PROJEKTIRANJE SOFTVERA

UVODNO PREDAVANJE

SADRŽAJ

- O predmetu
- Sofver, softverski inžinjering i softverski procesi
- Životni ciklus razvoja softvera
- Analiza i dizajn
- Modeliranje softvera
- Unified Modeling Language - UML
- Case alati za UML modeliranje

O PREDMETU

- Ciljevi predmeta
  - Upoznavanje sa osnovnim softverskim pojmovima
  - Razumijevanje procesa analize i dizajna softvera
  - Rad sa UML jezikom u cilju modeliranja što kvalitetnijeg softvera
- Sadržaj predmeta čini odabrani set UML dijagrama (detaljnije u syllabusu predmeta)
- Evaluacija
  - Ispit – 100%, ili
    - Kolokvij I – 50%
    - Kolokvij II – 50%

SOFTVER I SOFTVERSKI INŽINJERING

- **Softver** predstavljaju kompjuterski programi i pridružena mu dokumentacija (zahtjevi, modeli dizajniranja, korisnička dokumentacija)
- Vrste softvera:
  - Korisnički (namijenjen specifičnom korisniku)
  - Generički (proizveden da bi se prodao širem krugu različitih korisnika)
  - Ugradivi (ugrađuje se u hardver i teško ga je mijenjati)
  - Real-time (koristi se za upravljanje i nadzor sistema, mora djelovati trenutno)
  - Softver za obradu podataka (koristi se za upravljanje poslovanjem, ključni su tačnost i sigurnost)
- **Softverski inžinjering** je disciplina koja razmatra sve aspekte razvoja (proizvodnje) softvera. Uključuje teorijske osnove, metode i alate za razvoj profesionalnog softvera. Ima za cilj proizvesti kvalitetan softver, isporučen na vrijeme, unutar raspoloživog budžeta i zadovoljavajući sve korisničke zahtjeve i želje.

PROBLEMI I MOGUĆA RJEŠENJA

- Uzroci neuspjeha
  - Složenost problema
  - Nedostatak usmjerenosti korisniku
  - Zanemarivanje okruženja organizacije
  - Pretjerani optimizam
  - Izostanak praćenja napretka
  - Nedostatak komunikacije u timu
  - Loša organizacija i vođenje projekta
  - Loša izvedba projekta

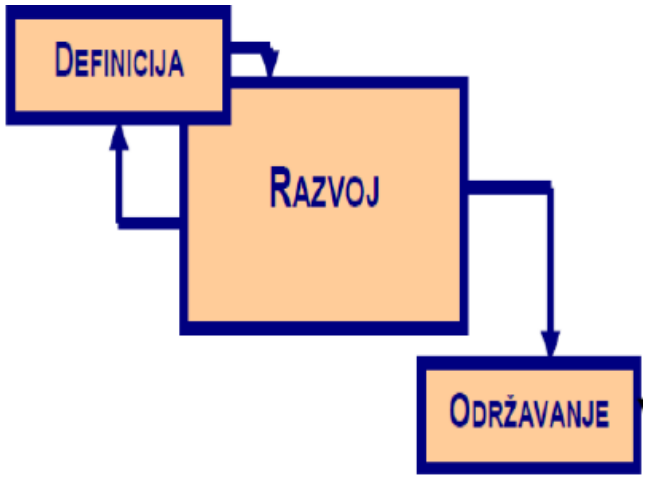
- Poboljšanje uspješnosti
  - Projektiranje
  - Planiranje, upravljanje i praćenje projekta
  - Uključivanje korisnika u razvoj
    - KORISNIK POZNAJE
    - INFORMATIČAR UPOZNAJE

SOFTVERSKI PROCESI

- Softverski proces predstavlja skup aktivnosti čiji je cilj razvoj i evolucija softvera
- Aktivnosti u softverskom procesu su:
  - Specifikacija (šta sistem treba raditi i koja su njegova ograničenja)
  - Razvoj (proizvodnja softverskog sistema)
  - Validacija (provjera da li softver zadovoljava korisničke zahtjeve, odnosno zahtjeve kupaca)
  - Evolucija (mijenjanje softvera kao odgovor na zahtjeve kupaca)
- Atributi dobrog softvera:
  - Održivost (softver se mora razviti kako bi zadovoljio promjene zahtjeva)
  - Pouzdanost (softver mora uživati povjerenje od strane korisnika)
  - Efikasnost (softver ne treba rasipnički koristiti sistemske resurse)
  - Prihvatljivost (softver mora biti prihvatljiv od krajnjih korisnika, razumljiv, upotrebljiv i komptibilan sa drugim sistemima)

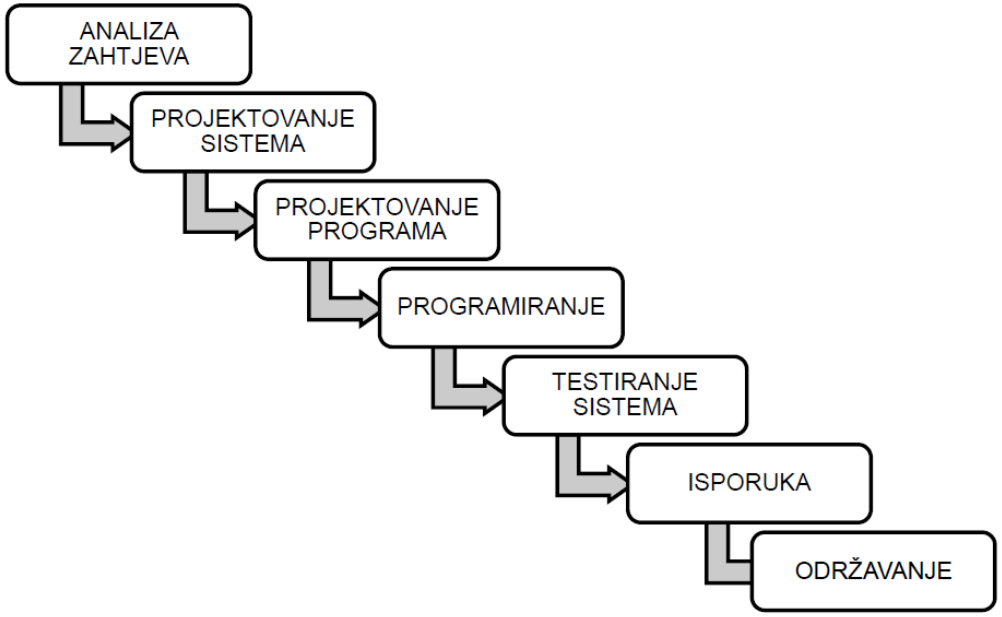
ŽIVOTNI CIKLUS RAZVOJA SOFTVERA

- Životni ciklus razvoja softvera (SDLC – System Development Life Cycle) opisuje „životni vijek“ od nastanka potrebe za softverom, preko implementacije, održavanja, pa sve dok se ne prestane koristiti.
- Model procesa razvoja softvera definira redoslijed razvojnih faza.
- Postoji više varijanti SDLC-a, a to su:
  - Tradicionalni linearni (sekvencijalni) “vodopadni” (eng. “waterfall”)
  - Paralelni model,
  - Iterativni model ili evolucijski model,
  - Spiralni model,
  - Agilne metode itd.



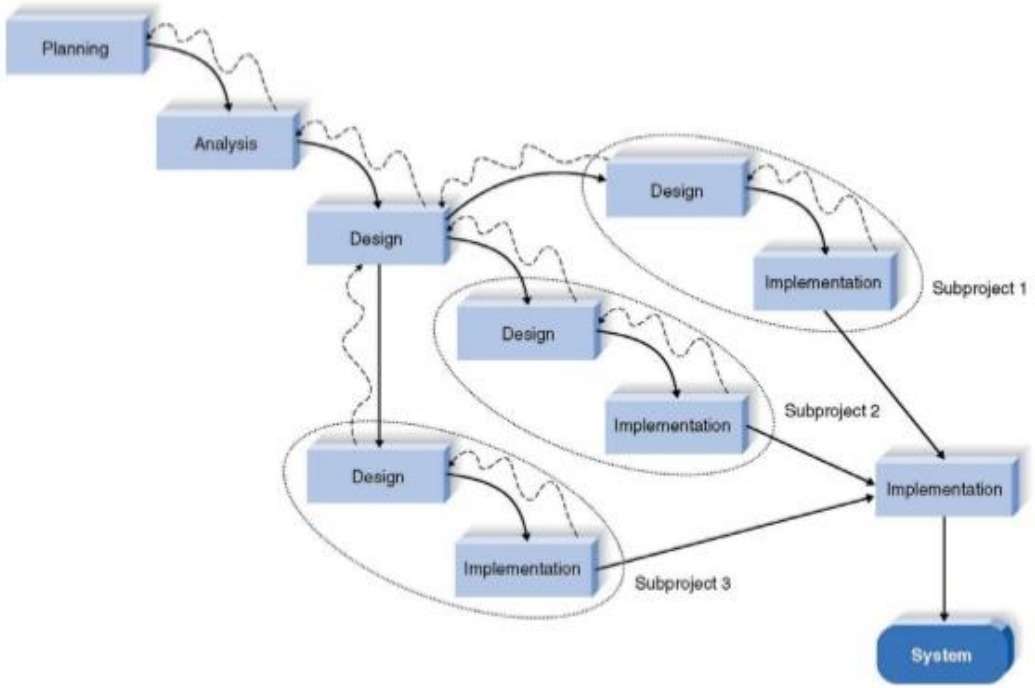
VODOPADNI MODEL

- Pri korištenju sekvencijalnih metodologija razvoja analitičar sistema i članovi razvojnog tima napreduju sekvencijalno od jedne do druge faze projekta. Svaka faza se u cjelosti završi i potom se prelazi na sljedeću fazu. Nije uobičajeno vraćati se na prethodnu fazu.
- Prednosti:
  - Unaprijed identifikovani zahtjevi (prije početka razvoja)
  - Promjene u zahtjevima su minimalne
- Nedostaci:
  - Dizajn mora biti u potpunosti završen prije početka implement
  - Vrijeme provedeno na svakoj od faza



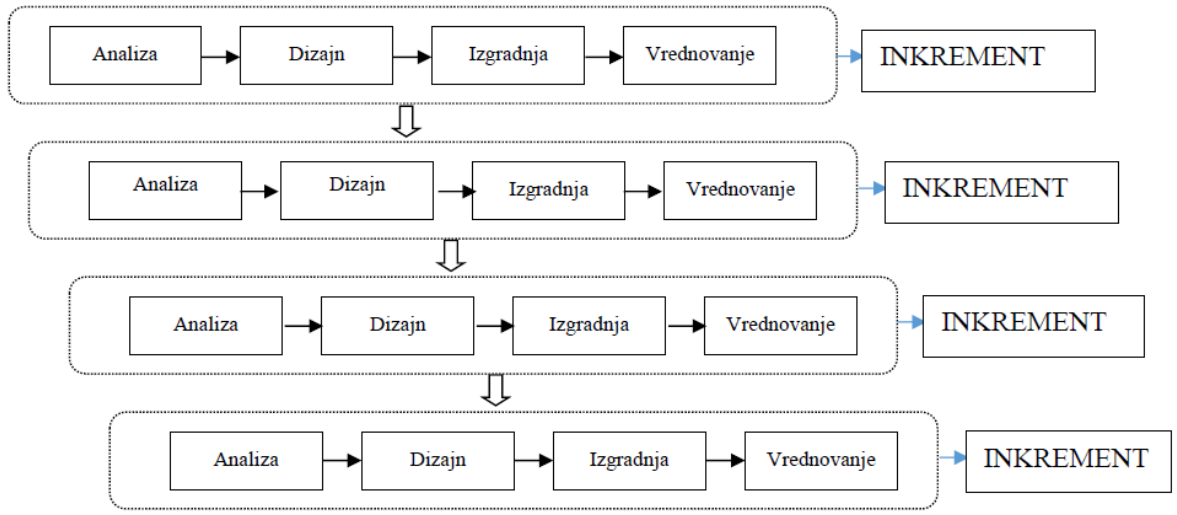
PARALELNI PRISTUP

- Metodologija paralelnog razvoja nastoji smanjiti vrijeme između faze analize i isporuke softverskog proizvoda. Prvo se napravi dizajn cjelokupnog proizvoda a zatim se projekt dijeli na više podprojekata. Svi podprojekti, ukoliko ne postoji međusobna ovisnost među njima, se realiziraju paralelno. Rezultati svakog od podprojekata potom se integriraju u jednu cjelinu, projekt.
- Prednosti:
  - Znatno se smanjuje vrijeme razvoja
  - Manje su šanse za prepravke
- Nedostaci:
  - Potrebno više resursa
  - Ponekad je teško integrisati podprojekte



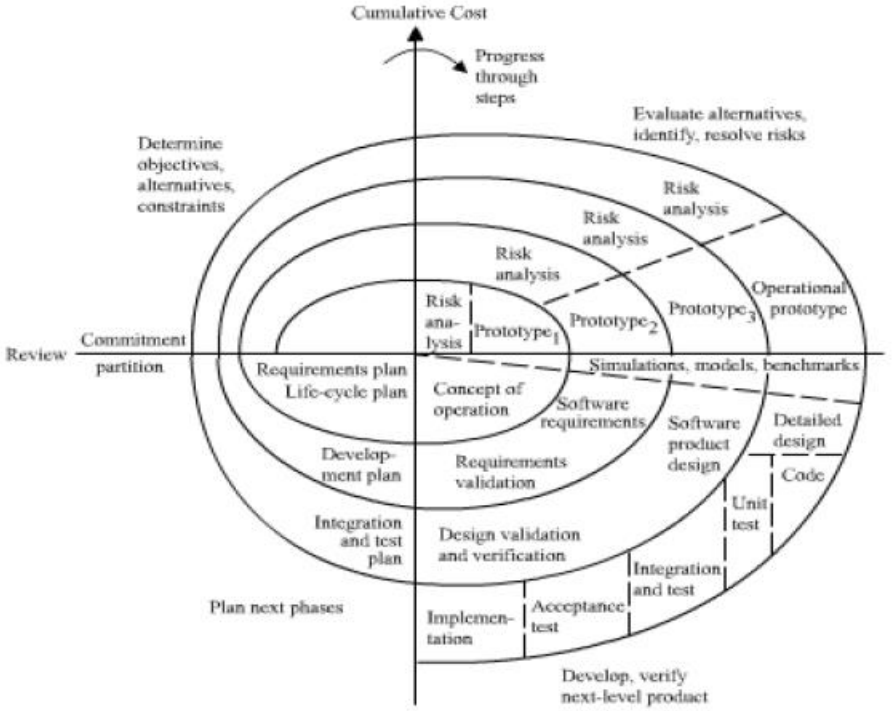
ITERATIVNI PRISTUP

- Iterativni ili evolucijski model je model u kojem korištenje softverskog proizvoda mijenja perspektivu korisnika u odnosu na softverski proizvod. Svaka promjena perspektive može rezultirati generiranjem novih korisničkih zahtjeva. Na taj način softverski proizvod raste zajedno sa organizacijom u kojoj se koristi. Ovakav pristup razvoja podrazumjeva brzo prototipiranje, odnosno, gradnju dijelova softverskog proizvoda u inkrementima.
- Prednosti:
  - Softver se proizvodi u ranim fazama životnog ciklusa
  - Korisnici u ranim fazama probaju rezultate
  - Pojednostavljeno testiranje
- Nedostaci:
  - Izmjene zahtjeva
  - Duže vrijeme razvoja



SPIRALNI MODEL

- Na početku svake faze radi se analiza rizika i nastoje se utvrditi mogući rizici. Cilj analize rizika je eliminirati rizike ili barem ih svesti na najmanju moguću mjeru. Ako je rizik nastavka projekta prevelik, tada se projekat prekida.
- Faze spiralnog modela su:
  - Analiza rizika i procjena alternativa
  - Razvoj i verifikacija sljedećeg "proizvoda„
  - Planiranje sljedeće faze
  - Pregled - Određivanje ciljeva, alternativa i ograničenja.
- Prednosti
  - Visok stepen analize rizika
  - Softver se proizvodi u ranim fazama životnog ciklusa
- Nedostaci
  - Može biti veoma skup model
  - Zahtjeva visok stepen ekspertize u analizi rizika
  - Nije dobar za male projekte



AGILNE METODE

- Agilne metode podrazumijevaju iteracije u kratkim vremenskim intervalima. Obavezan timski rad u stalno prisustvo naručioca projekta (klijenta).
- Metodologija zasnovana na principu kontinuirane isporuke manjih ali funkcionalnih dijelova sistema
  - SCRUM
  - Ekstremno programiranje (XP)
- Izbjegava se proces modeliranja softvera.

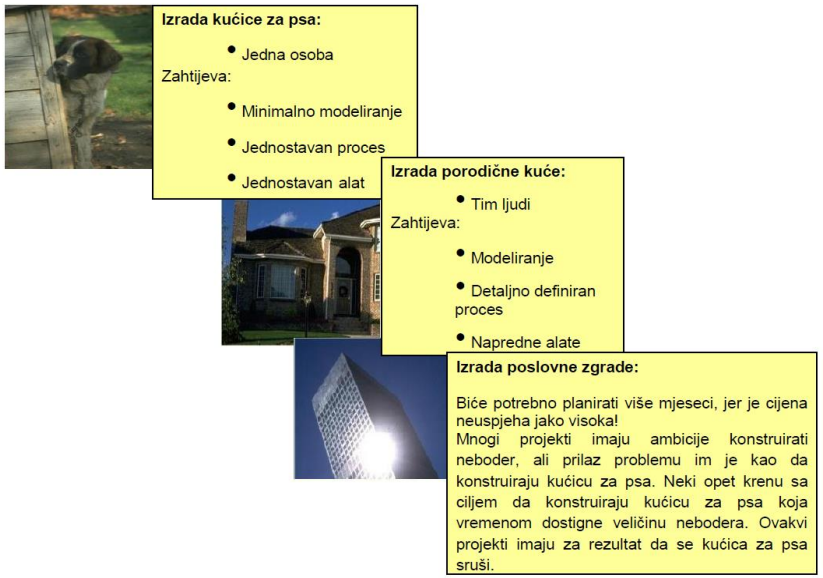
Procesi i alati	→	Interakcija
Detaljna dokumentacija	→	Funkcionalna aplikacija
Precizni ugovori	→	Kolaboracija
Detaljan plan	→	Reakcija na promjene

ANALIZA I DIZAJN

- Analiza je tehnika kojom rastavljamo sistem na njegove komponente s ciljem proučavanja kako te komponente međusobno funkcionišu. Predstavlja istraživanje problema i zahtjeva sistema (ne i rješenja).
- Cilj je prikupiti što više informacija o sistemu. Tom prilikom koristimo različite tehnike prikupljanja činjenica:
  - Pregled postojeće dokumentacije
  - Intervjuisanje osoba koje dobro poznaju sistem
  - Posmatranje korisnika u realnom okruženju
  - Anketiranje budućih korisnika
  - Analiza sličnih sistema
- Dizajn predstavlja identifikaciju konceptualnog rješenja, odnosno identifikaciju dijelova sistema iodnosa među njima.
- Ako analiza znači definiranje problema, onda dizajn je proces definiranja rješenja problema.

MODELIRANJE SOFTVERA

- Centralna aktivnost proizvodje dobrog softvera.
- Model je apstrakcija nekog problema (pojednostavljena realnost).
- Modeliranjem reduciramo domenu problema i fokusiramo se u svakom momentu samo na jedan aspekt.
- Konstruiramo modele kompleksnih sistema da bismo razumjeli sistem kao cjelinu.
- Modeli su vodič koji se koristi pri izgradnji sistema.
- U kojoj mjeri ćemo modelirati ovisi o kompleksnosti problema i veličini projekta



UML

- **UML** (**U**nified **M**odeling **L**anguage) je grafički jezik za predstavljanje, specifikaciju, konstrukciju i dokumentaciju komponenti softverskog sistema.
- Standardnu definiciju UML-a je postavio OMG (**O**bject **M**anagment **G**roup). OMG je osnovana 1989. godine kao neprofitabilna organizacija sa ciljem kreiranja tržišta softvera na bazi komponenti, standardiziranja objektno - orijentiranog pristupa, pružanja odgovarajućeg okruženja za razvoj softvera, itd.
- Riječnik UML-a obuhvata tri osnovna tipa gradivnih blokova:
  - Stvari
  - Relacije
  - Dijagrame

UML STVARI

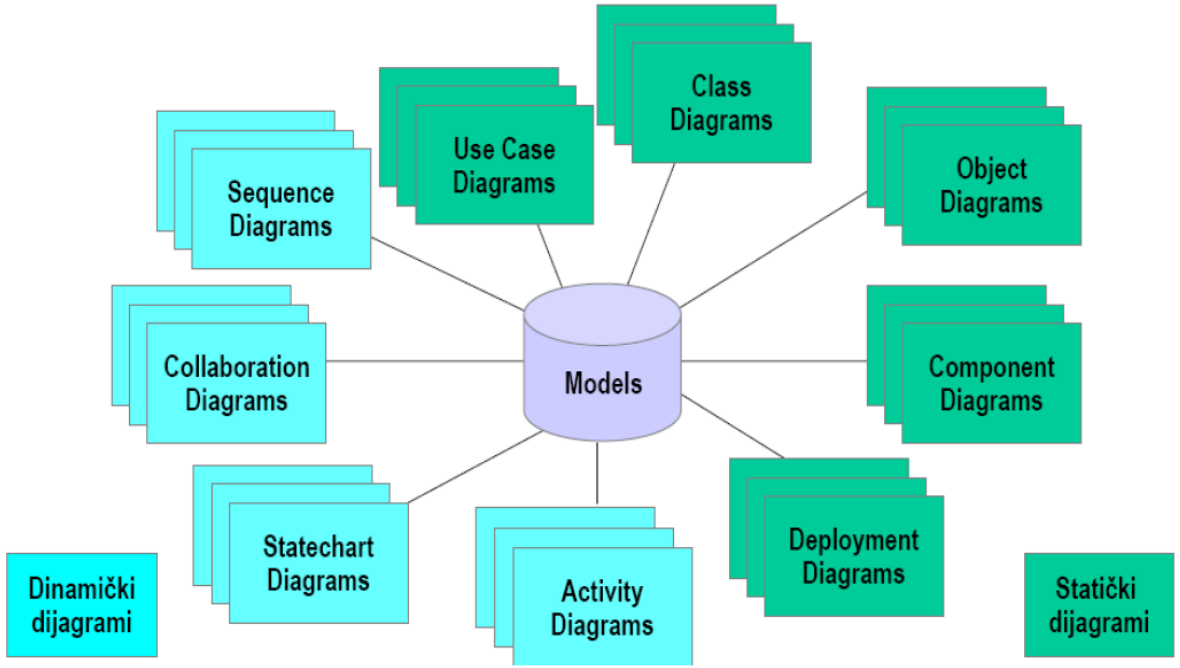
- Postoje četiri vrste stvari:
  - Strukturalne stvari (engl. structural things) su imenice u UML modelima i najčešće su statički dijelovi modela. Predstavljaju konceptualne ili fizičke elemente. U UML-u postoji sedam strukturalnih stvari: *klasa*, interfejs, saradnja, slučaj korištenja, aktivne klase, komponente i čvorovi.
  - Stvari sa ponašanjem (engl. behavioral things) su dinamički dijelovi UML-a. To su glagoli modela i predstavljaju ponašanje kroz prostor i vrijeme. Postoje dva tipa stvari sa ponašanjem: *interakcija* i *automat stanja*.
  - Grupirajuće stvari (engl. grouping things) su organizacijski dio UML-a i postoji samo jedan tip, *paketi*.
  - Označavajuće ili opisne stvari (engl. annotational things) daju neku vrstu objašnjenja i postoji samo jedan tip, *opis (engl. note)*.

UML RELACIJE

- Postoje četiri vrste ***relacija***:
  - Ovisnost (engl. dependency) je semantička relacija između dvije stvari u kojoj promjena u jednoj, neovisnoj stvari, može uticati na semantiku druge ovisne stvari.
  - Asocijacija (engl. association) je strukturalna relacija koja opisuje skup veza između objekata. Kombinacija ili agregacija je specijalni oblik asocijacije i predstavlja strukturalnu relaciju između cjeline i njenih dijelova.
  - Generalizacija (engl. generalization) je relacija specijalizacije/generalizacije u kojoj se objekti specijaliziranih elemenata mogu zamijeniti objektima generaliziranih elemenata.
  - Realizacija (engl. realization) je semantička relacija između klasifikatora, gdje jedan klasifikator specificira ugovor koji drugi klasifikator garantira izvršiti. Realizacija se susreće na dva mjesta: između interfejsa i klasa ili komponenata koje ih realiziraju, i između slučajeva korištenja i saradnji koje ih realiziraju

UML DIJAGRAMI

- Dijagram je grafička prezentacija skupa elemenata, najčešće prikazanih kao povezani graf stvari i relacija. Dijagrami se crtaju da bi se predstavio sistem iz različitih perspektiva, pa to dijagram čini jednom vrstom projekcije sistema.



CASE ALATI ZA MODELIRANJE

- CASE alati pružaju automatsku podršku aktivnostima softverskog procesa.
  - Upper-CASE sistemi za podršku ranim aktivnostima procesa (zahtjevi i dizajn)
  - Lower-CASE sistemi za podršku kasnijim aktivnostima (programiranje, debagiranje i testiranje)
- Primjeri CASE alata koji podržavaju UML modeliranje:
  - **Open ModelSphere -** http://www.modelsphere.com/org/
  - Power Designer
  - Rational Rose
  - Visual Paradigm
  - Architect
  - Star UML


why
when
what
who
where
how
where
what
who
how