Programmatūras attīstības tehnoloģijas:

Modeļvadāma arhitektūra (MDA)

Dr.sc.ing., asoc. prof. Oksana Ņikiforova
DITF LDI
Lietišķo datorzinātņu katedra
Rīga - LV1048, Meža 1/3, 510.kab., tel.67 08 95 98
oksana.nikiforova@rtu.lv

Mūsdienas programmatūras izstrādes vides

- Tīklā un telpā sadalītas sistēmas
- Augošs pieprasījums pēc savienojamības kompānijas iekšā un starp kompānijām
- Heterogēnas platformas, tīkli, valodas un lietojumsistēmas
- Bezvada un rokās ievietojamas ierīces
- Topošas tehnoloģijas: XML, Web Services

Iespējamie saskaņas punkti

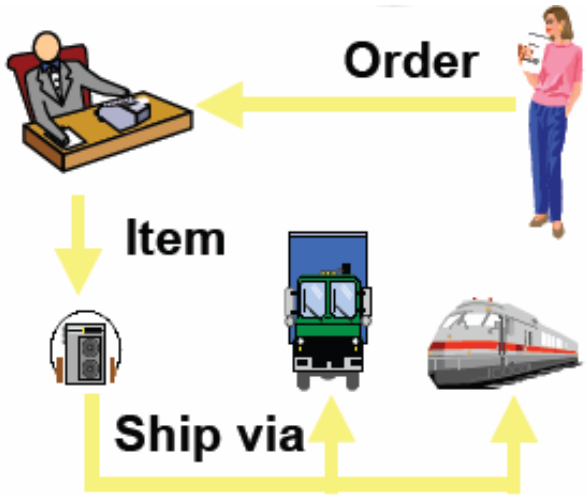
- Nav iespējams vienoties par aparatūras platformām
- Nav iespējams vienoties par operētājsistēmām
- Nav iespējams vienoties par tīkla protokoliem
- Nav iespējams vienoties par programmēšanas valodām
- *Ir jāvienojas par modeļiem, saskarni un sadarbspēju*

Programmēšanas valodu evolūcija

- Abstrakcijas līmeņa pacelšanos no 1GL līdz 4GL un atpakaļ uz 3GL
- 4GL ir produktīvas, bet ar zemu adoptācijas iespēju, lai realizētu specifiskās prasības
- Esošie 3GL pārsvarā ir standarta bibliotēkas un ietvari, kodēšanas sintakse ir mazāk svarīga

- Kāds būtu nākošais līmenis?

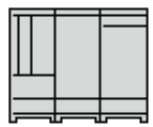
Vizuālā Modelēšana



Business Process

"Modeling captures essential parts of the system."
Dr. James Rumbaugh

Visual Modeling is modeling using standard graphical notations

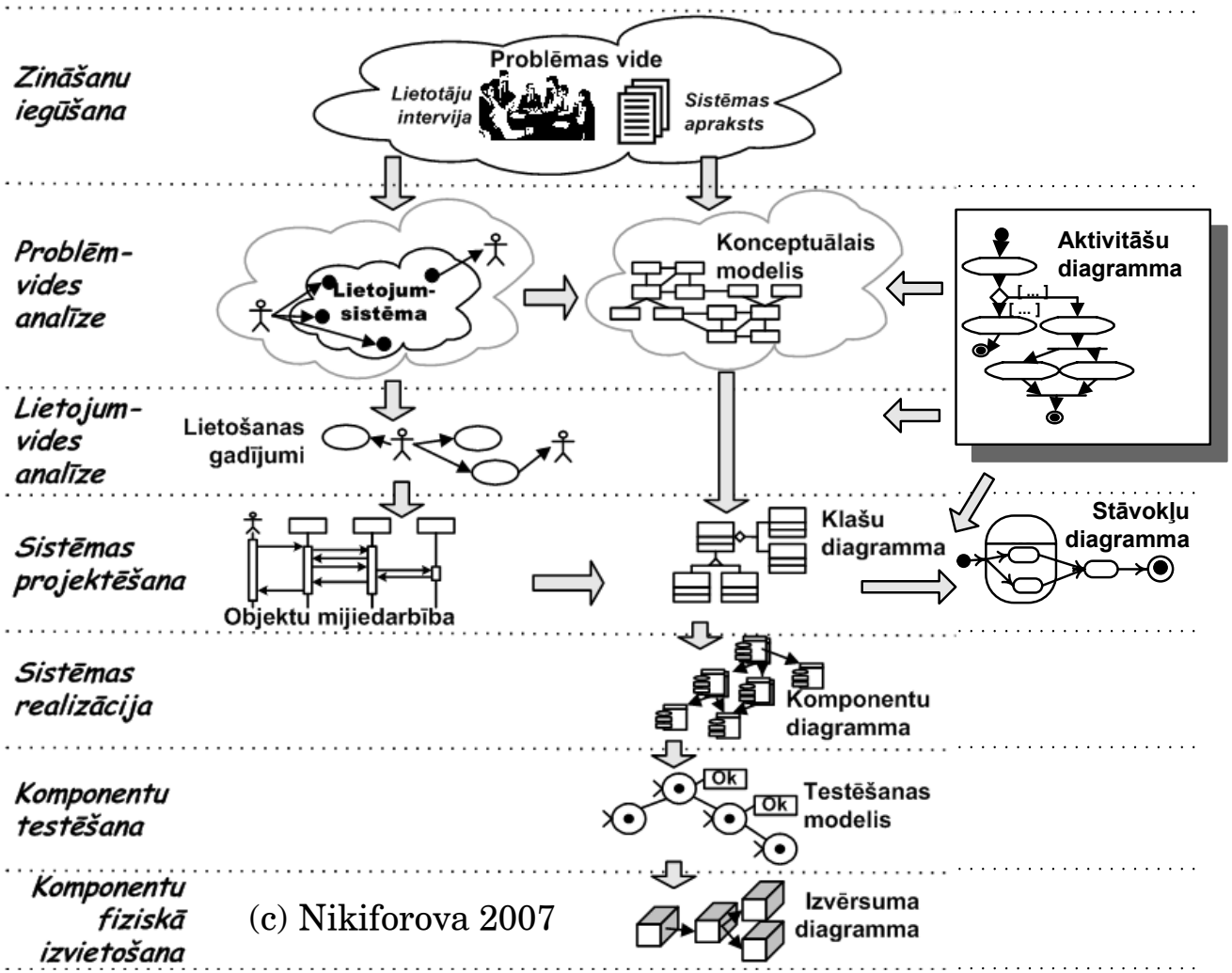


Computer System

Vienota modelēšanas valoda (UML)

- UML – Unified Modelling Language
- UML ir standarta modelēšanas valoda
- UML ir valoda programmatūras artefaktu specificēšanai, vizualizēšanai, konstruēšanai un dokumentēšanai
- UML nodrošina:
  - Modeļa elementi: koncepti un to semantika
  - Notācija: vizuāla modeļa elementu attēlošana
  - Vadlīnijas: Pieņēmumi elementu un to notācijas lietošanai

Modeļvadāmā programmatūras izstrāde, lietojot UML



Modeļvadāmā arhitektūra

- Model Driven Architecture (MDA)
- Object Management Group (OMG).
- Veids specificēt un uzbūvēt programmatūras sistēmu
  - Pamatā ir sistēmas modelēšana UML valodā
  - Atbalsta pilnu programmatūras izstrādes dzīves ciklu:
    - Analīze, projektēšana, implementēšana, izvietošana, uzturēšana, novērtēšana & integrācija ar esošām sistēmām

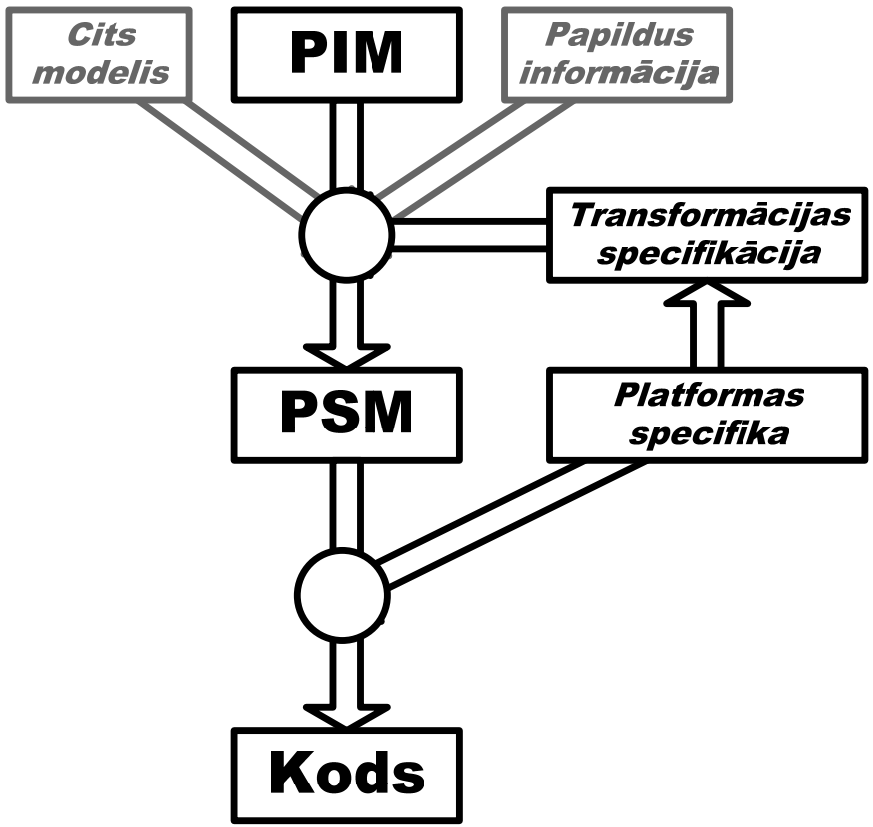
Modeļvadāmās arhitektūras principi

- Specificēt sistēmu neatkarīgi no tās realizācijas platformas
- Transformēt sistēmas specifikāciju vienā no izvēlētajām platformām

Modeļvadāmās arhitektūras īpašības

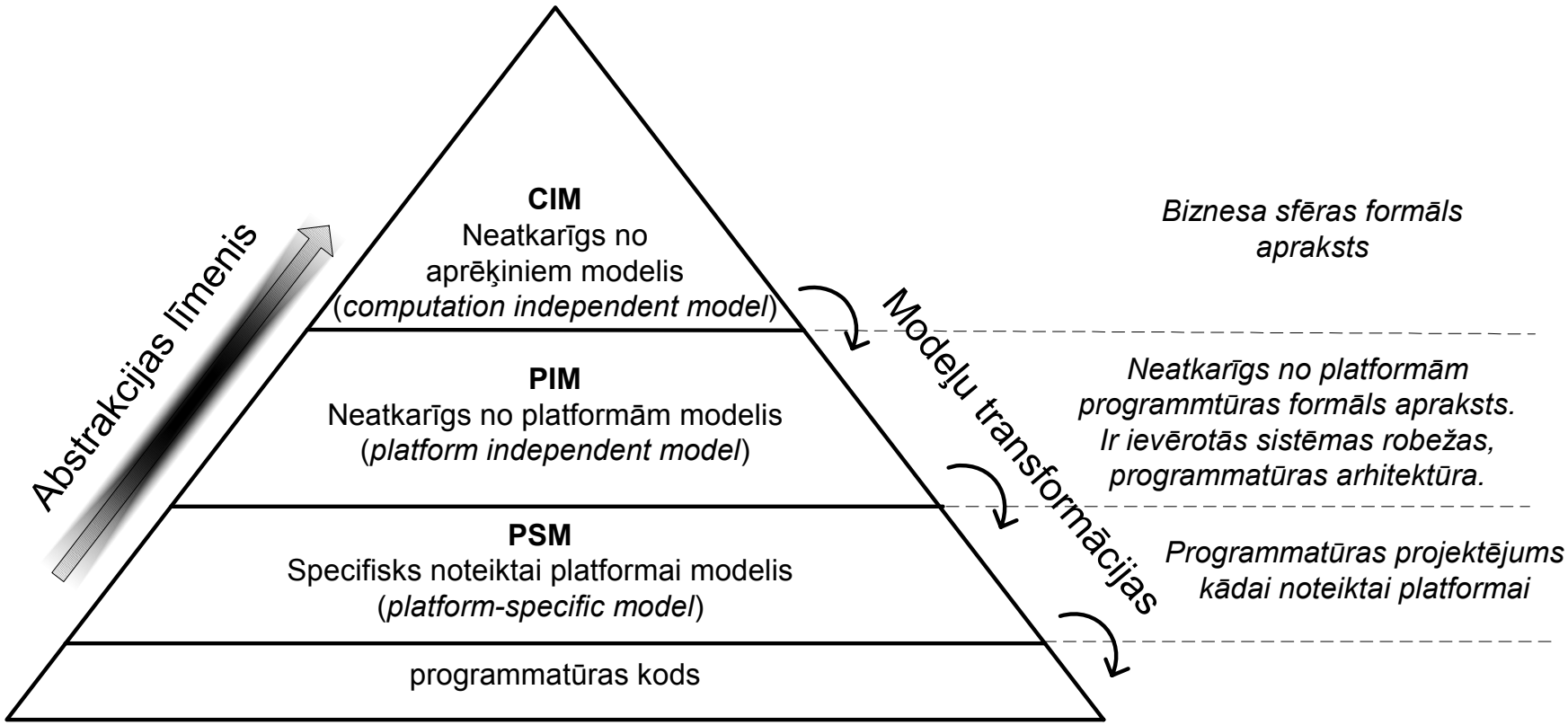
- Modeļu lietošana programmatūras izstrādes dzīves cikla vadībā
- Problēmvides zināšanu atdalīšana no platformas zināšanām
- Ātra pielāgošanās problēmvides un tehnoloģijas izmaiņām
- Strādājošo lietojumsistēmu ģenerēšana

Modeļvadamā arhitektūras etapi



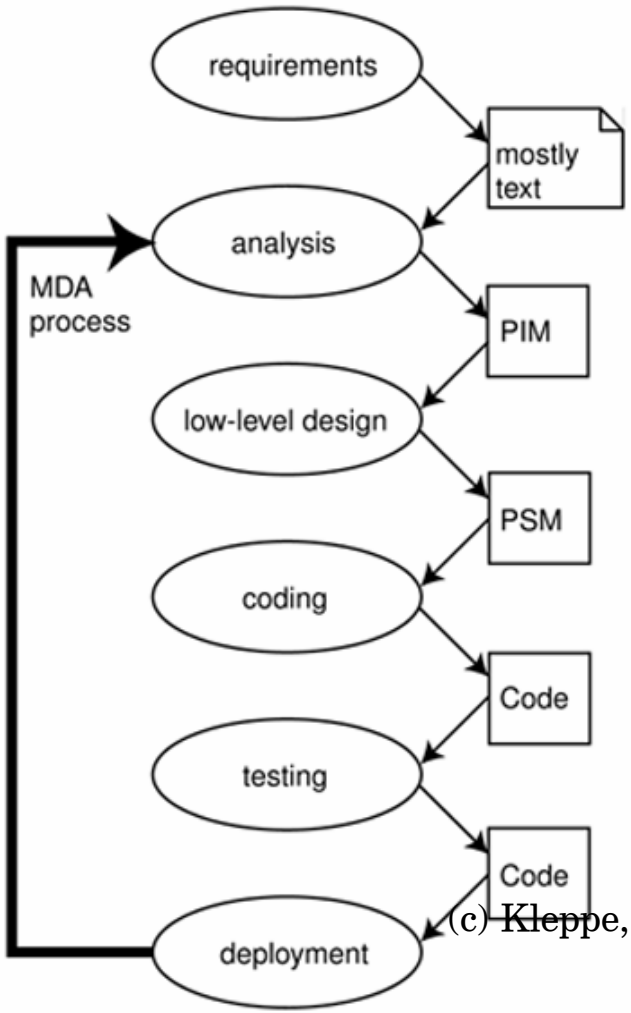
- Veidot no platformas neatkarīgo modeli (Platform Independent Model – PIM),
  - modeli, kas ir neatkarīgs no realizācijas tehnoloģijas
- Transformēt PIM vienā vai vairākos modeļos – Platformspecifiskos modeļos (Platform Specific Model - PSM),
  - modelis, kas ir piemērots sistēmas specificēšanai realizācijas konstrukcijas elementos, pieejamos konkrētajā realizācijas platformā
- Transformēt PSM kodā, kas ir automātiskā koda ģenerācija, jo PSM satur visas tai nepieciešamas detaļas

Modeļvadamā arhitektūra



(c) Nikiforova, Kuzmina, Pavlova 2006

MDA programmatūras izstrādes dzīves cikls



(c) Kleppe, Warmer, Bast 2003

Stereotipi

- Stereotipi ir "atslēgvārdi", kas var būt piekārtoti jebkuram elementam modelī (klases klase jeb klases tips)
- Stereotipi ir lietoti, lai
  - Nodrošinātu papildus semantisko nozīmi modelim
  - Definē modeļa elementu variācijas, kas var būt specifiski rīkam
  - Definē implementācijas un transformācijas nosacījumus

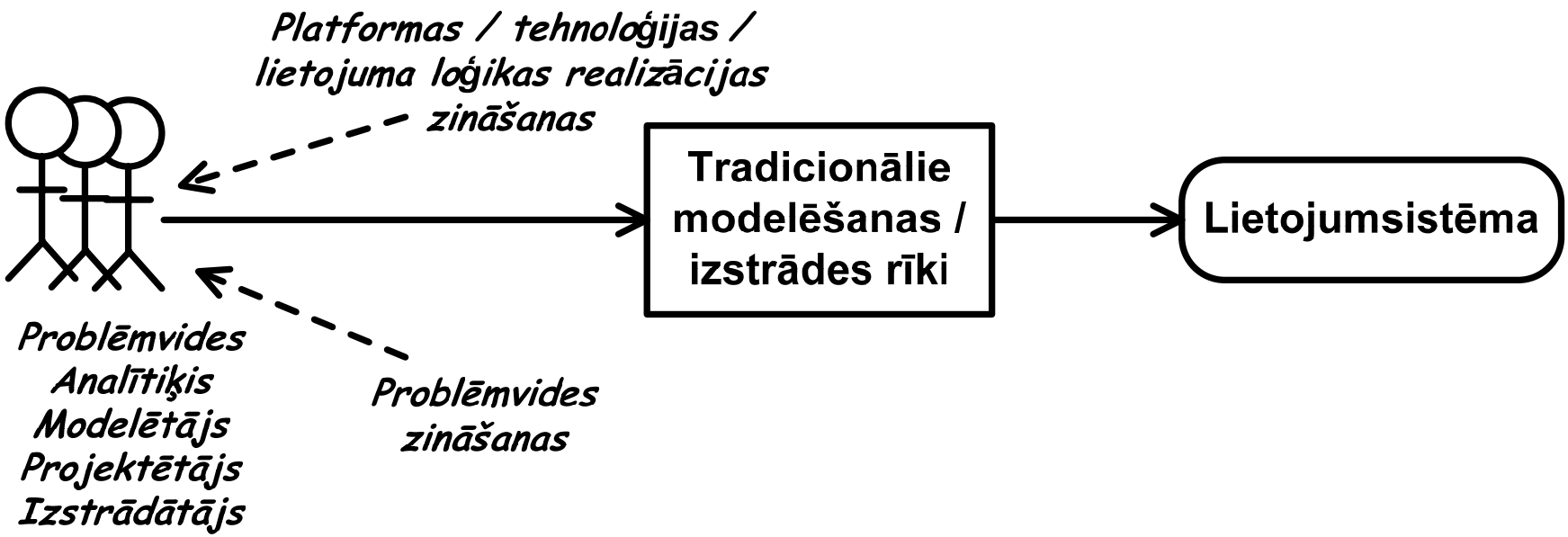
Tagged values = iezīmētas vērtības

- Iezīmēta vērtība ir nosaukums-vērtība pāris, kas tiek lietots, lai uzkrātu brīva formāta informāciju
- Iezīmētas vērtības ir lietotas,
  - lai definētu stereotipu atribūtus,
  - lai uzkrātu uz projektu attiecināmo informāciju vai specifisko rīka informāciju
  - lai nodrošinātu transformācijas vadību
- Iezīmēta vērtība ir papildus meta-atribūts, kas raksturo UML meta-klasi UML meta-modeli.
  - Iezīmēta vērtība līdzīga kā jebkuram atribūtam ir nosaukums un tips
  - Piekārtota stereotipam un bez stereotipa neeksistē

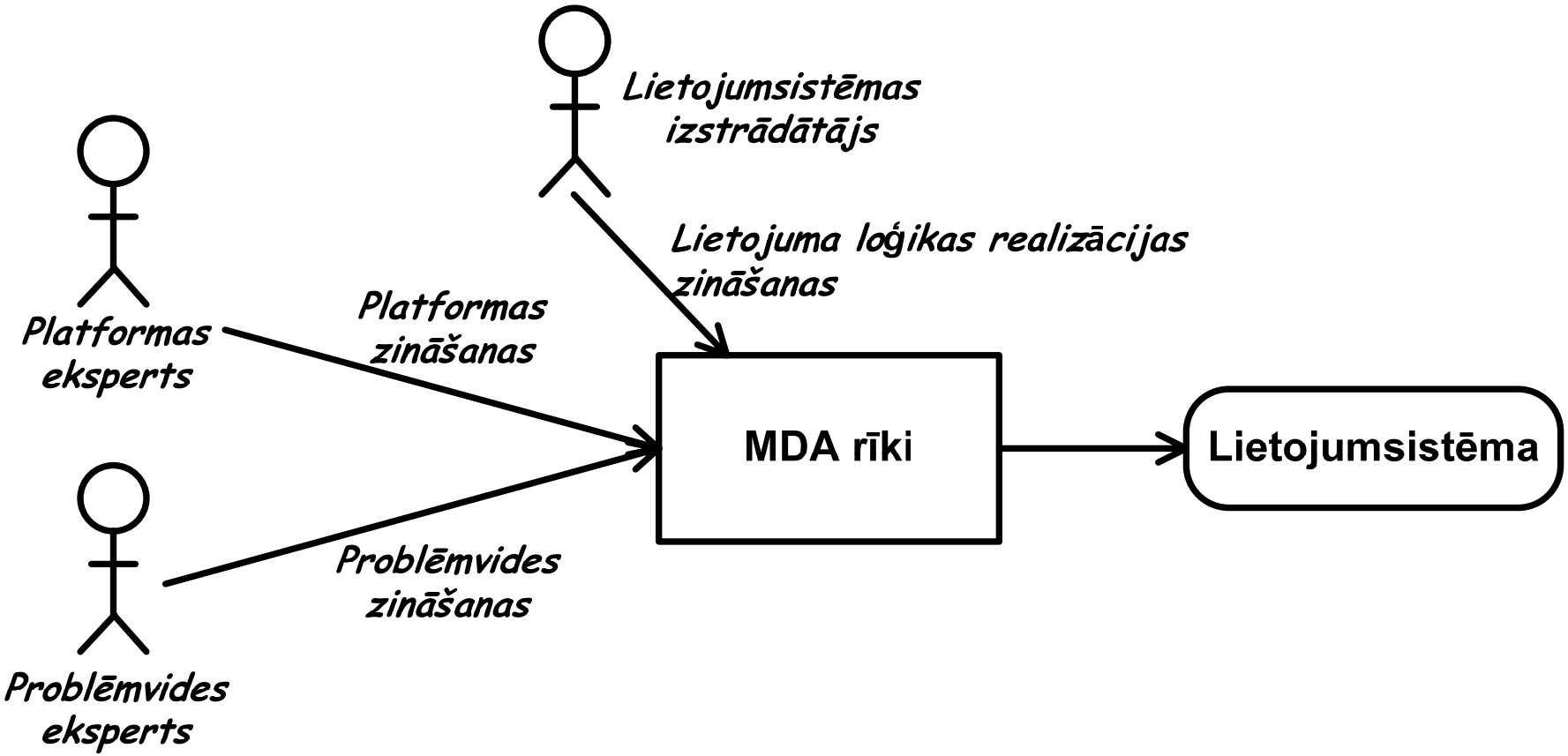
Profili (profile)

- Profils sastāv no iepriekšdefinētas stereotipu kopas, iezīmētām vērtībām (atribūtiem) un nosacījumiem
- Profili tiek lietoti, lai pielāgoto'noskaņojāmo modeli specifiskai problēmas videi, tehnoloģijai vai implementēšanai

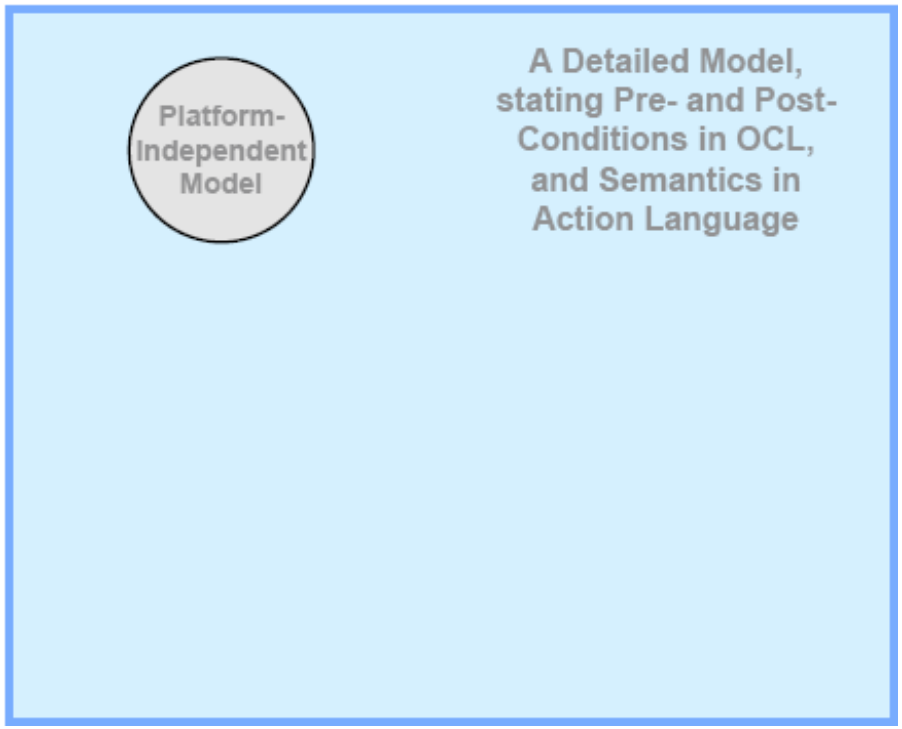
Tradicionālā modelēšana un izstrāde



Bāzēta uz MDA modelēšana un izstrāde

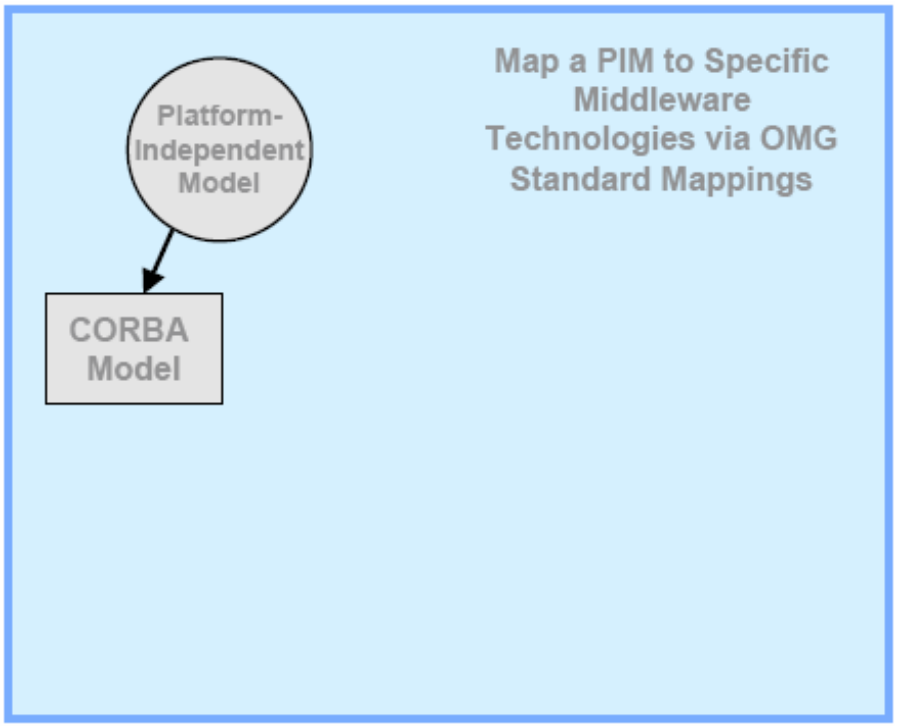


MDA lietojumsistēmas konstruēšana



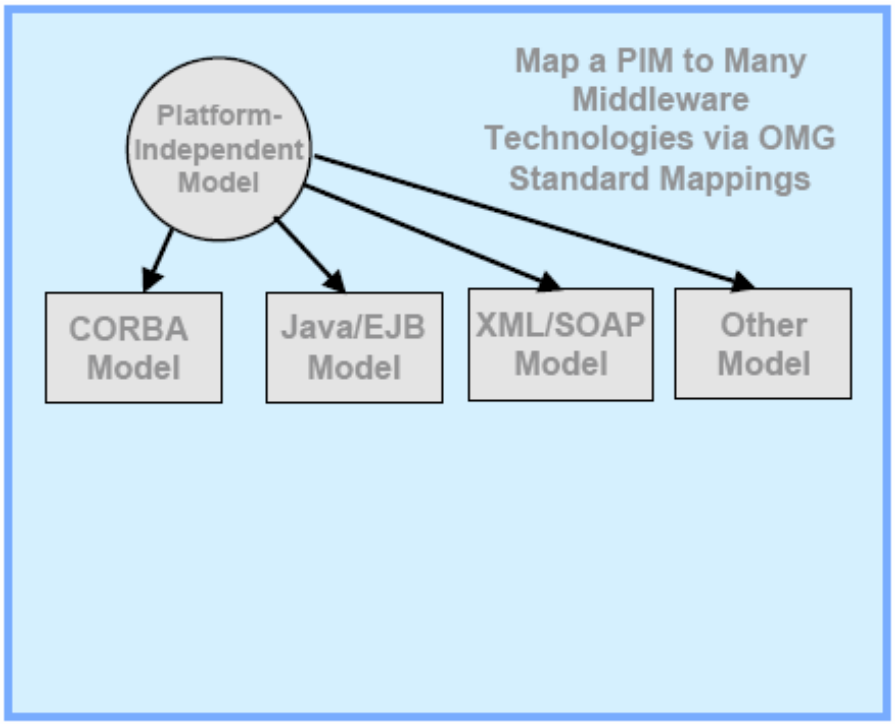
- Sāk ar platformneatkarīgo modeli, kas prezentē biznesa funkcionēšanu un loģiku, neattēlojot tehnoloģiskās detaļas.

Platformspecifiskā modeļa ģenerēšana



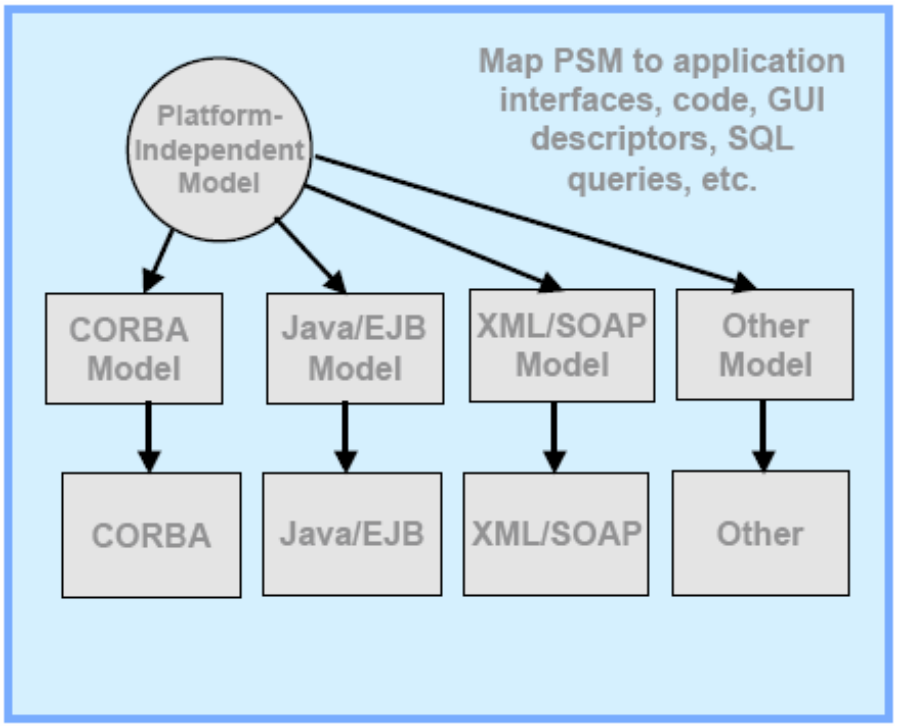
- MDA rīki lieto standarta kartēšanu (mapping), lai ģenerētu platformspecifisko modeli (PSM) no PIM.
- Kods ir daļēji automātisks, daļēji manuāli uzrakstīts.

Kartēšana vairākās izvietošanas tehnoloģijās



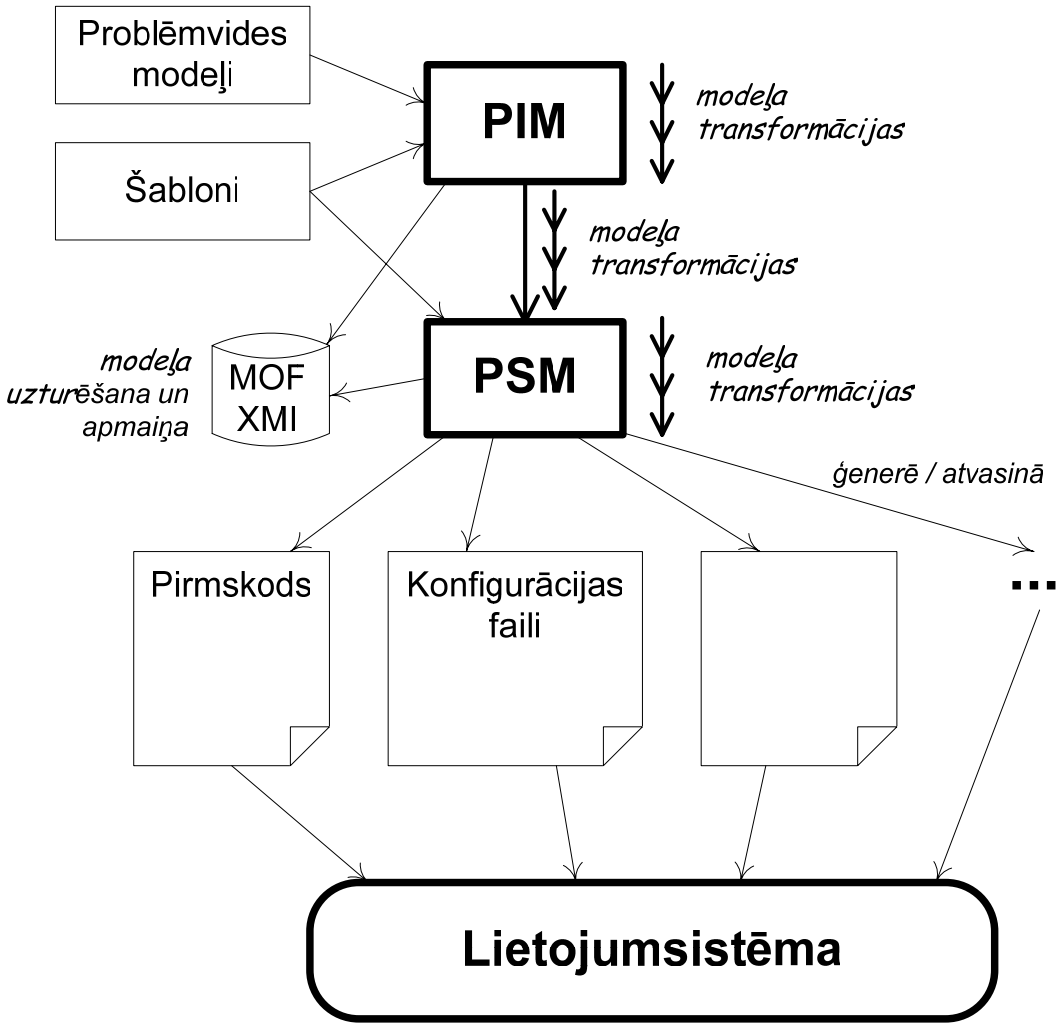
- MDA rīki lieto standarta kartēšanu (mapping), lai ģenerētu platformspecifisko modeli (PSM) no PIM.
- Kods ir daļēji automātisks, daļēji manuāli uzrakstīts.

Vairāku "implementāciju" ģenerēšana

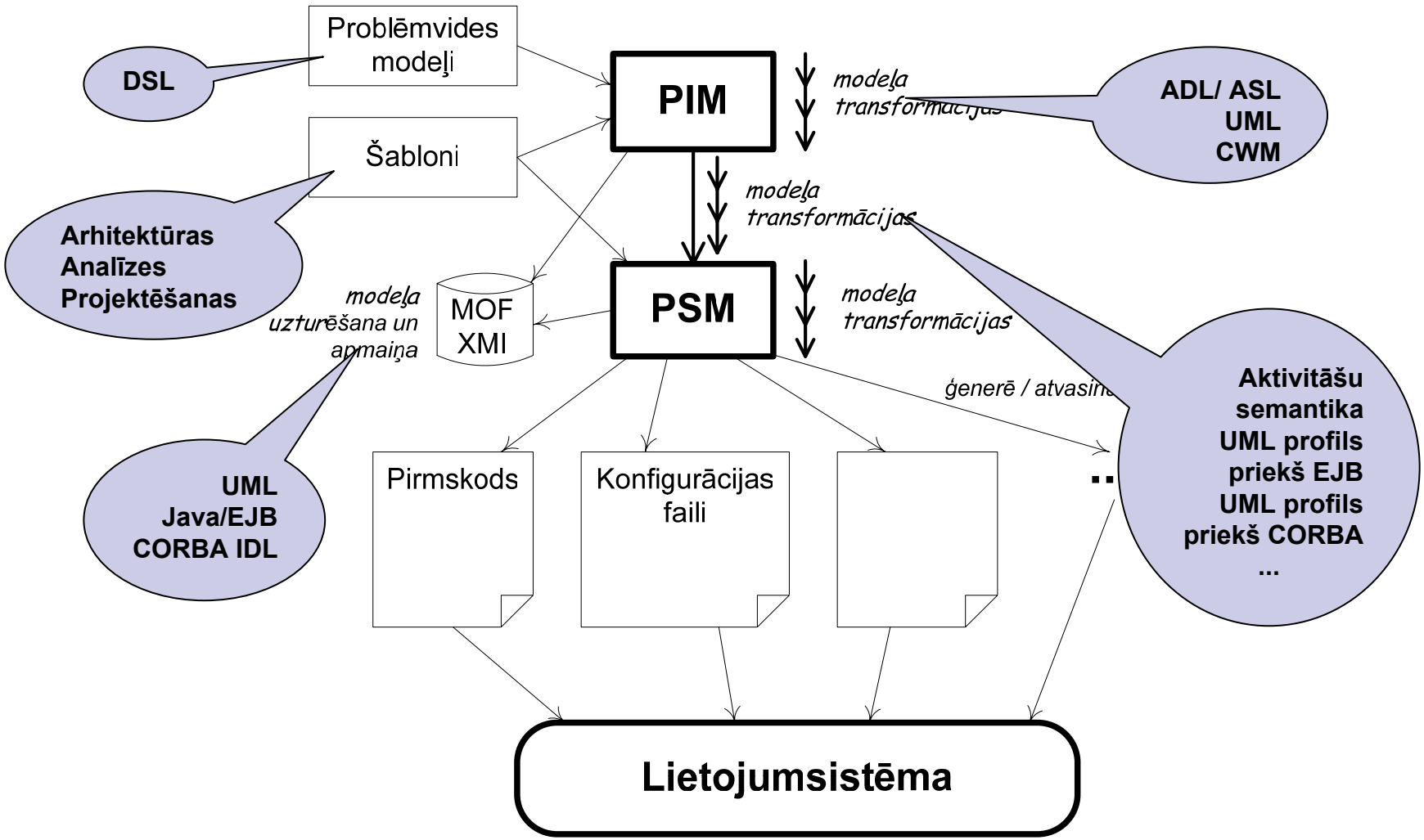


- MDA rīks ģenerē visu vai lielāko daļu no implementēšanas koda turpmākai izvietošanai strādājošā sistēmā

MDA realizācijas līdzekļi



MDA realizācijas līdzekļi



CASE rīki

- CASE – Computer Aided Software Engineering
- 80.gadu sākuma strukturētas metodoloģijas noveda līdz CASE rīku parādīšanos 80.gadu beigās
- CASE rīkiem bija deifnēti vairāki mērķi:
  - Modelēšana
  - Datu bāžu ģenerēšana
  - Koda ģenerēšana
  - Koda reinženierija
  - Programmēšana bez programmētāja

Problēmas ar CASE rīku lietošanu

- Grafisko modelēšanas valodu standartizācijas neesamība
- Kopējas starprogrammatūras (middleware) platformas neesamība
- Pārāk liela atstarpe starp modeļiem un implementēšanu
- Ģenerēta koda sarežģītība

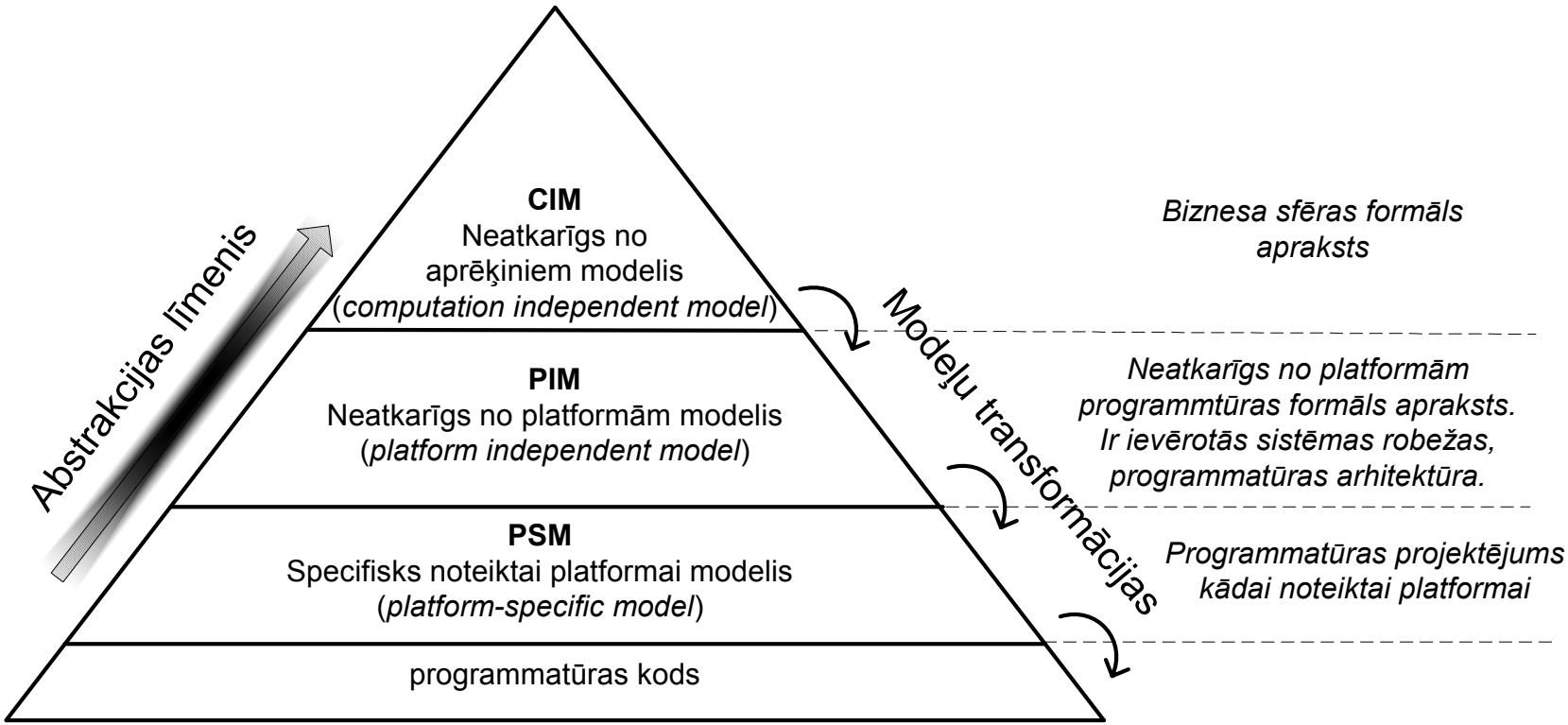
MDA vs CASE

- UML nodrošina modelēšanas pamatu
- Nobriedusi starpprogrammatūras platformu tehnoloģija
- Realizējamā pāreja no modeļiem uz implementēšanas platformu
- Nepārprotamā, noskaņojamā un paplašināmā transformācija uz implementēšanu

Abstrakcijas līmeņa pacelšana (2/2)

- Platformneatkarīgie modeļi
- Implementēšanas sarežģītības paslēpšana
- Problēmvides zināšanu atdalīšana no tehnoloģiju zināšanām

Abstrakcijas līmeņa pacelšana (2/2)

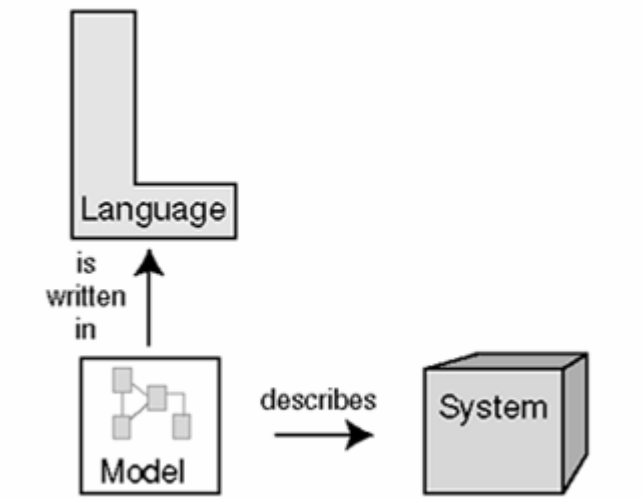


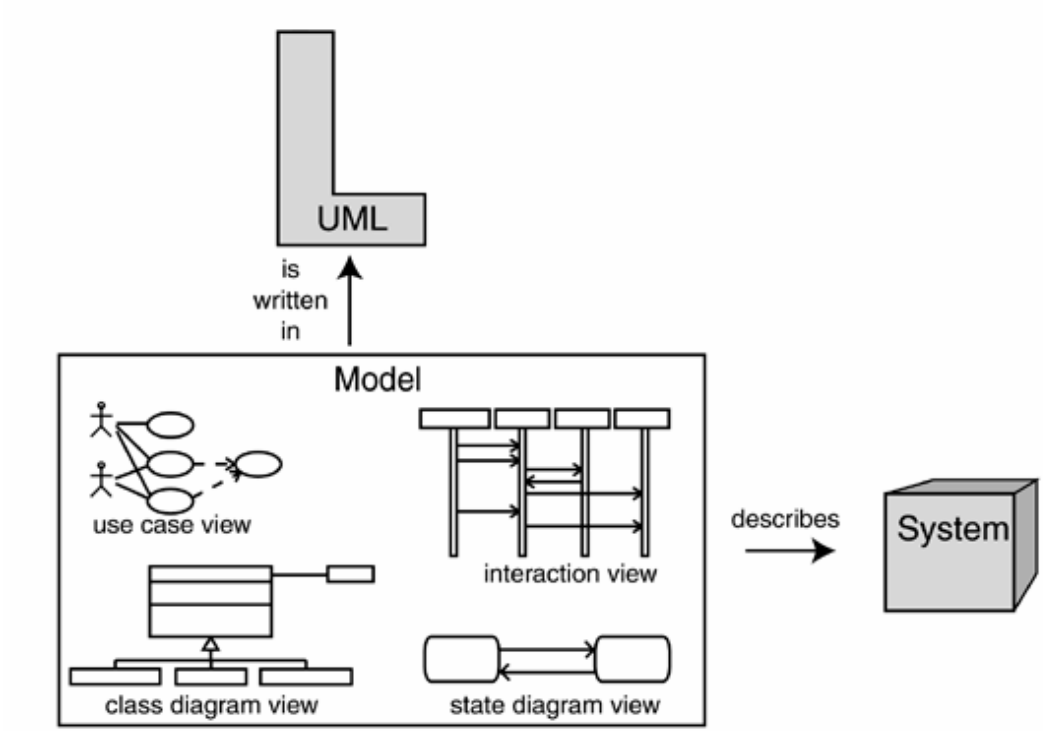
(c) Nikiforova, Kuzmina, Pavlova 2006

Metamodeļi

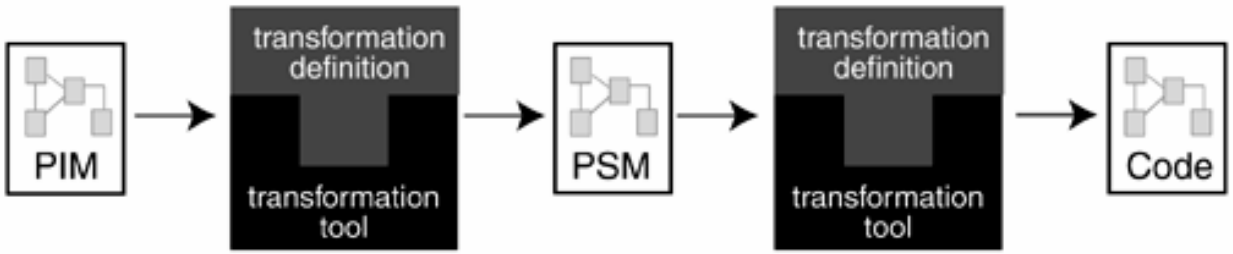
- Modelēšanas valodas modeļi
- Definē sintaksi un semantiku modelēšanas valodai
- Nodrošina sadarbības spēju modelēšanai un transformācijas rīkiem

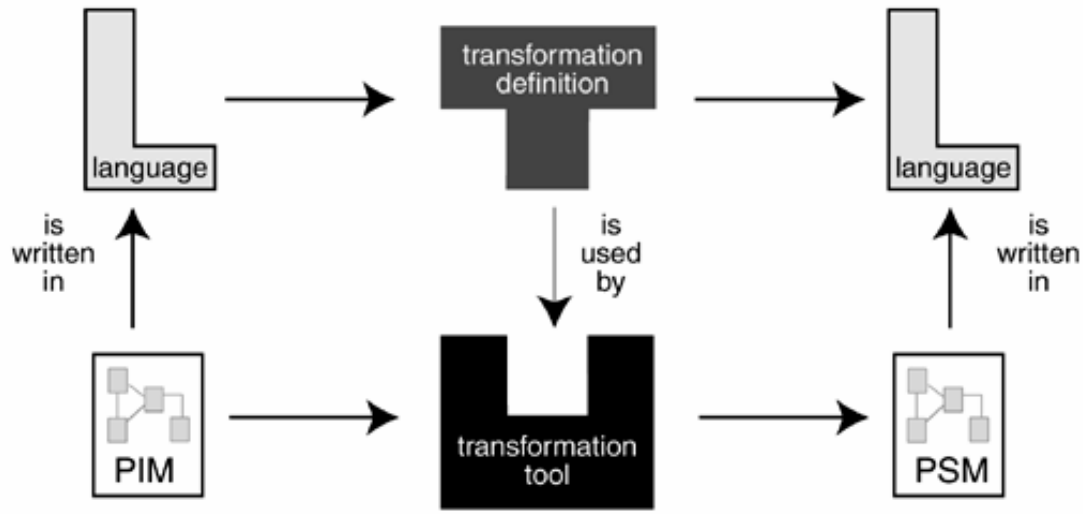
Modeļa modelēšanas valoda





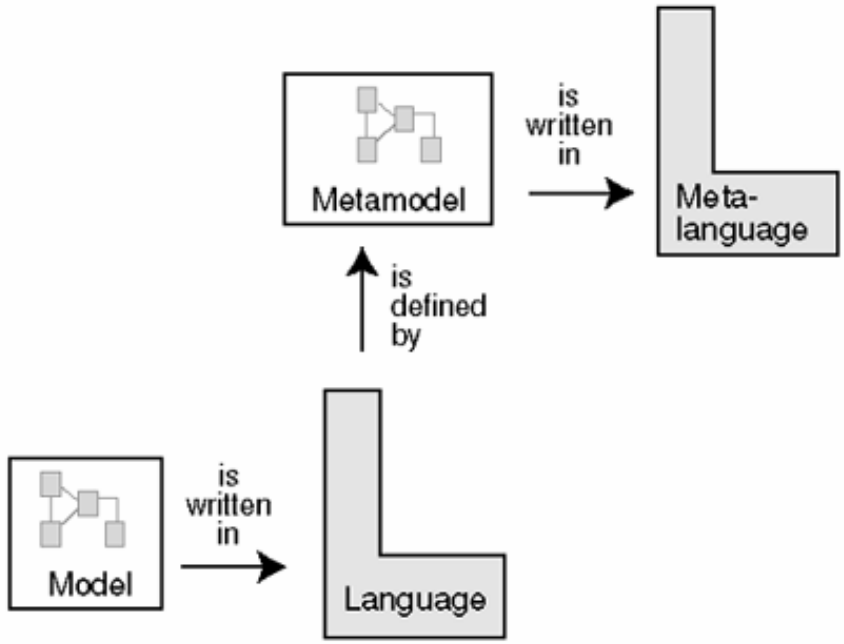
Modeļa transformācijas valoda





Modeļa modelis = metamodelis

- Modelēšanas valodas modeļi
- Definē sintaksi un semantiku modelēšanas valodai
- Nodrošina sadarbības spēju modelēšanai un transformācija rīkiem



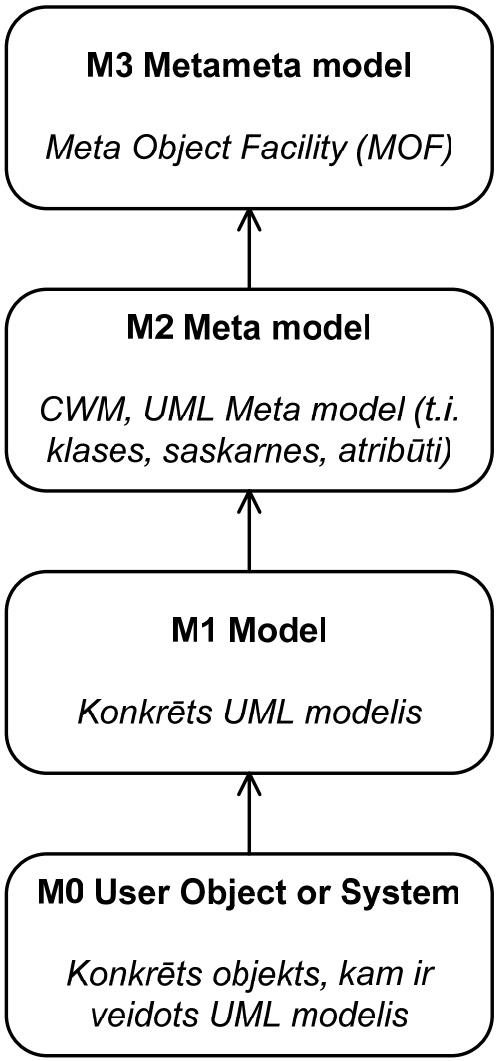
Metamodeļu nepieciešamība

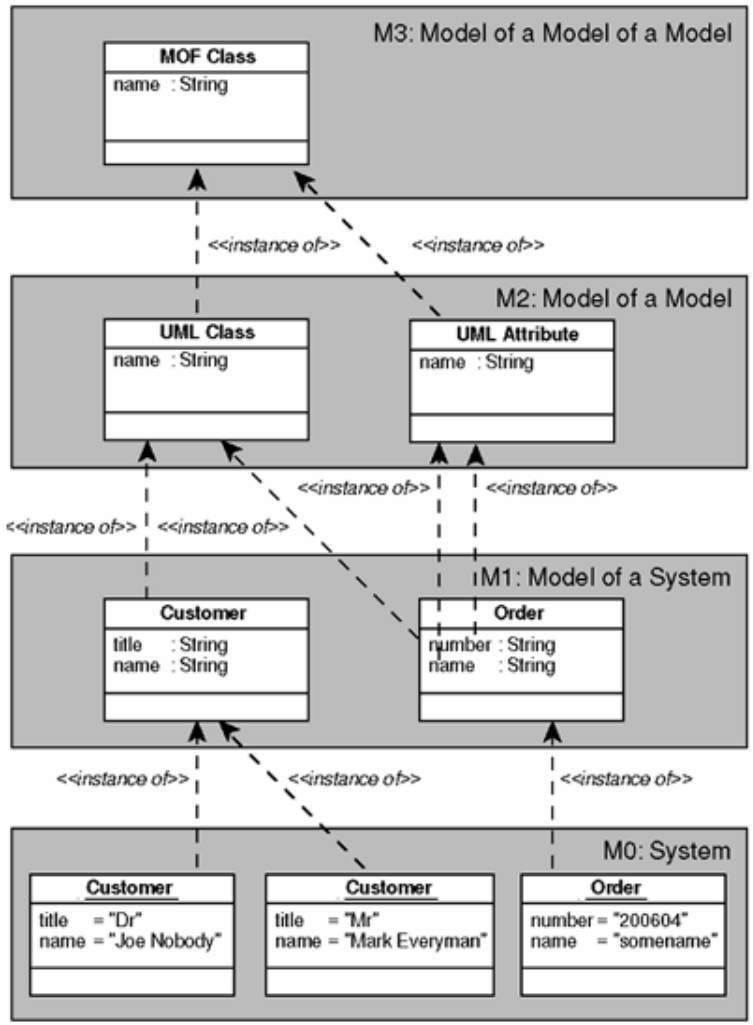
- Metamodeļi ļauj paplašināt un "noskaņot" MDA rīkus
- Metamodeļi ļauj paplašināt UML valodu
- Metamodeļi ļauj definēt pašu modelēšanas valodu
  - Domain-Specific Modelling Language
- Metamodeļi ļauj pielāgot MDA rīkus pašu modelēšanas valodas atbalstam

MOF

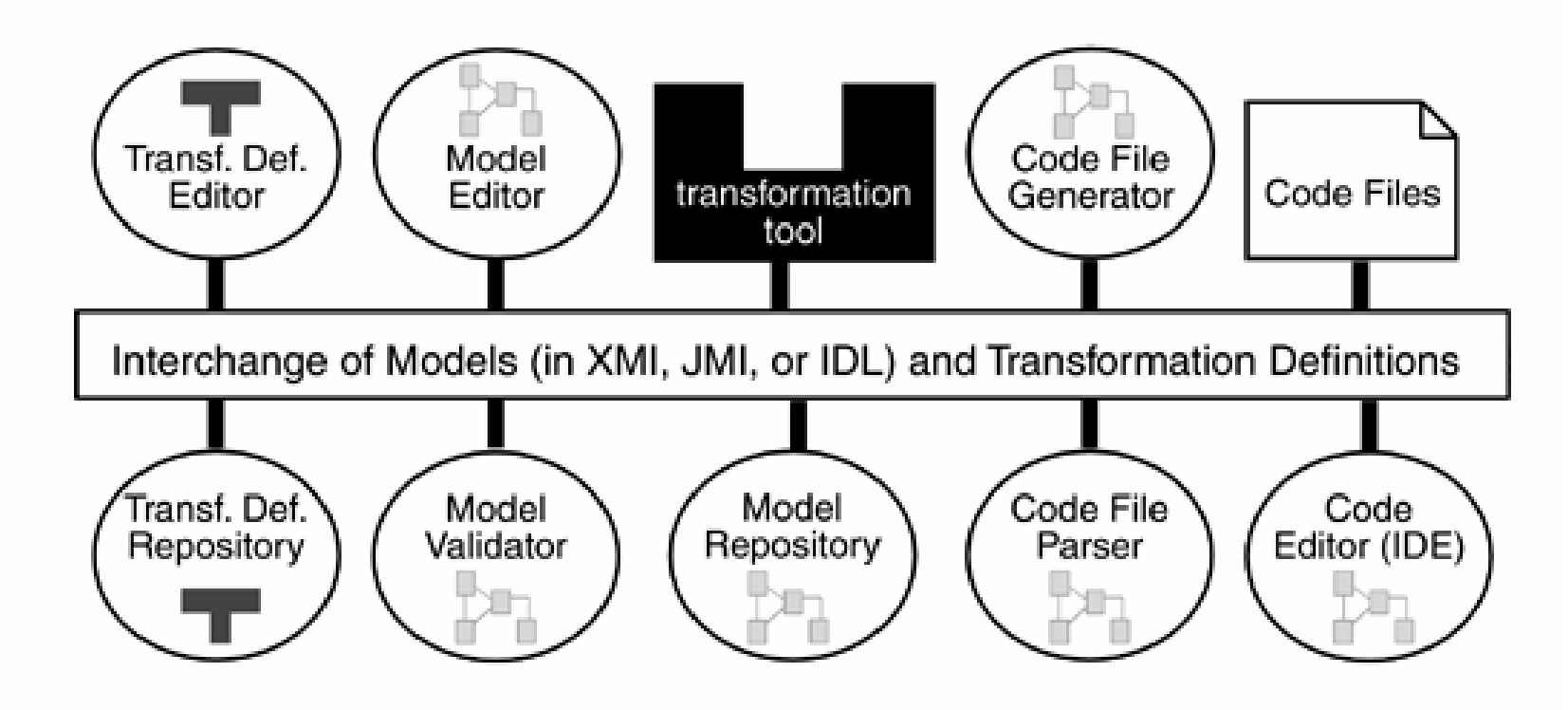
- MOF - Meta Object Facility – meta objekta spēja
  - Nodrošina meta-metamodelēšanu UML metamodeļiem
- Tas definē konceptu kopu (piem. pakete, clase, metode, atribūts), kas
  - Ļauj definēt un manipulēt ar modeļa metadatiem (dati par datiem)
  - Pats definēts UML notācijā

OMG 4-līmeņu arhitektūra





Funkcionēšana MDA izstrādes vidē



MDA komerciālie rīki

- **UMT** (UML Model Transformation Tool) ir rīks, kas atbalsta modeļu transformācijas un koda ģenerēšanu, kura pamatojas uz UML modeļiem XMI formā, importē XMI modeļus un pārveido tos starpformātā XML Light.
- **Ameos** (kompānijas Aonix /www.aonix.com/ products). Ameos apvieno UML 2.0. profilu atbalstu un uz MDA bāzētās Modeļu Transformācijas. Šis rīks nodrošina augsta līmeņa abstrakcijas modeļos un mērķneatkarīgo modelēšanu.
- **ARTiSAN Software Tools** (ARTiSAN kompānij) ir dažādi rīki, kuri realizē MDA pamatidejas, atbalsta UML 2.0 modeļu transformācijas.
- **Model-in-Action** (Mia Software) ir MDA rīku komplekts, atbalsta koda ģenerēšanu un no modeļa uz modeli transformācijas elastīgā vidē.
- **ModFact** ir MOF repozitorijs un līdzīga QVT mašīna no LIP6 (Parīze). Tā pamatā ir TRL valoda.
- **ArcStyler** – rīks importē UML modeļus no vairākiem citiem modelēšanas rīkiem, nodrošina J2EE un .NET komponentu ģenerēšanu, atbalsta web-servisu un web lietpjumsistēmu izstrādi, transformācijas balstās uz UML profiļiem un ir paplašināmās un noskaņojamās.

Komerciālās MDD vides

- **Enterprise Architect** (Sparx Systems) – ir Sparx Systems kompānijas komerciālais produkts, kas realizē modelēšanas un modeļu transformācijas idejas. Pašlaik ir pieejama Enterprise Architect bezmaksas Trial Version (30 dienām).
- **Rational Architect** (IBM Rational)– ir IBM Rational kompānijas komerciālais produkts, kas realizē modelēšanas un modeļu transformācijas idejas. Pašlaik ir pieejama Rational Architect bezmaksas Trial Version (30 dienām).

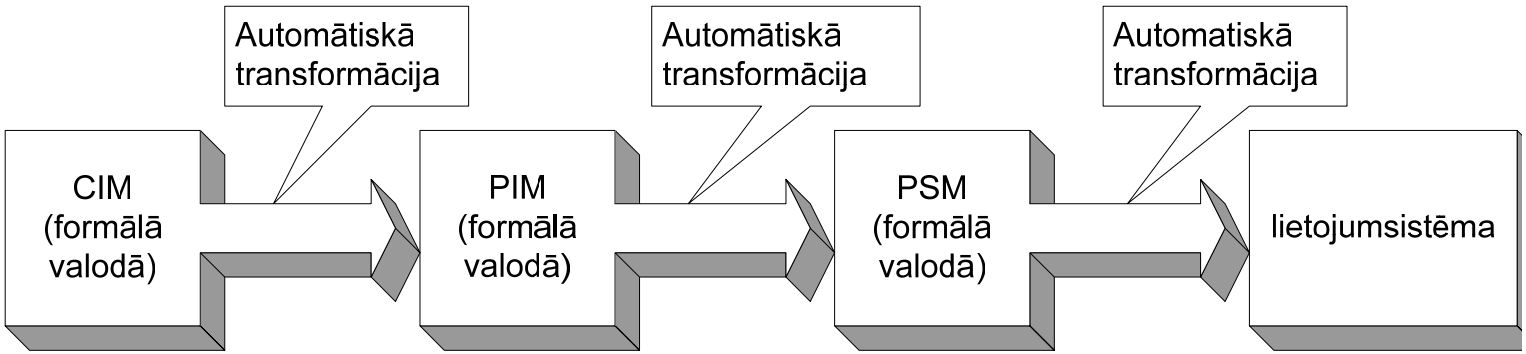
MDA open-source rīki

- **MTL** (Model Transformation Language) ir open-souce objektu un skatu orientēta valoda, kura ir paredzēta modeļu transformācijām. INRIA izveidoja rīku Umlaut NG, kurā šī valoda ir realizēta.
- **Compuware OptimalJ** vide ir realizēta JAVA valodā. Šajā vidē transformācijas likums ir realizēts kā metode ar ieejas parametriem un atgriežamo vērtību, t.i., Java objektu, kurš reprezentē mērķa modeļa elementa klasi. Gala mērķis ir automātiski uzģenerēt strādājošas lietojumprogrammas struktūru no OptimalJ biznesa modeļiem.
- **Jamda** (JAMDA Java Model Driven Architecture) ir Java vide lietojumprogrammu ģenerētāju veidošanai. Šāda tipa ģenerētāji veido Java kodu no biznesa domēna modeļiem. Atšķirībā no ģenerētājiem, kuri ražo vienu fiksētu arhitektūru, Jamda piedāvā struktūru un konstruēšanas blokus tādus, ka ir iespējams uzkonstruēt lietojumprogrammu ģenerētāju, kurš dara tieši to, kas ir nepieciešams tieši jūsu projektam.
- **OOMEGA** ir open-source apvienotas modelēšanas un metamodelēšanas rīks, valodu definēšanai un koda ģenerēšanai. Tas atbalsta db4objects, Hibernate, Versant, XML un SDF mdetamodeļu un modeļu glabāšanai. Koda ģenerēšanas šabloni ir uzrakstīti JSP valodā, kas atļauj sekot līdzi modeļu transformācijām.
- **AndroMDA** – ir open-source koda ģenerēšanas ietvars, atbalsta UML modeļus XMI formātā, ;generē komponentus valodās Java, HTML, PHP, platformās .NET u.c.

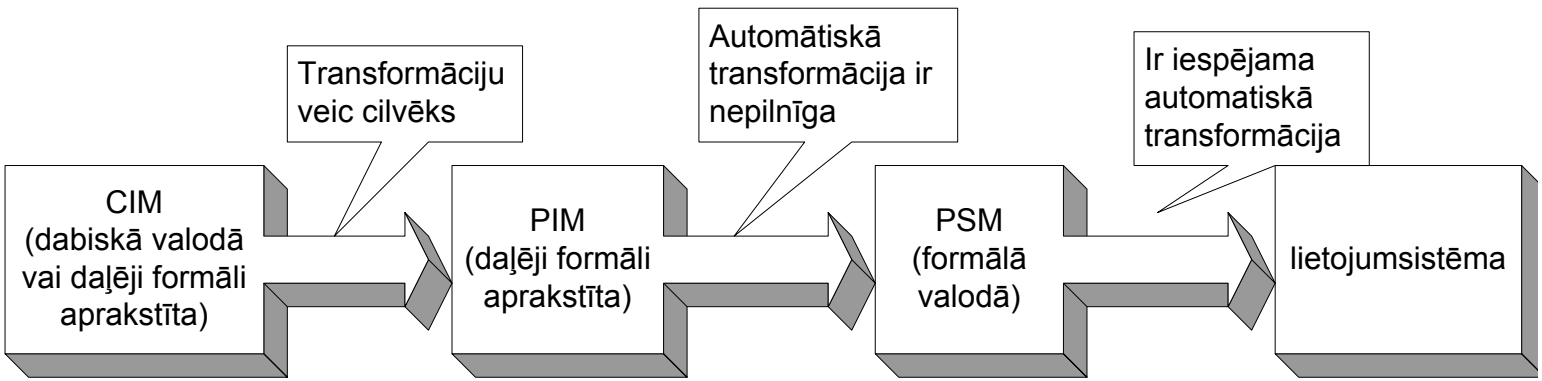
Eclipse EMF un saistītie rīki

- **MOFScript** ir rīks, kurš realizē transformācijas no modeļiem uz tekstu. Tā pamatā ir viens no OMG MOF Model to Text Transformation piedāvājumiem, kas ir izveidots kā Eclipse spraudnis.
- **ATL (ATLAS Transformation Language)** ir modeļu transformācijas valoda un rīku kopa, kurus izstrādāja ATLAS Grupa (INRIA un LINA). ATL ir realizēts uz Eclipse platformas. ATL apvienota vide nodrošina standarta izstrādes rīku kopumu, kuru mērķis ir vienkāršot izstrādē ar ATL transformācijām.
- **GMT (Generative Modeling Transformer)** ir Eclipse projekts, kurš nodrošina pētījumorientētu modeļu transformāciju tehnoloģijas Eclipse platformai. Vairāki rīki ir GMT daļas, proti, AMW (Model Viewing), Epsilon (Model Merging), MoDisco (Model Discovery) u.c.
- **MDWorkbench** ir teksta un modeļu transformāciju rīku komplekts, kas atbalsta kā ieeju jebkuru metamodeļa tipu. Rīks ir izstrādāts uz Eclipse un EMF pamata un ir pieejama arī bezmaksas versija (ir nepieciešama reģistrācija).
- **MDWorkbench** ir teksta un modeļu transformāciju (komerciāls) rīku komplekts, kas atbalsta kā ieeju jebkuru metamodeļa tipu. Rīks ir izstrādāts uz Eclipse un EMF pamata un ir pieejama arī bezmaksas versija (ir nepieciešama reģistrācija).
- **Kermeta** ir INRIA Triskell komandas izveidots rīks uz Eclipse pamata. Tā ir objektorientētas metamodelēšanas vide, kura ir paredzēta mdetamodelēšanas inženierijas aktivitātēm. Kermeta ir izveidota kā Eclipse EMF paplašinājums.

MDA transformācijas virkne



Šodien iespējamā MDA realizācija:

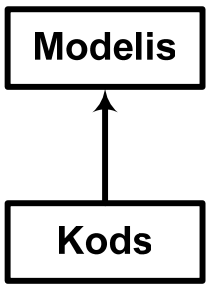


Kodēšana pret modelēšanu

Tikai kods

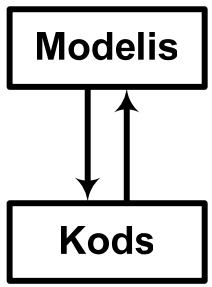
Kas ir modelis?

Koda vizualizācija



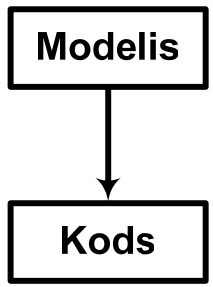
Kods ir modelis?

Reinženierija



Kods un modelis līdzeksistē?

Ar modeli vadīta



Modelis ir kods?

Tikai modelis

Pagaidam tikai modelē...

Modelēšanas gatavības (briduma) līmeņi

- Modeling Maturity Levels (pēc analoģijas ar CMM – Capability Maturity Model līmeņiem)
  - MML 0: Bez specifikācijas
  - MML 1: Teksta veidā
  - MML 2: Teksts ar diagrammām
  - MML 3: Modeļi ar tekstu
  - MML 4:Precīzie modeļi
  - MML 5:Tikai modeļi

MML 0: Bez specifikācijas

- Programmatūras sistēma tiek specificēta izstrādātāju galvās

MML 1: Teksta veidā

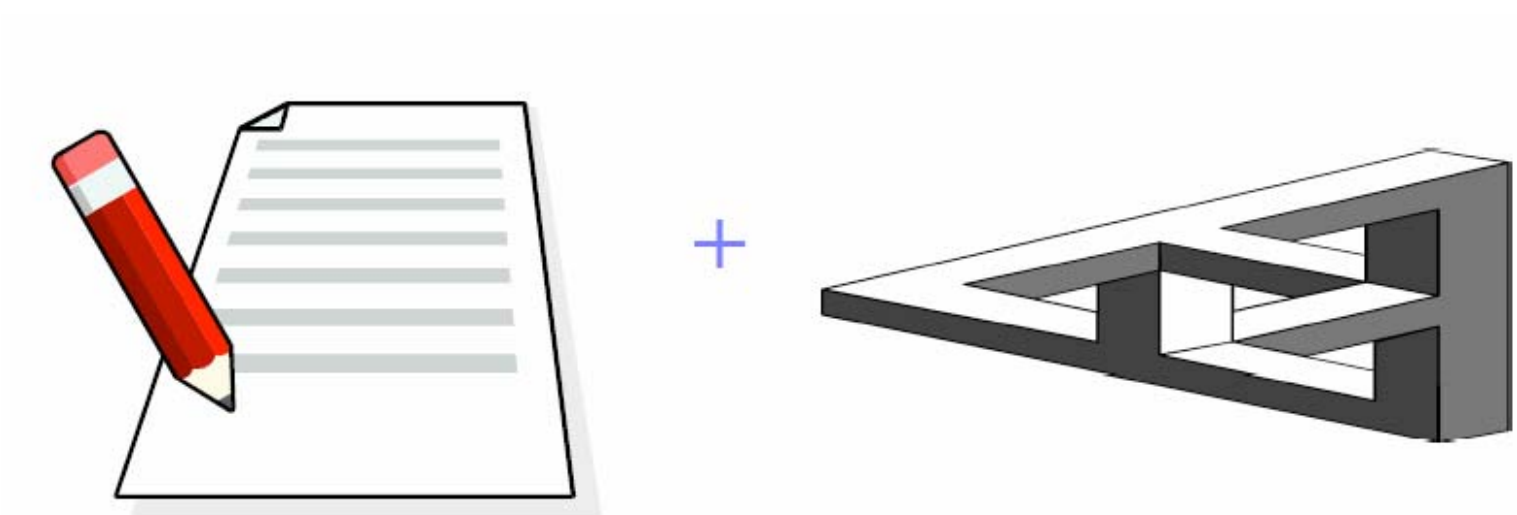
- Programmatūras specifikācija ir pierakstīta vienā vai vairākās dabīgajās valodās

MML 2: Teksts ar diagrammām

- Programmatūras specifikācija ir pierakstīta vienā vai vairākās dabīgajās valodās

  +

- Dažas augsta līmeņa diagrammas, kas skaidro kopējo sistēmas arhitektūru

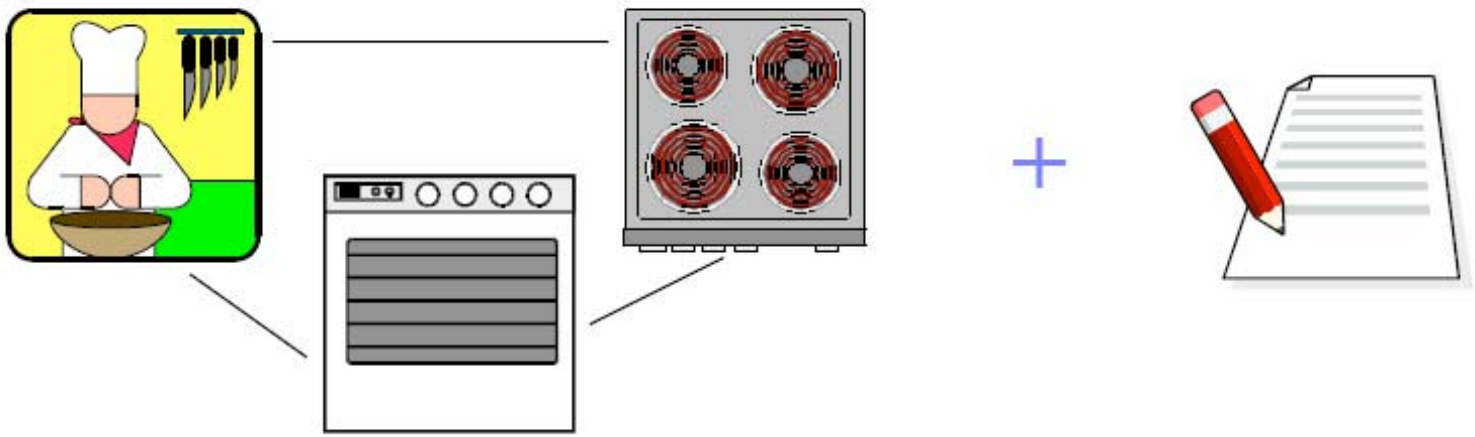


MML 3: Modeļi ar tekstu

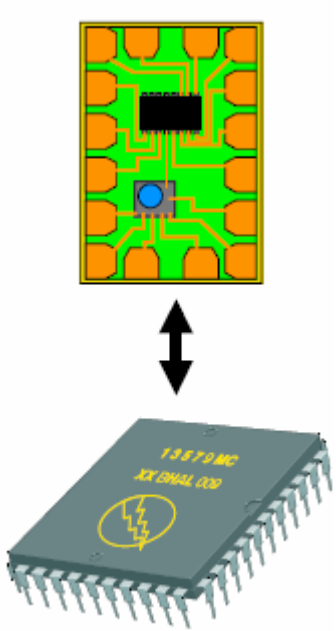
- Programmatūras specifikācija ir pierakstīta viena vai vairāku modeļu veidā

  +

- Papildus dabīgā valodā tiek lietota, lai izskaidrotu modeļu pamatus un motivāciju



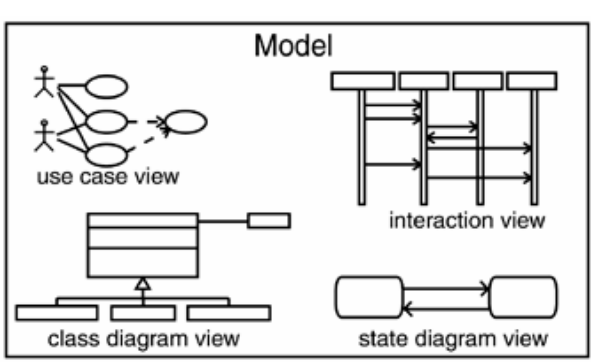
MML 4:Precīzie modeļi

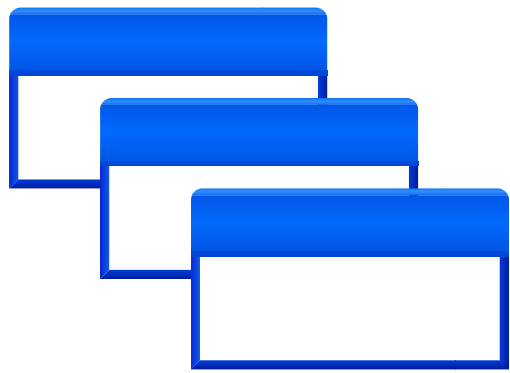


- Programmatūras specifikācija ir pierakstīta viena vai vairāku modeļu veidā
  - Papildus dabīgā valodā tiek lietota, lai izskaidrotu modeļu pamatus un motivāciju
- Modeļi ir pietiekami precīzi, lai no tiem būtu nodrošināta tieša pārēja pie programmas koda
- Tas ir modeļvadāmās izstrādes (MDD – Model Driven Development) pamats

MML 5:Tikai modeļi

- Modeļi ir pietiekami precīzi, lai no tiem būtu nodrošināta pilnā koda ģenerēšana
- Kods ir neredzams
- Modelēšanas valodas = augstā līmeņa programmēšanas valodas (5GL?)
- Tā var būt ir nākotnes perspektīva…





Modelis **Kods** **Strādājoša lietojumsistēma**