# Istorija razvoja modela I alata

## OSNOVE OBJEKTNOG PRISTUPA

Sistem je skup povezanih objekata

Pod objektom se podrazumeva entitet koji je sposoban da čuva svoja stanja i koji stavlja na raspolaganje okolini skup operacija preko kojih se ta stanja prikazuju ili menjaju.

Bitna karakteristika objekata u OO pristupima je učaurenje (encapsulation), sakrivanje informacija (information hiding). U strogo objektnim pristupima jednini vidljivi deo objekta su operacje i to ne način na koji su implementirane, već samo njihovi efekti (specifikacija).

U složenom sistemu postojaće mnoštvo objekata, pa bi i njihov direktan opis bio veoma složen. Zbog toga je neophodno uvesti apstrakcije klasifikacije i generalizacije.

U apstrakciji klasifikacije definišu se pojam tipa, odnosno klase objekata.

U najnovijim pristupima se insistira na razlici koncepata tipa i klase objekata:

Pod tipom se podrazumeva kategorija objekata koji imaju isti skup stanja. Neki konkretan objekat se tretira kao pojavljivanje tipa.

Tip objekta se definiše sa dva aspekta:

(1) kao jedan interfejs preko koga se definišu spoljne karakteristike objeketa toga tipa i

(2) kao jedna ili više klasa koje predstavljaju različite implementacije datog tipa.

## STANDARDNI OPIS KLASA U UML-u

Ime-klase   
  
 vidljivost naziv-atributa-1: tip-podatka-1 = početna vrednost-1 iskaz osobina  
 vidljivost naziv-atributa-2: tip-podatka-1 = početna vrednost-1 iskaz osobina   
.....................   
 vidljivost naziv-operacije-1 (lista-argumenata-1): tip-rezultata-1 iskaz osobina   
 vidljivost naziv-operacije-2 (lista-argumenata-1): tip-rezultata-2 iskaz osobina   
....................

Svaka operacija ima “pozvani” objekat kao implicitni argument. Ponašanje operacije zavisi od klase kojoj “pozvani” objekat pripada. Objekat “zna” svoju klasu, pa samim tim i način implementacije posmatrane operacije. Osobina da objekat poziva neku operaciju drugog objekta ne znajući kojoj klasi ovaj objekat pripada naziva se polimorfizam.

## MODELI

MODELI ZA OPIS FUNKCIJA SISTEMA:

Dijagrami slučajeva korišćenja: "Use Case Diagrams"

(Strukturna sistemska analiza)

MODELI ZA OPIS STRUKTURE SISTEMA:

Dijagrami klasa

(Model objekti veze)

MODELI ZA OPIS DINAMIKE:

Dijagrami sekvenci ( Sequence Diagrams)

Dijagrami kolaboracije (Collaboration Diagrams)

Dijagrami promene stanja (State Transition Diagrams)

Dijagrami aktivnosti (Activity Diagrams)

DIJAGRAMI ZA OPIS IMPLEMENTACIJE

Deployment Diagrams

# Analiza zahteva i specifikacija IS

## MODELI I MODELOVANJE

Model je "subjektivni odraz objektivne stvarnosti". Zbog toga može da postoji više različitih modela istog sistema, sa istog ili različitih aspekata.

Model je neka simplifikacija realnosti.

Modeli se izgrađuju da bi se bolje razumeo realni sistem.

Modelovanje je način da se savlada sločenost nekog sistema.

Modelovanje je opšti pristup u svim unženjerskim disciplinama.

U svakoj oblasti postoji, obično više, intelektualnih alata (jezika) za modelovanje sistema- UML za oblast razvoja softvera

## CILJEVI UML-a

OPŠTI VIZUELNI JEZIK ZA MODELOVANJE SOFTVERSKIH SISTEMA I RAZMENU DOBIJENIH MODELA

MOGUĆNOST PROŠIRENJA I SPECIJALIZACIJE OSNOVNIH KONCEPATA U SKLADU SA POTREBAMA VRSTE SISTEMA KOJA SE MODELIRA

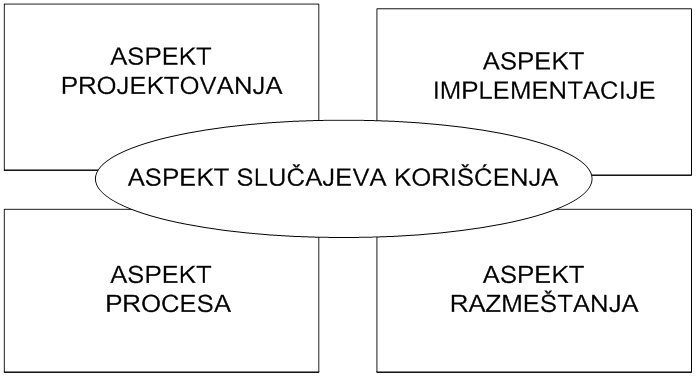
PODRŠKA SPECIFIKACIJI NEZAVISNOJ OD RAZVOJNE METODOLOGIJE I IMPLEMENTACIONOG OKRUŽENJA

UVOĐENJE FORMALIZMA KOJI ĆE OMOGUĆITI RAZUMEVANJE JEZIKA

PODRŠKA KONCEPTIMA VIŠIH NIVO APSTRAKCIJE: KOMPONENTA, KOLABORACIJA, APLIKACIONI KOSTUR

## ASPEKTI MODELA U UML-U

Za svaki aspekt daje se statički i dinamički opis sistema.



### ASPEKT SLUČAJEVA KORIŠĆENJA

Opisuje se ponašanje sistema sa tačke gledišta korisnika prvenstveno, a koristi se u analizi i testiranju takođe. *Pretstavljaju funkcionalnu specifikaciju sistema.*

Statički opis ovoga aspekta daje se preko *Dijagrama slučajeva korišćenja*, a dinamički, prvenstveno tekstualno, a zatim i preko *dijagrama interakcija, dijagrama promene stanja ili dijagrama aktivnosti.*

### ASPEKT PROJEKTOVANJA

Aspekt projektovanja pretstavlja realizaciju sistema u "objektnom prostoru stanja".

Statički opis ovoga aspekta daje se preko *Dijagram klasa i Dijagrama objekata.*

Dinamički opis se daje preko *dijagrama interakcija, dijagrama promene stanja i dijagrama aktivnosti.*

### ASPEKT IMPLEMENTACIJE

Aspekt implementacije pretstavlja komponente i fajlove preko kojih se sistem fizički ostvaruje.

Statički opis ovoga aspekta daje se preko *Dijagrama komponenti.*

Dinamički opis se daje preko *dijagrama interakcija, dijagrama promene stanja i dijagrama aktivnosti.*

### ASPEKT PROCESA

Aspekt procesa opisuje način odvijanja procesa u sistemu, "niti" procesa, konkurentnost i sinhronizaciju.

Koriste se isti dijagrami kao i u aspektu projektovanja i za statički i za dinamički opis, a najviše dijagrami aktivnosti

### ASPEKT RAZMEŠTANJA

Aspekt razmeštanja pretstavlja topologiju sistema, računarsko-komunikacionu mrežu. Pokazuje se kako su u ovoj mreži razmeštene komponente koje pretstavljaju fizičku realizaciju sistema

*Dijagrami razmeštaja* se koriste za statički opis.

Dinamički opis se daje preko *dijagrama interakcija, dijagrama promene stanja i dijagrama aktivnosti.*

## FUNKCIONALNI MODEL SISTEMA. STRUKTURNA SISTEMSKA ANALIZA SLUČAJEVI KORIŠĆENJA

Informacioni sistemi mogu biti veoma složeni. OO model složenog IS može sadržati i nekoliko hiljada različitih objekata sa mnoštvom njihovih atributa i veza. Zbog toga prvi modeli u razvoju nekog sistema ne mogu da budu objektni, moraju biti funkcionalni.

KAO ALATI ZA MODELOVANJE FUNKCIJA SISTEMA (TRANSFORMACIJE ULAZA U IZLAZ) KORISTIĆE SE STRUKTURNA SISTEMSKA ANALIZA (KONVENCIONALNI MODEL) I MODEL SLUČAJEVA KORIŠĆENJA (UML)

FUNKCIONALNI MODEL SISTEMA:

PREDSTAVLJA SISTEM KAO "CRNU KUTIJU"

PRETSTAVLJA SE FUNKCIONALNOST SISTEMA NA NAČIN KAKO JE VIDE SPOLJNI OBJEKTI

PRETSTAVLJAJU SE ULAZI I IZLAZI IZ SISTEMA I FUNKCIJE KOJE TRANSFORMIŠU ULAZE (POBUDU, STIMULACIJU) U IZLAZE

PRETSTAVLJA MODEL ZAHTEVA JER TREBA DA POKAŽE POTPUNO, PRECIZNO I NEDVOSMISLENO KAKO ĆE OBJEKTI VAN SISTEMA (KORISNICI, AKTERI) KORISTITI POSMATRANI SISTEM)

## MODEL SLUČAJEVA KORIŠĆENJA

Pod terminom “slučaj korišćenja” podrazumeva se jedan specifičan način korišćenja IS, jedna “atomska” funkcija IS. Preko “slučaja korišćenja” opisuje se interkcija nekog objekta van sistema sa samim IS. Skup “slučajeva korišćenja” predstavlja sve pretpostavljene načine korišćenja sistema.

Model slučajeva korišćenja je graf sa dve vrste čvorova: čvorovima koje predstavljaju aktere i čvorovima koji predstavljaju slučajeve korišćenja. Akter je objekat van sistema koji predstavlja tip (vrstu) korisnika. Akter je bilo šta što stupa u interakciju sa IS, ništa drugo van posmatranog IS nema nikakav uticaj na sistem. Akter može biti korisnik (čovek) ili neki drugi sistem. Treba praviti razliku izme|u korisnika i aktera. Korisnik je čovek koji koristi sistem, dok je akter specifična uloga koju korisnik ima u komunikaciji sa sistemom.

Direktna komunikacija između dva aktera i dva konkretna (oni sa kojima komuniciraju akteri) slučaja korišćenja se ne može predstaviti na modelu (grafu). Međutim, kako će kasnije biti prikazano, moguće je definisati asocijaciju između klasa slučajeva korišćenja i klasa aktera (apstraktni akteri i apstrakti slučajevi korišćenja), da bi se jednostavnije prikazao neki složeni model.

### VEZE U DIJAGRAMIMA SLUČAJEVA KORIŠĆENJA

ASOCIJACIJA prikazana veza izme|u aktera I slučaja korišćenja

GENERALIZACIJA veza opštijeg i specifičnijeg slučaja korišćenja koji nasleđuje opis opštijeg

<<extend>> stereotip veze zavisnosti koja referencira(ubacuje) moguće dodatno"ponašanje" opisano u posebnom apstraktnom SK, u osnovni SK

<<include>> stereotip veze zavisnosti koja eksplicitno ubacuje dodatno"ponašanje" opisano u posebnom apstraktnom SK, u osnovni SK.

ILUSTRACIJE VEZE <<include>>:

Osnovni SK eksplicitno uključuje ponašanje opisano sa apstraktnim SK. Služi da se izbegne višestruko opisivanje istog ponašanja

PRIMER VEZE <<extend>>

Osnovni SK implicitno proširuje ponašanje opisano u apstraktnom SK. Proširenje se vrši u tzv"tačkama proširenja"("uključi statistiku", za dati primer)

APSTRAKTNI AKTER

Kada dva aktera imaju slične uloge u odnosu na sistem oni mogu naslediti zajedničkog apstraktnog aktera. Ako se isti slučaj korićenja može da poveže sa različitim akterima, pogodno je definisati apstraktnog aktera i opisati samo jedan slučaj korišćenja. Koncept apstraktnog aktera je tako|e je koristan za opisivanje privilegija u korišćenju nekog sistema.

## STRUKTURNA SISTEMSKA ANALIZA

Strukturna sistemska analiza (SSA) je jedna potpuna konvencionalna metoda za specifikaciju informacionog sistema, odnosno softvera. Ona se na različite načine može povezati sa drugim metodama i modelima za projektovanje softvera, pa i sa objektnim metodama.

SSA posmatra informacioni sistem kao funkciju (proces obrade) koja, na bazi ulaznih, generiše izlazne podatke. Ulazni podaci se dovode u proces obrade, a izlazni iz njega odvode preko tokova podataka. Imajući u vidu zahtev da specifikacija treba da se oslobodi svih implementacionih detalja od interesa su samo sadržaj i struktura ulaznog toka, a ne i medijum nosilac toka.

Osnovni koncepti za specifikaciju IS u SSA su:

funkcije, odnosno procesi obrade podataka,

interfejsi, objekti van sistema sa kojima sistem komunicira preko tokova podataka,

tokovi podataka,preko kojih se podaci prenose izme|u interfejsa, funkcije i skladišta,

skladišta podataka, u kojima se permanentno čuvaju stanja sistema.

Njihov međusobni odnos se prikazuje preko dijagrama toka podataka koji prikazuje vezu interfejsa, odnosno skladišta kao izvora odnosno ponora podataka, sa odgovarajućim procesima, kao i međusobnu vezu procesa.

Imajući u vidu sve rečeno, jednu potpunu specifikaciju IS čine:

1.Hijerarhijski organizovan skup dijagrama toka podataka;

2.Rečnik podataka koji opisuje sadržaj i strukturu svih tokova skladišta podataka;

3.Specifikacija logike primitivnih procesa;

Pored dijagrama tokova podataka uobičajeno je za jedan sistem da se prikaže i Dijagram dekompozicije koji prikazuje celokupnu dekompoziciju sistema, od Dijagrama konteksta do primitivnih funkcija. Ovde se predlaže da se za izradu Dijagrama dekompozicije koriste Jackson-ovi dijagrami, jer se sa njima može opisati i logika primitivnuh procesa i time izvršiti bolja priprema za konstukciju Dijagram slučajeva korišćenja

### SSA- SINTAKSNA I METODOLOŠKA PRAVILA

Za formiranje DTP - ova postoji čitav skup formalnih (sintaksnih) pravila. Najznačajnije je pravilo koje se mora poštovati pri dekompoziciji procesa, pravilo balansa tokova: Ulazni i izlazni tokovi na celokunom DTP-u koji je dobijen dekompozicijom nekog procesa P moraju biti ekvivalentni sa ulaznim i izlaznim tokovima toga procesa P na dijagramu višeg nivoa. Pri tome se uzima u obzir dekompozicija tokova predstavljena u rečniku podataka.

Kao najvažnije metodološko pravilo koristi se pravilo da funkcije na DTP-u između sebe treba da komuniciraju isklljučivo preko skladišta. Korišćenje ovoga pravila vodi uvek ka identifikaciji nekih fundametalnih funkcija u sistemu, fundamentalnih sa sledeće dve tačke gledišta:

(I) funkcije su autonomne, jedna od druge zavise isključivo preko skladišta;

(II) bilo koja složenija funkcija dobija odgovarajućim kobinovanje fundamentalnih

# Konceptualno modelovanje

## Konceputulani model

Konceptualni model predstavlja suštinske karakteristike sistema za koji se projektuje baza podataka.

Opisuje se na visokom nivou apstrakcije, preko modela podataka.

Predstavlja celokupan mogući informacioni sadržaj baze podataka, nezavisno od toga koji je deo tog sadržaja i kako implementiran.

## Konstruisanje konceptulanog modela

Koriste se “intelektualni alati” kao što su model podataka ‘PMOV, IDEF1X, Dijagram klasa

## Pristupi konceptualnom modelovanju

Postoji više metodoloških pristupa konceptualnom modelovanju, a u razvoju modela nekog konkretnog sistema oni se gotovo uvek kombinuju.

1. Integracija podmodela;
2. Direktno modelovanje na bazi verbalnog opisa sistema;
3. Konkretizacija opštih (generičkih) modela, odnosno korišćenje uzora (“paterna”);
4. Normalizacija relacija; - (specifican za relacioni model)
5. Transformacija jednog modela u drugi (“direktno” i “inverzno” inženjerstvo).

## METODOLOGIJA MODELOVANJA

1. INTEGRISANJE PODMODELA
2. DIREKTNO MODELIRANJE
3. KOMBINOVANI METOD

### INTEGRACIJA PODMODELA

Postupak integracija podmodela podrazumeva:

1. Izgradnja poslovnog modela sistema i specifikacija identifikovanih aplikacija korišćenjem Strukturne sistemske analize Ili opisa slučajeva korišćenja preko Sistemskog dijagrama sekvenci sa, na primer XML opisom argumenata poruka ili Dijagramom aktivnosti sa strukturom objektnih tokova opisanih preko nekog modela podataka.
2. Izgradnja podmodela podataka za svaku specifikovanu aplikaciju, odnosno primitivni poslovni proces (funkciju).
3. Integracija podmodela u jedinstveni model celog sistema.

### DIREKTNO MODELOVANJE

*POD DIREKTNIM MODELOVANJEM PODRAZUMEVA SE RAZVOJ MODELA PODATAKA NA OSNOVU:*

POZNAVANJA REALNOG SISTEMA,

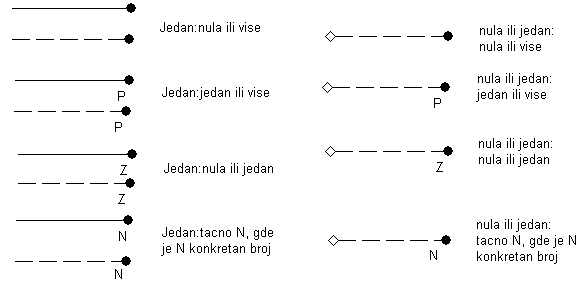
ISKUSTVA,

TEKSTUALNOG OPISA,

POZNAVANJA GENERIČKIH MODELA PODATAKA (PATERNA)

## VERZIJE MOV-a: IDEF1x standard





# UML Modeli - Aspekt projektovanja - (statički opis strukture sistema)

## ASPEKT PROJEKTOVANJA

Aspekt projektovanja pretstavlja realizaciju sistema u "objektnom prostoru stanja".

Statički opis ovoga aspekta daje se preko *Dijagram klasa i Dijagrama objekata.*

Dinamički opis se daje preko *dijagrama interakcija, dijagrama promene stanja i dijagrama aktivnosti.*

## DIJAGRAMI KLASA

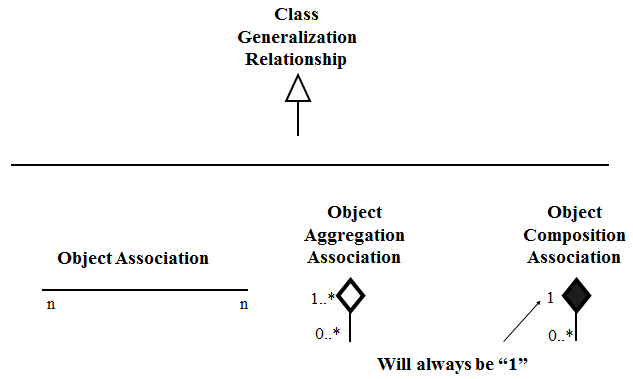
Dijagram klasa najčešće korišćeni dijagam u OO pristupima razvoju softvera

On predstavlja skupove klasa, interfejsa, kolaboracija i njihove međusobne veze

Koristi se da pretstavi:

* 1. *Osnovni "rečnik sistema" – definiše pojmove koji se u tom sistemu koriste,*

### Četiri tipa veze u dijagramu klasa



DEFINISANJE SEMANTIKE UML-a PREKO DIJAGRAMA KLASA

METAMODEL ZA MEHANIZME PROŠIRENJA



### Dijagram klasa (Stereotipi)

Važni stereotipi:

<<interface>> specificira kolekciju operacija za specifikaciju

servisa

<<type>> specificira strukturu i ponašanje (ne implementaciju)

<<enumeration>> specificira predefinisane vrednosti

<<implementationClass>> pomoćna klasa kreirani u detaljnom dizajnu

## DIJAGRAM OBJEKATA (POJAVLJIVANJA)

Dijagram objekata pokazuje skup objekata i njihovih veza u jednom trenutku vremena.

Dijagrami objekata modeliraju stanje sistema u jednom trenutku vremena, odnosno daju zamrznutu sliku sistema u jednom trenutku vremena.

Dijagrami objekata daju statičku strukturu za opis dinamičkog ponašanja sistema preko dijagrama interakcije. Oni su pojavljivanje odgovarajućeg dijagrama klasa i statički deo dijagrama interakcije

Pretstavljaju skup objekata i pojavljivanja veza, mogu da sadr`e i komentare i ograničenja, da budu podeljeni u pakete i podsisteme.

## DIJAGRAMI SEKVENCI I DIJAGRAMI KOLABORACIJE

Interakcije se mogu modelovati na dva načina:

Prikazujući vremenski redosled poruka:

DIJAGRAMI SEKVENCI

Prikazujući interakciju u kontekstu neke organizacije (strukture) objekata:

DIJAGRAMI KOMUNIKACIJE

Moguće je automatski prevesti jedan oblik u drugi.

## DIJAGRAMI PROMENE(PRELAZA) STANJA

Pri opisivanju dinamike sistema preko dijagrama prelaza stanja koriste se sledeći pojmovi:

1. Sistem (objekat) je skup objekata, njihovih atributa i njihovih veza. Struktura sistema - odnos njegovih objekata, veza i atributa opisuje se preko modela opisanih u prethodnom delu.

1. Stanje sistema (objekta) u jednom trenutku vremena predstavlja skup vrednosti atributa svih objekata i “vrednosti” svih veza u tom trenutku. Termin “vrednost veze” opisuje par (za binarne veze ili n-torku uopšte) identifikatora pojavljivanja objekata koji su u vezi.
2. Događaji iniciraju promene stanja sistema. Odziv sistema na neki događaj zavisi od stanja u kome se on nalazi. Događaj mo`e da prouzrokuje promenu stanja sistema i/ili da indukuje novi događaj. Događaj se zbiva u jednom trenutku vremena, događaj nema trajanje (u vremenskoj skali u kojoj se posmatra dati sistem). Ponekad se događaj i poruka tretiraju kao sisnonimi. Međutim, precizno, poruka je pojavljivanje događaja, što će kasnije biti pokazano.
3. Dijagram prelaza (promene) stanja je apstrakcija koja pokazuje stanja, događaje i prelaze (tranzicije) iz stanja u stanje kao mogući odziv na događaje.
4. Dijagram promene stanja povezuje stanja (konkretna, imenovana) sa događajima u sistemu. Promena stanja izazvana događajem naziva se tranzicija (prelaz). Dijagram promene stanja je usmereni graf u kome su čvorovi stanja, a grane tranzicije, sa usmerenjem od polaznog do prouzrokovanog stanja. Granama grafa daju se nazivi događaja koji prouzrokuju tranziciju. (Jedan događaj mo`e da prouzrokuje više tranzicija, pa više grana mo`e da ima isto ime !!!).
5. Početna i krajnja stanja. Može se uvesti i koncept početnog i krajnjeg stanja, za objekte (sisteme) koji imaju “ograničen život”. U tom slučaju, početno stanje je rezultat kreiranja odgovarajućeg objekta, a krajnje podrazumeva njegovo “uništenje” (nestanak). Početna i krajna stanja na grafu imaju specijalne oznake, a mogu imati i imena.
6. Uslovi. Uslov je Bulova funkcija nad vrednostima atributa i veza. Stanja sistema se mogu opisati preko uslova. Iskaz da je objekat u nekom stanju je uslov. Pored toga, uslovi se mogu koristiti da ograniče tranzicije prouzrokovane događajima. Ponekad, za prelaz sistema iz jednog stanja u drugo potrebno je, pored događaja, da bude ispunjen i neki uslov. Na dijagramu prelaza stanja uslov se iskazuje uz naziv događaja, unutar uglaste zagrade.
7. Akcija predstavlja jedno "atomsko sračunavanje"koje prouzrokuje promenu stanja sistema ili vraća neku vrednost. Neka akcija okida događaj koji će sistem prevesti iz jednog u drugo stanje.Akcija mo`e da pozove operaciju nekog objekta, da kreira ili uništi neki objekat ili da pošalje signal nekom objektu. Akcije se mogu pridružiti stanjima i tranzicijama. Ako se akcije pridružuju stanjima one mogu biti:

"entry" – akcija koja se obavlja uvek pri ulazu u stanje, bez obzira koja je tranzicija to prouzrokovala;

"exit" - akcija koja se obavlja uvek pri napuštanju stanja, bez obzira koja je tranzicija to prouzrokovala;

"interna tranzicija" – akcija koja ne menja stanje sistema.

Akcija se može obaviti i pri prelazu iz jednog u drugo stanje Na tranziciji akcije se iskazuju uz naziv događaja, iza uslova i oznake "/".

1. Aktivnosti. Kada je objekat (ili deo sistema) u nekom stanju, on je bilo neaktivan ili obavlja neku aktivnost dok ga događaj ne prevede u neko drugo stanje.Drugim rečima, dok obavlja određenu aktivnost, sistem je u datom stanju. Na dijagramu prelaza stanja aktivnosti se definišu nazivom iza ključne reči DO, u okviru stanja (čvora).
2. Sinhronizacija konkurentnih aktivnosti. U jednom stanju se može obavljati više konkurentnih aktivnosti.Ove aktivnosti ne moraju biti sinhronizovane, mogu se obavljati bilo kojim redom, ali sve one moraju biti obavljene pre nego što se izvrši tranzicija u drugo stanje. Konkurentne aktivnosti u jednom stanju se prikazuju podelom stanja (čvora) na delove razmaknute isprekidanom linijom.
3. Neoznačena ili automatska tranzicija. Koristi se da bi se prikazala automatska tranzicija iz jednog stanja u drugo koja se obavlja čim se aktivnost u nekom stanju obavi. Kraj aktivnosti u nekom stanju mo`e se tretirati kao neimenovani događaj. Taj neimenovani događaj "okida" neimenovanu tranziciju u drugo stanje.
4. Dekompozicija dijagrama prelaza stanja. Dijagrami prelaza stanja se mogu dekomponovati na sledeće načine:

*(i) Kompozitno stanje, odnosno sekvencijalna podstanja.*

*(ii) Generalizacija stanja.*

*(iii) Agregacija stanja - agregaciona konkurentnost.*

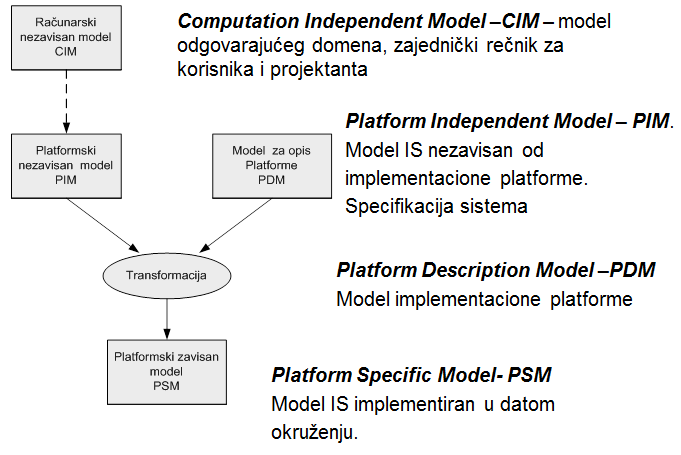
Kompozitno stanje. Svaka aktivnost u okviru stanja se može predstaviti posebnim Dijagramom dekompozicije. Dobijeni dijagram ima ulaz i izlaz, odnosno početno i završno stanje. Ova stanja i odgovarajući događaji su nevidljivi sa dijagrama višeg nivoa i predstavljaju samo detaljan opis aktivnosti. Ugnježdeni dijagram stanja “zamenjuje” jedno stanje na dijagramu višeg nivoa. Sve ulazne tranzicije sa dijagrama višeg nivoa prenose se na početno stanje, a sve izlazne transakcije na završno. Izlazno stanje može da generiše poruku - događaj svome nadređenom stanju.

Generalizacija stanja. Pod generalizacijom stanja će se ovde podrazumevati odnos između stanja i podstanja u kome podstanje nasleđuje osobine stanja, promenljive stanja i tranzicije stanja (precizno, izlazne tranzicije). Ako je neki događaj primljen kada je objekat u datom podstanju, sve tranzicije nadstanja su potencijalno primenljive, ako nisu “prekrivene” istoimenom tranzicijom na podstanju. Nadstanje se predstavlja kao kontura koja zaokružuje podstanja. Moguće su tranzicije od nadstanja u podstanje i obrnuto, kao i tranzicije iz podstanja u neko drugo stanje van konture.

Agregacija stanja - agregaciona konkurentnost. Konkuretna stanja su posledica postojanja složenih objekata koji su agregacija svojih komponenti ili postojanja višestrukih paralelnih aktivnosti.

U slučaju složenih objekata, svaka komponenta može da ima svoja stanja i svoj dijagram prelaza stanja. Stanje agregiranog objekta je Dekartov proizvod stanja njegovih komponenti.

# MODELOM VOĐENI RAZVOJ (ARHITEKTURA)



Modelom vođeni razvoj omogućava:

* + Specifikaciju sistema nezavisnu od bilo kakve implementacije;
  + Specifikaciju platforme;
  + Izbor platforme za implementaciju specifikaovanog sistema;
  + Transformaciju specifikacije sistema u izabranu platformu.

## MDA i interoperabilnost

Sa tačke gledišta korisnika heterogeni distribuirani računarski sistem treba da se posmatra kao “jedinstveni svet”, bez ikakvih “platformski specifičnih barijera”.

Broj različitih platformi stalno raste i ne može se očekivati standardizacija (ptpuna unificiranost) na tom nivou.

Preostaje da se standardizacija ostvari na nivo specifikacije, odnosno modela i da se preko tih opštih modela ostvari interoperabilnost delova sistema realizovanih na različitim platformama.

## MDA- osnovni pojmovi

“*Model driven*” zato što se u svim fazama razvoja (spefifikacija,projrktovanje, implementacija, održavanje) koriste modeli.

“*Arhitektura*”. Arhitektura nekog sistema, uopšte, je specifikacija njegovih delova i pravila koja definišu način komunikacije između delova. U MDA pristupu definišu se modeli koji se koriste u razvoju softvera i pravila njihovog povezivanja.

*Platforma*  je sistem sa definisanim skupom funkcija (bez detaljne specifikacije njihove implementacije) preko koji je moguće implementiratineki softver.

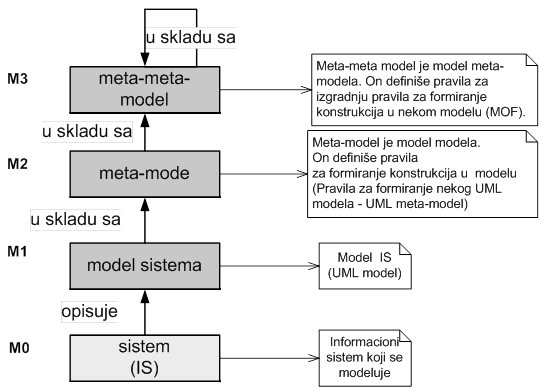
Vrste platformi:

*Generičke:Objektna,* *Batch, Tok podataka*

*Specifične tehnologije: CORBA, J2EE*

*Tehnologije pojedinih proizvođača*: *CORBA: Borland VisiBroker;J2EEs: IBM WebSphere, Oracle;Microsoft .NET*.

## MODELI - četvoronovoska hijerarhija meta-modela



## MDA OMG standardi

UML – OMG standardni jezik za modelovanje diskretnih sistema:

UML 2.0 standard dat je u dva osnovna dokumenta: (1) *UML 2.0:* *Infrastructure - osnovni koncept jezika i (2) UML 2.0: Superstructure*- nadgradnja sa konstrukcijama korisničkog nivoa.

MOF (Meta object facility) – standardni jezik za specifikaciju meta modela. Mof sdarži dva osnovna dela:

Abstrakni model generičkih objekata i njihovoh asocijacija (vrlo sličan sa UML jezgom)

Skup pravila za preslikavanje MOF modela u jezički nezavisne interfejse. Implementacija ovih interfejsa omogućava pristup i ažuriranje bilo kog modela zasnovanog na MOF-u.

XMI(XML Metadata Interchange). Definiše XML tagove preko kojih se može opisati serijalizovan model baziran na MOF-u. Služi prvenstveno za razmenu modela. Transformacija iz jednog u drugi često se vrši i na sledeći način:

Model1 🡪 XMI1🡪 XSLT 🡪XMI2 🡪 Model2

CWM *(The Common Warehouse Metamodel)* definiše metamodele koji predstavljaju stovarišta podataka (data warehousing) i odgovarajuće OLAP analize.

JMI (Java Metadata Interface) – omogućava preslikavanje MOF modela u Javu da bi se aostavrila mogućnost održavavanja MOF metabaza iz Jave.

## UML Infrastructure specifikacija

UML specifikacija se daje preko UML metamodela.

Jedan od osnovnih principa UML-a je proširljivost. Zbog toga UML 2.0 Infrastructure librarary sadrži dva osnovna dela:

Jezgro (Core)

Profiles - proširenja. Profil – mehanizam prilagođavanja modela konkretnim platformama (J2EE/EJB, .NET/COM+ i drugim)

### UML- Odnos “Jezgra” i ostalih modela

Jezgro se koristi za definisanje drugih metamodela i kreiranje drugih jezika preko koncepta profila. Jezgro je praktično zajednički deo za sve ove modele.

### Paketi u UML jezgru

*PrimitiveTypes* sadrže nekoliko predefinisanih tipova potrebnih za metamodelovanje, posebno za UML i MOF.

*Abstractions* sadrži nekoliko abstraktnih metaklasa koje se višestruko koriste(specijalizuju) i različitim metamodelima.

*Basic* sadrži podskup koncepata iz paketa *Constructs* koji se prvenstveno koriste za objektne jezike i XMI

*Constracts* sadrži metaklase koje su konkretneklase okrenute prema konceptima objektno-orjentisanog modelovanja (koriste se za UML i MOF i njihovo usaglašavanje)

## UML specifikacija

Svaki koncept iz odgovarajućeg paketa se opisuje sa:

Dijagramom klasa

Opštim opisom

Opisom svih atributa

Opisom svih asocijacija

Specifikacijom semantike

Generalization je podtip DirectRelationship

To znači da nasleđuje sve odgovarajuće osobine pa i asocijacije. Preslikavanje *generalization* je podskup preslikavanja *target*, a *specific* podskup preslikavanja *source.*

To znači da ako se za neko pojavljivanje klase *generalization* zahteva *target* dobiće se instanca asocijacije *general.*

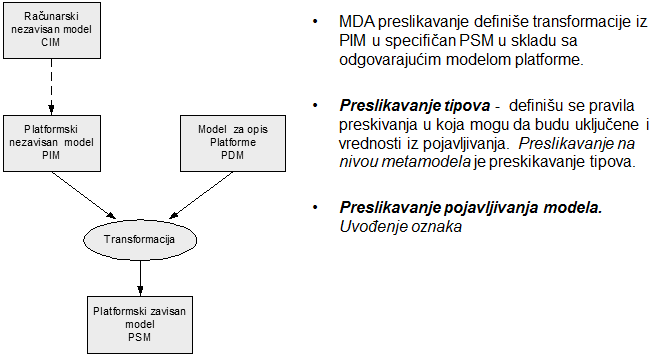
## MOF

Za konstrukciju meta-modela koristi se MOF.

Pošto je meta-model model modela, na njega se može primeniti isti jezik koji se koristi za izgradnju modela nekih sistema, pod uslovom da je takav jezik reflesivan (što praktično znači dovoljno opšt)

UML dijagram klasa ima tu osobinu i zato je on osnova MOF-a.

## Preslikavanje PIM 🡪PSM Osnovni pojmovi



*Oznake (Marks)*  mogu da budu:

Tipovi modela (klase, asocijacije, uloge u asocijacijama ili drugi tipovi)

Stereotipovi iz UML profila

Elementi metamodela (MOF-a, na primer)

Neki nefunkcionalni zahtevi (specifikacija kvaliteta ili zahtevanih performamsi).

*Tempejt* je model koji definiše transformaciju (neka vrsta “design pattern”).

*Jezik za preslikavanje:* prirodni jezik, neki algoritamski jezik ili jezik za preslikavanje modela (XSLT za XMI1🡪XMI2).The current. MOF Query/View/Transformation RFP (Request for Proposal) zahteva da se u okviru MOF-a definiše standardni jezik za preslikavanje modela.