# 1. METODOLOGIJA INFORMACIJSKIH I PROGRAMSKIH SUSTAVA

Metodologija je opcenito znanost o metodama I primjeni metoda.

**Metodologija informacijskih/programskih sustava:** je znanstvena disciplina o pravilima, pristupima,procesima, metodama, tehnikama i sredstvima razvoja,primjene i održavanja informacijskih i programskihsustava.

**Metodika je:** ureden skup nacela, pristupa,pravila, cinjenica, obrazaca, metoda I tehnika rješavanja nekog zadatka.

**Pristup (paradigma) je**: skup pocetnih pretpostavki o objektu projektiranja i skup opcih nacela, koja proizlaze iz pojedinih znanstvenih teorija ili iskustva.

**Pristupi odreduju**:

- svrhu,
- ulogu,
- strukturu,
- ponašanje,
- nacin razvoja ili korištenja informacijskog ili programskog sustava, njegov odnos s okolinom

**Metoda je**: planski postupak za postignuce zadanog cilja na nekomprakticnom ili teoretskom podrucju.

**Tehnika je:** skup prakticnih postupaka i vještina primjene zadanemetode i obavljanja posla u konkretnoj situaciji.

**Modeliranje je:** razvoj informacijskog ili programskog sustava.

**Metamodel je:** skup svih koncepata i nacina njihove primjene, u okviru neke metode ili tehnike modeliranja. To je model svih mogucih modela koje možemo izraditi uz pomocu koncepata neke metode i tehnike.

**Vrste modela:**

- Slikovni - slikovna predodžba predmeta, smanjenih, stvarnih ili povecanih dimenzija
- Analogni - neka svojstva originala u istom ili drugom fizickom mediju
- Matematicki (analiticki) – u deterministickima matematicke ili logicke relacije prikazuju veze medu pojavama i dogadajima, a nedeterministicki opisuju diskretne dogadaje i dogadaje cija je vjerojatnost pojavljivanja statisticki odredena

- Konceptualni – opis kvalitativnih aspekata podrucja od interesa ili dogovorenih koncepata (poznata simbolika, sintaksa i semantika)

**Upotrebna svrha modela:**

- Demonstracijski modeli - prikazuju ponašanje i/ili funkciju sustava ili njegovih dijelova
- Eksperimentalni modeli - služe za proucavanje i provjeru statickih i dinamickih svojstava
- Modeli odlucivanja - prikazuju stanja modela u trenucima donošenja odluka

**Mjesto modela:**

- Unutarnji (interni) modeli su ukljuceni u sustav i dio su njegove strukture
- Vanjski (eksterni) modeli su izvan strukture sustava

**Inženjerske discipline:**
Inženjerstvo sustava (System Engineering)
Informacijsko inženjerstvo (Information Engineering)
Programsko inženjerstvo (engl. Software Engineering)

## SOFTVERSKA KRIZA

**Softverska kriza je:** nesposobnost da softverska industrija proizvede dovoljne kolicine kvalitetnih proizvoda, na vrijeme i po prihvatljivim cijenama.

**Koji su uzroci softverske krize?**

- Složenost zadatka
- Tehnologija
- Ljudi
- Metodologija

**Dimenzije razvoja softvera:**
širina – horizontalna multidisciplinarnost i slojevitost svakog zadatka
visina – vertikalna znanja problemske domene (proizvodnja, financije, uprava, zdravstvo ...)
dubina – razine apstrakcije

# 2. PRISTUPI IZGRADNJI INFORMACIJSKIH I PROGRAMSKIH SUSTAVA

**Kriteriji razvrstavanja pristupa**

1. Kakva je ULOGA INFORMACIJSKOG SUSTAVA? (preslikavanje organizacije, upravljanje organizacijom)
2. Kakva je NAMJENA SUSTAVA? (obrada transakcija, podrška u odlucivanju, podrška uredskog rada, rudarenje podataka ..)
3. Kakva je USKLADENOST SA ZAHTJEVIMA ORGANIZACIJE I KORISNIKA? (tehnicki pristup, planski pristup, društveno-tehnicki pristup)
4. Što je OBJEKT MODELIRANJA, postojeci sustav, ili vizija buduceg sustava? (subjektivisticki pristup, objektivisticki pristup)
5. Kakva je PRILAGODLJIVOST STANJU I PROMJENAMA OBJEKTNOG SUSTAVA? (strukturni pristup, situacijski pristup)
6. Kako izgleda RAZVOJNI CIKLUS? (fazni pristup, inkrementalni pristup)
7. Što je OSNOVNI MODEL? (funkcijski pristup, podatkovni pristup, funkcijsko-podatkovni pristup, objektni pristup)
8. Kakav je NACIN IZRADE I PRIMJENE MODELA? (dokumentacijski pristup, CASE pristuop, prototipni pristup, ponovno iskoristivi objekti)
9. Kakva je GRADA I TEHNICKA OSNOVA CILJNOG SUSTAVA? (klijent/server, Web/Internet ...)

S obzirom na mjesto i ulogu informacijskog sustava u organizacijskom sustavu, razlikuju se dva pogleda na informacijske sustave:

- preslikavanje organizacije
- upravljanje organizacijom

Razlikuju se:

1. Tehnicki (tehnicisticki) pristup (izraduju se modeli procesa i podataka)

Usmjeren je povecanju ucinkovitosti, na temelju kompjutorske podrške I automatizacije pojedinih operacija

2. Planski pristup (sustavni i holisticki)

Ocituje se u izradi strategijskog plana informacijskog sustava, koji odražava poglede višeg poslovodstva i daje smjernice za jedinstvena tehnološka rješenja.

3. Društveno-tehnicki pristup (teška formalizacija)

U središtu interesa su pojedinci, njihov sustav vrijednosti i potreba, odnosi medu pojedincima, te odnosi pojedinca i organizacije. Korisnici informacijskog sustava aktivno sudjeluju u njegovom razvoju.

**Što je OBJEKT MODELIRANJA?**

1. **Subjektivisticki pristup**

Subjektivisticki pristup zanemaruje važnost modeliranja sadašnjeg sustava. Prema ovom pristupu, razvoj buduceg programskog sustava odmah zapocinje prikupljanjem, analiziranjem i specificiranjem zahtjeva korisnika prema buducem programskom sustavu, te izradom njegovog modela.

2. **Objektivisticki pristup**

Objektivisticki pristup polazi od objektivne stvarnosti, tj. modela sadašnjeg objektnog sustava (ukljucujuci model sadašnjeg informacijskog i korisnickog sustava).

**Ukratko:**
Subjektivisticki pristup je orijentiran oblikovanju novog sustava (dizajnerski pristup), a objektivisticki je orijentiran analizi postojeceg (analiticki pristup).

**Kakva je PRILAGODLJIVOST STANJU I PROMJENAMA OBJEKTNOG SUSTAVA?**
Sustavni i situacijski pristup

**Kako izgleda RAZVOJNI CIKLUS?**

1. **Fazni pristup**

Fazni pristup pretpostavlja da se svaka faza razvojnog ciklusa u jednom razvojnom poduhvatu prolazi samo jednom.
U svakoj se fazi potpuno dovršavaju i provjeravaju svi pripadajuci izlazni rezultati. Na kraju svake faze razvojnog ciklusa provodi se formalna provjera rezultata (verifikacija) i vrednovanje od strane korisnika (validacija). Ako je ocjena provjere ili vrednovanja negativna, faza ili dio faze se ponavlja.

2. **Djelomicno inkrementalni pristup**

3. **Inkrementalni pristup**

Inkrementalni pristup (razvoj u koracima) tj. pojedine faze razvojnog ciklusa izvode više puta. Svako ponovno izvodenje pojedine faze znaci dogradnju izlaznog modela, odnosno njegovo poboljšanje za odredeni inkrement (prirast, prinos). Svakim ponovnim prolazom kroz neku od ranijih raza razvojnog ciklusa nastaje nova verzija modela, koja je proširena ili poboljšana u odnosu na staru. Isto se odnosi i ponovni prolaz kroz fazu primjene, u kojoj se primjenjuje proširena ili poboljšana verzija informacijskog podsustava.
Nakon završetka jedne faze razvojnog ciklusa inkrementa prelazi se na sljedecu.

**Kojim se osnovnim modelom prikazuje predmet modeliranja?**

1. funkcijski pristup, tj. pristup orijentiran procesima,
2. podatkovni pristup, tj. pristup orijentiran podacima,
3. funkcijsko-podatkovni pristup i
4. objektni pristup, tj. pristup orijentiran objektima.

Osnovne informacije o objektnom sustavu su podaci i procesi objektnog sustava.

**Funkcijski pristup**

Prema funkcijskom pristupu, osnovno je specificiranje funkcionalnosti sustava.

**Modeli procesa** su formalizirani opisi hijerarhijskih struktura procesa, unutarnje logike procesa, medusobnih odnosa procesa, dogadaja u objektnim sustavima, te odnosa procesa i okolice.

**Modeli rašclanjivanja procesa** predstavljaju sastavne i klasifikacijske strukture složenih procesa, a modeli tokova podataka prikazuju kretanje I pretvorbe podataka kroz informacijski, odnosno objektni sustav.

**Funkcionalni modeli** ponašanja opisuju upravljacke koncepte procesa objektnog sustava i dogadaje u ovom sustavu.

**Osnovni koncepti modela procesa su:**

1. funkcionalne komponente (funkcije, procesi, potprocesi, aktivnosti, operacije, moduli)
2. tokovi podataka i njihov sadržaj,
3. izvori i odredišta podataka,
4. spremišta podataka,
5. dogadaji, koji pokrecu i prekidaju procese

**Podatkovni pristup**

Podatkovni pristup pretpostavlja da je model podataka osnovni koji se oblikuje tijekom razvoja informacijskog i programskog sustava.

**Model podataka** je stabilniji od modela procesa. Naime, struktura procesa I njihova unutarnja logika je više izložena promjenama od structure podataka.

**Osnovni koncepti ovakvog formaliziranog modela podataka su:**

1. skup koncepata za opis strukture podataka,
2. skup ogranicenja za ocuvanje integriteta podataka (skup pravila koja opisuju dozvoljena stanja sustava i dozvoljeni prijelaz iz stanja u stanje)
3. skup operatora kojima je moguce opisati promjenu stanja podataka
4. sustava (djelovanja ulaza na izlaz)

**Funkcijsko-podatkovni pristup**

Funkcijsko - podatkovni pristup, prema kojem su modeli procesa i modeli podataka podjednako važni i medusobno povezani.
**Modeliranje procesa i podataka cini nedjeljivu cjelinu.**

**Objektni pristup**

Objektni pristup pretpostavlja izradu modela objekata, koji u semantickom smislu objedinjavaju modele podataka i modele procesa. Ovi modeli predstavljaju objekte, metode posluživanja objekata i poruke koje objekti razmjenjuju s okolicom ili medusobno. Ovom vrstom modela mogu se prikazati staticka svojstva sustava (funkcija i struktura), ali i dinamicka svojstva, odnosno ponašanje sustava koji djeluje.

**Osnovni koncepti modela objekata su:**

1. tipovi objekata,
2. klasifikacijske i sastavne strukture objekata,
3. atributi, veze i njihova ogranicenja,
4. dogadaji i stanja,
5. operacije na objektima (metode posluživanja),
6. nasljedivanje, ucahurivanje, polimorfizam, preklapanje
7. pocetni i konacni uvjeti stanja,
8. prijelazi iz stanja u stanje,
9. spojišta poruka

**Kakav je NACIN IZRADE I PRIMJENE MODELA?**

1. Dokumentacijski pristup
2. CASE pristup (Computer Aided Software Engineering)
3. Prototipni pristup

**CASE je:** program za projektiranje I održavanje informacijskih sustava prema odredenoj metodologiji i namijenjen primjeni u odredenoj radnoj okolini.

**CASE alate dijelimo na:**

- Upper-CASE: podržavaju rane faze razvoja softvera, nezavisni su od implementacijskog okružja
- Lower-CASE: podržavaju faze dizajna i implementacije, te su usko vezani za implementacijsko okružje
- Integrated-CASE (I-CASE): kombiniraju Lower-CASE i Upper-CASE.

Rjecnik informacija pomagala CASE je **aktivna baza informacija.**

**Projektiranje je:** pretvorba skupa ulaznih informacija (ulaznih modela i specifikacija) u izlazne.

**Svrha procesa razvoja je:** isporuka konacnog proizvoda (cjelovitog modela ili primjenjivog proizvoda), pa izlazne rezultate pojedinih aktivnosti možemo smatrati medurezultatima.

**Više stupnjeva formalizma (Daniel Jackson)**

- formalno,
- polu-formalno,
- neformalno

Prednost: mogucnost automatske primjene i provjere
Problem: Formalne metode zahtijevaju matematicke gurue, cjena ucenja i primjene je visoka

Grada sustava – Temeljna pitanja

U kojoj je mjeri metodologija razvoja sustava ovisna o ciljnoj gradi sustava i tehnickoj osnovi?

Metodologija projektiranja informacijskih sustava nije tehnološki neovisna.

Kako grada sustava utjece na projektiranje novih informacijskih sustava i kada ulazi u razmatranje?

Grada ciljnog sustava bitno utjece na pristup projektiranju, te izbor razvojnog ciklusa, procesa razvoja, metoda, tehnika i pomagala

Kako preoblikovati postojece sustave?

Trend su otvoreni sustavi, široki raspon platformi strojne opreme, komunikacijsko povezivanje

1. Razina podataka (serveri baza podataka - DBMS)
2. Razina poslovne logike (aplikativni server - Poslovna pravila ugradena u programe)
3. Prezentacijska razina (korisnicka racunala - preglednik ili stolne aplikacije)

## 3. RAZVOJNI CIKLUS INFORMACIJSKOG SUSTAVA

**Razvojni ciklus informacijskog sustava je:** vremensko razdoblje izmedu donošenja formalne odluke o razvoju i formalne isporuke ili formalnog prekida razvoja

**Rezultat razvoja je ciljni proizvod.**

**Strategijsko planiranje informacijskog sustava**

Izraduje se grubi konceptualni model postojecih procesa i podataka organizacijskog sustava, te model postojeceg informacijskog sustava.

Modela poslovnog sustava:
- model ciljeva,
- model kljucnih cimbenika uspjeha,
- model kriticnih pretpostavki,
- model problema,
- model potreba za informacijama ...

Na temelju analize ovih modela odreduje se:
- gruba struktura informacijskog sustava (podjela na podsustave)
- prioriteti i redoslijed realizacije podsustava
- okvirni troškovi i izvodljivost

Rezultat je strategijski plan informacijskog sustava, s planskim obzorom 3 do 5 godina.
Utvrdivanje izvodljivosti poduhvata

**Cilj je:** utvrdivanje granica (opsega) i izvodljivosti planiranog poduhvata razvoja s tehnickog, tehnološkog, organizacijskog, ekonomskog i drugih gledišta.

Granice poduhvata se utvrduju na grubom konceptualnom modelu podataka i procesa

Rezultat je:
- studija izvodljivosti
- detaljni projektni zadatak buduceg poduhvata razvoja
- eventualno natjecajna dokumentacija

**Analiza i specifikacija zahtjeva**

U ovoj fazi se profinjava model podataka i procesa, odnosno objektni model. Detaljno se analiziraju i specificiraju zahtjevi prema buducem sustavu, koji se odnose na:
- podatkovne sadržaje
- funkcionalnost, i tehnologiju rada
- sucelje, odziv, performanse i ostale oblike ponašanja
- ostale nefunkcionalne zahtjeve

**Logicko modeliranje**

Ova faza obuhvaca izradu detaljnog logickog modela buduceg sustava, koji opisuje što on mora biti sto znaci:

- dekompoziciju procesa, dijagrame toka podataka i opise
- unutarnje logike elementarnih procesa
- model entiteti-veze, njegovu pretvorbu u relacijski model I relacijsku analizu.

**Fizicko modeliranje**

U ovoj se fazi izraduje fizicki model baze podataka (tablice, kljucevi, indeksi)

- Definiraju se programski moduli i njihova distribucija
- Konstruiraju se programi (izrada programskog koda)
- Detaljno se razraduju uloge korisnika i pogledi
- Detaljno se razraduje sustav autorizacije korisnika I sigurnosti
- Detaljno se oblikuje sucelje
- Testiraju se pojedini moduli
- Moduli se integriraju i testira se cjelina
- Model se dokumentira
- Pojedini dijelovi i sustav u cjelini se validiraju s korisnicima

**Isporuka i primjena**

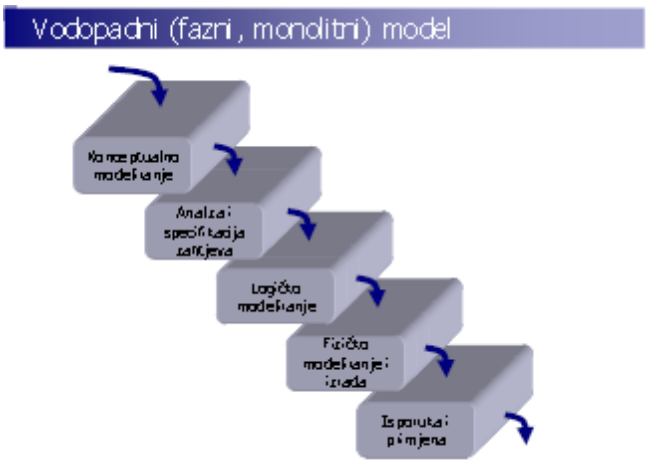
Instalira se oprema i programi

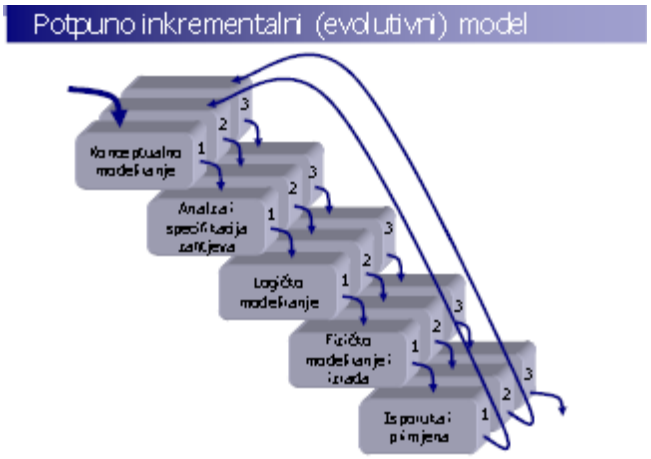
- Osposobljavaju se korisnici
- Obavlja se konverzija i integracija podataka
- Testira se prihvatljivost sustava tijekom razdoblja probnog rada.
- Po potrebi se obavljaju odredene korektivne aktivnosti
- Daje se ocjena prihvatljivosti

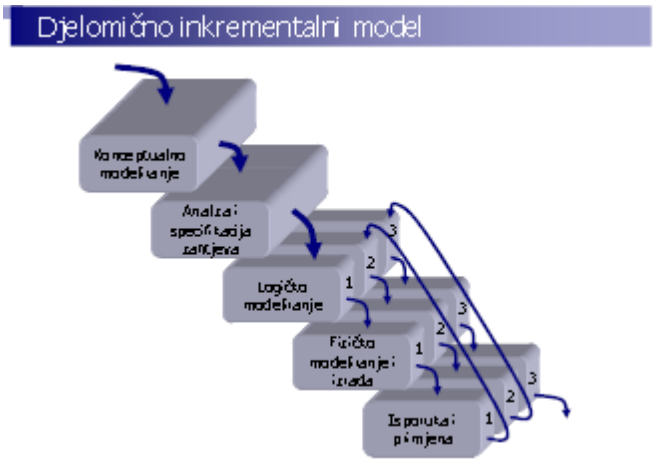
**Održavanje i poboljšavanje**

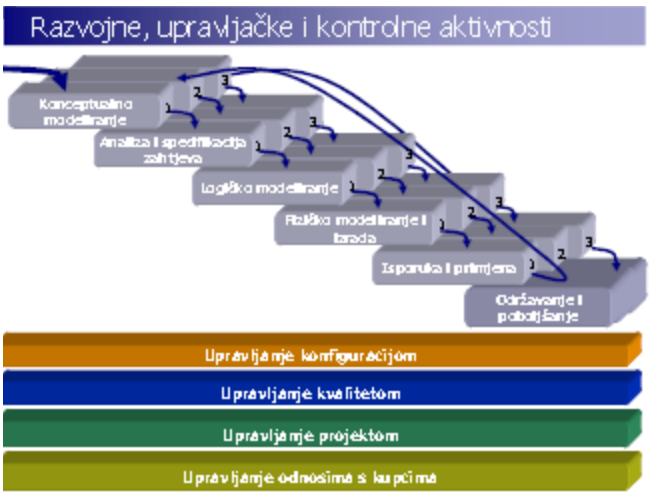
Pružaju se razliciti oblici podrške korisnicima

- Otklanjaju se uocene greške u funkciji ili ponašanju
- Sustav se prilagodava novim izdanjima operacijskog sustava, poboljšanjima i proširenjima opreme, novim komunikacijskim mogucnostima i sl.
- Sustav se prilagodava promjenama poslovnih pravila, poslovne tehnologije, zakona i sl.
- Proširuje se funkcionalnost sustava, u skladu sa zahtjevima korisnika




## 4. PROCES RAZVOJA INFORMACIJSKOG SUSTAVA

**Proces razvoja informacijskog sustava je:** skup medusobno povezanih aktivnosti, koje se izvode tijekom razvojnog ciklusa.

Aktivnosti se, unutar procesa razvoja, izvode vremenski slijedno (sljedeca zapocinje nakon završetka prethodne), vremenski usporedno, ili ciklicki (nakon završetka neke od sljedecih aktivnosti, vraca se na prethodnu).

**Predložak procesa razvoja opisuje:**

- rašclanjivanje procesa i aktivnosti na aktivnosti niže razine,
- veze i slijed izvodenja aktivnosti,
- uvjete pocetka i završetka aktivnosti,
- ulazne i izlazne podatke i informacije, potrebne za izvodenje svake aktivnosti,
- metode, tehnike, pomagala i sredstva, koja se koriste tijekom izvodenja aktivnosti

**Svrha aktivnosti upravljanja je upravljanje drugim vrstama aktivnosti.**

**Aktivnosti ucinka**

Aktivnosti ucinka su aktivnosti prikupljanja i analize informacija, oblikovanja modela, programiranja, izrade dokumentacije i sl.

**Aktivnosti provjere**
Svrha aktivnosti provjere je provjeravanje i vrednovanje rezultata aktivnosti
Dvije osnovne vrste aktivnosti provjere su:

- verificiranje (formalno provjeravanje uskladenosti s ulaznim specifikacijama)
- validiranje (vrednovanje od strane korisnika)

Model podataka je stabilniji od modela procesa. Naime, struktura procesa i njihova unutarnja logika je više izložena promjenama od strukture podataka.

**Definicija i osnovni koncepti**

**Model podataka je:** apstraktni prikaz skupova podataka, njihovih medusobnih veza i ogranicenja, te nacina manipulacije podacima.

**Osnovni koncepti modela podataka su:**

- skup koncepata za opis strukture podataka (podaci koji opisuju objekte i pojave)
- skup ogranicenja za ocuvanje integriteta podataka (skup pravila koja opisuju dozvoljena stanja sustava i dozvoljeni prijelaz iz stanja u stanje)
- skup operatora kojima je moguce opisati promjenu stanja
- podataka sustava (mehanizmi promjene vrijednosti podataka)

**Staticki i dinamicki model**

Cjeloviti model sustava prikazuje staticka i dinamicka svojstva sustava.

**Staticki modeli** prikazuju strukturu i stanja podataka sustava.
**Dinamicki modeli** opisuju promjene stanja sustava, odnosno njegovih podatkovnih objekata i veza medu njima.

**Poznatije metode modeliranja podataka:**

- Model entiteti-veze (najpoznatiji logicki staticki model)
- Model životnog ciklusa entiteta (najpoznatiji logicki dinamicki model)
- Model binarnih veza
- Mjehuricasti model podataka
- Semanticki model
- Relacijski model (najpoznatiji fizicki staticki model)
- Mrežni model
- Hijerarhijski model

**Scenarij modeliranja podataka**

1. Izrada modela entiteti-veze i modela životnog ciklusa entiteta (konceptualno i logicko modeliranje)
2. Pretvorba u relacijski model i relacijska analiza (pretvorba u fizicki model)

**Model entiteti-veze**

Prikazuje podatke sustava u obliku tipova entiteta, koji su opisani tipovima atributa i povezani tipovima veza.
Modeli entiteti-veze su osnovna i najcešce primjenjivana vrsta konceptualnih i logickih modela podataka.

**ENTITET**

**Entitet je:** nedvosmisleno prepoznatljiv koncept, predmet, dogadaj ili bice o kojemu se u informacijskom sustavu prikupljaju i pamte podaci.

**ATRIBUT**

**Atribut (obilježje, svojstvo) je:** podatak koji opisuje entitetili omogucava njegovo prepoznavanje.

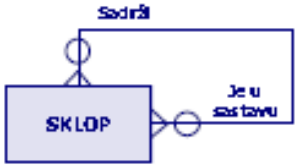
**Atributi entiteta razvrstavaju se na:**

- identifikacijske (omogucavaju prepoznavanje pojava entiteta)
- opisne (izražavaju kvalitetu entiteta, stanje i položaj u klasifikacijskoj strukturi, kvantificiraju svojstvo entiteta ili odreduje odnos prema drugim pojavama entiteta)
- izvedene (vrijednosti im se izvode iz vrijednosti drugih tipova atributa).

**U jednom vremenskom trenutku jedna pojava entiteta možeimati samo jednu vrijednost atributa za svaki tip atributa, što omogucava pracenje stanja pojava entiteta (dinamika sustava).**

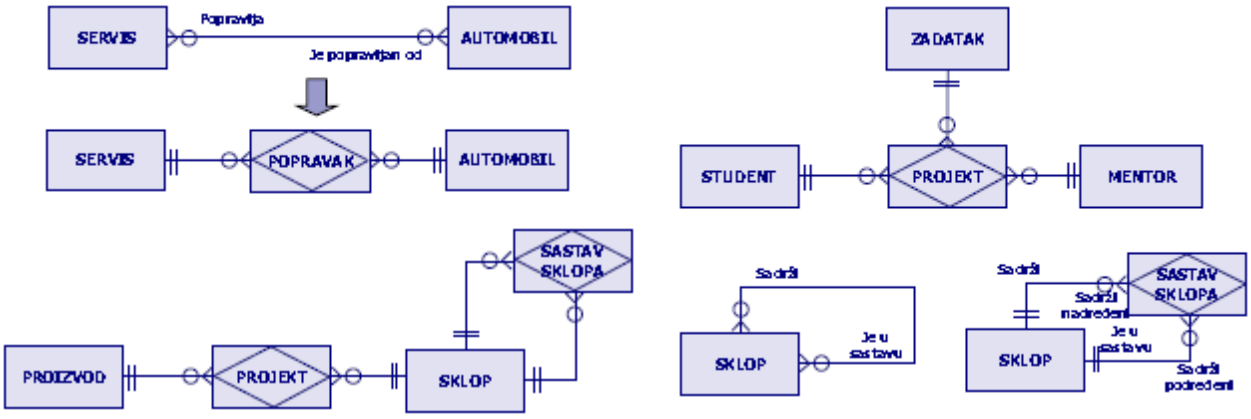
**VEZA**

Veza povezuje pojave dva tipa entiteta ili razlicite pojave istog tipa entiteta.
**Tip veze** se predocava spojnom crtom.
**Red veze** odreduje koliko tipova entiteta sudjeluje u vezi (unarna,binarna n-arna).
Unarna (rekurzivna, refleksivna) veza povezuje razlicite dvije pojave istog tipa entiteta.
Najcešce pokazuje strukturu, odnosno da pojava entiteta sadrži pojave istog tipa, ali niže razine složenosti (sastavnica, sklop - podsklop).




Asocijativni tip entiteta nastaje od:

- veze kardinalnosti više:više (M:M),
- veze koja sadrži atribute,
- veze tri ili više tipova entiteta (n-arna veza).



# 5. MODELI PROCESA I FUNKCIONALNI (PROCESNI) PRISTUP

**Modeli procesa**

**Modeli procesa** su formalizirani opisi hijerarhijskih struktura procesa, unutarnje logike procesa, medusobnih odnosa procesa, dogadaja u objektnim sustavima, te odnosa procesa i okolice.

**Modeli rašclanjivanja procesa** predstavljaju sastavne i klasifikacijske strukture složenih procesa, a modeli tokova podataka prikazuju kretanje i pretvorbe podataka kroz informacijski, odnosno objektni sustav.

**Funkcionalni modeli** ponašanja opisuju upravljacke koncepte procesa objektnog sustava i dogadaje u ovom sustavu.

**Posebne vrste** modela opisuju unutarnju logiku elementarnih procesa.

**Osnovni koncepti modela procesa**

- funkcionalne komponente (funkcije, procesi, potprocesi, aktivnosti, operacije)
- tokovi podataka i njihov sadržaj,
- izvori i odredišta podataka,
- spremišta podataka,
- dogadaji, koji pokrecu i prekidaju procese

**Poznatije vrste modela procesa**

- Model funkcionalnog rašclanjivanja
- Model tokova podataka
- Funkcionalni modeli ponašanja (model tokova upravljackih podataka, model ovisnosti procesa)
- Modeli unutarnje logike procesa (proceduralni modeli unutarnje logike procesa i programskih modula, modeli odlucivanja, modeli ponašanja)
- Strukturni dizajn aplikacija

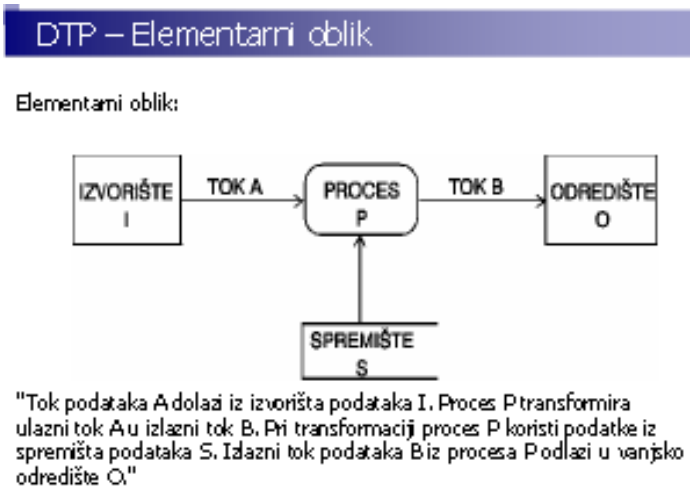
**Dijagram funkcionalnog rašclanjivanja (dekompozicije) je:** hijerarhijski dijagram koji predocava strukturu (kompoziciju) neke funkcionalne komponente, koja se sastoji od složenih ili elementarnih komponenata niže razine složenosti.

Funkcionalne komponente su: funkcija, proces, aktivnost, operacija

**Razlika procesa i funkcije:**
Za razliku od procesa, funkcija nema odreden pocetni i završni dogadaj (ne pokrece se i ne prekida se), proces se ponavlja (postoje vremenski odsjecci kad se ne izvodi), a funkcija uvijek postoji, kod funkcije je naglasak na statickoj komponenti (strukturi dijelova, logicki povezanim procesima), a kod procesa na dinamickoj komponenti (strukturi veza).

**Dijagram toka podataka**

**Dijagram toka podataka je:** mrežni dijagram koji prikazuje tokove podataka kroz sustav, njihova izvorišta i odredišta, te procese koji tokove podataka transformiraju. Opisuje staticku funkcionalnost sustava (opisuje funkciju, ali ne i ponašanje).




**Osnovni koncepti dijagrama toka podataka:**
- tokovi podataka,
- procesi koji transformiraju ulazne tokove podataka u izlazne,
- spremišta podataka,
- vanjska izvorišta i odredišta podataka,
- spojišta konteksta.

**Tok podataka je** skup ulaznih podataka u proces i/ili izlaznih podataka iz procesa.
**Proces je** skup aktivnosti kojima se skup ulaznih podataka transformira u skup izlaznih podataka.
**Spremište podataka je** koncept koji reprezentira pohranjivanje izlaznih podataka procesa koji su na raspolaganju drugim procesima.
**Vanjska izvorišta i odredišta** su procesi ili sustavi koji se nalaze izvan podrucja analize. Njihova struktura, funkcija i ponašanje nas ne zanima, nego nas zanimaju samo tokovi podataka koje iz njih izlaze ili ulaze I povezuju ih s procesima prikazanim na dijagramu toka podataka.
**Spojište konteksta je** mjesto na kojem ulazni tok ulazi u proces ili izlazni izlazi iz njega.

**Modeli unutarnje logike procesa**

Procesi najniže razine na dijagramima funkcionalnog raščlanjivanja i dijagramima toka podataka su **elementarni (primitivni).**

Da bi konceptualni procesa bio potpun, svakom elementarnom procesu mora biti pridružen odgovarajuci model unutarnje logike.

**Model unutarnje logike** opisuje kako se proces izvodi, te kako se pokrece i prekida njegovo izvodenje.

**Najpoznatiji modeli unutarnje logike procesa su:**

- proceduralni modeli (dijagrami akcija, strukturne specifikacije i sl.),
- modeli odlucivanja (stabla i tabele odlucivanja)
- dijagrami prijelaza stanja

**Proceduralni modeli** opisuju kako se ostvaruje svrha elementarnih procesa objektnog sustava. Pritom se koriste koncepti strukturnog programiranja.

**Modeli odlucivanja** koriste koncept izbora (selekcije) tijeka procesa. Tijek procesa ovisni o ispunjenosti ili neispunjenosti uvjeta selekcije, odnosno o stanju sustava predstavljenom skupom podataka.

Dijagrami prijelaza stanja opisuju stanja objektnog sustava, dogadaje u objektnom sustavu koji su pobude procesa i odzive na ove pobude.

## 6. OSNOVE OBJEKTNOG PRISTUPA I MODELI OBJEKATA

Objektni pristup pretpostavlja izradu modela objekata, koji u semantickom smislu objedinjavaju modele podataka i modele procesa.
Ovom vrstom modela mogu se prikazati staticka svojstva sustava (funkcija i struktura), ali i dinamicka svojstva, odnosno ponašanje sustava koji djeluje.

Osnovni koncepti
1. Klase, objekti, instance
2. Ucahurivanje i sakrivanje informacija
3. Poruke - komuniciranje objekata
4. Nasljedivanje i klasifikacijske strukture
5. Višeoblicnost
6. Dinamicko vezivanje

7. Kontrolne strukture i podrucja (arrays)
8. Apstraktne klase
9. Sucelja
10. Iznimke

**Objektni pristup razvoju**

Umjesto raščlanjivanja funkcionalnosti naglasak je na modeliranju problemske domene, kao skupa objekata u interakciji. Analiza i dizajn se izvode interaktivno i medusobno su povezani:

- Klasifikacija (klasa / podklasa) se rješava u fazi analize
- Integralni pogled na funkcionalnost i podatke i njihovo pakiranje je prirodan i prilagodljiv promjenama
- Dizajn i implementacija se razlikuju samo po razini apstrakcije, koncepti i jezici su isti

**Klasa**

Predstavlja skup objekata koji posjeduju ista strukturna svojstva, ponašanje ili druga obilježja

**Klasa sadrži:**

- Atribute (varijable instance i klasne varijable)
- Mehanizme za kreiranje i destrukciju (constructors, destructors)
- Operacije, ukljucujuci mehanizme za kreiranje i destrukciju (constructors, destructors)
- Dozvoljena stanja i prijelaze stanja
- Poruke
- Strukturne veze s drugim klasama

**Objekt u informacijskom sustavu**

- Može se adresirati
- Može imati graficku reprezentaciju
- Zauzima memorijski prostor
- Ima stanje koje je odredeno skupom podataka (vrijednosti atributa) u odredenom trenutku
- Ima pridružene mehanizme za operacije nad podacima (procedure, funkcije..)

**Invarianti**

Invariants – stanja koja su dozvoljena za odredenu klasu

**“Design by contract”**

Objekt-poslužitelj jamci objektima-klijentima da ce se uvijek ocekivano ponašati, odnosno da ce uvijek biti u nekom od dozvoljenih stanja i da ce uvijek ocekivano odgovarati na ispravno adresiranu poruku koja sadrži sve potrebne parametre.

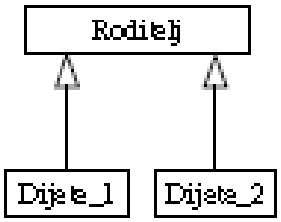
**Princip "otvoreno/zatvoreno"**
Klasa je u svakom trenutku istovremeno:

- zatvorena, što znaci da su njezini potencijalni korisnici sigurni da se sucelje klase nece promijeniti
- otvorena, što znaci da su dozvoljena proširenja stvaranjem podklasa

**Nasljedivanje**

Klase–djeca nasljeduju astribute i metode od klasa roditelja

**Superklasa (roditelj, nadklasa, supertip) je:** klasa cije instance sadrže atribute i metode koji su zajednicki za jednu ili više podklasa.
**Subklasa (dijete, podklasa) je**: klasa cije instance nasljeduju zajednicke atribute i metode posluživanja od nadklase, a sadrže i atribute i metode svojstvene podklasi.




**Koristi od nasljedivanja**

Sustavi se izgraduju iz ponovno iskoristivih dijelova (reusability, programming by extension)

**Primjena koncepata nasljedivanja**
Pristup atributima i metodama klase-roditelja
**Višeoblicnost – polimorfizam je:** sposobnost da se iza istog sucelja sakriju razlicite implementacije

**Polimorfno referenciranje je**: svojstvo da se tijekom vremena referenciraju razlicite instance:

- iste klase – staticka višeoblicnost
- razlicitih klasa, najcešce instance razlicitih podklasaiste klase – dinamicka višeoblicnost

**Virtual** kljucna rijec za definiranje polimorfnih funkcija

**Dinamicko vezivanje**
Vezivanje poziva procedure s kodom koji ce se izvršiti kao rezultat poziva:

- staticko vezivanje – obavlja ga compiler ili linker
- dinamicko vezivanje – obavlja se i vrijedi samo tijekom izvodenja

Problemi uvodenja i primjene OO pristupa

- Razliciti OO jezici teško medusobno komuniciraju
- Nedostaju standardi i iskustva
- Brze i dramaticne promjene OO tehnologije
- Puno programiranja
- Djelomicna implementacija u OO SUBP
- Stari programi (i programeri)

**Klasicni kriteriji kvalitete dizajna**

1. Što manja vanjska povezanost (coupling)

plus: povezanost se ostvaruje porukama
minus: nasljedivanje je oblik vanjske povezanosti

2. Što veca unutarnja povezanost (cohesion):

plus: klase su kohezivne po definiciji

3. Inkrementalni razvoj

plus: faze razvoja su povezane istim konceptima modeliranja, metodama, tehnikama...

**Booch - kriteriji kvalitete klasa**

1. Unutarnja povezanost (Cohesion)
2. Vanjska povezanost (Coupling)
3. Dovoljnost (Sufficiency)
4. Kompletnost (Completeness)
5. Primitivnost (Primitiveness)

**Unutarnja povezanost (Cohesion)**

Jesu li i kako su dijelovi klase (prije svega operacije) medusobno povezani
Nema nepovezanih dijelova (slucajna povezanost)
**Primjer:**
registracija automobila i upis u registar brodova upucuju na razlicite klase

**Vanjska povezanost (Coupling)**

Jaka vanjska povezanost vidi se po tome što je klasu teško neovisno razumjeti ili održavati
Jaka unutarnja i slaba vanjska povezanost su suprotni zahtjevi!

**Dovoljnost (Sufficiency)**

Klasa sadrži dovoljno elemenata za ucinkovito i smisleno korištenje i medudjelovanje
**Primjer:**
Sucelje je u stvarnom svijetu GUI, aktivni medusklop,pasivni poveznik i sl.

**Kompletnost (Completeness)**
Sucelje klase sadrži sve što je potrebno za smislenu komunikaciju
**– Dovoljnost (Sufficiency)** – sve što je nužno i dovoljno
**– Opcenitost** – mogucnost primjene u razlicitim situacijama, što vodi na mogucnost proširenja

**Primitivnost (Primitiveness)**
Klasa osigurava samo primitivne operacije, dok se ostale moraju ostvariti izvan klase
**Primjer:**
klasa Poligon sadrži koordinate vrhova, postoje operacije za njihovo dohvacanje, ali ne postoji operacija za racunanje površine, to mora korisnik sam napraviti

**Povezivanje s metrikom softvera**
Ponovna iskoristivost (Reusability)

- Može li se primijeniti u drugom kontekstu?

Složenost (Complexity)

- Da li je teško shvatiti i implementirati?

Primjenjivost (Applicability)

- Poznavanje implementacije

# 1. UML

**Pojam objekta**

Objekt je nedvosmisleno prepoznatljiva jedinka, predmet, entitet ili dogadaj, bilo stvarni ili apstraktni, sa dobro definiranom ulogom u problemskoj domeni
Objekt je moguce opisati skupom podataka i nacinom rukovanja tim podacima

**UML Dijagrami strukture**

- Dijagram klasa
- Dijagram paketa

**Strukturni dijagrami u UML 2.0**

1. Dijagrami klasa (class diagrams)
2. Objektni dijagrami (object diagrams)
3. Dijagrami komponenata (component diagrams)
4. Dijagrami kompozicijskih struktura (composite structure diagrams)
5. Dijagrami paketa (package diagrams)
6. Dijagrami razmještaja (deployment diagrams)

**UML Dijagrami klasa (Class Diagrams)**

Prikazuju:

- klase, atribute i operacije
- staticke veze klasa, ukljucujuci "obicne" asocijacije,generalizacije

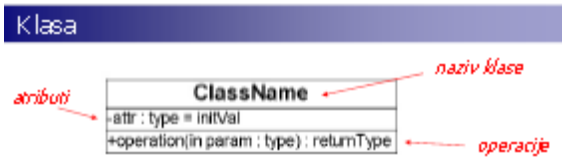
**Dijagrami klasa: Razine apstrakcije**

- Konceptualna - analiticki modeli poslovnih pravila
- Logicka (specifikacijska) - detaljni dizajn, prikaz sucelja objekata
- Fizicka (implementacijska) - kod, primjenjive komponente

**Apstrakcija je:** kljucni princip objektno-orijentirane tehnologije kojom se rješava složenost problema.

Klasa je apstrakcija jer:

- Naglašava relevantne osobine
- Sakriva druge osobine i njihovu implementaciju
- Objekt je instanca klase




**ATRIBUTI**

Vidljivost (visibility): + public, - private, # protected

**Atributi: Razine apstrakcije**

- Konceptualna - atributi koje objekt može imati i njihov opis
- Logicka (specifikacijska) - nacin kako se atributima pristupa i poslovna pravila
- Fizicka (implementacijska) - stvarni podaci objekta

**Operacija** – sucelje objekta prema okolini, koncept za promjenu stanja i izvršavanje zadatka objekta

**Metoda** – implementacija operacije, mehanizam promjene stanja, ista operacija može imati razlicite implementacije u hijerarhiji

**Operacije: Razine apstrakcije**

- Konceptualna - naznake i osnovni principi operacija – kreiraj, dohvati, prikaži promijeni_
- Logicka (specifikacijska) - detaljnija specifikacija operacija, posebno javnih
- Fizicka (implementacijska) - sve metode isprogramirane

**Asocijacija je** strukturna veza kojom se odreduje da je objekt jedne klase povezan s objektom druge ili iste klase

Prikazuje:

- suradnju na razini klasa
- veze medu instancama klasa

**Asocijacije: Razine apstrakcije**

- Konceptualna - konceptualne veze
- Logicka (specifikacijska) - dozvole pristupa i promjene stanja
- Fizicka (implementacijska) - pokazatelji (pointeri)

**Usmjernost (Navigability)**

Asocijacije su:

- jednosmjerne (unidirectional)
- dvosmjerne (bidirectional)

Smjer poruka odreduje da li je asocijacija u jednom smjeru ili dvosmjerna (svojstvo navigability)

**Kardinalnost je** stupanj asocijacije

**Asocijacija: Vrste**

**Generalizacija** (konceptualna, logička, fizička razina)
Prikazuje nasljedivanje tako što klasa dijete nasljeduje svojstva klase roditelja

**Agregacija**
Agregacija prikazuje da se nešto sastoji od dijelova (part-of) veza

Agregacija je tranzitivna, asimetricna veza, a može biti i rekurzivna

**Kompozitna agregacija** (composite aggregation):
jaci oblik, svaki dio može u jednom trenutku biti clan samo jedne kompozicije (cjeline)

- ako se cjelina obriše, svi se dijelovi brišu s njom
- dio se može obrisati a da cjelina ostane

**Djeljiva agregacija (shared aggregation):**
slabiji oblik, cesto oznacava virtualno grupiranje

- jedan klasifikator koristi drugog ali ga ne sadrži i može ga dijeliti

**Višestruka i dinamicka klasifikacija**

1. Jednostruka (single) klasifikacija – objekt uvijek ima jedan tip, kojeg nasljeduje od supertipa
2. Višestruka (multiple) klasifikacija - objekt može imati više tipova
3. Staticka (static) ili konacna klasifikacija – objekt nikada ne mijenja tip
4. Dinamicka (dynamic) klasifikacija – objekt može promijeniti tip

**Osnovni stereotipovi**
Koncept stereotipa omogucava razvrstavanje klasa prema funkcijama
koje izvršavaju.

**Tri primarna stereotipa klasa:**

1. Granicna klasa nalazi se na granici izmedu našeg sustava i ostatka svijeta (npr. forme, izvješca, hardverska sucelja kao što su printer ili skeneri i sucelja ostalih sustava).
2. Granicna klasa dozvoljava sudioniku interakciju s sustavom. <<Interface object>>
3. Kontrolna (upravljacka) klasa je odgovorna za koordinaciju ponašanja ostalih klasa u skladu s poslovnim pravilima i nema drugu svrhu. <<Control object>>

4. Klasa entiteta ima za korisnika veliki znacaj, tako da se njezine informacije trajno pohranjuju i nakon terminiranja slucaja korištenja. <<Entity object>>

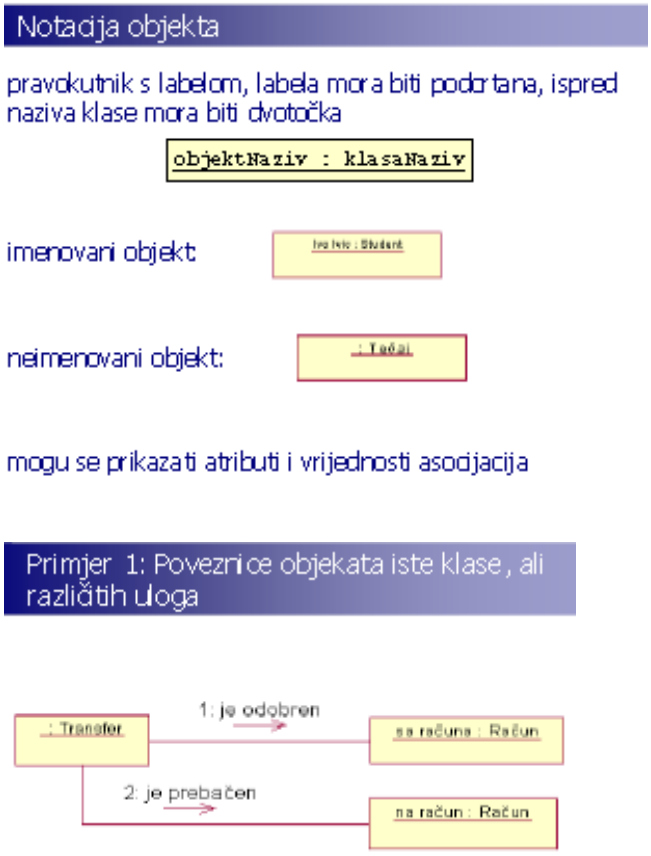
**Sucelje (Interface)**

**Sucelje je** specifikacija ponašanja koja implementira "suglasnost za suceljavanje", "ugovor o suradnji"

## 2. UML Dijagrami strukture

**Dijagram objekata**

Prikazuje objekte i poveznice objekata u odredenom vremenskom trenutku
Koristi se da bismo proucili strukture objekata ili kad nas zanima dinamicka struktura objekata (npr. GUI dizajn)



**Poruke - komuniciranje objekata**

**Poruka je** koncept kojim jedan objekt poziva (aktivira) jednu ili više metoda drugog objekta, s ciljem da dobije odredenu informaciju, promijeni stanje objekata ili drugu akciju.

Zahtjev (message request) sadrži naziv metode i parametre za izvršavanje

## UML Dijagram slucajeva korištenja (Use Case Diagram)

Model slucajeva korištenja omogucava opis:

- osnovnih funkcionalnih cjelina i njihovog ponašanja
- vanjskih ucesnika (uloga) i njihove interakcije sa sustavom
- slucajeva za testiranje, na razini sustava

Model sadrži:

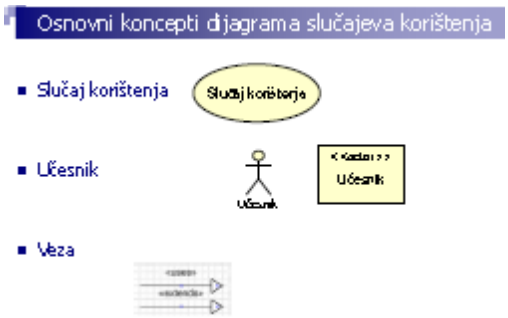
- Opise slucajeva korisnika
- Dijagrame slucajeva korištenja

**Svrha**

Najviša razina opisa funkcionalnosti sustava i odnosa s okolinom

**Opisi slucajeva korištenja**

- Skraceni – samo glavni scenarij
- Puni – svi scenariji, iznimke, greške



**Ucesnik (Actor)**

Ucesnik je netko izvan promatranog sustava (SuD – System under discussion) koji je s njim u medudjelovanju, ali nije njegov dio. Predstavlja skup uloga koje korisnici slucajeva korištenja igraju tijekom njihove interakcije s slucajevima korištenja

- Koristi slucaj korištenja za obavljanje nekog dijela – korisnik
- Slican je vanjskom entitetu u SSA
- Može biti živo bice (Korisnik, Pacijent, Pilot), ili drugi sustav (SustavNaplate, Banka, Prijevoznik)

**Vrste ucesnika**

**Primarni ucesnik** pokrece slucaj korištenja (osim «extend» i «include», koje pokrecu drugi slucajevi korištenja

**Sporedni ucesnik** nadzire rad slucaja korištenja ili obavlja nešto nakon što ga slucaj korištenja pozove

**Slucaj korištenja**

Slucaj korištenja je prica koja opisuje kako sudionici koriste sustav da bi postigli odredene ciljeve ili obavili poslove. On predstavlja apstraktni zadatak (sadrži skup aktivnosti) kojeg izvode sudionici

**Generalizacija ucesnika**

Koristi se kad su razliciti ucesnici u jednakom medudjelovanju sa slucajevima korištenjima i imaju zajednicko ponašanje.

**«include»**

Koristi se da ukljuci ponašanje drugog slucaja korištenja u osnovni
Koncept slican pozivu procedure (call):

- Mora biti navedeno mjesto ukljucivanja u slucaju korištenja
- Nakon ukljucivanja, kontrola se vraca osnovnom slucaju korištenja
- Osnovni slucaj korištenja nije kompletan bez ukljucenja!!

**«extend»**

Koristi se da se ponašanje osnovnog slucaja korištenja proširi proširenjem.
Osnovni slucaj korištenja ima oznacena mjesta gdje se može proširiti (hooks)
Za razliku od «include», osnovni slucaj je kompletan i bez proširenja