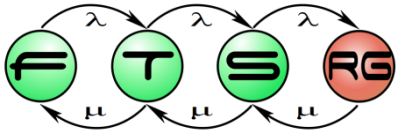
Kódgenerálás



Áttekintés

- Metamodellezés
  - Absztrakt szintaxis
    - EMF
  - Konkrét szintaxis
    - GEF, GMF, IMP, Xtext
  - Jólformáltsági szabályok
    - Részben
  - Szemantika
  - Leképezés más nyelvre
    - Ez lesz most

Kódgenerálás

MDD projektek szerkezete

Modell

Generált kód

MDD projektek szerkezete

Modell

Generált kód

Kézzel írt kód

Célok

- Fejlesztési idő rövidítése
- Modell/Követelmény/Terv alapján
  - Dokumentáció
  - Forráskód
  - Konfigurációs állományok
  - Kommunikációs üzenetek
  - Objektumsorosítás
  - …

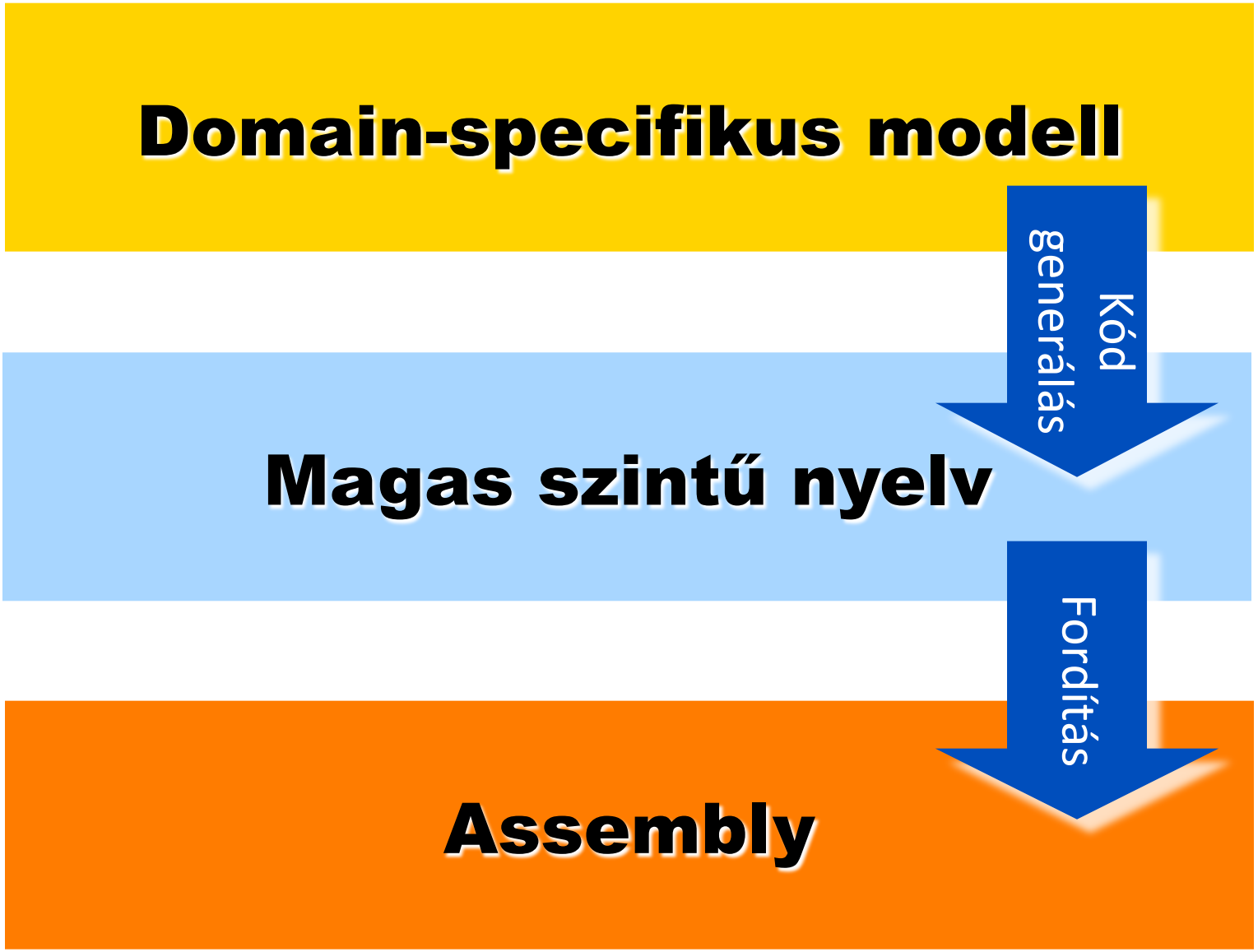
Szöveges állományok előállítása

- Magas szintű modell megvalósítása
  - Implementációs platform fölött
- Választási pontok attribútumoknál (kompromisszum)
  - Kompatibilitás
  - Teljesítmény
  - Karbantarthatóság
  - Újrahasznosíthatóság

Fordítóprogramok

- Absztrakciós szintek áthidalása
  - Pl. C és assembly
- Tervezési minták felhasználása
  - Pl. függvényhívások C-ben
- Sok hasonlóság – nem szigorú különbségtétel
  - Pl. C++ template-ek, automatikusan generált konstruktor/destruktor

Példa: MDD kód generálás



Megközelítések

Megközelítések

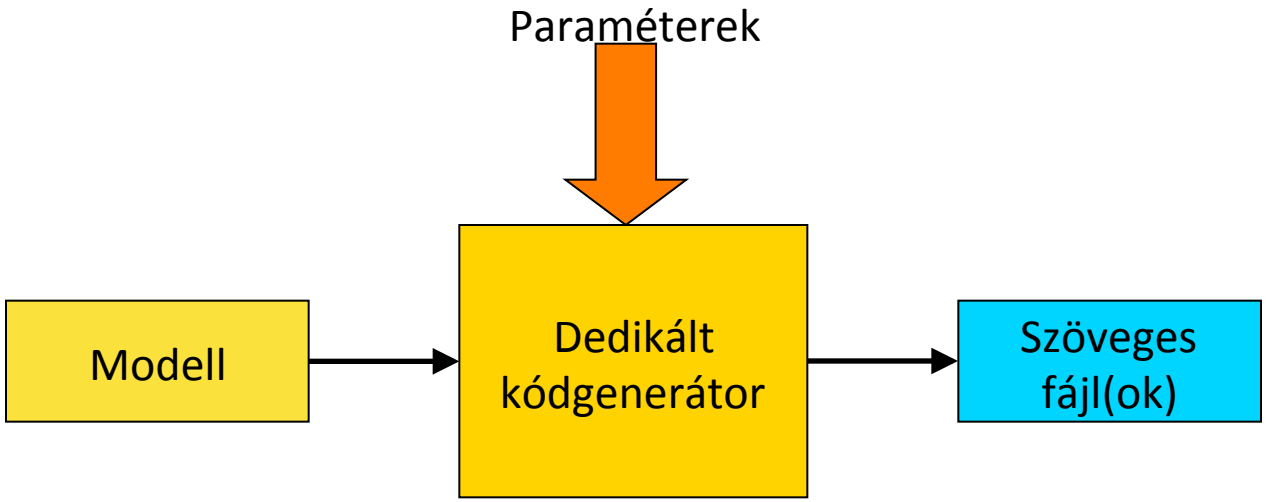
- Dedikált
  - Speciális, ad-hoc
  - Általános kód generátor alapú
- Sablon alapú

Speciális, ad-hoc

```
sourceFile.write("   temp = ((AIDA_PARTITION_TYPE*) selfModule.partitions.elements);\n" )
i = 0
for partition in partitions:
 numPorts = getNumberOfAllCommPorts_Partition(currModuleComm, interPartitionComm, partition.partitionName)
 sourceFile.write("   temp[" + str(i) + "].partition_id = " + str(partition.partitionID) + ";\n" )
 sourceFile.write("   strcpy( &temp[" + str(i) + "].partition_name[0], \"" + str(partition.partitionName) + "\");\n")
 sourceFile.write("   temp[" + str(i) + "].ports.type = CONST_AIDA_PORTS_TYPE;\n")
 sourceFile.write("   temp[" + str(i) + "].ports.elements = &mem_ports_" + str(partition.partitionName) + "[0];\n")
 sourceFile.write("   temp[" + str(i) + "].ports.numOfElements = " + str(numPorts) + ";\n")
 sourceFile.write("\n")
 i = i + 1
## end for
sourceFile.write("\n")
```

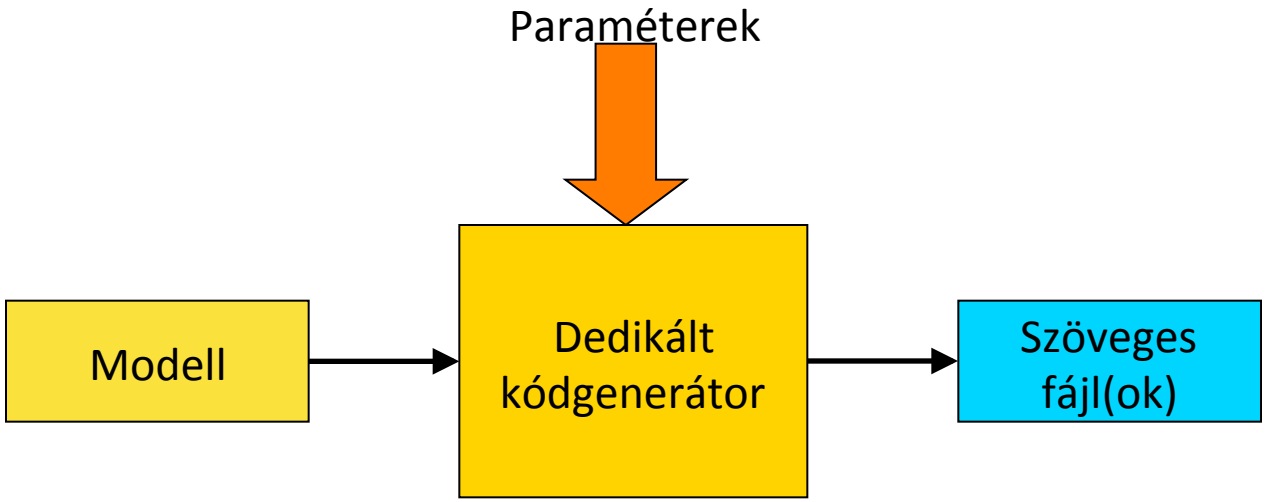
- Problématerületre optimalizálva
  - Legjobb teljesítmény
  - „Quick-and-dirty”
  - Hosszas fejlesztés, karbantarthatóság problémás
  - Újrahasznosíthatóság 0
  - Speciális területekhez alkalmas
    - Minimális változtatások életciklus során (biztonságkritikus beágyazott rendszerek, védelem)
    - Hitelesíthetőség
  - Példa
    - ARINC653 Multistatic configuration generator (Python script)

Dedikált kódgenerátor



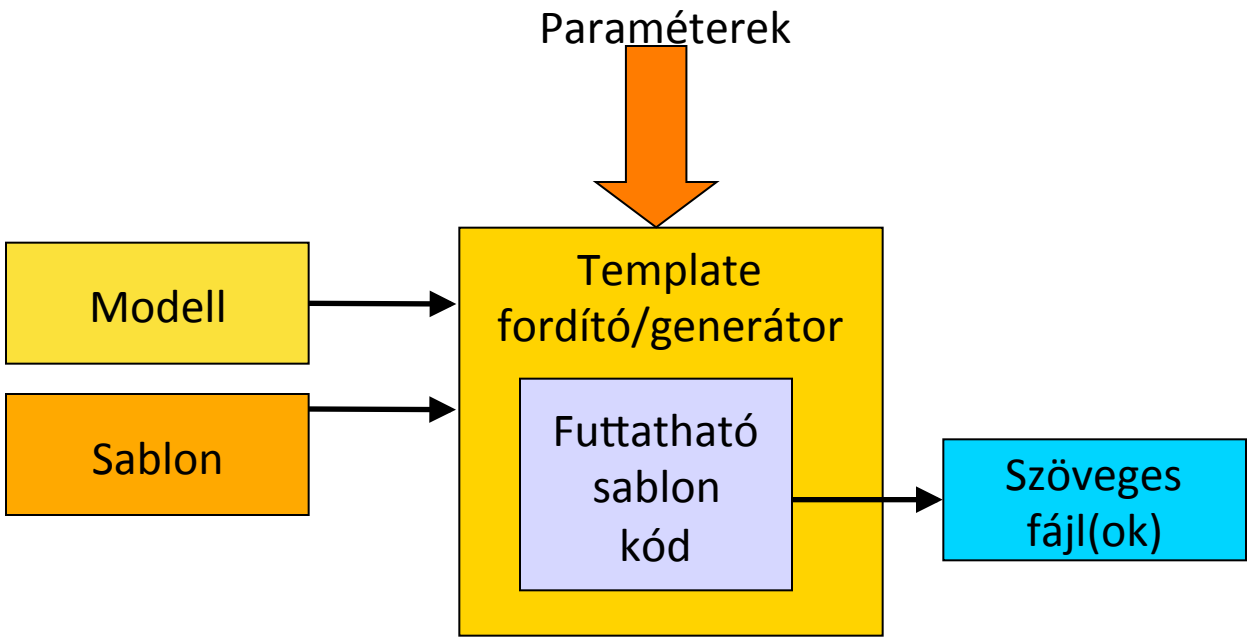
- Keretrendszer alapú
  - Gyorsabb fejlesztés
  - Gyengébb teljesítmény, jobb újrahasznosíthatóság
  - Beágyazott rendszerek, közepes mennyiségű változtatás

Dedikált kódgenerátor

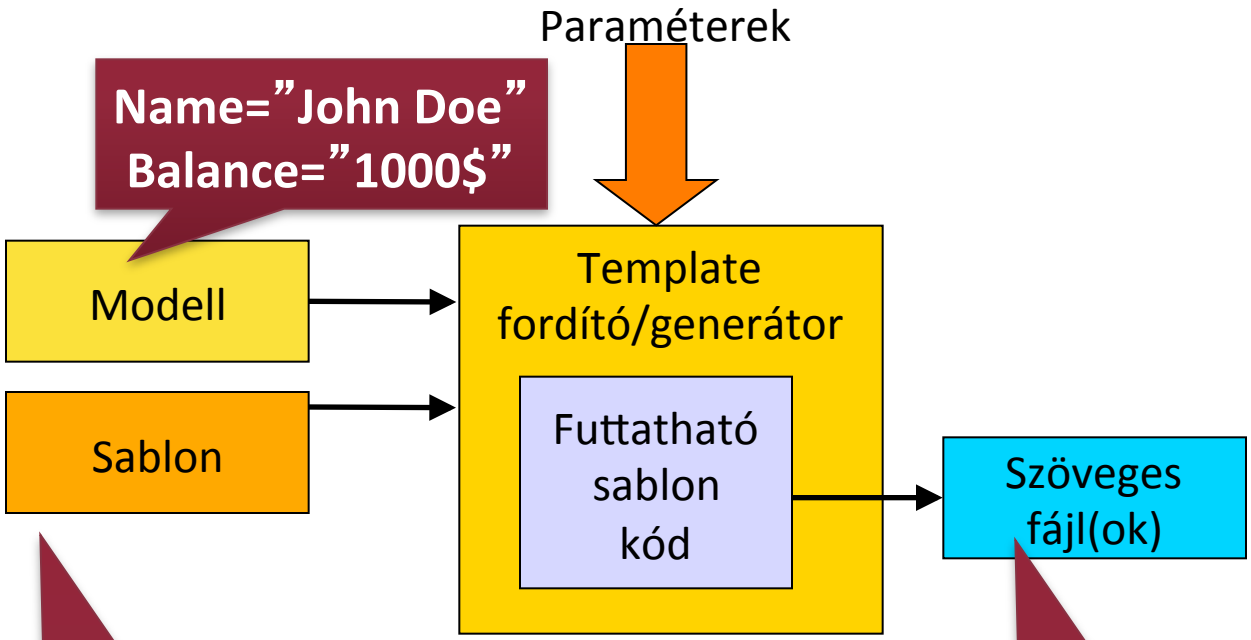


- Példák:
  - IBM Rational Software Architect
  - VASP (DO-178B Level A) Display graphics in avionics
  - Mathworks
  - Matlab Simulink
  - Esterel Scade suite

Sablon alapú



Sablon alapú



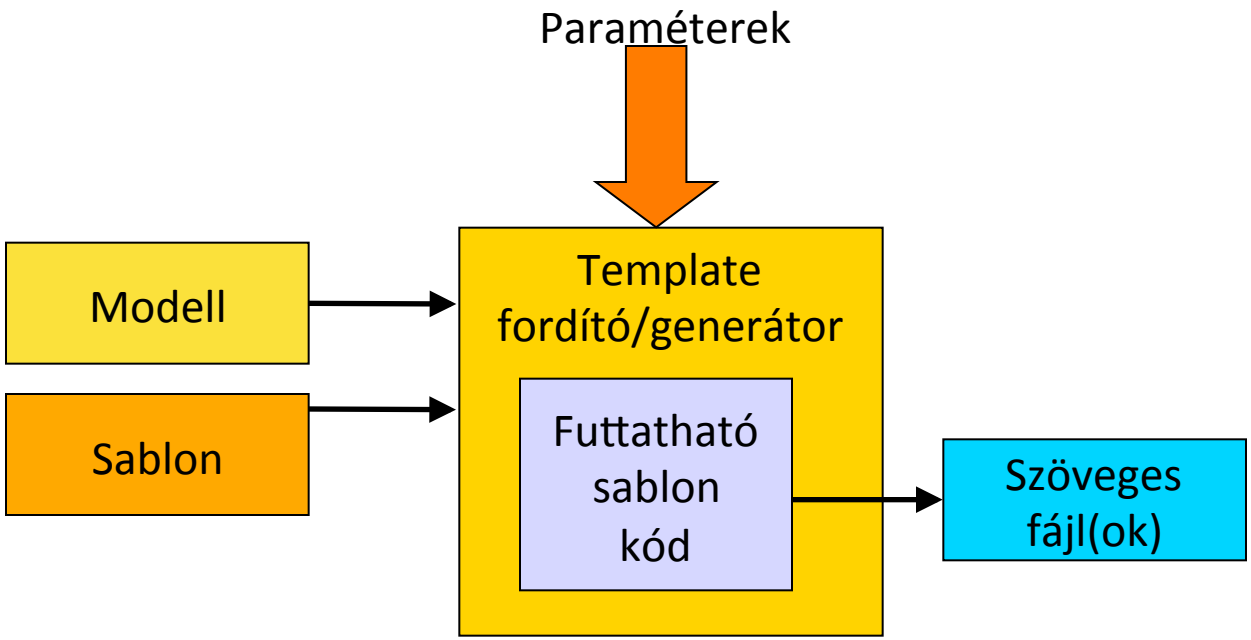
Dear [Name].

I would like to inform you that your current balance is [Balance]

Dear John Doe.

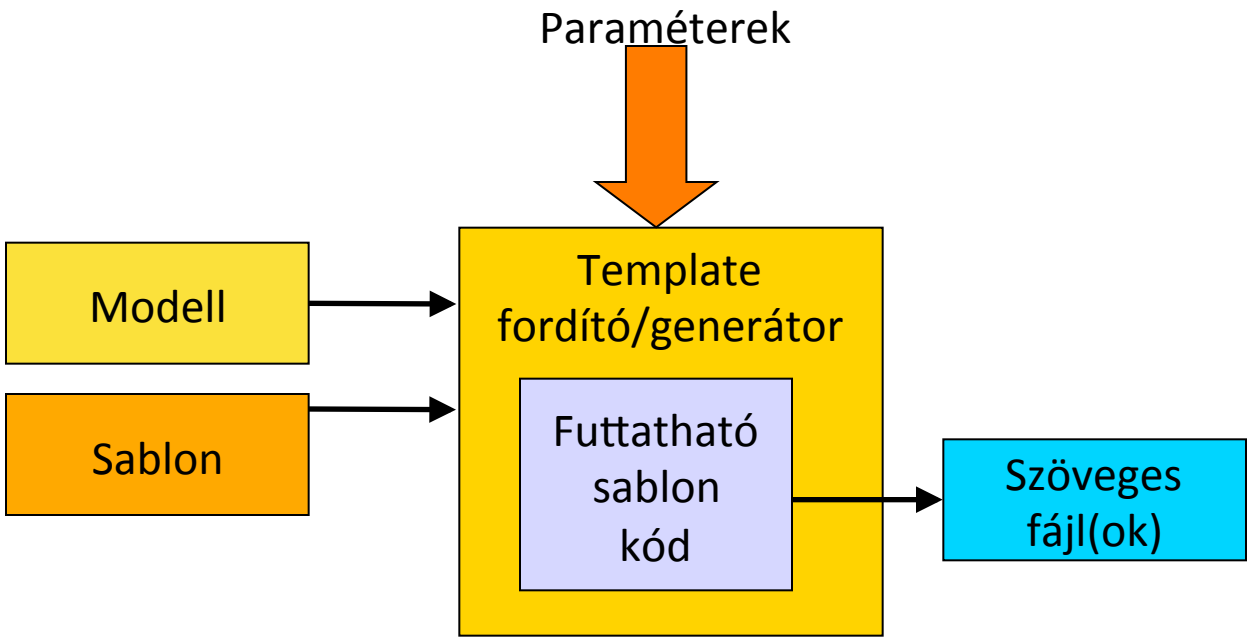
I would like to inform you that your current balance is 1000$

Sablon alapú



- Leggyorsabb fejlesztés
- Leggyengébb teljesítmény, legjobb újrahasznosíthatóság
- Gyorsan változó környezet (pl. webes technológiák)
- Összetett változtatások az életciklus során
  - Modell és sablon külön-külön módosítható

Sablon alapú

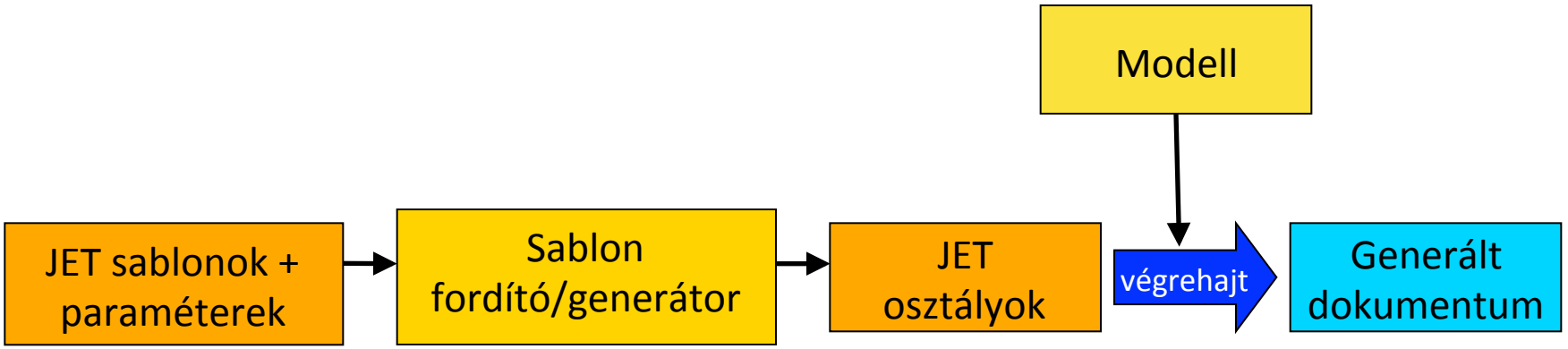


- Példák:
  - JET (for EMF models)
  - Velocity (/JSP)
  - OpenArchitectureWare/ XPand (MDD approach)
  - AutoFilter (Kalman filters)
  - Smarty (php)

Kódgenerátor technológiák EMF modellekhez

Java Emitter Templates (JET),
Acceleo és Xpand

Java Emitter Templates



- Java Emitting Templates (JET)
  - JSP-szerű template nyelv Java vezérlési szerkezettel
  - **Java kódra fordítva**
  - Nyílt kimeneti formátum (szöveg)
  - Paraméterek: Java objektumok
  - EMF része
    - EMF kódgenerálás JET alapon történik

JET példa

```
<%@ jet package="hello"
imports="java.util.*" class="XMLDemoTemplate"
%>
<% List elementList = (List) argument; %>

<?xml version="1.0" encoding="UTF-8"?>
<demo>

<% for (Iterator i = elementList.iterator();
i.hasNext(); ) { %>
<element><%=i.next().toString()%></element>

<% } %>

</demo>
```

JET példa

```
<%@ jet package="hello"
imports="java.util.*" class="XMLDemoTemplate"
%>
<% List elementList = (List) argument; %>

<?xml version="1.0" encoding="UTF-8"?>
<demo>

<% for (Iterator i = elementList.iterator();
i.hasNext(); ) { %>
<element><%=i.next().toString()%></element>

<% } %>

</demo>
```

Jet Header

JET példa

```
<%@ jet package="hello"
imports="java.util.*" class="XMLDemoTemplate"
%>
<% List elementList = (List) argument; %>

<?xml version="1.0" encoding="UTF-8"?>
<demo>

<% for (Iterator i = elementList.iterator();
i.hasNext(); ) { %>
<element><%=i.next().toString()%></element>

<% } %>

</demo>
```

Package of representing class

JET példa

```
<%@ jet package="hello"
imports="java.util.*" class="XMLDemoTemplate"
%>
<% List elementList = (List) argument; %>

<?xml version="1.0" encoding="UTF-8"?>
<demo>

<% for (Iterator i = elementL
i.hasNext(); ) { %>
<element><%=i.next().toString()%></element>

<% } %>

</demo>
```

Name of the Class representing the Template

JET példa

```
<%@ jet package="hello"
imports="java.util.*" class="XMLDemoTemplate"
%>
<% List elementList = (List) argument; %>

<?xml version="1.0" encoding="UTF-8"?>
<demo>

<% for (Iterator i = elementList.iterator();
i.hasNext(); ) { %>
<element><%=i.next().toString()%></element>

<% } %>

</demo>
```

Packages to import

JET példa

```
<%@ jet package="hello"
imports="java.util.*" class="XMLDemoTemplate"
%>
<% List elementList = (List) argument; %>

<?x                    1.0" encoding="UTF-8"?>
<demo>

<% for (Iterator i = elementList.iterator();
i.hasNext(); ) { %>
<element><%=i.next().toString()%></element>

<% } %>

</demo>
```

Start of code section

JET példa

```
<%@ jet package="hello"
imports="java.util.*" class="XMLDemoTemplate"
%>
<% List elementList = (List) argument; %>

<?xml version="1.0" encod"?>
<demo>

<% for (Iterator i = elementList.iterator();
i.hasNext(); ) { %>
<element><%=i.next().toString()%></element>

<% } %>

</demo>
```

Input parameter

JET példa

```
<%@ jet package="hello"
imports="java.util.*" class="XMLDemoTemplate"
%>
<% List elementList = (List) argument; %>

<?xml version="1.0" encoding="UTF-8"?>
<demo>

<% for (Iterator i = elementList.iterato;
i.hasNext(); ) { %>
<element><%=i.next().toString()%></element>

<% } %>

</demo>
```

End of code section

JET példa

```
<%@ jet package="hello"
imports="java.util.*" class="XMLDemoTemplate"
%>
<% List elementList = (List) argument; %>

<?xml version="1.0" encoding="UTF-8"?>
<demo>

<% for (Iterator i = elementList.iterator();
i.hasNext(); ) { %>
<element><%=i.next()
```

Start of target document

```
</element>

<% } %>

</demo>
```

JET példa

```
<%@ jet package="hello"
imports="java.util.*" class="XMLDemoTemplate"
%>
<% List elementList = (List) argument; %>

<?xml version="1.0" encoding="UTF-8"?>
<demo>

<% for (Iterator i = elementList.iterator();
i.hasNext(); ) { %>
<element><%=i.next().toString()%></element>

<% } %>

</demo>
```

Loop with the input parameter

JET példa

```
<%@ jet package="hello"
imports="java.util.*" class="XMLDemoTemplate"
%>
<% List elementList = (List) argument; %>

<?xml version="1.0" encoding="UTF-8"?>
<demo>

<% for (Iterator i = elementList.iterator();
i.hasNext(); ) { %>
<element><%=i.next().toString()%></element>

<% } %>

</demo>
```

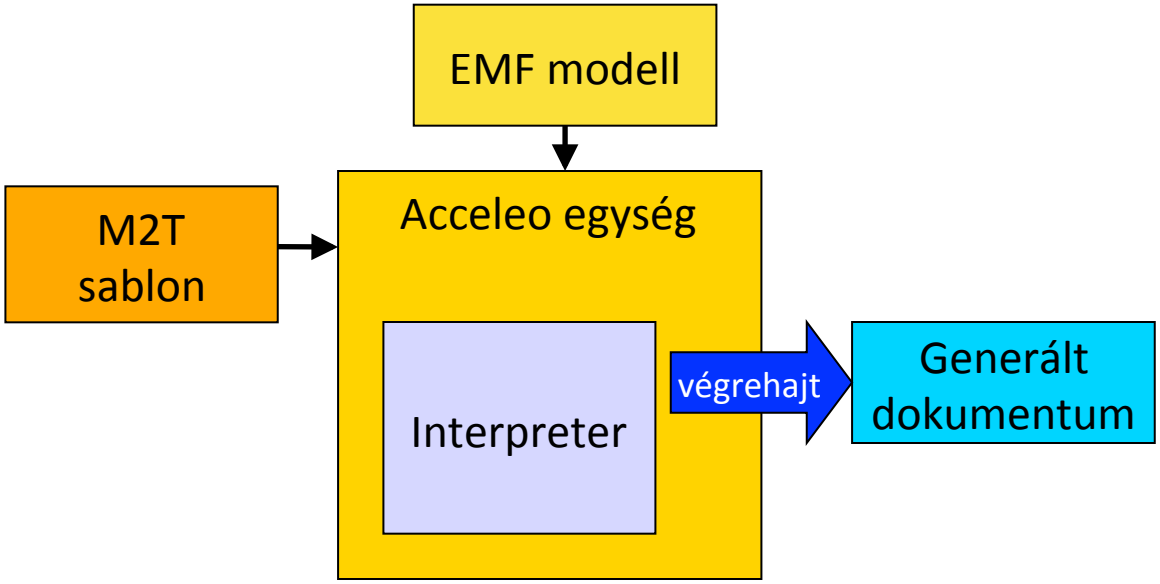
Returns value of the argument

JET példa

```
<%@ jet package="hello"
imports="java.util.*" class="XMLDemoTemplate"
%>
<% List elementList = (List) argument; %>

<?xml version="1.0" encoding="UTF-8"?>
<demo>

<% for (Iterator i = elementList.iterator();
i.hasNext(); ) { %>
<element><%=i.next().toString()%></element>

<% } %>

</demo>
```

Loop body

Acceleo



- OMG M2T implementáció
  - EMF modellek
  - OCL feltételek kiértékelése (EMF-OCL függőség)
- Interpretált sablon
- Egyszerű vezérlés
  - if-else, for és let

Acceleo példa

```
[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/emf/2002/Ecore')/]

[template public generate(e : EClass)]

[comment @main /]
[file (e.name, false, 'UTF-8')]
<?xml version="1.0" encoding="UTF-8"?>
  <demo name=”[e.name/]">
    [for (it : EAttribute | e.eAttributes)]
       <attr name="[it.name/]" />
       [/for]
  </demo>
[/file]

[/template]
```

Acceleo példa

Import EMF metamodel

```
[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/emf/2002/Ecore')/]

[template public generate(e : EClass)]

[comment @main /]
[file (e.name, false, 'UTF-8')]
<?xml version="1.0" encoding="UTF-8"?>
  <demo name=”[e.name/]">
    [for (it : EAttribute | e.eAttributes)]
      <attr name="[it.name/]" />
      [/for]
  </demo>
[/file]

[/template]
```

Acceleo példa

```
[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/emf/2002/Ecore')/]

[template public generate(e : EClass)]

[comment @main /]
[file (e.name, false, 'UTF-8')]
<?xml version="1.0" encoding="UTF-8"?>
  <demo name=”[e.name/]">
    [for (it : EAttribute | e.eAttributes)]
      <attr name="[it.name/]" />
      [/for]
  </demo>
[/file]

[/template]
```

Template

Acceleo példa

```
[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/emf/2002/Ecore')/]

[template public generate(e : EClass)]

[comment @main /]
[file (e.name, false, 'UTF-8')]
<?xml version="1.0" encoding="UTF-8"?>
  <demo name=”[e.name/]">
    [for (it : EAttribute | e.eAttributes)]
      <attr name="[it.name/]" />
      [/for]
  </demo>
[/file]

[/template]
```

Entry point

Acceleo példa

```
[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/emf/2002/Ecore')/]

[template public generate(e : EClass)]

[comment @main /]
[file (e.name, false, 'UTF-8')]
<?xml version="1.0" encoding="UTF-8"?>
  <demo name=”[e.name/]">
    [for (it : EAttribute | e.eAttributes)]
      <attr name="[it.name/]" />
      [/for]
  </demo>
[/file]

[/template]
```

File information

Acceleo példa

```
[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/emf/2002/Ecore')/]

[template public generate(e : EClass)]

[comment @main /]
[file (e.name, false, 'UTF-8')]
<?xml version="1.0" encoding="UTF-8"?>
  <demo name=”[e.name/]">
    [for (it : EAttribute | e.eAttributes)]
       <attr name="[it.name/]" />
       [/for]
  </demo>
[/file]

[/template]
```

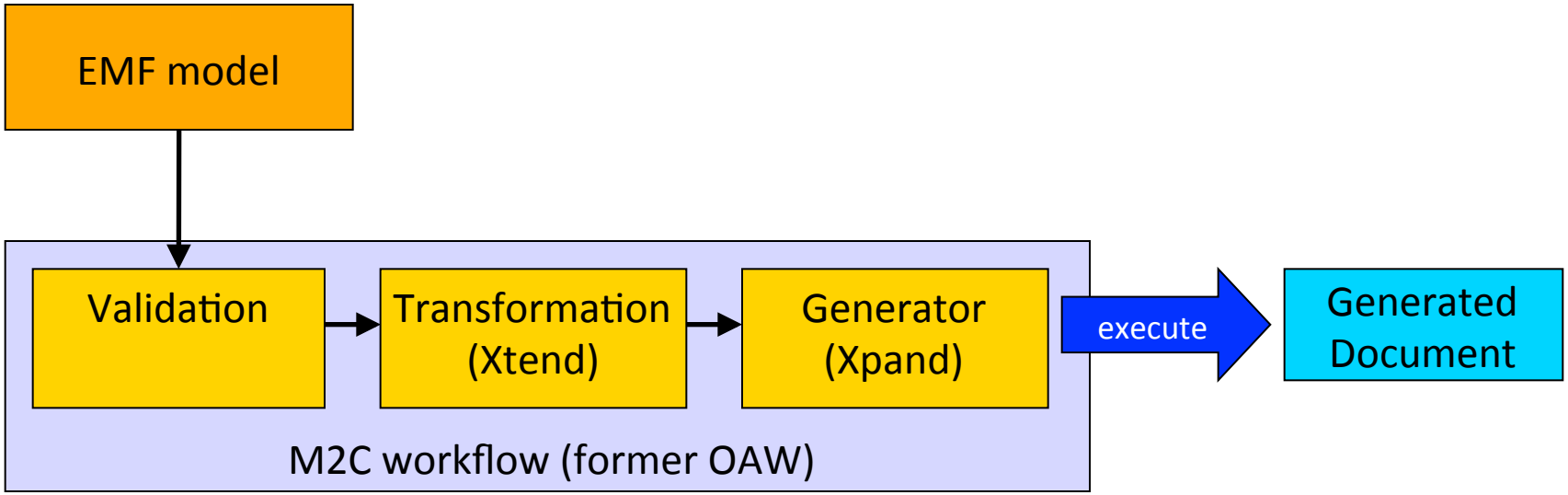
Reference

Acceleo példa

```
[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/emf/2002/Ecore')/]

[template public generate(e : EClass)]

[comment @main /]
[file (e.name, false, 'UTF-8')]
<?xml version="1.0" encoding="UTF-8"?>
  <demo name=”[e.name/]">
    [for (it : EAttribute | e.eAttributes)]
      <attr name="[it.name/]" />
      [/for]
  </demo>
[/file]

[/template]
```

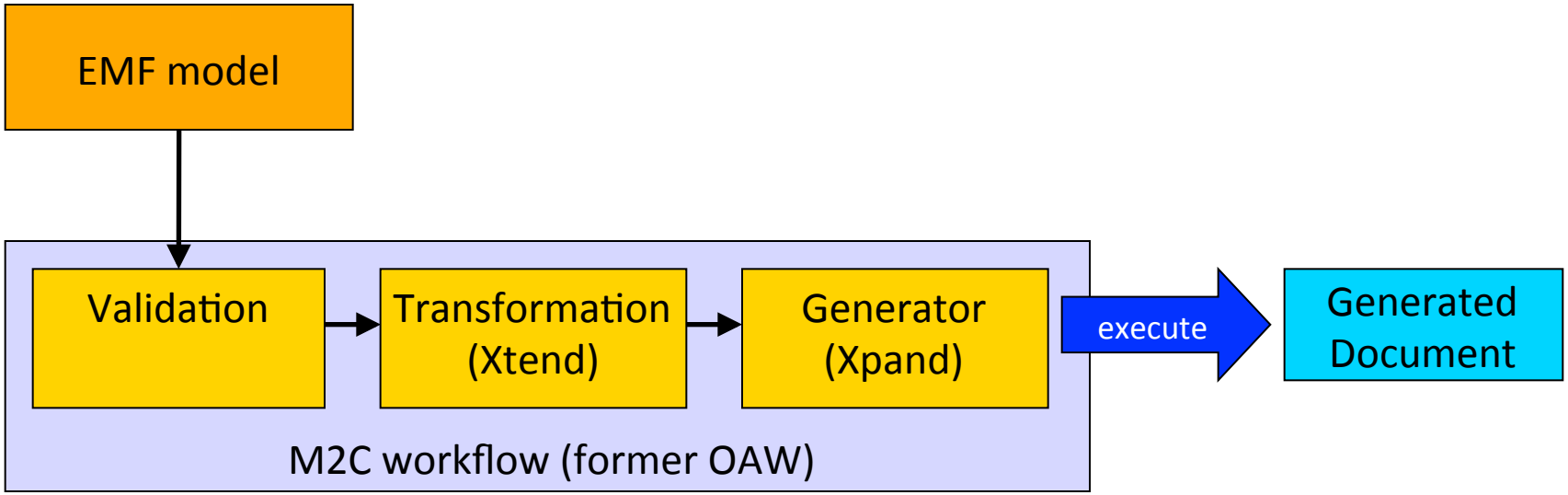
For structure

Xpand (v1)



- Eclipse M2C (korábban OAW)
  - Teljes M2C munkafolyamat
    - Ellenőrzés
    - Transzformáció (Xtend nyelven)
    - Kódgenerálás (Xpand nyelven)
- Alapvetően EMF modellekre kihegyezett
- Rugalmas munkafolyamat definíció

Xpand (v1)



- **Interpretált**
- Sablonnyelv statikus típusokkal
- Polimorf sablonhívások
- Aspektus-orientált programozás támogatása
- Hibakezelés
- Nem-látható karakterek specifikációjának támogatása ☺

Xpand példa

```
«IMPORT XMLmetamodel»
«DEFINE main FOR Model»
   «FILE this.name + ".myxml"»
   <?xml version="1.0" encoding="UTF-8"?>
   <demo>
   «EXPAND listElement FOREACH elements»
   </demo>
   «ENDFILE»
«ENDDEFINE»

«DEFINE listElement FOR Element»
<element> «this.toString()»</element>
«ENDDEFINE»
```

Xpand példa

```
«IMPORT XMLmetamodel»
«DEFINE main FOR Model»
   «FILE this.name + ".myxml"»
   <?xml version="1.0" encoding="UTF-8"?>
   <demo>
   «EXPAND listElement FOREACH elements»
   </demo>
   «ENDFILE»
«ENDDEFINE»

«DEFINE listElement FOR Element»
<element> «this.toString()»</element>
«ENDDEFINE»
```

Import EMF metamodel

Xpand példa

```
«IMPORT XMLmetamodel»
«DEFINE main FOR Model»
   «FILE this.name + ".my xml"»
   <?xml version="1.0" encodi "UTF-8"?>
   <demo>
   «EXPAND listElement FOREACH elements»
   </demo>
   «ENDFILE»
«ENDDEFINE»

«DEFINE listElement FOR Element»
<element> «this.toString()»</element>
«ENDDEFINE»
```

Define template for specific type

Xpand példa

```
«IMPORT XMLmetamodel»
«DEFINE main FOR Model»
   «FILE this.name + ".myxml"»
   <?xml version="1.0" encoding="UTF-8"?>
   <demo>
   «EXPAND listElement FOREACH elements»
   </demo>
   «ENDFILE»
«ENDDEFINE»

«DEFINE listElement FOR Element»
<element> «this.toString()»</element>
«ENDDEFINE»
```

Output file definition

Xpand példa

```
«IMPORT XMLmetamodel»
«DEFINE main FOR Model»
   «FILE this.name + ".myxml"»
   <?xml version="1.0" encoding="UTF-8"?>
   <demo>
   «EXPAND listElement FOREACH elements»
   </demo>
   «ENDFILE»
«ENDDEFINE»

«DEFINE listElement FOR Element»
<element> «this.toString()»</element>
«ENDDEFINE»
```

Start of target document

Xpand példa

```
«IMPORT XMLmetamodel»
«DEFINE main FOR Model»
   «FILE this.name + ".myxml"»
   <?xml version="1.0" encoding="UTF-8"?>
   <demo>
   «EXPAND listElement FOREACH elements»
   </demo>
   «ENDFILE»
«ENDDEFINE»

«DEFINE listElement FOR Element»
<element> «this.toString()»</element>
«ENDDEFINE»
```

EReference holding the elements

Xpand példa

```
«IMPORT XMLmetamodel»
«DEFINE main FOR Model»
   «FILE this.name + ".myxml"»
   <?xml version="1.0" encoding="UTF-8"?>
   <demo>
   «EXPAND listElement FOREACH elements»
   </demo>
   «ENDFILE»
«ENDDEFINE»

«DEFINE listElement FOR Element»
<element> «this.toString()»</element>
«ENDDEFINE»
```

Invoke other template with type definition

Xtend2

- Elvileg Java kiegészítő nyelv
  - Java osztályokra fordul
  - Sokféle funkció
- Különösen jól használható
  - EMF modellek feldolgozására
  - Kódgenerálásra

Xtend2

- Új nyelvi elemek
  - Típus következtetés (type inference)
  - Lambda-függvények (closures)
  - Bővítő metódusok
  - Többsoros stringek

Lambda függvények

- Példa:
  - Gyűjtsük össze személyek neveit!
- Java

```
List<String> names = new ArrayList<String>();
for (String name : persons) {
 names.add(name);
}
```

- Xtend

```
persons.map( p | p.name )
```

Példa: Többsoros stringek

```
@GET
@Produces("text/html")
def sayHighToEverybody(@PathParam("names") String names) '''
    <!DOCTYPE html PUBLIC "-//IETF//DTD HTML 2.0//EN">
    <HTML>
       <HEAD>
          <TITLE>
             Hello «names»!
          </TITLE>
       </HEAD>
    <BODY>
       «FOR name : names.split(',')»
          «greet(name)»
       «ENDFOR»
    </BODY>
    </HTML>
'''

def private greet(String name) '''
        <H1>Hi «name»!</H1>
'''
```

Xtend példa – Minden együtt

```
html [
  head [
   title [$("XML encoding with Xtend")]
  ]
  body [
   h1 [$("XML encoding with Xtend")]
   p [$("this format can be used as an alternative to XML")]

    // an element with attributes and text content
   a("http://www.xtend-lang.org") [$("Xtend")]

    // mixed content
   p [
     $("This is some")
     b[$("mixed")]
     $("text. For more see the")
     a("http://www.xtext.org")[$("Xtext")]
     $("project")
   ]
   p [$("some text")]

   // content generated from arguments
   p [
     for (arg : args)
       $(arg)
   ]
  ]
]
```

Xpand 1 vs Xtend 2

- Xpand 1 a régebbi technológia
  - Munkafolyamat támogatás van
  - Ugyanakkor editor támogatás gyengébb
- Xtend 2
  - Jobb editor támogatás
  - Szűkebb scope - egyelőre

További kérdések

(“ördög a részletekben”)

Forráskód vagy AST

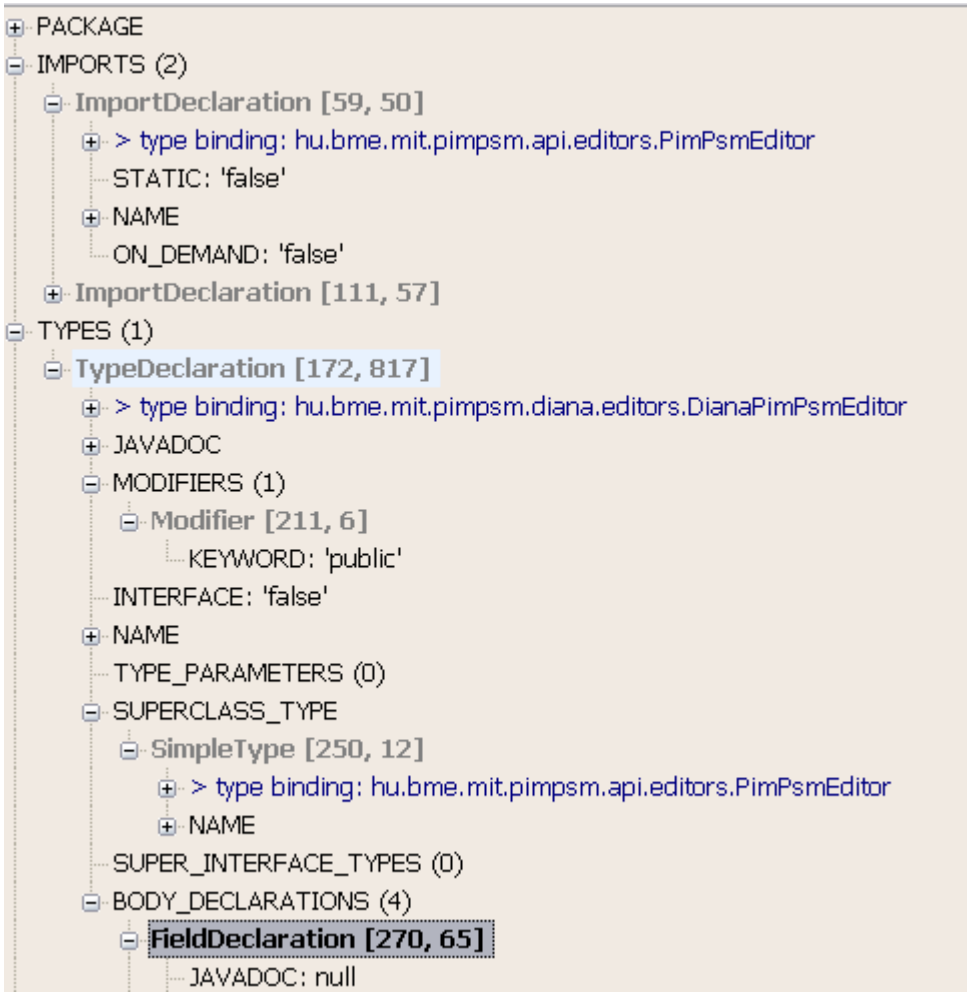
- Közvetlen forráskód generálás
  - Egyszerű struktúra
  - Alacsony komplexitás
  - Gyors fejlesztés
  - Lineáris kimenet generálás
    - (Egymenetes)
  - Formázás?
  - M2C szinkronizáció?

Output:

```
package hu.bme.mit.pimpsm.diana.editors;

import hu.bme.mit.pimpsm.api.editors.PimPsmEditor;

/**
 * @author Akos Horvath
 *
 */
public class DianaPimPsmEditor extends PimPsmEditor
{

    public static final String PLUGIN_ID = "hu.mit.bme.pimp

    /* (non-Javadoc)
     * @see hu.bme.mit.pimpsm.api.editors.PimPsmEditor#cre
     */
    @Override
    public PimPsmModelManager createModelManager() {
        // TODO Auto-generated method stub
        return new DianaPimPsmModelManager(this);
    }

    /** Have to return the exact id of the project for the
     * in order to be able to include the icons
     */
```

Forráskód vagy AST

Output:



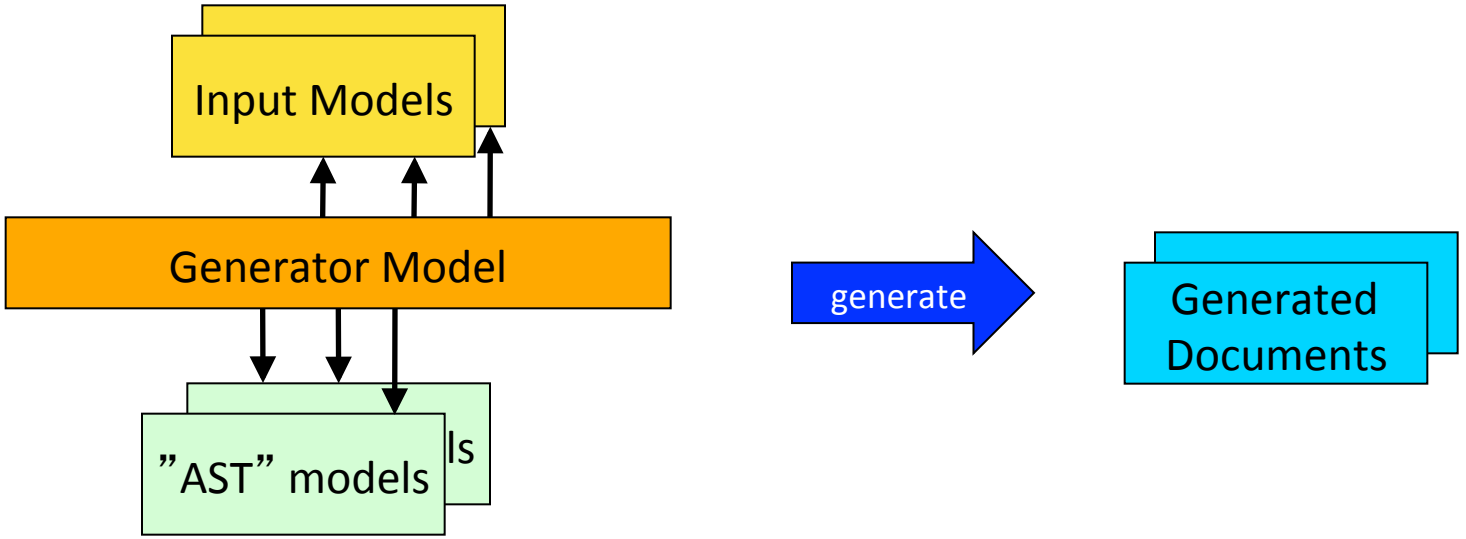
- AST generálás
  - Program struktúra a kimenet
  - Komplex lehet
  - Lassabb fejlesztés
  - Nem-lineáris folyamat
    - Többlépcsős
  - M2C támogatás
  - Inkrementális kódgenerálás
  - ”pretty printing”
  - Pl.
    - JDT (Java AST),
    - IMP, Xtext

Forráskód vagy AST

- Közvetlen forráskód generálás
  - Egyszerű struktúra
  - Alacsony komplexitás
  - Gyors fejlesztés
  - Lineáris kimenet generálás
    - (Egymenetes)
  - Formázás?
  - M2C szinkronizáció?

- AST generálás
  - Program struktúra a kimenet
  - Komplex lehet
  - Lassabb fejlesztés
  - Nem-lineáris folyamat
    - Többlépcsős
  - M2C támogatás
  - Inkrementális kódgenerálás
  - ”pretty printing”
  - Pl.
    - JDT (Java AST),
    - IMP, Xtext

Generator modell



- Több forrás modell → **”generator” modell**
- Minden beállítás itt adható meg
- Hivatkozik a bemenetre és az “AST” modellekre
- Segíti a kódgenerálást
  - Modellhierarchiák kezelése (több AST, csomagok, stb.)
  - Nem-lineáris, többmenetes kódgenerálás támogatása
  - Nyomonkövethetőség modellek közt→ hivatkozások
  - Több output stream használata

Modell-kód szinkronizáció

- Mi történik, ha a generált szöveg megváltozik?
  - M2C szinkronizáció
- Csak AST alapú megközelítéseknél működik
- Feltételek
  - Szöveg és modell közti nyomkövetési információ
  - Modellek összehasonlítása
  - Változás helyének meghatározása
- Inkrementális modellépítés jobb teljesítményért
- Példa
  - Eclipse JDT: Java forrás és AST
  - EMF: generátor modell

Kód formázás

- Hol vegyük fel?
  - Modellben
    - Nem követi a szokásos MVC paradigmát
  - Sablonban
    - Elem-szintű formázás
  - AST felvételekor
    - Minden információt tárol
    - Bonyolult lehet
- Legjobb megoldás
  - Külön lépés a kódgenerálási munkafolyamatban
  - Külső eszközt is igénybe lehet venni
    - Eclipse JDT formatter
    - XML DOM serializer

Kulcsszavak és speciális karakterek

- Foglalt kulcsszavak a célmodellben
  - Java: abstract, class
  - XML: ‘<‘ , ‘>’
  - …
- Kódgenerálás előtt a modellt ellenőrizni kell
  - Bonyolult lehet → külön lépés generálás előtt
  - Példa
    - Java: isJavaIdentifierStart() (in Character)
    - EMF validation
- Escaping
  - Modellben
  - Csak a generált kódban

Rögzített kódrészletek generálása

- Nem-változó részek elhelyezése
  - Modellben
    - Újrahasznosíthatóság
    - Bonyolult
  - Sablonban
    - Jól működik az egyszerű esetekre
  - AST
    - Fix AST a fix részekre→ a kód generátor csinálja meg a kódot
  - Közvetlen kód
    - Java → nincs támogatás☹
      - Öröklődés segít
    - C# partial classes ☺

Kódgenerálás kezdeményezése

- Kézzel
  - Eddig ilyen
- Automatikusan modellváltozáskor
  - Eclipse builder mechanizmus
- Melyiket érdemes?

Kódgenerálás kezdeményezése

- Kézzel
  - Összetett műveletek
  - Workflow alapú megközelítés szóbajöhet
  - Lehet viszonylag lassú
- Modellváltozásra
  - Egyszerűbb műveletek
  - Változások figyelése
    - Bonyolult vezérlés
  - Teljesítménykényszer
  - Kényelmes, ha a kódgenerátor egy osztály

Kézzel és generált kódok kezelése

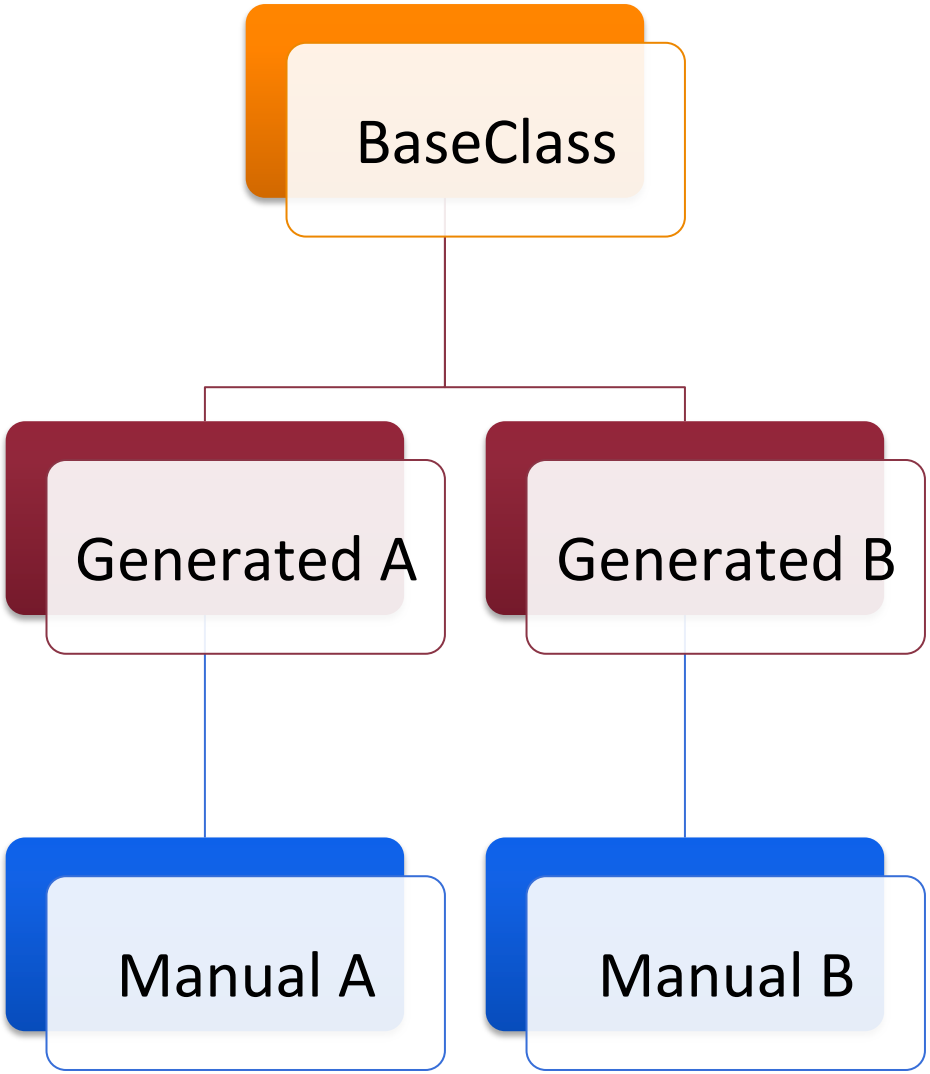
Generált kód kezelése - Ökölszabályok

- Ne keverjük a generált és a kézi kódot
  - Nem szeretjük, ha az újragenerálás felülírja ☺
- Ne töltsük a generált kódot verziókezelőbe
  - Reprodukálható
- Kódgenerálás a fordítási folyamat része
  - Szinkron modell és kód között
- Formázott kód generálás
  - Még ha nem is írjuk, de olvassuk!
- Használjunk jó runtime környezetet!
  - Kevesebb kódot kell generálni

Technika

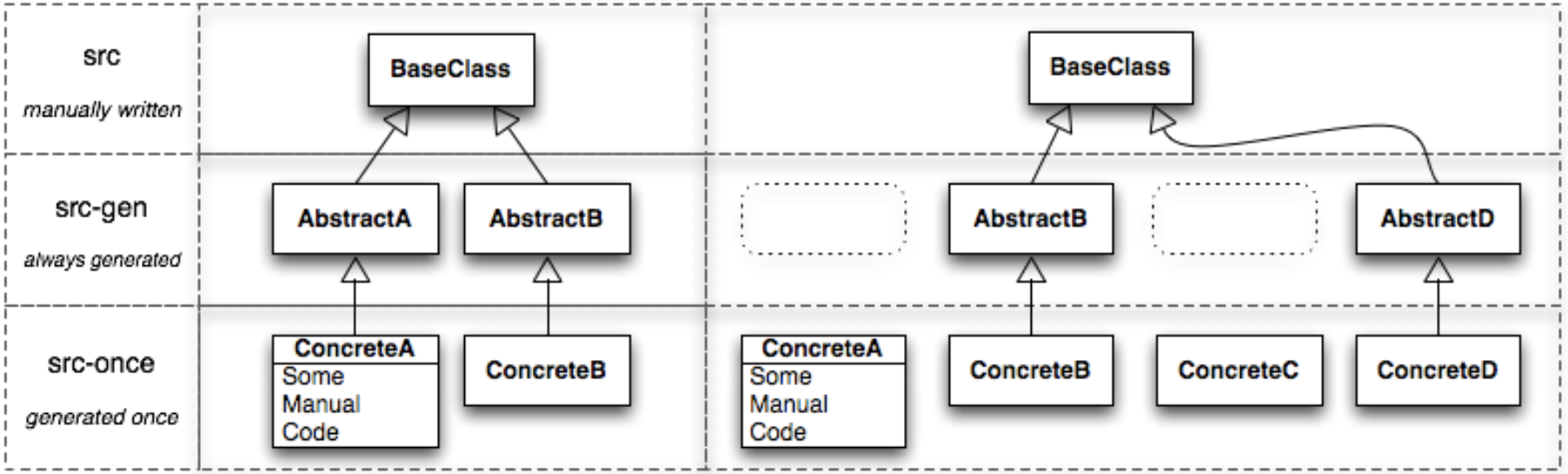
- Java projekt
  - Több source folder lehet
  - Eclipse fordításkor összefésüli őket
    - Külön mappa generált és kézzel írt kódnak

Generation Gap minta



- Feltételek
  - Automatikus kódgenerálás
  - Egy vagy több osztályt generálunk
  - Általában megmaradnak az interfész és a példányváltozók
  - A generált osztályok nem kerülnek be létező osztálykönyvtárakban

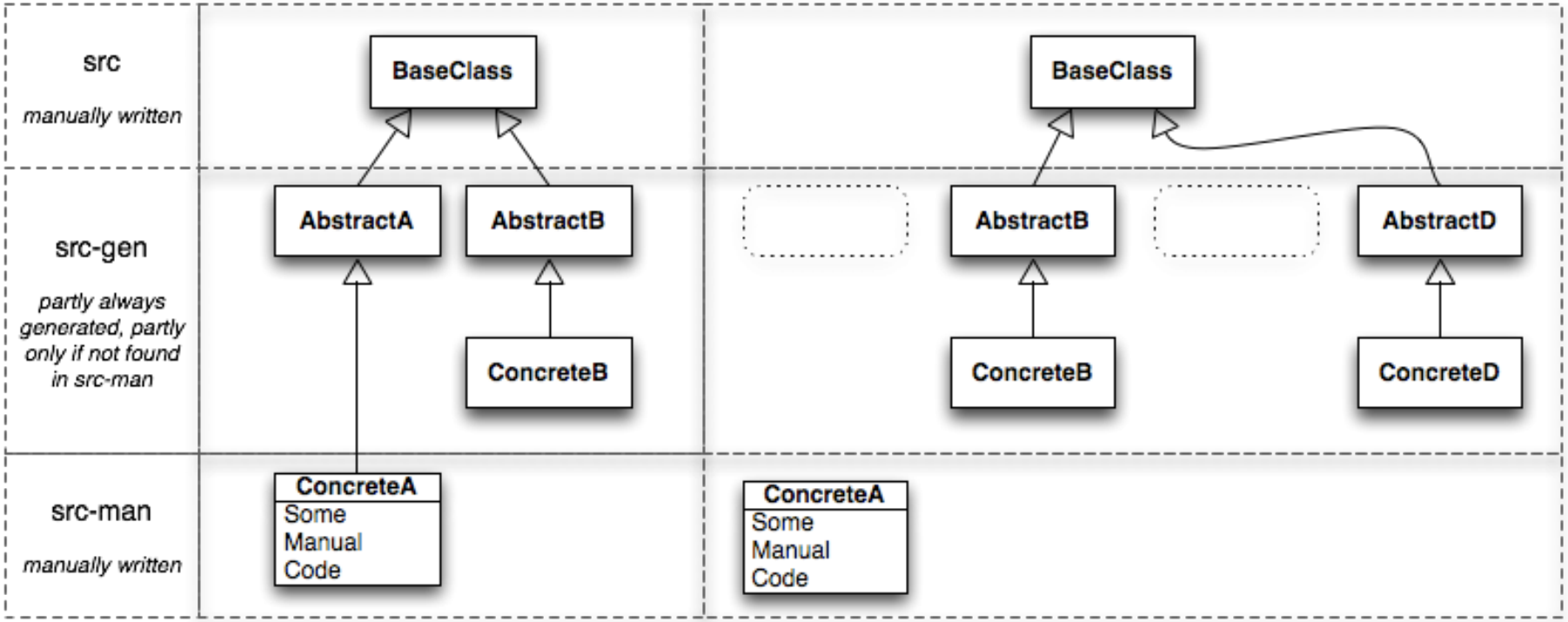
“Generate once”



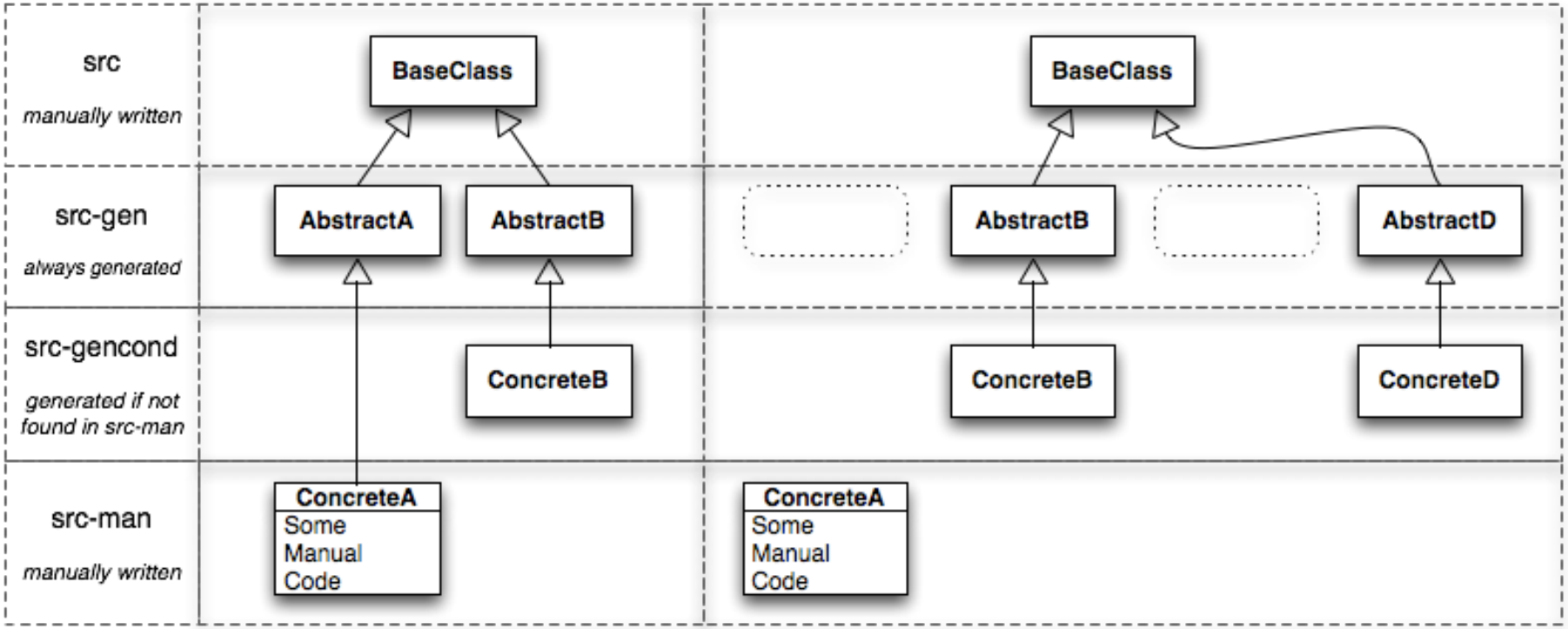
Generate once

- Generáljunk egy kezdeti változatot
  - Később nem generáljuk újra
  - Nyugodtan módosítható
- Példa
  - Mintakód készítése

Feltételes kódgenerálás – 1.



Feltételes kódgenerálás – 2.



Feltételes kódgenerálás

- Kódgenerálás
  - Ha nincs felülírva a kézi mappában
- Cél
  - Generation Gap
    - De nincsen leszármazott osztály kézi kódnak
    - Pro: Kevesebb osztály, többször használható
    - Kontra: Nem örököljük a frissítéseket

Kódgenerálási stratégiák

- Stratégia megadása
  - JET
    - vezérlő Java kód
  - Acceleo
    - Nincs közvetlen támogatás
  - Xpand
    - MWE segít
  - Xtend
    - Vezérlő kód írható

Összefoglalás

Összefoglalás

- **Kiindulás: forráskód generálás**
  - UML -> Java, C++, ….
- Más területeken is használható
  - Dokumentumgenerálás (web)
  - Jelentésgenerálás (XML, XLS, CSV, print)
  - Konfiguráció generálás (wsdl)
- Jó eszköztámogatás
  - JET
  - Acceleo, Xpand, Xtend2
  - (CodeDOM)
- Nem csak MDD környezetben hasznos!!!