50. Round-Trip Engineering for the Consistency of Megamodels

Prof. Dr. U. Aßmann

Technische Universität Dresden

Institut für Software- und Multimediatechnik

http://st.inf.tu-dresden.de/teaching/most

Version 15-0.5, 30.01.16

1)Single-source principle

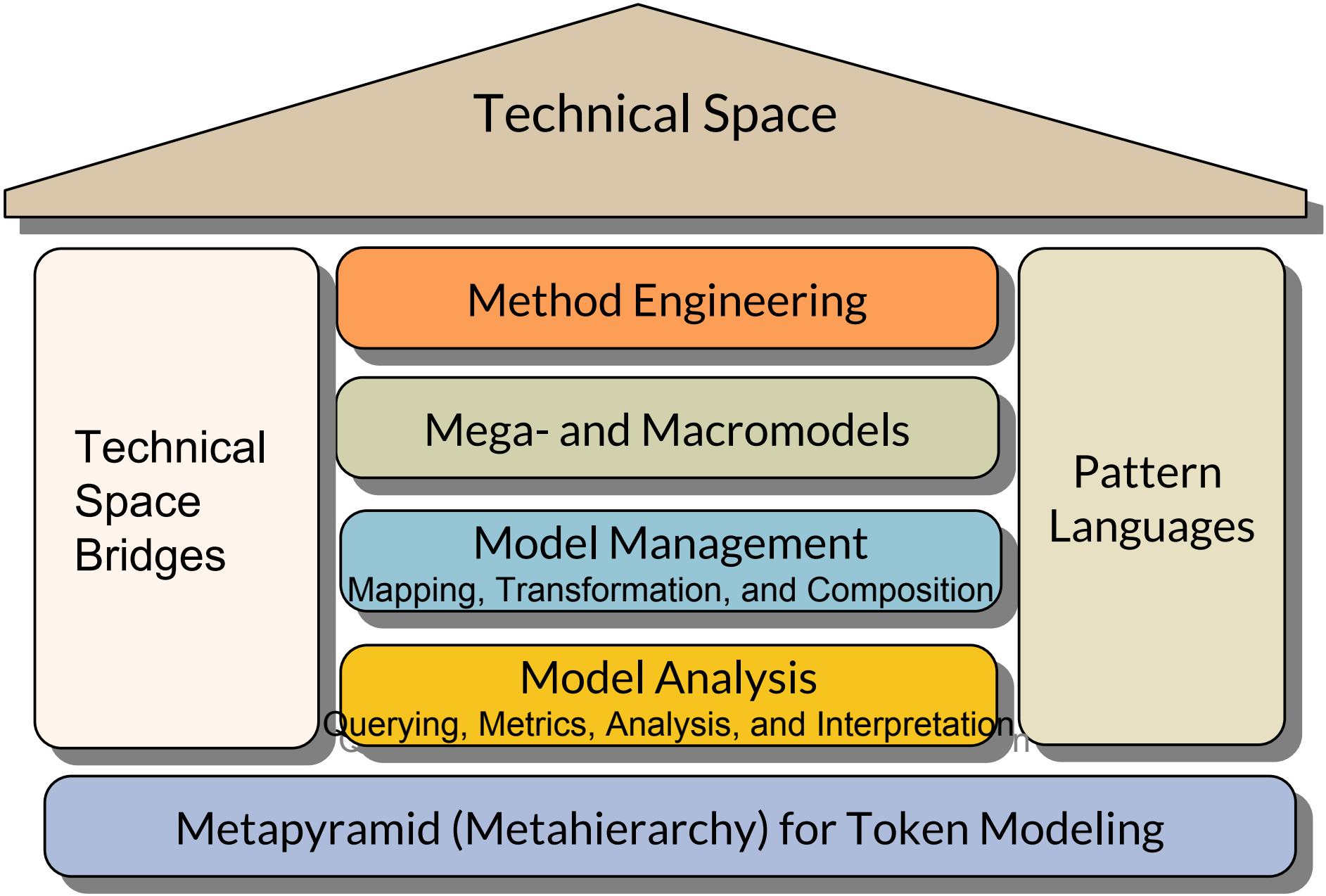
2)Code generation techniques

- Template-based Code generation

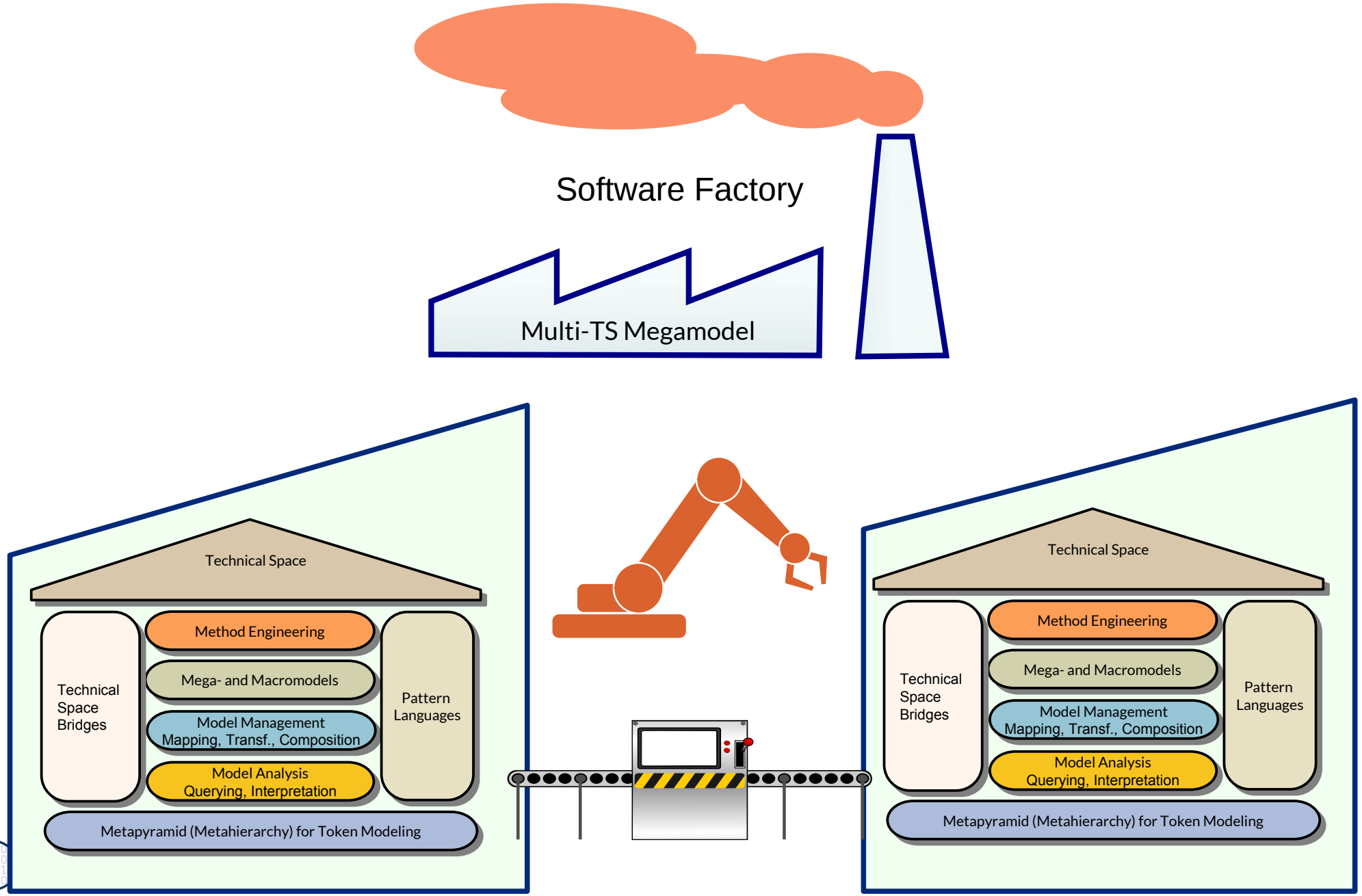
3)Re-parsing

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

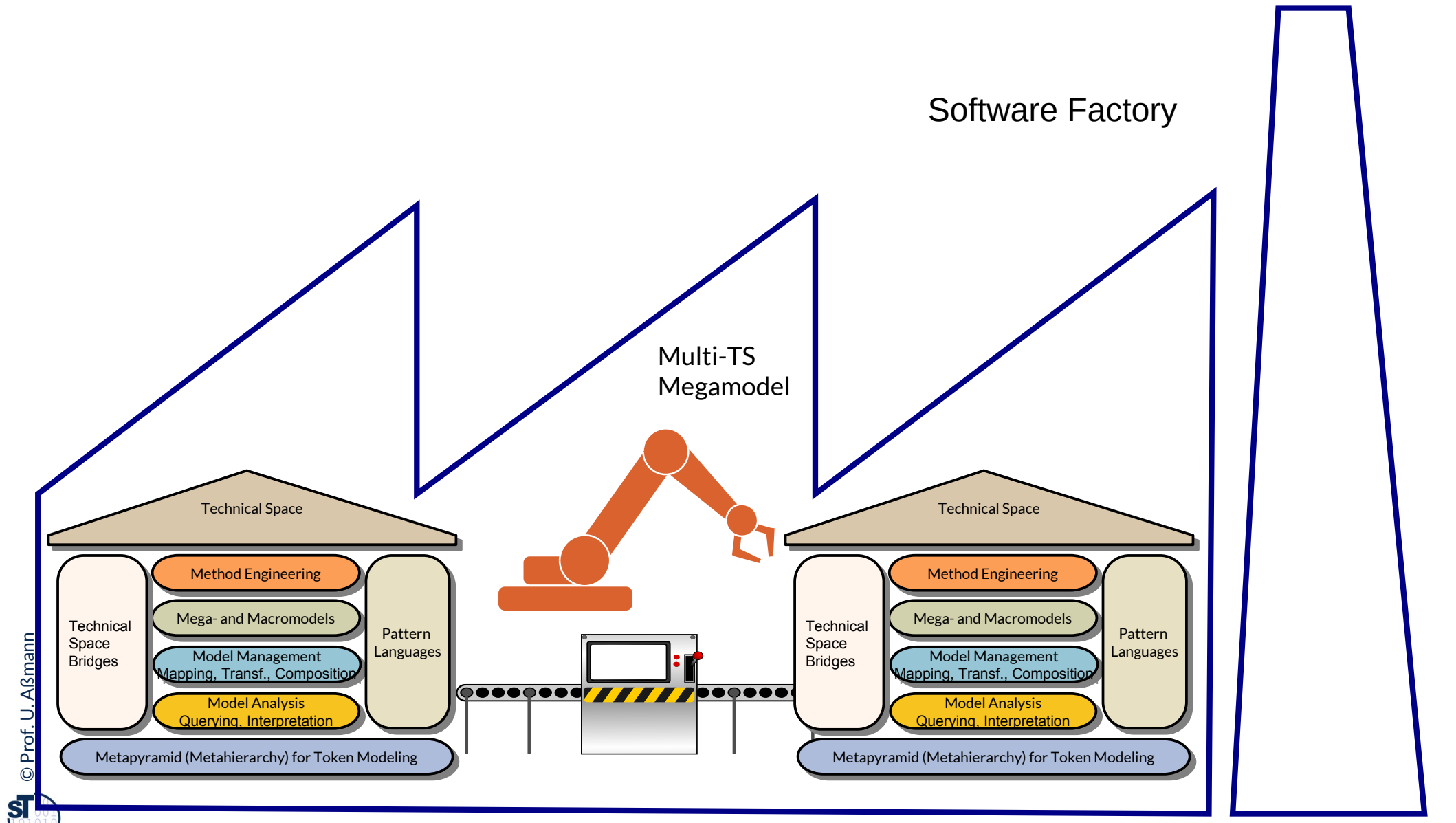
Q10: The House of a Technical Space



Q11: A Software Factory's Heart: the Multi-TS Megamodel



Q12: A Software Factory's Heart: the M[illegible]egamodel



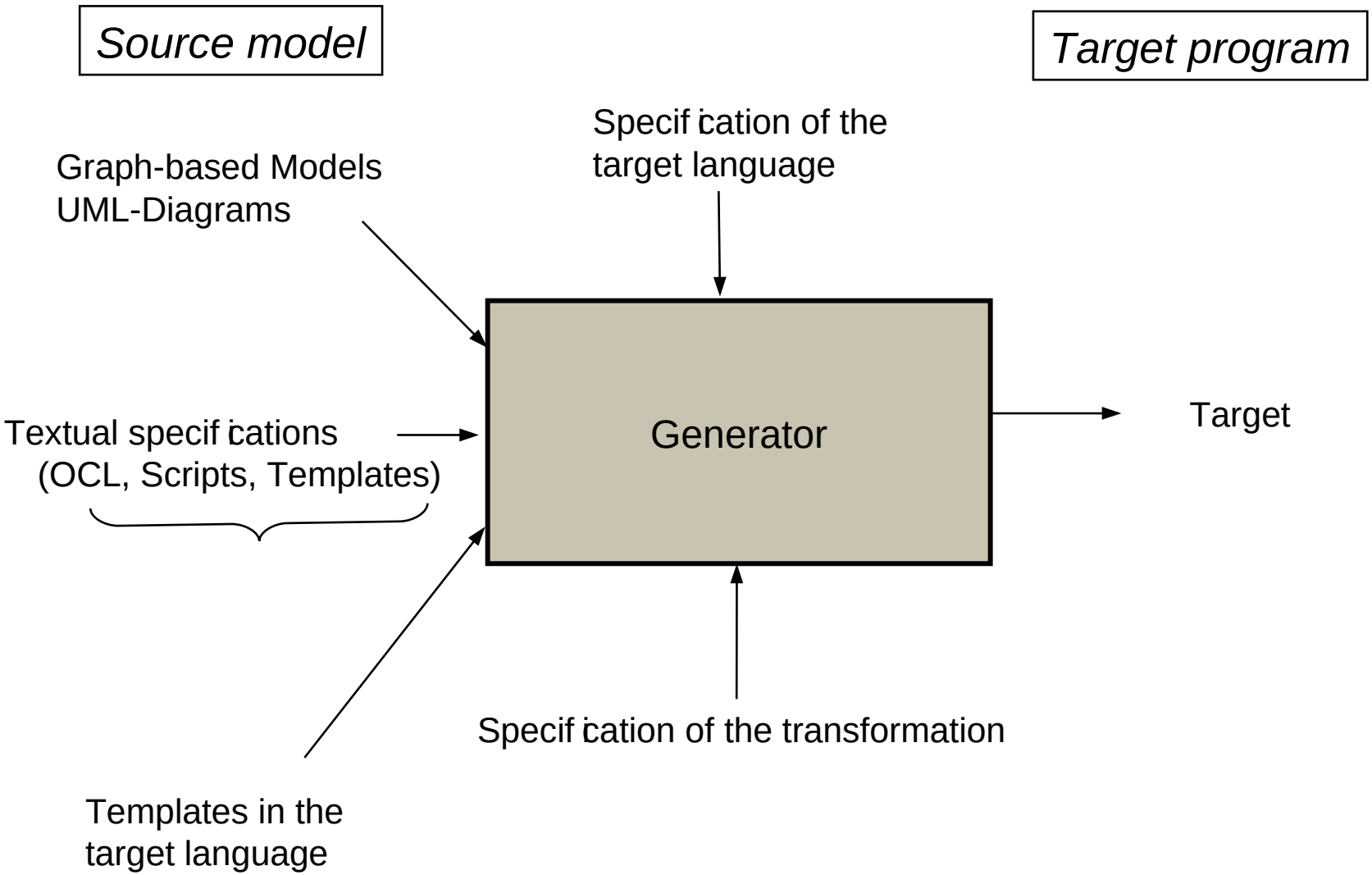
Literature

- http://www.codegeneration.net/
- www.programtransformation.org
- http://www.codegeneration.net/tiki-read_article.php?articleId=65
- Paul Bassett. Frame-based software engineering. IEEE Software, 4(4):9-16, 1987.
  - http://doi.ieeecomputersociety.org/10.1109/MS.1987.231057
- Chris Holmes, Andy Evans. A review of frame technology. University of York, Dept. of Computer Science, 2003
  ftp://www-users.cs.york.ac.uk/reports/2003/YCS/369/YCS-2003-369.pdf
- Daniel Weise and Roger Crew. Programmable syntax macros. In Proceedings of the ACM SIGPLAN '93 Conference on Programming Language Design and Implementation, pages 156-165, Albuquerque, New Mexico, June 23-25, 1993.
- Optional
  - Völter, Stahl: Model-Driven Software Development, AWL 2005.

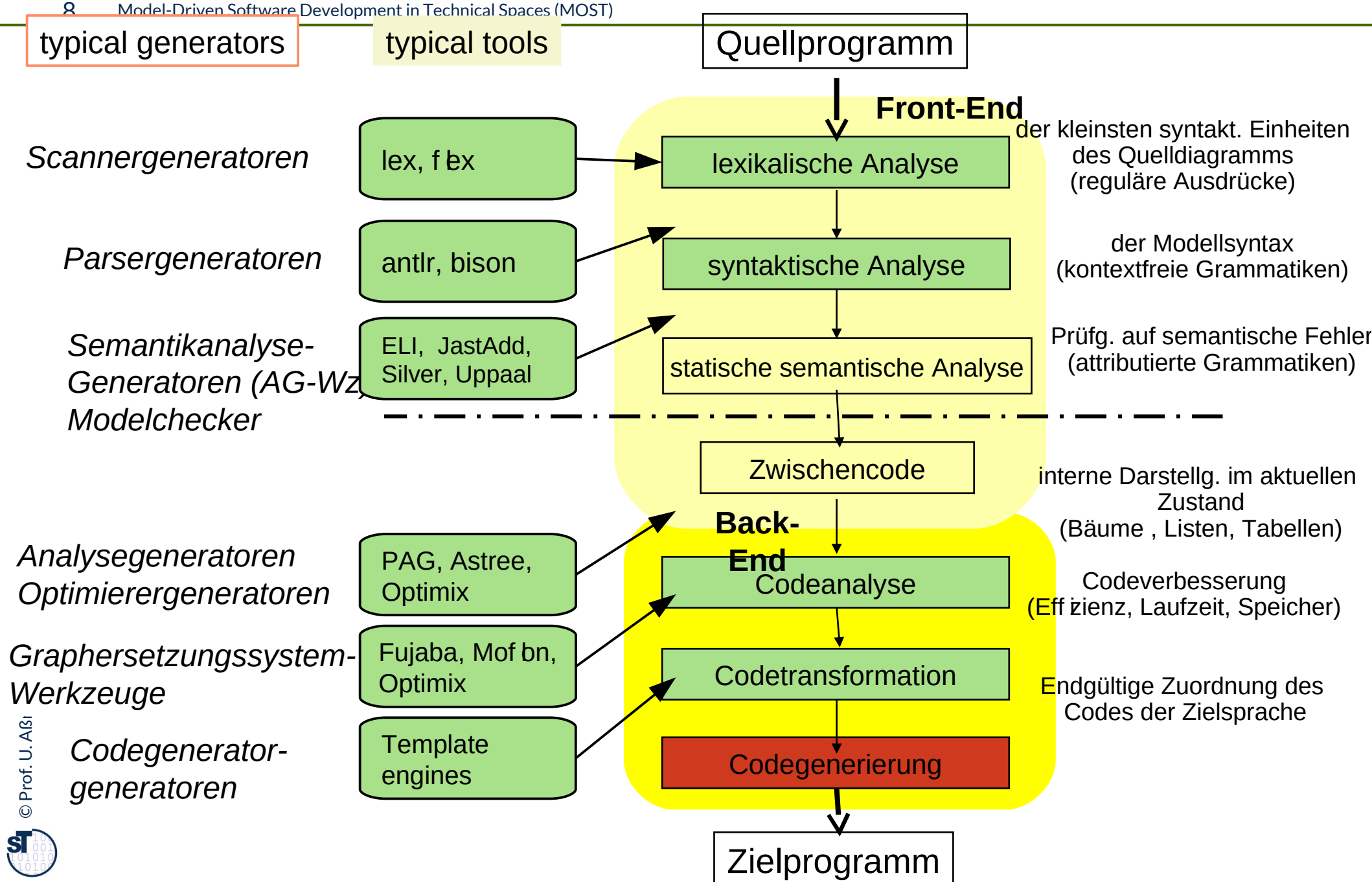
50.1 Model2Code Translation (Code Generation)

Transforming models into code (Programmüberführung)

CASE-Code-Generators



Phases of a Tool and its Generators

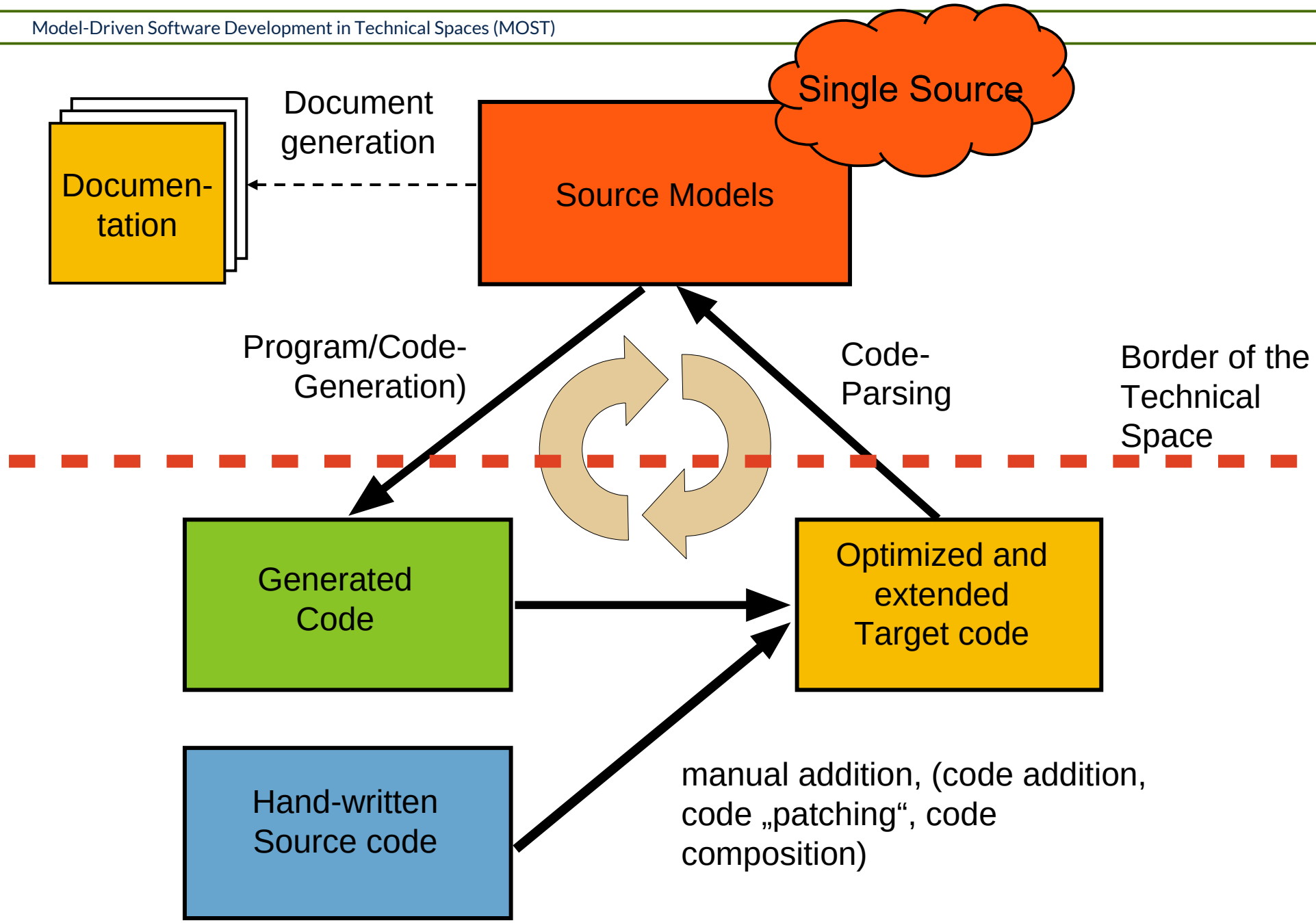


Kinds of Code Generators

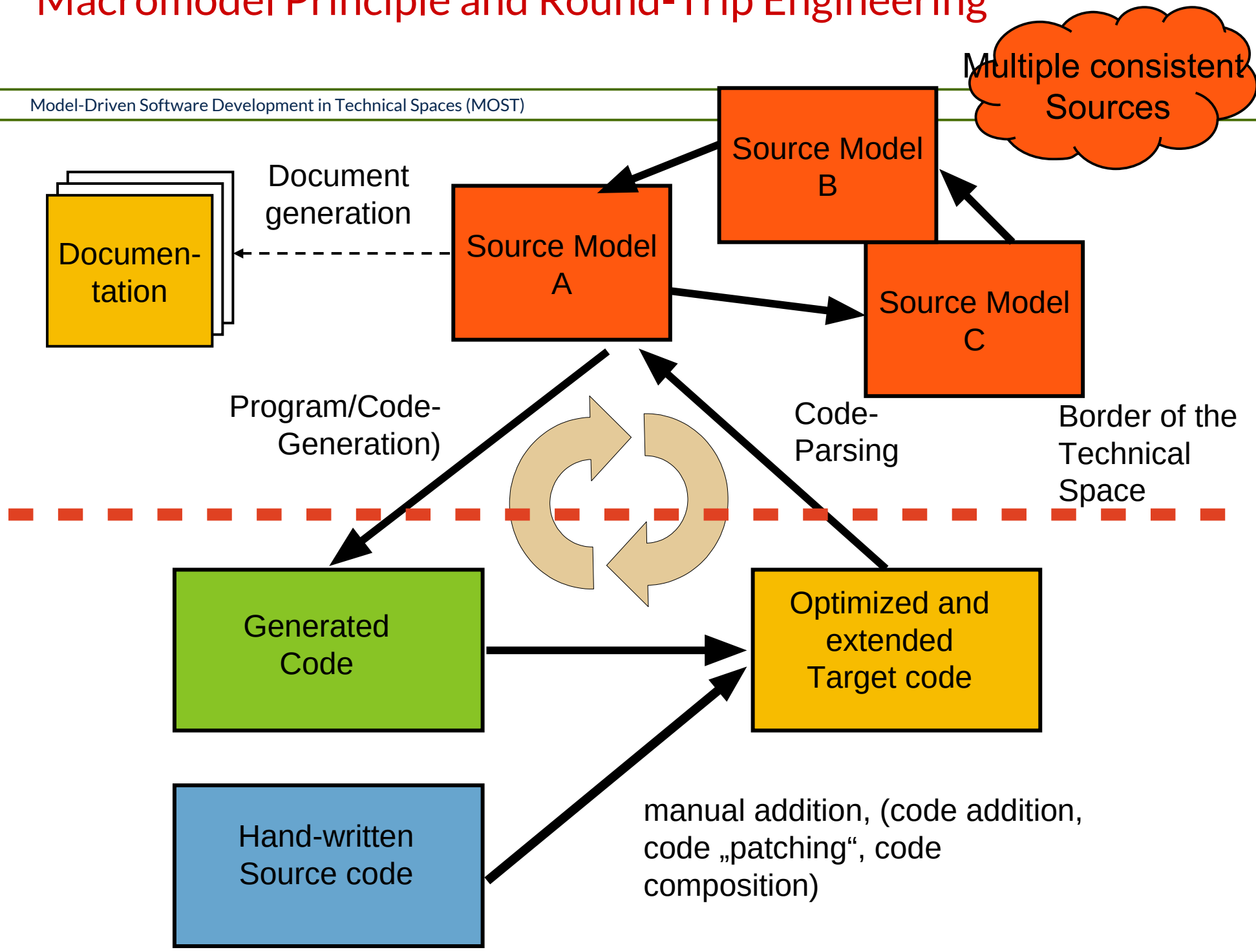
- A **code selector** is a transformation system (term, link trees, graphs) covering the input models with rules (**code coverage**) transforming the model elements once
- A **code scheduler** orders instructions in an optimized manner
  - Code scheduling runs after code selection
- **Metaprogramming code generators**:
  - A **template expander** generates code by filling code templates with *inset snippets*
  - An **invasive fragment composer (invasive software composition)** composes templates in a typed and wellformed way (CBSE)

50.1.1 Single-Source Principle and Macromodels

Single-Source-Principle, Code Addition, and Round-Trip Engineering



Macromodel Principle and Round-Trip Engineering



Single Source Principle and Macromodel Principle

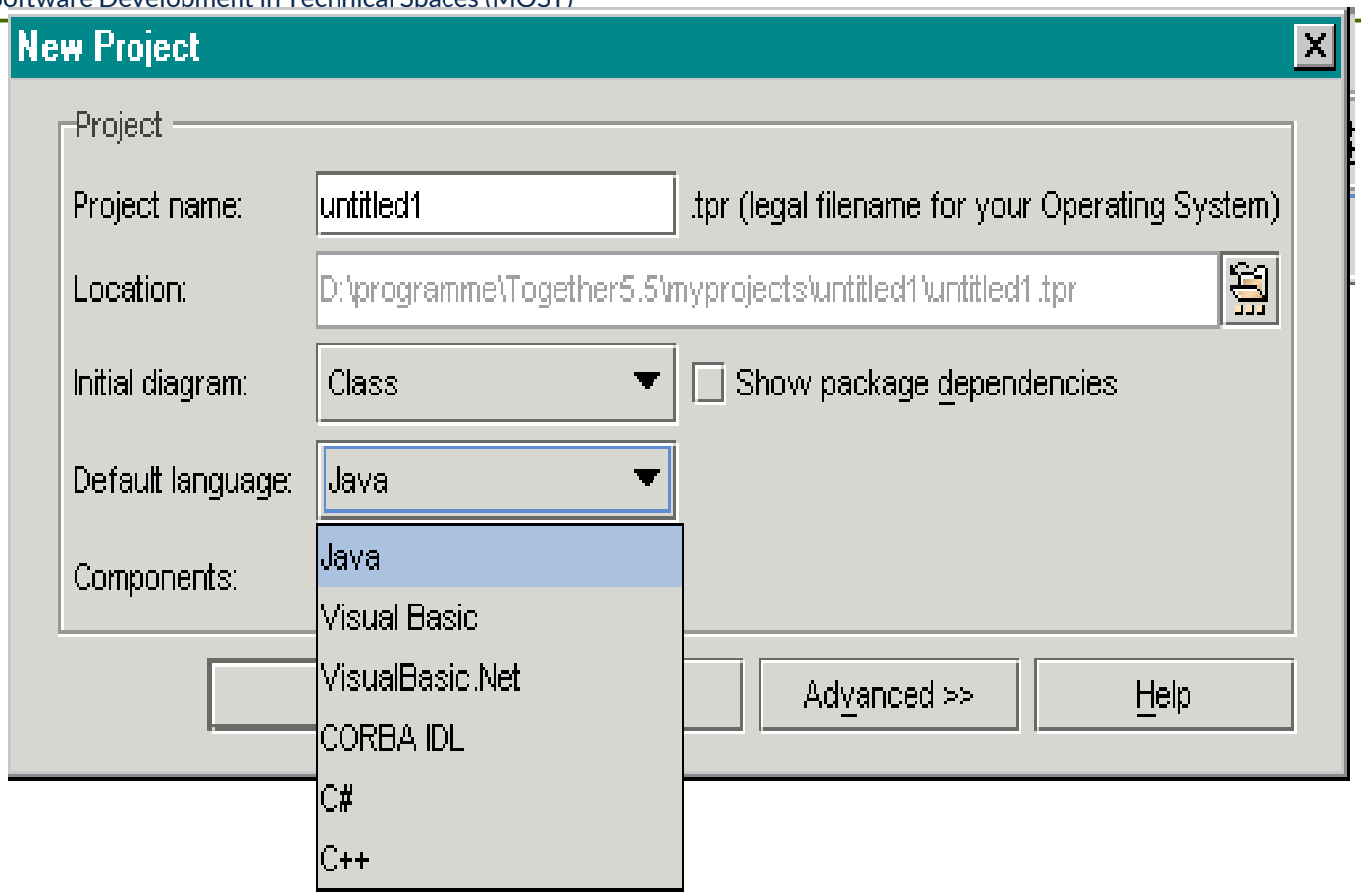
- A **Single-Source-Technology** with automatic synchronisation and consistency between one model (single source), code, tests
  - 1997 introduced by Peter Coad in the Together-CASE-tool, now in all CASE tools
- A **Macromodel technology** with automatic synchronisation and consistency between ALL models, code, tests, and documentation (all models of a megamodel)
- Technically, the Single-Source-Principle and the Macromodel principle needs **Round-Trip-Engineering** (RTE) between ModelWare and GrammarWare, to achieve
  - **Model-to-code synchronisation** with
    - **Codegeneration** into several programming languages
    - **Template-based codegeneration** inserts code snippets itno code templates
    - **Code reparsing** of the changed source code into models
  - **Model-to-model synchronization** with
    - **Bidirectional transformations** (with TGG)
    - **View based transformations** (with SUM)

Round-Trip Engineering in Together (Coad)

- Single-Source-Technology with automatic synchronisation and consistency between UML model, code and documentation
- Supported Programming Languages: Java, Visual Basic, VisualBasic.Net, CORBA IDL, C++, C#
  - Synchronisation by reparsing of generated, modified and extended code
- Round-trip Engineering:
  - Changes of class diagrams will be transformed to code
  - Changes of code reparsed to class diagrams
  - Reverse Engineering of entire projects

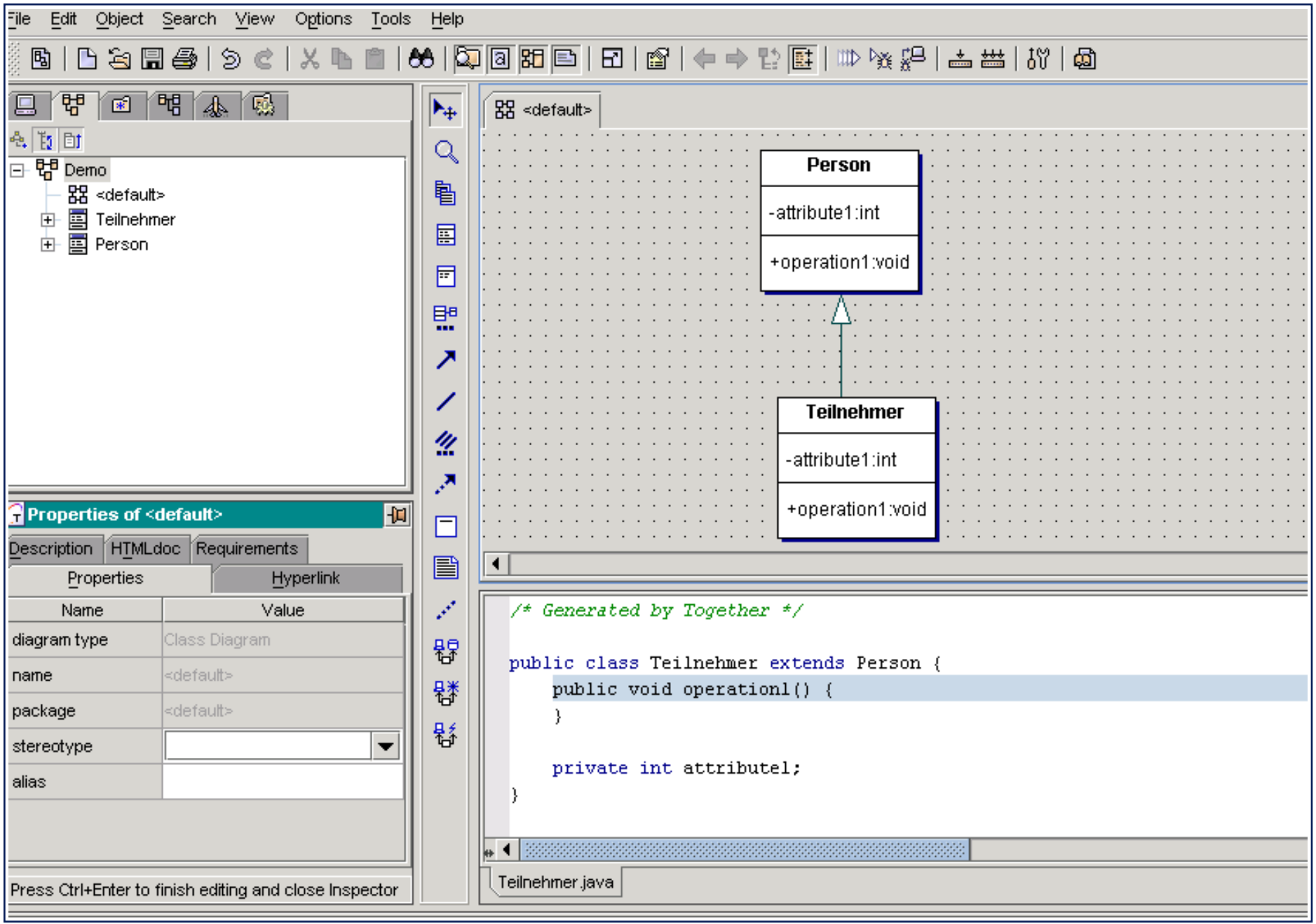
http://www.borland.com/downloads/download_together.aspx

Code Generation in Different Languages in Together



- Basierend auf den Rollen: Business Modeler, Designer, Developer und Programmie-rer werden Sichten auf Arbeitsbereich automatisch konf guriert(View-Management).
- Das Einbinden von Patterns, Templates und vorgefertigten source-basierten Frame-works (Komponenten incl. EJBs) wird unterstützt.
- Zur Qualitätssicherung werden Metriken und Audits angeboten.

Together Screenshot



50.2 Technologies for Code Generation

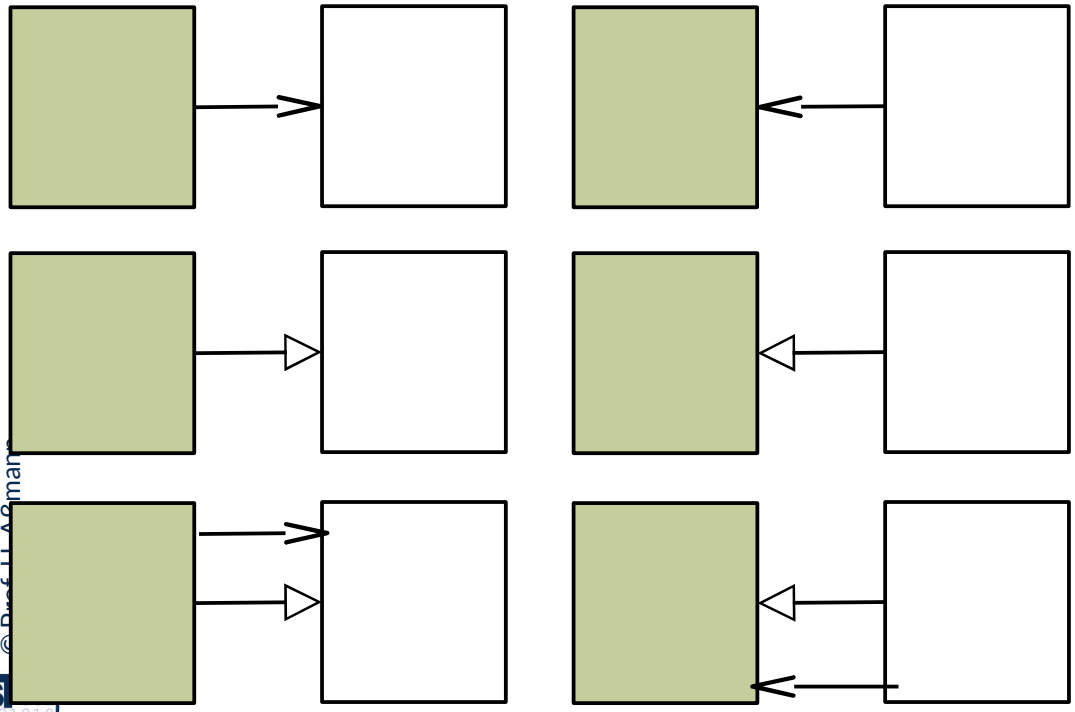
Composition of Separated Hand-Written and Generated Code

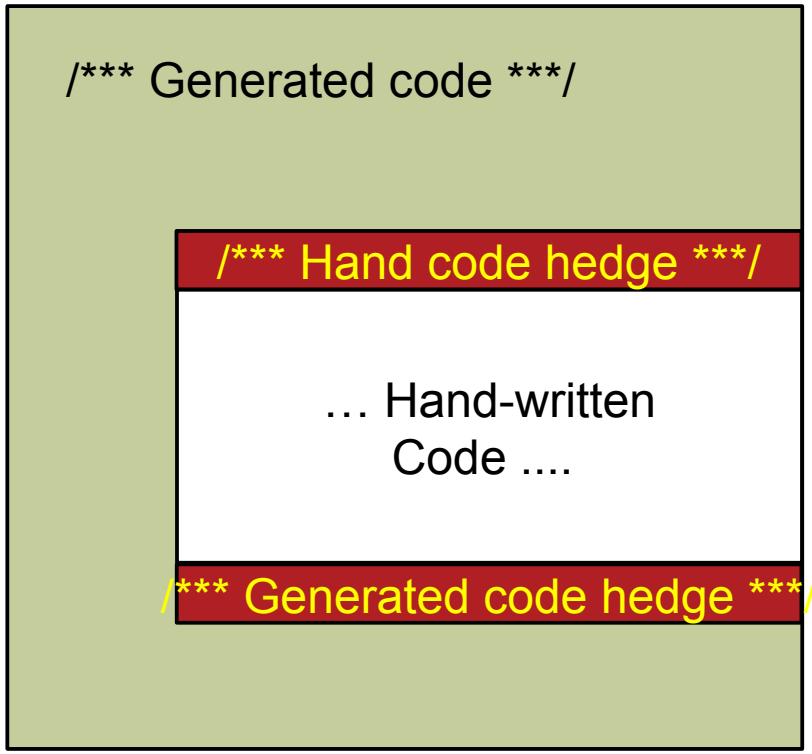
- **In separate files:**
- Coupling by implementation pattern [Völter/Stahl]
- Use class composition like delegation, TemplateMethod, Composite,

  Decorator, etc

In one file:

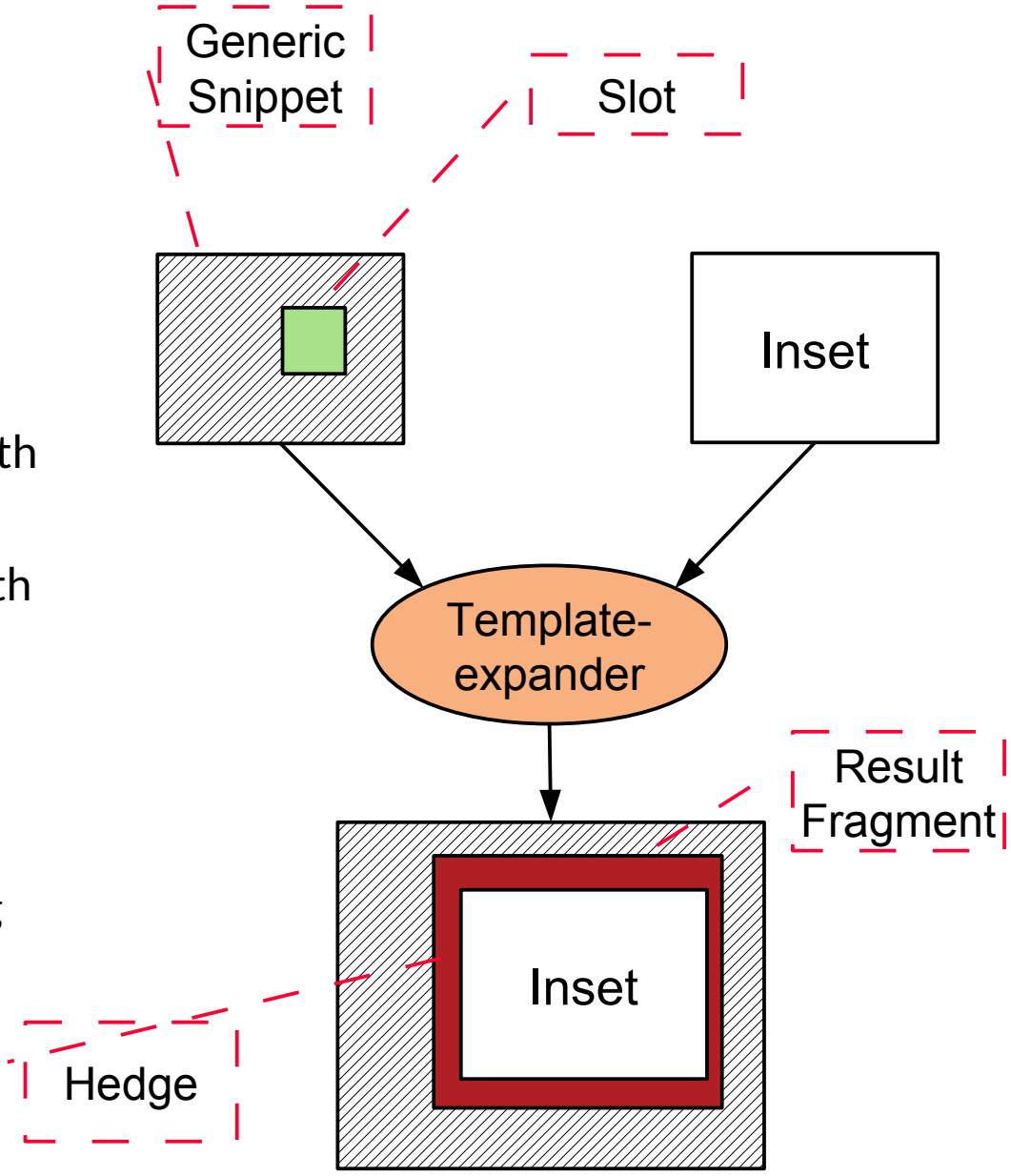
Coupling with **hegdes (Trennmarkierung)**





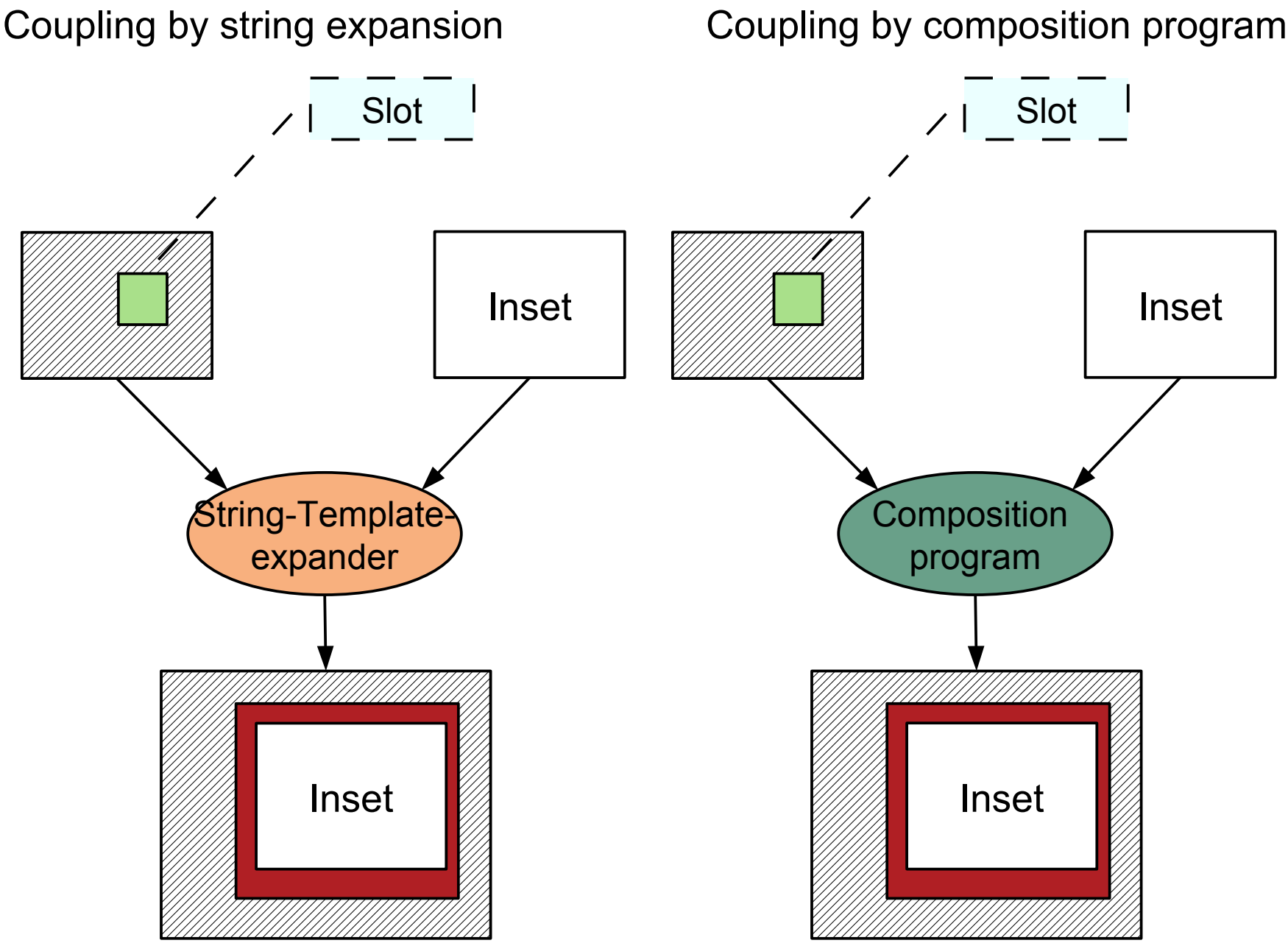
Snippet Programming

- **A fragment (snippet)** is a incomplete sentence of a language, derived from a nonterminal of the grammar, or described by a metaclass
- A **generic fragment (template, form, frame)** is a fragment with **slots (holes, code parameters, variation points)**, which can be *bound (filled, expanded)* with an **inset fragment** to a **result fragment**
- A extensible fragment is a fragment with **hooks (extension points)**, which can be *extended* to a fragment
- **Generic programming** is programming with generic fragments (templates).
- **Invasive programming** is programming with generic and extensible fragments (templates with hooks)



50.2.1 Template-based Code Generation (Schablonenbasierte Programmüberführung)

Template Expansion by Composition of Insets



Slots are Marked by Hedges

- **Hedges** are delimiters that do not occur in the base nor in the slot language
- **Slot hedges** are template2slot hedges marking the transition from the code language to the slot language
- **Inset hedges** are metaprogramming2code hedges marking the transition from the metaprogramming language to the code language

```
// code hedges << >>
Template (superclass:CLASS, t:TYPE) {
  class Worker extends << superclass >>
{
        <<t>> attr = new <<t>>();
        <<t>> getAttr();
       void setAttr(<<t>>);
   }
}
```

Tools for Untyped Template Expansion

- **Frame processing** was invented in [P. Bassett] as an *untyped string template expansion technology*, universal for all textual languages [Holmes/Evans]
  - Frame processing is the main technology for web engineering today: it organizes reuse of page templates
  - The original frame processor used $ as a hedge symbol for slots (slot variables)
- **Macro processing** is not much different
  - Because only slot variables hold insets, macro parameters correspond to slot variables
- XML template engine XVCL [Jarzabek] is an XML-controlled frame processor
  - http://sourceforge.net/projects/fxvcl/files/XVCL%20Specification/Version%202.10/
- String template engines in use today
  - Apache Velocity http://velocity.apache.org/
  - Parr's template engine StringTemplate
  - Jenerator for Java http://www.voelter.de/data/pub/jeneratorPaper.pdf

Velocity String Template Language

- Velocity Template Language (VTL) is a frame processing language with metaprograms in slots
- {#, $} are slot hedges
- < (from XML) is the inset hedge

```
<HTML>
<BODY>
Hello $customer.Name!
<table>
#foreach( $mud in $mudsOnSpecial )
   #if
( $customer.hasPurchased($mud) )
      <tr>
        <td>
          $flogger.getPromo( $mud )
        </td>
      </tr>
   #end
#end
</table>
```

```
<html>
<body>
#set( $foo = "Velocity" )
Hello $foo World!
</body>
<html>
```

http://velocity.apache.org/engine/releases/velocity-1.7

Velocity Template Language

- Velocity Template Language (VTL) is a simple scripting language in the spirit of TCL
- It has control structures (if, switch, foreach), assignments (set), and macros

http://velocity.apache.org/engine/releases/velocity-1.7

```
#macro( inner $foo )
  inner : $foo
#end

#macro( outer $foo )
   #set($bar = "outerlala")
   outer : $foo
#end

#set($bar = 'calltimelala')
#outer( "#inner($bar)" )
```

Problem: the result of string template expansion may not be syntactically correct, nor well-formed, target language (error-prone)

Typed Template Expansion

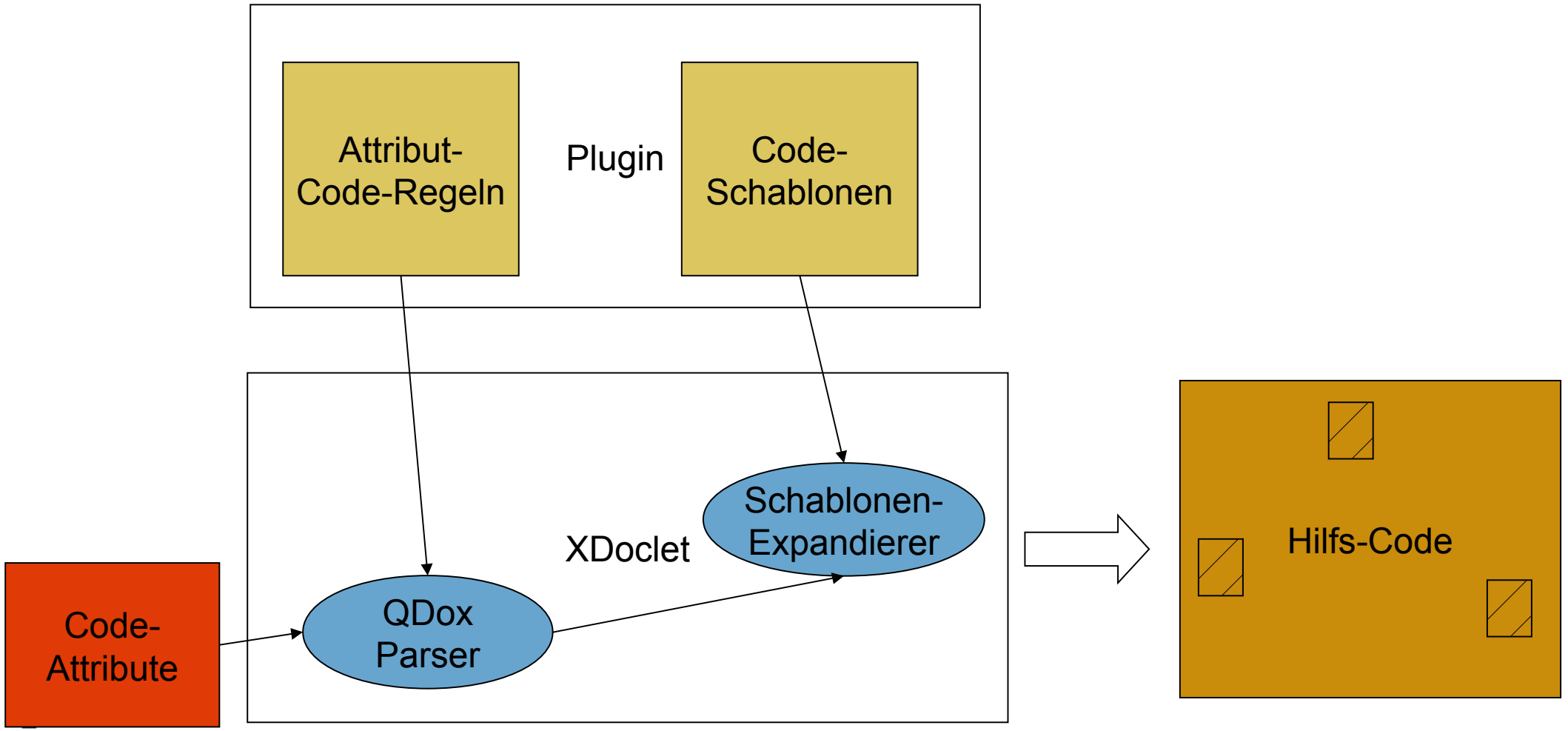
- Metamodel-controlled template engines
  - Open Architecture Ware's Scripting language
- Invasive Softwarekomposition bietet volltypisierte Schablonenexpansion (siehe CBSE)
  - Getypte Schablonen-Expansion und -erweiterung
  - Kann für beliebige Programmiersprachen instantiiert werden
  - http://www.the-compost-system.org
  - http://www.reuseware.org

Semantic Macros

- **Semantic Macros** are metaprogramming procedures which are typed parameters and results.
  - They allow for type-safe static metaprogramming.
- Examples:
  - Scheme

Xdoclet (xdoclet.sf.net)

- Xdoclet wandet Attribute (Metadaten) in Code um
  - Schablonen-gesteuerte Codegenerierung



Slot Markup Languages

- are special reuse languages

50.3 Code Modification and Reparsing (Codemodifikation und -rückführung)

Example of Code Reparsing Technique

- Code-Reparsing in Fujaba:

http://www.fokus.fraunhofer.de/en/fokus_events/motion/ecmda2008/_docs/rs01_t03_ManuelBork_EMCDA2008_slides.pdf

- Parallel Parsing of Template and Generated Code, with comparision to resolve indeterministic situations of re-parsing

Vorgehen der Coderückführung

- **Aufgabe**: Erkennen geänderter „Code“-Teile und Rückführung in die Entwurfsmodelle
- **Prinzip**: Die modifizierte Quellcodedatei stammt in jedem Fall aus der Single-Source-Spezifikation eines CASE-Tools, in die der geänderte Programmcode zurückgeführt werden soll
  - Kennzeichnungen der Single Source-Spezifikation sind noch vorhanden.
  - Strukturierung der Quellcodefiles ist so, dass Abschnitte erkennbar sind und ihnen eindeutig die Objekte der Entwurfsspezifikation zugeordnet werden können, beispielsweise durch:
    - Trennmarkierungen (-kommentare oder -attribute, hedges) zwischen den Abschnitten (Markup) wird zum Erkennen der Grenzen benutzt
    - Vorhandensein von „Code“-Teilen als zielsprachenspezifische Freiräume (hooks)
    - Weitere Rückführinformationen gegebenenfalls aus dem Quellfilekopf oder -kommentaren

Quelle: Lempp, P., Torick R. J.. Software Reverse Engineering: An Approach to Recapturing Reliable Software; 4th Ann. Joint Conf. on Softw. Quality and Productivity, Crystal City, VA, March 1-3, 1988

- **Trace hedges** are hedge symbols inserted by a template expander to demarcate the template from the inset.

The End