26. Documentation as Synchronized Dependent Model in a Macromodel

Documentation Generation as App for RAG

Prof. Dr. rer. nat. Uwe Aßmann
Institut für Software- und Multimediatechnik
Lehrstuhl Softwaretechnologie
Fakultät für Informatik
TU Dresden
http://st.inf.tu-dresden.de
Version 19-0.4, 07.12.19

1) Tasks
2) Template-Driven Documentation Tools
3) Literate Programming
4) Elucidative Modeling and Documentation Tools

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Literature

- C. Wilke, A. Bartho, J. Schroeter, S. Karol, U. Aßmann. Elucidative Development for Model-Based Documentation and Language Specification (Extended Version). Technische Universität Dresden. Institut fur Software- und Multimediatechnik. Technical Reports TUD-FI12-01-Januar 2012, ISSN 1430-211X.
  - http://nbn-resolving.de/urn:nbn:de:bsz:14-qucosa-83442
- Andreas Bartho. Elucidative Modeling. PhD thesis, Technische Universität Dresden, Fakultät Informatik, May 2014.
  - http://nbn-resolving.de/urn:nbn:de:bsz:14-qucosa-208060
  - https://www.linkedin.com/pub/andreas-bartho/ba/922/8a4?trk=pub-pbmap

26.1 Tasks of Documentation Tools

http://en.wikipedia.org/wiki/Software_documentation

Basics of Software Documentation

- Documentation is a means of **communication**
  - between developers and future developers
  - between coders and testers
  - between developers and managers (for reviews and audits)
- Problems:
  - Documentation ***ages*** because code is modified and evolved
  - Good documentation costs time and money
- Different kinds of documentation:
  - **Generated documentation** is derived from code and models
  - **Integrated Documentation** is derived from the code (e.g., in comments), e.g., JavaDoc
  - **Elucidative Documentation**, derives both from another and keeps it consistent (generative or round-trip engineering)
- Standards:
  - national DIN 66230, 66231, 66232, 66270(1998)
  - international ISO/IEC 6592(2000), ISO/IEC 18019(2004)

Without documentation, a program is not software

Quelle: [24 S. 241 ff.]

Taxonomy of Documentation Documents

- **User documentation** (Benutzerdokumentation) explains the program to end users
  - Tutorials, user handbook, online documentation
- **System documentation** for installation, test cases, code documentation, maintenance, operations
  - **API documentation** documents interfaces of the system or framework, to let programmers use them for writing apps
- **Project documentation**
  - Developer documentation
  - Project documentation (project plan, requriements specification, status reports, after study)
- **Quality documentation**
  - Test-, review, audit documentation
- **Process documentation**
  - Standards, processes

Quelle: [24 S. 245 ff.]

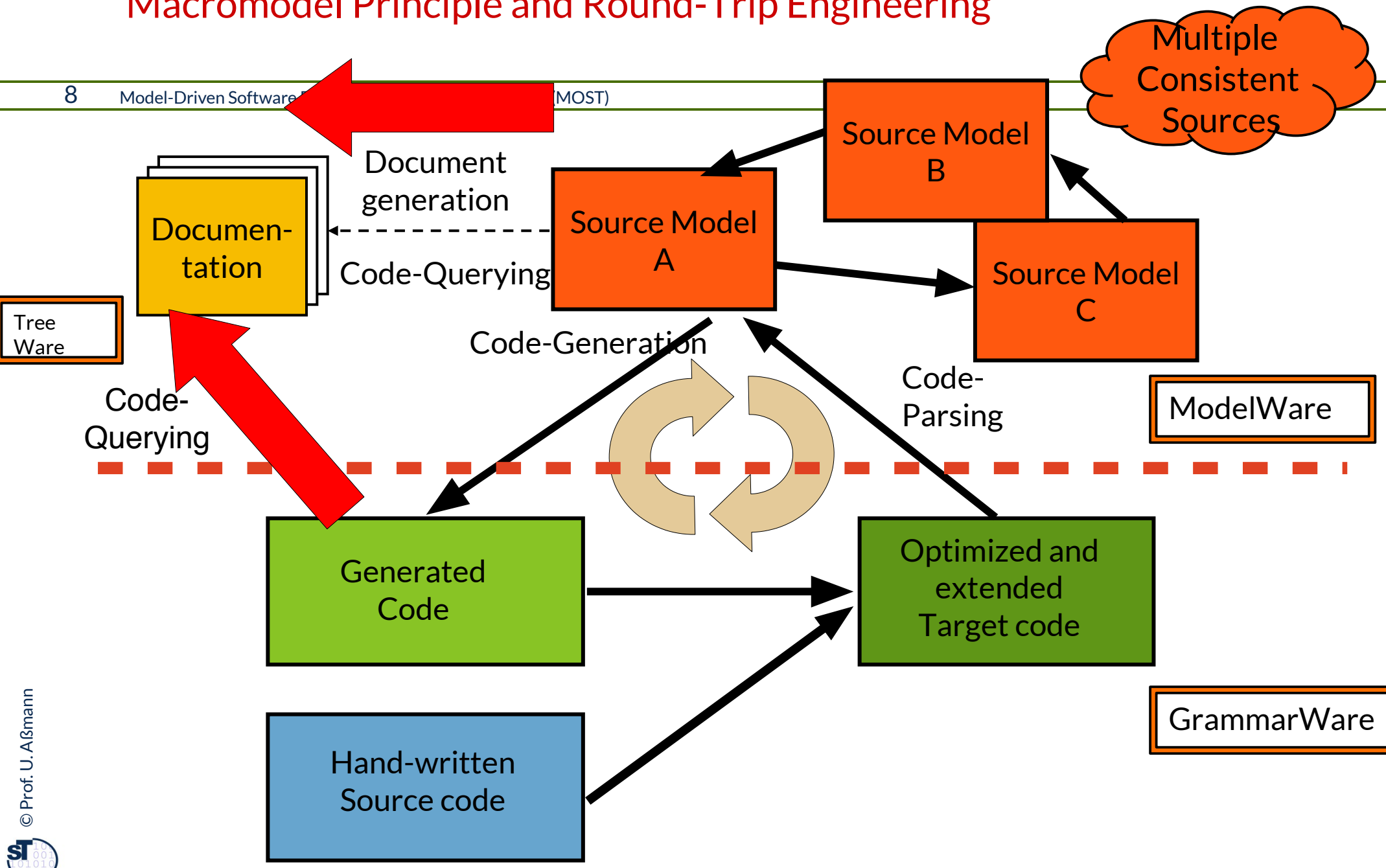
Tasks of Documentation Tools

- Basically, documentation generation is similar to code generation. Documentation is created in higher-order attributes on a link tree by a RAG
- **Documentation generation is an app of a RAG**
- **Generation** of derived documents from code and models
  - Generation of Word, odt, rtf, xml, html formats
  - Generation of figures (svg, png, pdf)
  - Generation of snippets and generic snippets
  - Back-linking to originals
- **Filling** of documentation templates (hedge-principle)
- **Parameterization** with layouts
  - via css-style sheets

26.2 Generative, Template-Driven Documentation Tools

.. Documentation derived from code and models, based on template-based code generation

Macromodel Principle and Round-Trip Engineering



Documentation Tool JavaDoc is a Template Expander

- JavaDoc reads Java source code and extracts html from the code comments, based on **html templates**
  - Typical hedge-based code generation with generic snippets
- Generation of additional contents and indices
- Controlled by Java metadata attributes
  - @author, @date, @param
- Layouting via plugin classes called *doclets*
- JavaDoc has been realized for all programming languages

[JavaDoc; Wikipedia: http://de.wikipedia.org/wiki/JavaDoc]

JavaDoc is a Typical HRAG Application

- The html documentation is computed in a higher-order synthesized attribute `htmldoc:HTML`

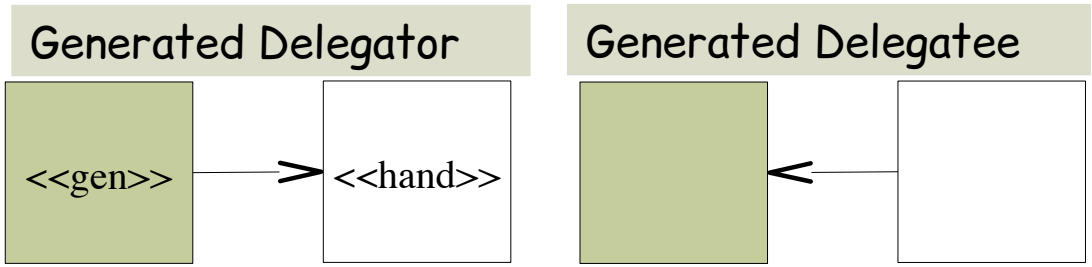
```
// schematic, synthesis from bottom to top
Interpretation javaDoc(Tree → Tree)  {
  Attributions of Root(classes[]) {
    this.htmldoc := map + classes.htmldoc;
    <println(„Result is %S“, this.htmldoc)>
  }
  Attributions of Class(superclass:Class,methods{}) {
     this.htmldoc := <superclass.Name + st2.htmldoc;
  }
  Attributions of Method(name,comment) {
     this.htmldoc := „<h1>“+name+“</h2>“+comment.htmldoc;
  }
  Attributions of Comment(text) {
     this.htmldoc := text;
  }
}
```

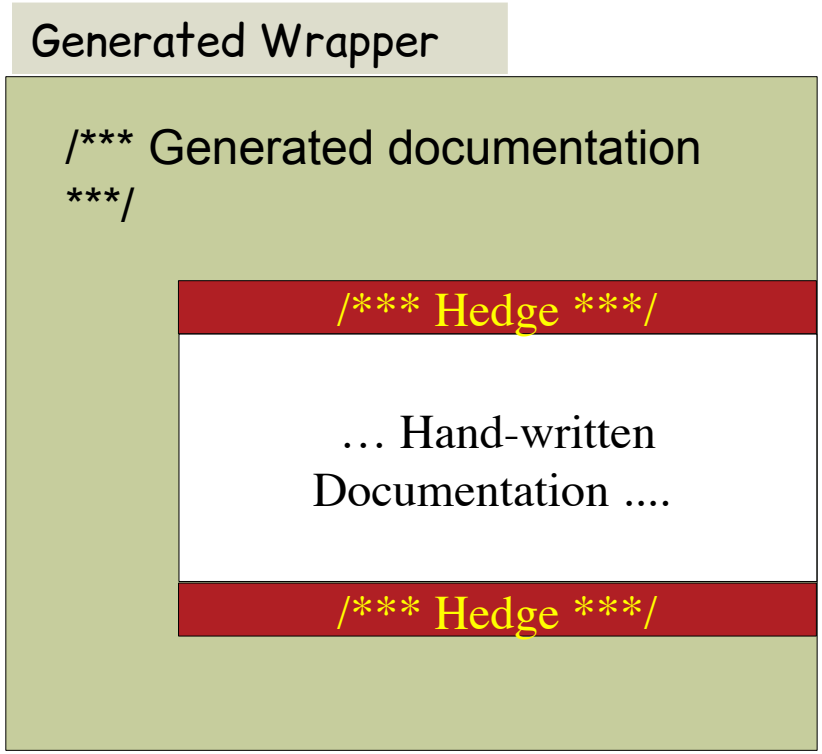
Composition of Separated Hand-Written and Generated Documentation Snippets

- **In separate files:** Coupling by “include”
- Only possible if document format supports subdocument inclusion
  - e.g., TeX or Framemaker

In one file:

Coupling with **hegdes (Trennmarkierung)**



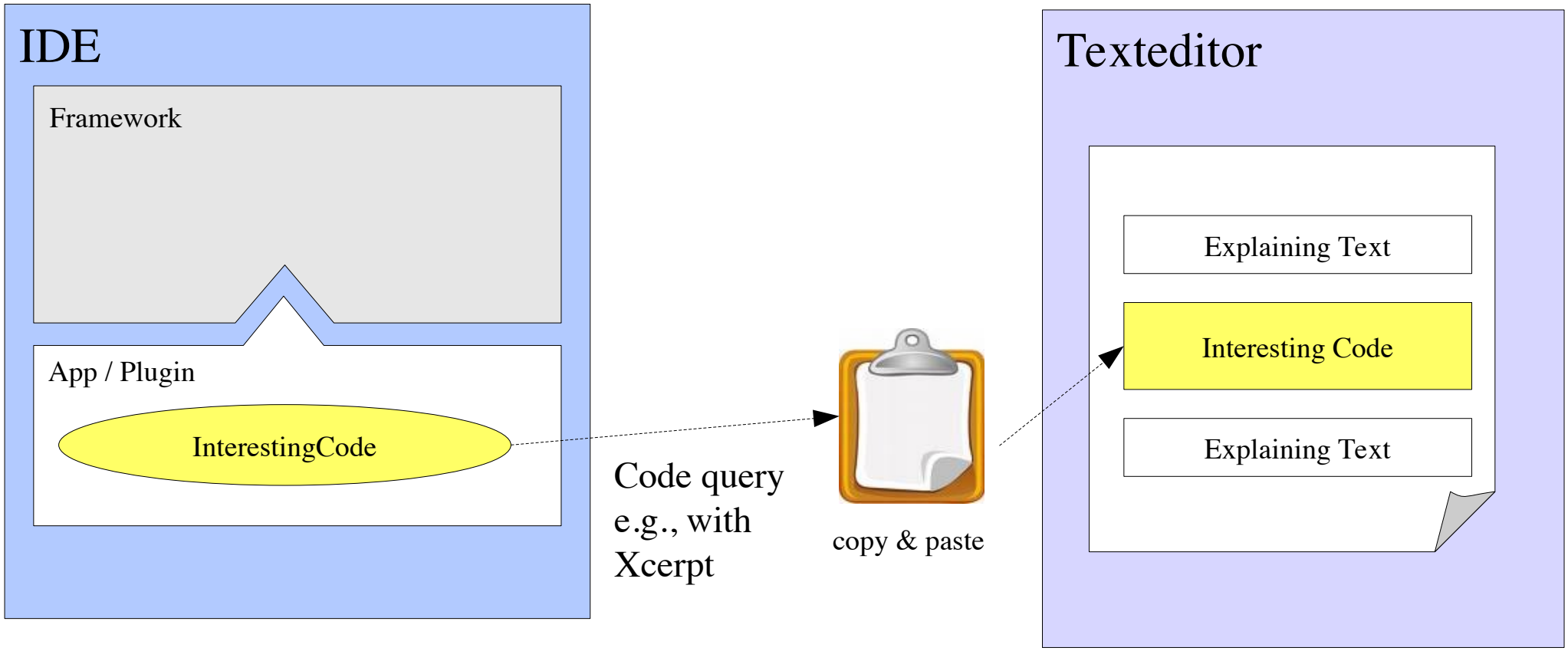


26.3 Literate Programming

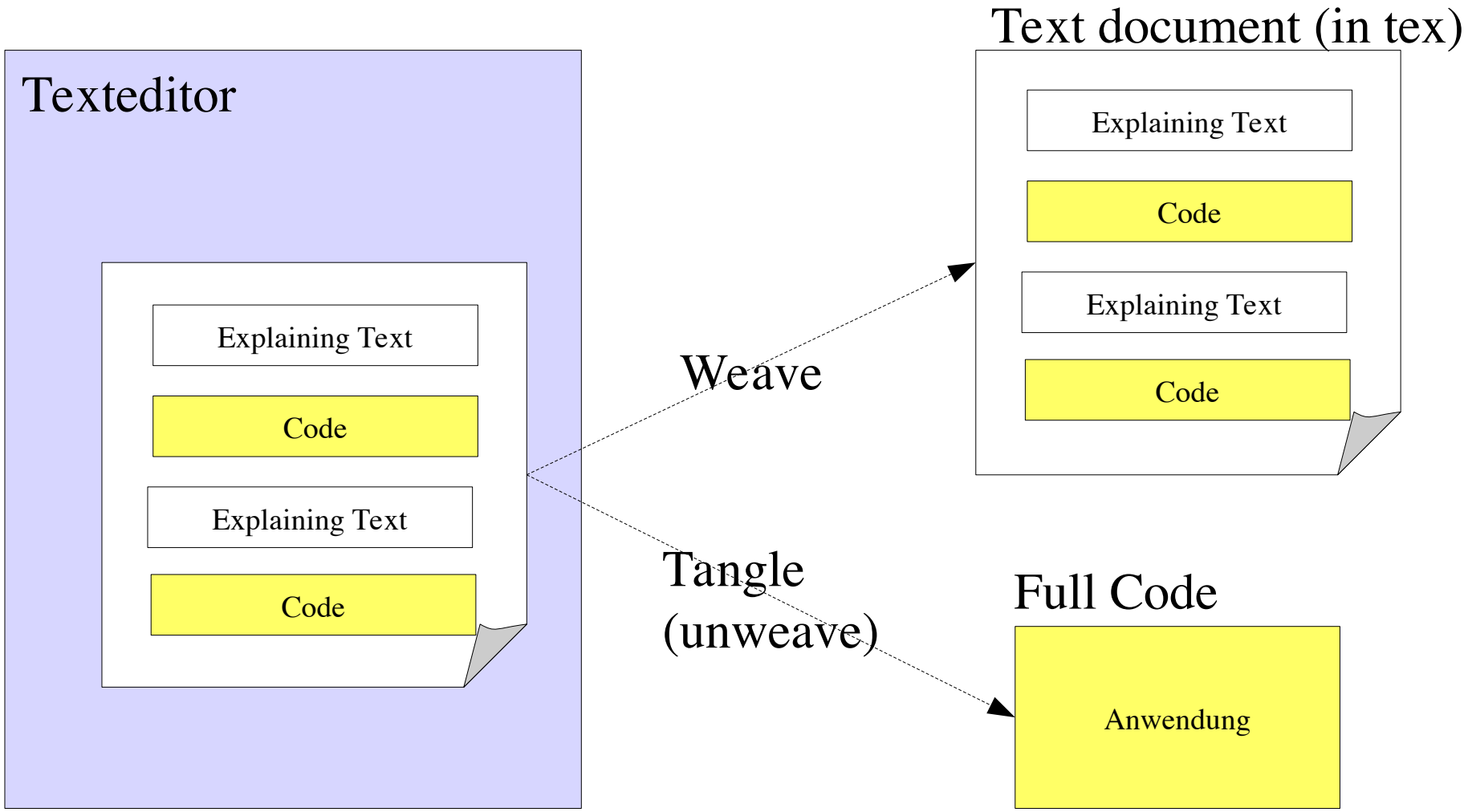
- They integrate code, models and documentation by **separating code from documentation**

How to Write Documentation and Tutorials?

Manual Writing of Tutorials



Literate Programming by Code Unweaving



Literate Programming

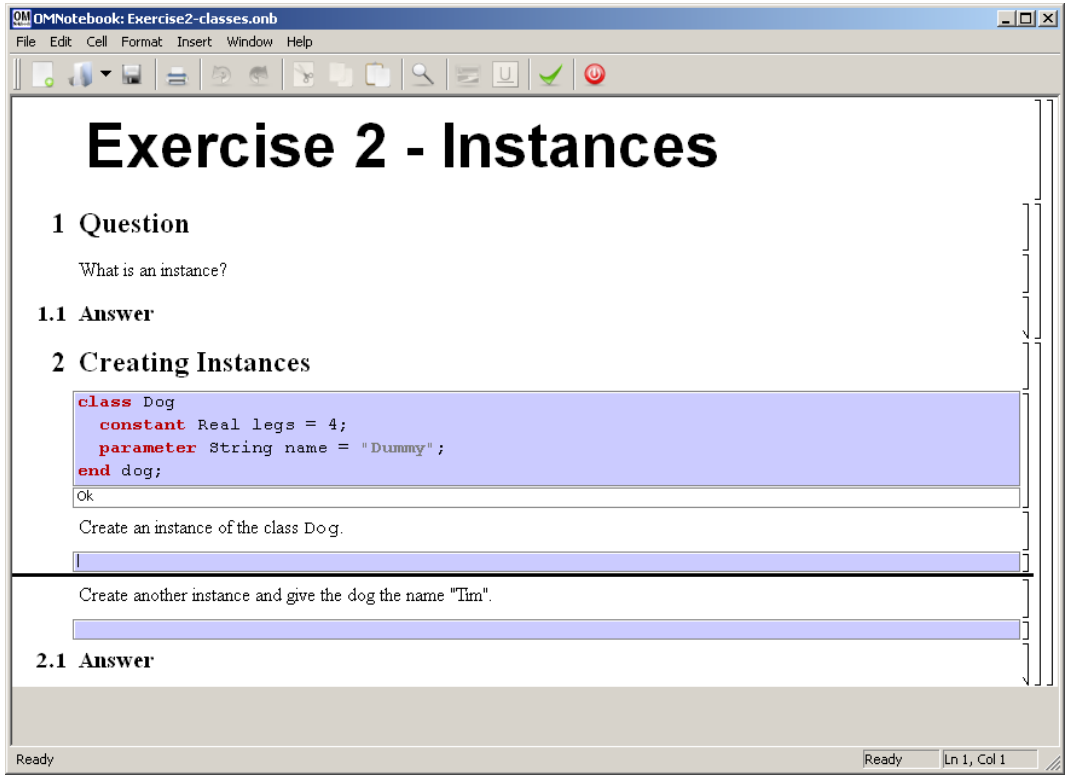
⟦The program text below specifies the "expanded meaning" of ⟨Program to print ... numbers 2⟩; notice that it involves the top-level descriptions of three other sections. When those top-level descriptions are replaced by their expanded meanings, a syntactically correct PASCAL program will be obtained.⟧

```
⟨Program to print the first thousand prime
      numbers 2⟩ ≡
program print_primes(output);
  const m = 1000;
    ⟨Other constants of the program 5⟩
  var ⟨Variables of the program 4⟩
    begin ⟨Print the first m prime numbers 3⟩;
    end.
```

[Literate Programming von Donald E. Knuth]

- The TeX engine is programmed literately
- Overview: http://www.literateprogramming.com/
- OMNotebook/DrModelica: http://www.modelica.org/tools

OMNotebook – Literate Programming with DrModelica

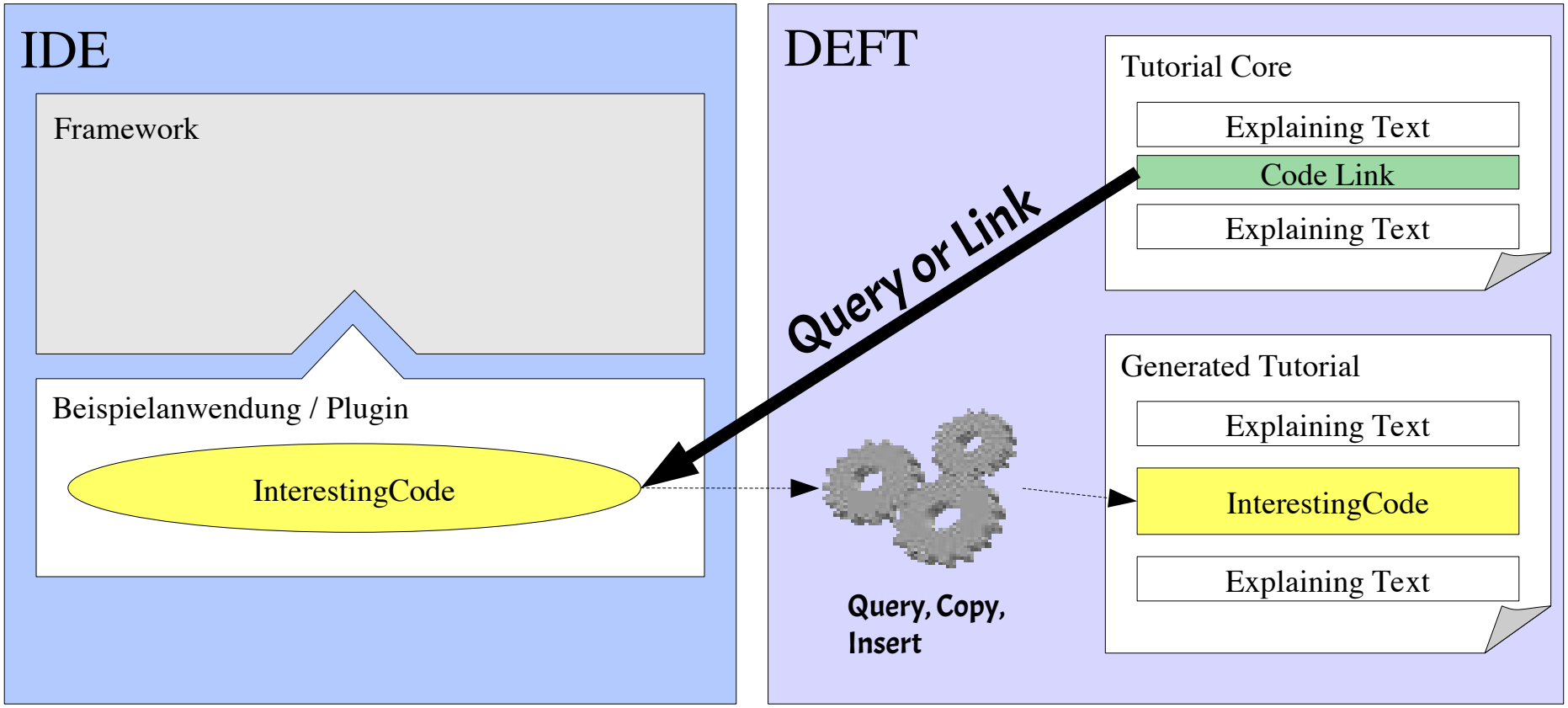


- Linked documents with interactive exercises
- Inspired by DrScheme und DrJava, learning tools for Scheme resp. Java
- www.openmodelica.org

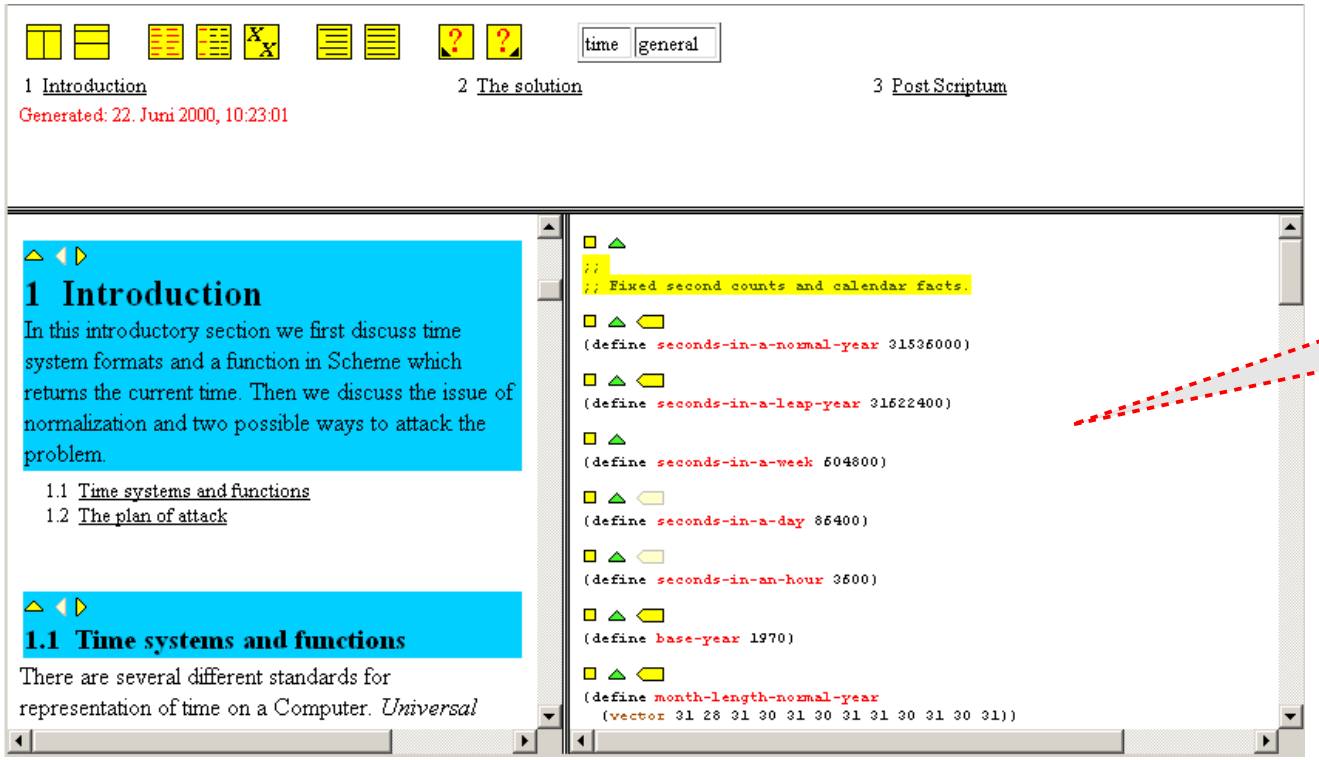
26.4 Elucidative Documentation Tools

- They link code, models and documentation by **model and code mapping**

Elucidative Programming Links Documentation with Queries to Code



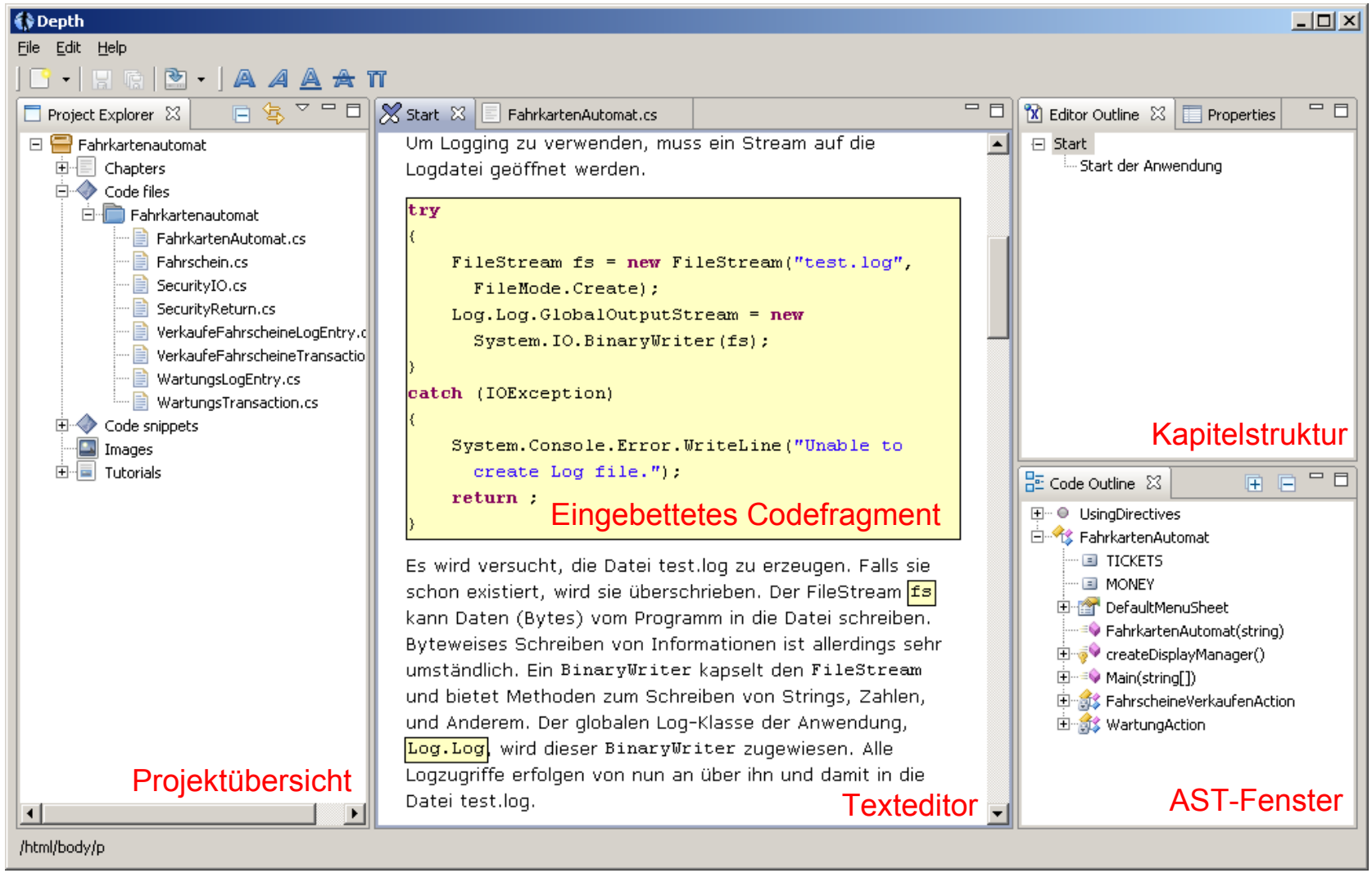
Elucidative Programming



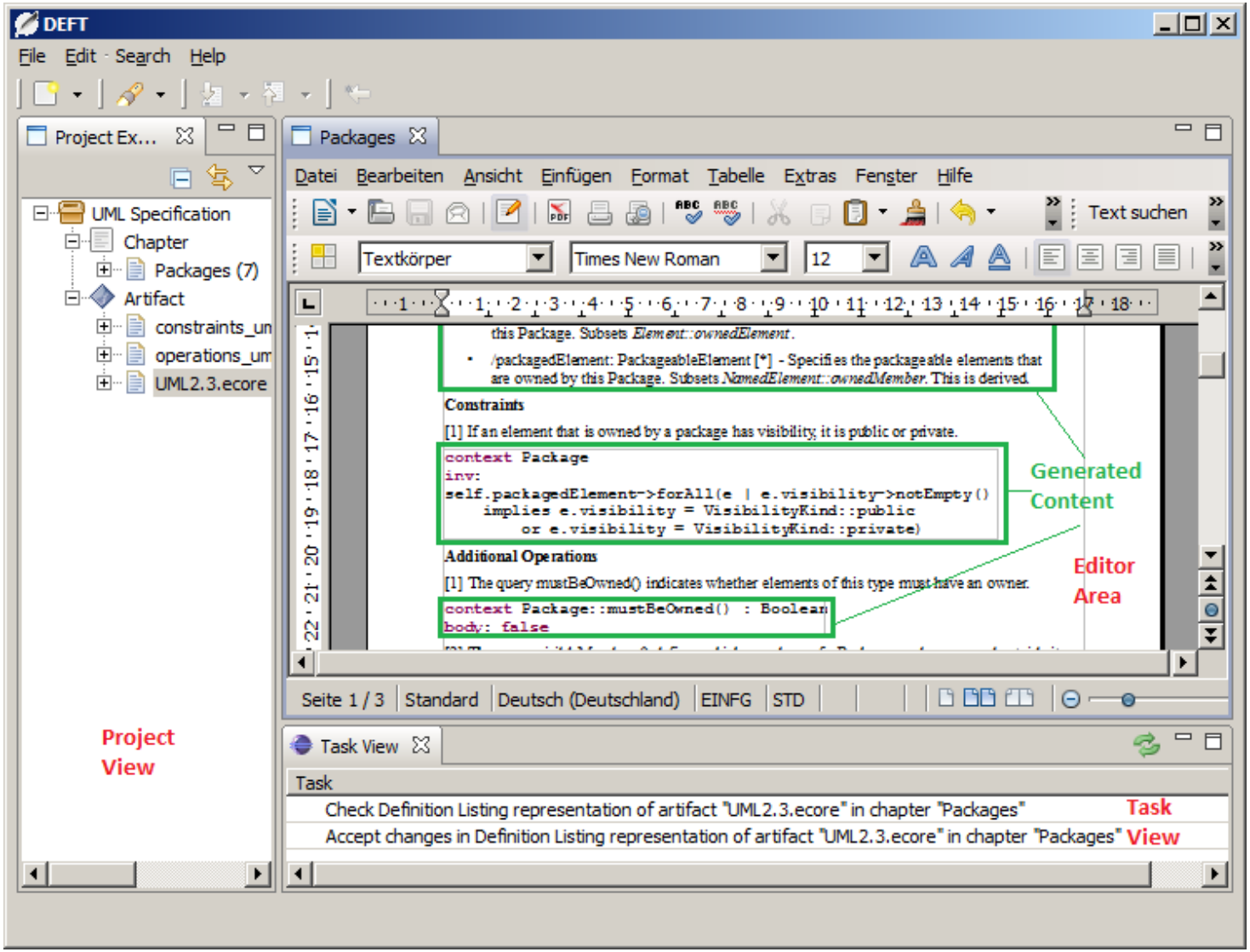
„Scheme Elucidator" Environment

- Elucidative Programming shows documentation and code in parallel
- http://www.cs.aau.dk/~normark/elucidative-programming/
- http://deftproject.org

Development Environment For Tutorials (DEFT www.deftproject.org)



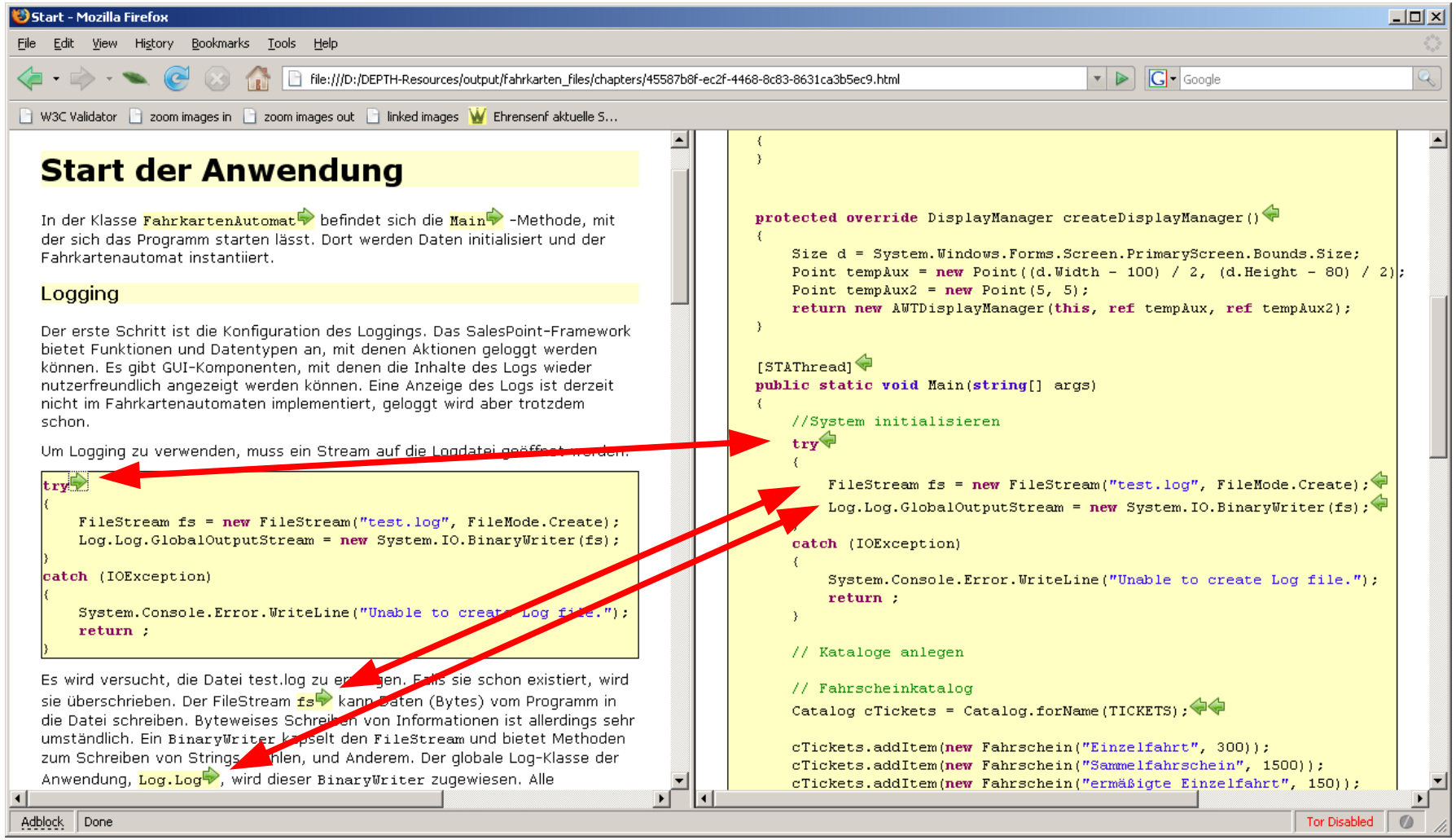
Embedding UML Constraints for UML Models into Documentation



Development Environment For Tutorials

- Eclipse RCP application, language independent
- Management of code, models and text
- Prettyprinting of code fragments from code templates
- Hot update of generated documentation
  - Automatic update of embedded code fragments
  - Notiviation if code fragments have changed

Generated HTML Tutorial



The End

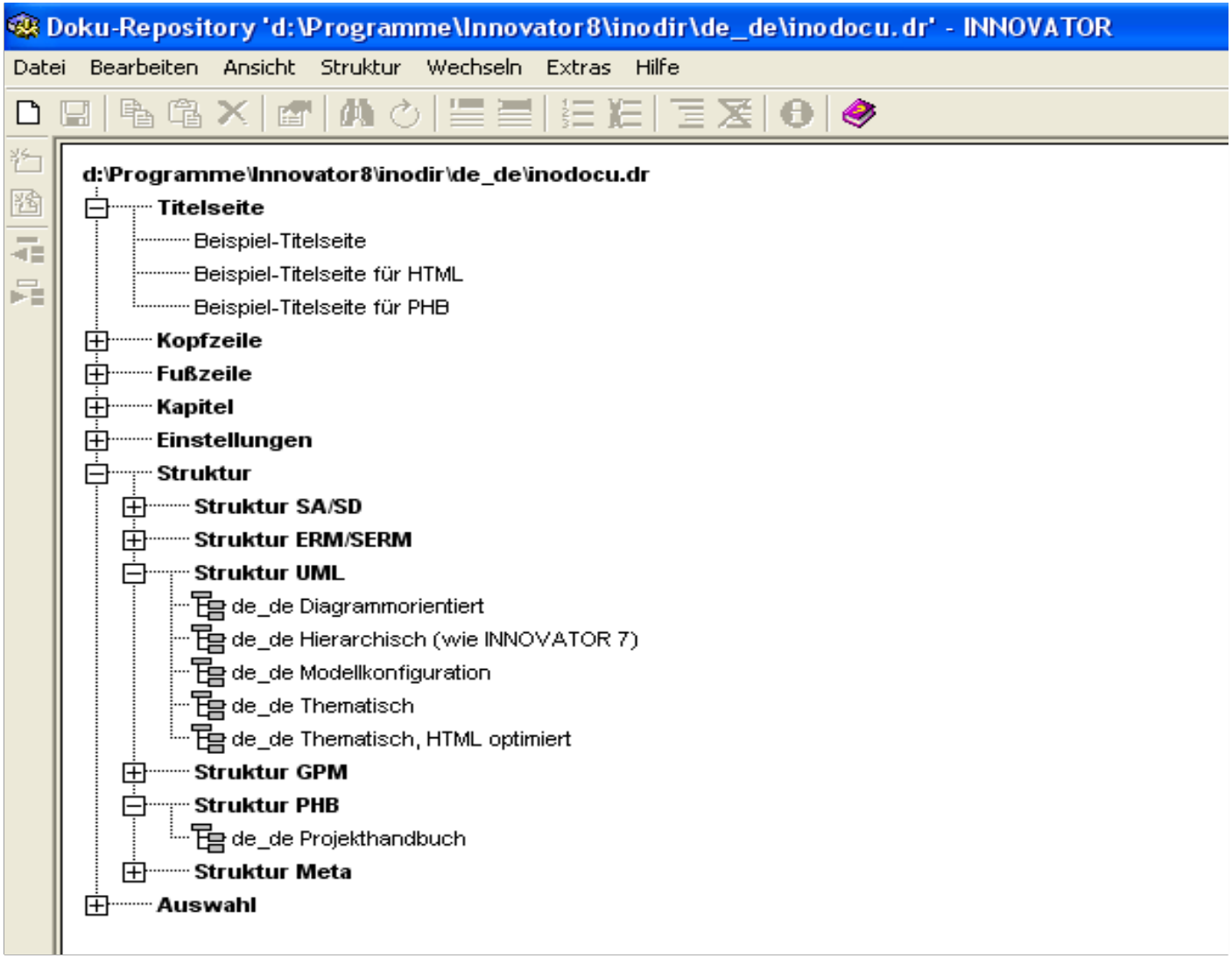
- Why is generation of documentation similar to code generation?
- Explain why a higher-order RAG is useful for documentation generation
- Which role does a pattern-matching language such as Xcerpt play in documentation generation?
- Why is the generation of documentation part of a macromodel?
- Why is a documentation a *derived model?*
- What happens if text from the API documentation flows back into the code as comments?

A.1 Other Template Expanders for Documentation Generation

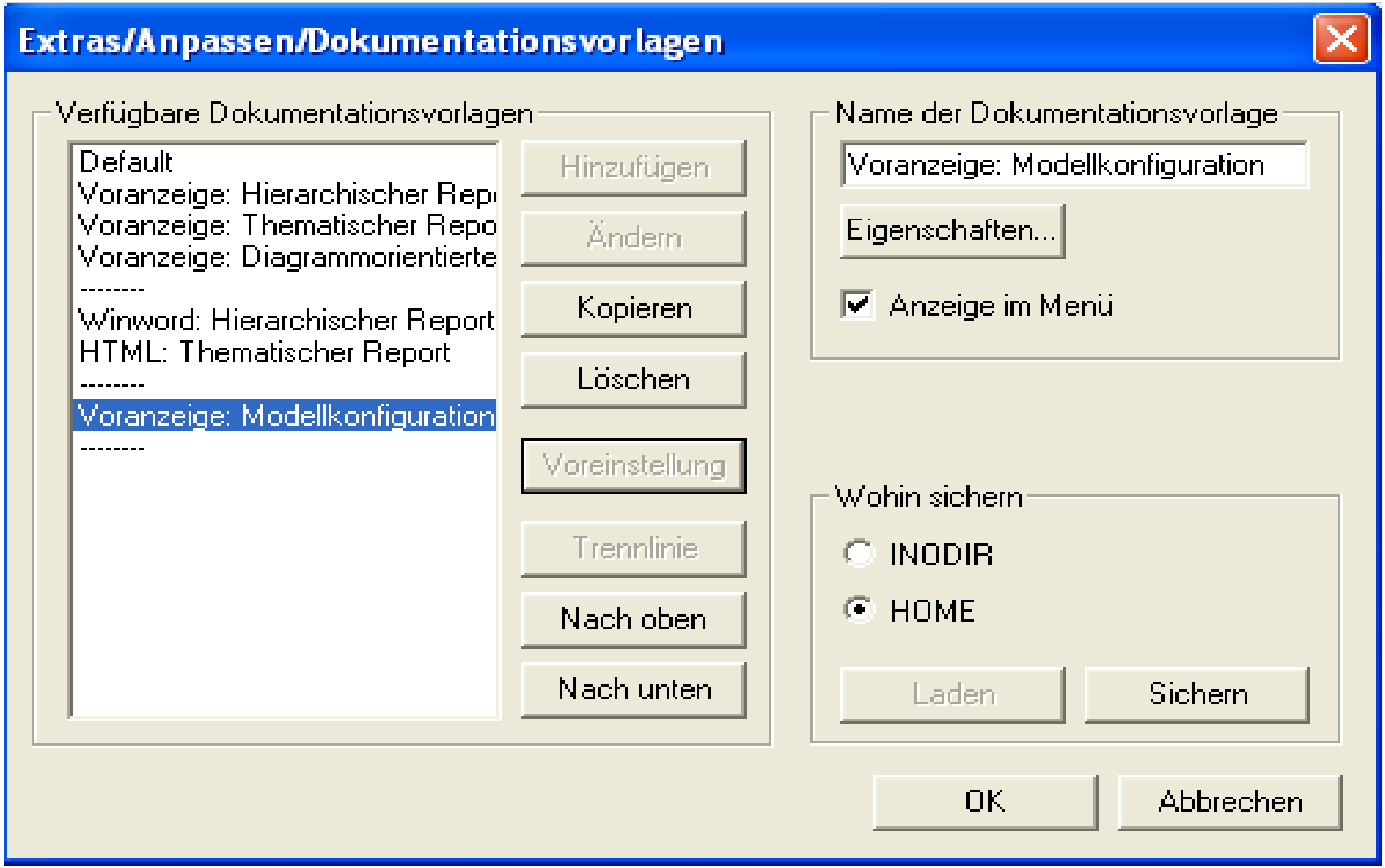
Documentation Tools of MID Innovator

- Innovator provides documentation templates, into which diagrams, models, code can be embedded
- Several formats:
  - pdf
  - Word
  - ASCII
  - XML

Ex.: Innovator Documentation Template (Dokumentationsvorlage)



Ex.: Innovator Documentation Template (Dokumentationsvorlage): Adaptation



Innovator - Generated Example Word Document

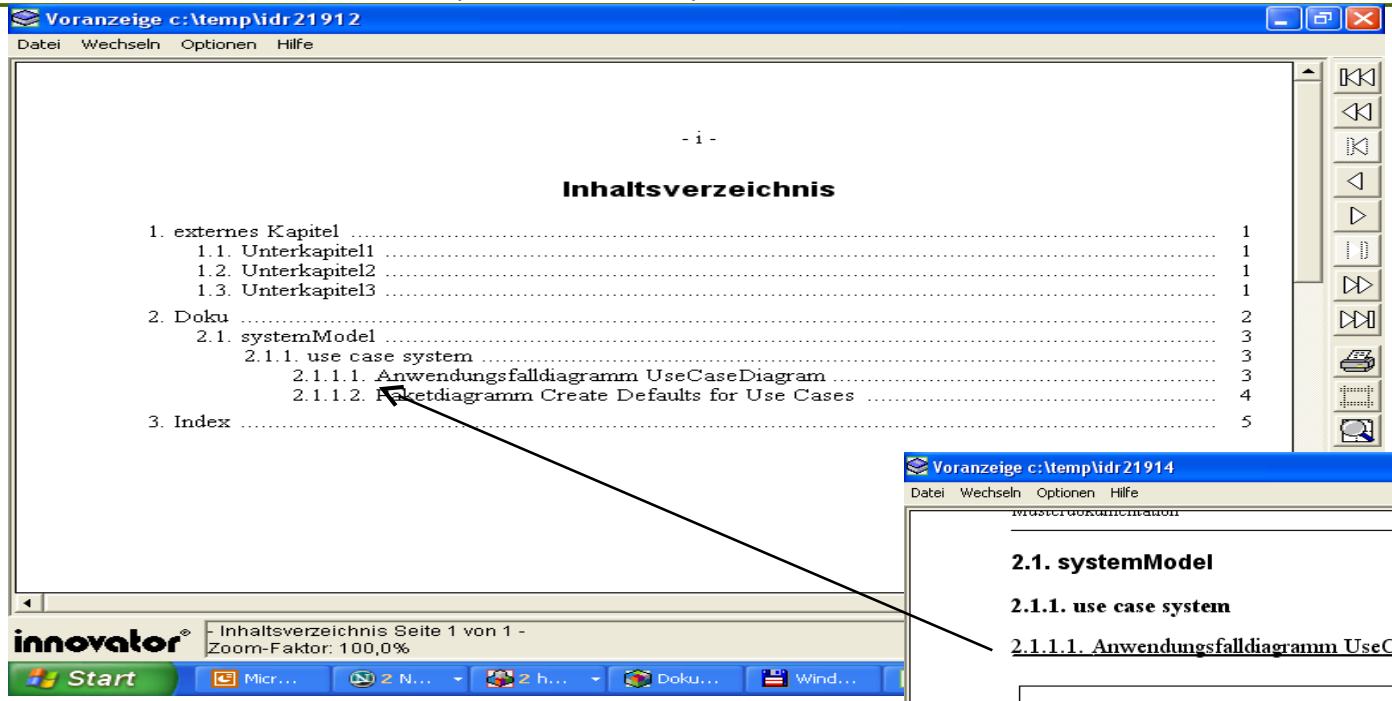
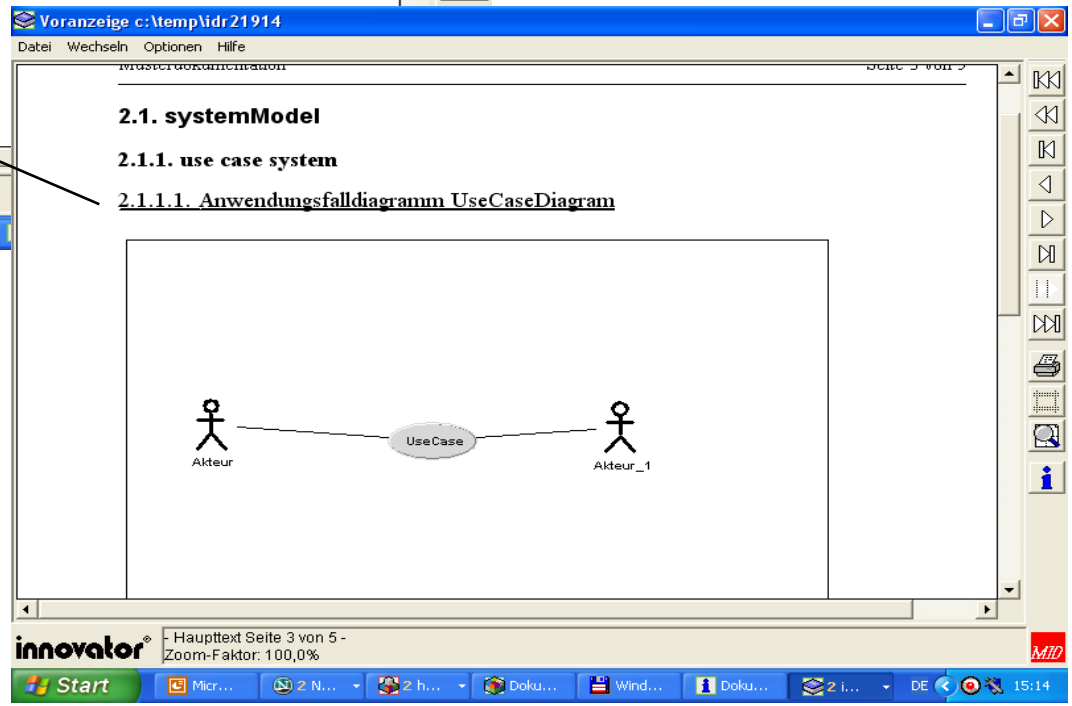


Table of Contents



Integration of a Use Case Diagram (section 2.1.1.1.)

Index is generated