> Another way is to keep only references to transformations in the feature model (this is what we did)

> The actual code of the transformations is in the corresponding files

> Parse the feature model configuration from the XML file

> Parse and execute the metatransformations contained within the transformations from the XML file

> Check the requirements of the parsed transformations

> Execute the transformations

> In the end, transformations are objects that share a common interface

```
public interface ITransformation {
  CheckPrerequisites();
  ExecuteTransformation();
  GetParameterNames();
  GetMetaTransformations();
  ...
}
```

> The generator takes these objects and executes the corresponding methods

> The transformations have to be executed in the right order: can be adjusted using the priority attribute

Challenges

> Assess the comprehensibility and maintainability of a software system expressed by a number of transformations: does this preserve the intent?

> Explore a non-exclusive application of the approach in which only some features would be realized as transformations

> How to deal with complex feature models consisting of several separate feature diagrams with references between them

> Foster the practical adoption of the approach by providing directly reusable transformations, transformation templates, and transformation schemas or examples

Summary

> Feature model driven generation of software artifacts

> Besides an enhanced feature model, no other models are necessary

> Transformations are not limited to affecting the software system

> The concept of metatransformations: modify the common transformations prior to their execution

> Challenges: comprehensibility and maintainability, non-exclusive application, complex feature models, practical adoption, and changes as transformations

vranic@stuba.sk
fiit.sk/~vranic

Feature Model Driven Generation of Software Artifacts

Roman Táborský and Valentino Vranić

Institute of Informatics and Software Engineering

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

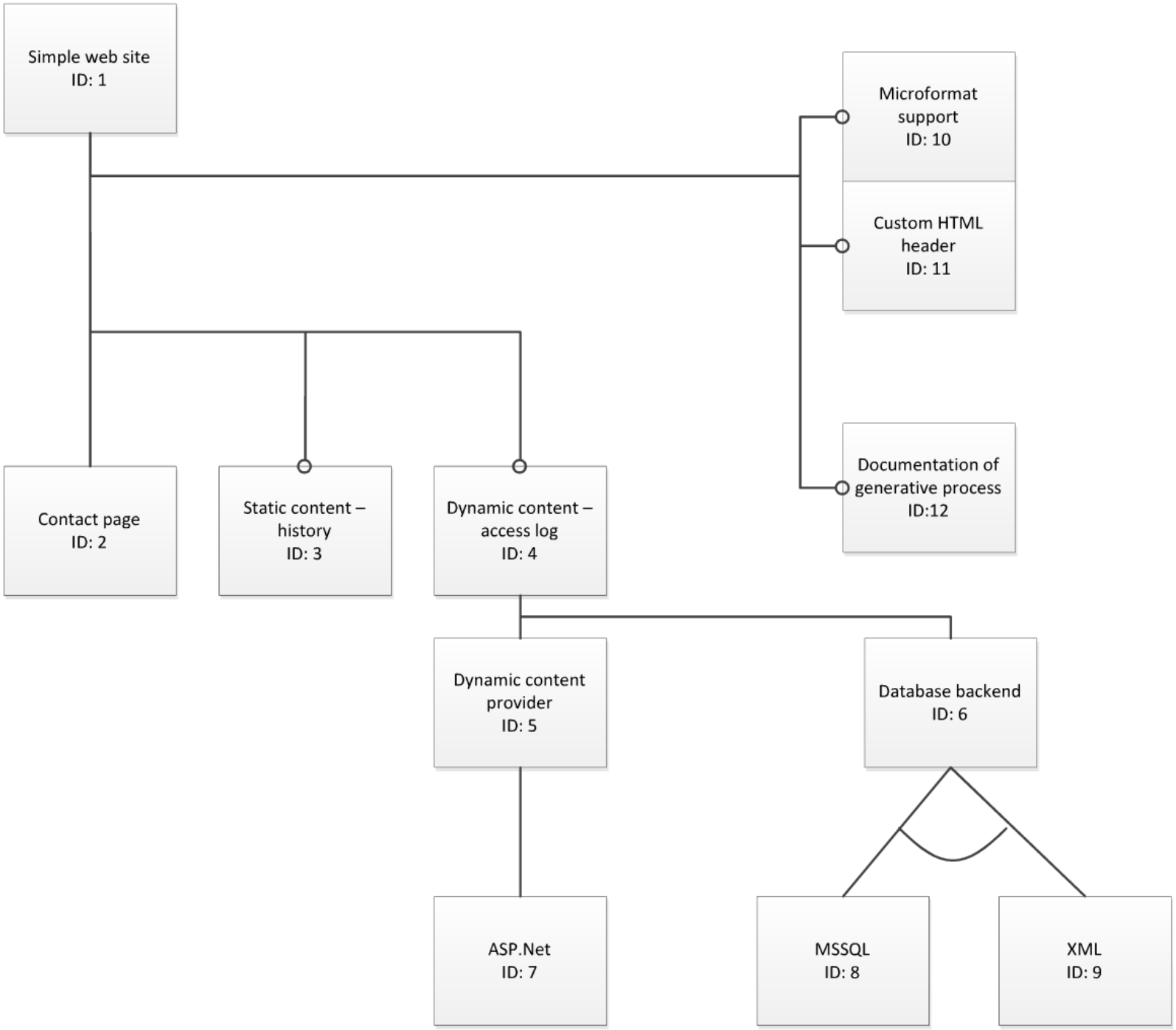
crudecrude@gmail.com

vranic@stuba.sk
fiit.sk/~vranic

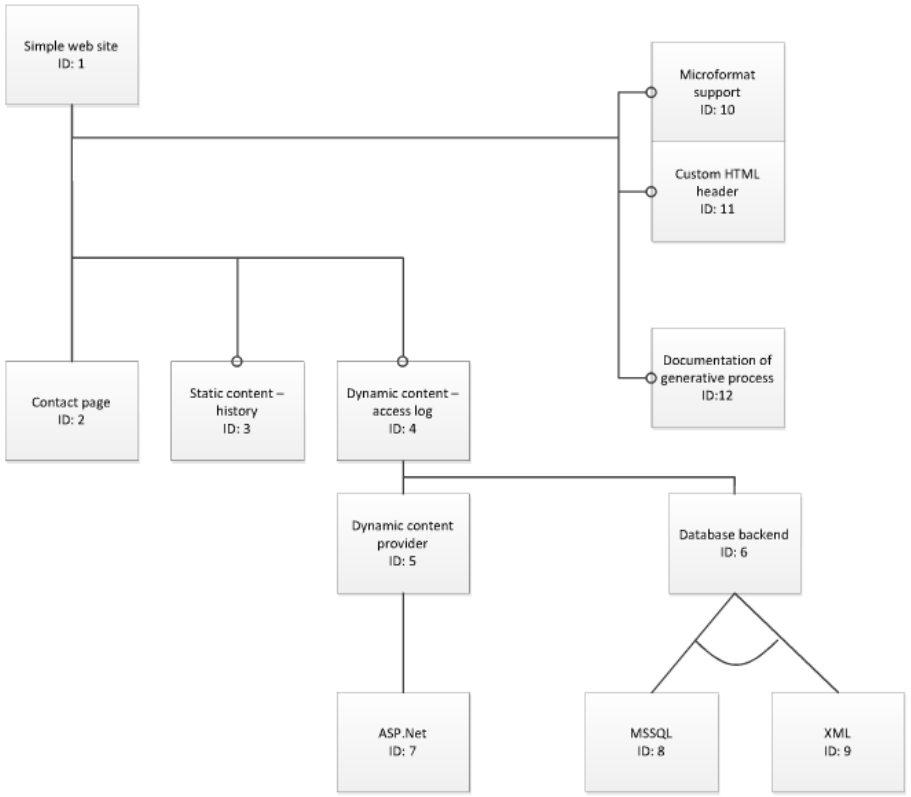
WAPL 2015 @ FedCSIS 2015

Łódź, September 13, 2015

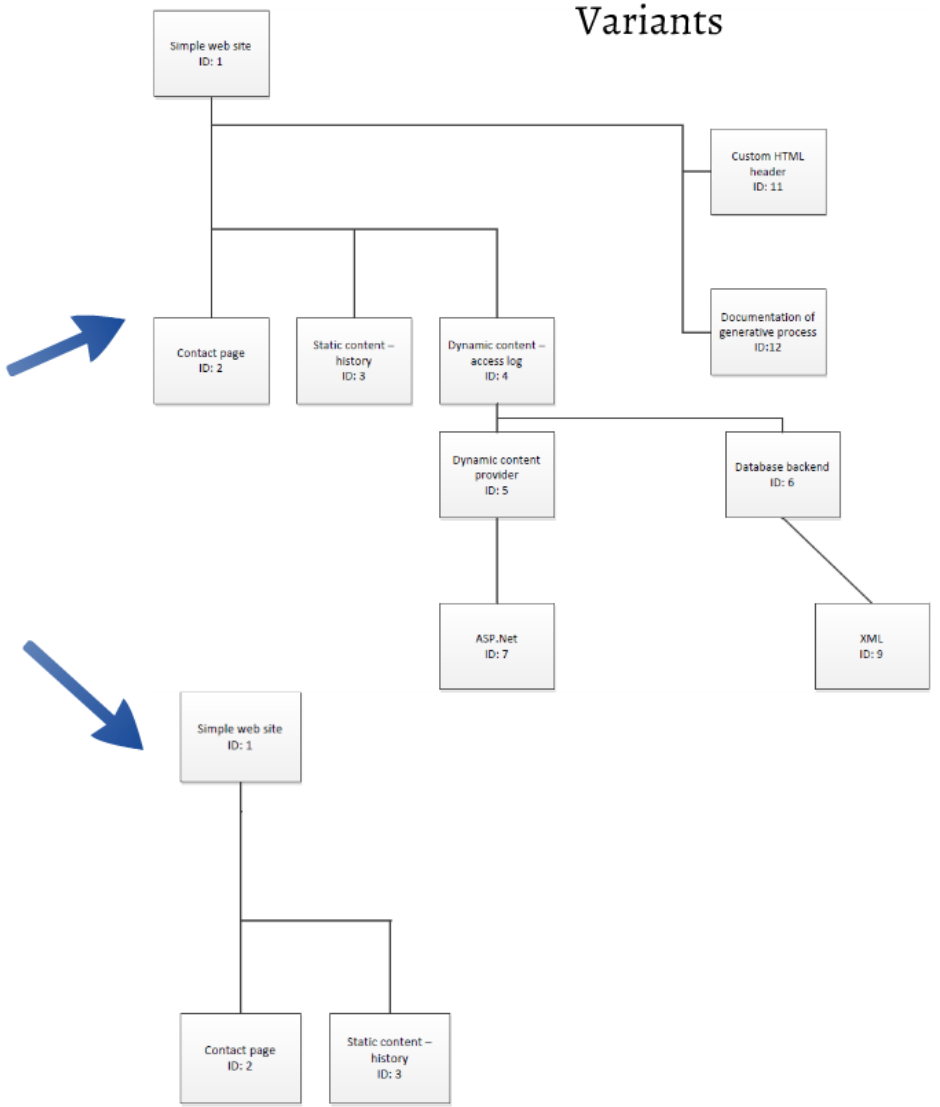
Feature Modeling



Feature Modeling



Variants



> A feature model defines variants in a structured way

> Domain engineering / software product lines

> Configurability first; appropriate implementation mechanisms selected accordingly

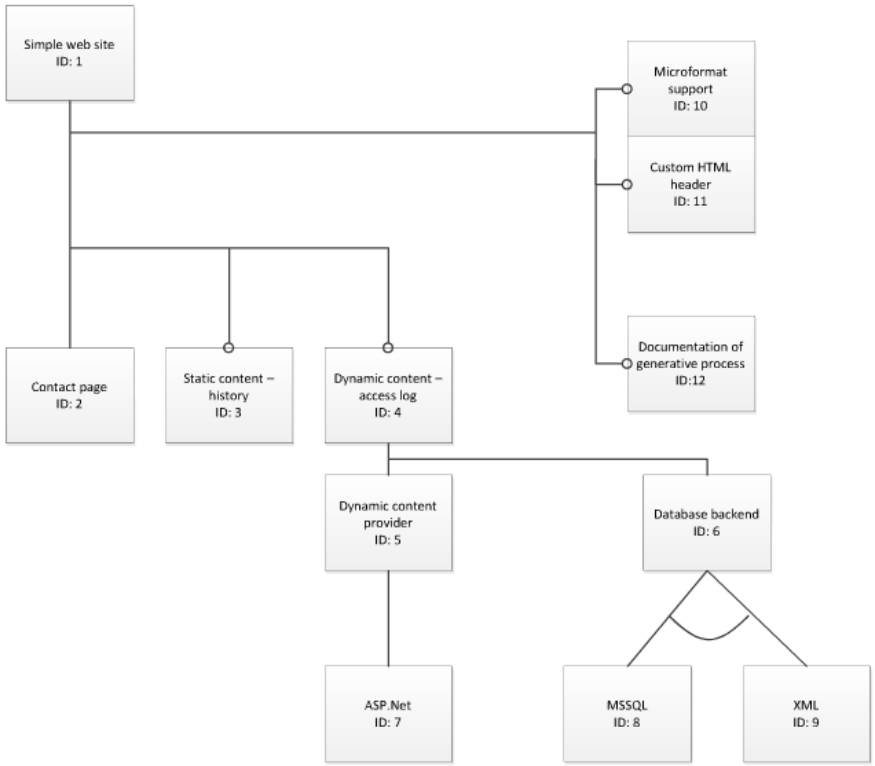
> Need not necessarily involve *any* academic feature modeling notation

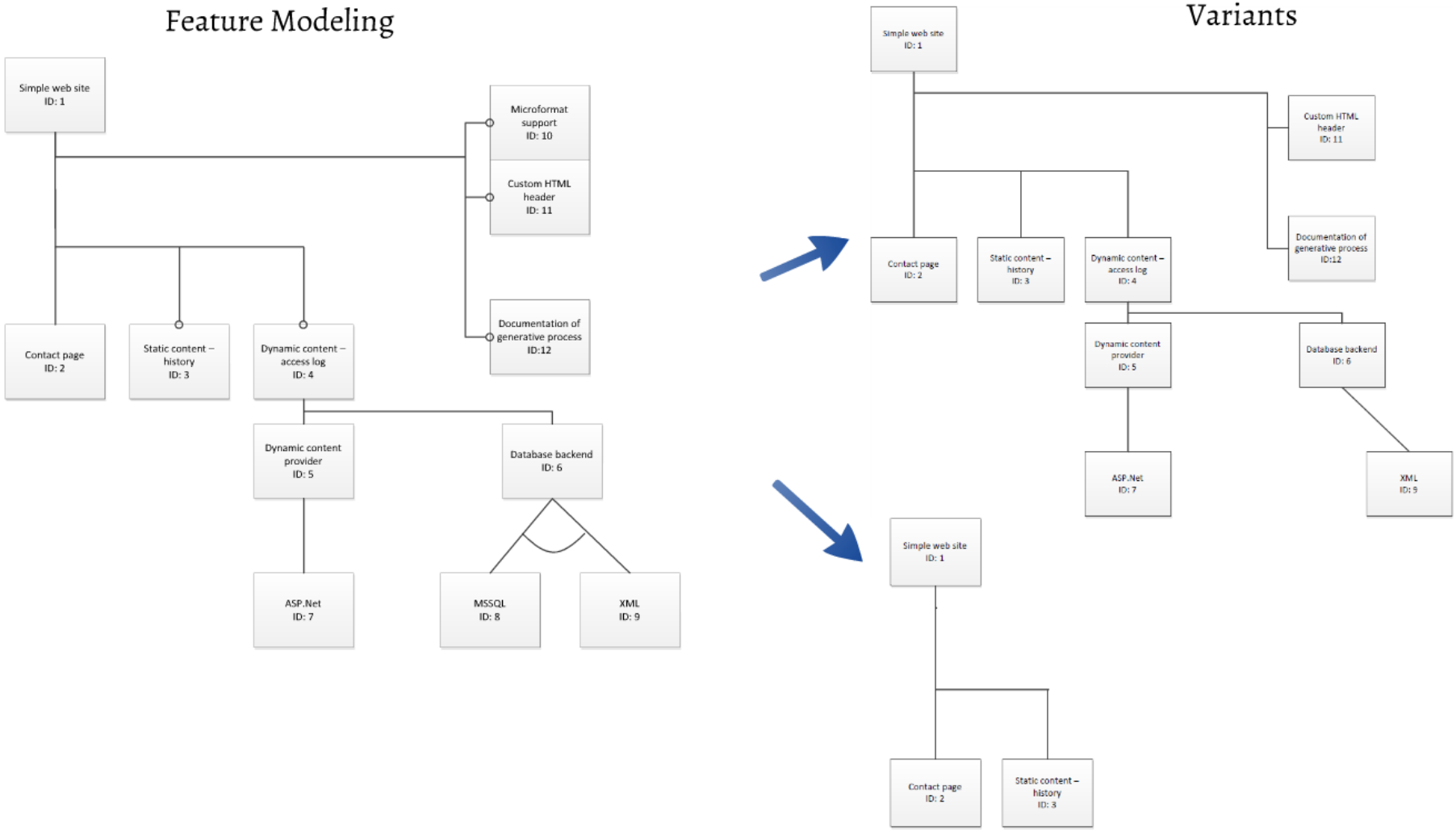
A. Hubaux, A. Classen, M. Mendonca, and P. Heymans: A Preliminary Review on the Application of Feature Diagrams in Practice, VaMoS 2010

> A simple FODA-like style used here

> Abstracting from advanced concepts like binding time, additional constraints among features, feature cardinality...

Feature Modeling





But how exactly to implement features?

What are they in code?

> Features sometimes can be implemented as well localized components

> But often the realization of one feature is spread over several components

> Sometimes this involves crosscutting: a case for aspect-oriented programming

> Features sometimes can be implemented as well localized components

> But often the realization of one feature is spread over several components

> Sometimes this involves crosscutting: a case for aspect-oriented programming

> A simple conditional compilation can cope with this, but the code of crosscutting features would be scattered

> Generative approaches rely on the set of selected features to generate highly specialized code

> Generators can be external, but also metaprogramming based

> In either case, features are understood as the structure and behavior to be added, adapted, or removed

> Our idea: make each feature's realization a transformation

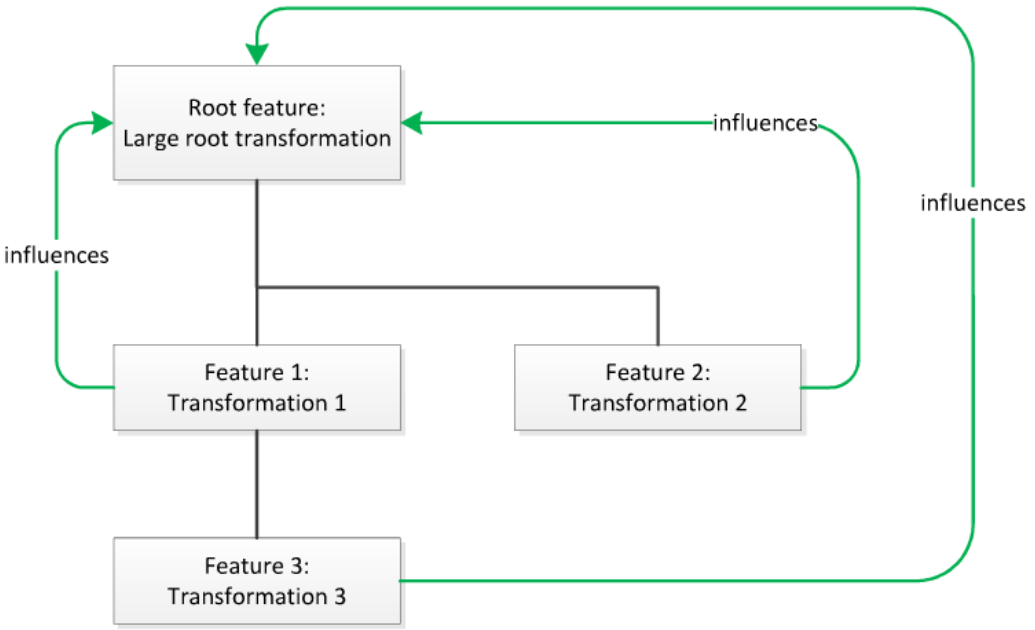
> Selecting a feature activates the corresponding transformation

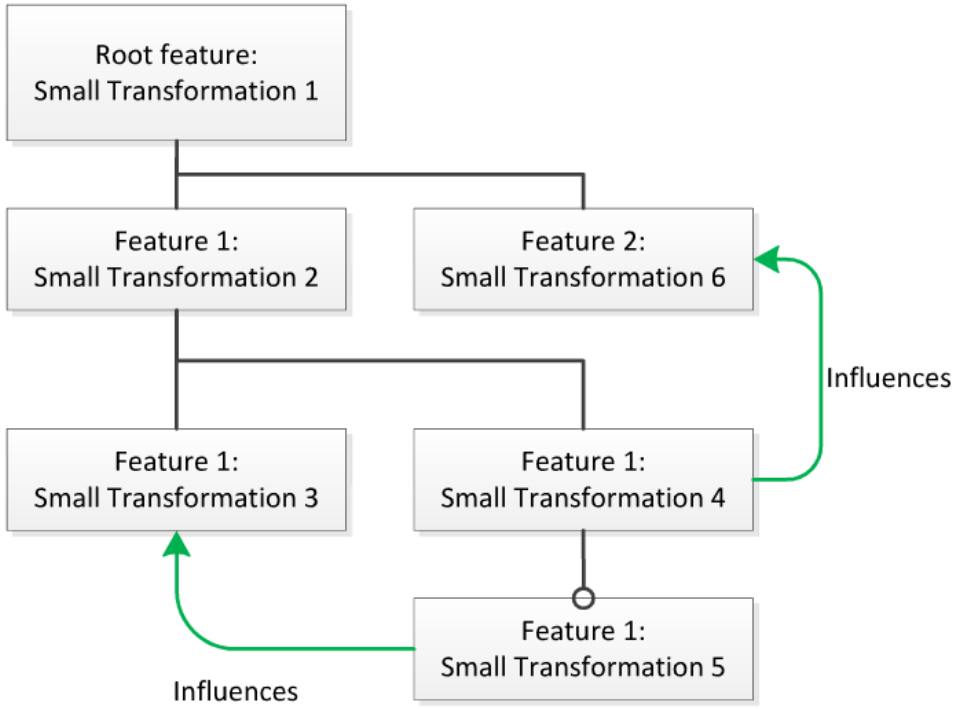
> A transformation can affect all code, including the other transformations

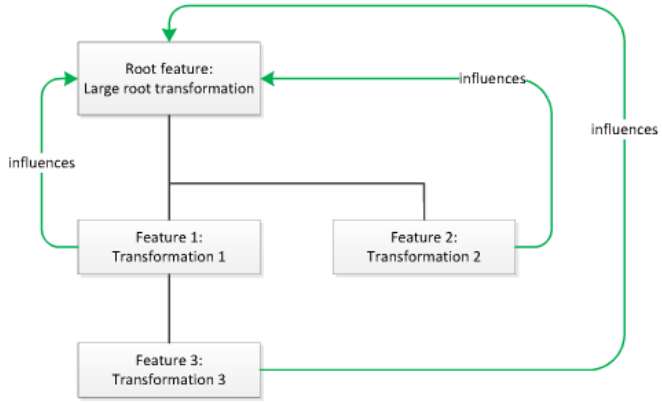
> Virtually, this is a distributed generator embedded into a feature model

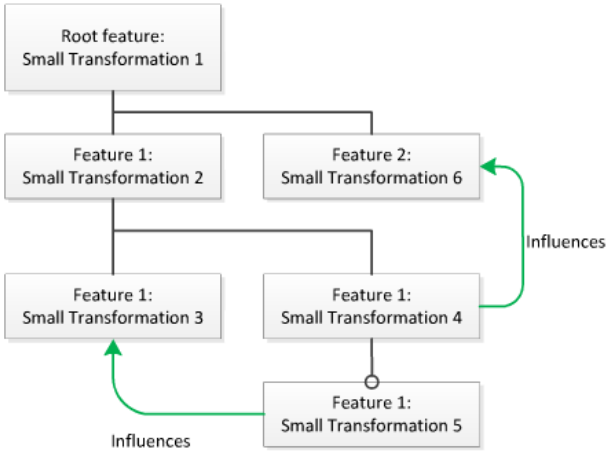
> The actual generator is then simple and generic

> This raises a number of issues...


Root feature:
Large root transformation
influences
influences
influences
Feature 1:
Transformation 1
Feature 2:
Transformation 2
Feature 3:
Transformation 3


Root feature:
Small Transformation 1
Feature 1:
Small Transformation 2
Feature 2:
Small Transformation 6
Influences
Feature 1:
Small Transformation 3
Feature 1:
Small Transformation 4
Feature 1:
Small Transformation 5
Influences





> Metatransformations

> Complexity of transformations

> Where to keep the transformations?

> One way is to attach them directly to the feature model that may conveniently be kept in the XML format

```
<feature>
  <feature>
    <transformation>
      <!-- Transformation information including
      the event chain, metatransformation
      information, requirements, etc. -->
    </transformation>
  </feature>
</feature>
```

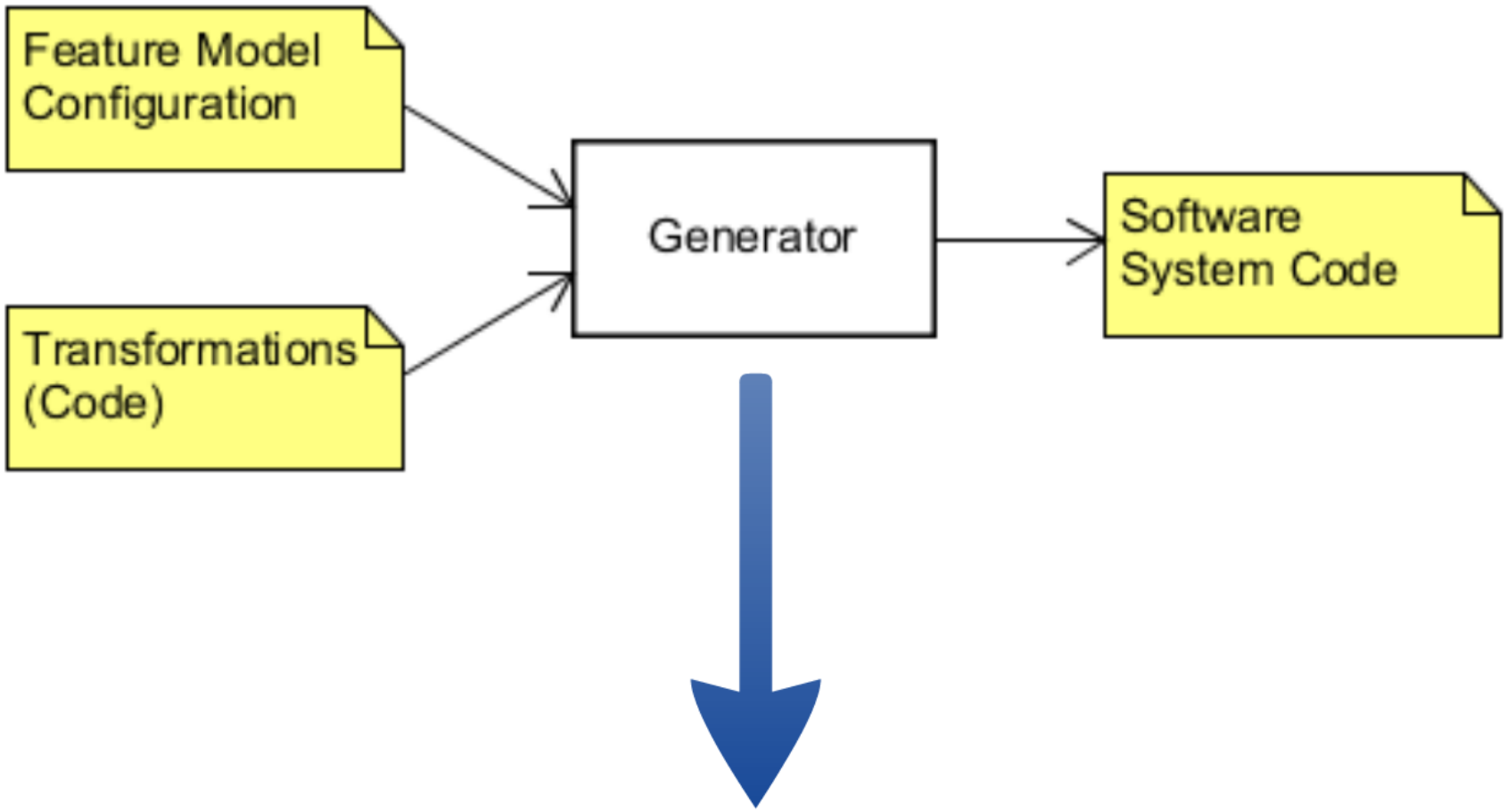
> Another way is to keep only references to transformations in the feature model (this is what we did)

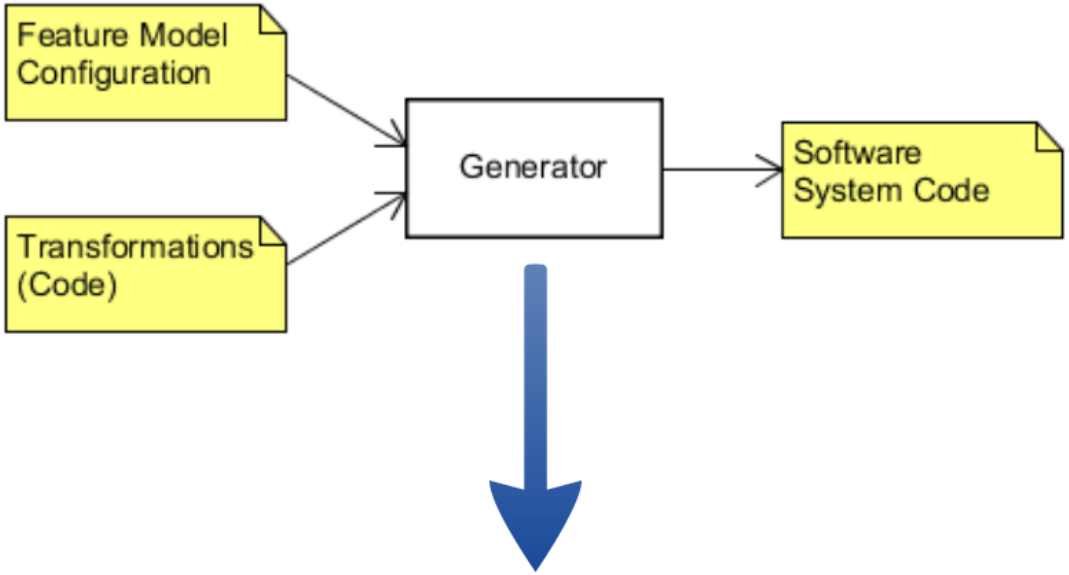
> The actual code of the transformations is in the corresponding files

```
<?xml version="1.0" encoding="utf-8" ?>
<featuremodel>
 <feature name="SimpleWeb" ID="1"
Transformation="org.crd.dp.CaseStudy.SimpleWebFinal,org.crd.dp.CaseStudy.SimpleWebFinal.Transformations.CreateFolder"
FolderName="Site" >
   <feature name="StaticContent-History" ID="2"
Transformation="org.crd.dp.CaseStudy.SimpleWebFinal,org.crd.dp.CaseStudy.SimpleWebFinal.Transformations.CreateStaticHTMLPage"
htmlContent="&lt;h1&gt;History&lt;/h1&gt;&lt;p&gt;Lorem Ipsum&lt;/p&gt;" htmlTitle="StaticPage- History" PageName="Site\\History.html" /
>
   <feature name="ContactPage" ID="3"

Transformation="org.crd.dp.CaseStudy.SimpleWebFinal,org.crd.dp.CaseStudy.SimpleWebFinal.Transformations.CreateStaticHTMLPage_Cre
ateContact_Composite"
        street="Ulica 29. Augusta 4" city="Bratislava" Folder="Site"/>

   <feature name="DynamicContent-AccessLog" ID="4"
Transformation="org.crd.dp.CaseStudy.SimpleWebFinal,org.crd.dp.CaseStudy.SimpleWebFinal.Transformations.DynamicContent"
htmlTitle="Dynamic page - Access log" PageName="Site\\log.html"/>
    <feature name="DynamicContentProvider" ID="5"
Transformation="org.crd.dp.CaseStudy.SimpleWebFinal,org.crd.dp.CaseStudy.SimpleWebFinal.Transformations.Empty" >
     <feature name="ASP.Net" ID="7"
Transformation="org.crd.dp.CaseStudy.SimpleWebFinal,org.crd.dp.CaseStudy.SimpleWebFinal.Transformations.ASP" />
    </feature>
    <feature name="DatabaseBackend" ID="6"
Transformation="org.crd.dp.CaseStudy.SimpleWebFinal,org.crd.dp.CaseStudy.SimpleWebFinal.Transformations.Empty" >
     <feature name="MSSQL" ID="8"
Transformation="org.crd.dp.CaseStudy.SimpleWebFinal,org.crd.dp.CaseStudy.SimpleWebFinal.Transformations.MSSQL" />
      <feature name="XML" ID="9"
Transformation="org.crd.dp.CaseStudy.SimpleWebFinal,org.crd.dp.CaseStudy.SimpleWebFinal.Transformations.XML" />
    </feature>
   <feature name="MicroformatSupport" ID="10"
Transformation="org.crd.dp.CaseStudy.SimpleWebFinal,org.crd.dp.CaseStudy.SimpleWebFinal.Transformations.MicroformatSupport"/>
   <feature name="CustomHTMLHeader" ID="11"
Transformation="org.crd.dp.CaseStudy.SimpleWebFinal,org.crd.dp.CaseStudy.SimpleWebFinal.Transformations.CustomHtmlHeaders"/>
   <feature name="GeneratorDoc" ID="12"
Transformation="org.crd.dp.CaseStudy.SimpleWebFinal,org.crd.dp.CaseStudy.SimpleWebFinal.Transformations.MetaDoc" />
 </feature>
</featuremodel>
```

Transformation="org.crd.dp.CaseStudy.SimpleWebFinal,org.crd.dp.CaseStudy.SimpleWebFinal.Transformations.CreateStatic
ateContact_Composite"
        street="Ulica 29. Augusta 4" city="Bratislava" Folder="Site"/>

  <feature name="DynamicContent-AccessLog" ID="4"
Transformation="org.crd.dp.CaseStudy.SimpleWebFinal,org.crd.dp.CaseStudy.SimpleWebFinal.Transformations.DynamicCon
htmlTitle="Dynamic page - Access log" PageName="Site\\log.html"/>
    <feature name="DynamicContentProvider" ID="5"
Transformation="org.crd.dp.CaseStudy.SimpleWebFinal,org.crd.dp.CaseStudy.SimpleWebFinal.Transformations.Empty" >
     <feature name="ASP.Net" ID="7"
Transformation="org.crd.dp.CaseStudy.SimpleWebFinal,org.crd.dp.CaseStudy.SimpleWebFinal.Transformations.ASP" />
    </feature>
    <feature name="DatabaseBackend" ID="6"
Transformation="org.crd.dp.CaseStudy.SimpleWebFinal,org.crd.dp.CaseStudy.SimpleWebFinal.Transformations.Empty" >
     <feature name="MSSQL" ID="8"
Transformation="org.crd.dp.CaseStudy.SimpleWebFinal,org.crd.dp.CaseStudy.SimpleWebFinal.Transformations.MSSQL" />
     <feature name="XML" ID="9"
Transformation="org.crd.dp.CaseStudy.SimpleWebFinal,org.crd.dp.CaseStudy.SimpleWebFinal.Transformations.XML" />
    </feature>
  <feature name="MicroformatSupport" ID="10"
Transformation="org.crd.dp.CaseStudy.SimpleWebFinal,org.crd.dp.CaseStudy.SimpleWebFinal.Transformations.Microformat
  <feature name="CustomHTMLHeader" ID="11"
Transformation="org.crd.dp.CaseStudy.SimpleWebFinal,org.crd.dp.CaseStudy.SimpleWebFinal.Transformations.CustomHtml
  <feature name="GeneratorDoc" ID="12"
Transformation="org.crd.dp.CaseStudy.SimpleWebFinal,org.crd.dp.CaseStudy.SimpleWebFinal.Transformations.MetaDoc" />
 </feature>
</featuremodel>


Feature Model
Configuration
Transformations
(Code)
Generator
Software
System Code



> Parse the feature model configuration from the XML file

> Parse and execute the metatransformations contained within the transformations from the XML file

> Check the requirements of the parsed transformations

> Execute the transformations

> In the end, transformations are objects that share a common interface

```
public interface ITransformation {
   CheckPrerequisites();
   ExecuteTransformation();
   GetParameterNames();
   GetMetaTransformations();
   ...
}
```

> The generator takes these objects and executes the corresponding methods

> The transformations have to be executed in the right order: can be adjusted using the priority attribute

Challenges

> Assess the comprehensibility and maintainability of a software system expressed by a number of transformations: does this preserve the intent?

> Explore a non-exclusive application of the approach in which only some features would be realized as transformations

> How to deal with complex feature models consisting of several separate feature diagrams with references between them

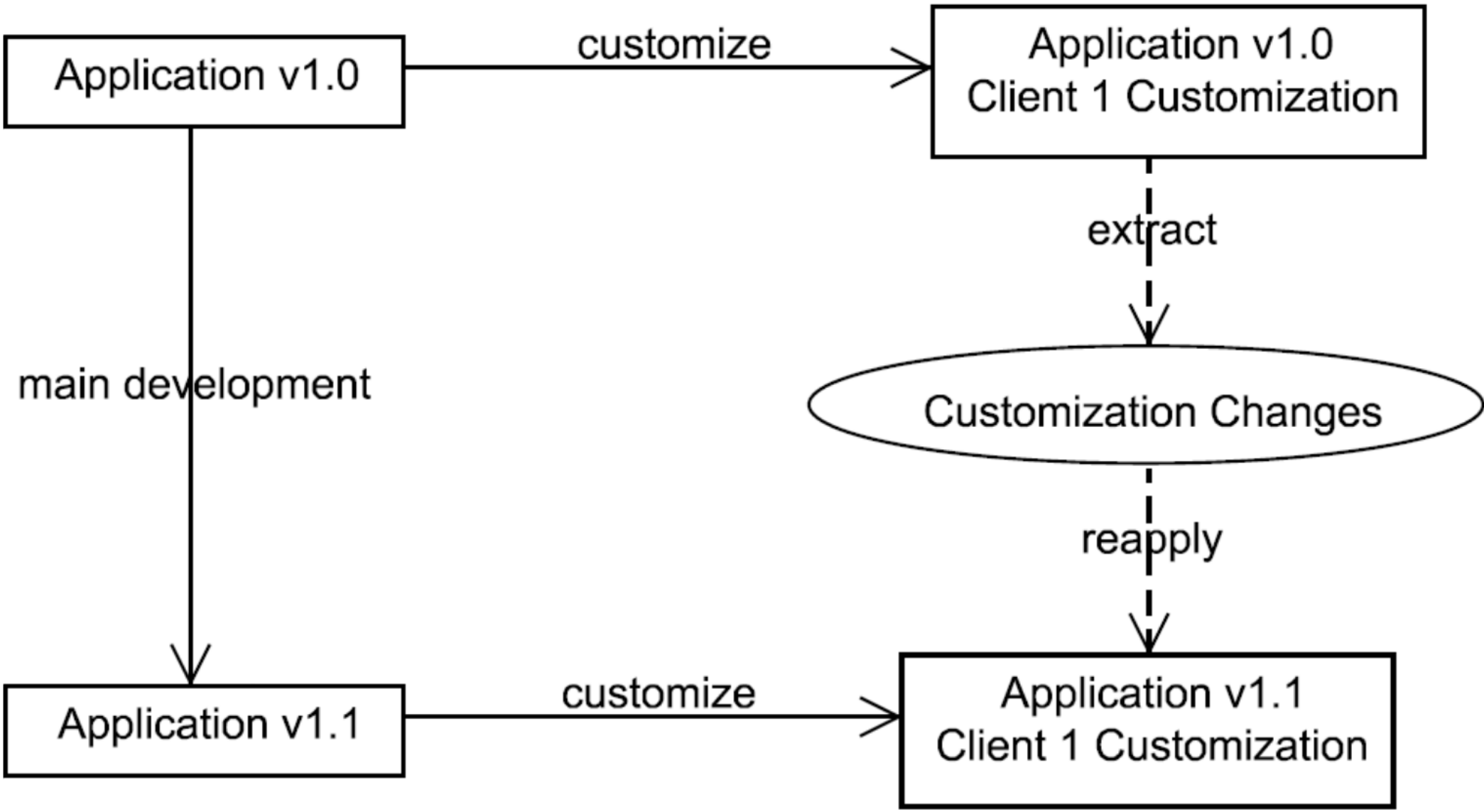
> Foster the practical adoption of the approach by providing directly reusable transformations, transformation templates, and transformation schemes or examples

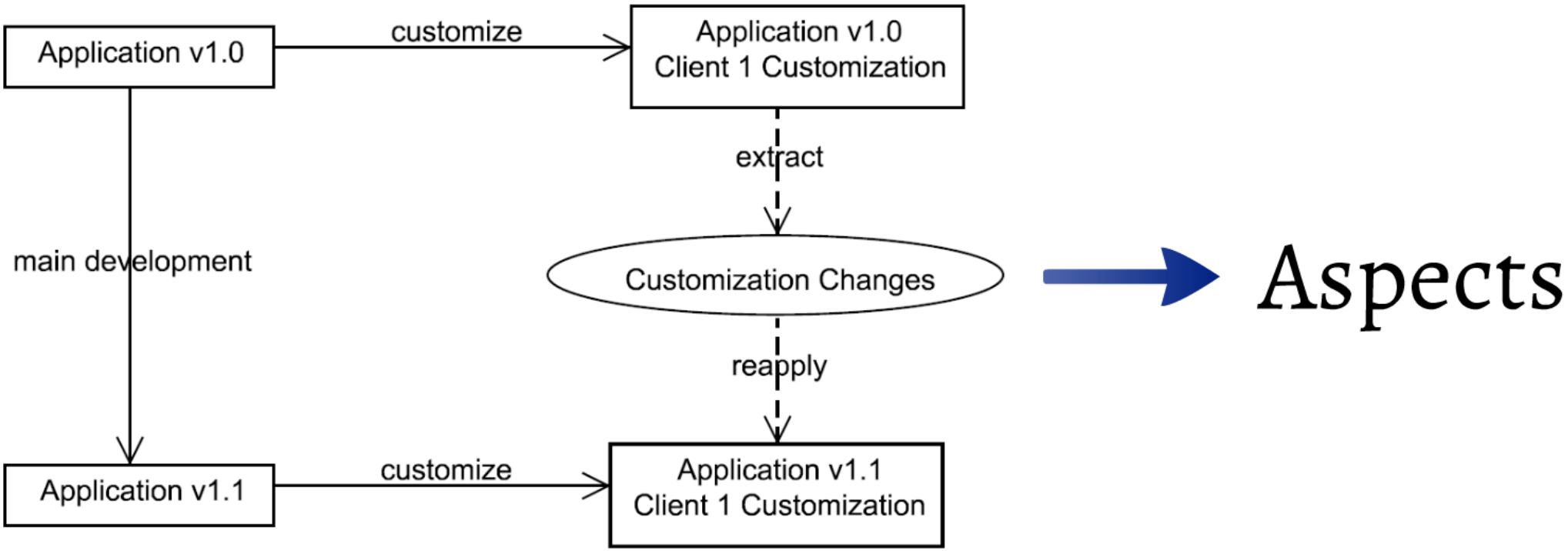
> Digression:

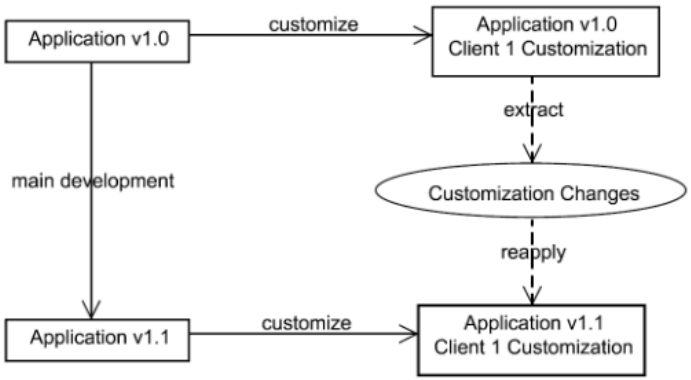
Transformations as changes

> Aspect-oriented change realization

V. Vranić. Aspect-Oriented Change Realization. Habilitation thesis, Slovak University of Technology in Bratislava, April 2010. http://fiit.sk/~vranic/pub/AOCR.pdf

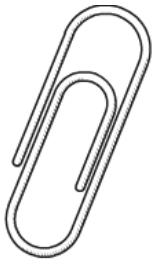

Application v1.0
customize
Application v1.0
Client 1 Customization
extract
main development
Customization Changes
reapply
Application v1.1
customize
Application v1.1
Client 1 Customization


Application v1.0
customize
Application v1.0
Client 1 Customization
extract
main development
Customization Changes
Aspects
reapply
Application v1.1
customize
Application v1.1
Client 1 Customization



Aspects

Features
Transformations



Summary

> Feature model driven generation of software artifacts

> Besides an enhanced feature model, no other models are necessary

> Transformations are not limited in affecting the software system

> The concept of metatransformations: modify the common transformations prior to their execution

> Challenges: comprehensibility and maintainability, non-exclusive application, complex feature models, practical adoption, and changes as transformations

vranic@stuba.sk
fiit.sk/~vranic