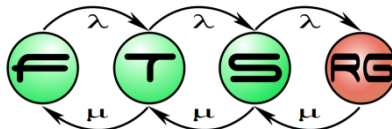


Kódgenerálás

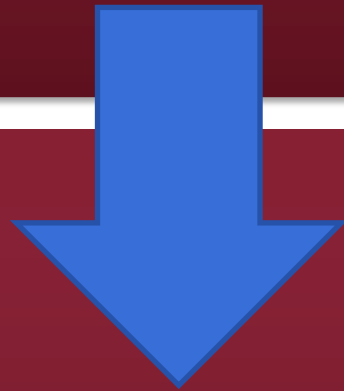


- Metamodellezés
 - Absztrakt szintaxis
 - EMF
 - Konkrét szintaxis
 - GEF, GMF, IMP, Xtext
 - Jólformáltsági szabályok
 - Részben
 - Szemantika
 - **Leképezés más nyelvre**
 - Ez lesz most

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

Kódgenerálás - Példák

Ecore modell

- EMF modell, edit és editor

GMF Diagram modell

- Grafikus editor

Xtext nyelvtan

- Szöveges editor

Javadoc komment generálás - JAutodoc

```
public void setNumberOfQuestions(int numberOfQuestions)
    throws IllegalArgumentException {
    if (numberOfQuestions < 0 ) {
        throw new IllegalArgumentException("numberOfQuestions < 0");
    }
    this.numberOfQuestions = numberOfQuestions;
}
```


Javadoc komment generálás - JAutodoc

```
/**
 * Sets the number of questions.
 *
 * @param numberOfQuestions the number of questions
 * @throws IllegalArgumentException the illegal argument exception
 */
public void setNumberOfQuestions(int numberOfQuestions)
                                throws IllegalArgumentException {
    if (numberOfQuestions < 0 ) {
        throw new IllegalArgumentException("numberOfQuestions < 0");
    }
    this.numberOfQuestions = numberOfQuestions;
}
```

Javadoc komment generálás - JAutodoc

- Nem túl jó példa
 - Buta generátor
- Kódgenerátor
 - Modell jelentésére épít
 - Nem hoz létre új tudást

```
/**
 * Sets the number of questions.
 *
 * @param numberOfQuestions the number of questions
 * @throws IllegalArgumentException the illegal argument exception
 */
public void setNumberOfQuestions(int numberOfQuestions)
    throws IllegalArgumentException {
    if (numberOfQuestions < 0 ) {
        throw new IllegalArgumentException("numberOfQuestions < 0");
    }
    this.numberOfQuestions = numberOfQuestions;
}
```

Javadoc komment generálás - JAutodoc

```
/**
 * Sets the number of questions.
 *
 * @param numberOfQuestions the number of questions
 * @throws IllegalArgumentException the illegal argument exception
 */
public void setNumberOfQuestions(int numberOfQuestions)
                                throws IllegalArgumentException {
    if (numberOfQuestions < 0 ) {
        throw new IllegalArgumentException("numberOfQuestions < 0");
    }
    this.numberOfQuestions = numberOfQuestions;
}
```

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

Domain-specifikus modell

Kód
generálás

Magas szintű nyelv

Fordítás

Assembly

Megközelítések

Megközelítések

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

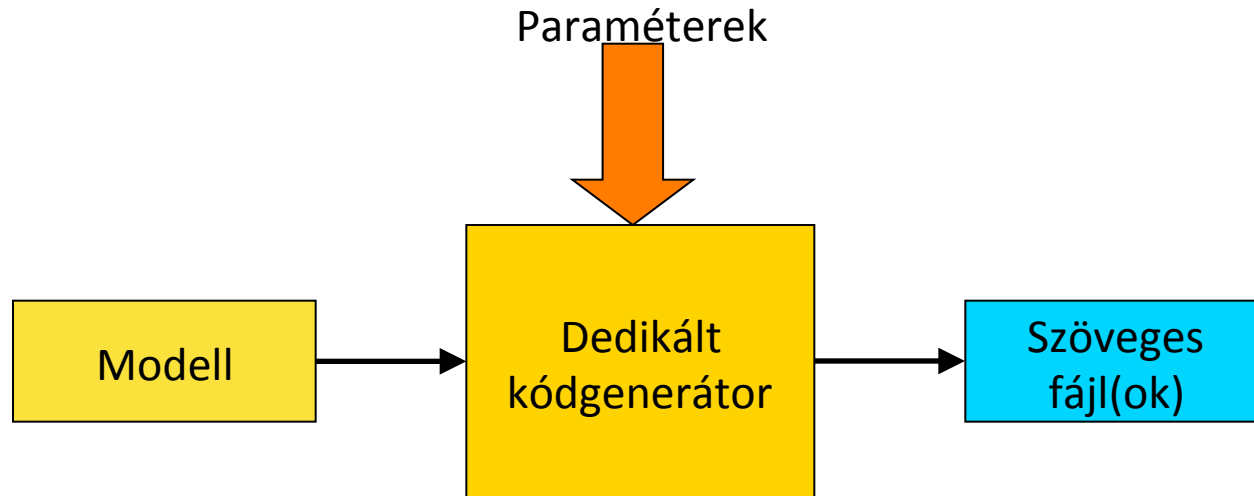
Speciális, ad-hoc

```
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)

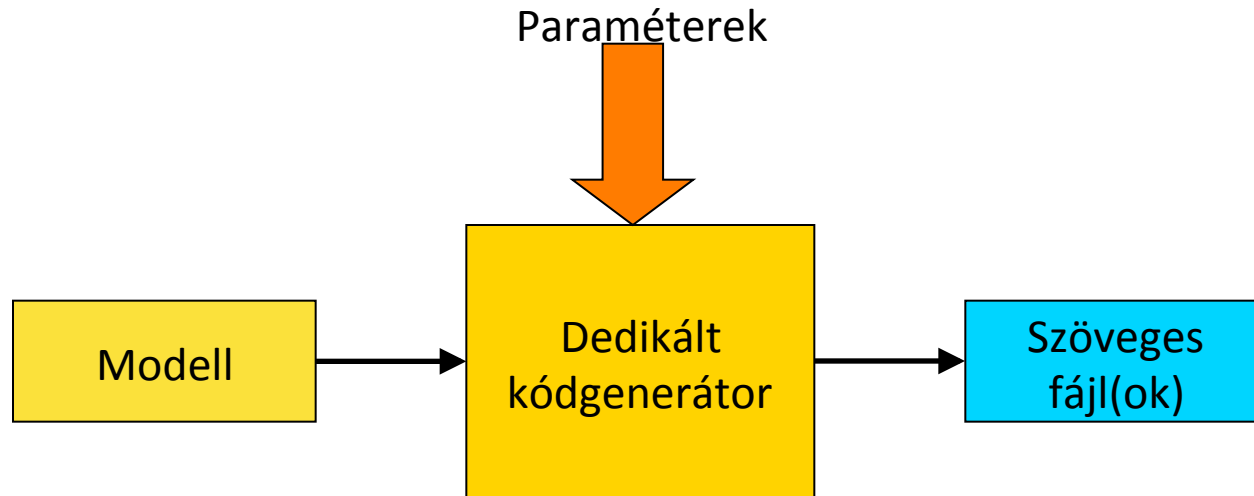
Dedikált kódgenerátor



■ Keretrendszer alapú

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

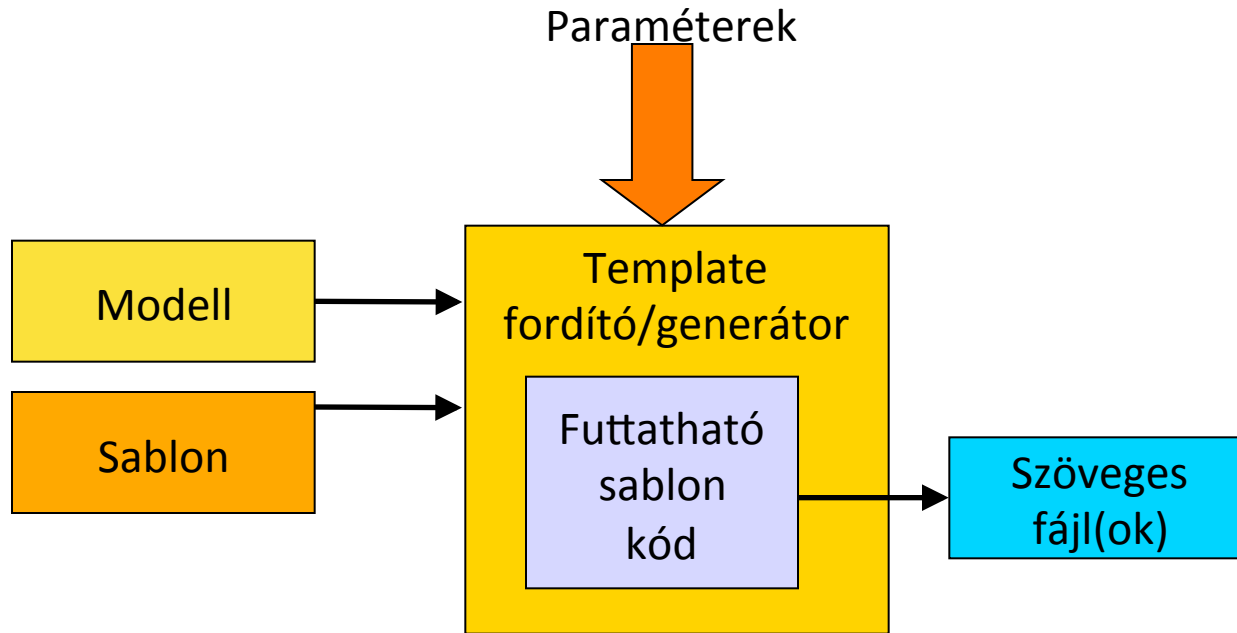
Dedikált kódgenerátor



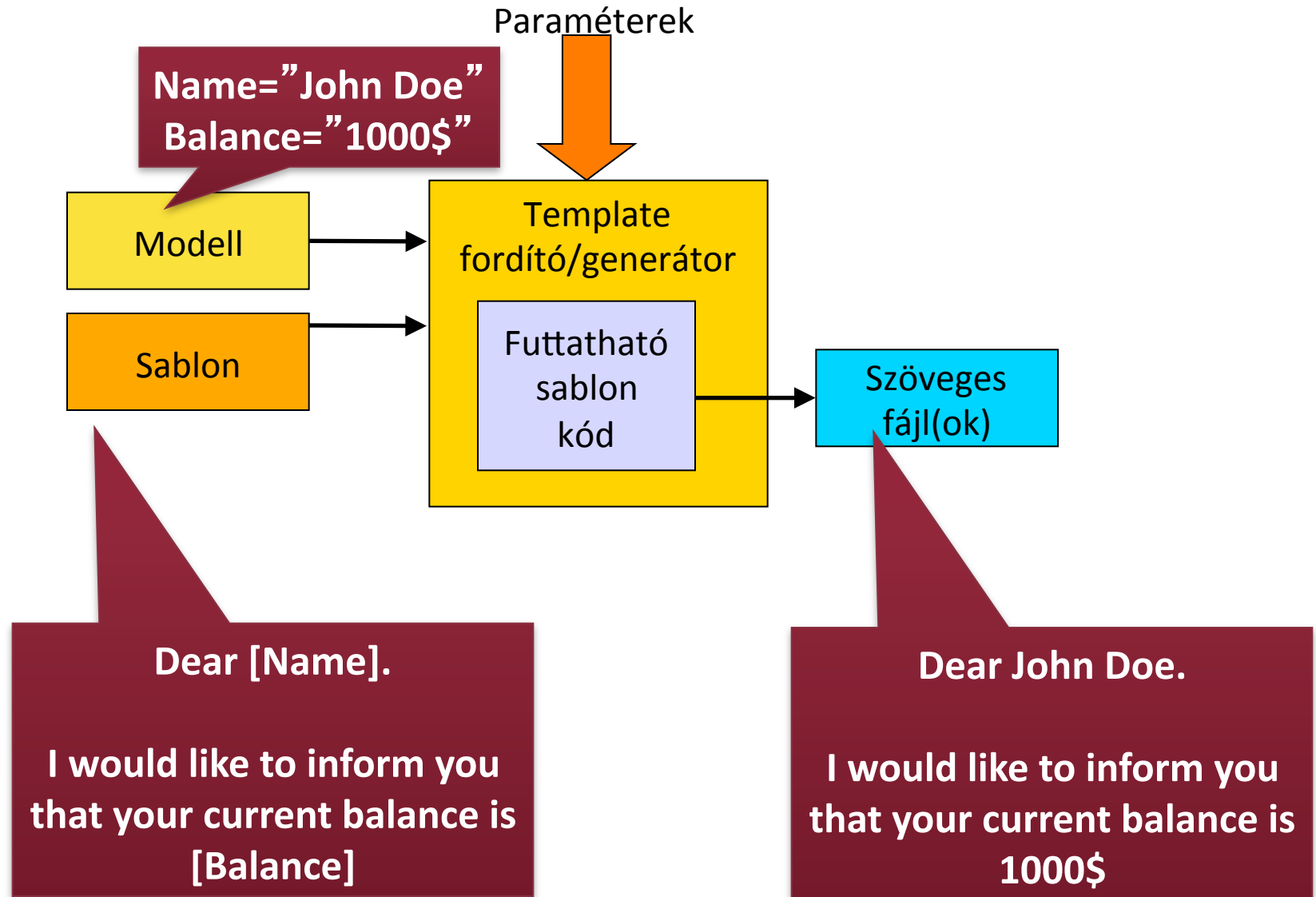
■ Példák:

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

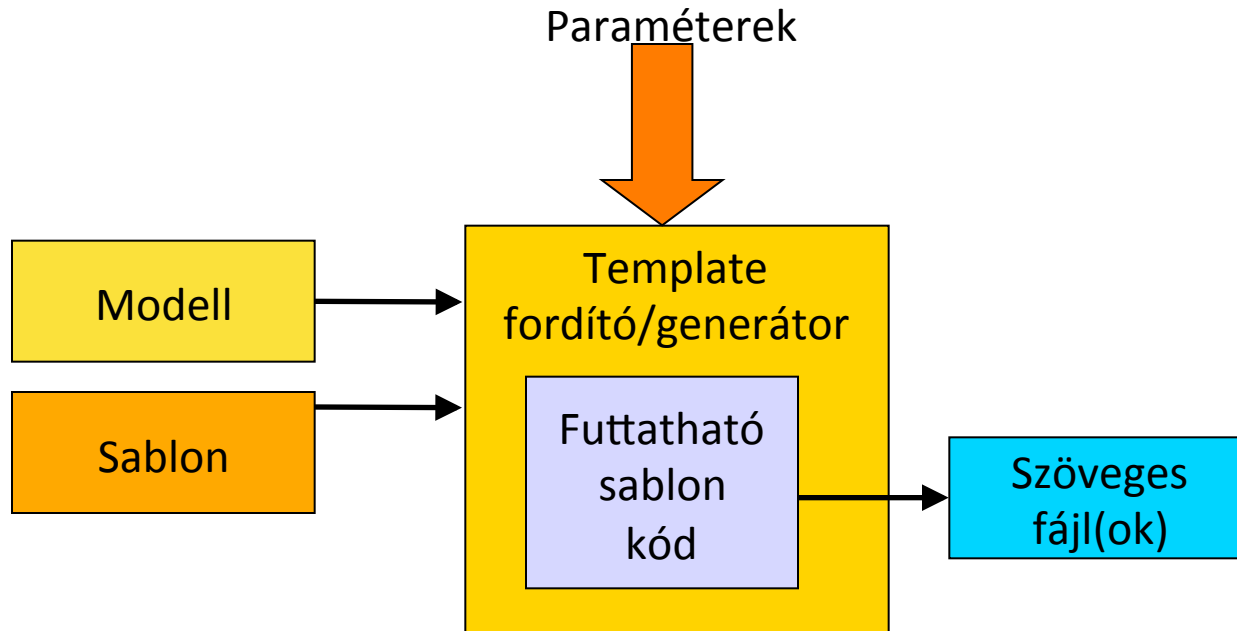
Sablon alapú



Sablon alapú

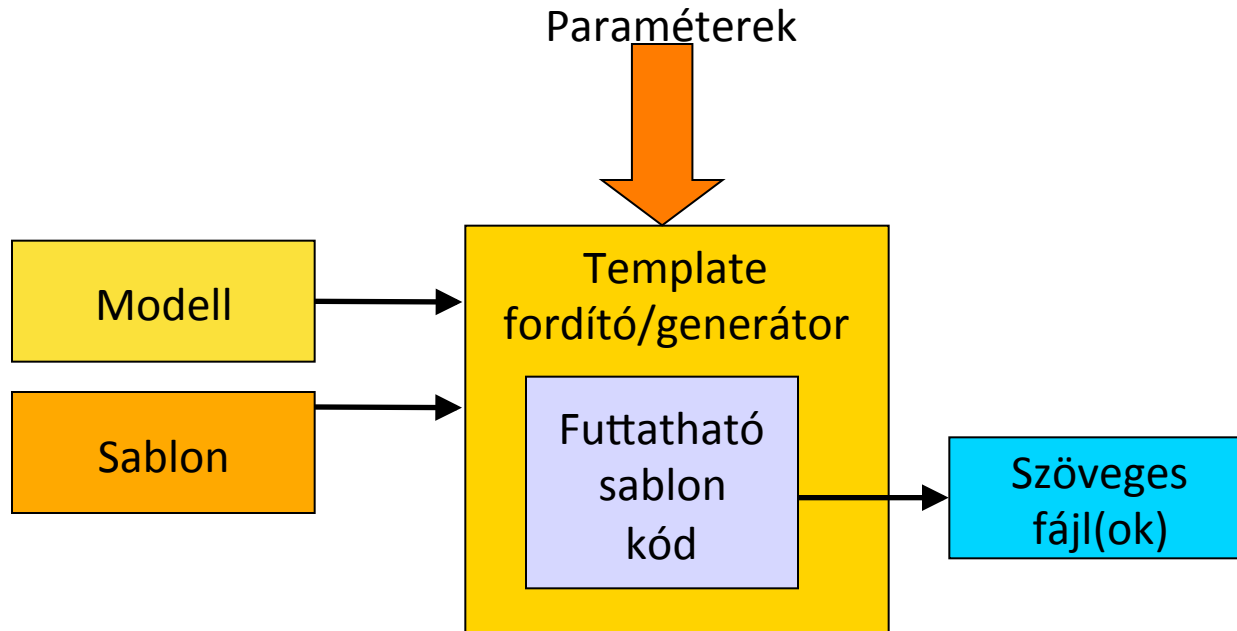


Sablon alapú



- 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



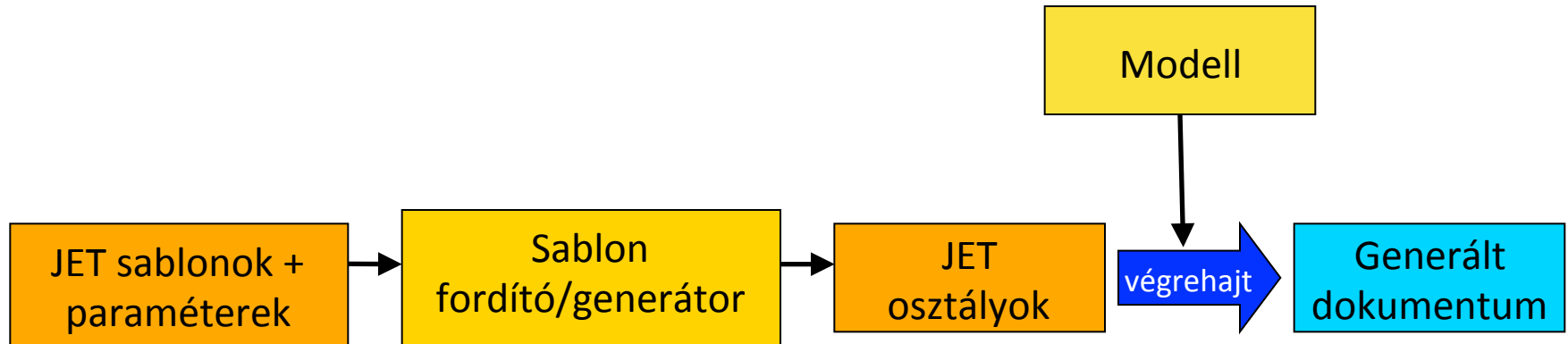
■ Példák:

- 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-hez készült
 - 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 = elementList.iterator(); 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; %>
<?xml version="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" encoding="Input parameter" ?>  
<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>
```

End of code section

JET példa

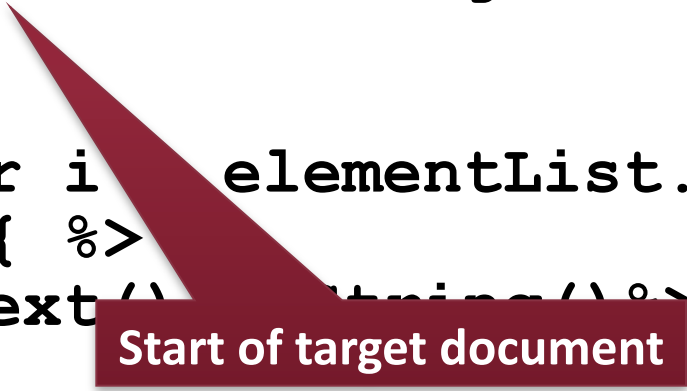
```
<%@ jet package="hello"
imports="java.util.*" class="XMLDemoTemplate"
%>
<% List elementList = (List) argument; %>

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

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

<% } %>

</demo>
```



Start of target document

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>

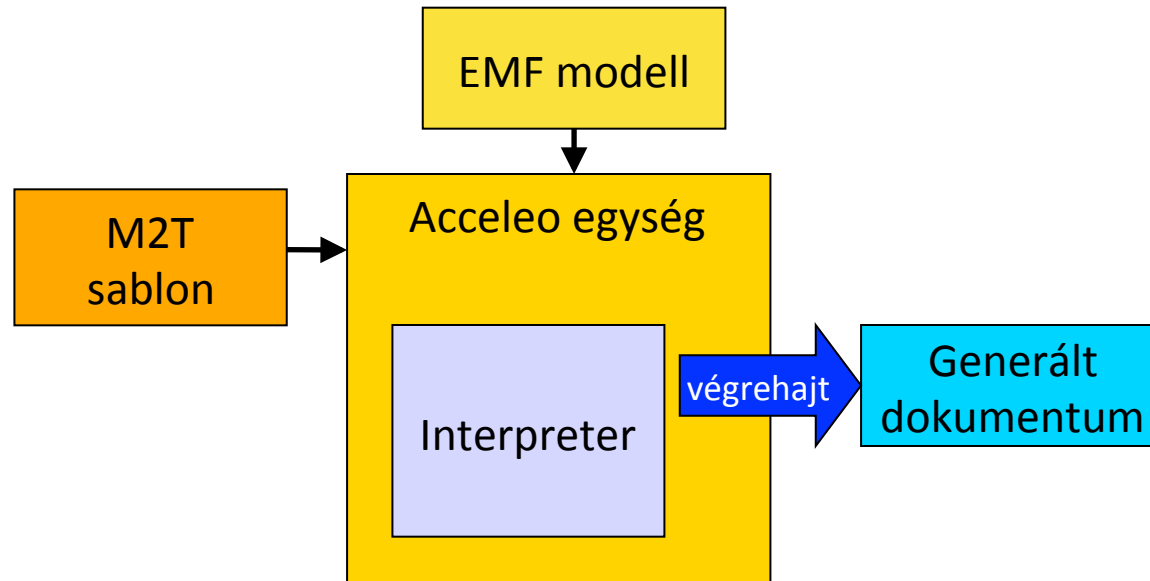
<% for (Iterator i = elementList.iterator();
i.hasNext(); ) { %>
<element><%=i.next().toString()%></element>

<% } %>

</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)]
```

Template

```
[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]
```

Acceleio példa

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

```
[template public generate(e : EClass)]
```

```
[comment @main /] Entry point
```

```
[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]
```

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' )/]

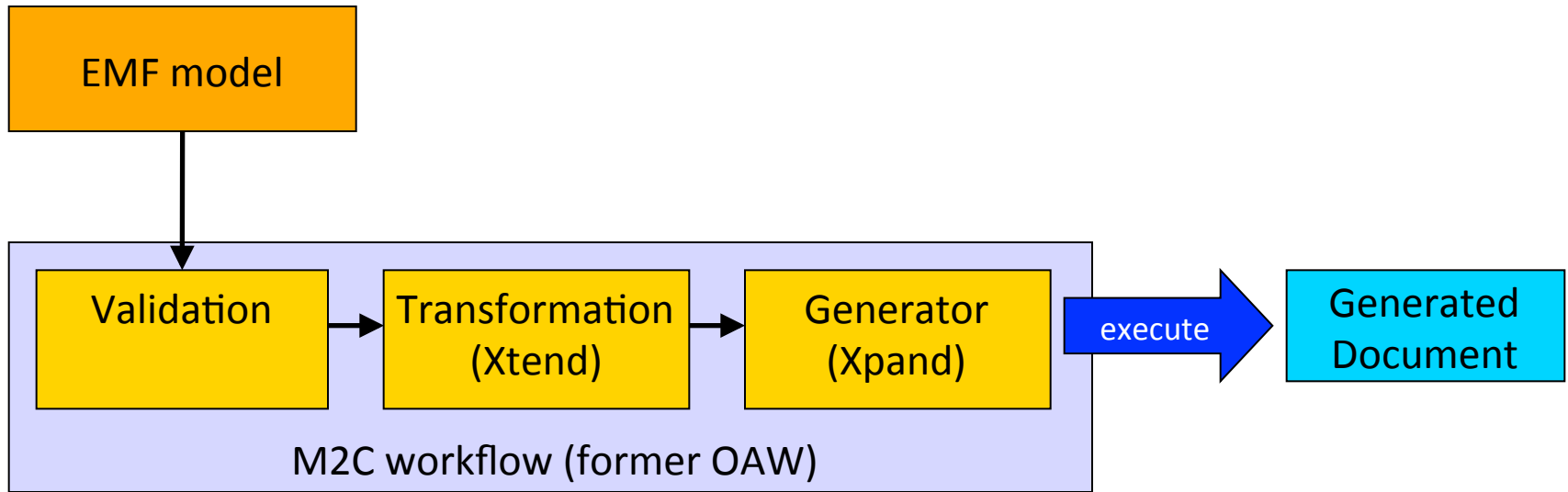
[template public generate(e : EClass)]

[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]
```

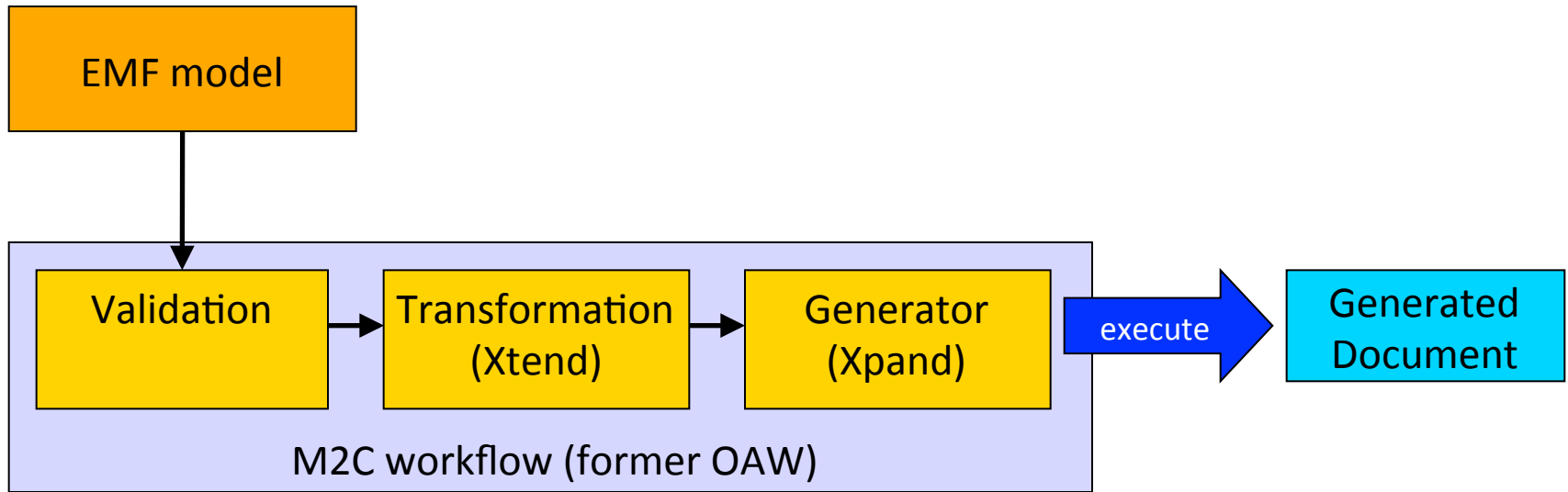
For structure

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»
```

Import EMF metamodel

```
«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 + ".my.xml"»  
  <?xml version="1.0" encoding="UTF-8"?>  
  <demo>  
    «EXPAND listElement FOREACH elements»  
  </demo>  
  «ENDFILE»  
«ENDDEFINE»
```

Define template for specific type

```
«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»
```

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»
```

Invoke other template with type definition

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

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

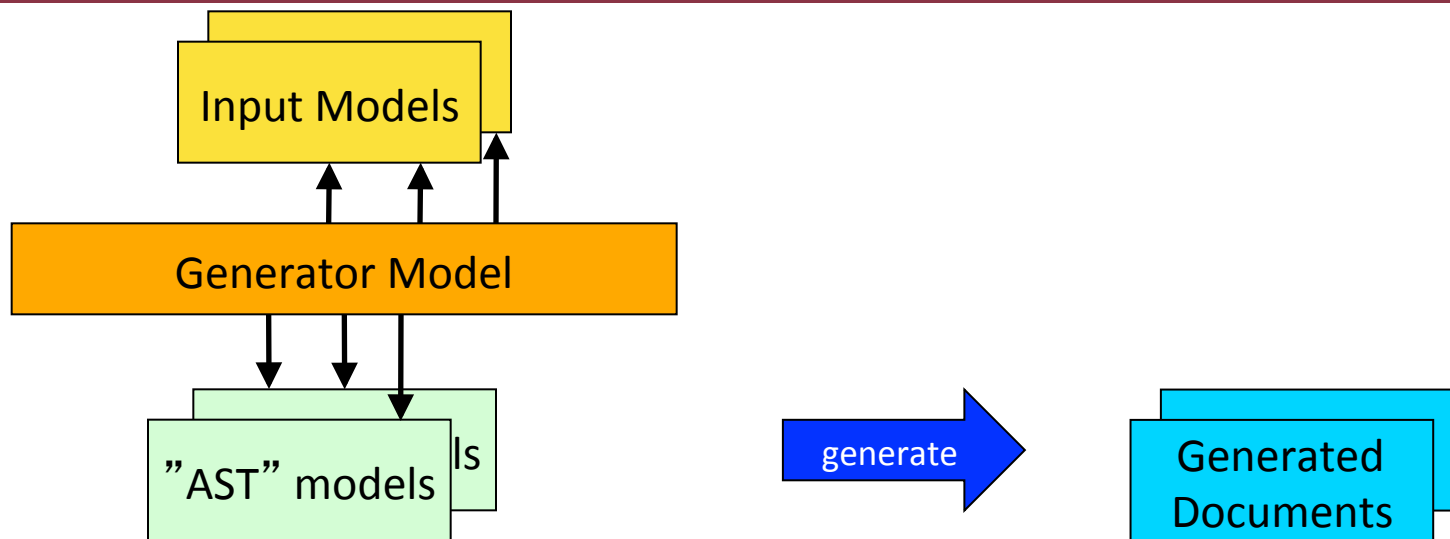
További kérdések

(“ördög a részletekben”)

Hibakezelés

- Kódgenerátor hibák keresése
 - Igényli forrás- és célnyelv együttes ismeretét!
 - Ne legyen rá szükség!
- Forrásnyelv szintű validáció
 - Statikusan
 - Nyelv-specifikus jól-formáltsági kényszerek
 - Kódgenerálás során
 - Pl. célnyelv foglalt kulcsszavainak kezelése

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

Forráskód vs AST generálás

	Forráskód generálás	AST generálás
Kimenet	Szöveg	Program struktúra
Komplexitás	Alacsony	Komplex lehet
Fejlesztés	Gyors	Lassú
Végrehajtás	Egylépcsős Lineáris	Többmenetes Nem-lineáris
Inkrementalitás	Legfeljebb fájl szinten	Megoldható fájlon belül is
“Pretty printing”	Utófeldolgozásként	AST sorosítás közben
Modell-kód szinkronizáció	?	AST szinten megoldható
Értékelés	Gyors megoldás Egyszerű esetekre	Komplex esetek kezelésére

Forráskód vs AST generálás

	Forráskód generálás	AST generálás
Kimenet	Szöveg	Program struktúra
Komplexitás	Alacsony	Komplex lehet
<pre> 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#createModelManager() */ @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 */ </pre>		Lassú
		Többmenetes Nem-lineáris
	Inten	Megoldható fájlban belül is
	ént	AST sorosítás közben
		AST szinten megoldható
re		Komplex esetek kezelésére

Forráskód vs AST generálás

	Forráskód generálás	AST generálás
Kimenet	Szöveg	
Komplexitás	Alacsony	

<pre>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.pim /* (non-Javadoc) * @see hu.bme.mit.pimpsm.api.editors.PimPsmEditor#create */ @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 */ }</pre>	<pre>+ PACKAGE - IMPORTS (2) - ImportDeclaration [59, 50] + > type binding: hu.bme.mit.pimpsm.api.editors.PimPsmEditor - STATIC: 'false' + NAME - ON_DEMAND: 'false' - ImportDeclaration [111, 57] - TYPES (1) - TypeDeclaration [172, 817] + > type binding: hu.bme.mit.pimpsm.diana.editors.DianaPimPsmEditor + JAVADOC - MODIFIERS (1) - Modifier [211, 6] - KEYWORD: 'public' - INTERFACE: 'false' + NAME - TYPE_PARAMETERS (0) - SUPERCLASS_TYPE - SimpleType [250, 12] + > type binding: hu.bme.mit.pimpsm.api.editors.PimPsmEditor + NAME - SUPER_INTERFACE_TYPES (0) - BODY_DECLARATIONS (4) - FieldDeclaration [270, 65] - JAVADOC: null</pre>
--	--

Modell-kód szinkronizáció

- Mi történik, ha a generált szöveg megváltozik?
 - M2C szinkronizáció
- Csak AST alapú megközelítésekénél 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

Kódgenerálás kezdeményezése

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

Kódgenerálás kezdeményezése

Kézzel

- Összetett műveletek
- Workflow alapú megközelítés szóba jöhet
- Lehet viszonylag lassú

Modellváltásra

- Egyszerűbb műveletek
- Változások figyelése
 - Bonyolult vezérlés
- Teljesítménykényszer

Kódgenerálás kezdeményezése

IDE környezetben ez praktikus

Kézzel

- Összetett műveletek
- Workflow alapú megközelítés szóbjöhet
- Lehet viszonylag lassú

Modellváltásra

- Egyszerűbb műveletek
- Változások figyelése
 - Bonyolult vezérlés
- Teljesítménykényszer

Build automatizációhoz ez kell!

Kézzel és generált kódok kezelése

Generált és kézzel írt kód

Modell

Generált
kód

Kézzel írt
kód

Kézi kód

Generált kód

Runtime kód

Kódminőség

- Generált kód legyen olyan jó, mint a kézzel írt!
 - Még ha nem is írjuk, de olvassuk!
 - Formázott kód generálás
 - Kód dokumentáció
- Használjunk jó runtime környezetet!
 - Kevesebb kódot kell generálni
 - Saját runtime: absztrakt őssosztályok használhatóak

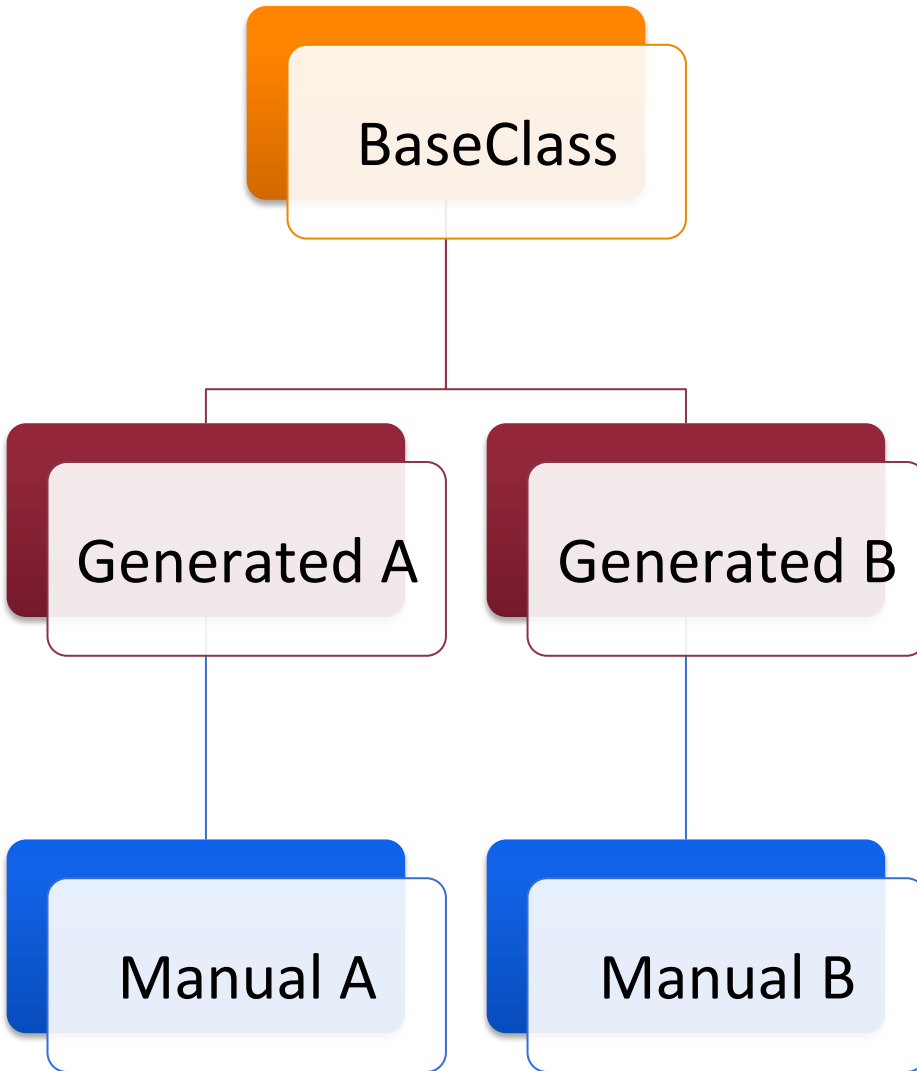
Generált kód kezelése – Ökölszabályok 1.

- Ne töltsük a generált kódot verziókezelőbe
 - Reprodukálható
 - Felesleges konfliktust okozhat
 - Pl. fájlvége karakterek megváltozhatnak
- Kódgenerálás a fordítási folyamat része
 - Beleértve automatikus fordítást is!
 - Szinkron modell és kód között

Generált kód kezelése – Ökölszabályok 2.

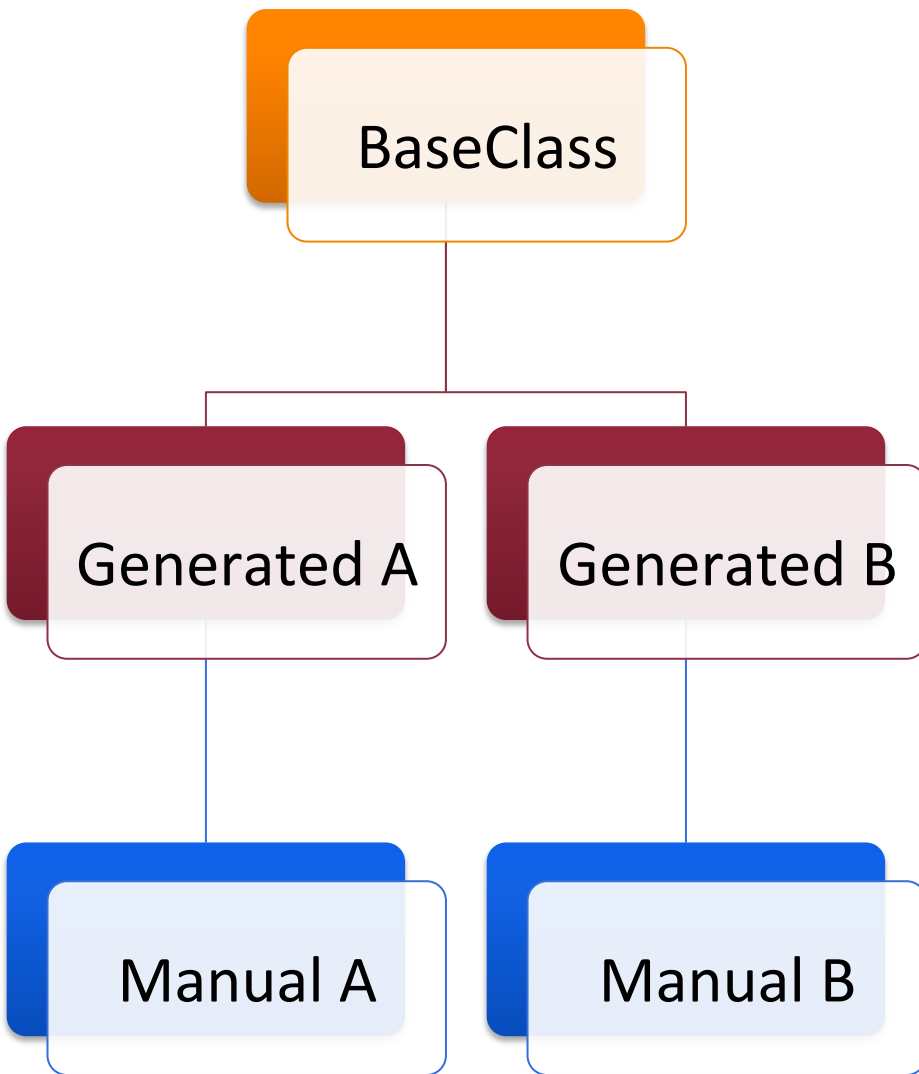
- Egy fájl legyen vagy generált vagy kézi
 - Nem szeretjük, ha az újragenerálás felülírja 😊
 - Átláthatóbb
 - Verziókezelő rendszer!

Generation Gap minta



- Testreszabás
 - Öröklődés mentén
 - Leszármazottban felülírás
- Generált kódhoz nem nyúlunk
- Kérdés:
 - EMF modellek módosításához megfelelő?

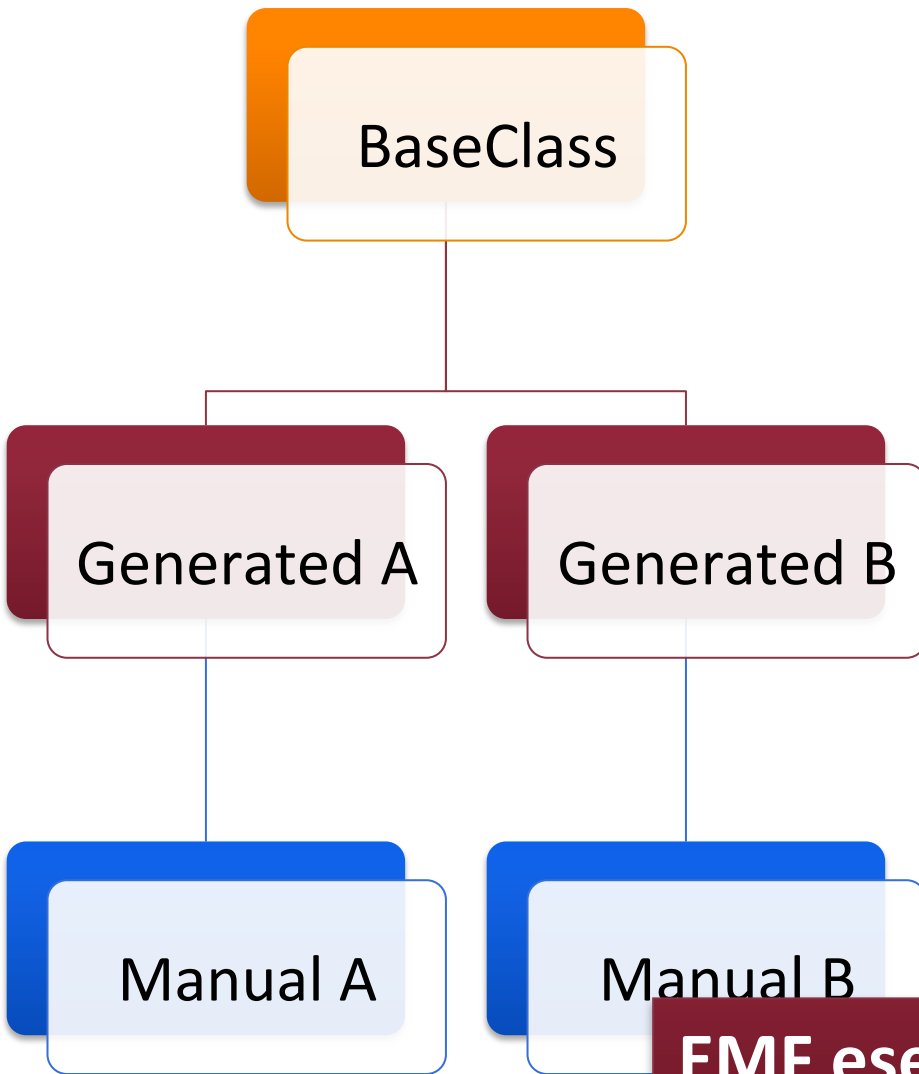
Generation Gap minta



■ Feltételek

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

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

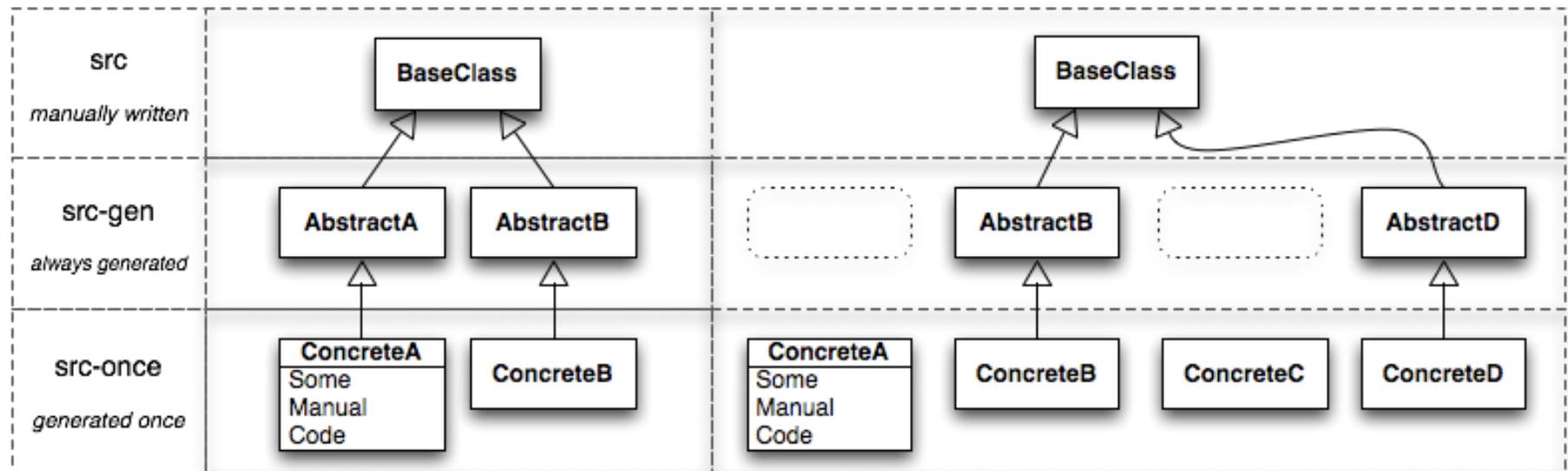
EMF esetén ez az utolsó feltétel sérül!

Kód elhelyezési stratégiák

■ Java projekt

- Több source folder lehet
 - Választhatunk megfelelő célt
- Eclipse fordításkor összefésüli őket
 - Külön mappa generált és kézzel írt kódnak

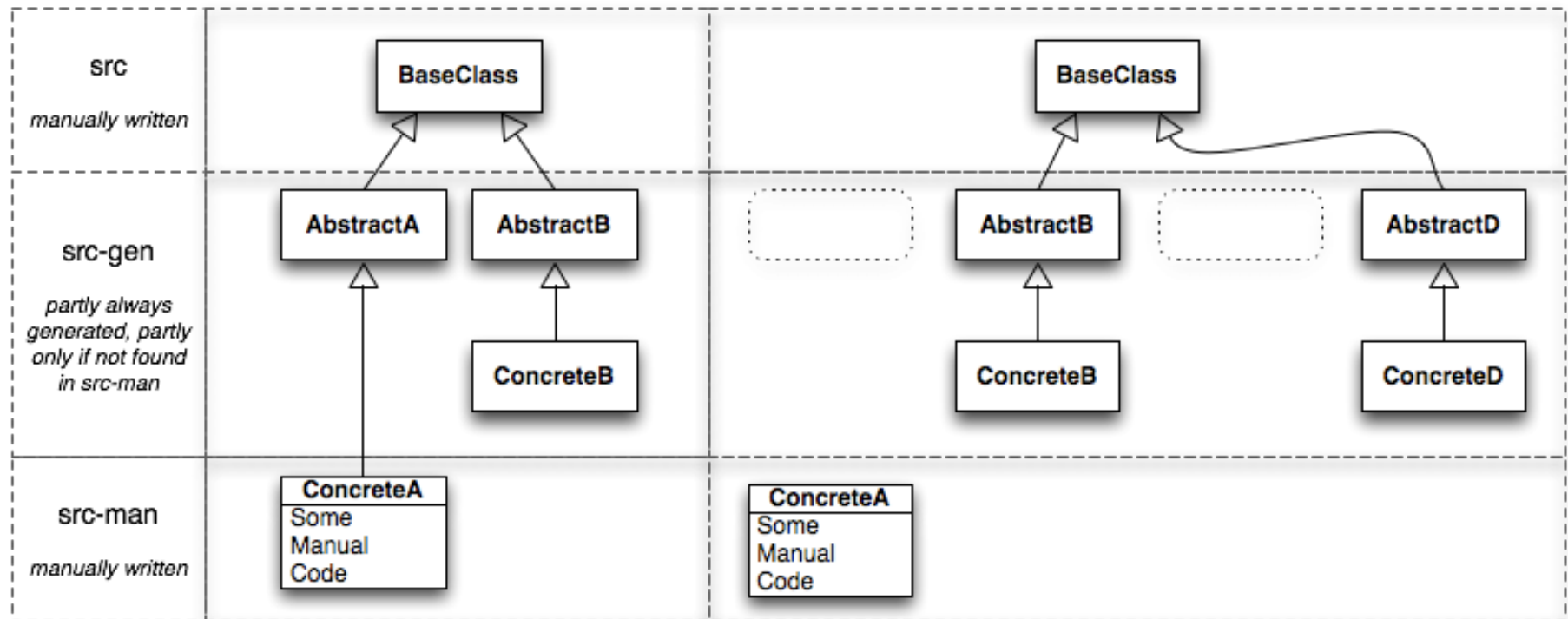
“Generate once”



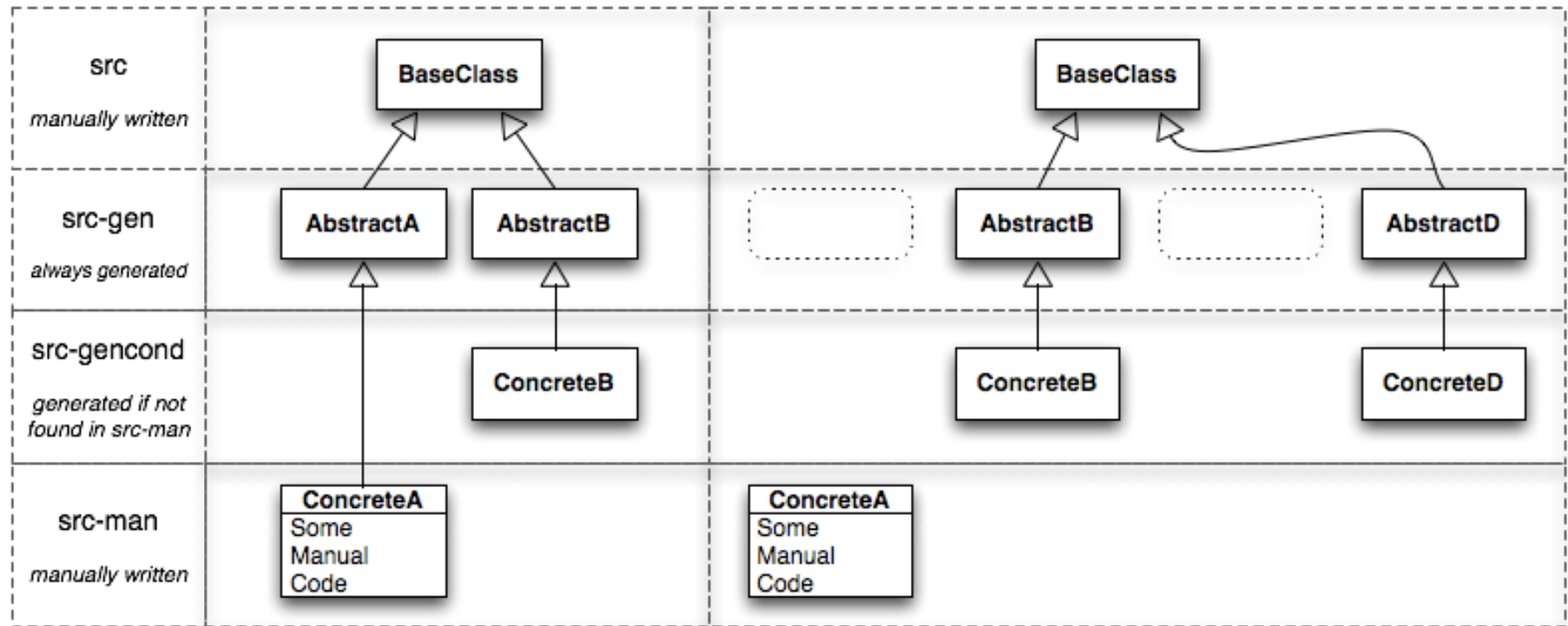
Generate once

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

Feltételes kódgenerálás – 1.



Feltételes kódgenerálás – 2.



Feltételes kódgenerálás

- Kódgenerálás
 - Ha nincs felülírva a kézi mappában
- Módosított Generation Gap minta
 - Pro: Kevesebb osztály, többször használható
 - Kontra: Nem örököljük a frissítéseket

Kódgenerálási stratégiák implementációja

■ 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!!!