SISTEMSKA INFORMATIKA IN LOGISTIKA

predmet izbirnega modula študijskega programa 2. stopnje smer Avtomatika in informatika

izr. prof. dr. Gašper Mušič

Cilji izbirnega modula

- Predstavitev problematike vodenja naprednih tehnoloških sistemov
  - področje operativnega upravljanja proizvodnih procesov
  - industrijska informatika
  - metode inteligentnega vodenja
- Pridobitev osnovnih znanj o
  - upravljanju kompleksnih sistemov
  - planiranju v proizvodnji in logistiki
  - planiranju in vodenju projektov
  - uporabi metod umetne inteligence v vodenju
    - pametne naprave, stroji in sistemi

Pregled predmetov

LMSV
LAMS

- Proizvodni management
  - izvedbeno naravnan pogled na tehnike operativnega upravljanja
  - celovit pregled področja s poudarkom na načrtovalskih metodah in strategijah
- Sistemska informatika in logistika
  - navezava na Proizvodni management
  - podrobnejša obravnava izbranih vsebin
- Seminar iz inteligentnega vodenja
  - poglobljeno samostojno projektno delo

Vsebine predmetov
- Proizvodni management

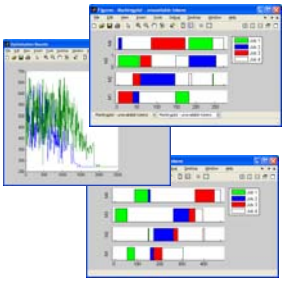
LMSV
LAMS

1. Uvod v operativno upravljanje proizvodnje
2. Sistemi za planiranje in vodenje proizvodnje
   ERP, MRP II, MES
3. Kompleksni krmilni sistemi
   načrtovanje, modularni pristopi, varnost
4. Kazalniki učinkovitosti
   mere učinkovitosti, sistemi za merjenje učinkovitosti
5. Uporabniški vmesniki
   SCADA, spletne tehnologije

Vsebine predmetov
- Sistemska informatika in logistika

LMSV LAMS

1. Uvod v proizvodno informatiko
2. Podatkovne baze
   poizvedbe in obdelava podatkov (T-SQL)
3. Diskretno-dogodkovna simulacija
   simulacijska orodja (SimEvents, Arena)
4. Planiranje in vodenje projektov
   načela planiranja in vodenja projektov ter pripadajoča programska orodja
5. Planiranje in razvrščanje opravil
   diskretna optimizacija, uporaba inteligentnih metod (hevristične metode, genetski algoritmi)

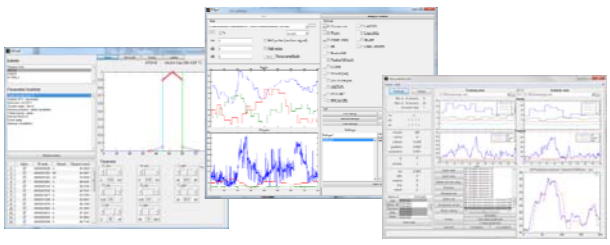


Vsebine predmetov
- Seminar: Inteligentno vodenje

LMSV LAMS

Osnovni cilji predmeta

- Seminar združuje znanja strokovnih predmetov celotnega študija in omogoča študentu **poglobljeno samostojno projektno delo na različnih področjih**.



Način dela

LMSV LAMS

- Predavanja
  - interaktivno delo, primeri uporabe programskih orodij
  - ponazoritev obravnavane snovi s praktičnimi primeri
- Vabljena predavanja
  - predstavnikov vodilnih slovenskih podjetij na področju avtomatike in industrijske informatike
  - tujih strokovnjakov
- Laboratorijske vaje
  - delo s programskimi orodji (SQL, MES, SCADA, SimEvents …)
  - seminar v obliki raziskovalnih/projektnih nalog

Literatura

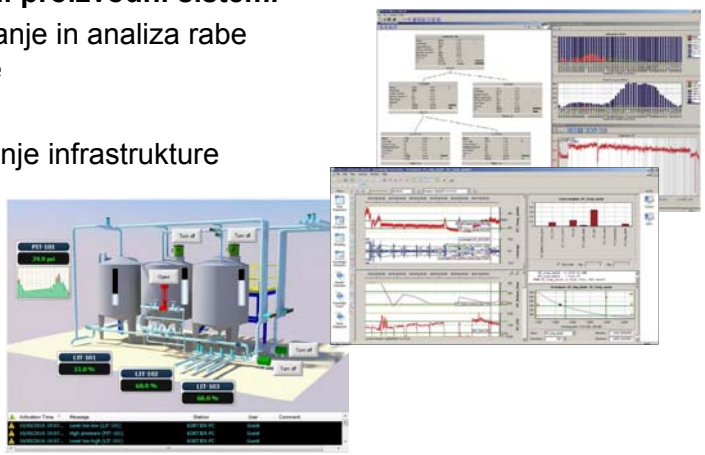
LMSV LAMS

- T. Mohorič, Podatkovne baze, Bi-tim, 2002
- T. O.Boucher, A. Yalçin, Design of industrial information systems, Academic Press, 2006
- T. Altiok, B. Melamed, Simulation Modeling and Analysis with ARENA, Academic Press, 2007
- M. L. Pinedo, Planning and Scheduling in Manufacturing and Services, Springer, 2009

Poudarki

LMSV
LAMS

- Informacijska podpora vodenju in upravljanju
  - **sodobni proizvodni sistemi**
  - spremljanje in analiza rabe energije
  - promet
  - upravljanje infrastrukture



1. Uvod v proizvodno informatiko

LMSV
LAMS

- Kaj je proizvodna informatika?
- Sistemski pristop k načrtovanju proizvodnih informacijskih sistemov
- Modeli v proizvodni informatiki
  - Avtomati, Petrijeve mreže
  - Diagrami stanj (Statecharts)
  - Poenoteni jezik modeliranja (angl. Unified Modelling Language – UML)

1.1 Kaj je proizvodna informatika?

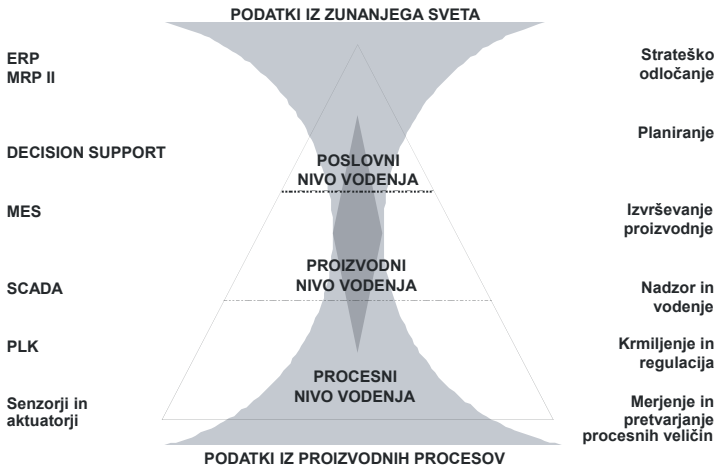
LMSV LAMS

- Informacijski sistemi v podjetjih
  - poslovni informacijski sistemi
  - sistemi za neposredno vodenje industrijskih procesov
  - vrzel operativnega upravljanja proizvodnje

Proizvodna informatika obravnava informacijske sisteme in metode, ki zapolnjujejo vrzel med sistemi za podporo poslovanja in neposrednim vodenjem procesov

Povezava poslovnega in procesnega nivoja

LMSV LAMS



Proizvodna informatika

- Glavne funkcije
  - nadzor in spremljanje proizvodnih virov
  - nadzor in spremljanje proizvodnega procesa
  - zagotavljanje sledljivosti izdelkov
  - informatizacija procesa planiranja
- Napredne funkcije
  - analiza arhiviranih podatkov
  - simulacija
  - podpora odločanju

1.2 Sistemski pristop k načrtovanju

- Sistemski pristop
  - reševanje problemov preko snovanja rešitev, ki izhajajo iz opredeljenih potreb in namena
  - izhajamo iz celovitega sistema, ne iz komponent
  - možne popolnoma nove, od obstoječih različne rešitve
- Uporaben tudi pri načrtovanju informacijskih sistemov
  - večinoma po meri narejeni sistemi
  - predvsem pomembno pri sistemih, ki delujejo v tesni povezavi s fizičnimi procesi

Sistemski pristop

LMSV LAMS

- Konvencionalni pristop
  - zbiranje informacij in dejstev
  - formulacija problema
  - raziskava alternativnih rešitev
  - izbira rešitev
  - podrobneje razdelana rešitev
  - implementacija rešitve
- Sistemski pristop
  - opredelitev potreb in namena
  - generiranje alternativ, ki ustrezajo namenu
  - opredelitev cilja, ki ga je možno realizirati
  - razvoj rešitve
  - implementacija rešitve

sestavni deli, komponente

komplementarnost

sistemi

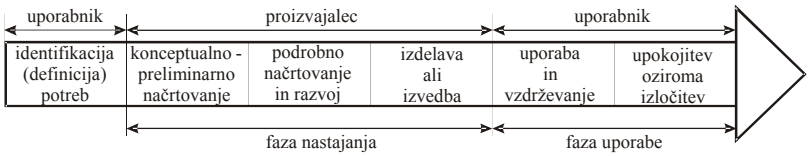
Sistemsko inženirstvo

LMSV LAMS

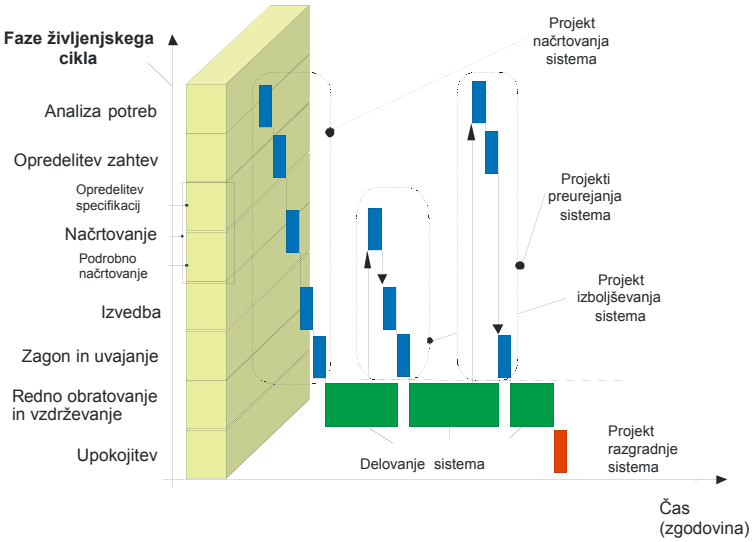
- Bistveni elementi
  - celostni pristop
  - interdisciplinarnost in multi-disciplinarnost
  - koncept življenjskega cikla
- Celostni pristop
  - sočasno načrtovanje
    - produkta
    - procesa njegove izdelave
    - podpornih funkcij, ki omogočajo prodajo in učinkovito uporabo proizvoda

Življenjski cikel sistemov

- Razvojna pot sistema
  - tudi umetni sistemi, ki jih ustvarja človek, imajo razvojno pot »od rojstva do smrti«
  - razvojna pot je proces, ki se prične in konča pri uporabniku
- Poenostavljena shema življenjskega cikla



Življenjski cikel sistemov



1.3 Modeliranje informacijskih sistemov

- Modeli sistemov
  - fizični
  - shematski
  - matematični
    - zvezni modeli
    - diskretno-dogodkovni modeli
    - hibridni modeli
- Modeli informacijskih sistemov
  - shematski
    - razredni diagrami, diagrami poteka, diagrami stanj
  - matematični
    - diskretno-dogodkovni modeli

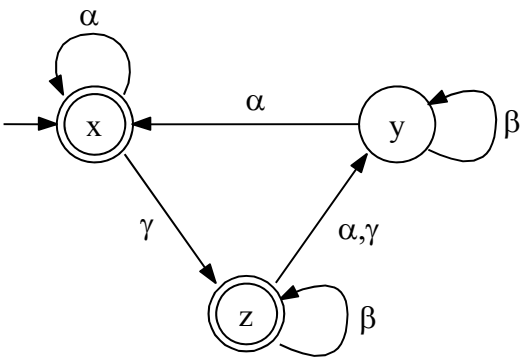
1.3.1 Avtomati

(angl. *automata*)

- Deterministični avtomat $G = (X, \Sigma, \delta, x_0, X_m)$
  - X je množica stanj avtomata
  - Σ je množica simbolov, ki so pridruženi prehodom avtomata
  - $\delta : X \times \Sigma \to X$ je funkcija prehajanja stanj avtomata
  - x_0 je začetno stanje avtomata
  - $X_m \subseteq X$ je množica obeleženih stanj
- Opombe:
  - možni so dogodki, ki ne spreminjajo stanja $\delta(x, \sigma) = x$
  - možni so različni dogodki, ki povzročijo enak prehod
  - δ je delna funkcija na domeni $X \times \Sigma$

Predstavitev avtomata z diagramom prehajanja stanj

LMSV LAMS



označuje začetno stanje (angl. *initial state*)

označuje obeleženo stanje (angl. *marked state*)

Formalni jeziki
(angl. *formal languages*)

LMSV LAMS

- Osnove
  - abeceda $\Sigma = \{\sigma_i ; \sigma_i \text{ predstavlja dogodek v sistemu}\}$
  - sled (niz, beseda) $s = \sigma_1\sigma_2 ...\sigma_n$
  - stik dveh sledi $s_1s_2 = \sigma_{11}\sigma_{12} ...\sigma_{1n}\sigma_{21}\sigma_{22} ...\sigma_{2m}$
  - prazna sled $\varepsilon s = s\varepsilon = s$
  - Kleenovo zaprtje Σ^* (angl. *Kleene-closure*)
    - množica vseh končnih sledi iz elementov Σ
- Primer:

$\Sigma = \{a, b, c\}$ $\quad s_1 = abbc,\ s_2 = bacbcc,$
$s_1s_2 = abbcbacbcc$

$\Sigma^* = \{\varepsilon, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, aab, \ldots\}$

Formalni jeziki /2

- Jezik
  - definiran kot podmnožica Σ^*
  - operacije nad jeziki: unija, presek, razlika, komplement (na Σ^*)
  - stik jezikov (angl. *concatenation*)

    $L_1, L_2 \subseteq \Sigma^*$:

    $L_1 L_2 = \{ s \in \Sigma^*; (s = s_1 s_2) \wedge (s_1 \in L_1) \wedge (s_2 \in L_2) \}$
  - predponsko zaprtje (angl. *prefix-closure*)

    $L \subseteq \Sigma^*$:

    $\overline{L} = \{ s \in \Sigma^*; \exists t \in \Sigma^*, st \in L \}$
  - v splošnem $L \subseteq \overline{L}$; jezik je *predponsko zaprt*, če $L = \overline{L}$

Formalni jeziki /3

- Primer:

  $\Sigma = \{a, b, c\}$

  $\Sigma^* = \{\varepsilon, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, aab, \ldots\}$

  $L_1 = \{\varepsilon, a, abb\}$, $L_2 = \{\text{sledi treh simbolov, ki se zacnejo z } a\}$

  $L_1 \cup L_2 = \{\varepsilon, a, aaa, aab, aac, aba, abb, abc, aca, acb, acc\}$

  $L_1 \cap L_2 = \{abb\}$

  $L_1 \setminus L_2 = \{\varepsilon, a\}$

  $L_1 L_2 = \{aaa, \ldots, acc, aaaa, \ldots, aacc, abbaaa, \ldots, abbacc\}$

  $\overline{L_1} = \{\varepsilon, a, ab, abb\}$ L_1 ni predponsko zaprt

Zveza med jeziki in avtomati

- Avtomat predstavlja dva jezika
  - Jezik, ki ga avtomat G generira - $L(G)$

    $$\delta(x,\varepsilon)=x,\quad \delta(x,s\sigma)=\delta(\delta(x,s),\sigma),\quad s\in\Sigma^*,\sigma\in\Sigma$$
    $$L(G)=\{s\in\Sigma^*;\delta(x_0,s)\text{ je def.}\}$$

    sestavljajo ga končna zaporedja simbolov (sledi, besede), ki nastajajo s proženjem prehodov v avtomatu
  - Jezik, ki ga avtomat G obeleži - $L_m(G)$

    $$L_m(G)=\{s\in L(G);\delta(x_0,s)\in X_m\}$$

    sestavljajo ga sledi iz $L(G)$, ki se končajo v obeleženem stanju

Blokiranje
(angl. *blocking*)

- Med jeziki, povezanimi z danim avtomatom G, veljajo v splošnem naslednje relacije:

  $$\varnothing\subseteq L_m(G)\subseteq\overline{L_m(G)}\subseteq L(G)$$

  - $L(G)$ je že po definiciji predponsko zaprt, medtem ko pri $L_m(G)$ to v splošnem ne drži, ker niso vsa stanja obeležena ($X_m\subset X$)
- Mrtva točka (angl. *deadlock*)
  - avtomat lahko preide v stanje x, kjer δ ni definiran za noben σ; če $x\notin X_m$, govorimo o blokiranju v obliki mrtve točke

Blokiranje /2

LMSV LAMS

- Mrtvi tek (angl. *livelock*)
  - avtomat lahko preide v eno od stanj, ki tvorijo zanko; možni so sicer nadaljnji prehodi, a le znotraj te množice stanj; če nobeno od teh stanj ni obeleženo, spet govorimo o blokiranju, tokrat o mrtvem teku
- Ali je katera od vrst blokiranja možna, je razvidno iz relacije med $\overline{L_m(G)}$ in $L(G)$:

$\overline{L_m(G)} \subset L(G)$ ... sistem lahko blokira (blokirajoč sistem)

$\overline{L_m(G)} = L(G)$ ... sistem ne more blokirati (neblokirajoč sistem)

Končni avtomati
(angl. *finite automata*)

LMSV LAMS

- Z avtomati predstavimo jezik na kompakten način (namesto naštevanja sledi jezika)
- Število stanj takšnega avtomata
  - v splošnem ni končno
  - če je končno ➔ končni avtomat
- Jezik, ki ga obeleži končni avtomat, je **regularen jezik** (angl. *regular language*)
  - število sledi takšnega jezika je lahko neskončno
  - operacije nad regularnimi jeziki dajo kot rezultat vedno regularen jezik

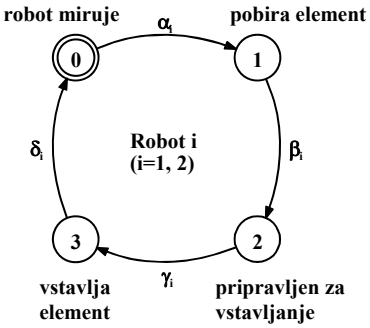
Primer – dva robota za vstavljanje elementov na tiskana vezja

- Dva robota “pick and place”

Oznake dogodkov

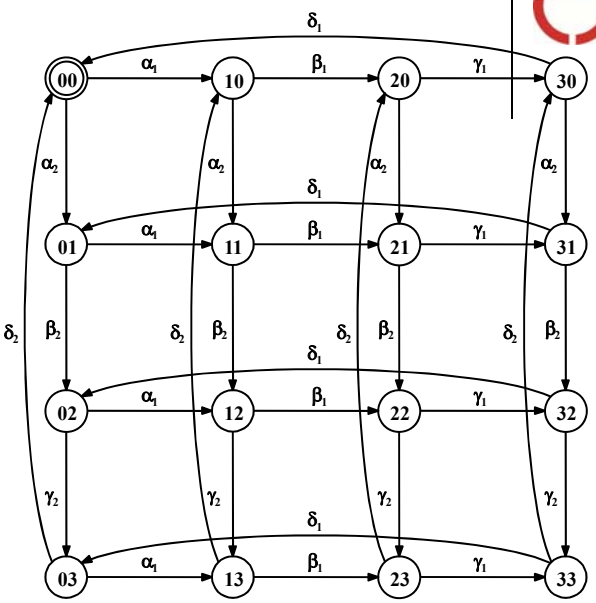
α_1, α_2	pričetek pobiranja
β_1, β_2	konec pobiranja
γ_1, γ_2	pričetek vstavljanja
δ_1, δ_2	konec vstavljanja

- Model posameznega robota
  - stanja in dogodki



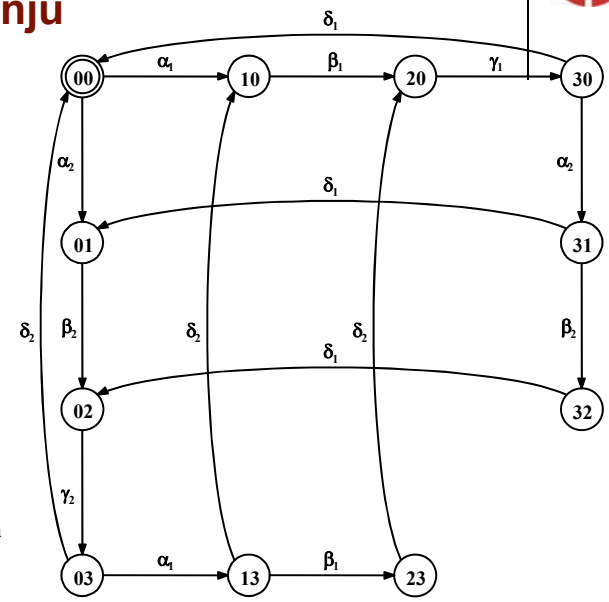
Model obeh robotov

- Število različnih dogodkov se ne poveča
  - se pa isti dogodek pojavi večkrat
- Število različnih stanj se eksponentno povečuje



Model obeh robotov ob upoštevanju omejitev

- Robota ne moreta hkrati pobirati ali vstavljati elementov



Oznake dogodkov

α_1, α_2	pričetek pobiranja
β_1, β_2	konec pobiranja
γ_1, γ_2	pričetek vstavljanja
δ_1, δ_2	konec vstavljanja

Značilnosti modeliranja z avtomati

LMSV
LAMS

- Enostavno modeliranje
- Dober pregled nad stanji, dokler so modeli enostavni
- Problem pri paralelnih aktivnostih
  - eksplozija stanj
- Potreben boljši način za opis vzporednih aktivnosti, sinhronizacije ipd.

1.3.2 Petrijeve mreže

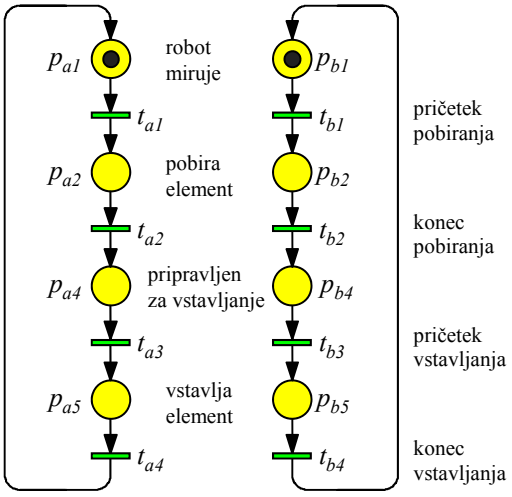
LMSV LAMS

- Carl Adam Petri, nemški matematik
  - "Kommunikation mit Automaten", disertacija iz leta 1961/62
- Grafični in matematični opis sistema diskretnih dogodkov
  - enostavno modeliranje vzporednih dogajanj in sinhronizacije aktivnosti
  - diskretni prostor stanj je predstavljen na bolj
  - kompakten način
  - možnost vpeljave časa ➔ časovni modeli
- Uporaba: komunikacije – protokoli, industrijski procesi, ekspertni sistemi, ...

Primer – dva robota za vstavljanje elementov na tiskana vezja

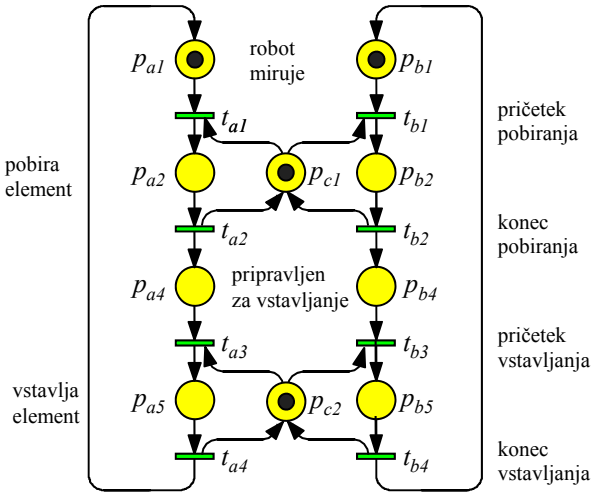
LMSV LAMS

- Model v obliki Petrijeve mreže
  - označitev stanja je porazdeljena po modelu
  - neposredno modeliranje vzporednih aktivnosti
  - število elementov mreže ne narašča eksponencialno



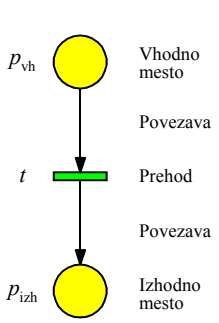
Primer – dva robota za vstavljanje elementov na tiskana vezja /2

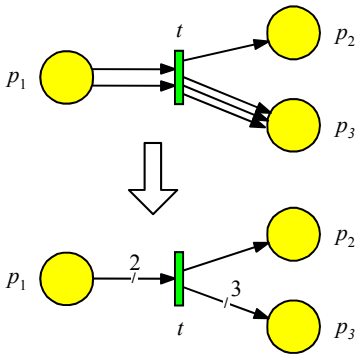
- Model z omejitvami
  - z dodatnimi elementi mreže omejimo prehajanje stanj
  - enostavno modeliramo sinhronizacijo aktivnosti



Graf Petrijeve mreže

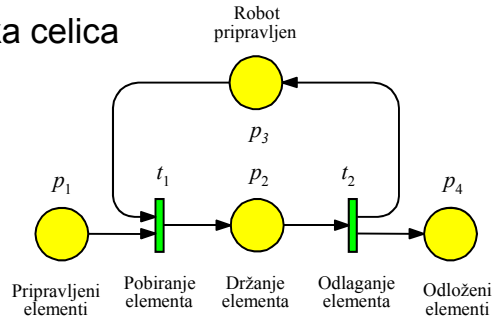
- Dve vrsti vozlišč (bipartitni graf)
  - mesta, prehodi
  - povezave mesto→prehod in prehod→mesto
  - uteži na povezavah





Primer grafa

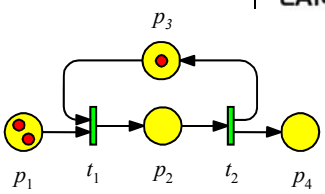
- Robotska celica



  - robot lahko pobere element le, če je element na predvidenem mestu, in če je robot na razpolago
  - element lahko odloži le v primeru, da ga je predhodno držal, nato je robot zopet na razpolago za nov cikel

Označitev Petrijeve mreže

- Pomen vozlišč
  - prehodi – dogodki
  - mesta – pogoji za dogodke
- V mesta Petrijeve mreže postavimo žetone – označitev mreže



  - žetoni določajo, ali so izpolnjeni pogoji za proženje prehodov = ali se lahko zgodi dogodek
  - prehod se lahko sproži, če je v vsakem vhodnem mestu vsaj en žeton (če ni uteženih povezav)
  - razporeditev žetonov po mestih določa stanje sistema
  - v grafični predstavitvi narišemo žetone kot črne pike ali številko v mestu

Razredi Petrijevih mrež

LMSV LAMS

- Elementarne mreže (Elementary net systems)
  - osnovni model
  - samo pogoji in dogodki
- Mreže mest in prehodov (Place/transition nets)
  - centralni model
  - enostaven model, razvitih veliko tehnik analize
- Visokonivojske mreže (High-level Petri nets)
  - napredni model
  - kompaktna predstavitev kompleksnih sistemov
- Časovne mreže (Timed and stochastic nets)
  - časovni model
  - simulacija in optimizacija

Matematični opis označene Petrijeve mreže mest in prehodov

LMSV LAMS

- Označena Petrijeva mreža = Petrijev mrežni sistem
- Matematični zapis $PN = (P, T, A, W, M_0)$

$P = \{p_1, p_2, ..., p_m\}$	končna, neprazna množica m mest
$T = \{t_1, t_2, ..., t_n\}$	končna, neprazna množica n prehodov med mesti in je množici P tuja množica: $P \cup T \neq \emptyset, \quad P \cap T = \emptyset$
$A \subseteq (P \times T) \cup (T \times P)$	množica usmerjenih povezav med vhodnimi mesti in prehodi ter med prehodi in izhodnimi mesti
$W : A \to N$	utežna funkcija - definira število vzporednih povezav med dvema vozliščema
$M : P \to N$	označitev mreže, M_0 je začetna označitev; N je množica nenegativnih celih števil

Matematični opis označene Petrijeve mreže mest in prehodov

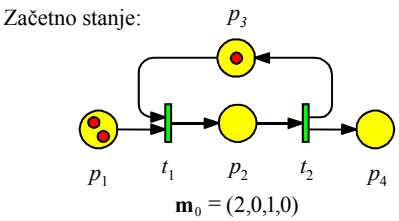
- Druga možnost opisa povezav
- Matematični zapis $PN = (P, T, Pre, Post, M_0)$

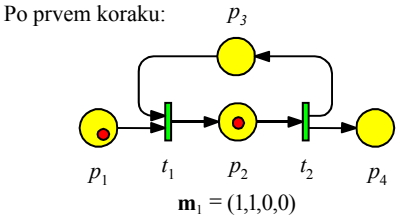
$P = \{p_1, p_2, ..., p_m\}$	• končna, neprazna množica m mest
$T = \{t_1, t_2, ..., t_n\}$	• končna, neprazna množica n prehodov med mesti in je množici P tuja množica
$Pre : (P \times T) \to N$	• funkcija vhodnih povezav; če obstaja s k utežena povezava med p in t, potem je $Pre(p,t) = k$, sicer je $Pre(p,t) = 0$
$Post : (P \times T) \to N$	• funkcija vhodnih povezav; če obstaja s k utežena povezava med t in p, potem je $Post(p,t) = k$, sicer je $Post(p,t) = 0$
$M : P \to N$	• označitev mreže, M_0 je začetna označitev; N je množica nenegativnih celih števil

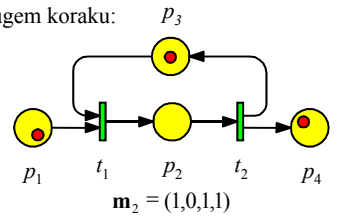
Dinamika Petrijeve mreže

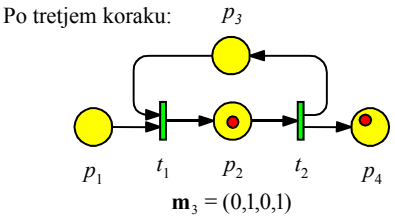
- Dinamika
  - prehajanje žetonov med mesti ob proženju prehodov (token game)
- Pravilo proženja prehodov v Petrijevi mreži (firing rule, no-concurrency setting)
  - prehod je omogočen, če vsako od vhodnih mest tega prehoda vsebuje vsaj toliko žetonov, kot je utež pripadajoče povezave,
  - omogočen prehod se lahko sproži, ali pa tudi ne, kar je lahko stvar dodatne interpretacije mreže,
  - sprožitev prehoda je hipna (brez zakasnitve) in odstrani toliko žetonov, kot je utež pripadajoče povezave, iz vsakega vhodnega mesta ter doda toliko žetonov, kot je utež pripadajoče povezave, v vsako izhodno mesto.

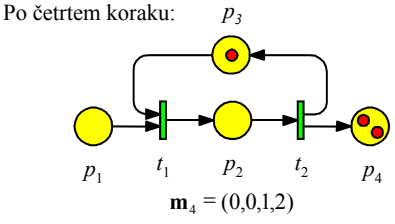
Primer











Matrični zapis

- Matrika vhodnih povezav **Pre**
  - element v i-ti vrstici in j-tem stolpcu opiše povezavo med p_i in t_j

$$c_{ij}^{-} = \begin{cases} W(p_i,t_j), & (p_i,t_j) \in A \\ 0, & (p_i,t_j) \notin A \end{cases}$$

- Matrika izhodnih povezav **Post**
  - element v i-ti vrstici in j-tem stolpcu opiše povezavo med t_j in p_i

$$c_{ij}^{+} = \begin{cases} W(t_j,p_i), & (t_j,p_i) \in A \\ 0, & (t_j,p_i) \notin A \end{cases}$$

- Matrika povezav (incidenčna matrika): $\mathbf{C} = \mathbf{Post} - \mathbf{Pre}$

$$(p_i,t_j) \in A \wedge (t_j,p_i) \in A \Rightarrow c_{ij} = W(t_j,p_i) - W(p_i,t_j)$$

Matrični zapis /2

- Vektor proženj prehodov $\mathbf{u} = [0,\ldots,0,1,0,\ldots,0]^T$
  - enica na j-tem mestu predstavlja proženje prehoda t_j
- Prehod je omogočen, če $\mathbf{m} \geq \mathbf{Pre} \cdot \mathbf{u}$
- Enačba prehajanja stanj

$$\mathbf{m}' = \mathbf{m} + \mathbf{C} \cdot \mathbf{u}$$

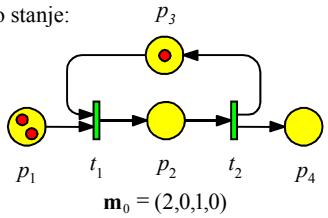
- Potreben pogoj za dosegljivost označitve
  - označitev **m** je dosegljiva iz označitve $\mathbf{m}_0$, samo če ima enačba

$$\mathbf{m}_0 + \mathbf{C} \cdot \mathbf{x} = \mathbf{m}$$

celoštevilsko rešitev za **x**

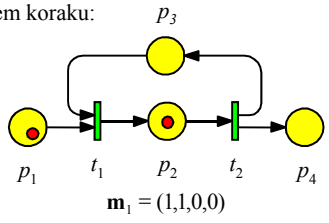
Primer

Začetno stanje:



$$\mathbf{C} = \begin{bmatrix} -1 & 0 \\ 1 & -1 \\ -1 & 1 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Po prvem koraku:



$$\mathbf{m}_1 = \mathbf{m}_0 + \mathbf{C} \cdot \mathbf{u} =$$

$$= \begin{bmatrix} 2 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ 1 & -1 \\ -1 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Analiza Petrijevih mrež

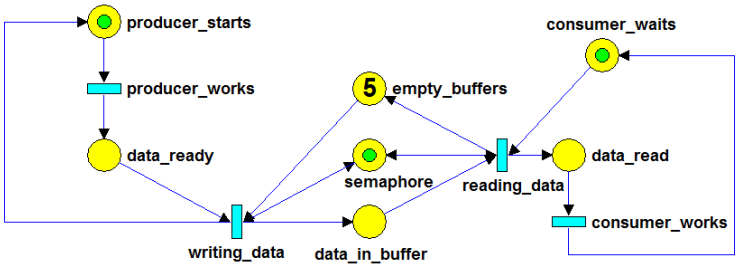
LMSV LAMS

- Kaj omogoča analiza lastnosti mreže:
  - Kolikšno je največje možno število žetonov v mestih?
  - Ali mreža lahko zaide v mrtvi tek ali mrtvo točko (live-lock, dead-lock)?
- Vedenjske lastnosti
  - odvisne od strukture mreže in začetne označitve
  - omejenost, dosegljivost, živost, …
- Strukturne lastnosti
  - odvisne le od strukture
  - veljajo pri poljubni začetni označitvi mreže
  - strukturna omejenost, strukturna živost, …
- Časovne mreže – simulacija

Primer

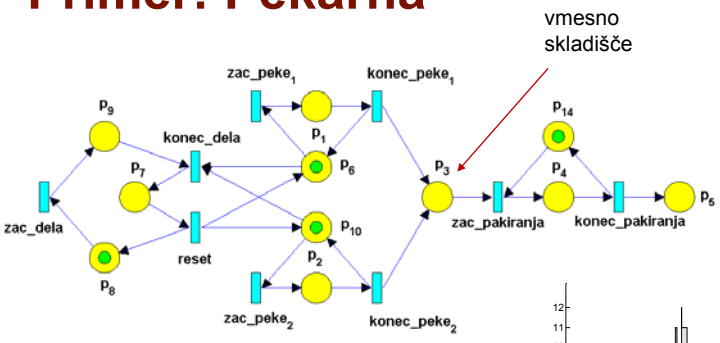
LMSV LAMS

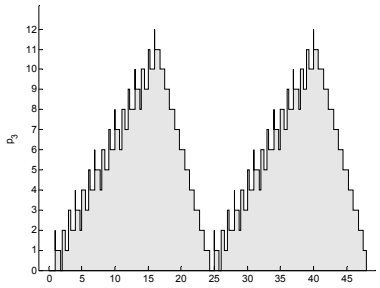
- Proces „producer-consumer“



Če mesto ‚empty-buffers‘ odstranimo, lahko število žetonov v mestu ‚data_in_buffer‘ naraste preko vsake meje;
če ‚empty-buffers‘ ne vsebuje začetnih žetonov, ima mreža mrtvo točko.

Primer: Pekarna





Peka šarže traja 1 uro in poteka v dveh izmenah, pakiranje šarže traja ¾ ure in poteka v treh izmenah.

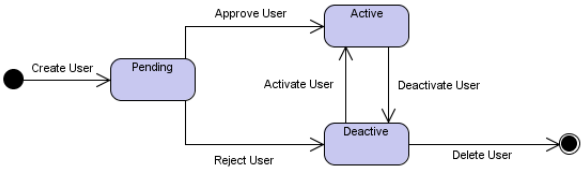
Prikazano je število šarž (vsaka ima 100 kosov) v vmesnem skladišču po urah.

1.3.3 Diagrami stanj

- Problem specifikacije in načrtovanja velikih in kompleksnih reaktivnih sistemov
- Reaktivni sistem je nasprotje transformacijskega sistema
  - v veliki meri dogodkovno gnan
  - stalno se mora odzivati na zunanje in notranje vzbujanje
- Primeri reaktivnih sistemov
  - avtomobili, telefoni, komunikacijska omrežja,
  - letalski sistemi, sistemi vodenja avtonomnih vozil,
  - računalniški operacijski sistemi,
  - uporabniški vmesniki običajne programske opreme

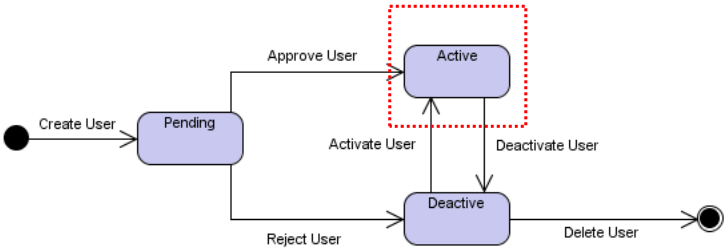
Diagrami stanj – Statecharts

- D. Harel and M. Politi, Modeling Reactive Systems with Statecharts: The STATEMATE Approach, McGraw-Hill, 1998
- Povezava stanj in dogodkov
- Osnova:
  - stanje
  - prehod
  - začetno stanje
  - končno stanje



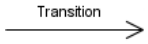
Gradniki diagramov stanj

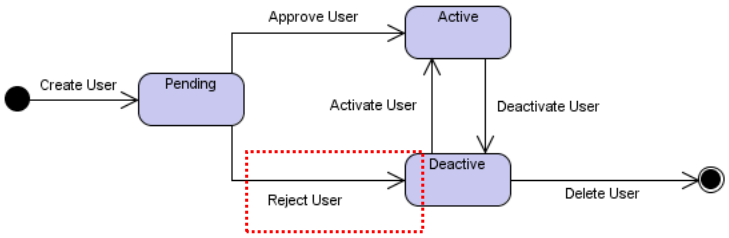
- Stanje: način (mode) entitete
- Pravokotnik z zaobljenimi vogali in vpisanim imenom stanja 



Gradniki diagramov stanj

LMSV LAMS

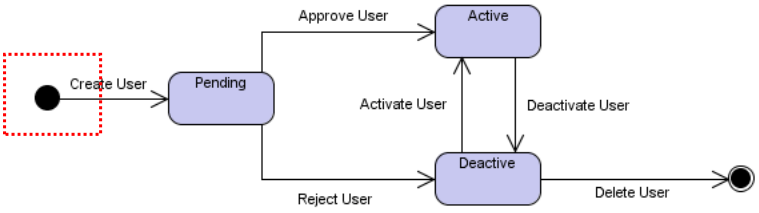
- Prehod: sprememba stanja
- Puščica označena z imenom dogodka 



Gradniki diagramov stanj

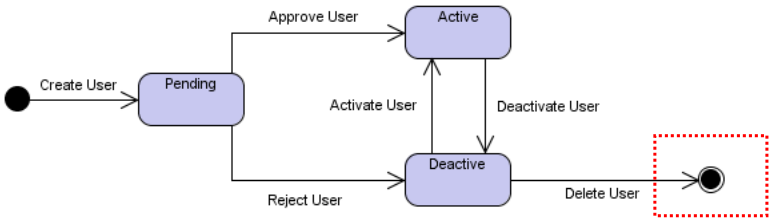
LMSV LAMS

- Začetno stanje: stanje objekta pred kakršnimkoli prehodom
- Označeno s polnim krogom ●
- Diagram lahko vsebuje le eno začetno stanje

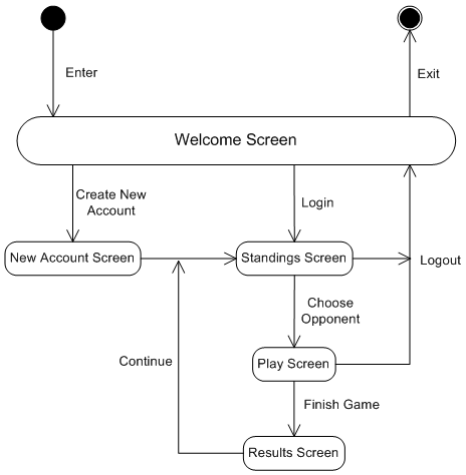


Gradniki diagramov stanj

- Končno stanje: brisanje objekta
- Poln krog obdan s še enim krogom

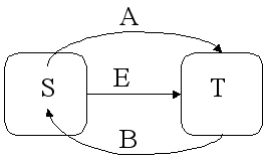


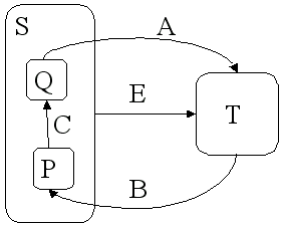
Primer: Spletna igra



Hierarhija diagramov stanj

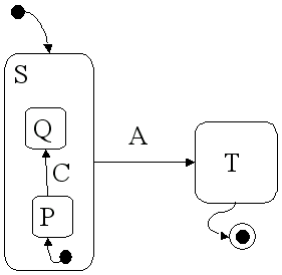
- Diagrami stanj so hierarhični
  - avtomate sestavlja en sam nivo (are flat)
- Puščice so lahko povezane z notranjostjo vozlišča S
- Pod-stanja v stanju S

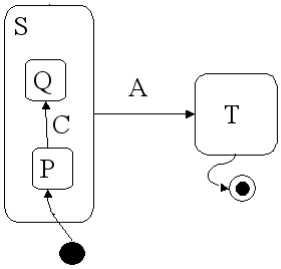




Privzeto stanje v diagramih stanj

- Če ni očitno, mora biti v gnezdenem diagramu označeno začetno in končno stanje

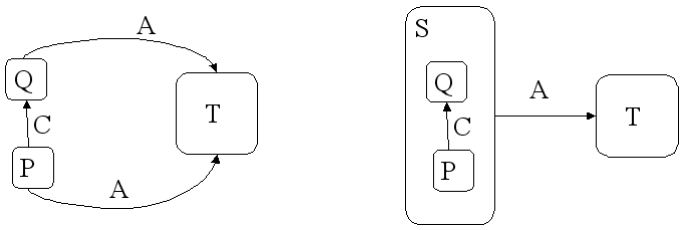




Rojenje stanj

LMSV LAMS

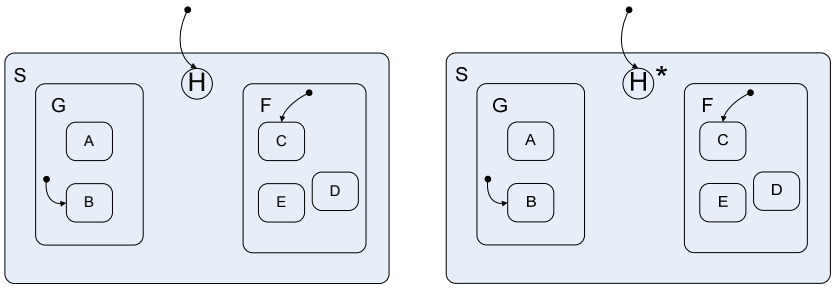
- Roj stanj: oblika hierarhije
- Prednost: bolj pregleden diagram



Mehanizem zgodovine

LMSV LAMS

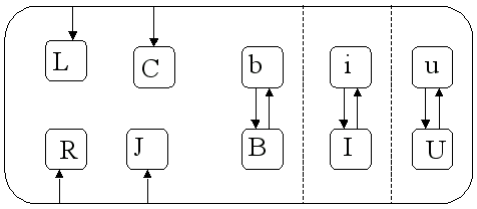
- Nadaljevanje iz zadnjega aktivnega stanja znotraj gnezdenega diagrama
- Inicializacija zgodovinske spremenljivke



*označuje propagacijo zgodovine po vseh podstanjih

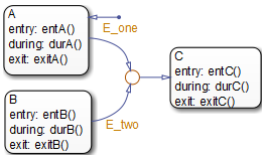
Hkratna stanja

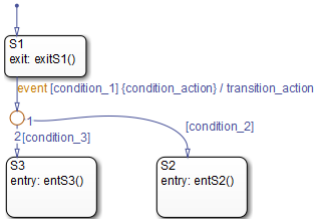
- Sočasno je lahko aktivnih več stanj
- Primer: načini pisave krepko, ležeče in podčrtano



Spojniki (konektorji)

- Možnost vejitve in združitve povezav
- Odločitveni spojnik (C-connector, S-connector)
  - predstavitev različnih možnosti prehajanja istega prehoda
  - povezava se razcepi, pri čemer so na vejah označeni različni pogoji (C) ali dogodki (S)
- Združevalni spojnik





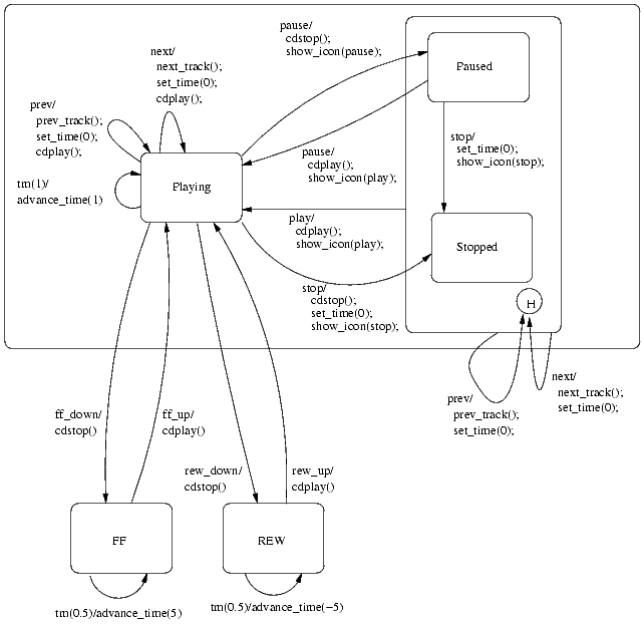
Zakasnitve

- Zakasnitev:
  - oznaka **„casZakasnitve <“** znotraj stanja, npr. „10 sec <“
  - stanje je zamrznjeno (se ne spreminja), dokler zakasnitev ne poteče
- Prekinitev
  - oznaka **„< časPrekinitve“** znotraj stanja, npr. „< 5 min“
  - po poteku določenega časa nastopu poseben dogodek „timeout“, temu mora ustrezati puščica ven iz stanja

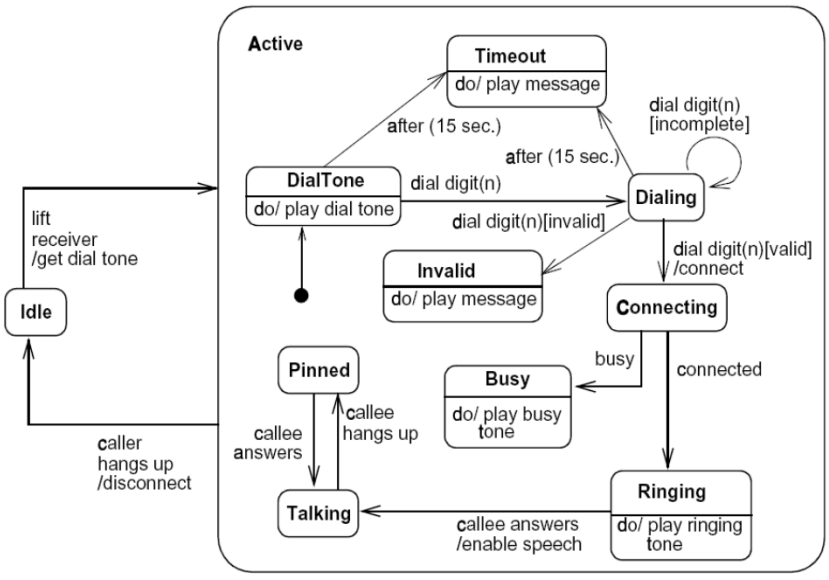
Oznake stanj in prehodov

- Lahko vsebujejo akcije in prehodne pogoje
- Oznake stanj
  - akcije:
    do/akcija, during/akcija
  - akcije ob vstopu/izstopu iz stanja
  - funkcija in(state)
- Oznake prehodov
  - dogodkovni izrazi (and, or, not …)
  - prehodni pogoj:
    dogodek [pogoj]
  - akcije od prehodu:
    dogodek/akcija

Primer: CD-predvajalnik



Primer: telefon

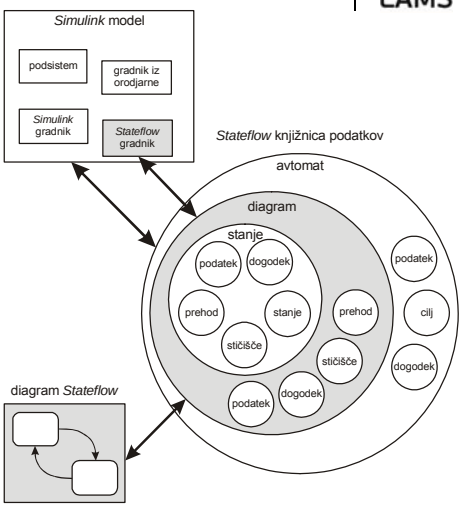


Stateflow – orodje za modeliranje in simulacijo z diagrami stanj

- Diskretna stanja in prehodi med njimi
- Dogodkovno prožena simulacija
- Gradniki modela:
  - grafični: stanja, prehodi, zgodovinski stiki, privzeti prehodi, stikališča
  - negrafični: dogodki – omogočajo simulacijski tek, podatki – olajšajo modeliranje in omogočajo povezavo z modeli v Simulink-u
  - akcije in akcijski jezik

Zgradba modela v orodju Stateflow

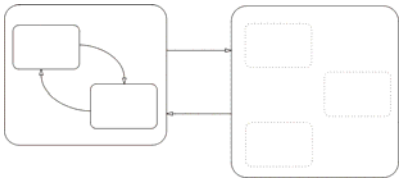
- Model vstavimo kot poseben blok v simulacijsko shemo v Simulink
- Dogodki
  - vhodni (določajo trenutke »izvajanja« diskretnega modela), izhodni, notranji
- Podatki
  - vhodni podatki omogočajo prenos vrednosti iz Simulinka v Stateflow; izhodni v obratni smeri; tudi notranji podatki



Osnovni grafični gradniki orodja Stateflow

LMSV LAMS

- Stanja
  - možna hierarhična zgradba – vzporedna in izključujoča se stanja
  - oznaka stanja in akcije:



ime_stanja/ entry:*akcija_ob_vhodu*; exit:*akcija_ob izhodu*; during:*akcija_med_bivanjem*; on *dogodek*:*akcija_ob_dogodku*

- Prehodi
  - povezave med stanji
  - oznaka prehoda:

    dogodek [pogoj]{akcija_ob_pogoju}/akcija_ob_prehodu
- Privzeti prehod
  - določi prvo od izključujočih se stanj, ki se aktivira

Povezava orodij Stateflow in Simulink

LMSV LAMS

- Različni načini izvajanja
  - proženi način

    Stateflow se izvaja ob nastopu dogodkov – izbrani signal v simulacijski shemi prečka ničlo v izbrani smeri
  - dedovani način

    Stateflow se izvaja v vsakem glavnem integracijskem koraku
  - vzorčeni način

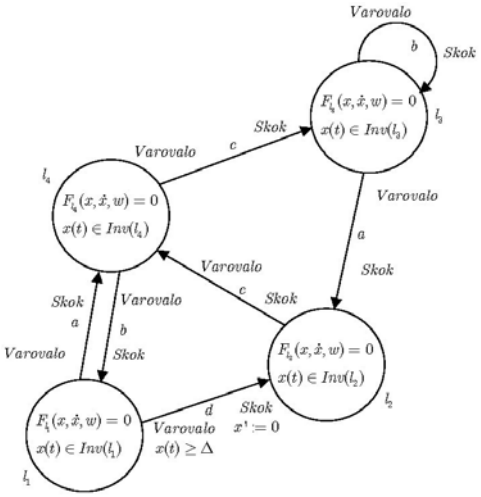
    Stateflow se izvaja z definirano frekvenco vzorčenja
  - zvezni način

    Stateflow se izvaja v vsaki iteraciji (glavnem in pomožnih integracijskih korakih)
- Možnost prenosa podatkov – vrednosti signalov v obeh smereh

Simulacija hibridnega avtomata

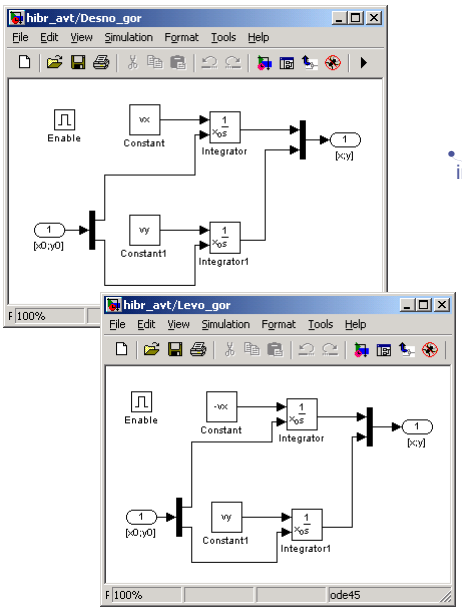
LMSV LAMS

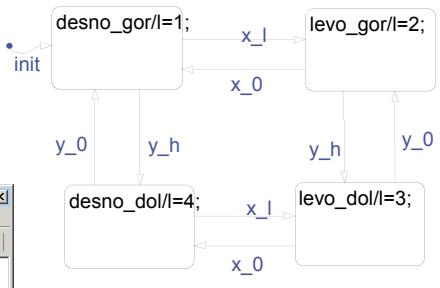
- Prehajanje diskretnih stanj realiziramo v Stateflow-u
- Enačbe zvezne dinamike realiziramo kot omogočene podsisteme v Simulinku
- Stateflow prožijo dogodki, ki nastanejo pri prehodu zveznih signalov preko določene vrednosti
- Izhod Stateflow-a omogoča podsisteme

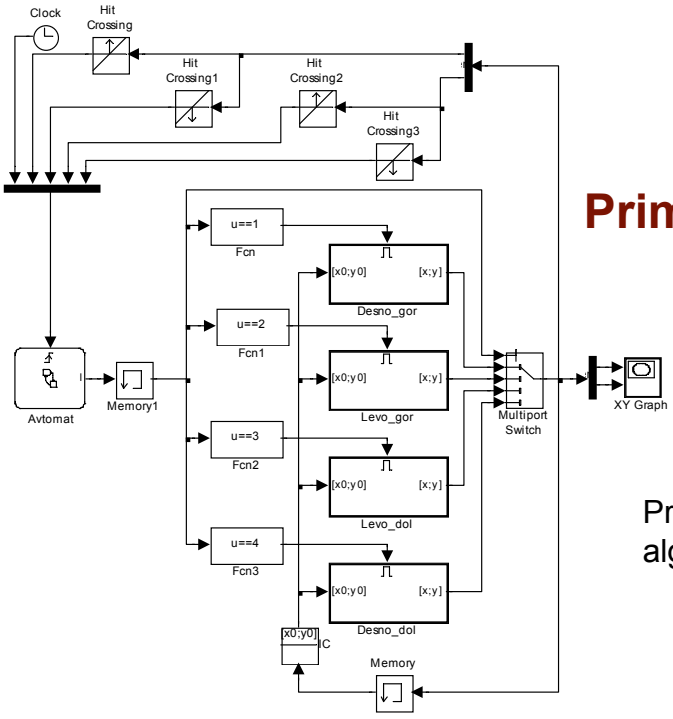


Primer – krogla na biljardni mizi

LMSV LAMS







LMSV LAMS

Primer – krogla na biljardni mizi /2

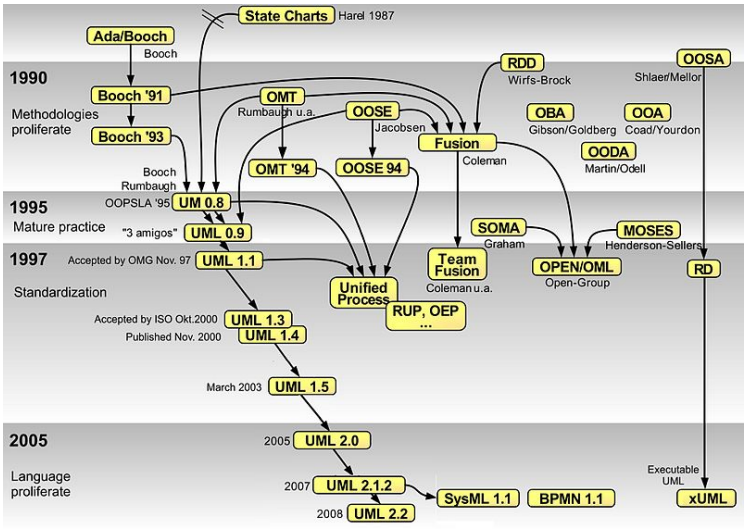
Problem: pojavi se algebrajska zanka

1.3.4 Poenoteni jezik modeliranja (UML)

LMSV LAMS

- **Unified Modeling Language** (**UML**)
  - standardiziran (**ISO/IEC 19501:2005**), splošnonamenski modelirni jezik na področju programskega inženirstva
  - vključuje vrsto tehnik grafične predstavitve za ustvarjanje vizualnih modelov objektno usmerjenih programskih sistemov
- Kombinacija
  - tehnik podatkovnega modeliranja (entity relationship diagrams)
  - modeliranja poslovnih procesov (work flows)
  - modeliranja objektov
  - modeliranja komponent

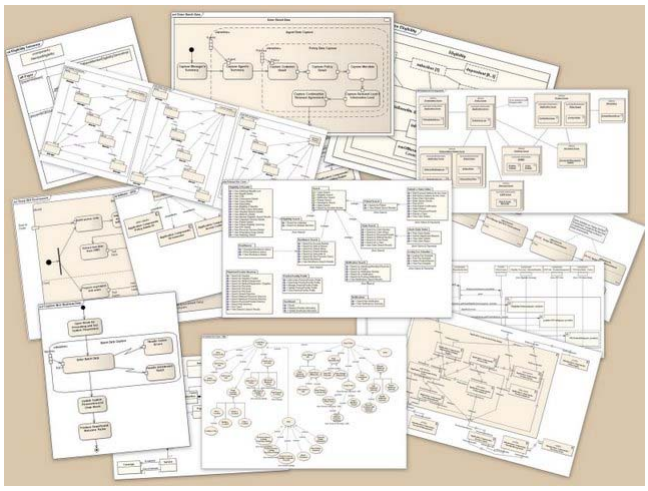
Zgodovina



Jezik UML

- UML omogoča standardni način predstavitve načrtov sistema
- Vključuje različne elemente, kot so:
  - aktivnosti (activities)
  - akterji (actors)
  - poslovne procese (business processes)
  - podatkovne sheme (database schemas)
  - (logične) komponente
  - stavke programskih jezikov
  - ponovno uporabljive programske komponente

Diagrami UML



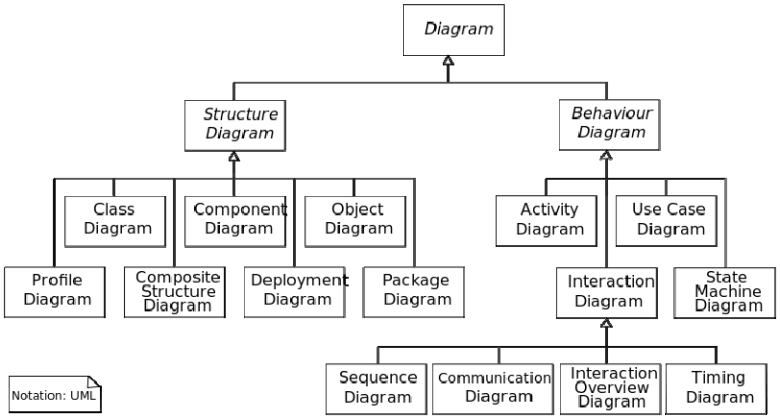
Diagrami UML

- V verziji UML 2.2 (2009) je definiranih 14 tipov diagramov, ki se delijo v dve kategoriji
  - sedem tipov diagramov predstavlja *strukturne* informacije
    - statična struktura sistema
  - preostalih sedem tipov predstavlja splošne tipe *obnašanja*
    - dinamična struktura sistema
    - pri tem so vključeni štirje tipi, ki predstavljajo različne tipe sodelovanja (*interakcij)*

Diagrami UML

LMSV LAMS

- Diagrame lahko kategoriziramo hierarhično



Diagrami strukture

LMSV LAMS

- Diagrami strukture se osredotočajo na stvari, ki jih mora vsebovati modelirani sistem
  - široka raba pri dokumentiranju programske arhitekture
- Razredni diagram (Class diagram)
  - prikazuje množico razredov, njihove lastnosti ter vmesnike, sodelovanja in razmerja med razredi
  - najpogosteje uporabljeni diagram pri opisovanju objektno usmerjenih sistemov
- Diagram komponent
  - prikaže, kako je programski sistem razdeljen na komponente in prikazuje odvisnosti med komponentami sistema

Diagrami strukture /2

- Diagram kompozitne strukture (Composite structure diagram, dodan v ver. 2.0)
  - opisuje notranjo strukturo razreda in sodelovanja, ki jih ta struktura omogoča
- Diagram porazdelitve (Deployment diagram)
  - opisuje pri implementaciji in delovanju sistema uporabljeno strojno opremo ter porazdelitev komponent po delih te opreme
- Objektni diagram
  - prikazuje celotni ali delni pogled na strukturo vzorčnega primera modeliranega sistema v določenem trenutku
  - trenutni statični posnetek objektov in njihovih povezav

Diagrami strukture /3

- Paketni diagram (uveden z ver. 2.0)
  - opisuje, kako je sistem razdeljen na logične skupine (skupine razredov oz. packages) in kakšne so odvisnosti med temi skupinami
- Profilni diagram (uveden z ver. 2.0)
  - ukvarja se z razširitvami, in sicer s t.i. stereotipi in profili; prve prikaže kot razrede in druge kot pakete

Diagrami obnašanja

- Diagrami obnašanja se osredotočajo na to, kaj naj bi se v modeliranem sistemu dogajalo
  - široka raba pri opisovanju funkcionalnosti sistemov programske opreme
- Diagram aktivnosti
  - opisuje zaporedje korakov znotraj komponent, ki sodelujejo v poslovnem ali operativnem procesu
  - prikazuje množico aktivnosti, prehode med aktivnostmi, stičišča, tok objektov, uskladitve, kjer se vzporedni tokovi izvajanja sinhronizirajo in se po izpolnitvi pogojev izvajanje nadaljuje z naslednjo aktivnostjo; ter razvejitve, kjer se izvajanje nadaljuje v določeni smeri

Diagrami obnašanja /2

- Diagram stanj
  - opisuje stanja sistema in prehode med njimi ter pripadajoče dogodke in aktivnosti
  - diagram je pomemben predvsem pri modeliranju obnašanja vmesnikov
- Diagram primerov uporabe
  - opisuje funkcionalnost sistema v smislu akterjev, njihovih vlog, ki so predstavljene kot primeri uporabe, in soodvisnosti med primeri uporabe
  - določa funkcije, ki jih bo sistem podpiral, ne obravnava pa podrobnosti izvedbe teh funkcij

Diagrami interakcij

- Diagrami interakcij so podmnožica diagramov obnašanja
  - osredotočajo se na prehajanje kontrolnega toka in podatkov med elementi (objekti) modeliranega sistema
- Komunikacijski diagram
  - prikazuje interakcijo med objekti ali deli sistema v obliki zaporedja sporočil
  - združuje kombinacijo informacij iz razrednih diagramov, diagramov zaporedja in diagramov primerov uporabe
  - opisuje tako statično strukturo kot tudi dinamično obnašanje sistema

Diagrami interakcij /2

- Pregledni diagram interakcij
  - zagotavlja pregled interakcij, v katerem vozlišča predstavljajo komunikacijske diagrame
- Diagram zaporedja
  - prikazuje, kako objekti komunicirajo med seboj s časovnim zaporedjem sporočil
  - vključuje akterje, objekte, povezave in sporočila
  - ilustrira scenarije iz diagramov primerov uporabe
  - prikaže tudi trajanja objektov glede na izmenjana sporočila
- Časovni diagram
  - specifična oblika diagrama interakcij, kjer je poudarek na časovnih omejitvah

Razredni diagram

- Množica razredov, vmesnikov, sodelovanj in razmerij med razredi
- Razmerja med razredi
  - odvisnost
  - posplošitev
  - asociacija
  - agregacija

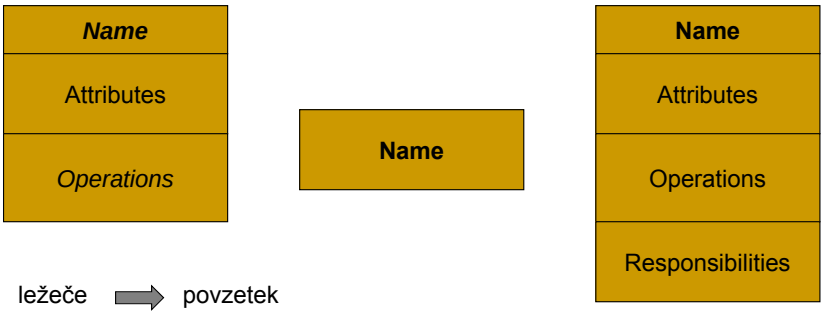
- Temelj za izdelavo diagrama komponent in diagrama porazdelitve
- Uporaben tudi za generiranje programske kode neposredno iz modela UML

Razred

- Razred je opis množice objektov, ki si delijo iste lastnosti (atribute), operacije (metode), razmerja in pomen
- Atribut je poimenovana lastnost razreda
  - opisuje razpon vrednosti, ki jih imajo lahko posamezne instance te lastnosti
- Operacija oz. metoda je storitev, ki jo lahko zahtevamo od objekta z namenom vpliva na obnašanje

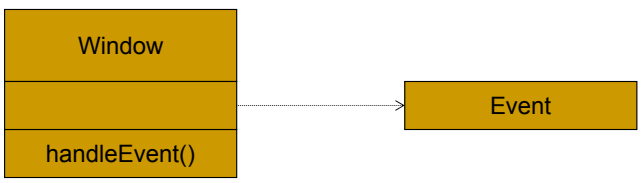
Predstavitev razreda

- Pravokotnik, ki vsebuje ime, atribute in operacije razreda



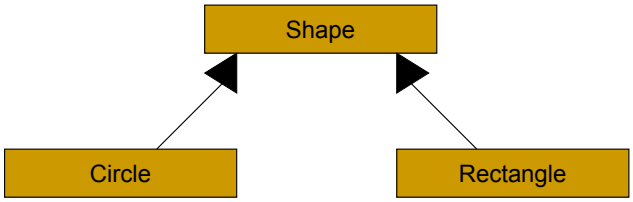
Odvisnost

- Odvisnost je razmerje tipa “uporablja”
  - v okviru tega lahko sprememba specifikacije enega razreda vpliva na drug razred, ki to specifikacijo uporablja
  - grafično je odvisnost označena s črtkano črto, ki se lahko zaključi s puščico in ima lahko ime
- Primer: razred uporablja drug razred v operaciji



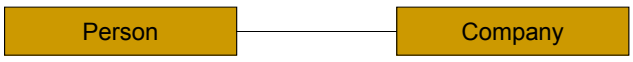
Posplošitev oz. generalizacija

- Posplošitev je razmerje tipa “vrsta nečesa” ali “je” med splošnim objektom (nad-razred, parent) in bolj specializiranim objektom (pod-razred, child)
  - objekti danega razreda so zamenljivi s primerki bolj specifičnih razredov.
  - pod-razred podeduje vse lastnosti nad-razreda.



Asociacija

- Asociacija je strukturno razmerje, ki opisuje množico zvez med objekti
  - asociaciji med objekti pravimo povezava
  - grafično je asociacija predstavljena s polno črto, ki se lahko zaključi s puščico
  - lahko so pripete še dodatne oznake

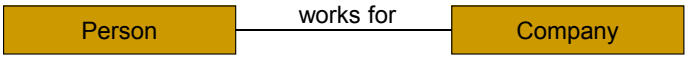


Dodatne označitve asociativne povezave

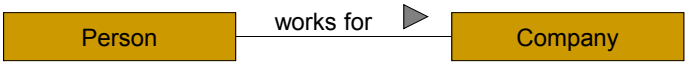
- Asociacija je lahko označena z vlogami in dodatnimi simboli
  - ime (name)
  - vloga (role)
  - številnost (multiplicity)
  - agregacija (aggregation)
  - kompozicija (composition)

Ime asociacije

- Opisuje naravo razmerja:

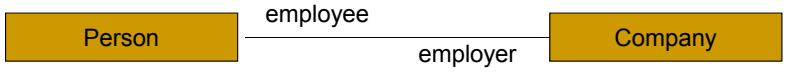


- in lahko kaže tudi smer, v kateri je potrebno razumeti ime asociacije:



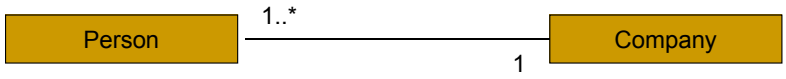
Asociativne vloge

- Opisujejo, kako se razredi predstavljajo drug drugemu v okviru asociacije
- V različnih asociacijah lahko razredi igrajo iste ali različne vloge



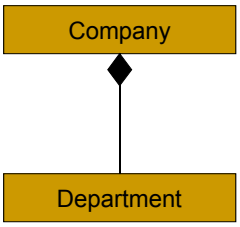
Številnost asociacije

- Možne vrednosti: eksplicitna vrednost, razpon vrednosti ali *, ki pomeni “mnogo”
- Primer: oseba (Person) je zaposlena pri enem podjetju (Company); podjetje zaposluje eno ali več oseb



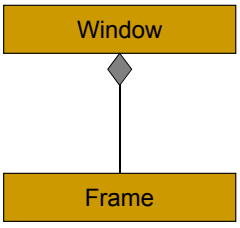
Agregacija

- Agregacija je razmerje tipa “celota/del” ali “vsebuje”
  - en razred predstavlja večjo stvar, ki je sestavljena iz manjših stvari
  - poseben primer asociacije
- Primer: Podjetje (Company) vsebuje oddelke (Department)



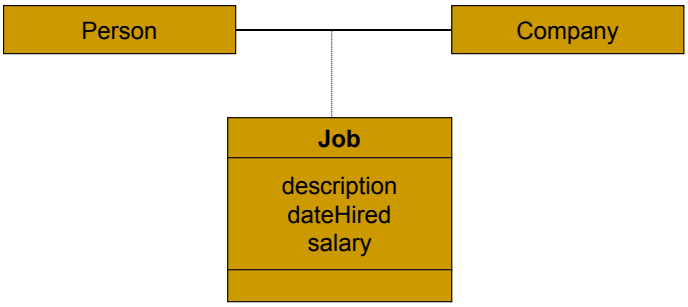
Kompozicija

- Kompozicija je posebna oblika agregacije, pri kateri delov ni možno ločiti od celote
- Primer: okno in okvir



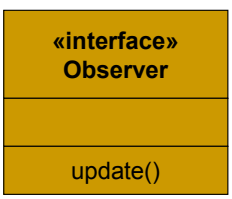
Asociativni razredi

- Asociativni razred ima tako lastnosti asociacije kot razreda



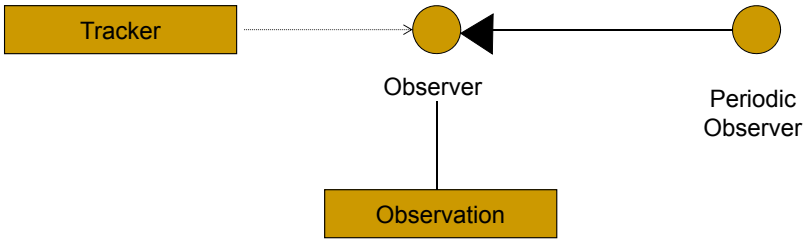
Vmesniki

- Vmesnik je poimenovana zbirka operacij
  - uporabimo ga za specifikacijo storitve razreda, ne da bi diktirali njeno implementacijo
- Primer:



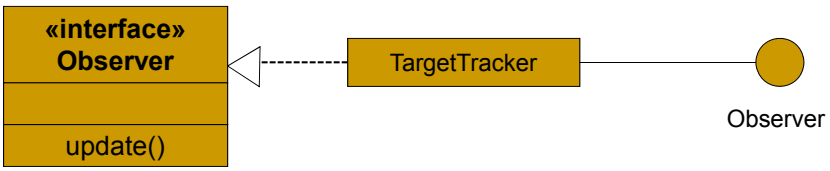
Vmesniki in razmerja

- Vmesnik lahko sodeluje v generalizaciji, asociaciji in odvisnih razmerjih
- Primeri:

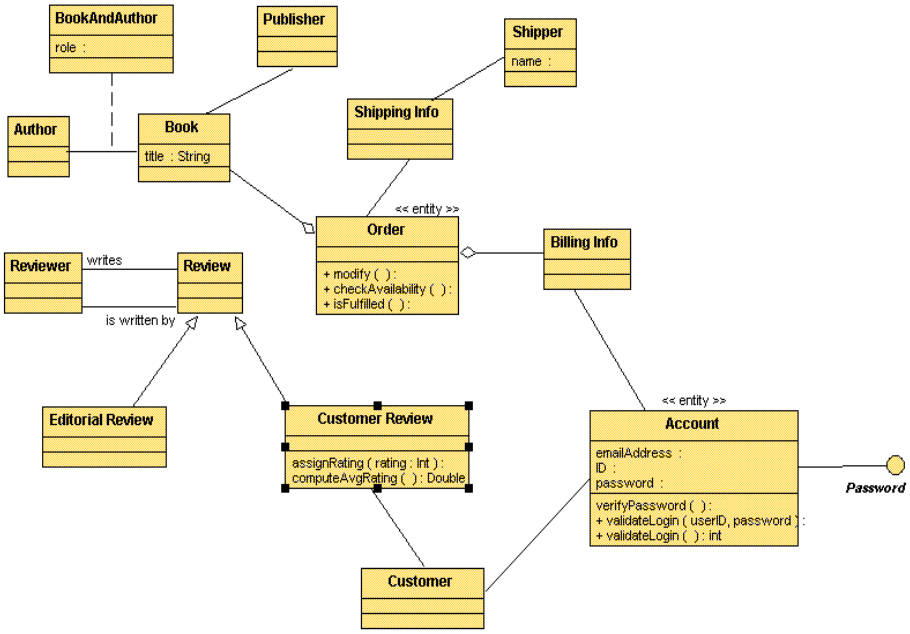


Realizacija

- Realizacija je razmerje med moduli, kjer ima modul pogodbo z drugim modulom, za katero garantira izvedbo
  - med vmesniki in pripadajočimi razredi ali komponentami, ki jih realizirajo

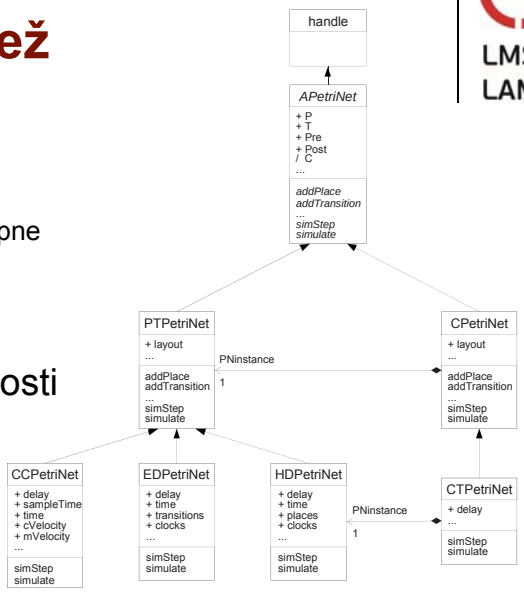


- Razred lahko realizira več vmesnikov



Primer: Razredi Petrijevih mrež

- Razredi, atributi, metode
  - \+ označuje javno dostopne lastnosti
  - / označuje izpeljane lastnosti
- Prikazane odvisnosti
  - generalizacija
  - asociacija
    - številnost
    - agregacija



Uporaba UML

LMSV LAMS

- UML kot celota je dokaj nov
  - šele prodira v praktično uporabo
  - vse več orodij za podporo modeliranju z UML
  - generiranje kode iz modelov UML
- Posamezni diagrami so uveljavljeni in so v uporabi že dalj časa
  - razredni diagram – osnova pri razvoju objektno usmerjenih programov
  - diagram aktivnosti – osnova pri specifikaciji proceduralnega vedenja (razširitev diagrama poteka)
  - diagram stanj – izhodišče pri načrtovanju uporabniških vmesnikov in drugih vrst reaktivnih sistemov

Prilagoditve za načrtovanje sistemov – SysML

LMSV LAMS