



31. 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 21-0.3, 08.01.22

- 1) Tasks
- 2) Template-Driven Documentation Tools
- 3) Literate Programming
- 4) Elucidative Modeling and Documentation Tools
- 5) Web-based API Documentation Generators

References

- ▶ D. E. Knuth, Literate Programming, The Computer Journal, Volume 27, Issue 2, 1984, Pages 97–111, <https://doi.org/10.1093/comjnl/27.2.97>
- ▶ D. Cordes and M. Brown, "The literate-programming paradigm," in Computer, vol. 24, no. 6, pp. 52-61, June 1991, doi: 10.1109/2.86838.
- ▶ Kurt Nørmark. Elucidative programming. Nordic Journal of Computing, 2000. Citeseer: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.408.2506&rep=rep1&type=pdf>
- ▶ 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 für 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>

Interesting

- ▶ <https://www.writethedocs.org/> is a conference for documentation practitioners
- ▶ <https://waset.org/software-implementation-and-software-documentation-conference>



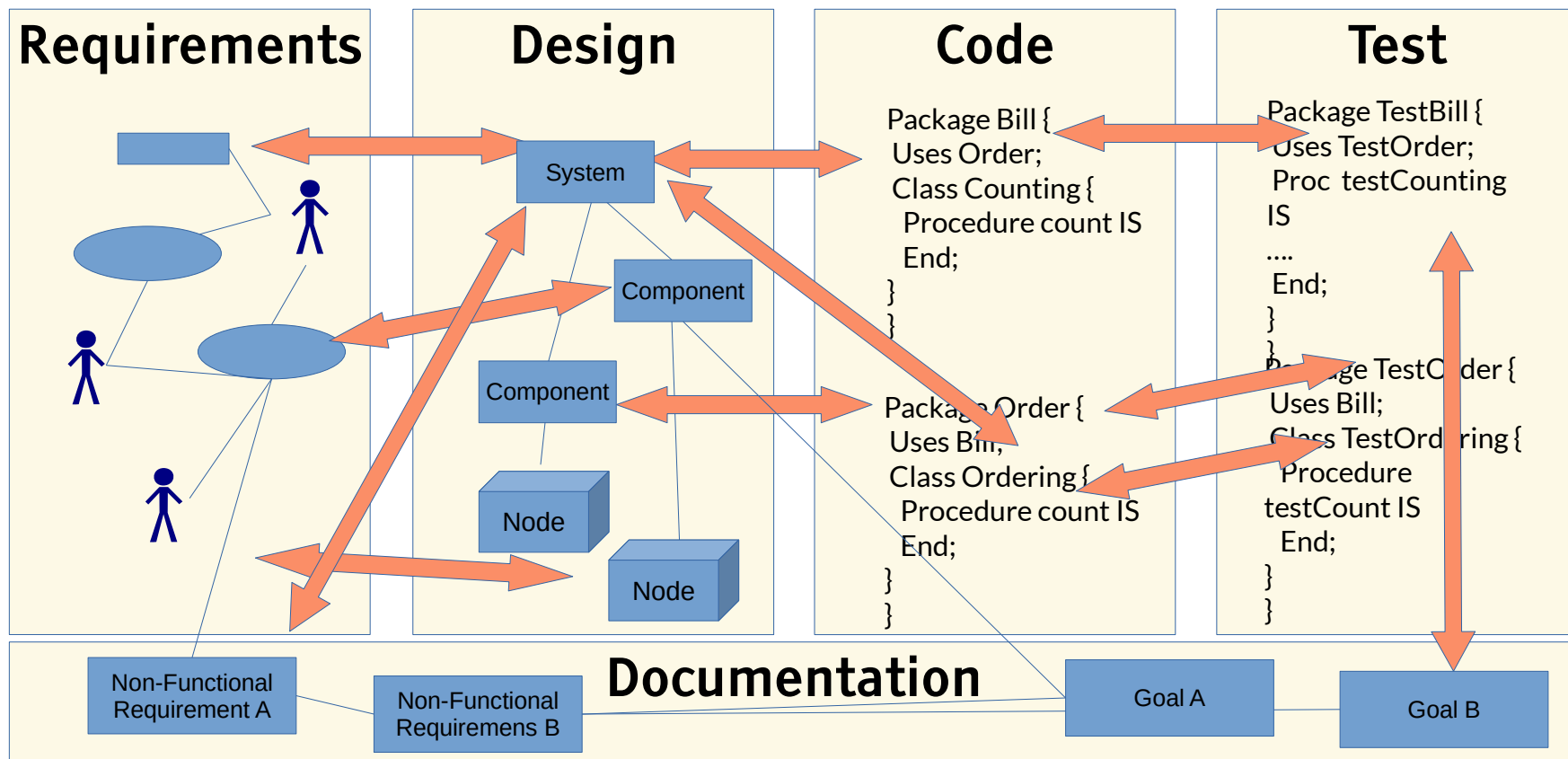


31.1 Tasks of Documentation Tools

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

Q12: The ReDoDeCT Problem and its Macromodel

- ▶ The **ReDoDeCT problem** is the problem how requirements, documentation, design, code, and tests are related (→ V model)
- ▶ Mappings between the Requirements model, Documentation files, Design model, Code, Test cases
- ▶ A **ReDoDeCT macromodel** has maintained mappings between all 5 models



Basics of Software Documentation

- ▶ Documentation is a means of **communication** to keep software alive
 - 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 (**documentation aging**)
 - 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

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
 - **Architecture documentation** to highlight the architectural structure of the software, e.t., with arc42 (<https://www.arc42.de/>)
- ▶ **Project documentation**
 - Developer documentation
 - Project documentation (project plan, requirements specification, status reports, after study)
- ▶ **Quality documentation**
 - Test-, review, audit documentation
- ▶ **Process documentation**
 - Standards, processes

Tasks of Documentation Tools

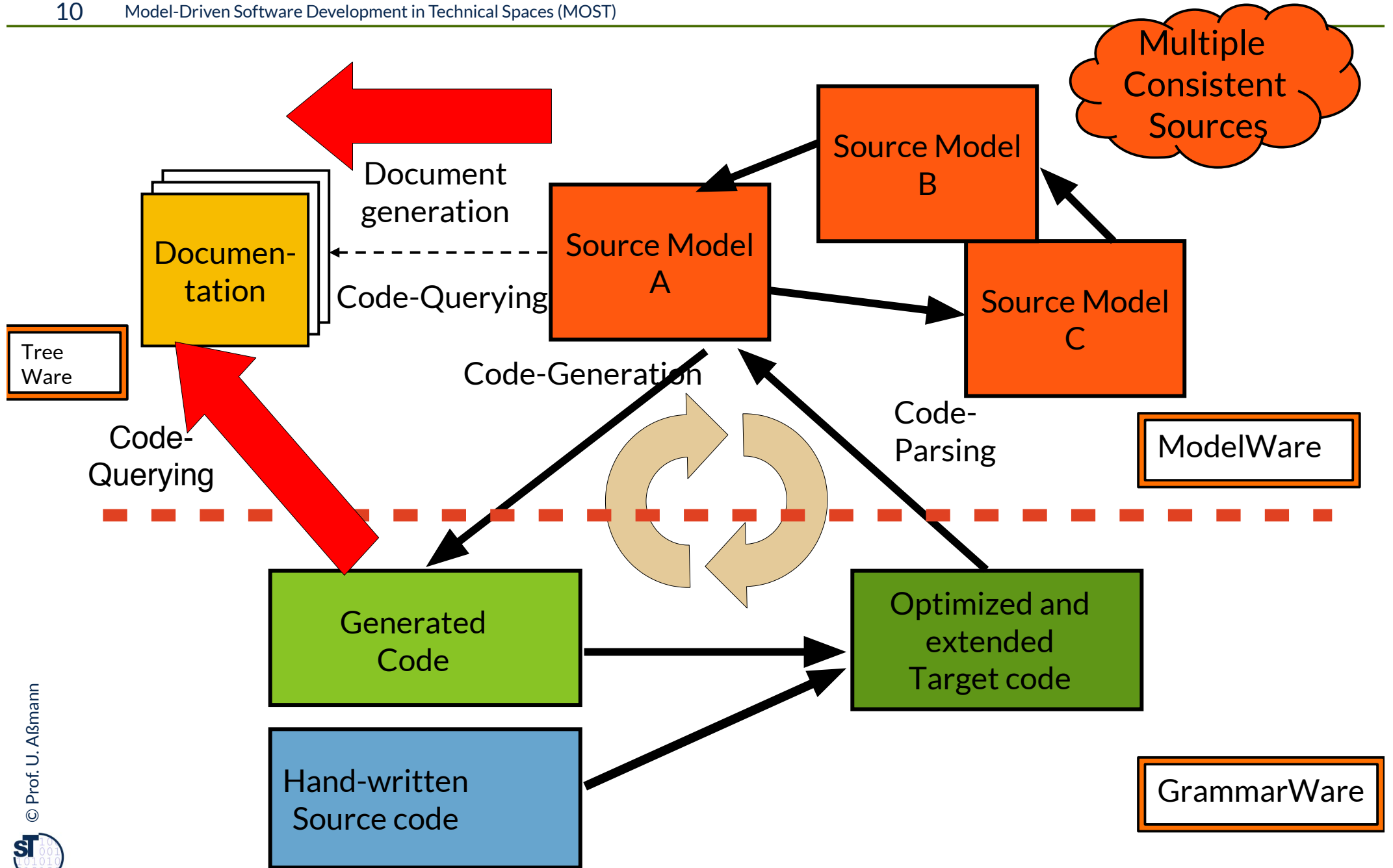
- ▶ Basically, documentation generation is similar to code generation. Documentation is created in higher-order attributes on a link tree by a RAG
- ▶ **Documentation generation is an application areas for RAG**
- ▶ **Generation** of derived documents from code and models
 - Generation of Word (docx), LibreOffice (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 (with the hedge-principle)
- ▶ **Parameterization** with layouts
 - via css-style sheets



31.2 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 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 + methods.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 **hedges** (Trennmarkierung)

Generated Wrapper

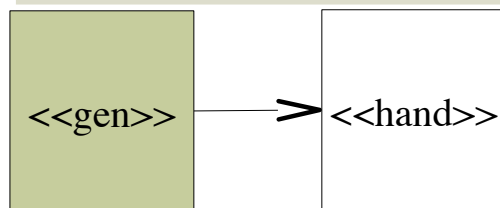
```
/** Generated documentation  
***/
```

```
/** Hedge **/
```

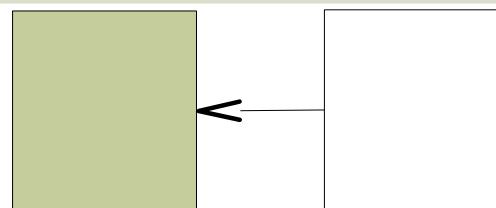
```
... Hand-written  
Documentation ...
```

```
/** Hedge **/
```

Generated Delegator



Generated Delegatee

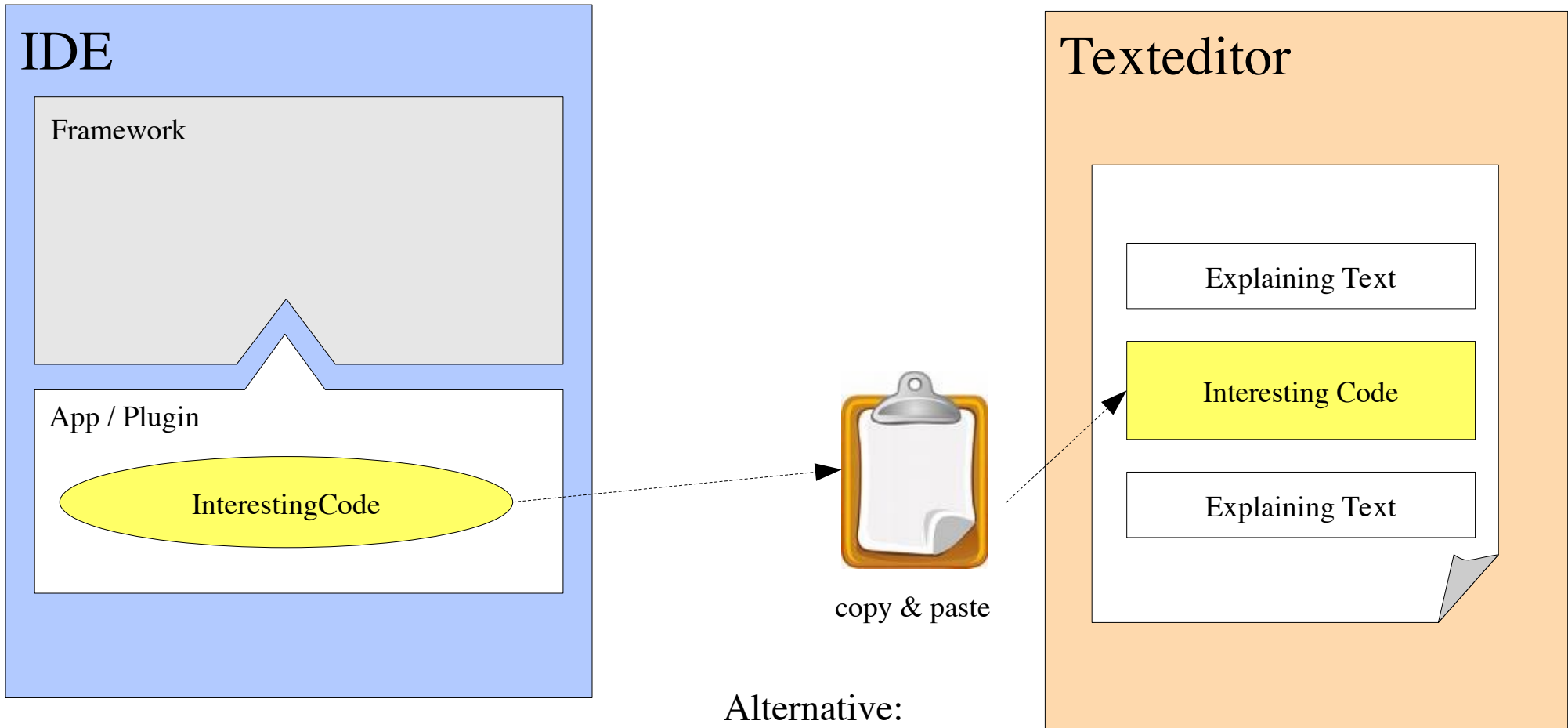




31.3 Literate Programming

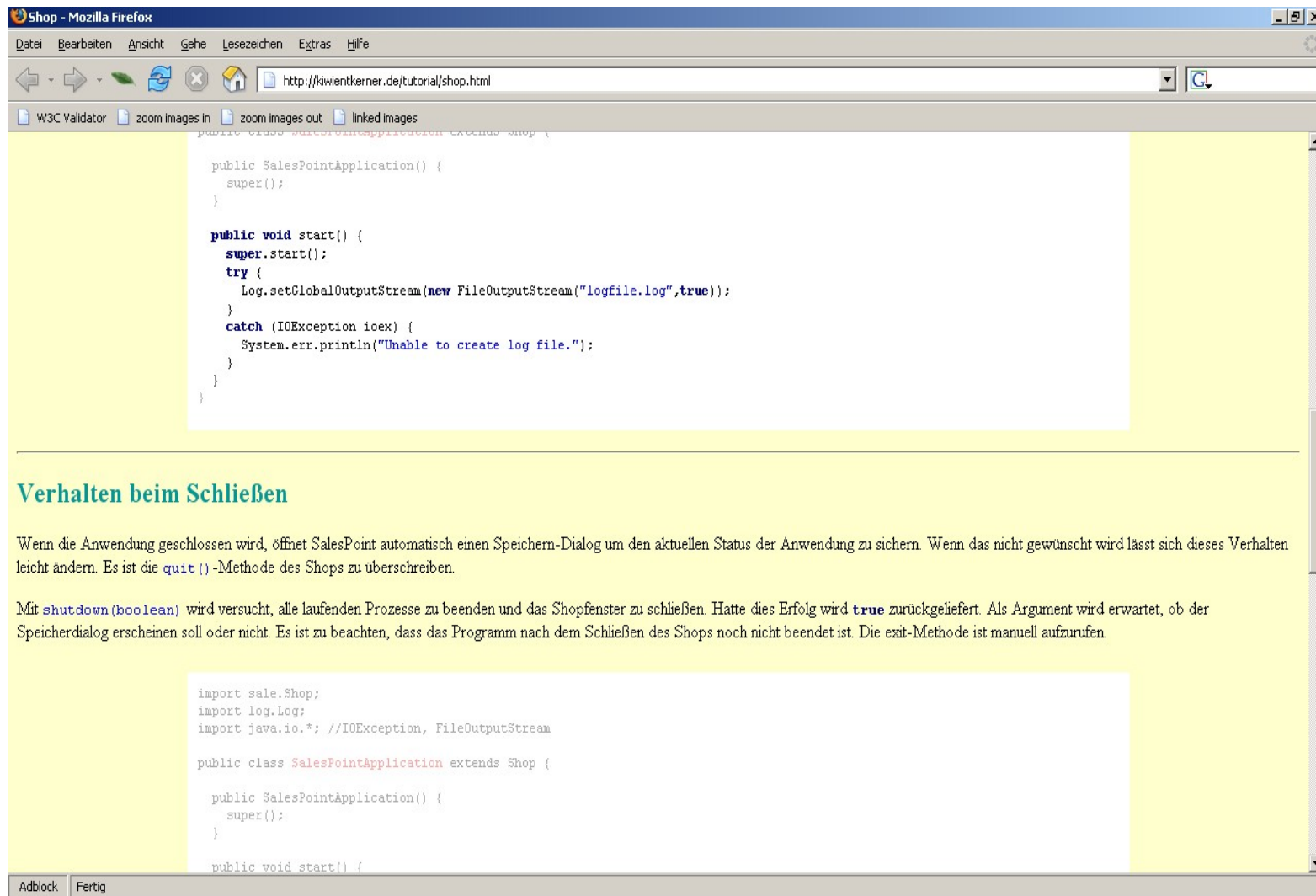
- They integrate code, models and documentation by **separating code from documentation**

Classic: Manual Writing of Tutorials



Alternative:
Code query
e.g., with
Xcerpt or QL

How to Write Integrated Documentation and Tutorials?



The screenshot shows a Mozilla Firefox browser window titled "Shop - Mozilla Firefox". The address bar contains the URL "http://kiwientkerner.de/tutorial/shop.html". The page content is a tutorial for a Java application named "Shop".

```
public class SalesPointApplication extends Shop {  
  
    public SalesPointApplication() {  
        super();  
    }  
  
    public void start() {  
        super.start();  
        try {  
            Log.setGlobalOutputStream(new FileOutputStream("logfile.log",true));  
        }  
        catch (IOException ioex) {  
            System.err.println("Unable to create log file.");  
        }  
    }  
}
```

Verhalten beim Schließen

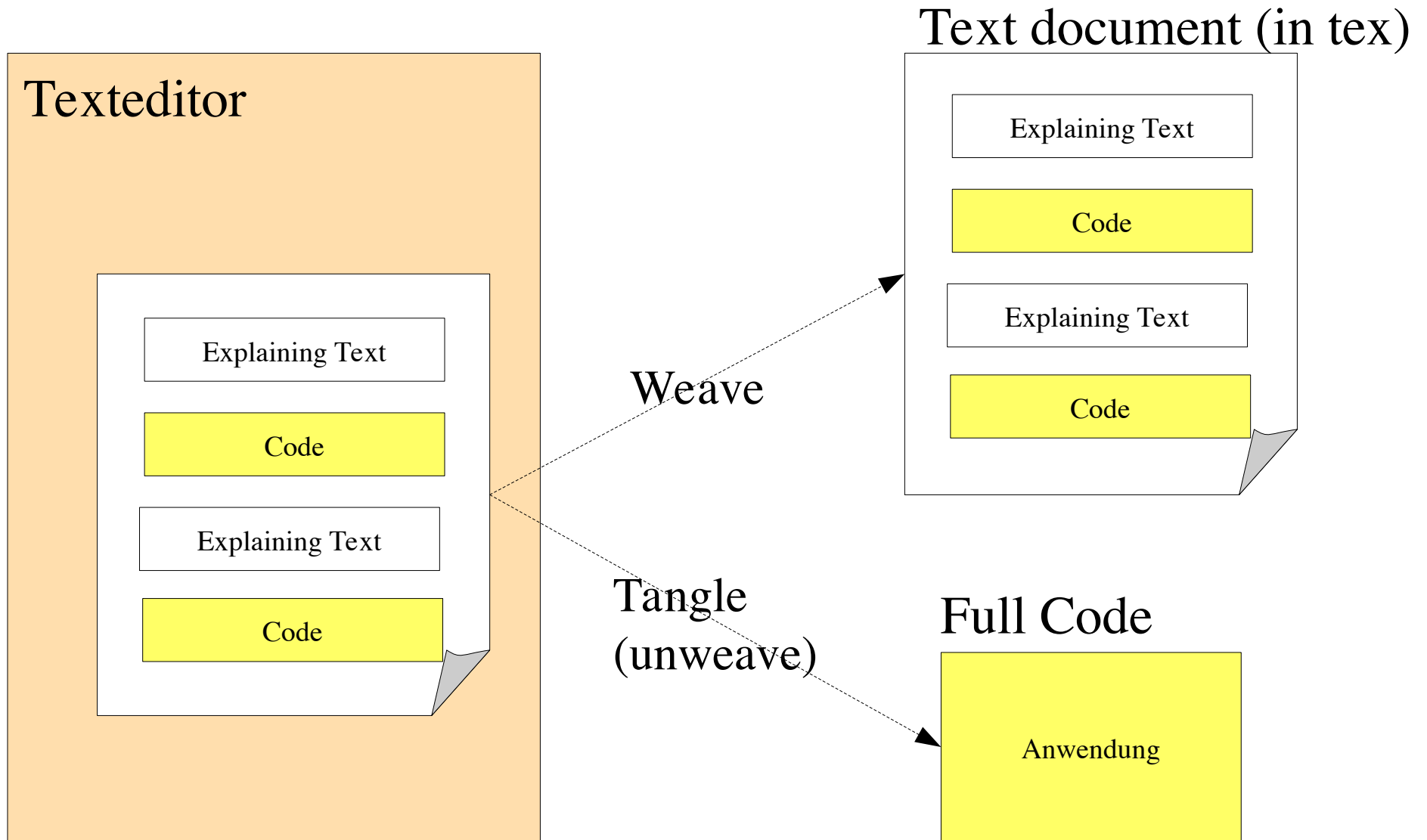
Wenn die Anwendung geschlossen wird, öffnet SalesPoint automatisch einen Speichern-Dialog um den aktuellen Status der Anwendung zu sichern. Wenn das nicht gewünscht wird lässt sich dieses Verhalten leicht ändern. Es ist die `quit()`-Methode des Shops zu überschreiben.

Mit `shutdown(boolean)` wird versucht, alle laufenden Prozesse zu beenden und das Shopfenster zu schließen. Hatte dies Erfolg wird `true` zurückgeliefert. Als Argument wird erwartet, ob der Speicherdialog erscheinen soll oder nicht. Es ist zu beachten, dass das Programm nach dem Schließen des Shops noch nicht beendet ist. Die `exit`-Methode ist manuell aufzurufen.

```
import sale.Shop;  
import log.Log;  
import java.io.*; //IOException, FileOutputStream  
  
public class SalesPointApplication extends Shop {  
  
    public SalesPointApplication() {  
        super();  
    }  
  
    public void start() {
```

At the bottom of the browser window, there are buttons for "Adblock" and "Fertig".

[Knuth] 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.
```

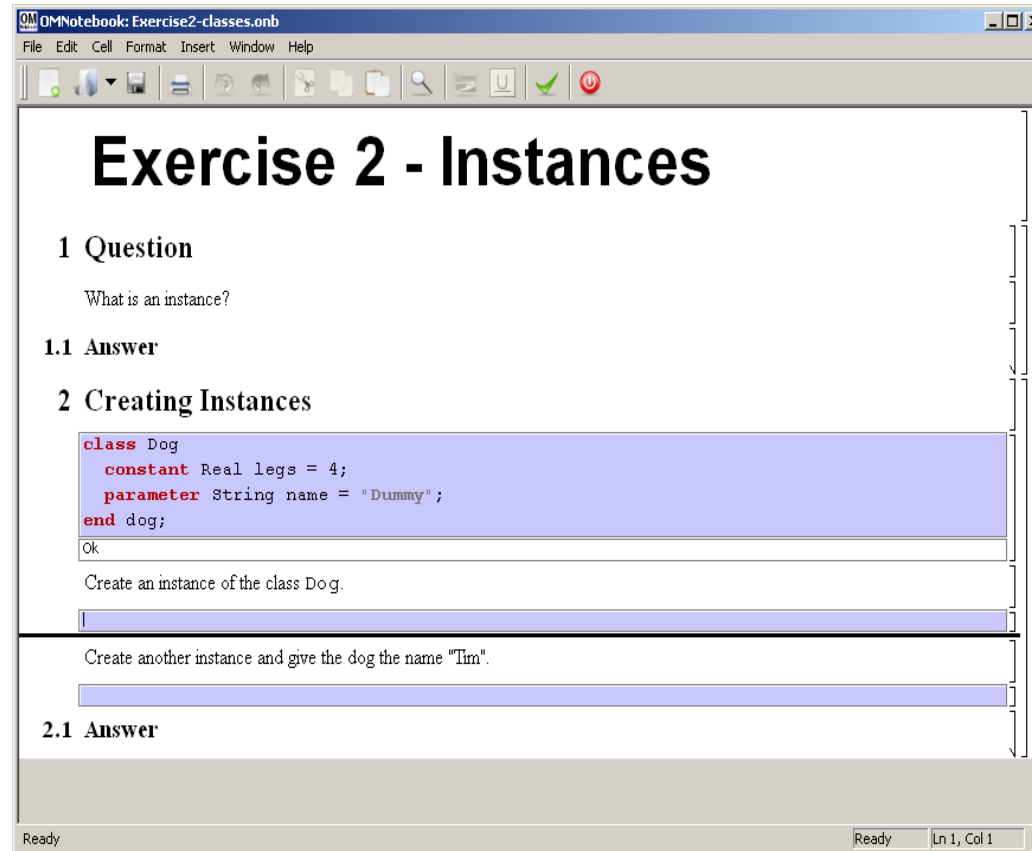
[Literate Programming
von Donald E. Knuth]

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

OMNotebook – Literate Programming with DrModelica

19

Model-Driven Software Development in Technical Spaces (MOST)



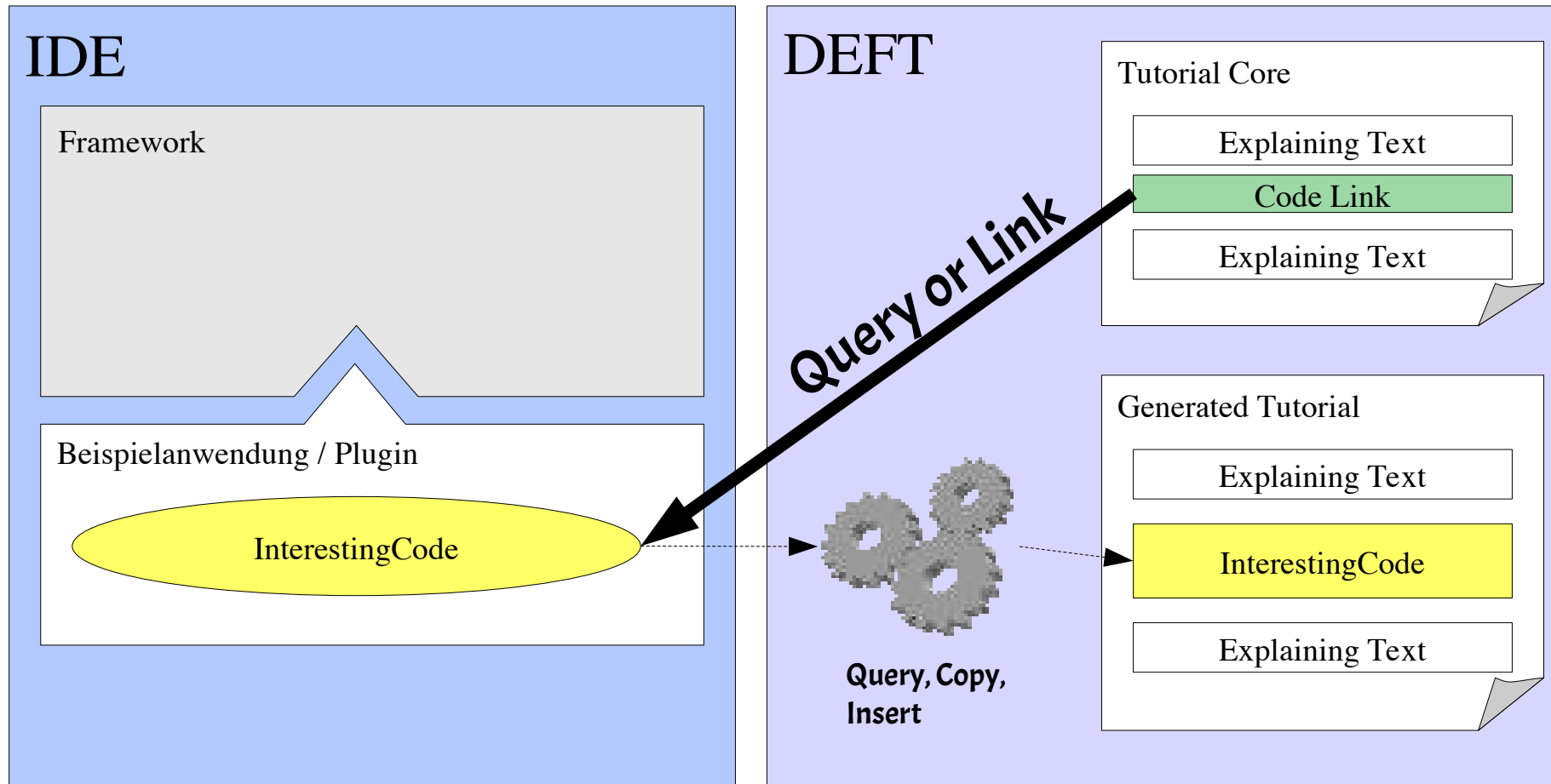
- ▶ Linked documents with interactive exercises
- ▶ Inspired by DrScheme und DrJava, learning tools for Scheme resp. Java
- ▶ www.openmodelica.org



31.4 Elucidative Documentation Tools

- They link code, models and documentation by **model and code mapping**
- **and renew the documentation by *hot updates***

Elucidative Programming Links Documentation with Queries to Code



hot update
(hot synchronisation):
round-trip engineering



Elucidative Programming

The screenshot shows the 'Scheme Elucidator' environment. At the top, there is a toolbar with icons for file operations and a 'time general' button. Below the toolbar, there are three sections: '1 Introduction', '2 The solution', and '3 Post Scriptum'. The '1 Introduction' section is highlighted in blue and contains the text: 'In this introductory section we first discuss time system formats and a function in Scheme which returns the current time. Then we discuss the issue of normalization and two possible ways to attack the problem.' Below this, there are sub-sections '1.1 Time systems and functions' and '1.2 The plan of attack'. The '1.1 Time systems and functions' section is also highlighted in blue and contains the text: 'There are several different standards for representation of time on a Computer. Universal'. To the right of the documentation, there is a code editor showing Scheme code for defining constants and functions. The code includes:

```
;; Fixed second counts and calendar facts.
(define seconds-in-a-normal-year 31536000)
(define seconds-in-a-leap-year 31622400)
(define seconds-in-a-week 604800)
(define seconds-in-a-day 86400)
(define seconds-in-an-hour 3600)
(define base-year 1970)
(define month-length-normal-year
  (vector 31 28 31 30 31 30 31 31 30 31 30 31))
```

„Scheme Elucidator“ Environment

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

hot update
(hot synchronisation):
round-trip engineering

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

23

Model-Driven Software Development in Technical Spaces (MOST)

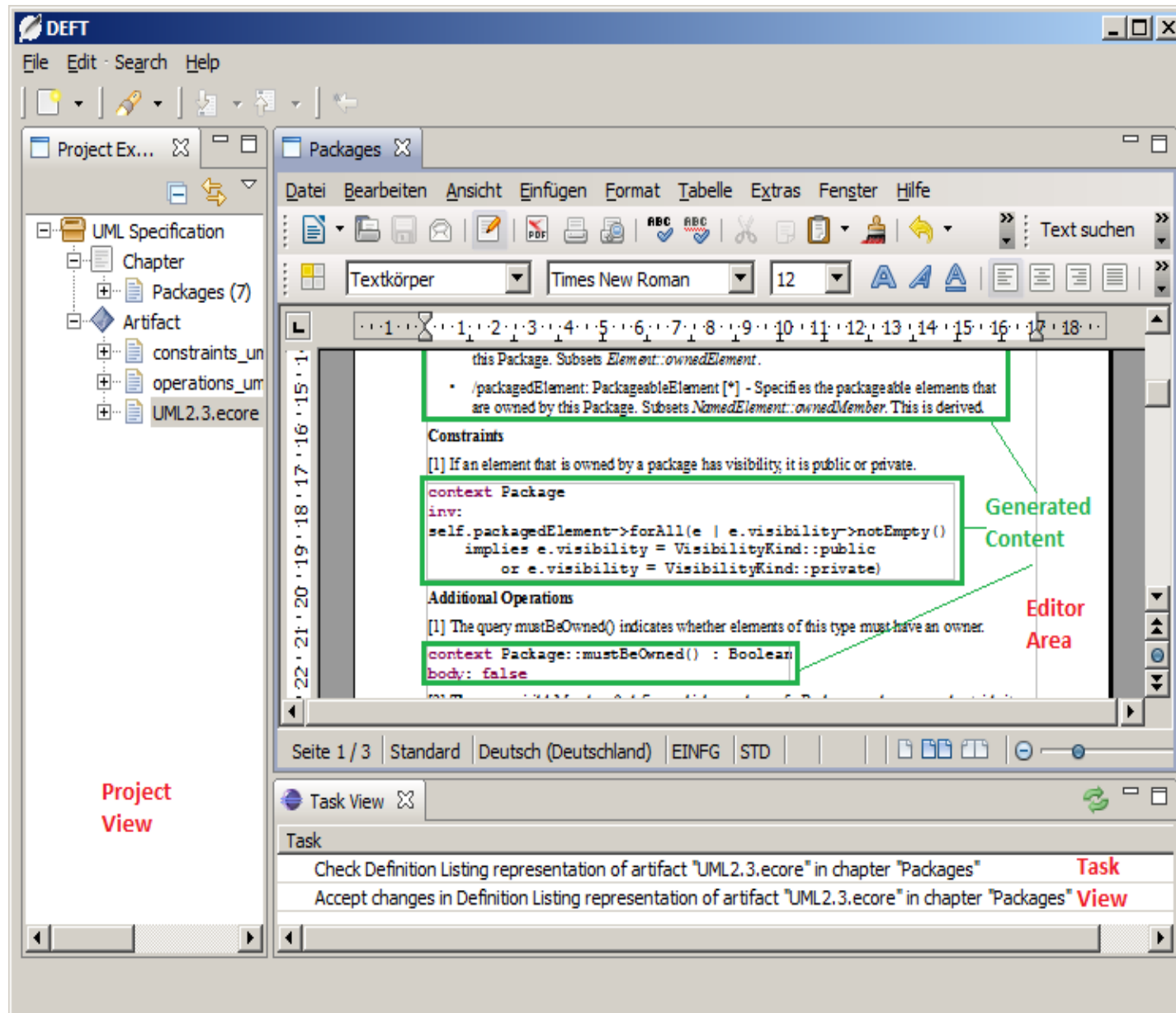
The screenshot displays the DEFT development environment with the following components:

- Project Explorer (Projektübersicht):** Shows a project named 'Fahrkartenautomat' with a tree structure including 'Chapters', 'Code files', 'Code snippets', 'Images', and 'Tutorials'. The 'Code files' folder is expanded to show files like 'FahrkartenAutomat.cs', 'Fahrerschein.cs', 'SecurityIO.cs', etc.
- Texteditor:** Displays the source code for 'FahrkartenAutomat.cs'. It contains a text block: 'Um Logging zu verwenden, muss ein Stream auf die Logdatei geöffnet werden.' followed by a **Eingebettetes Codefragment** (Embedded Code Fragment) in a yellow box:

```
try
{
    FileStream fs = new FileStream("test.log",
        FileMode.Create);
    Log.Log.GlobalOutputStream = new
        System.IO.BinaryWriter(fs);
}
catch (IOException)
{
    System.Console.Error.WriteLine("Unable to
        create Log file.");
}
return ;
```
- Editor Outline (Kapitelstruktur):** Shows a simple tree structure for the application, starting with 'Start' and 'Start der Anwendung'.
- Code Outline (AST-Fenster):** Shows the Abstract Syntax Tree (AST) for the code, including 'UsingDirectives', 'FahrkartenAutomat', 'TICKETS', 'MONEY', 'DefaultMenuSheet', 'FahrkartenAutomat(string)', 'createDisplayManager()', 'Main(string[])', 'FahrerscheinVerkaufenAction', and 'WartungAction'.

Additional text in the image includes 'Texteditor' at the bottom right of the code editor and 'AST-Fenster' at the bottom right of the code outline window.

Embedding UML Constraints for UML Models into Documentation



Development Environment For Tutorials (DEFT)

- ▶ 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
 - Notification if code fragments have changed

Start der Anwendung

In der Klasse `FahrkartenAutomat` befindet sich die `Main`-Methode, mit der sich das Programm starten lässt. Dort werden Daten initialisiert und der Fahrkartenautomat instanziiert.

Logging

Der erste Schritt ist die Konfiguration des Loggings. Das SalesPoint-Framework bietet Funktionen und Datentypen an, mit denen Aktionen geloggt werden können. Es gibt GUI-Komponenten, mit denen die Inhalte des Logs wieder nutzerfreundlich angezeigt werden können. Eine Anzeige des Logs ist derzeit nicht im Fahrkartenautomaten implementiert, geloggt wird aber trotzdem schon.

Um Logging zu verwenden, muss ein Stream auf die Logdatei geöffnet werden.

```
try
{
    FileStream fs = new FileStream("test.log", FileMode.Create);
    Log.Log.GlobalOutputStream = new System.IO.BinaryWriter(fs);
}
catch (IOException)
{
    System.Console.Error.WriteLine("Unable to create Log file.");
    return ;
}
```

Es wird versucht, die Datei `test.log` zu erzeugen. Falls sie schon existiert, wird sie überschrieben. Der `FileStream fs` kann Daten (Bytes) vom Programm in die Datei schreiben. Byteweises Schreiben von Informationen ist allerdings sehr umständlich. Ein `BinaryWriter` kapselt den `FileStream` und bietet Methoden zum Schreiben von Strings, Zahlen, und Anderem. Der globale Log-Klasse der Anwendung, `Log.Log`, wird dieser `BinaryWriter` zugewiesen. Alle

```
{
}

protected override DisplayManager createDisplayManager()
{
    Size d = System.Windows.Forms.Screen.PrimaryScreen.Bounds.Size;
    Point tempAux = new Point((d.Width - 100) / 2, (d.Height - 80) / 2);
    Point tempAux2 = new Point(5, 5);
    return new AWTDisplayManager(this, ref tempAux, ref tempAux2);
}

[STAThread]
public static void Main(string[] args)
{
    //System initialisieren
    try
    {
        FileStream fs = new FileStream("test.log", FileMode.Create);
        Log.Log.GlobalOutputStream = new System.IO.BinaryWriter(fs);
    }
    catch (IOException)
    {
        System.Console.Error.WriteLine("Unable to create Log file.");
        return ;
    }

    // Kataloge anlegen

    // Fahrscheinkatalog
    Catalog cTickets = Catalog.forName(TICKETS);

    cTickets.addItem(new Fahrschein("Einzelfahrt", 300));
    cTickets.addItem(new Fahrschein("Sammelfahrschein", 1500));
    cTickets.addItem(new Fahrschein("ermäßigte Einzelfahrt", 150));
}
```





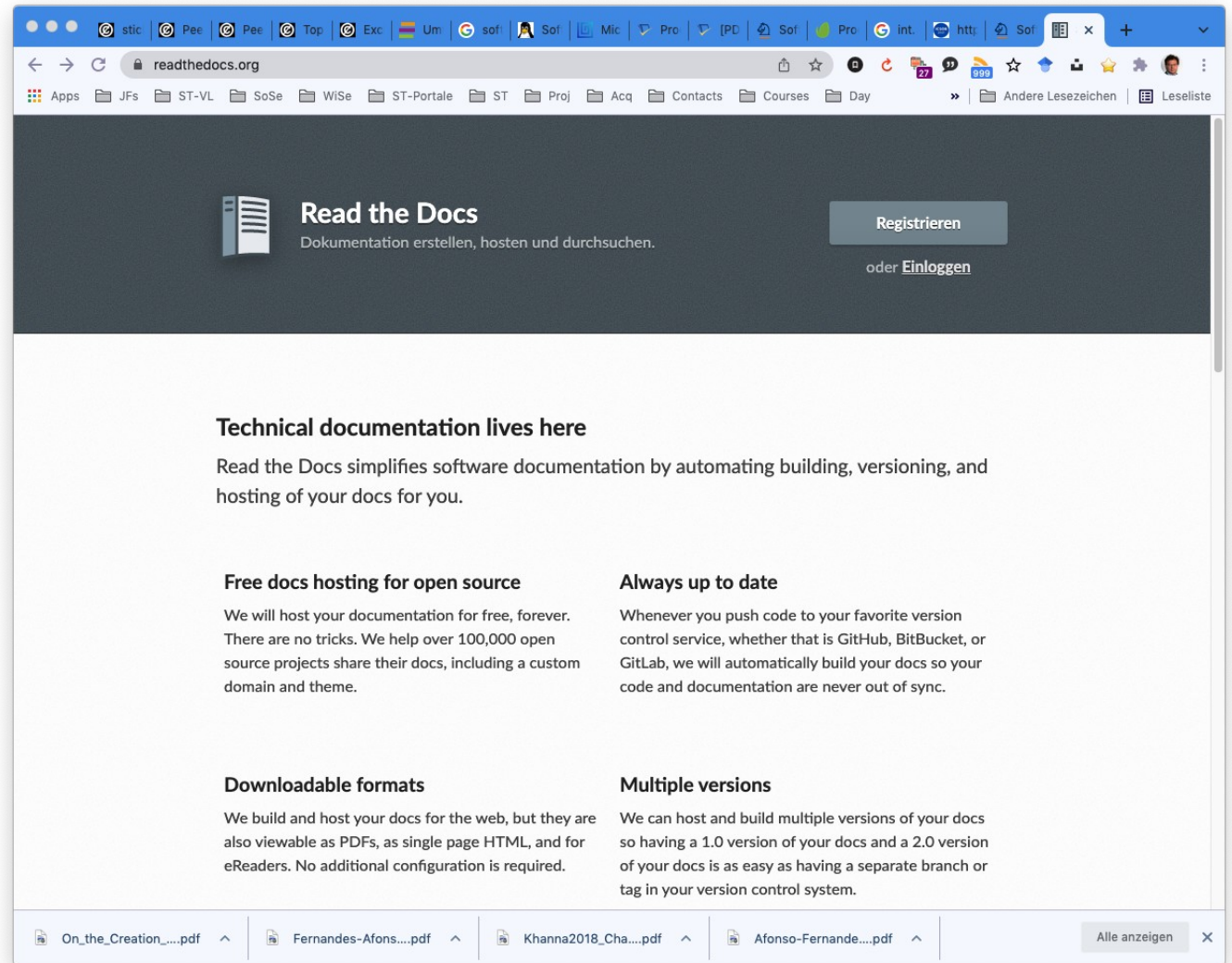
31.5 Web-based Documentation Generators based on Markdown

Sphinx and the Documentation Cloud readthedocs.org

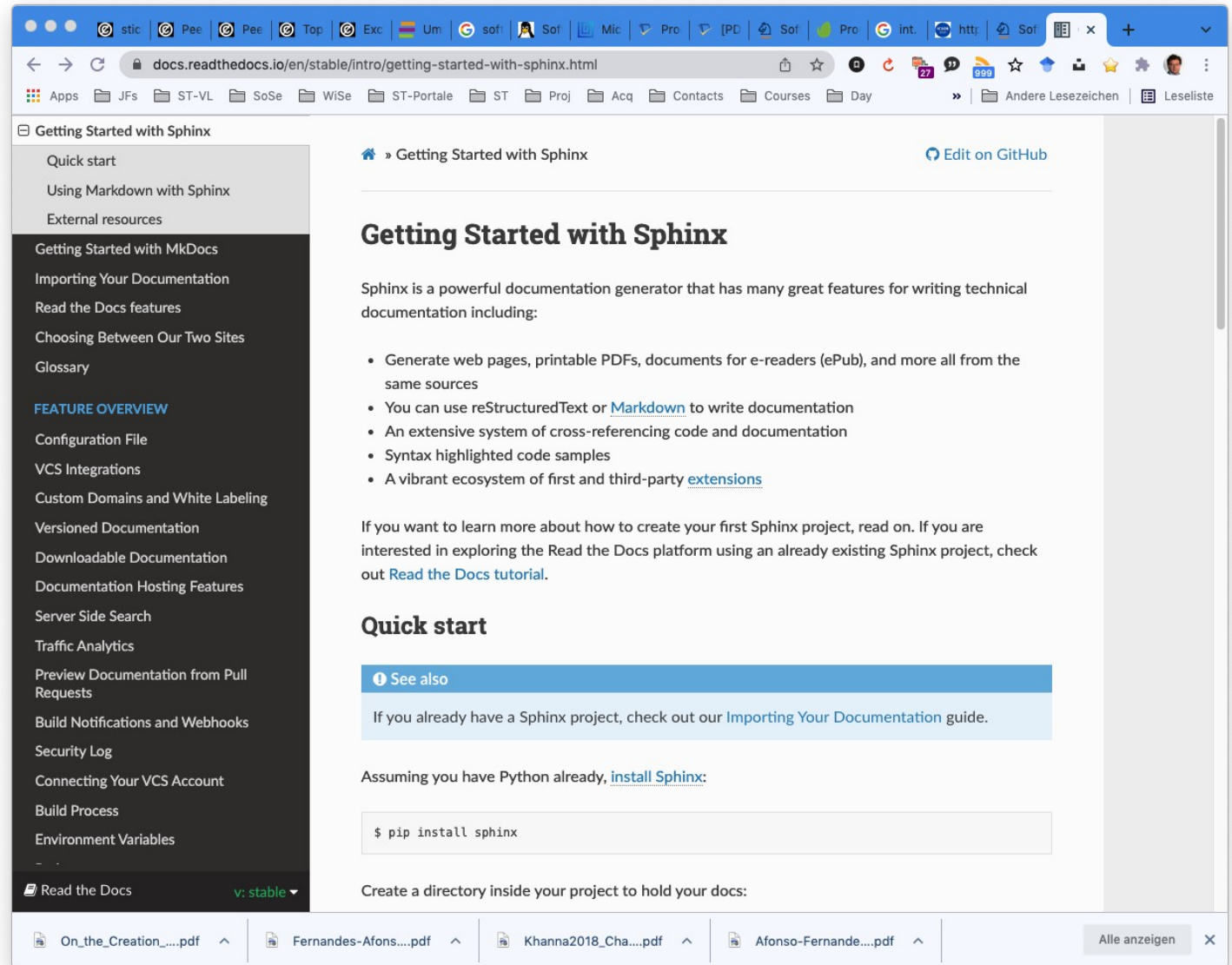
28

Model-Driven Software Development in Technical Spaces (MOST)

- ▶ readthedocs is a cloud for documentation projects
- ▶ supporting two documentation generators *sphinx* and *mkdocs*

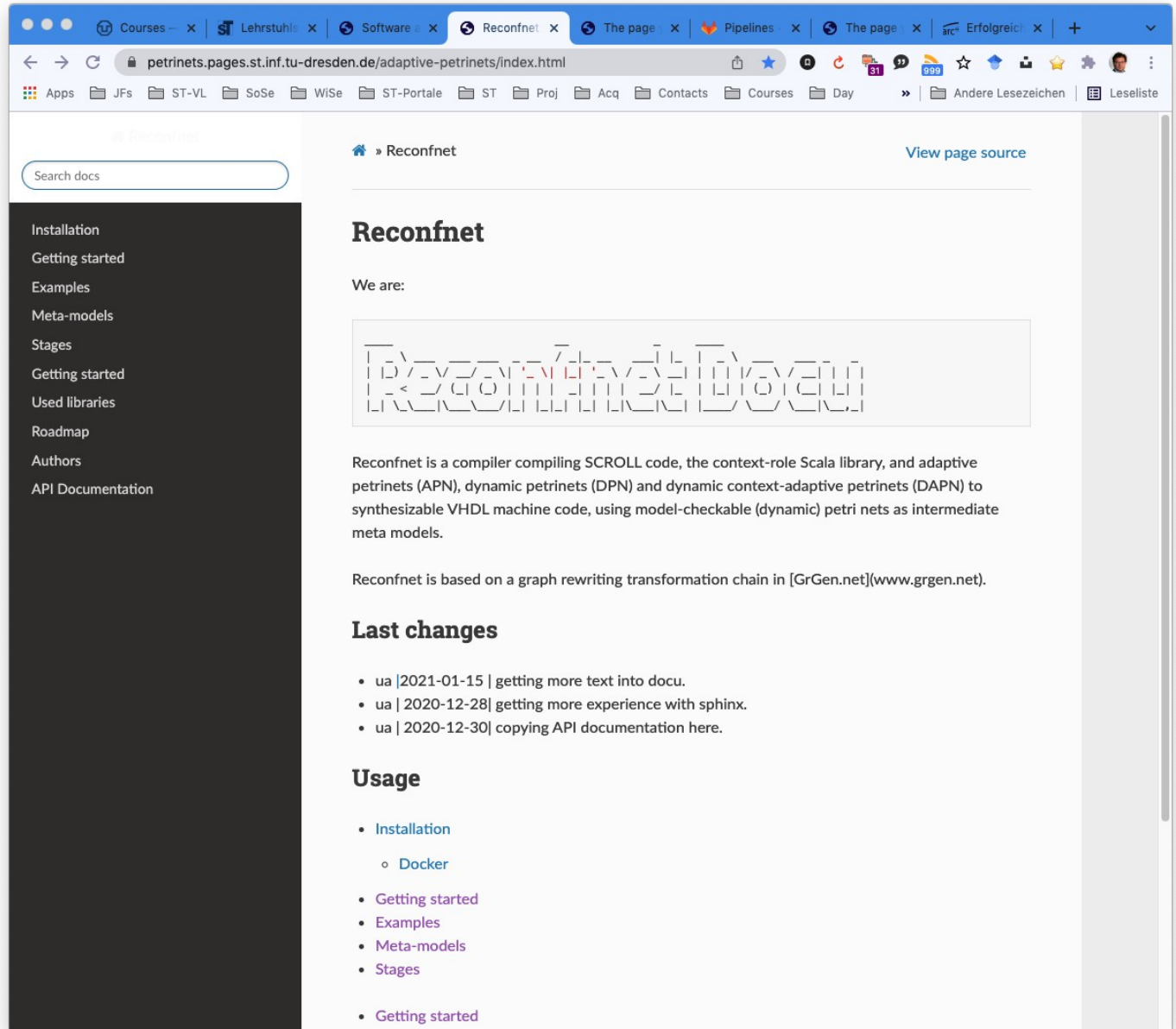


- ▶ Architecture documentation
- ▶ User documentation
- ▶ Files in formats reStructuredText and Markdown are transformed to HTML
- ▶ Treats entire directories
- ▶ many output formats (e.g., Latex)
- ▶ Can be coupled with Javadoc or similar API doc generators

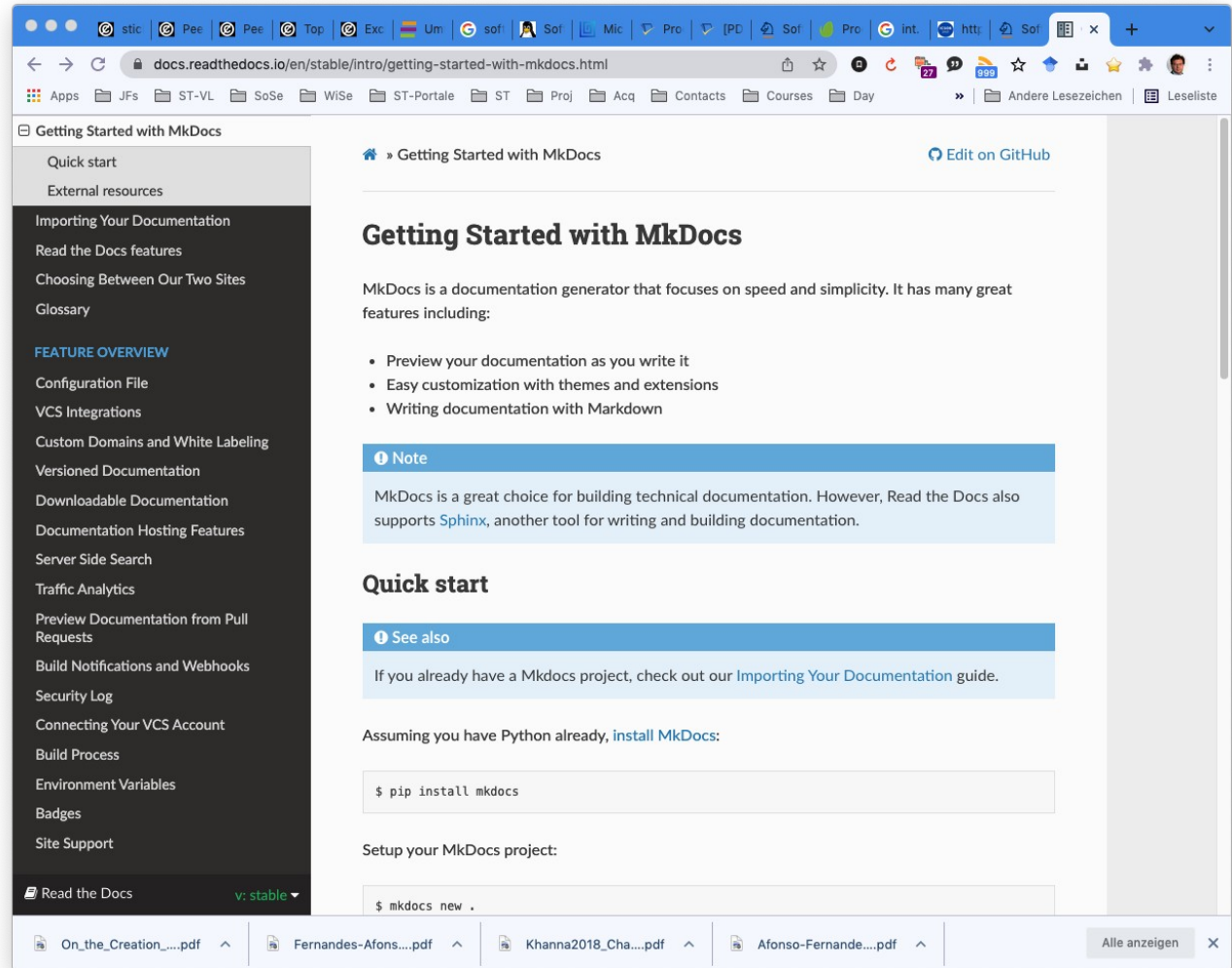


Example Sphinx Project

- ▶ Petrinet compiler
Reconfnet
<https://petrinets.pages.st.inf.tu-dresden.de/adaptive-petrinets/index.html>

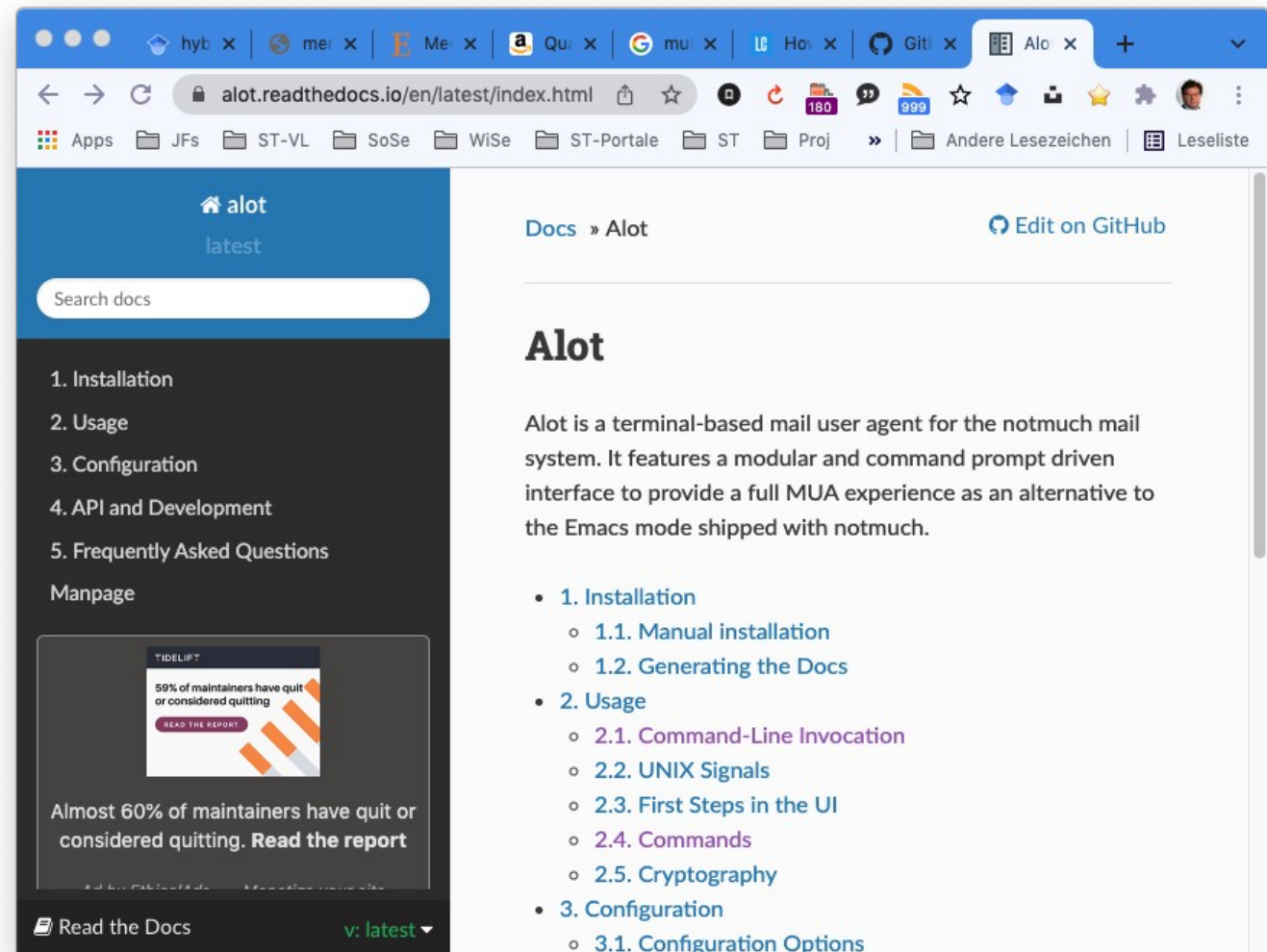


- ▶ Markdown files to HTML files
- ▶ several output formats



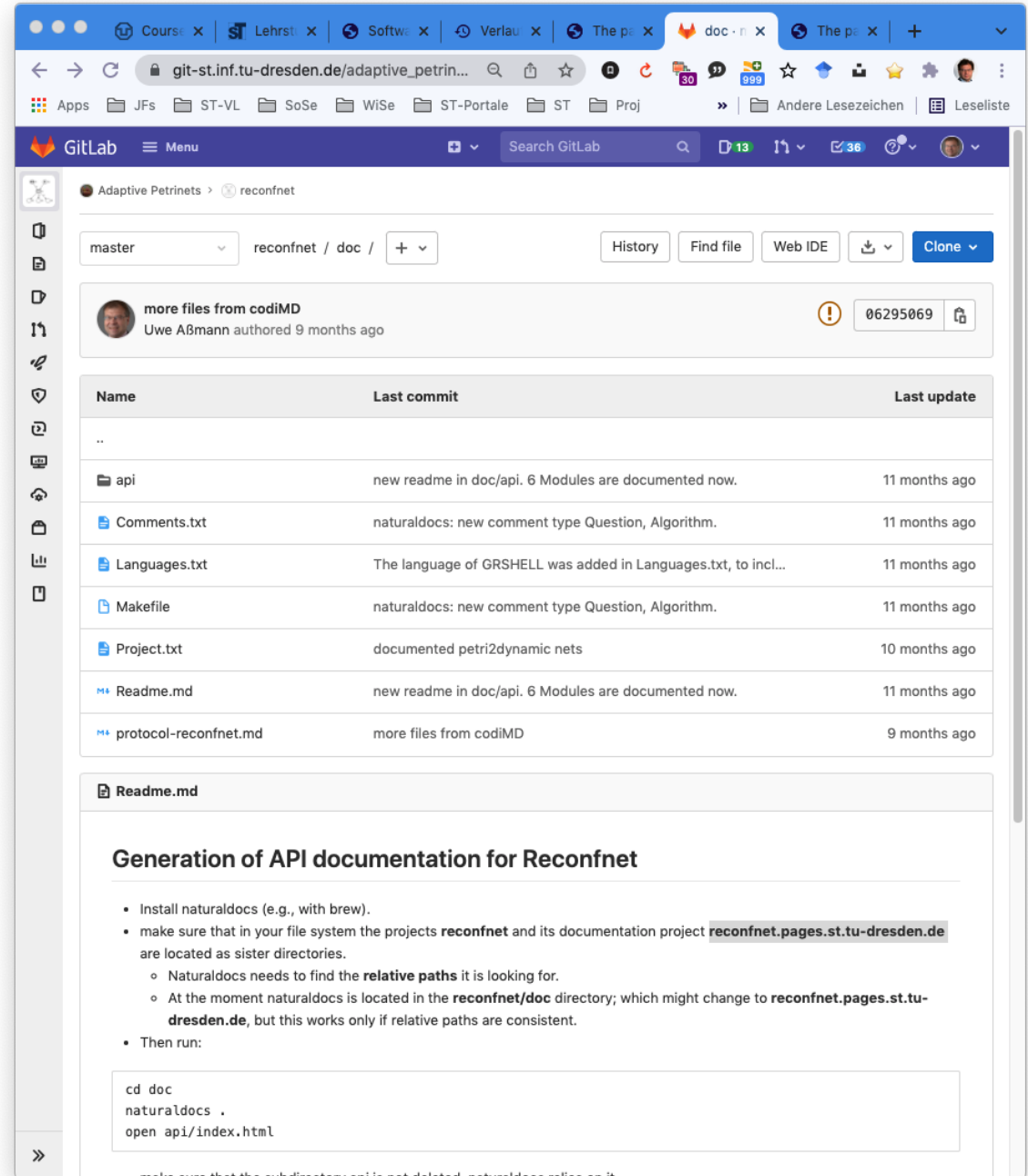
Alot – a Mail User Agent Documented on readthedocs

- ▶ <https://alot.readthedocs.io/>



NaturalDocs Generic API Documentation Generator

- ▶ Similar to JavaDoc, but more than 20 languages
- ▶ own keywords can be defined
- ▶ Example gitlab project from which API documentation for GrGen can be generated
 - https://git-st.inf.tu-dresden.de/adaptive_petrinets/reconfnet/-/tree/master/doc



Example NaturalDocs API generated for GrGen

- ▶ GrGen.net is a generator for graph rewrite specifications (see Part IV)
- ▶ There is no specific API doc generator for GrGen, but NaturalDocs can be tailored to it

Reconfnet Compiler
for Dynamic Adaptive Petrinets, Version 0.6.3

context / ContextDependencyModel.gm

This defines the types for contexts in Dynamic Adaptive Petrinets.

PROPERTIES

Author: CM. Commented by UA (2020-12-29)

We need two approaches, vertical and horizontal extension/reconfiguration.

GRAPHTYPES

Weak Inclusion

empty triangle on activation/deactivation trigger this with target $act(source) \rightarrow act(target) deact(source) \rightarrow deact(target) act(target) \rightarrow deact(target) \rightarrow$

Strong Inclusion

when target gets deactivated source also empty triangle $act(source) \rightarrow act(target) deact(source) \rightarrow deact(target) act(target) \rightarrow deact(target) \rightarrow deact(source) \rightarrow deact(target)$

requirement

can only be activated when target is already empty triangle $act(source) \rightarrow$ only if already: $act(target) deact(source) \rightarrow act(target) \rightarrow deact(target) \rightarrow deact(source)$

exclusion

empty boxes both can not be active at same time $act(source) \rightarrow deact(target) act(target) \rightarrow deact(source)$

arrow



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?



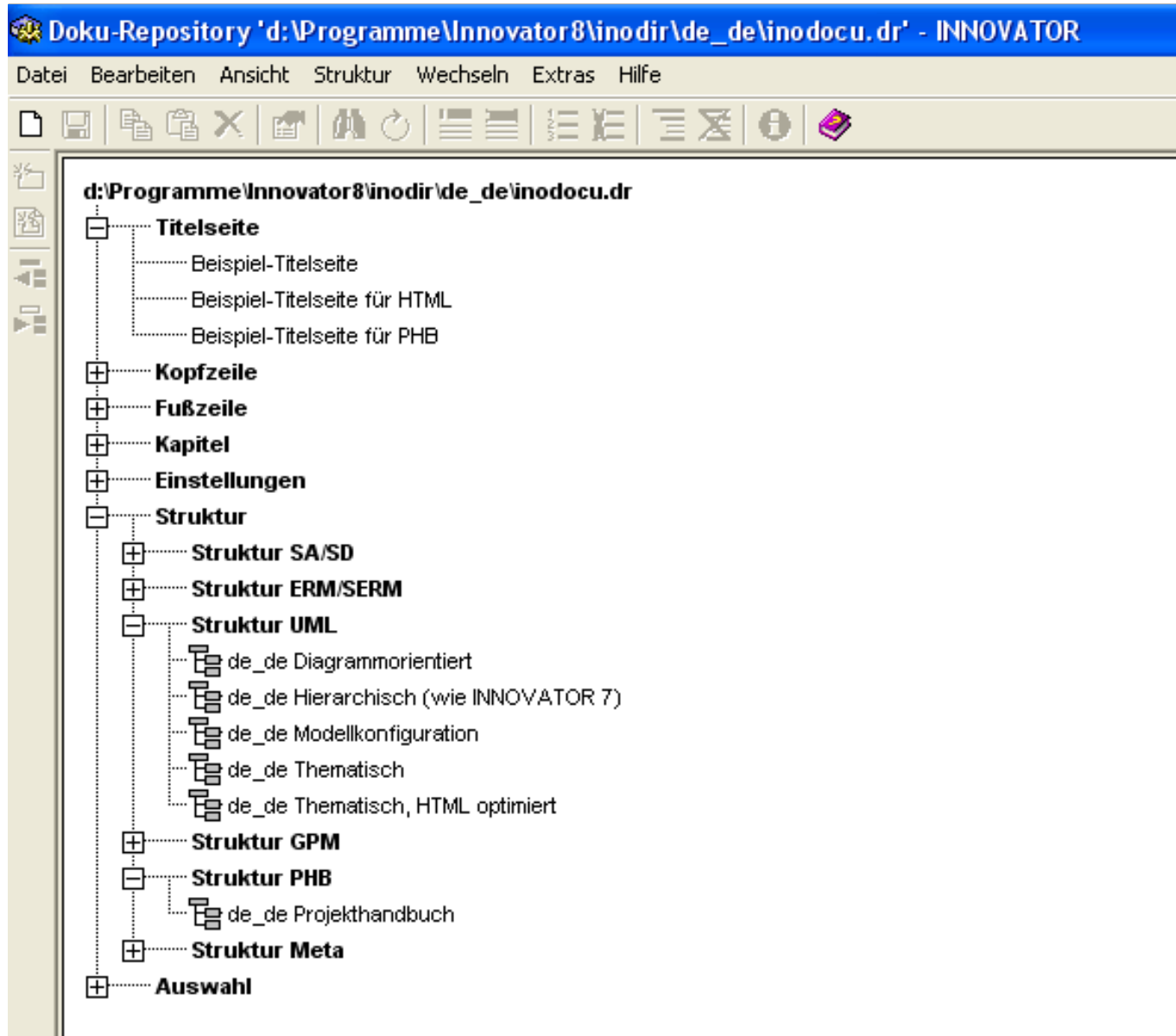
A.1 Other Template Expanders for Documentation Generation

Documentation Tools of MID Innovator

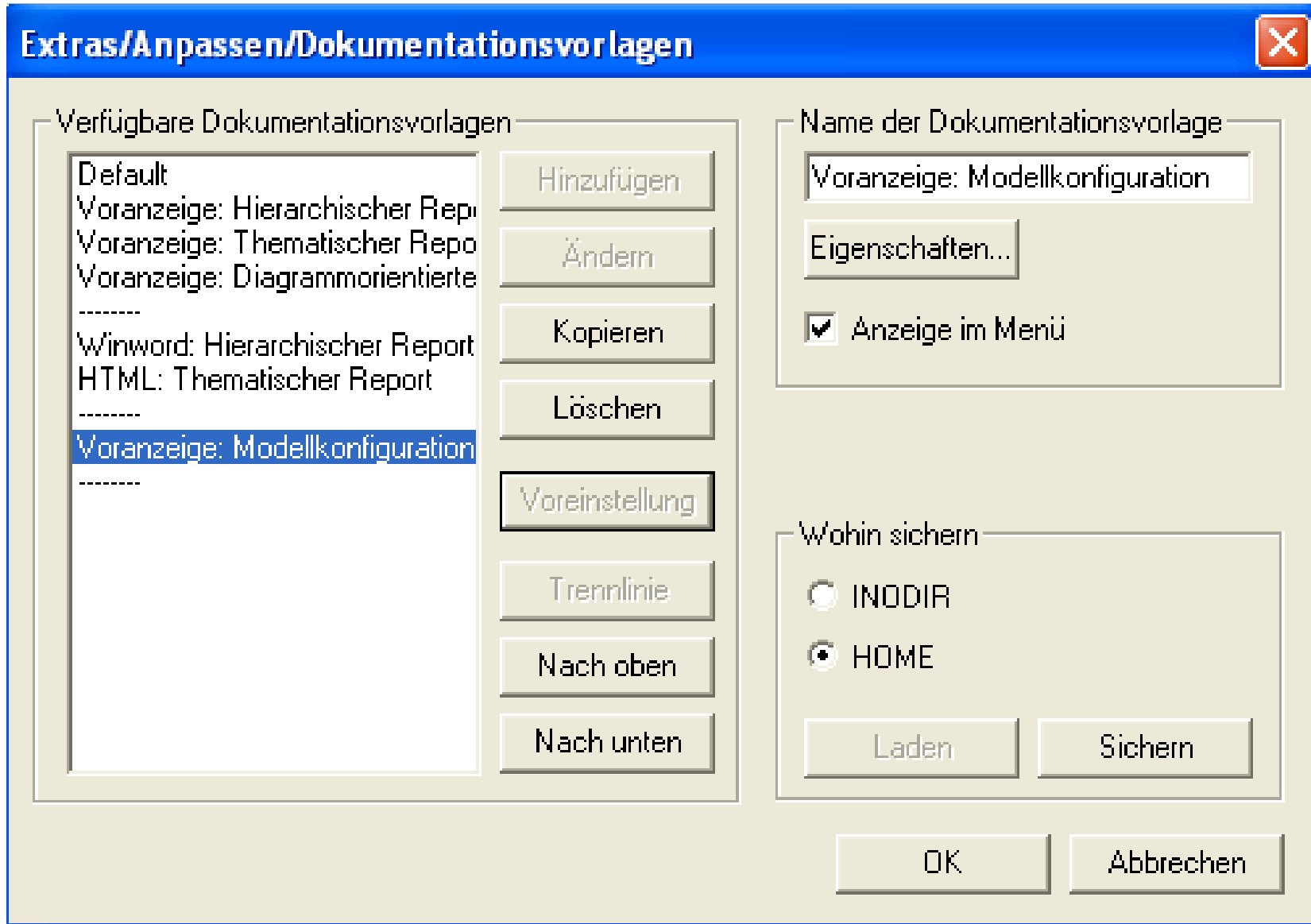
- ▶ Innovator provides documentation templates, into which diagrams, models, code can be embedded
- ▶ Several formats:
 - pdf
 - Word
 - ASCII
 - XML



Ex.: Innovator Documentation Template (Dokumentationsvorlage)



Ex.: Innovator Documentation Template (Dokumentationsvorlage): Adaptation



Innovator - Generated Example Word Document

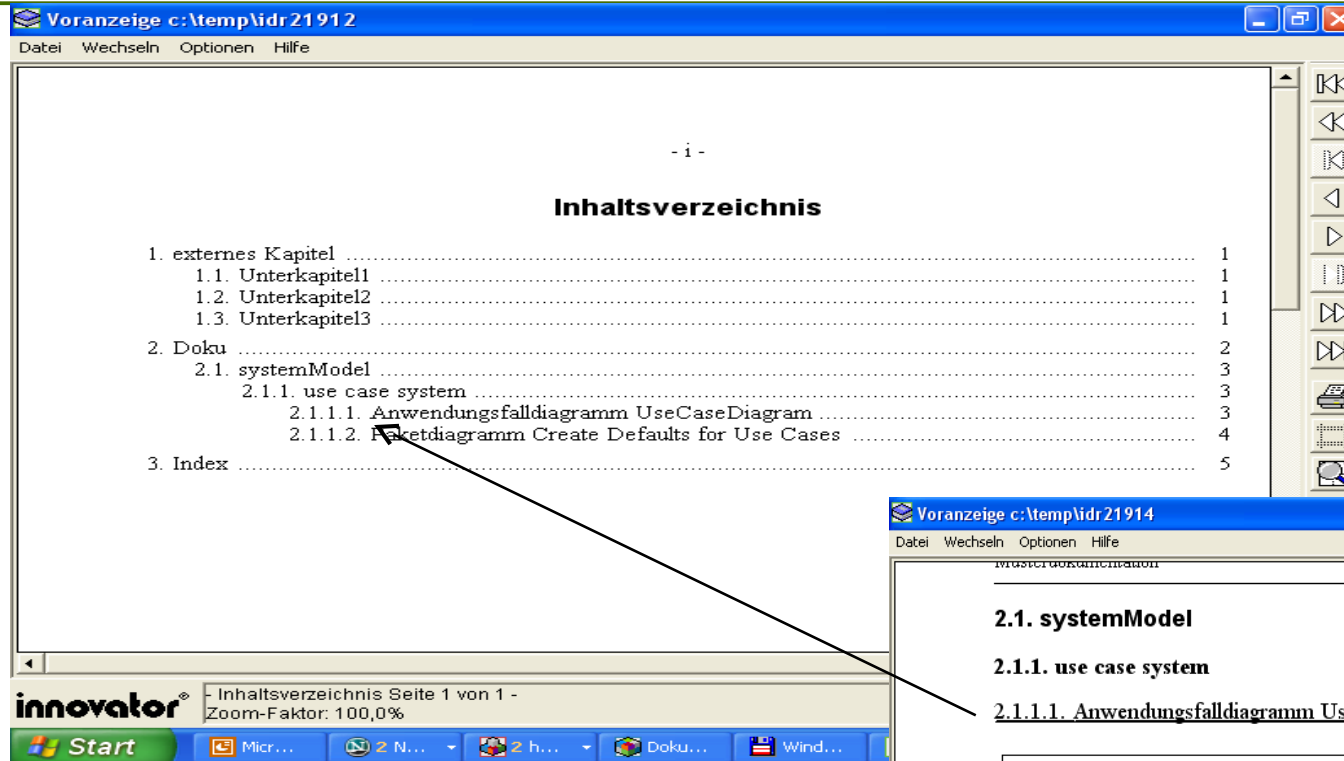
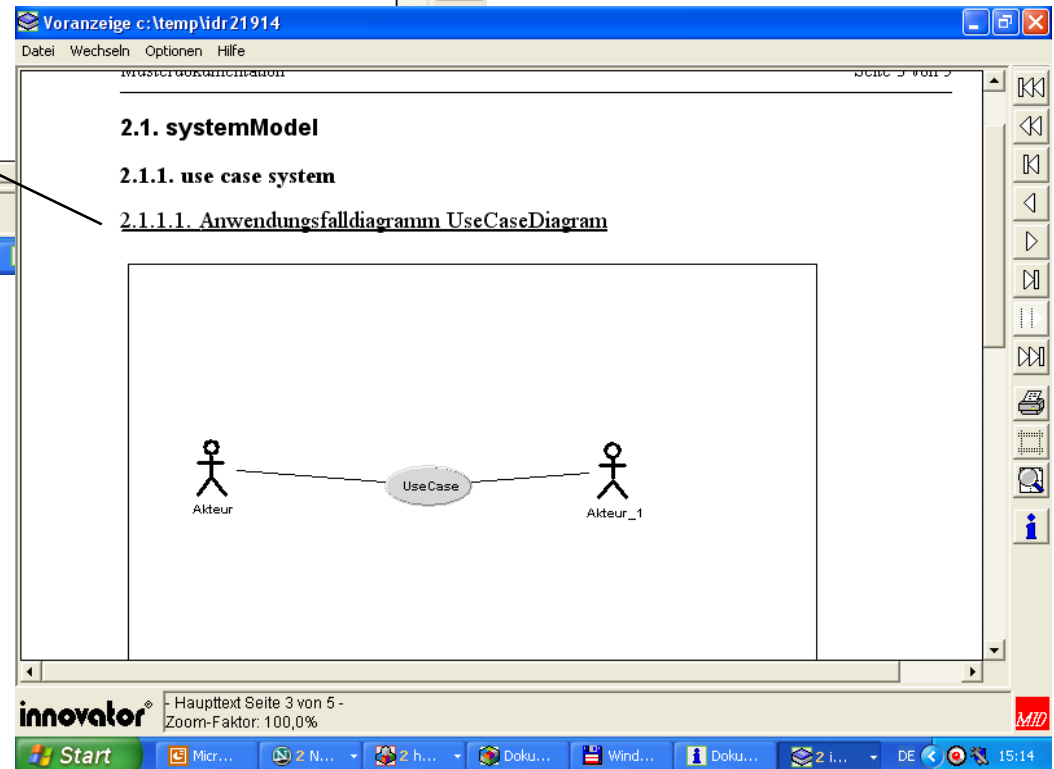


Table of Contents



Integration of a Use Case Diagram
(section 2.1.1.1.)

Index is generated

