, mala veličina i dugo trajanje baterije.
- *Visoka upotrebljivost*, odnosno, velika (ili bar dovoljna) računarska snaga i upotrebljiv korisnički interfejs.

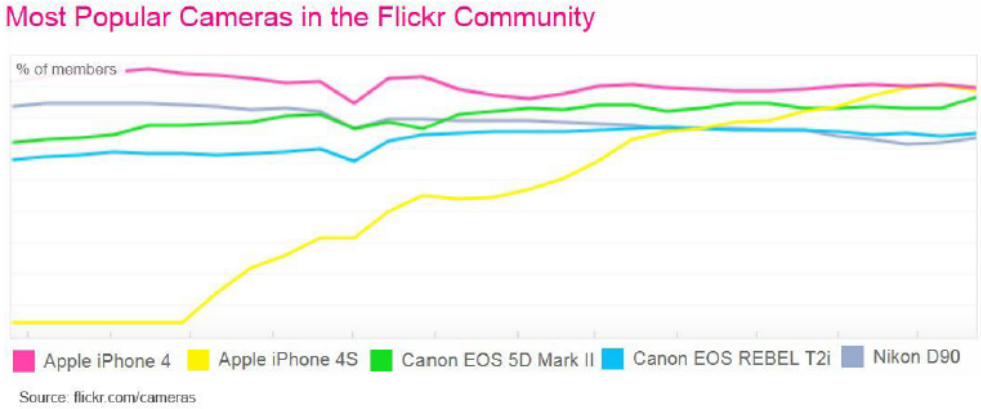
Naredne slike imaju za cilj da približe napredak pokretnog računarstva.



Slika 12.1: Balans u prenosnom računarstvu.



Slika 12.2: Brzina prenosa podataka raste.



Slika 12.3: Mali uređaji se koriste sve više.

Slika 12.4: Raznolikost upotrebe raste.

Neki termini koji se koriste u ovoj oblasti su:

- **Nomadsko računarstvo**. Označava da je pokretljivost moguća samo u nekom ograničenom domenu. Na primer, poslovni centar ili univerzitet (pokretljivost je ograničena politikom upotrebe), ili laptop koji ima samo žično povezivanje (pokretljivost je ograničena mogućnostima uređaja).

- **Pokretno/mobilno računarstvo (u užem smislu)**. Označava globalnu pokretljivost bez ograničenja. Na primer, bežično povezivanje (*WiFi*, *GSM*, itd.).

- **Sveprisutno/raširenо računarstvo**. Predstavlja "ideal" pokretnog računarstva, odnosno, povezivanje je moguće u svako doba, na svakom mestu, i svakome.

12.2 Odnos sa informacionim sistemima

Iz ugla informacionih sistema pokretno računarstvo ima sledeće uloge: proširuje oblast primene, proširuje domen korisnika informacionog sistema, i donosi nove izazove (čitaj "probleme").

Na primeru informacionog sistema fakulteta:

- Da li možemo da prepoznamo nove primene?
  - stalna povezanost
  - redovnija komunikacija
  - automatsko identifikovanje
  - upotrebljivost na svakom mestu
  - podrška naučnim istraživanjima
- Da li prepoznajemo nove korisnike?
  - praktično svi, stalno, svuda
  - terenski rad
- Da li prepoznajemo nove probleme?
  - povezivanje komponenti
  - bezbednost

12.2.1 Izazovi

Opšti izazovi su: aplikacije svesne pokretljivosti, prepoznavanje lokacije i nadležnosti, prosleđivanje podataka i poruka, pouzdanost u slučaju prekida veze ili promene načina povezivanja, upravljanje podacima, transakcioni modeli, i bezbednost.

Mnogi autori dele izazove na:

- **Fizičke izazove**. U ove izazove spadaju: fizička slabost prenosivih uređaja, ograničenost izvora energije, i mogućnost gubljenja ili krađe uređaja.
- **Izazove povezivanja**. U ove izazove spadaju problemi oko uspostavljanja i održavanja aktivne veze.
- **Izazove performansi**. U ove izazove spadaju ograničenja performansi prenosnih uređaja.
- **Izazove terminala**. U ove izazove spadaju ograničenja korisničkog interfejsa prenosnih uređaja.

12.3 Distribuiranost. Podaci. Mobilni informacioni sistemi

Većina savremenih sistema imaju neke odlike distribuiranih sistema. Tipičan primer je Veb. Što veću operativnu funkcionalnost ima pokretni klijent, to je složenost distribuiranja veća.

Uticaj na podatke se ogleda kroz pojavu nekih problema, kao što su: položaj podataka, položaj jedinica za obradu, upiti "svesni" položaja i vremena, i implementacija transakcija.

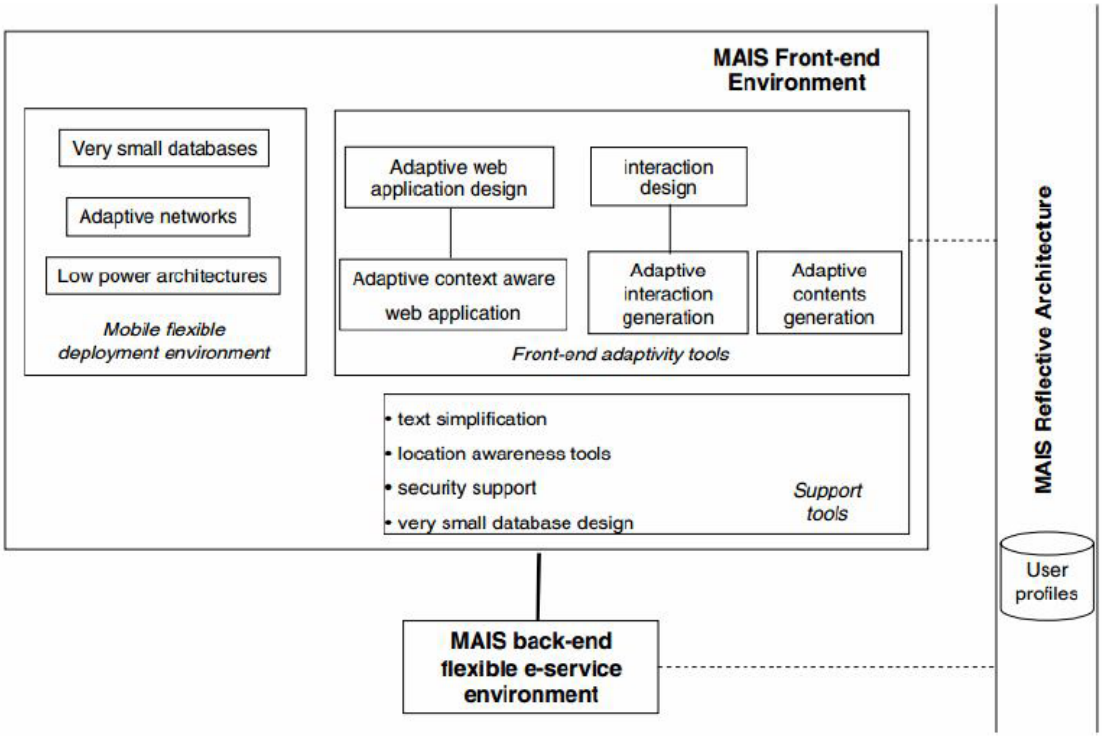
Karakteristike savremenih mobilnih informacionih sistema (MIS) su:

- *Mobilni uređaji sa višekanalnim pristupom.* Na primer, *GSM*, *GPRS*, *GPS*, *WiFi*, itd.
- *Kontekstno zavisne funkcije.* To su prostorno i vremenski zavisne funkcije koje uzimaju u obzir lični i socijalni kontekst korisnika. Javlja se potreba za adaptivnim rešenjima.
- *Bežična infrastruktura.* Podrazumeva se slobodnije kretanje.
- *SOA i Kooperativni mobilni IS.* Podrazumeva se modularni razvoj.

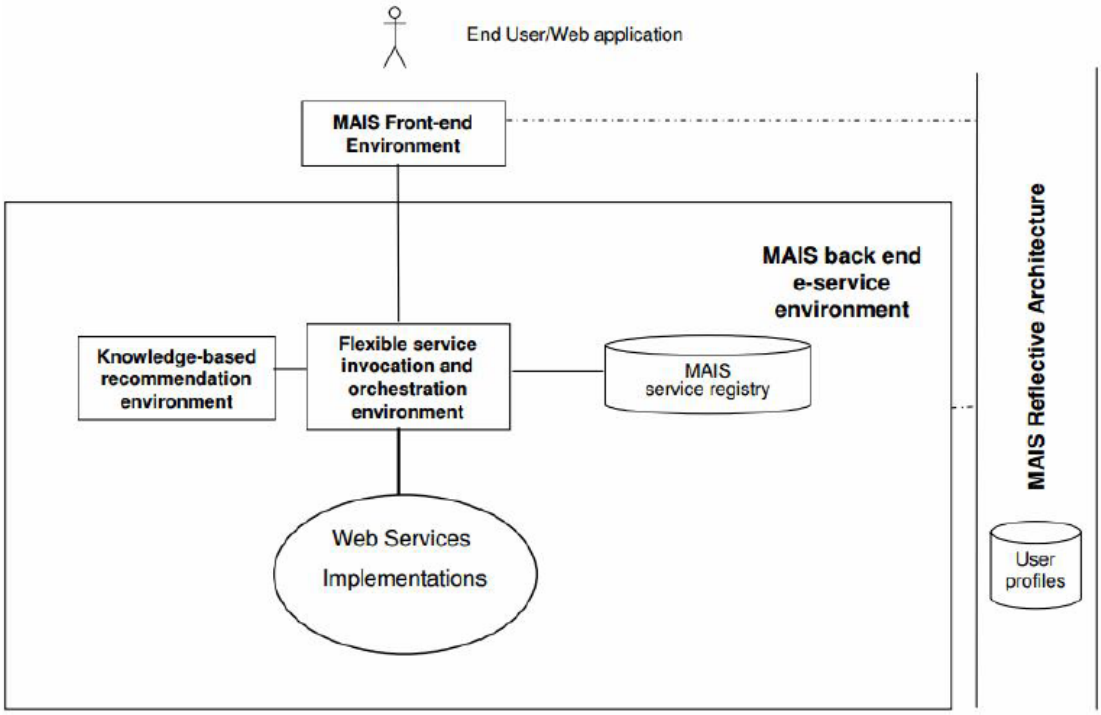
12.3.1 Primer arhitekture MIS

Osnovne komponente jedne ovakve arhitekture su:

- **Korisničke komponente** (engl. *front-end*). Pružaju interakciju sa korisnicima i izgrađuju Veb aplikacije ([18]).
- **Pozadinske komponente** (engl. *back-end*). Pružaju servise u mobilnom informacionom sistemu, sa različitim kontekstom i kanalima pružavanja servisa ([18]).
- **Reflektivna infrastruktura**. Pruža informacije o stanju komponenti, kvalitetu servisa, i karakteristikama i kontekstu korisnika.
- **Infrastrukturne komponente**. Pruža infrastrukturu za čuvanje i prenos podataka, i komunikacionu infrastrukturu.



Slika 12.5: Korisnički deo MIS.



Slika 12.6: Pozadinski deo MIS.

12.3.2 Kategorizacija podataka

Neophodno je uvođenje koncepta podataka zavisnih od položaja:

- **Podaci zavisni od položaja** (engl. *Location-Dependent Data* (*LDD*)). Predstavlja tip podataka u kojem su vrednosti usko povezani sa specifičnom geografskom lokacijom ([19]). Na primer, iznos poreza ili telefonski pozivni broj.

- **Upiti zavisni od položaja** (engl. *Location-Dependent Query* (*LDQ*)). To su upiti čiji rezultat zavisi od mesta sa koga se postavljaju. Na primer, "pronaći najbliži restoran", ili "izračunati udaljenost od aerodroma do *ovde*".

- **Upiti "svesni" položaja** (engl. *Location-Aware Query* (*LAQ*)). To su upiti koji sadrže položajne odrednice, ali ne zavise od mesta sa koga se postavljaju. Na primer, "izračunati razdaljinu od Dalasa do Kanzas Sitija" ([19]).

Uobičajeni vidovi distribuiranja su:

- **Particionisanje**. Podrazumeva da se različiti podaci čuvaju na različitim lokacijama.

- **Delimična replikacija**. Podrazumeva da se neki podaci ponavljaju na različitim lokacijama.

- **Puna replikacija**. Podrazumeva da se svi podaci ponavljaju na svim lokacijama.

Distribuiranje podataka zavisnih od položaja se vrši na tri osnovna načina:

- **Horizontalna fragmentacija**. Podaci se raspodeljuju prema položajima na osnovu ključeva koji obuhvataju podatak o položaju.

- **Vertikalna fragmentacija**. Jedan oblik je da se podaci raspodeljuju prema položajima tako da različite tabele odgovaraju različitim položajima.

- **Particionisanje baze podataka**. Raspodelivanje se vrši na osnovu podataka o položaju.

Prostorna replikacija podrazumeva postojanje više "kopija" podataka, koje mogu imati različite ispravne vrednosti u bilo kom trenutku vremena. Ispravnost podataka se posmatra u odnosu na konkretan položaj (oblast). Jedna kopija se naziva **prostorna replika**.

Vremenska replikacija se odnosi na kopije podataka koji mogu imati samo jednu ispravnu vrednost u bilo kom trenutku vremena. Jedna kopija se naziva **vremenska replika**.

12.4 Transakcije

Pokretna jedinica može "proizvoljno" uspostavljati i prekidati vezu sa sistemom. Veza se može prekinuti i neplanirano. Pokretna jedinica može da promeni položaj u toku transakcije. Pokretljivost proizvodi dodatne probleme u odnosu na obradu transakcija, a pre svega u odnosu na konzistentnost podataka.

12.4.1 Uticaj pokretljivosti na atomičnost

Atomičnost je osobina transakcije da baza podataka nikada ne sme da sadrži podatke koji su proizvod samo delimično izvršene transakcije.

Pokretljivost ne utiče na pojam, ali otežava implementaciju atomičnosti. **Dnevnici transakcija** su uobičajeno sredstvo za obezbeđivanje atomičnosti u centralizovanim sistemima. U slučaju pokretljivosti ne mogu da se primenjuju u istom obliku.
Dnevnici transakcija mogu da se vode na tri mesta:

- **Pokretna jedinica** (engl. *Mobile Unit* (*MU*)). Nije dobro usled slabih mogućnosti pokretnih jedinica.
- **Bazna stanica** (engl. *Base Station* (*BS*)). Ovo je najčešće korišćeno rešenje.
- **Centar za priključivanje mobilnih jedinica** (engl. *Mobile Switching Center* (*MSC*)). Nije dobro usled velikog ukupnog obima operacija u sistemu.

12.4.2 Uticaj pokretljivosti na konzistentnost

Konzistentnost je osobina transakcije da je baza podataka u konzistentnom stanju ako su ispoštovani svi uslovi integriteta definisani za tu bazu podataka.

U centralizovanim i distribuiranim sistemima postoji samo jedna ispravna vrednost za svaki podatak. Kažemo da je replicirana baza podataka u **uzajamno konzistentnom stanju** ako sve kopije podataka imaju iste vrednosti.

U pokretnim sistemima postoje dve vrste konzistntnosti:

- **Prostorna konzistentnost**. Sve vrednosti podataka prostorne replikacije se odnose na tačno jedan region i zadovoljavaju uslove konzistentnosti definisane za taj region.
- **Vremenska konzistentnost**. Sve vremenske replike (na različitim položajima) podataka moraju imati iste vrednosti

Svaka pokretna jedinica koja započne transakciju u nekom regionu mora dobiti konzistentan pogled na region. Baza podataka mora da obezbedi trajnost efekata izvršavanja transakcije u tom regionu. Zbog toga region mora da zadovolji i prostornu i vremensku konzistentnost.

12.4.3 Uticaj pokretljivosti na izolovanost

Izolovanost je osobina transakcije koja nalaže da nijedna transakcija ne sme da ometa rad druge transakcije.

Svaki segment izvršavanja mora da zadovoljava uslove izolovanosti na nivou na kome se izvršava. Izolovanost se može obezbeđivati nezavisno za različite regione, ali samo tako da bude ostvarena u odnosu na celovitu bazu podataka.

12.4.4 Uticaj pokretljivosti na trajnost

Trajnost je osobina transakcije da sve potvrđene izmene moraju ostati trajno zapisane u bazi podataka.

U slučaju pokretljivih sistema, trajnost se odnosi kako na globalne, tako i na regionalne podatke. Na slučajeve prostornih i vremenskih replika odnose se (redom) regionalna i globalna trajnost podataka.

12.4.5 Uticaj pokretljivosti na potvrđivanje transakcija

Koncept potvrđivanja se ne razlikuje od distribuiranih sistema. Kažemo da fragment izvršavanja **zadovoljava potvrđivanje zavisno od položaja** akko se sve operacije u tom fragmentu završavaju potvrđivanjem i postoji preslikavanje položaja u podatke.

Tada sve operacije u fragmentu rade nad prostornom replikom koja odgovara konkretnom regionu. Transakcija i njeno potvrđivanje se odvijaju na jednom položaju.

Skup podataka na kome se radi može da bude neposredno određen položajem ili da zavisi od položaja. Na primer, "rezervisati 2 mesta u ribljem restoranu u krugu od 1km", ili "rezervisati 2 mesta u ribljem restoranu u krugu od 1km od fakulteta". Čitava transakcija je locirana u kopiji podataka koja se odnosi na jedan region. Region se određuje na početku transakcije.

12.5 Povezanost pokretne jedinice

Pokretna jedinica može da bude:

- **Neprekidno povezana**. Mobilna jedinica je neprekidno povezana sa serverom baze podataka. Opciono može da kešira potrebne podatke radi podizanja performansi ili ih svaki put ponovo tražiti od servera. Ako je potrebno, može da uđe u **pritajeni režim** (engl. *doze*), u kojem ne komunicira, ali sluša baznu stanicu, radi uštede električne energije i zatim da se ponovo aktivira. Ovaj režim je skup za održavanje i obično nije neophodan za obrađivanje korisničkih poslova.

- **Nepovezana**. Nakon osvežavanja podataka mobilna jedinica samoinicijativno prekida vezu i nastavlja obradu podataka lokalno. U planirano (unapred određeno) vreme se povezuje i šalje sve potrebne podatke serveru. Server primenjuje primljene podatke tako da se očuva globalna konzistentnost.

- **Povremeno povezana**. Sličan koncept kao pri nepovezanosti. Veza se može prekinuti u bilo kom trenutku od strane sistema, korisnika, ili usled spoljnih faktora. Međutim, ne postoji planirano vreme uključivanja ili isključivanja veze. Korisno je za pokretne agente koji rade sa korisnicima (isporuka pošiljki, uređivanje artikala u samoposlugama, itd.). *Povezivanje na zahtev* je jedan vid povremene povezanosti.

12.5.1 Problemi usled nepovezanosti

Konzistentnost podataka se teško definiše za nepovezane i povremeno povezane pokretne jedinice. Posledica je da nije trivijalno određivanje okvira i obima transakcija.

U zavisnosti od vrste problema, neke transakcije su bezbedne i u slučaju nepovezanosti, a druge su rizične (nisu bezbedne).

12.6 Pokretni SUBP (PSUBP)

Pokretni SUBP (engl. *Mobile DBMS* (*MDS*)) pruža punu funkcionalnost SUBP-a uz pokretljivost korisnika i delova sistema. Korisnik može započeti transakciju bez obzira na svoj položaj. Garantuje se lokalna konzistentnost, globalna konzistentnost, i mogućnost oporavka sistema.

Osnovne osobine PSUBP su:

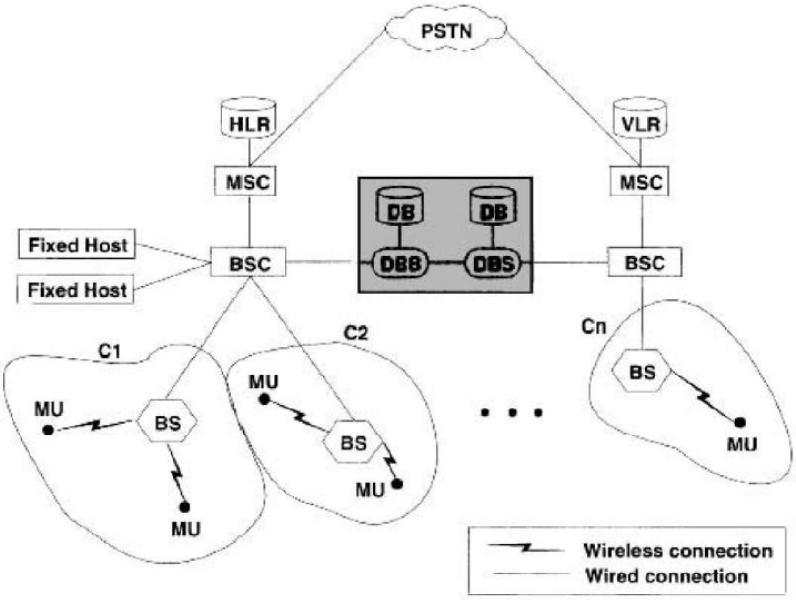
- geografska pokretljivost (klijenata),

- mogućnost uspostavljanja i raskidanja veze,

- obrada podataka (na obe strane),
- bežična komunikacija,
- transparentnost (obrada podataka na strani klijenta ne utiče na njegove komunikacije), i
- skalabilnost.

U sistemu računari imaju naredne uloge:

- Fiksno povezani računari predstavljaju: nepokretne čvorove (engl. *Fixed Host* (*FH*)), bazne stanice (engl. *Base Station* (*BS*)), i stanice za podršku pokretljivosti (engl. *Mobile Support Stations* (*MSS*)).
- Pokretne jedinice mogu biti: pokretni čvorovi (engl. *Mobile Host* (*MH*)), ili pokretne jedinice (engl. *Mobile Unit* (*MU*)).



Slika 12.7: Referentna arhitektura PSUBP-a.

Referentna arhitektura PSUBP-a data je na slici 12.7. Oznake koje se koriste u arhitekturi su:

- BSC – kontroler baznih stanica (ili kontroler položaja ćelija) (engl. *Base Station Controler*),
- MSC – centar za priključivanje mobilnih jedinica (engl. *Mobile Switching Center*),
- HLR – glavni registar položaja (engl. *Home Location Register*),

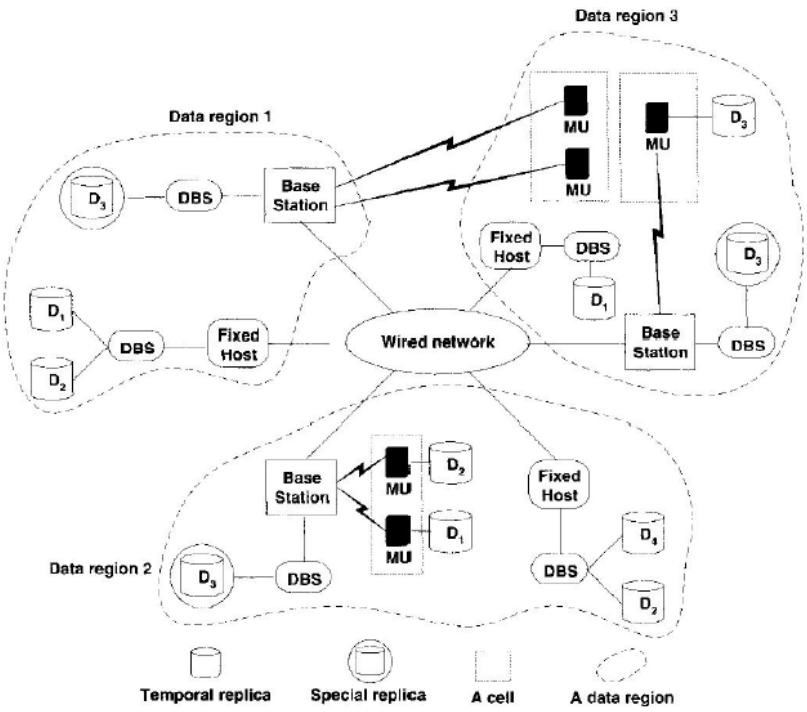
- VLR – registar položaja posetilaca (engl. *Visitor Location Register*),
- PSTN – javna komunikaciona mreža (engl. *Public Switched Telephone Network*),
- DB – baza podataka, i
- DBS – SUBP.

Jedna bazna stanica pokriva: jednu ćeliju (prostor) čija veličina zavisi od snage bazne stanice, i sve pokretne jedinice u okviru te ćelije.

Vrste replikacije su date narednom tabelom, a primeri različitih replikacija su prikazani na slici 12.8.

	Bez replikacije	Vremenska replikacija	Prostorna replikacija
Kopije	Jedna	Više	Više
Broj ispravnih vrednosti	Jedna	Jedna po trenutku	Jedna po položaju i trenutku
Arhitektura	Centralizovana	Distribuirana	Mobilna
Pokretljivost	Ne	Ne	Da

Tabela 12.1: Vrste replikacija.



Slika 12.8: Primeri različitih replikacija.

12.6.1 Transakcije u PSUBP-u

Različiti delovi transakcija se potencijalno izvršavaju na različitim računarima. Neophodna je upotreba **koordinatora** i **mehanizma dvofaznog potvrđivanja transakcija**.

Koordinator može da bude:

- **Centralizovan**. Jedan od čvorova ima ulogu koordinatora transakcija. Ovo je uobičajeno za PSUBP.
- **Delimično repliciran**. Neki od čvorova mogu da imaju ulogu koordinatora.
- **Potpuno repliciran**. Svaki od čvorova može da ima ulogu koordinatora.

Koordinator bi trebalo da ima: neposrednu i stalnu vezu sa drugim čvorovima, visoku raspoloživost, i, u idealnom slučaju, i neprekidno napajanje i dovoljno prostora za podatke.

12.6.1.1 Položaji koordinatora

Mogući položaji koordinatora su na:

- *MU*, što nije pogodno zbog odsustva stalne veze sa ostalim komponentama sistema i slabih kapaciteta.
- *DBS*, *FH*, što predstavlja rešenje koje se obično izbegava zato što obično nemaju neposrednu vezu sa svim *DBS* ili *MU*.
- *MSC*, što predstavlja potencijalnog kandidata, ali ima samo posrednu vezu sa *MU*, preko *BS*.
- *BS*, što je uobičajeno rešenje. *BS* imaju vezu sa svim *MU* u svojoj zoni odgovornosti (ćeliji), imaju vezu sa fiksnim čvorovima (*DBS*), ali nemaju neposrednu vezu sa *MU* van zone odgovornosti, što može predstavljati problem. Međutim, čak i tako je to otprilike jednako složeno kao i u slučaju *MSC*.

12.6.1.2 Položaji transakcije

Položaji transakcije su sledeći:

- Transakcija se može inicirati od strane *MU*, i *DBS*.
- Transakcija može menjati podatke na *MU*, i *DBS*.
- Ako se odvija na jednom mestu, koordinator ima manji značaj.

Transakcije koje započinje *MU* nude tri moguća scenarija:

- transakcija se izvršava na *MU* i skupu *DBS*,
- transakcija se izvršava samo na skupu *DBS*, i
- trasakcija se izvršava samo na *MU*.

U svakom slučaju konačan rezultat se nalazi na *MU* koja je inicirala transakciju. Nijedna druga *MU* nije uključena u izvršavanje transakcije, i to zbog privatnosti i zbog nepouzdanosti upotrebe (drugih) *MU*.

Transakcija koju započinje *DBS* se izvršava samo na skupu *DBS*. Konačan rezultat se nalazi na *DBS* koji je inicirao transakciju. Ovakve transakcije se odvijaju kao da nije u pitanju pokretljiv sistem.

Postoje dva načina izvođenja transakcija *MU*:

- Ako *MU* može samostalno da izvrši celu transakciju, onda sama izvršava i potvrđuje transakciju na keširanim podacima.
- Ako *MU* ne može da samostalno izvrši celu transakciju, onda postoje dve mogućnosti:
  - Prenosi potrebne podatke od *DBS* prema *MU*. Postupak je sledeći:
    1. identifikuje potrebne podatke,
    2. komunicira sa *DBS* preko *BS* i čita podatke,
    3. lokalno izvršava i potvrđuje transakciju, i
    4. šalje ažurirane podatke na *DBS* preko *BS* kako bi se ugradili u bazu podataka.

    Ovaj pristup ne zahteva koordinatora transakcije, ali i potencijalno proizvodi nepotrebno veliki protok podataka i dugačko zauzeće (zaključavanje) podataka.
  - Prenosi transakciju skupu čvorova. Postupak je sledeći:
    1. *MU* deli transakciju na skup podtransakcija,
    2. zadržava za sebe one delove koje može sama da izvrši,
    3. ostale delove raspoređuje nekom skupu *DBS*, i
    4. trenutna *BS* preko koje je *MU* povezana postaje koordinator transakcije.

    Ovaj tehnički složeniji pristup omogućava efikasniju komunikaciju i upotrebu podataka.

U odnosu na kretanje *MU* tokom izvođenja transakcije postoji nekoliko mogućnosti:

- ***MU* se ne pomera**. Transakcija počinje i dovršava se na *MU*. *MU* se ne pomera tokom trajanja transakcije, ili se eventualno pomera samo u okviru "svoje" ćelije, što je iz ugla sistema ekvivalentno mirovanju. Slično je uobičajenoj centralizovanoj obradi.

- ***MU* se pomera uz lokalnu obradu**. Transakcija počinje i u potpunosti se izvršava na *MU*. Potrebni podaci se prikupljaju sa drugih čvorova. Tokom trajanja transakcije *MU* se premešta u druge ćelije. Ovakvo izvršavanje se uobičajeno naziva "lokalno atomično". Ovim se minimizuje komunikacija tokom transakcija.

- ***MU* se pomera uz distribuiranu obradu**. Transakcija započinje na *MU* i zatim se deli po drugim čvorovima (fragmentacija). Uobičajeno su to fiksni čvorovi. Podtransakcije se distribuiraju između *MU* i skupa *DBS*. Određuje se koordinator transakcije koji upravlja izvršavanjem. Tokom trajanja transakcije *MU* se premešta u druge ćelije.

12.6.1.3 Povezivanje *MU* i koordinatora

Usluge koordinatora moraju biti neprekidno raspoložive pokretnoj jedinici tokom trajanja transakcije. Veza može da se prekine ako se *MU* premesti u drugu ćeliju. Veza može da se održava na dva načina: statički, i dinamički.

Statički metod držanja veze podrazumeva sledeće: transakcija započinje na *MU*, koordinator transakcije postaje *BS* te *MU*, i koordinator se ne menja do kraja transakcije. Ako se *MU* premesti u drugu ćeliju, mora da obavesti novu *BS* o tome ko je koordinator tekuće transakcije. Slabost metoda je što i koordinator i *MU* mogu da budu prinuđeni da komuniciraju posredstvom više *BS*. Posledica je složenije vođenje dnevnika transakcije, i složenije potvrđivanje transakcije.

Dinamički metod držanja veze podrazumeva sledeće: transakcija započinje na *MU*, i koordinator transakcije postaje *BS* te *MU*. Ako se *MU* premesti u drugu ćeliju, tada nova *BS* postaje koordinator tekuće transakcije. Svaki od čvorova (*DBS*) koji učestvuju u transakciji mora biti obavešten o promeni koordinatora, pri čemu je odgovornost za obaveštavanje na novom koordinatoru. Slabost metoda je što u slučaju višestruke promene ćelije (iz jedne u drugu i nazad) može doći do gubitka sinhronizacije u procesu menjanja koordinatora. Ovo se prevazilazi izbegavanjem menjanja koordinatora u takvim slučajevima.

12.6.1.4 Modeli transakcija. Kretanje i konzistentnost.

Uobičajeni model transakcija *ACID* ne odgovara u potpunosti. Razlozi za to su: prekidi veze (engl. *Handoff*), specifični režimi (pritajeni (engl. *doze*), nepovezan

(engl. *disconnected*), i zahtevano nepovezan (engl. *forced disconnected*)), nedostaci resursa (memorija, bežična veza, itd.), i položajno-zavisni podaci.

Novi pojmovi koji se uvode su:

- **Geografski domen**, u oznaci G, ukupna je oblast pokrivena svim ćelijama (u oznaci $C_i, i = \overline{1, n}$):

$$G = (C_1 + C_2 + \ldots + C_n).$$

- **Položaj** je jedna tačka u geografskom domenu koja predstavlja najmanju prepoznatljivu poziciju u okviru domena. Svaki položaj se označava jedinstvenom oznakom L.

U praksi, položaj se obično određuje u odnosu na *BS*.

U slučaju PSUBP koncept konzistentnosti je složeniji:

- uključuje podatke koji zavise od položaja,
- geografske podatke,
- konzistentan pogled se dobija u odnosu na položaj,
- transakcija može da postavi vrednost podatka koja ima smisla samo na konkretnom položaju, i
- identična transakcija na drugom mestu može biti predmet drugačijih uslova integriteta.

Zbog toga se uvodi se koncept **prostorne konzistentnosti**, a uobičajena konzistentnost se naziva **vremenska konzistentnost**.

12.7 Pitanja

1. Šta je pokretno računarstvo?
2. U čemu je značaj pokretnog računarstva u odnosu na informacione sisteme?
3. Šta je pokretni sistem za upravljanje bazama podataka? Objasniti osnovne razlike u odnosu na nepokretne sistema za upravljanje bazama podataka.
4. Objasniti odnos pokretljivosti pokretnih jedinica i načina izvršavanja transakcija.

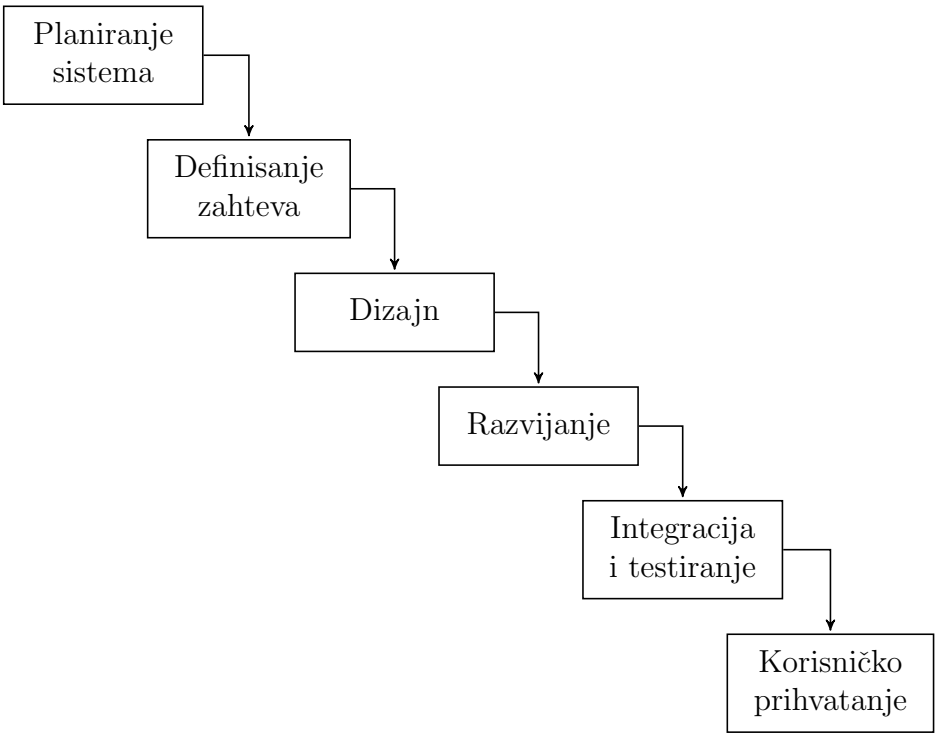
12.8 Literatura

Literatura za ovu temu su odabrana poglavlja iz [18] i [19].

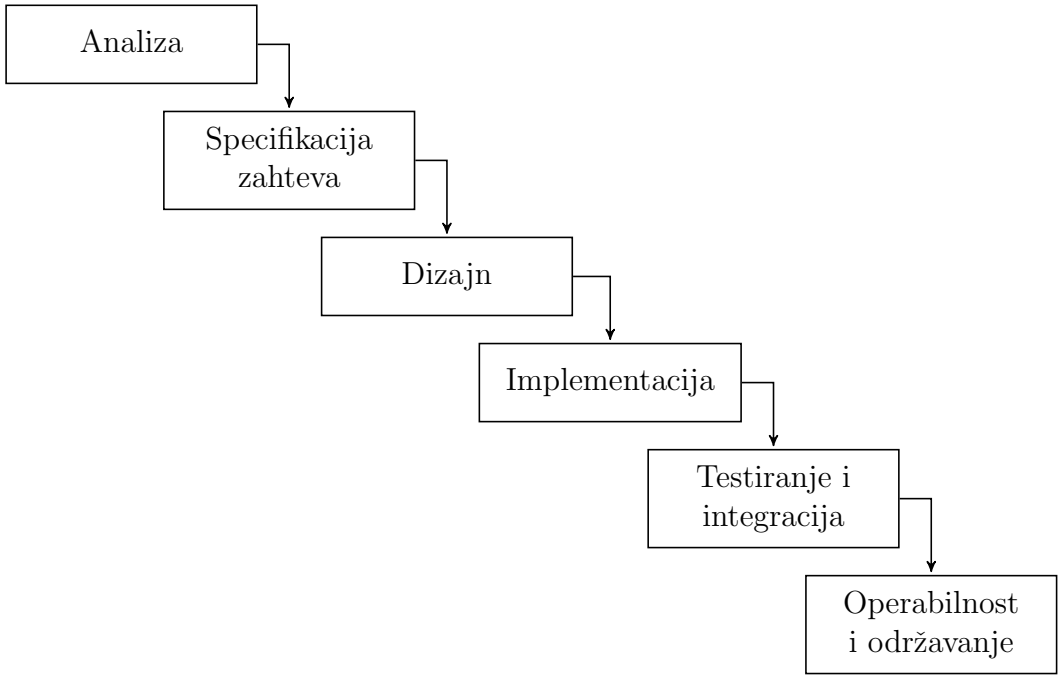
Dodaci

Varijante modela vodopada

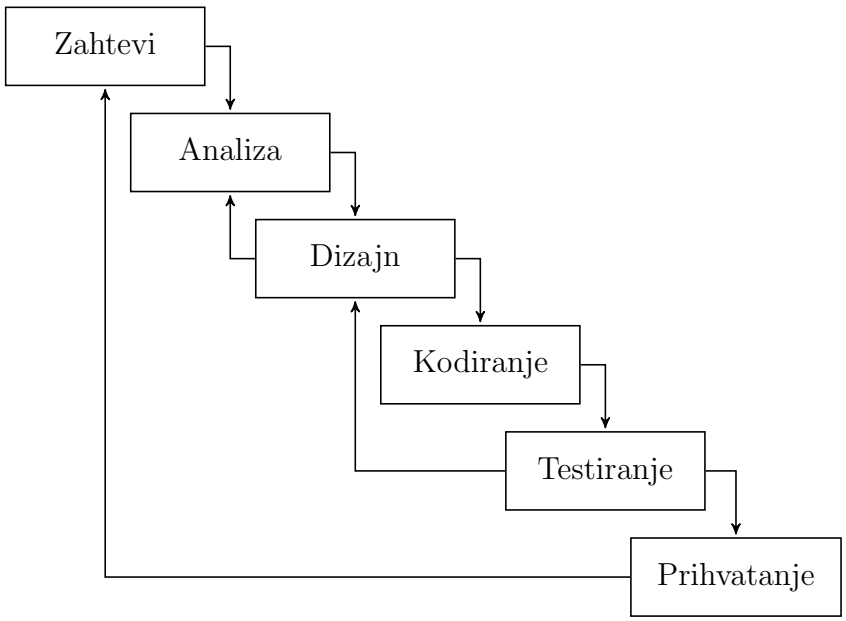
Na slikama 9 i 10 nalaze se varijante modela vodopada koje su slične već prikazanoj varijanti na slici 2.1. Jedina razlika između ove tri varijante jesu nazivi faza.



Slika 9: Varijanta modela vodopada – primer 1.



Slika 10: Varijanta modela vodopada – primer 2.

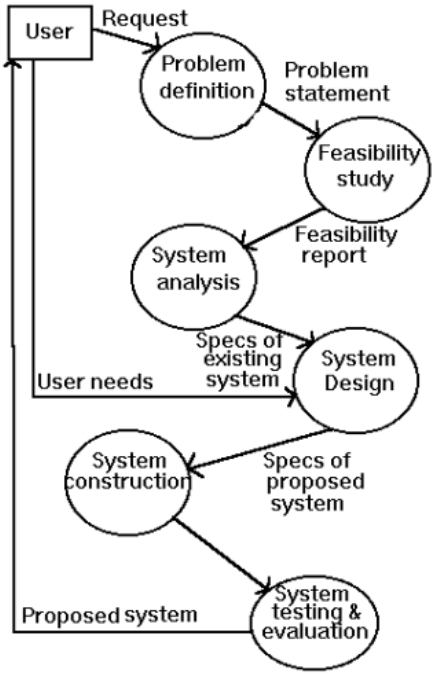


Slika 11: Varijanta modela vodopada – primer 3.

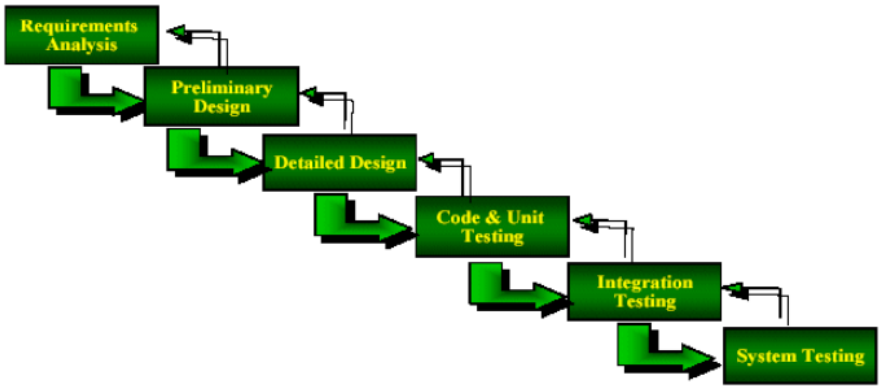
Na slici 11 prikazana je varijanta modela vodopada u kojoj je dozvoljeno vraćanje na neku prethodnu fazu, u slučaju da je došlo do greške u nekoj od faza. Takvi prelazi mogu biti između susednih faza (iz faze *dizajna* se može vratiti na prethodnu fazu *analize*), ali iz nekih faza (*testiranje* i *prihvatanje*) moguće je vratiti

se u tačno jednu, nesusednu fazu (*dizajn* i *zahtevi*, redom).

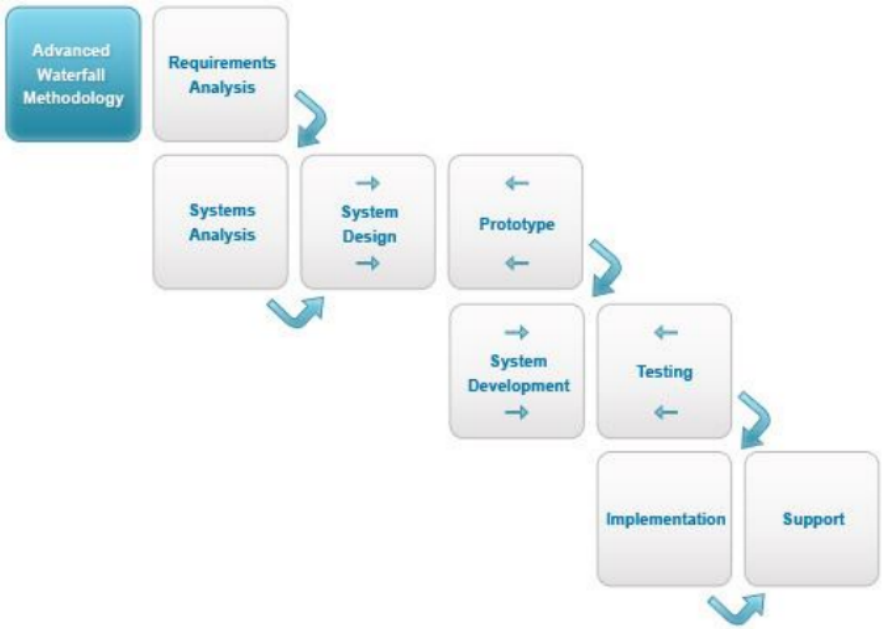
Na slikama 12, 13, i 14 dati su još neki primeri naprednijih modela vodopada. Ostavljamo čitaocu da ih protumači i pronađe sličnosti i razlike za već obrađenim modelima.



Slika 12: Varijanta modela vodopada – primer 4.



Slika 13: Varijanta modela vodopada – primer 5.



Slika 14: Varijanta modela vodopada – primer 6.

Literatura

[1] Avison David, Fitzgerald Guy. *Information Systems Development: Methodologies, Techniques and Tools, 4th Edition*. McGraw–Hill Education. 2006.

[2] Whitten Jeffrey, Bentley Lonnie, Dittman Kevin. *Systems Analysis and Design Methods, 5th Edition*. Irwin/McGraw–Hill Education. 2001.

[3] Olivé Antoni. *Conceptual Modeling of Information Systems*. Springer Science & Business Media. 2007.

[4] Buckingham Richard. *Information Systems Education: Recommendations and Implementation*. Cambridge University Press. 1987.

[5] Dennis Alan, Wixom Barbara, Roth Robert. *Systems Analysis and Design*. John Wiley & Sons. 2014.

[6] Langer Arthur. *Analysis and Design of Information Systems*. Springer Science & Business Media. 2007.

[7] Wasson Charles. *System Analysis, Design, and Development: Concepts, Principles, and Practices*. John Wiley & Sons. 2005.

[8] Davis William, Yen David. *The Information System Consultant's Handbook: Systems Analysis and Design*. CRC Press. 1998.

[9] Bennett Simon, McRobb Steve, Farmer Ray. *Object Oriented Systems Analysis and Design (Using UML)*. McGraw–Hill Education. 2002.

[10] Graovac Jelena. *Skripta iz predmeta Projektovanje baza podataka*. 2016.

[11] Fortune Joyce, Peters Geoff. *Information Systems: Achieving Success by Avoiding Failure*. John Wiley & Sons. 2005.

[12] Solomon David. *An Introduction to Computer Security: The NIST Handbook*. National Institute of Standards and Technology. 2010.

[13] Andress Jason. *The Basics of Information Security*. Syngress. 2011.

[14] Solomon David. *Elements of Computer Security*. Springer Science & Business Media. 2010.

[15] Smith James, Nair Ravi. *The architecture of virtual machines*. IEEE. 2005.

[16] Mell Peter. Grance Timothy. *The NIST Definition of Cloud Computing*. National Institute of Standards and Technology. 2011.

[17] Hassan Qusay. *Demystifying Cloud Computing*. Faculty of Computers and Information, Mansoura University, Egypt. 2011.

[18] Pernici Barbara. *Mobile Information Systems: Infrastructure and Design for Adaptivity and Flexibility*. Springer Science & Business Media. 2006.

[19] Kumar Vijay. *Mobile Database Systems*. John Wiley & Sons. 2006.