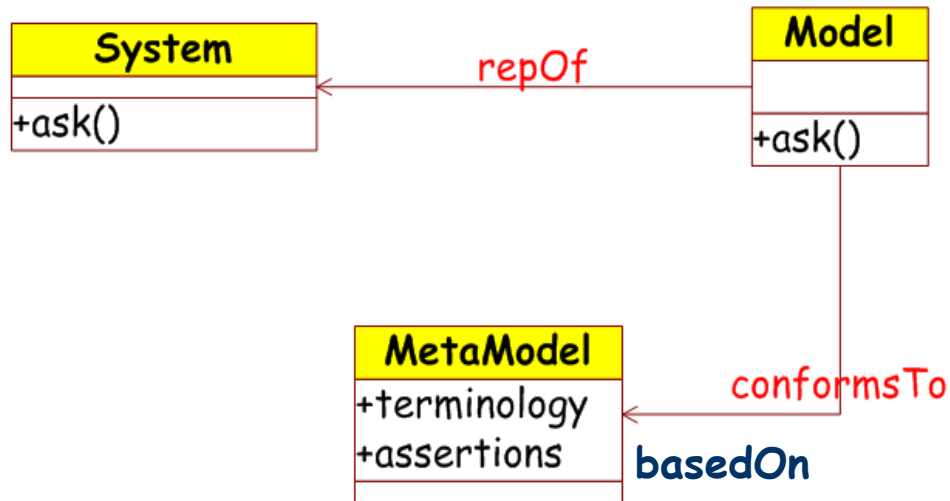


1. Praktiskā darba rezultāti

- ▶ Novērtējumi ir ORTUS e–studiju vidē sadaļā Vērtējumi.
- ▶ Tipiskās kļūdas:
 - Visi aizmirsas par atribūtiem, izņemot vienu cilvēku
 - Nosaukumos palika diakritiskas zīmes – garumzīmes un mikstinājumi
 - *Pastāvošām* klasēm nepieciešams paredzēt servisu, kas saglabā datus pirms lietojuma beigšanās un atjauno datus pēc lietojuma starta
 - “*Agregāts/nedalāma klase*”:
 - 1. Sadarbības klašu sarakstam nedrīkst būt tukšam
 - 2. Sadarbības pretējo virzienu (atbildes virzienu!) nav jānorāda

Sistēma, modelis, metamodelis, metametamodelis (1)



Metamodelis ir modeļa leģenda.

Metametamodelis ir metamodeļa leģenda.

Sistēma, modelis, metamodelis, metamodelis (2)



Sistēma S



Modelis A

Okeāns
Kontinents
Jūra
Līcis
Kalns
...

Modeļa A metamodelis

*Konkrēta
sintakse,
Abstrakta
sintakse,
Formēšanas
likumi.*

*Metamodelis ir modeļa leģenda.
Metamodelis ir metamodelis leģenda.*

Sistēma, modelis, metamodelis, metamodelis (3)



Sistēma S



Modelis B

Valsts
Galvaspilsēta
Lielā pilsēta
Robeža
Karogs
Platība

*Konkrēta
sintakse,
Abstrakta
sintakse,
Formēšanas
likumi.*

Modelis B metamodelis

Metamodelis ir modeļa leģenda.

Metamodelis ir metamodelis leģenda.

Modeļa B metamodelis



Modelis B

Konkrēta sintakse	Abstrakta sintakse	Formēšanas likumi
Iekrasota platība, kas ir atdalīta ar robežas līniju	Valsts	Katrai valstij ir sava unikāla krasa. Katrai valstij ir viena galvaspilsēta. ...
⊙	Galvaspilsēta	Var būt tikai viena vienai valstij
●	Pilsēta	
-----	Robeža	Apzīmē valsts robežu

Metametamodelis



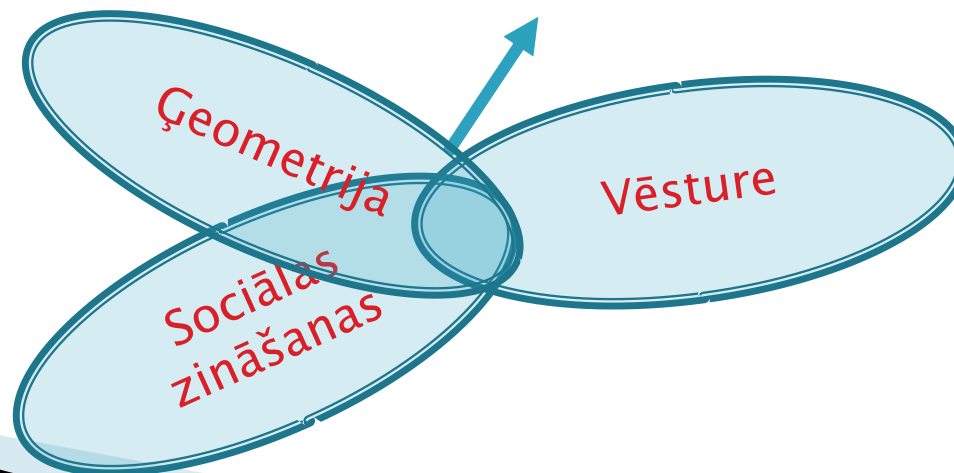
Modelis B

Konkrēta sintakse	Abstrakta sintakse	Formēšanas likumi
Iekrasota platība, kas ir atdalīta ar robežas līniju	Valsts	Katrai valstij ir sava unikāla krāsa. Katrai valstij ir viena galvaspilsēta. ...
⊙	Galvaspilsēta	Var būt tikai viena vienai valstij
●	Pilsēta	
-----	Robeža	Apzīmē valsts robežu



Modeļa B metamodelis

Modeļa B
metamodeļa
metamodelis?



Platformneatkarīgais (PIM) un platformai specifiskais (PSM) modeļi (uz UML klašu diagrammas piemēra)

2. Lekcija

Ērika Asņina, “Modeļvadāmas programmatūras izstrādes
praktikums”, RTU, 2011

Ieskats UML vēsturē

- ▶ 1994. gadā bija vairāk par 50 OO metodēm
 - Fusion, Shlaer–Mellor, ROOM, Class–Relation, Wirfs–Brock, Coad–Yourdon, MOSES, Syntropy, BOOM, OOSD, OSA, BON, Catalysis, COMMA, HOOD, Ooram, DOORS ...
- ▶ OO metožu īpašības
 - “Metamodeļi” līdzīgi viens otram
 - Dažāda grafiskā notācija
- ▶ **BET industrija pieprasa standartu!**

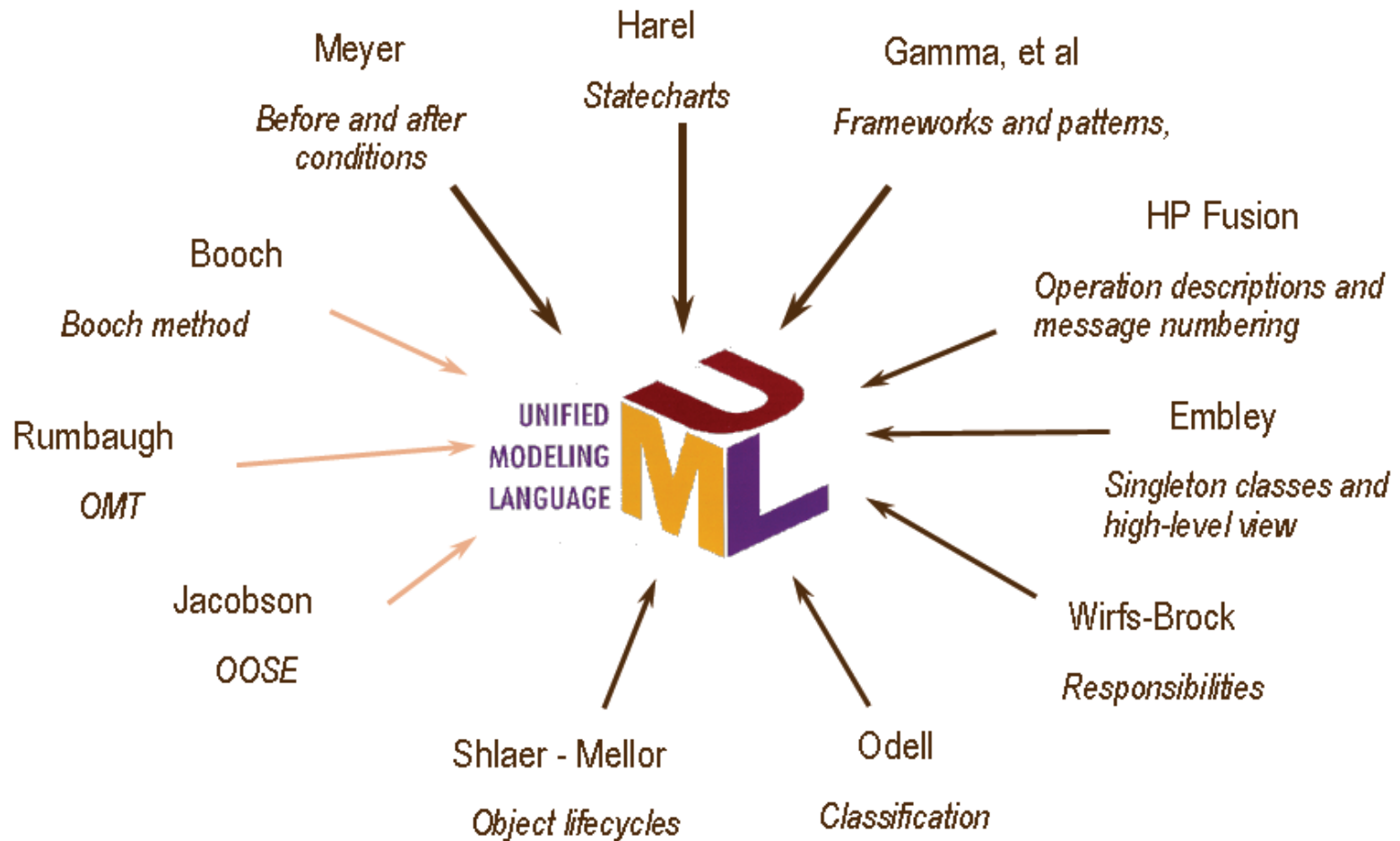
Ieskats UML vēsturē (turp.)

- ▶ 1994. – 1995. gads
 - 3 amigo: Grady Booch (Booch's method) + Jim Rumbaugh (OMT) + Ivar Jacobson (OOSE)
 - Motivācija
 - Kopīgā attīstība, samazinot atšķirības
 - Semantikas un notācijas unifikācija
 - Metožu trūkumu skaita samazināšana un metodes labāka uzturēšana
- ▶ 1996. gads – UML 0.9 un UML 0.91
- ▶ **BET industrija joprojām pieprasa standartu!**

Ieskats UML vēsturē (turp.)

- ▶ 1996. gads – OMG RFP for UML 1.0:
 - Digital Equipment Corp., HP, i-Logix, IntelliCorp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Rational Software, TI, un Unisys
- ▶ 1997. gads – UML 1.0
- ▶ ... UML 1.1, UML 1.3., UML 1.4 (1.4.2 = ISO/IEC 19501), UML 1.5.
- ▶ 2003. gadā pēc lielas pārskates iznāca UML 2.0
- ▶ 2007. gadā – UML 2.1.2 (ISO/IEC 19505, izstrādē)
- ▶ 2009. gadā – UML 2.2
- ▶ 2010. gadā – UML 2.3
- ▶ 2011. gadā – UML 2.4 Beta 2 (spēkā esošā OMG grupā)

UML ietver... (pēc R.Wuyts)



Roel Wuyts - ULB - Génie Logiciel et Gestion de Projets - 2004/2005

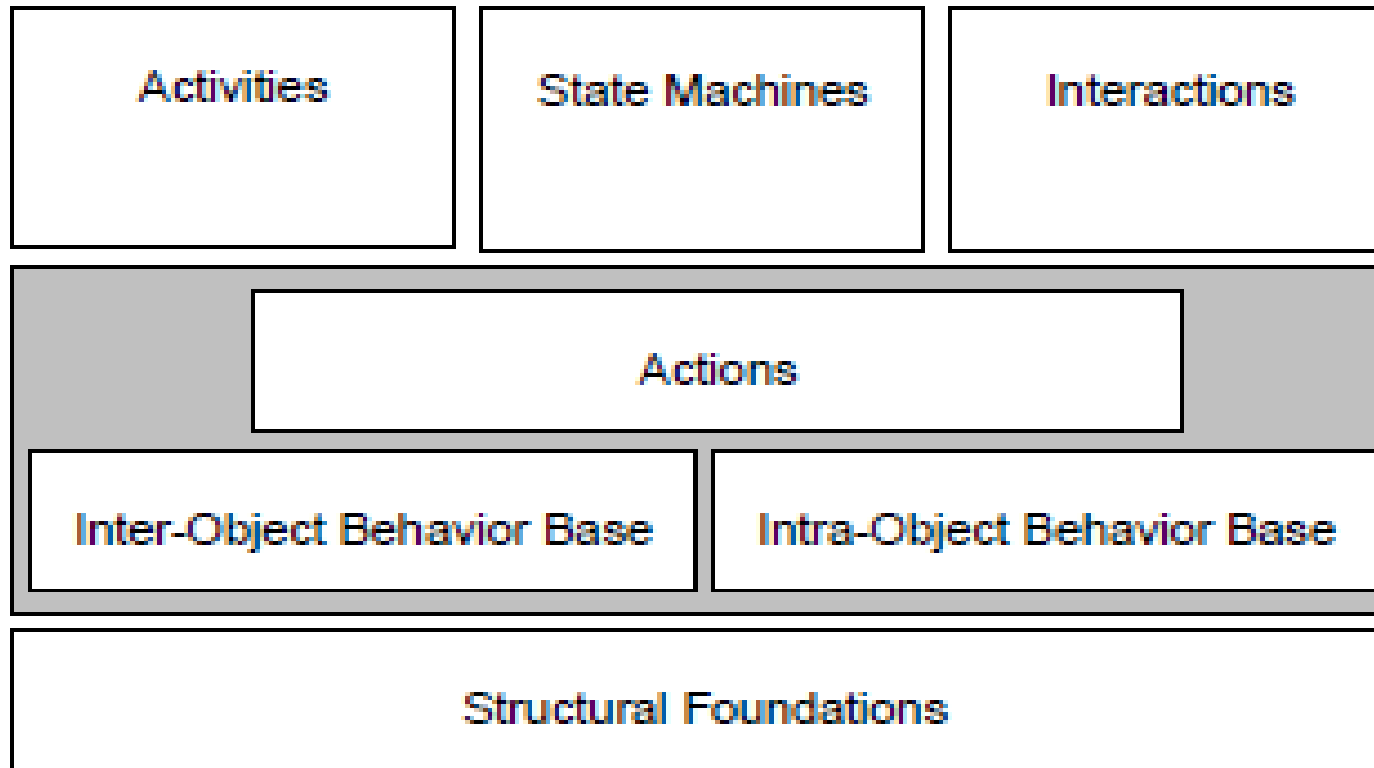
Pieejas UML lietošanai:

- ▶ UML kā skice – diagrammās ieskicē programmatūras sistēmas vizuālo izskatu (izmanto sākotnējos posmos; neuztur)
- ▶ UML kā projektējums – diagrammās atspoguļo programmatūras sistēmas detaļas formālāk un precīzāk, nekā skicē (uztur; viens no projektu rezultātiem)
- ▶ UML kā izpildāmais modelis – izmantojot MDA UML modeļi var būt izmantoti kā programmēšanas valoda (piem. xUML)

UML 2 struktūra

- ▶ Diagrammas ir skats uz modeli
- ▶ **Pašas diagrammas nav modelis!**
- ▶ UML modelis izskata sistēmu no diviem aspektiem:
 - Statiskā struktūra – kādu objektu tipi ir svarīgi un kā tie ir saistīti
 - Dināmiskā uzvedība – šo objektu dzīvesciklu apraksts un kā šie objekti mijiedarbojas, lai nodrošinātu prasītu sistēmas funkcionalitāti

UML2 semantiskie apgabali

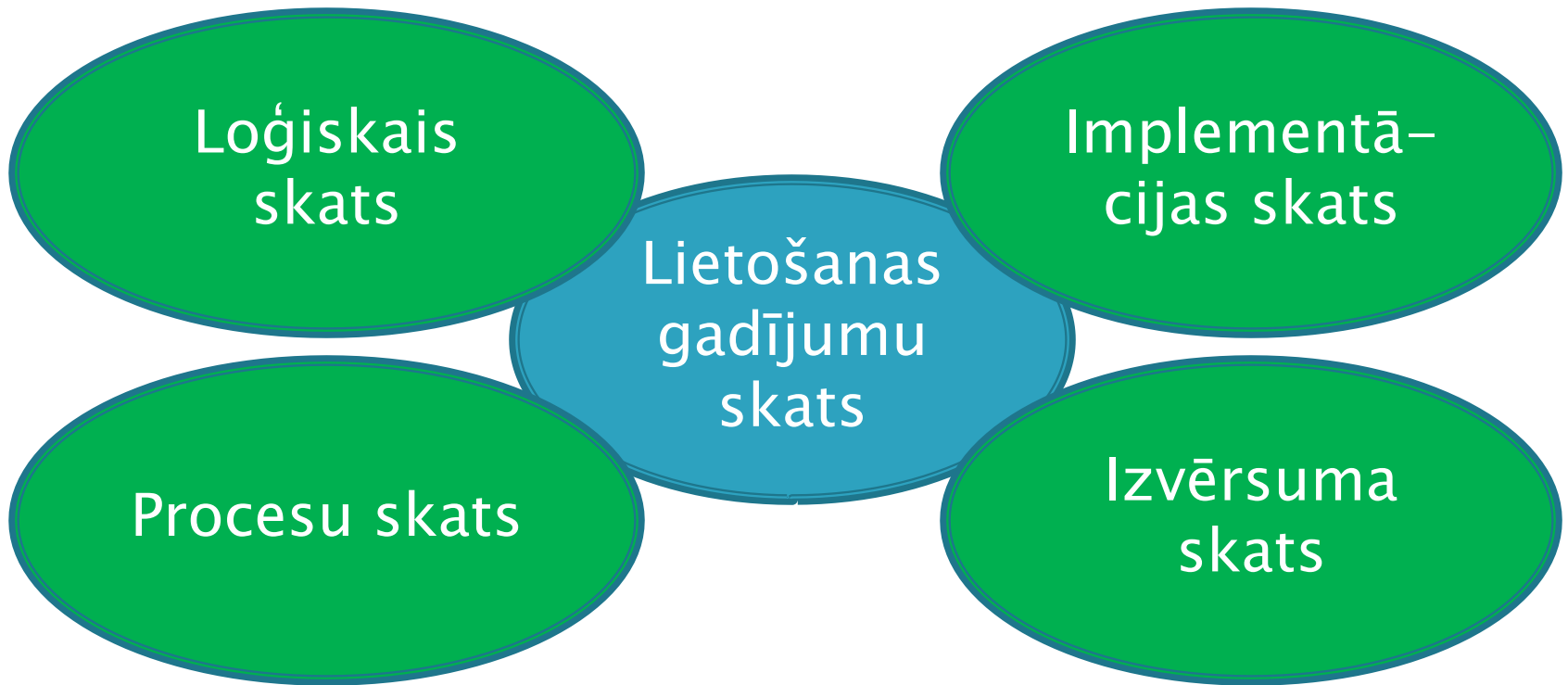


- ▶ No [OMG Unified Modeling Language, Superstructure, version 2.4., 2011. At: <http://www.omg.org/spec/UML/2.4/Superstructure>]

UML 2 diagrammas

- ▶ Struktūras diagrammas
 - **Klašu diagramma (Class Diagram)**
 - Salikto struktūru diagramma (Composite Structure Diagram)
 - Komponentu diagramma (Component Diagram)
 - Izvēsuma diagramma (Deployment Diagram)
 - Objektu diagramma (Object Diagram)
 - Pakotņu diagramma (Package Diagram)
- ▶ Uzvedības diagrammas
 - Aktivitāšu diagramma (Activity Diagram)
 - Mijiedarbības diagramma (Interaction Diagram)
 - Secību diagramma (Sequence Diagram)
 - Komunikāciju diagramma (Communication Diagram)
 - Mijiedarbības kopainas diagramma (InteractionOverview Diagram)
 - Laika noteikšanas (hronometrāža) diagramma (Timing Diagram)
 - Lietošanas gadījumu diagramma (UseCase Diagram)
 - Stāvokļu mašīnas diagramma (StateMachine Diagram)

UML skati (4+1)



UML lietošanas gadījumu skats

- ▶ Parāda sistēmas funkcionalitāti kā to uztver pasūtītāji (pamata prasības)
 - Aktieris – lietotājs vai citā sistēma
 - Tiek aprakstīts, izmantojot lietošanas gadījumu diagrammu, aktivitāšu diagrammu, reti – mijiedarbību diagrammas
 - Centrālais skats, kas vada citu skatu izstrādi
 - Izmanto pasūtītāji/klienti, projektētāji, izstrādātāji, testētāji
- ▶ No MDA skatupunkta – skats uz CIM modeli (dažās pieejās to uzskata arī skatu uz PIM)

UML loģiskais skats

- ▶ Parāda problēmas domēna vārdnīcu – objektus un klases.
- ▶ Akcentē kā sistēmas funkcionalitāte ir nodrošināta
 - Izmanto pakotņu, klašu, objektu un saliktu struktūru diagrammas, lai attēlotu statisku struktūru
 - Izmanto stāvokļu mašīnas diagrammu, lai attēlotu dināmisku uzvedību
 - Pielieto analītiķi, testētāji, pasūtītāji
- ▶ No MDA skatupunkta – skats uz PIM (daļēji arī CIM) un PSM modeļiem

UML procesu skats

- ▶ Parāda sistēmas izpildāmas plūsmas un procesus aktīvu klašu formā
 - Izmanto klašu, salikto struktūru un objektu diagrammas
 - Loģiskā skata uz procesiem orientēts paveids, kurš satur tos pašus artefaktus
 - Pielieto analītiķi, testētāji un sistēmas integratori
- ▶ No MDA skatupunkta – skats uz PIM un PSM modeļiem

UML implementācijas skats

- ▶ Parāda failus un komponentes, kas sastāda sistēmas fiziskā koda pamatu
 - Izmanto komponentu diagrammas
 - Ilustrē atkarības starp komponentēm
 - Ilustrē komponentu kopas konfigurācijas pārvaldību, lai definētu sistēmas versiju
 - Pielieto projektētāji un programmētāji
- ▶ No MDA skatupunkta – skats uz PSM modeli

UML izvērsuma skats

- ▶ Parāda artefaktu izvērsumu fizisko skaitļošanas mezglu (datori, perifērija) kopā
 - Ļauj modelēt artefaktu izkārtojumu izkliedētu sistēmu mezglos
 - Izmanto izvērsuma un hronometrāža diagrammu
 - Pielieto sistēmas inženieri, projektētāji un programmētāji
- ▶ No MDA skatupunkta – skats uz PSM modeli

UML atbalsts MDA izmantošanai

- ▶ Valodu saime
 - UML + Profilēšanas mehānisms (pielietošanas sfērām, platformām, izstrādes pieejām, u.c.)
- ▶ Sistēmas apraksts neatkarīgi no platformām
 - Loģisko un fizisko modeļu izstrāde
- ▶ Platformu aprakstīšanai
 - Profilēšanas mehānisms, platformai specifiskās izpildes konstrukcijas
- ▶ Atsevišķās platformas izvēle sistēmai*
- ▶ Sistēmas specifikācijas transformācija specifikācijā konkrētai platformai*
 - PIM→PSM (Realization, Refine un Trace)

PIM un PSM uz UML klašu diagrammas piemērā

- » Elementi, attiecības, pieraksta formāts PIM un PSM modeļiem

Objekts, klase, abstrakta klase

ModelDrivenArchitecture : Course

name : string
id : long = 0

- Objekta vārds: Klases vārds (ar pasvītrojumu)
- atribūta vārds: atribūta tips = atribūta vērtība

«implementation class» Course

-name : string
-id : long = 0
+getDescription() : string

<<stereotips>> Klases vārds (ar lielu burtu)
Atribūta redzamība, vārds (ar mazu burtu), tips, vērtība pēc noklusēšanas
Operācijas redzamība, vārds (ar mazu burtu), parametri, kols, atgriežamās vērtības tips

<i>Person</i>

-name
-surname
-personCode
+getBirthDate() : int

Abstraktās klases vārds (*italic*) !

Abstraktajai klasei nevar inicializēt objektu!

Atribūta definīcija

- ▶ Atribūts ir klases vai objekta raksturojošā īpašība
- ▶ Atribūts var būt *iekļauts* (nested) vai *attiecības veidā*
- ▶ Biznesa/konceptuālajā līmenī (CIM līmenis)
 - `</atribūta vārds {var būt norādīti ierobežojumi}>`
 - / – vai ir atvasināts atribūts
name {līdz 25 simboliem}
- ▶ Analīzes/projektējuma līmenī (PIM līmenis)
 - `<redzamība / vārds [daudzkāršība] : tips {ierobežojums}>`
– name : String {līdz 25 simboliem}
- ▶ Detalizētā projektējuma līmenī (PSM līmenis)
 - Atribūta tips un redzamība atbilst platformā specificētajiem
 - Ierobežojums ir uzrakstīts formālā valodā (piem., programmēšanas valodā vai OCL valodā)

Redzamība (visibility)

- ▶ Izmanto klases atribūtiem un operācijām, kā arī attiecību lomu vārdiem
- ▶ Implementācijas valodās var dažādi interpretēt visus redzamības veidus, izņemot publisku

Redzamība	UML semantika	Java semantika	C# semantika
public (+)	Elements, kam ir piekļuve šai klasei	Kā UML	Kā UML
private (-)	Tikai šīs klases operācijas	Kā UML	Kā UML
protected (#)	Tikai šīs klases vai "bērna" klases operācijas	Kā UML + klases no tas pašas pakotnes	Kā UML
package (~)	Elements no tas pašas pakotnes vai ligzdotas pakotnes	Pēc noklusēšanas ligzdotām klasēm ligzdotās apakšpakotnēs piekļuve nav	-----

Redzamība (turpinājums)

Redzamība	UML semantika	Java semantika	C# semantika
private protected	-----	Tas pats kā UML <i>protected</i>	-----
internal	-----	-----	Pieejams jebkuram elementam tajā pašā programmā
package	-----	-----	Kombinē <i>protected</i> un <i>internal</i> semantiku – izmanto tikai atribūtiem

Platformu neatkārīgie tipi

	Primitīvs tips	Semantika
UML	Integer	Vesels skaitlis
	UnlimitedNatural	Vesels skaitlis ≥ 0 Bezgalību parāda ar zvaigznīti “*”
	Boolean	Var pieņemt vērtības <i>true</i> vai <i>false</i>
	String	Simbolu virkne String tipa literālis tiek ņemts pēdiņās, piemēram, “Balvis”
OCL	Real	Skaitlis ar peldošo punktu

Primitīvi tipi ir iekšēji UML tipi, kas nav saistīti ne ar vienu platformu!
Tipu nosaukumi sākas ar lielu burtu!

Daudzkāršība (multiplicity)

- ▶ Izmanto projektēšanas un analīzes modeļos
- ▶ Parāda “priekšmetu/lietu skaitu”, kuri piedalās kādā attiecībā
- ▶ Ļauj modelēt divas pavisam atšķirīgas lietas:
 - Kolekcijas (collections)
 - Nulles vērtības
- ▶ Piemērs:

PersonDetails
-name[2..*] : String
-address[3] : String
-emailAddress[0..1] : String

➤ *name* ir sastādīts no diviem vai vairāk simbolu virknēm

➤ *address* ir sastādīts no trīs simbolu virknēm

➤ *emailAddress* ir sastādīts no vienas simbolu virknes vai *null*

Operācijas definīcija

- ▶ Operācija ir funkcija, kura ir piesaistīta konkrētai klasei. Klases metode ir operācijas realizācija.
- ▶ Operācija attēlo klases *atbildību* par sistēmas uzvedības nodrošināšanu (*par servisu*)
- ▶ Biznesa līmenī (CIM) norāda **operācijas vārdu**
 - Operācija attēlo klases atbildību (*responsibility*).
Pieraksta kā darbību, piemēram, getBirthDate();
- ▶ Analīzes un projektējuma līmenī (PIM, PSM) norāda
 - Vārdu
 - Parametru sarakstu
 - Atgriežamas vērtības tipu

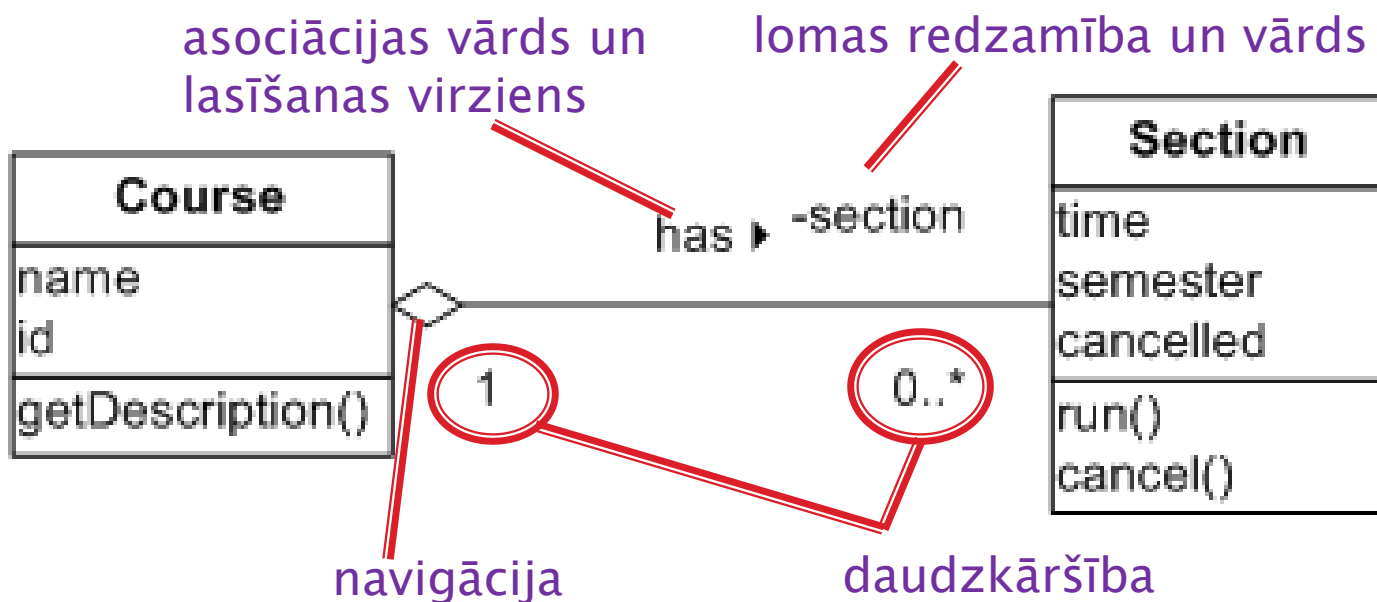
Operācijas definīcija (turp.)



Saite, asociācija, loma

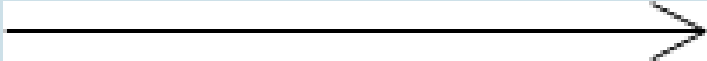
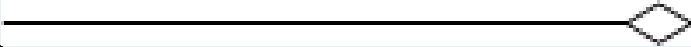



- ▶ Saite –**objektu** semantiskais savienojums, kas ļauj vienam objektam sūtīt otram objektam ziņojumu (**sadarboties!**)
- ▶ Asociācija – attiecība starp **klasēm**. Ja starp objektiem ir saite, tad starp šo objektu klasēm obligāti ir jābūt asociācijai!
- ▶ Asociācijas sintakse:
 - asociācijas vārds
 - lomu vārdi
 - daudzskāršība
 - navigācija

Asociācijas pieraksts

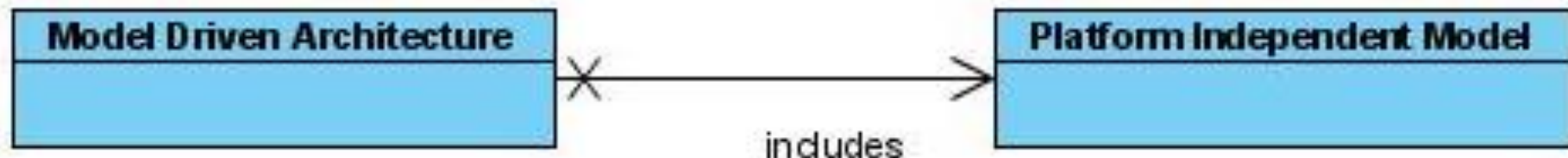


Jautājums: kādam MDA modelim atbilst šis klašu pieraksts? Kāpēc?

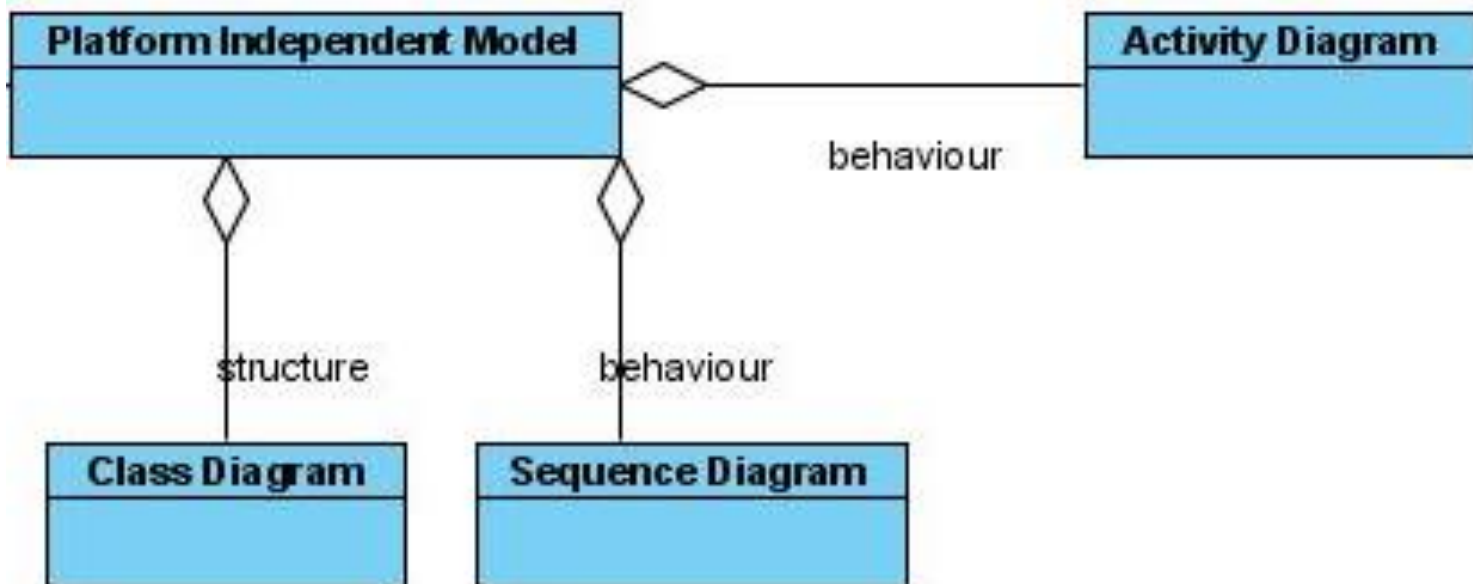
Asociāciju veidi

Nosaukums	Sintakse un semantika
Asociācija	
Agregācija	 Daļa Vesels
Kompozīcija	 Daļa Vesels
Atkarība	 Atkarīgais elements Neatkarīgais elements
Vispārinājums	 Apakšklase Superklase

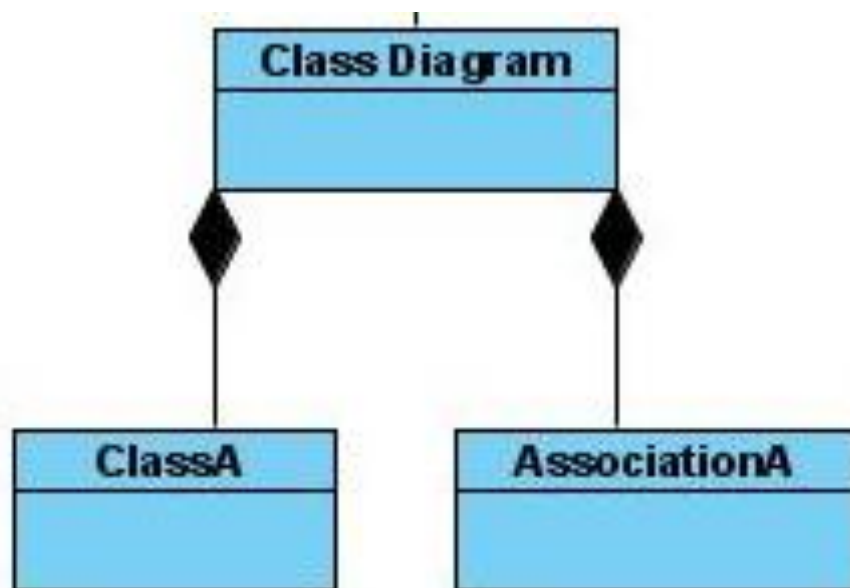
Asociācijas piemērs



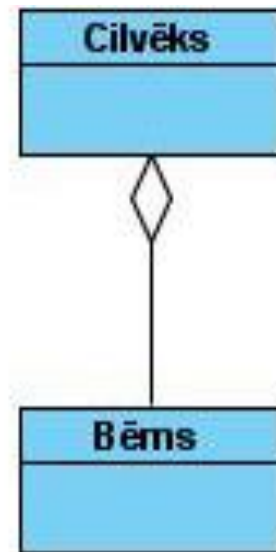
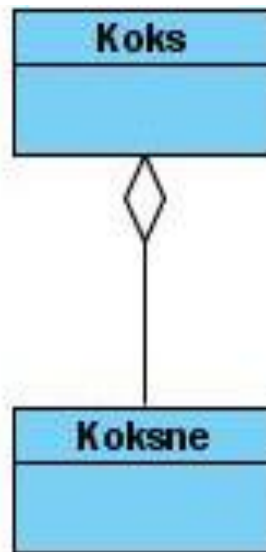
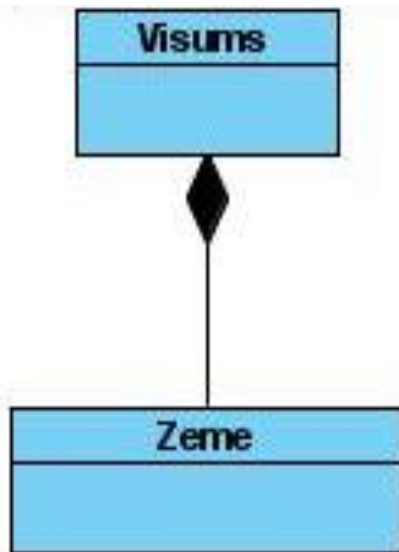
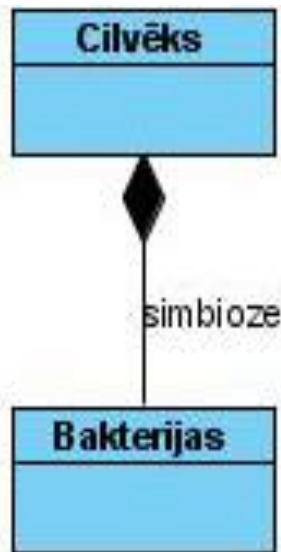
Aggregācijas piemērs



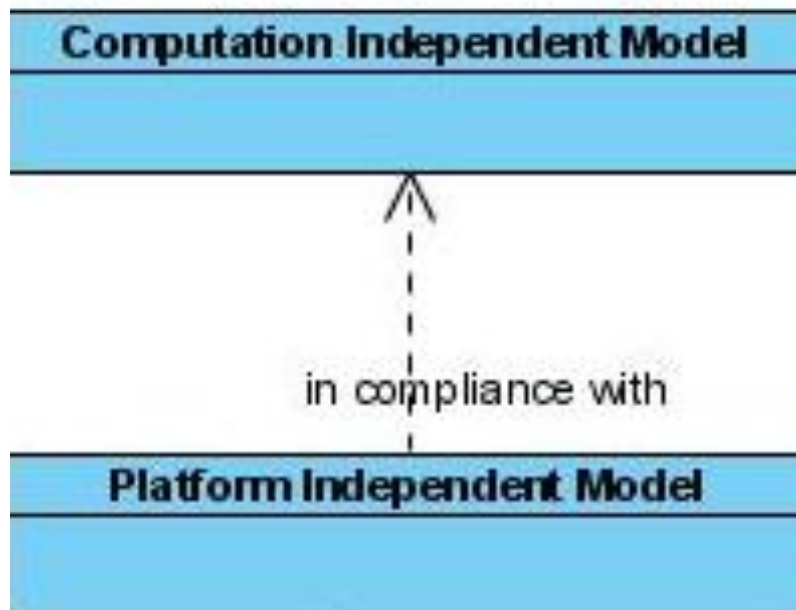
Kompozīcijas piemērs



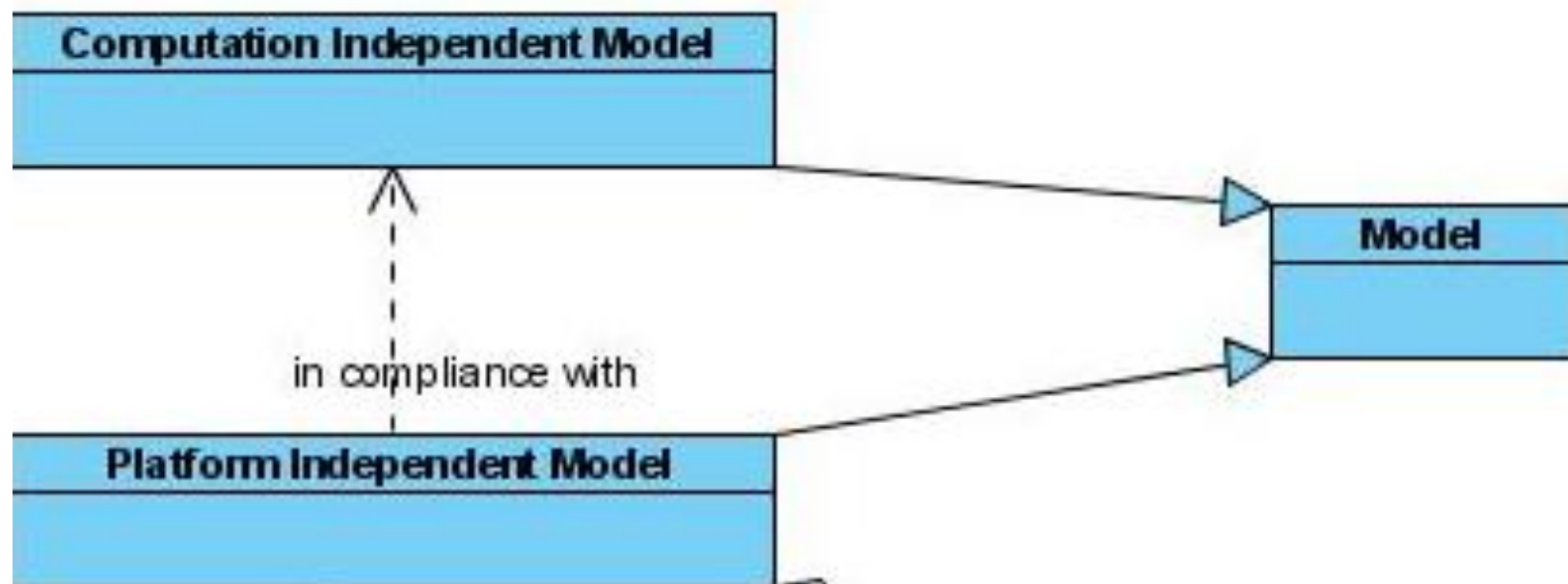
Agregācija vs Kompozīcija



Atkarības piemērs



Vispārinājuma piemērs



Daudzkāršība

Pieraksts	Semantika
0..1	Nulle vai viens
1	Tieši viens
0..*	Nulle vai vairāk
*	Nulle vai vairāk
1..*	Viens vai vairāk
1..6	No viena līdz sešiem
1..3, 7..10, 15, 19..*	No 1 līdz 3 <i>vai</i> no 7 līdz 10 <i>vai</i> tieši 15 <i>vai</i> 19 un vairāk

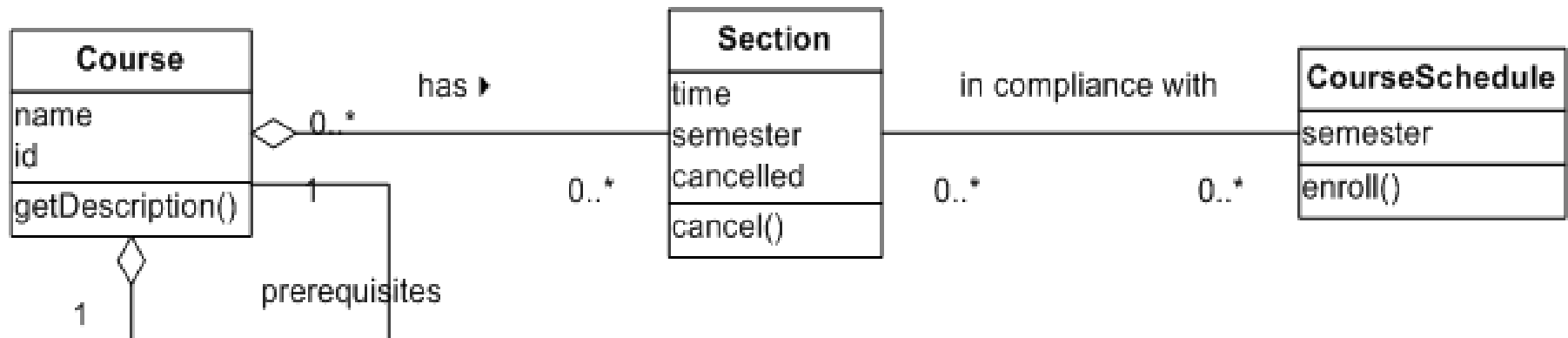
Attiecību standarta īpašības

Standarta īpašība	Semantika
{ordered}	Elementi kolekcijā tiek glabāti striktā secībā
{unordered}	Elementiem kolekcijā nav nekādas definētas secības
{unique}	Katrs elements kolekcijā ir unikāls – kolekcijā elements parādās tikai un vienīgi vienu reizi
{nonunique}	Kolekcijā ir atļauti dublējoši elementi

Attiecību īpašību piemēri

- ▶ {a, b, c, d}
 - {unique, ordered}
- ▶ {a, a, b, c, d, e}
 - {nonunique, ordered}
- ▶ {a, c, b, e, d}
 - {unique, unordered}
- ▶ {a, a, e, e, c}
 - {nonunique, unordered}

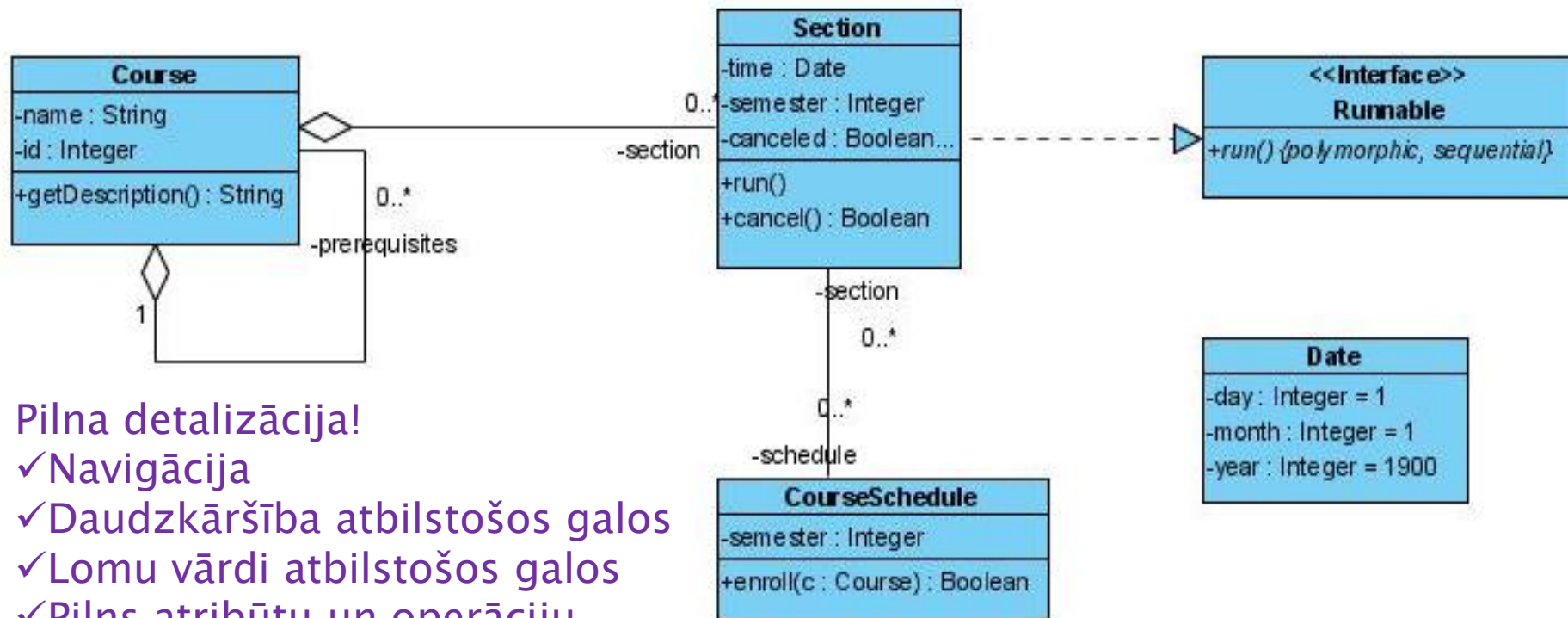
Klašu diagramma *CIM* līmenī



Klase ar atribūtiem un operācijām, nenorādot atribūtu un operāciju detaļas

Attiecības: konceptuālas attiecības (var nebūt navigācija, nav kompozīcija), daudzkāršība

Klašu diagramma *PIM un PSM* līmeņos



Pilna detalizācija!

- ✓ Navigācija
- ✓ Daudzkāršība atbilstošos galos
- ✓ Lomu vārdi atbilstošos galos
- ✓ Pilns atribūtu un operāciju pieraksts
- ✓ PIM – platformneatkarīgie tipi, PSM – platformspecifiskie tipi un platformspecifiskie stereotipi