

Pregled UML

Iztok Savnik

Uvod

- Standarden jezik za pisanje specifikacij programske opreme.
 - Poslovni informacijski sistemi
 - Porazdeljene spletne aplikacije
 - Vgnezdjeni sistemi v realnem času
- Kreiranje konceptualnega modela UML
 - Tri vrste osnovnih elementov UML
 - 1) Osnovni gradniki UML
 - 2) Pravila, ki povejo kako se gradniki sestavljajo
 - 3) Skupni mehanizmi jezika

Uvod

- UML je neodvisen od vrste procesa, ki ga opisujemo:
 - Proces temelji na primerih uporabe (use-case driven)
 - Arhitekturno usmerjen (architecture-centric) proces
 - Iterativen proces
 - Inkrementalen proces
- UML je jezik za:
 - Vizualizacijo
 - Specifikacijo
 - Konstrukcijo
 - Dokumentacijo

UML je jezik

- Jezik vsebuje slovar besed in pravila za konstrukcijo besed jezika v stavke, ki služijo za komunikacijo.
 - Jezik za modeliranje vsebuje besede in gradnike s katerimi predstavimo konceptualni in fizični ustroj sistema.
- Modeliranje vodi do poznavanja sistema.
 - Običajno potrebujemo več različnih med sabo povezanih modelov za opis sistema.
 - Jezik pove na kakšen način kreiramo dobro-definirane modele sistema.
 - Poznavanje razvoja programske opreme omogoča izbor pravih modelov.

Jezik za vizualizacijo

- Pogled programerja
 - Razmišljanje = implementacija = kodiranje
 - Minimalno in direktno orodje za izražanje algoritmov.
- Problemi „tekstovnega“ modeliranja
 - Komunikacija takšnega modela je naslonjena na pomen, ki je bil zgrajen lokalno v firmi
 - Klasifikacija, porazdelitev modulov, itd. je lažje zapisati grafično–tekstovni model ne shrani te informacije direktno.
- Nekateri modeli je lažje predstaviti grafično druge tekstualno
 - Nekateri strukture programov transcendirajo tekstovno predstavitev
 - Grafični modeli prikražejo prereze, abstrakcije, itd.

Jezik za specifikacijo

- Specifikacija je gradnja modelov, ki so **precizni**, **nedvoumni** in **kompletni**.
- UML je uporabljen kot jezik za specifikacijo:
 - Analize sistema
 - Načrtovanja sistema
 - Implementacije sistema

Jezik za konstrukcijo

- UML ni vizualen programski jezik
- Model zgrajen v UML
 - Prevod v kodo programskega jezika (Java,C++,C#)
 - Prevod v podatkovni model SUPB (relacije,razredi)
 - Grafični modeli uporabljeni za vizualne konstrukcije
 - Tekstovni model uporabljen za algoritmične konstrukcije
- UML omogoča konstrukcijsko in vzvratno inženirstvo
 - **Generiranje kode** iz diagramov in diagramov iz kode
 - Krožno inženirstvo uporablja oboje

Jezik za dokumentacijo

- Softverska firma producira naslednje izdelke:
 - Analiza zahtev, arhitektura, načrt, izvorna koda, projektni plani, testi, prototipi, izdaje, itd.
 - Nekateri izdelki obravnavamo bolj druge manj formalno
- Opisani izdelki so rezultati projekta
 - Izdelki so tudi osnova za komunikacijo, ocene, in kontrolo nad sistemom med razvojem in pri uporabi
- UML se dotika dokumentacije sistema
 - Zraven načrta arhitekture, izvedbenega načrta, itd. vsebuje tudi možnost za dokumentacijo analize zahtev, testov, planiranje projekta in druge.

Konceptualni model UML

- Poznavanje treh osnovnih elementov UML:
 - 1) Osnovni gradniki UML
 - 2) Pravila, ki povejo kako se gradniki sestavljajo
 - 3) Skupni mehanizmi jezika
- Osnovni konceptualni model UML
 - Omogoča branje in pisanje enostavnih UML modelov
 - Osnovna konceptualna struktura, ki jo lahko nadgradimo z izkušnjami in bolj podrobnim poznavanjem posameznih gradnikov

Osnovni gradniki UML

- Slovar UML vsebuje naslednje tri vrste gradnikov
 - **Stvari**—abstrakcije, ki so osnovni koncepti UML
 - **Relacije**—povezujejo osnovne stvari
 - **Diagrame**—grupirajo zanimive kolekcije stvari in relacij

Stvari v UML

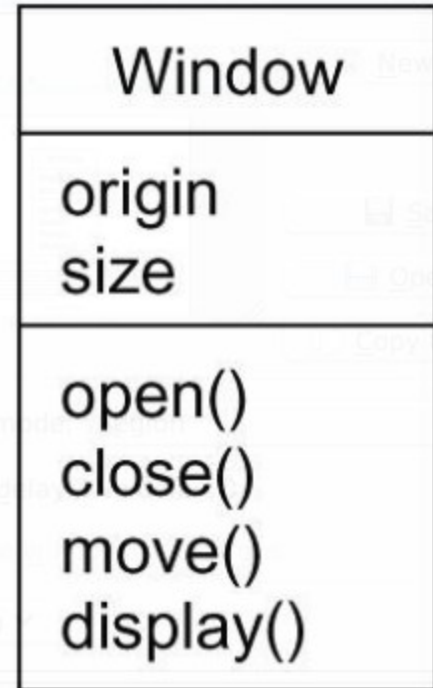
- Štiri vrste stvari v UML
 - Strukturirane stvari
 - Dinamične stvari
 - Skupinske stvari
 - Anotacijske stvari
- To so osnovni objektno-usmerjeni gradniki UML.
- Uporabljamo jih za izgradnjo dobro-definiranih modelov.

Strukturirane stvari

- Strukturirane stvari so osnovni objekti modeliranega okolja.
 - Statičen del modela—konceptualne ali fizične stvari
- Strukturirane stvari:
 - Razred
 - Vmesnik
 - Sodelovanje
 - Primer uporabe
 - Aktivni razredi
 - Komponente
 - Vozlišča

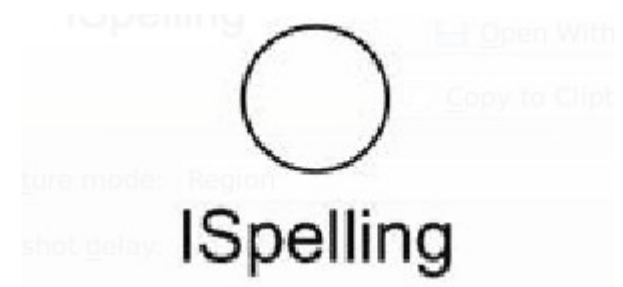
Razred

- Opis množice objektov, ki imajo skupne lastnosti, operacije, razmerja in pomen.
- Ima enega ali več vmesnikov.
- Predstavimo ga s pravokotnikom, ki vsebuje ime, attribute in operacije razreda.



Vmesnik

- Vmesnik je kolekcija operacij, ki deločajo servise razreda.
 - Opisuje obnašanje vidno navzven.
 - Lahko predstavlja kompletno obnašanje ali samo del obnašanja.
 - Vmesnik je torej definiran z množico signatur, ki predstavljajo operacije razreda.
 - Vmesnik ne opisuje implementacije operacij!
- Grafično je vmesnik določen s krogom in pripadajočim imenom.
- Vmesnik je praviloma povezan z razredom.

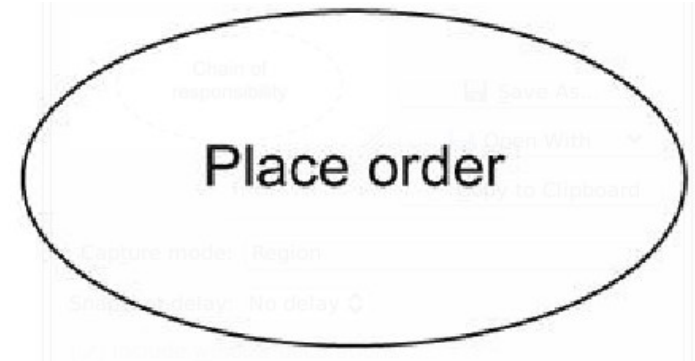


Sodelovanje



- Sodelovanje definira interakcijo v združbi objektov in vlog, ki skupno omogočajo kooperativno obnašanje.
 - Skupno obnašanje je lahko večje od vsote delov.
- **Strukturna in dinamična dimenzija.**
 - Struktura: razredi, objekti
 - Interakcije (proženje metod): sekvence, sodelovanja, stanja
- Razred je lahko del večih struktur sodelovanja.
 - Sodelovanja predstavljajo implementacijo vzorcev, ki sestavljajo sistem.
- Grafično je sodelovanje predstavljeno s črtkanim ovalom, ki vsebuje ime sodelovanja.

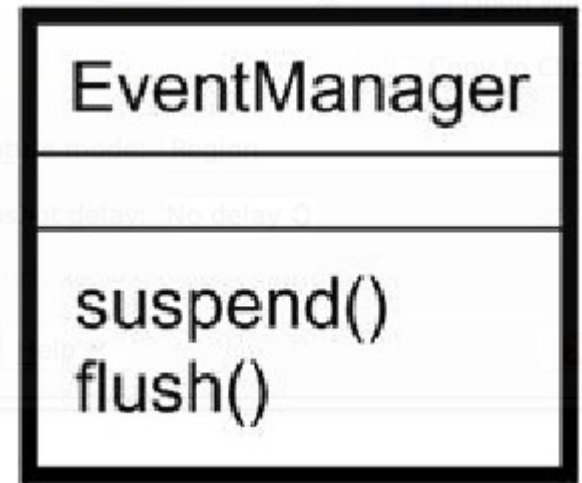
Primer uporabe



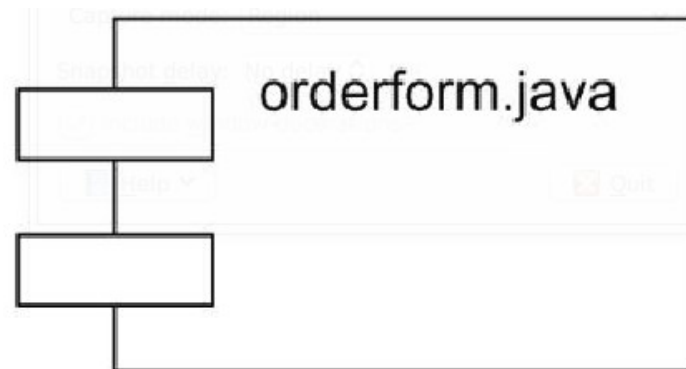
- Primeri uporabe predstavlja opis sekvence akcij, ki jih izvaja sistem.
 - Primer uporabe vodi do rezultata, ki je pomemben za konkretnega akterja.
- **Primeri uporabe** služijo za organizacijo obnašanja danega modela.
- **Primeri uporabe se realizirajo s sodelovanjem.**
- Grafično so primeri uporabe predstavljeni z elipso, ki ima polno črto in vsebuje ime primera uporabe.

Aktivni razredi

- Aktiven razred je razred katerega primerki imajo en ali več **procesov ali niti**.
 - Lahko sprožijo kontrolno aktivnost.
 - Razred, ki se izvaja vzporedno z drugimi aktivnostmi.
- Grafično je aktivni razred predstavljen s provokotnikom, ki ima odebeljeno črto.
 - Razred vsebuje, enako kot navadni razredi, ime, attribute in operacije.

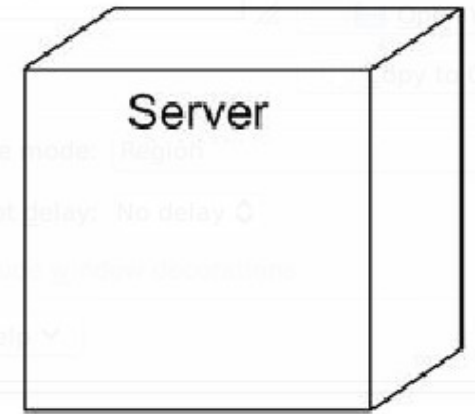


Komponente



- Komponenta je fizični in zamenljiv del sistema, ki vsebuje realizacijo množice vmesnikov.
- Sistem običajno vsebuje
 - različne namestitvene komponente, npr. komp.J.Beans, in
 - komponente v razvoju, npr. datoteke v izvorni kodi.
- Komponente tipično predstavljajo fizično pakiranje logičnih elementov kot so npr. razredi, vmesniki ali sodelovanja.
- Grafično je komponenta predstavljena s pravokotnikom z zavihki, ki ima običajno zapisano samo ime.

Vozlišča



- Vozlišče je fizični element, ki obstaja v realnosti in **predstavlja računski vir**.
 - Vir ima običajno spomin in pogosto zmožnosti računanja.
- Na vozlišču je lahko **množica komponent**, ki se lahko selijo med vozlišči.
- Grafično je vozlišče predstavljeno s kocko, ki običajno vsebuje samo ime.

Dinamične stvari

- Dinamični del UML jezika.
- Dinamične stvari predstavljajo glagole, akcije, interakcije, ... modela.
- Obnašanje v času in prostoru.
- **Dinamične stvari:**
 - Interakcija
 - Stroji stanj

Interakcija



- Obnašanje opisano z množico sporočil med množico objektov v določenem kontekstu definiranim za določeno nalogo.
- Obnašanje združbe objektov ali ene same operacije
 - sporočila—proženje akcij
 - sekvence akcij—obnašanje sproženo s sporočilom
 - povezave—zveze med objekti
- Sporočilo je predstavljeno s puščico, ki je označena z imenom metode.

Stroji stanj



- Stroj stanj definira **sekvenco stanj preko katerih gre objekt ali interakcija** kot odziv na sporočilo.
- Predstavitev obnašanja razreda ali interakcije več razredov s strojem stanj.
- **Stroj stanj** vsebuje:
 - **stanja**—stanja objektov ali interakcije
 - **prehode med stanji**—tok med stanji
 - **dogodke**—s katerimi se prožijo prehodi
 - **aktivnosti**—odzive na prehode med stanji
- Grafično so stroji stanj predstavljeni s pravokotniki z zaokroženimi robovi in imenom.

Skupine stvari



- Skupine stvari se uporabljajo za **organiziranje modela v dele—škatle**, ki vsebujejo komponente modela.
- Osnoven **mehanizem za tvorjenje skupin so paketi**.
 - Vsebujejo strukturirane stvari, dinamične stvari in druge skupine stvari.
 - Za razliko od **komponent**, ki obstajajo v času izvajanja sistema, je **paket** samo konceptualna entiteta, ki obstajajo samo v času razvoja.
 - **Osnoven mehanizem za organizacijo UML modela**.
 - Variante paketov: ogrodja, modeli in pod-sistemi.
- Grafično so paketi predstavljeni kot mape z zavihki.

Anotacijske stvari

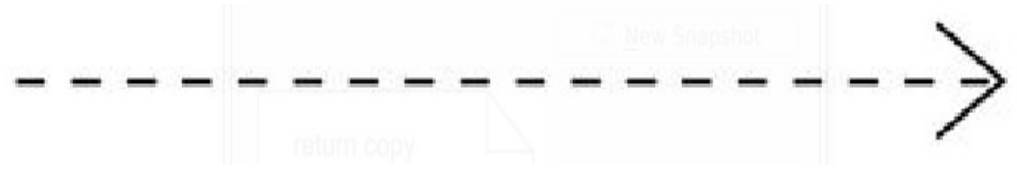


- Z anotacijskimi stvarmi **pojasnjujemo dele UML modela.**
 - Komentarji s katerimi opišemo, osvetlimo in dokumentiramo dele modela.
- Osnovna anotacijska stvar je **beležka.**
 - Beležka je medij za **komentiranje elementa ali množice elementov** UML modela.
 - Beležke opisujejo omejitve, implementacijske podrobnosti in komentarje, ki so predstavljeni v naravnem jeziku.
- Grafično je beležka predstavljena kot pravokotnik z upognjenim vogalom.

Relacije v UML

- Imamo štiri vrste relacij:
 - Odvisnost
 - Asociacija
 - Generalizacija
 - Realizacija
- Relacije so osnovni gradniki UML.
- Z relacijami predstavimo dobro-definirane modele.

Odvisnost



- Odvisnost je **semantično razmerje** med dvema stvarmi, kjer sprememba na prvi stvari vpliva na pomen druge (odvisne) stvari.
- Grafično je odvisnost označena s črtkano črto, ki se lahko zaključi s puščico in ima lahko ime.

Asociacija

0..1

*

employer

employee

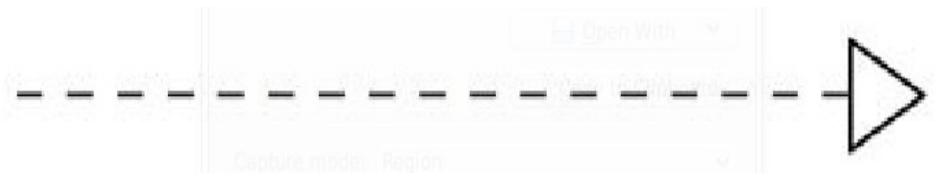
- Asociacija je **strukturno razmerje**, ki opisuje množico zvez med objekti.
- **Agregacija** je poseben primer asociacije.
 - Predstavlja strukturno razmerje med celoto in deli.
- Grafično je asociacija predstavljena s polno črto, ki se lahko zaključi s puščico.
 - Asociacija je lahko označena z **vlogami** in dodatnimi simboli kot je npr. števnost.

Generalizacija



- Generalizacija je **razmerje med razredi** s katerim opišemo povezavo med razredom in njegovo specializacijo/generalizacijo.
 - Objekti danega razreda so **zamenljivi** s primerki bolj specifičnih razredov.
 - Pod-razred **podeduje** vse lastnosti nad-razreda.
- Grafično je generalizacija predstavljena s črto, ki se zaključi v polni puščici in kaže na starša oz. generalizacijo.

Realizacija



- Realizacija je **semantično razmerje med moduli**, kjer ima modul pogodbo z drugim modulom za katero garantira izvedbo.
- Razmerje realizacije najdemo na **dveh mestih**:
 - Med vmesniki in pripadajočimi razredi ali komponentami, ki jih realizirajo.
 - Med primeri uporabe in sodelovanji s katerimi so realizirani.
- Grafično je realizacija predstavljena s črto, ki se zaključi s puščico, ki ima prazno glavo.

Diagrami v UML

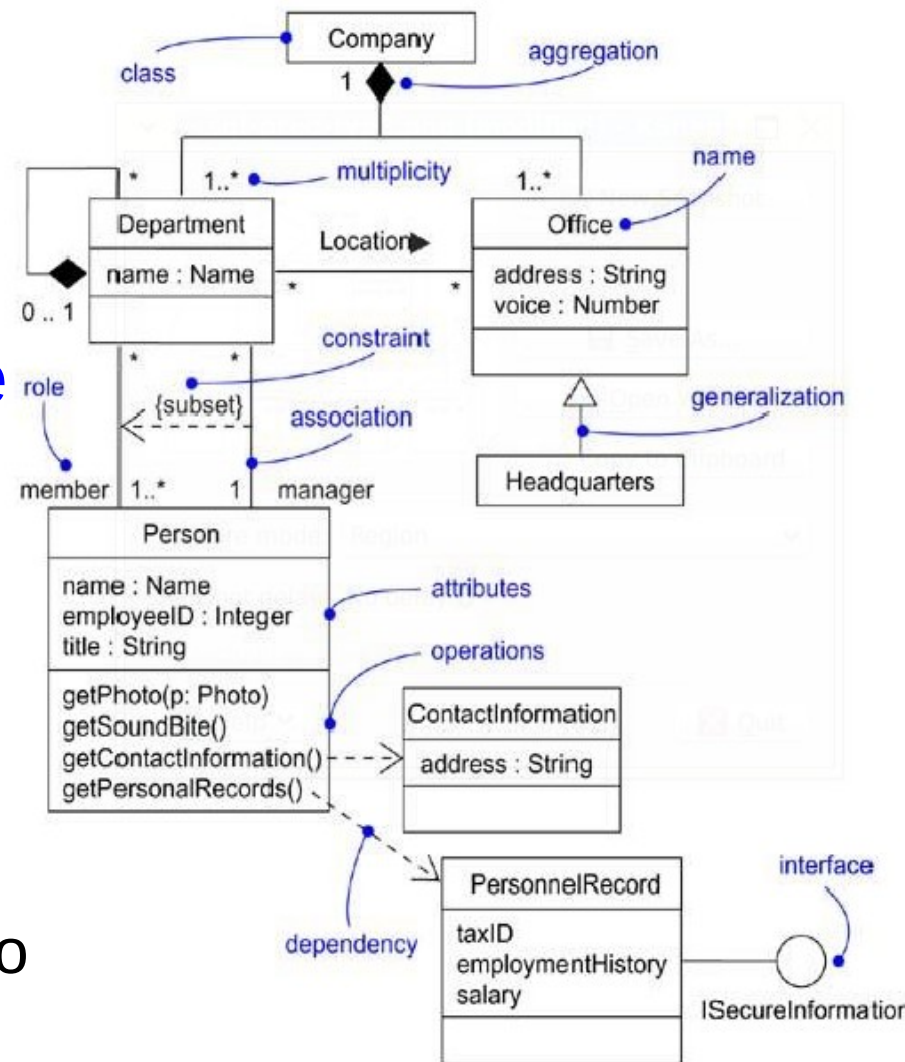
- Diagram je grafična predstavitev množice med sabo povezanih elementov.
 - Graf vozlišč, ki predstavljajo stvari, in povezav s katerimi
- Diagrame rišemo zato, da predstavimo sistem iz različnih perspektiv.
 - Diagram je **projekcija v sistem**—abstrakten vpogled do elementov predstavljenih z danim diagramom.
 - Element se lahko pojavlja v vseh diagramih, v samo nekaterih ali v izjemnih primerih v nobenem diagramu.
 - Diagram v splošnem lahko **vsebuje poljubne stvari in relacije** med njimi—v realnosti uporabljamo bolj majhno število kombinacij.

Vrste diagramov

- 1) Diagram razredov
- 2) Diagram objektov
- 3) Diagram primerov uporabe
- 4) Diagram sekvenc
- 5) Diagram sodelovanja
- 6) Diagram stanj
- 7) Diagram aktivnosti
- 8) Diagram komponent
- 9) Diagram postavitve

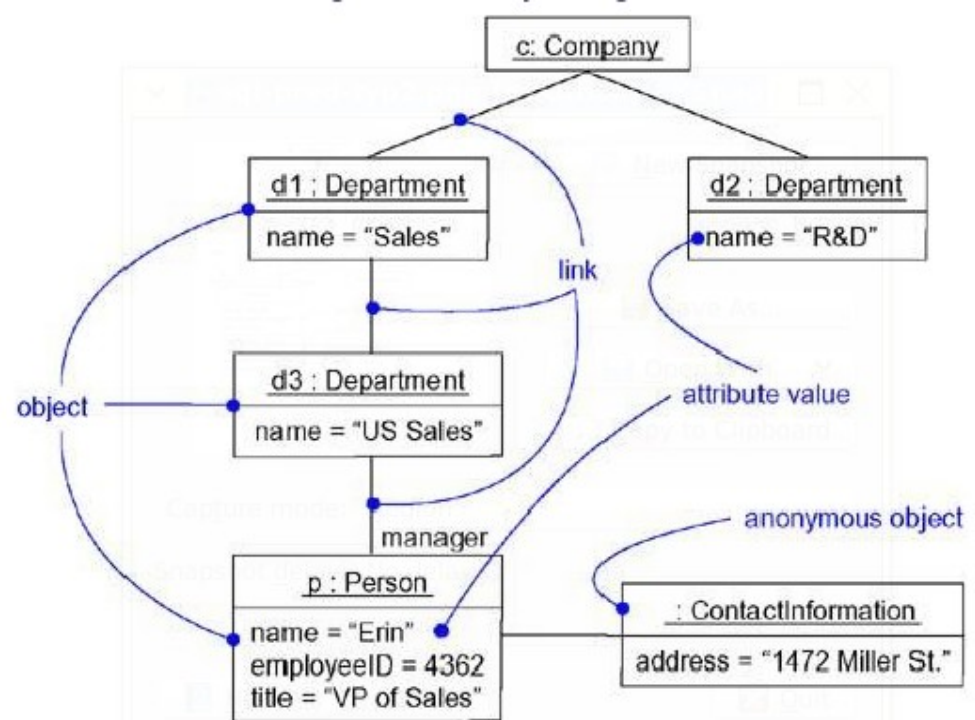
Diagram razredov

- Diagram razredov prikaže množico razredov, vmesnikov, sodelovanje in relacije med temi.
 - Najbolj pogosti diagram v OO sistemih.
 - Diagrami razredov opisujejo **statičen načrt sistemov**.
- Diagrami, ki vsebujejo tudi aktivne razrede predstavljajo statičen procesni pogled sistema.



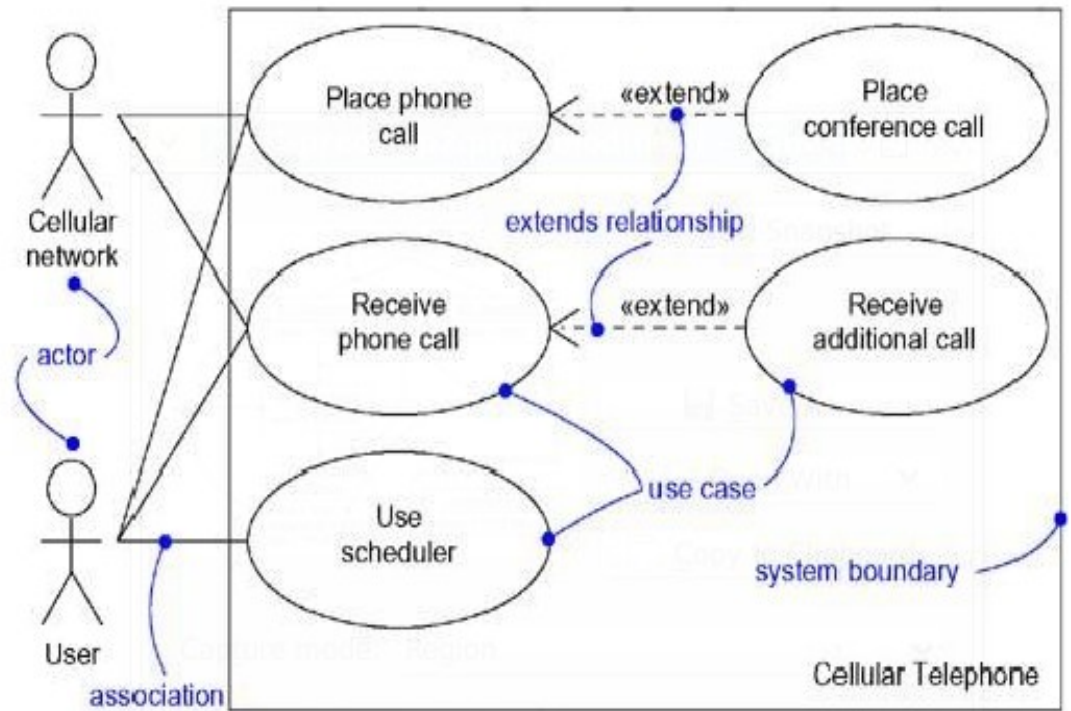
Diagrami objektov

- Diagrami objektov predstavljajo množico objektov in razmerja med njimi.



- Diagrami objektov predstavljajo statično sliko primerkov stvari, ki jih najdemo v diagramih razredov.
- Te diagrami predstavljajo **statičen načrtovalski ali procesni pogled sistema** iz perspektive realnih prototipnih primerov.

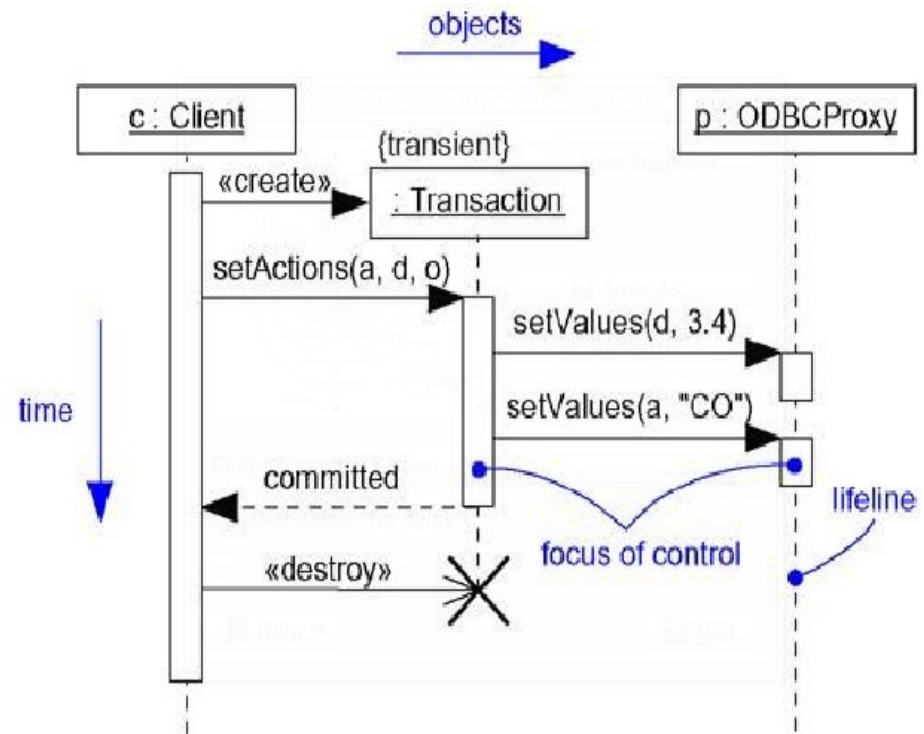
Diagrami primerov uporabe



- Diagrami prikazujejo primere uporabe skupaj z akterji in povezavami med njimi.
- Diagrami predstavljajo **statičen pogled primerov uporabe** sistema.
- Te diagrami so posebj pomembni za organizacijo in modeliranje obnašanja sistema.

Diagrami interakcij

- Diagrami sekvenc in diagrami sodelovanja so primeri diagramov interakcije.



- Prikažejo interakcijo med objekti iz množice, njihovimi razmerji, skupaj s sporočili, ki se jih lahko razpošlje med objekti.
- Diagrami sekvenc prikažejo interakcijo, kjer je poudarjena časovna ureditev sporočil.

Diagrami interakcij

- **Diagrami sodelovanja** prikažejo interakcijo, kjer je poudarjena struktura objektov, ki sodelujejo.
- Diagrami sekvenc in diagrami sodelovanja so izomorfni.

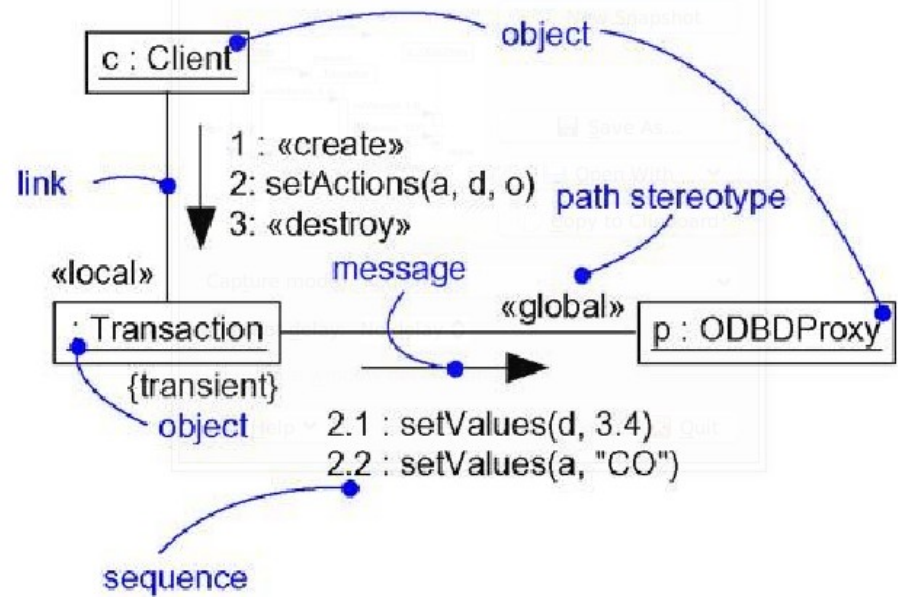
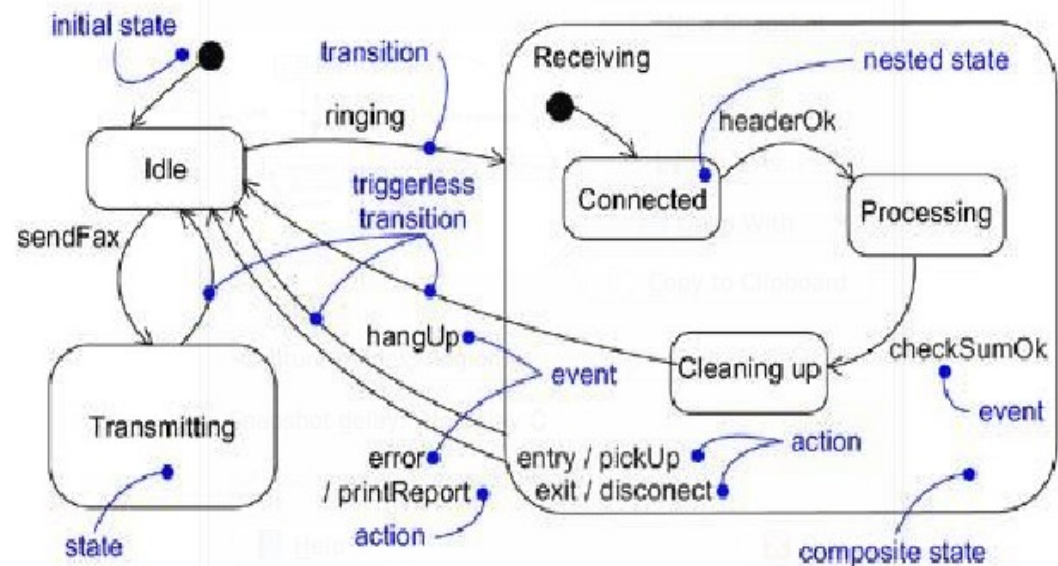


Diagram stanj

- Predstavlja stroj, ki temelji na stanjih, prehodih med stanji, dogodkih in aktivnostih.
- Diagram stanj predstavlja **dinamičen pogled na sistem**.
- Te diagrami so pomembni za modeliranje obnašanja vmesnika, razreda in sodelovanja med razredi.
- Poudarek je na **dogodkovno urejenem obnašanju objektov**.
- Osnovani na Moore in Mealy končnih avtomatih.
- Uporabno za modeliranje reaktivnih sistemov.



Diagrami aktivnosti

- Diagram aktivnosti je poseben primer diagrama stanj, ki prikaže tok izvajanja od aktivnosti do aktivnosti.
 - Diagram poteka izvajanja!
 - Stanja, odločitvene operacije, spreminjanje stanja, razvejitev, vzporednost, ...
 - Diagram aktivnosti predstavlja dinamičen pogled na sistem.
- Pomembni so za modeliranje funkcij sistema in poudarjajo tok kontrole med objekti sistema.

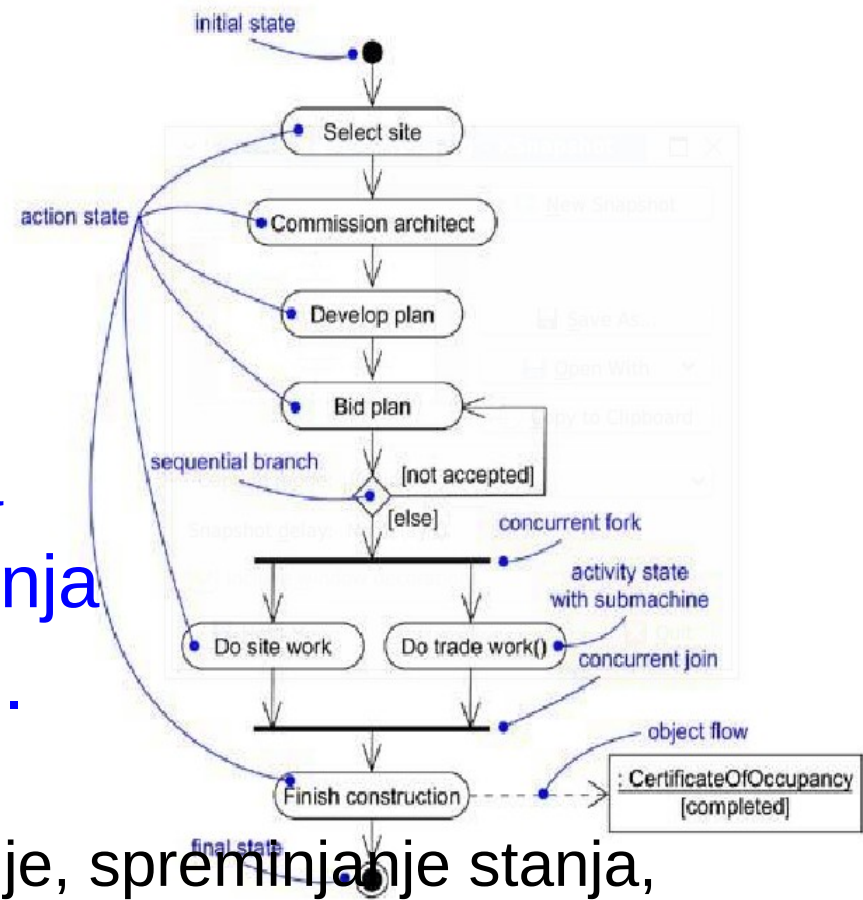


Diagram komponent

- Prikaže organizacijo in odvisnosti množice komponent.
- Diagrami komponent prikažejo **statičen implementacijski pogled** na sistem.
- Komponente so tipično blizu diagramom razredov.
- Vsebujejo množico razredov, vmesnikov in opisov sodelovanja.

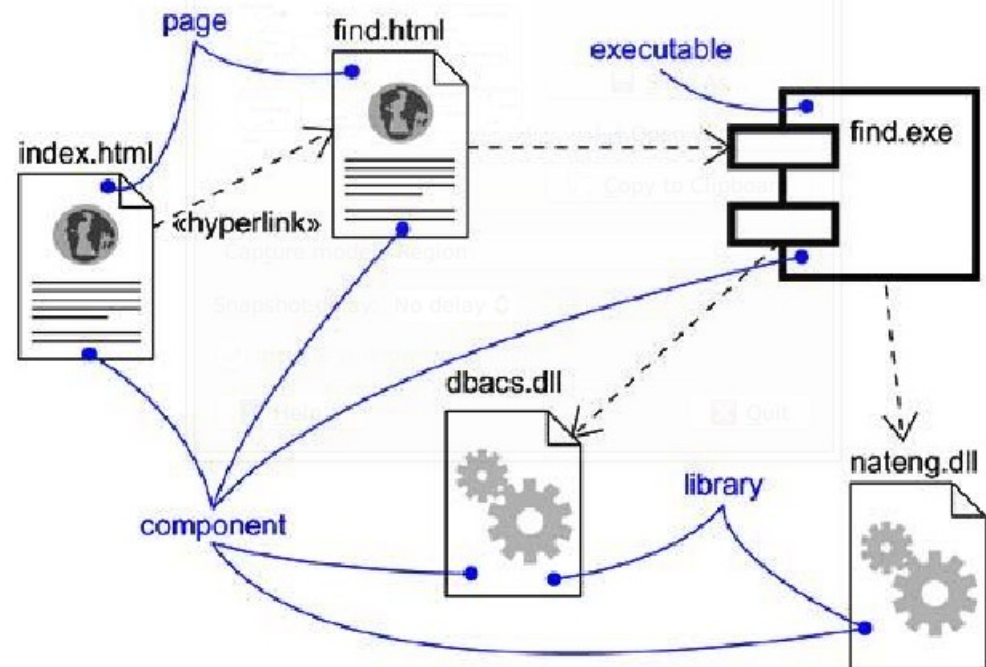
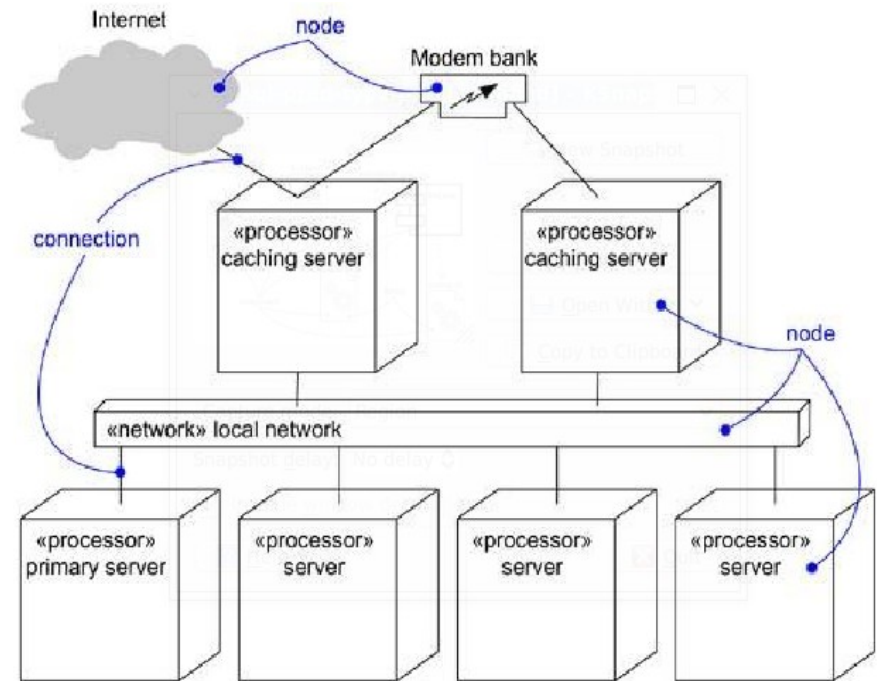


Diagram postavitev

- Prikaže **konfiguracijo** izvajalnega vozlišča in **postavitev** komponent, ki se izvajajo na vozliščih.
 - Diagram postavitve predstavlja **statičen pogled postavitve** dane arhitekture.
 - Diagram postavitve je soroden diagramu komponent.
- Vozlišče tipično vsebuje eno ali več komponent.



Pravila UML

- Gradniki UML se ne morejo sestavljati poljubno.
- Imamo množico **pravil**, ki povedo kako lahko sestavljamo gradnike, da dobimo dobro-definiran načrt:
 - **Semantično samo-konsistenten in v soglasju z vsem preostalimi modeli.**

Pravila UML

- UML ima semantična pravila za:
 - Imena
 - Kako imenujemo stvari, razmerja in diagrame
 - Obseg
 - Kontekst, ki daje specifičen pomen imenu.
 - Vidnost
 - Kako so ta imena vidna in dostopna iz strani drugih.
 - Skladnost
 - Kako so ta imena skladna in konsistentna z drugimi imeni.
 - Izvajanje
 - Pomen izvajanja ali simuliranja dinamičnega modela.

Pravila UML

- Modeli zgrajeni pri razvoju večjega programskega sistema se **razvijajo**.
 - Različni razvijalci jih vidijo različno v različnem času.
- Zaradi tega vidimo modele tudi kot:
 - Abstraktne
 - Nekateri elementi so skriti za poenostavitev pogleda.
 - Nepopolne
 - Nekateri elementi manjkajo.
 - Nekonsistentne
 - Integriteta modelov ni zagotovljena.
- Pravila vzpodbujajo (ne silijo) k uporabi najpomembnejših analiz, načrtov, itd. ki vodijo do definicije dobro-definiranih modelov.

Skupni mehanizmi UML

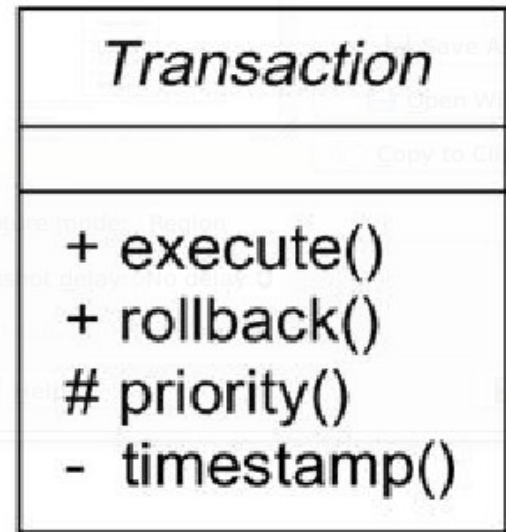
- Zgradba je enostavnejša in bolj harmonična, če se uporablja skupne vzorce:
 - Viktorijanski stil, Francoski podeželski stil, Istrski stil, itd.
 - Stil dosežemo z uporabo primernih vzorcev.
- Podobno je pri programski opremi.
- Stil izgradnje programskega sistema dosežemo tudi z naslednjimi:
 - Specifikacije
 - Podrobnosti razreda
 - Pogoste delitve
 - Mehanizmi razširljivosti

Specifikacije

- UML ni samo grafični jezik.
 - Za vsako grafično notacijo je specifikacija, ki podaja tekstoven opis grafičnega koncepta.
 - Npr. za razredom je specifikacija atributov, operacij, vključno s signaturami atributov in operacij, skupaj z obnašanjem.
- Grafična notacija je za vizualizacijo sistema in UML specifikacija je za predstavitev podrobnosti.
 - S takšno razdelitvijo je možno graditi model inkrementalno:
 - 1) risanje diagramov in
 - 2) dodajanje pomena s pomočjo specifikacij.
- UML specifikacije nudijo semantično ozadje, ki vsebuje vse dele sistema.
 - Vsi deli so povezani v konsistentno celoto.
- UML diagrami so samo vizualne projekcije specifikacij.

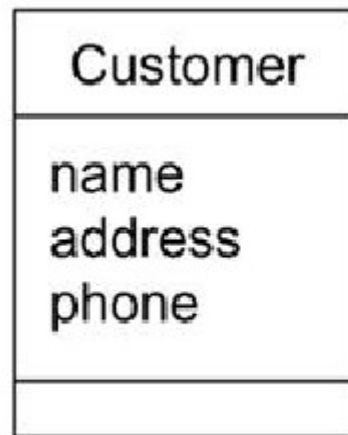
Podrobnosti razreda

- UML gradniki so predstavljeni z različnimi simboli, ki predstavljajo osnovno abstrakcijo.
 - Na primer, razred je opisan z imenom, atributi in metodami.
 - Na nivoju diagrama predstavljeni podatki razreda zadoščajo.
- Vsak gradnik UML ima lahko več podrobnosti.
 - Grafično so podrobnosti predstavljene kot dodatni simboli, ki bolj natančno definirajo gradnik ali povezave gradnika z okolico.
 - Načrtovalska orodja omogočajo urejanje dodatnih simbolov.



Pogoste delitve

- Pri modelirnju objektno-usmerjenih sistemov se svet običajno deli na več načinov.
 - Delitev med razredi in objekti.
 - Z UML lahko modeliramo razrede in objekte.
- Vsi drugi gradniki vsebujejo neke vrste razcepom med konkretnim in abstraktnim.
 - Primer uporabe /
 instanca p.uporabe
 - Komponente /
 instance komponente



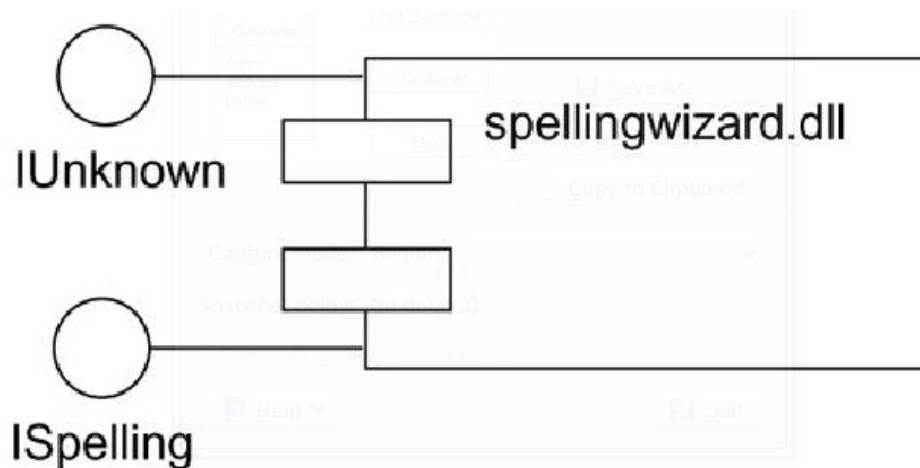
Jan : Customer

: Customer

Elyse

Pogoste delitve

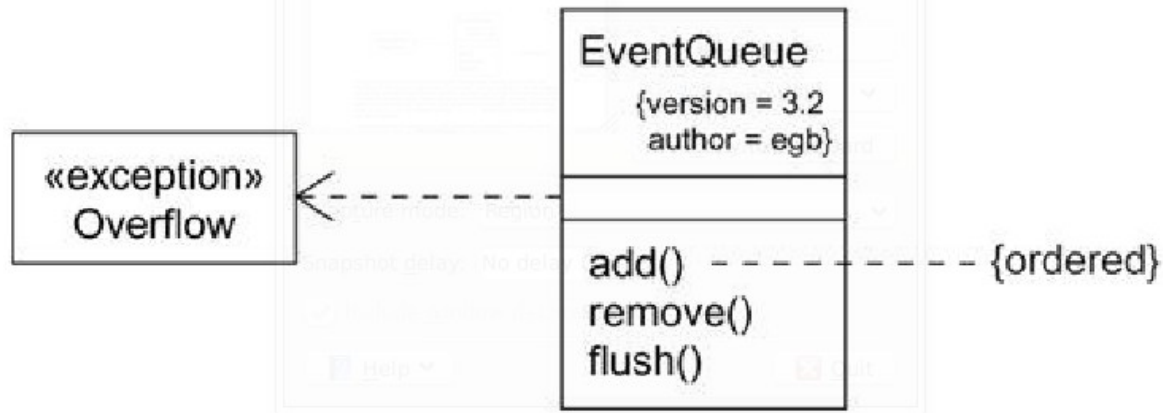
- Druga pogosta delitev je med vmesnikom in implementacijo.
 - Vmesnik je pogodba, ki mora biti izpolnjena.
 - Implementacija predstavlja eno konkretno izvedbo pogodbe.
- Vsak UML gradnik ima delitev na vmesnik in implementacijo
 - Vmesnik razreda / implementacija razreda
 - Vmesnik komponente / implementacija komponente
 - Primeri uporabe / sodelovanja



Mehanizmi razširljivosti

- UML je standarden jezik za pisanje načrtov programskih sistemov.
 - Ni možno, da bi lahko z enim jezikom opisali vse možne gradnike vseh modelov v vsakem času.
- UML je odprt in omogoča razširitve jezika.
 - Stereotipi
 - Označene vrednosti
 - Omejitve

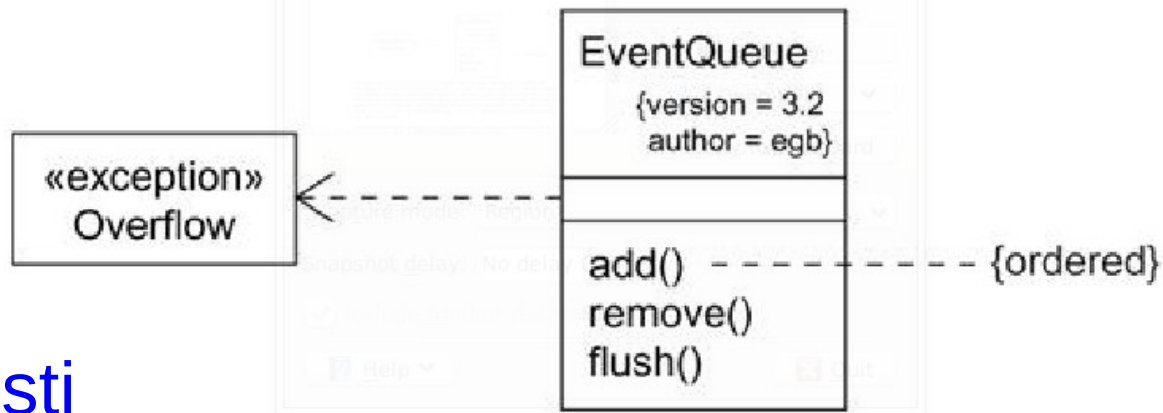
Mehanizmi razširljivosti



- Stereotipi

- Stereotipi razširijo slovar UML tako da dovolijo definicijo novih gradnikov iz obstoječih.
- Recimo, da delamo v konkretnem programskem jeziku npr. Java ali C++.
 - Potrebujemo izjeme.
 - Izjeme so razredi, ki so obravnavani na poseben način.
 - Izjeme lahko definiramo kot gradnik, ki je označen s stereotipom.

Mehanizmi razširljivosti



- **Označene vrednosti**

- Lahko razširijo lastnosti gradnika UML.
 - Dodamo nove informacije obstoječim gradnikom.
- Recimo, da bi želeli dodati verzijo in avtorja vsem ključnim abstrakcijam sistema.
 - Lahko uporabimo označene vrednosti.
 - Glej primer zgoraj.

- **Omejitve**

- Razširitev pomena obstoječega gradnika tako, da dodamo nova pravila.
- Na primer, razred `EventQueue` je omejen tako, da dodajmo nove dogodke v vrstnem redu proženja.

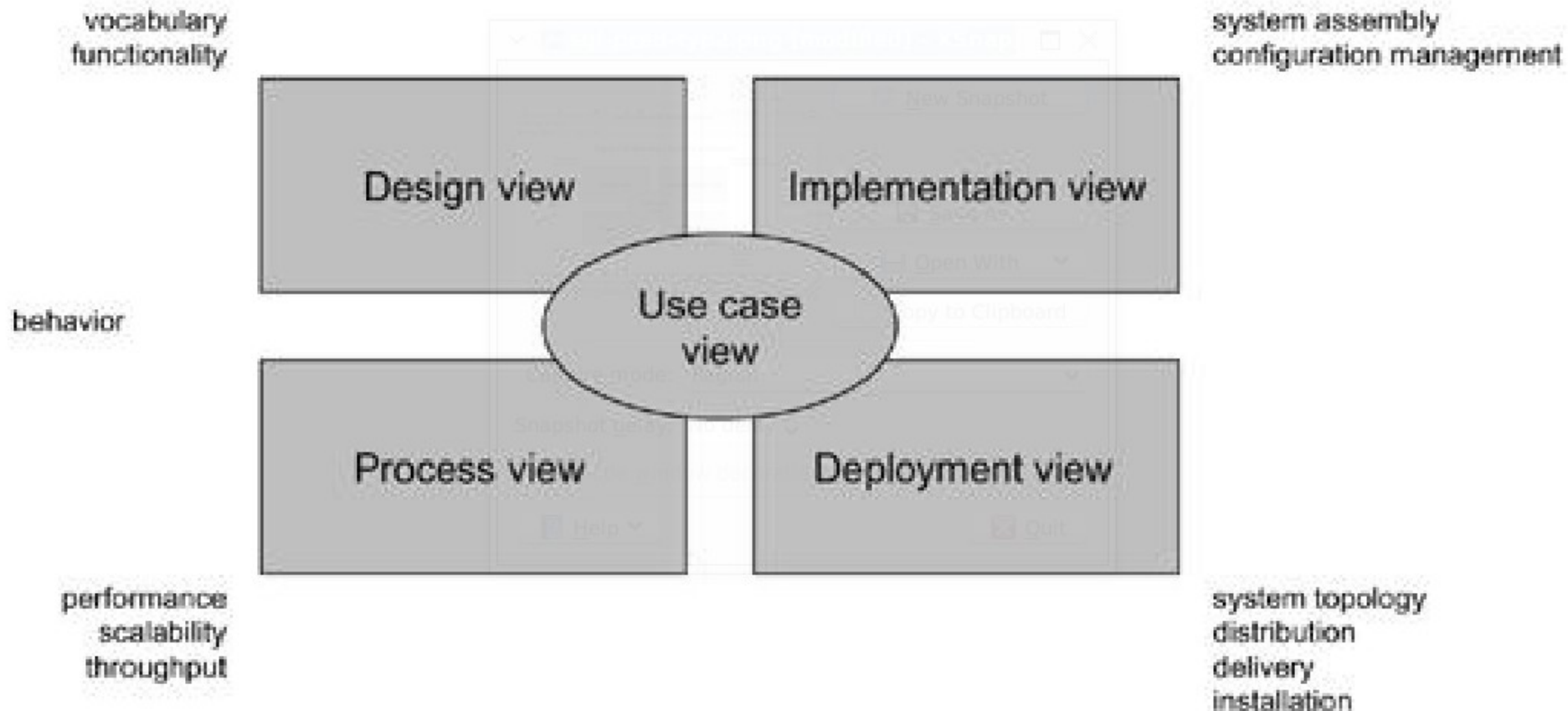
Arhitektura sistema

- Vizualizacija, specifikacija, konstrukcija in dokumentacija programskega sistema zahteva možnost pogleda na sistem iz različnih perspektiv.
 - Končni uporabniki, analitiki, sistemski integratorji, pisci dokumentacije, projektni vodje in drugi, vsak vidi projekt iz svojega vidika.
- Arhitektura sistema je najbolj pomemben načrt sistema, ki omogoča upravljanje z različnimi pogledi.
 - Omogoča kontrolo iterativnega in inkrementalnega procesa razvoja v kompletnem razvojnem ciklu sistema.

Arhitektura sistema

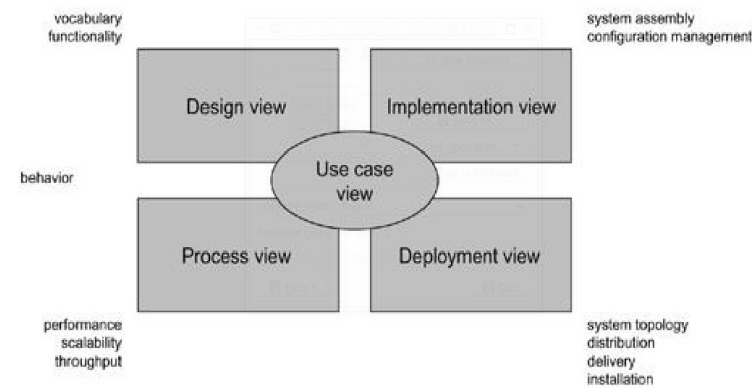
- Arhitektura sistema vsebuje množico pomembnih odločitev o organizaciji prog.sistema.
 - Izbor strukturnih elementov in njihovih vmesnikov, ki sestavljajo programski sistem.
 - Obnašanje specificirano s sodelovanji med elementi.
 - Kompozicija strukturnih in dinamičnih elementov v sestavljene pod-sisteme.
 - Arhitekturni stil, ki vodi organizacijo: statični in dinamični elementi, njihovi vmesniki, njihova sodelovanja in sestavo.
- Arhitektura programske opreme se ne dotika samo strukture in obnašanja ampak vsebuje tudi:
 - Uporabo, funkcionalnost, performanse, ponovno uporabo, kompletnost in tehnologijo.

Arhitektura sistema

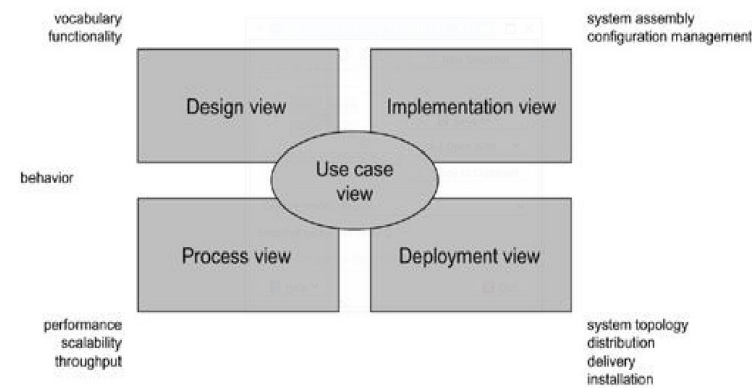


Pogled primerov uporabe

- **Primeri uporabe**, ki opisujejo pogled na obnašanje sistema kot ga vidijo uporabnik, analitik in testiranje.
 - Ta pogled ne specificira organizacije sistema.
 - Predstavlja sile, ki vodijo oblikovanje arhitekture sistema.
 - Statični aspekti tega pogleda so predstavljeni s primeri uporabe.
 - Dinamični aspekti so predstavljeni z diagrami interakcije, diagrami stanj, in diagrami odvisnosti.

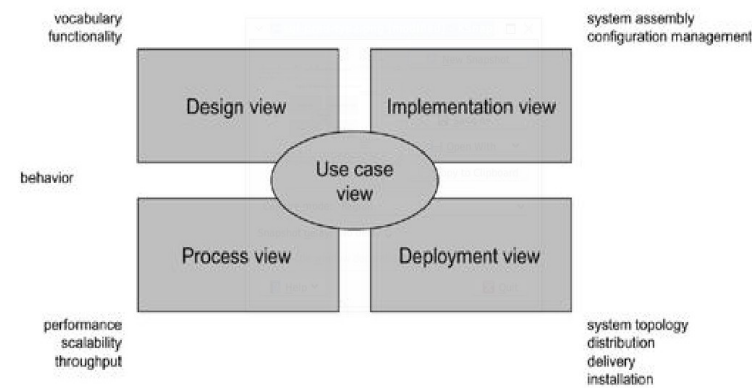


Pogled načrta



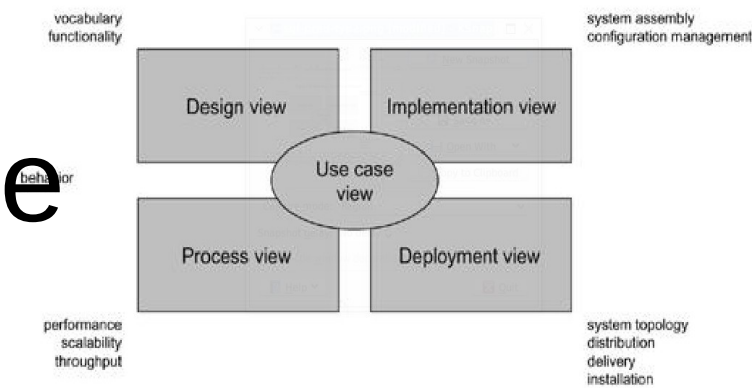
- **Načrt sistema** predstavlja pogled, ki vsebuje razrede, vmesnike in sodelovanja, ki tvorijo slovar problema in njegovo rešitev.
 - Predstavlja funkcionalne zahteve sistema.
 - Servisi, ki jih sistem nudi končnim uporabnikom.
 - Statični aspekt tega pogleda je definiran z razredi.
 - Dinamični aspekt pogleda je definiran z diagrami interakcije, diagrami stanj in aktivacijskimi diagrami.

Pogled procesov



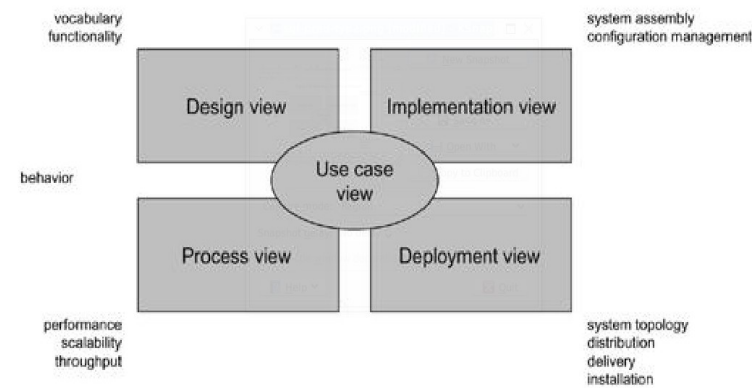
- **Procesni pogled** opisuje niti in procese, ki sestavljajo sistem, vzporednost in sinhronizacijske mehanizme.
 - Ta pogled se ukvarja s performansami sistema ter s skalabilnostjo in prehodnostjo sistema.
 - V UML je statični in dinamični aspekt tega pogleda pokrit z istimi diagrami kot pogled načrta:
 - diagrami razredov in diagrami objektov ter
 - diagrami interakcij, diagrami stanj in diagrami aktivnosti.
 - Fokus je na aktivnih razredih, ki predstavljajo niti in razrede.

Pogled implementacije



- **Implementacijski pogled** vsebuje komponente in datoteke, ki so potrebne za izgradnjo programskega sistema.
 - Ta pogled se ukvarja s konfiguracijami sistema in z upravljanjem verzij sistema, ki so sestavljene iz množice med sabo odvisnih komponent in datotek.
 - Statični aspekti tega pogleda so definirani z diagrami komponent, diagrami stanj in diagrami aktivnosti.

Pogled postavitve



- **Postavitveni pogled** se ukvarja z vozlišči, ki tvorijo strojno opremo na kateri se sistem izvaja.
 - Ta del se ukvarja z pripravo in namestitvijo delov, ki tvorijo celoten fizičen sistem.
 - Statičen aspekt tega pogleda je pokrit z diagrami postavitve.
 - Dinamičen del pogleda je pokrit z diagrami interakcije, diagrami stanj, in diagrami aktivnosti.

Literatura

- Grady Booch, James Rumbaugh, Ivar Jacobson, The Unified Modeling Language User Guide, 2000, Addison Wesley