

72. Dokumentationswerkzeuge

1

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 12-0.2, 31.01.13

- 1) Aufgaben
- 2) Templategesteuerte Dokumentationswerkzeuge
- 3) Elucidative Werkzeuge

In 2012/13 weggelassen
nur zur Info



Softwareentwicklungswerkzeuge (SEW) © Prof. Uwe Aßmann

46.1 Aufgaben der Dokumentationswerkzeuge

2

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



Softwareentwicklungswerkzeuge (SEW) © Prof. Uwe Aßmann

Ziel der Softwaredokumentation

3

- ▶ Dokumente dienen dem Know-how-Transfer zwischen AG und AN sowie innerhalb des Entwicklerteams
 - Sie retten und bewahren das Wissen über Programme, sind aktuell zu halten und dienen als Quelle für Personen, die die Software weiterentwickeln und warten müssen.
 - Sie dienen als Basis für die Durchführung von Prüfungen (Tests, Audits usw.)
 - Dienen zur Messung des Projektfortschritts (Terminerefüllung, Meilensteine)
 - ▶ Basis der Anwendung von Regeln für die Einhaltung von Standards und der Revisionssicherheit.
Man unterscheidet:
 - ▶ **Separate Dokumentation**, die nicht direkt Bestandteil der Programme ist. Meist von Hand geschrieben, und schnell veraltet (documentation aging)
 - ▶ **Generierte Dokumentation**, die aus dem Programm und seinen Spezifikationen erzeugt wird
 - ▶ **Integrierte Dokumentation**, die im Programm, z. B. als Kommentare enthalten ist. Mit Werkzeugen (z. B. JavaDoc) können diese Informationen extrahiert werden und in einer Code-Dokumentation zusammengefasst werden.
 - ▶ **Elucidative Dokumentation**, die beides miteinander verbindet und konsistent hält
- Zur Gestaltung von Softwaredokumentationen geben folgende Standards Anleitung:
- ▶ auf nationaler Ebene DIN 66230, 66231, 66232, 66270(1998);
 - ▶ auf internationaler Ebene ISO/IEC 6592(2000), ISO/IEC 18019(2004)

Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)



Quelle: [24 S. 241 ff.]

Aufgaben der Dokumentationswerkzeuge

4

- ▶ Erstellung von **Dokumenten** aus den Ergebnissen bzw. Objekten der Software-Entwicklung in Textform, Graphik, Quellcode u.a.
 - Ermöglichen des Single-Source-Prinzips
 - **Generierbarkeit** der Dokumente aus verschiedenen Quellen (Repository, Quellprogramme u.a.)
 - **Verknüpfung (Verbinden) der Dokumentation** zu Code und Modellen (model mappings)
- ▶ **Strukturierung** von Dokumenten entsprechend Aufbau des Softwaresystems bzw. von Dokumentationsvorschriften
 - Berücksichtigung vorgegebener **Dokumentationsstandards** (z.B. DoD-STD-2167A)
 - Variable **Gestaltbarkeit** der Dokumente nach Struktur, Inhalt, Layout, unterschiedlichen DTP-Formaten (PDF, Postscript, MS-Word,..)
- ▶ **Wiederverwendbarkeit** zur Dokumentenproduktion auch unterschiedl. für Projekte

Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)



Taxonomie der Dokumente

5

- ▶ **Benutzerdokumentation** erklärt dem Anwender die Benutzung des Programms
 - Bedienhandbuch
 - Online-Dokumentation
 - Hilfe-Handbuch
- ▶ **Systemdokumentation** zur Installation, Spezifikation der Systemtestfälle und zur Wartung, einschließlich Code-Dokumentation
- ▶ **Projektdokumentation**
 - Entwicklerdokumentation
 - Projektführungsdokumentation (Auftrag, Projektplan, Statusberichte, Abnahmedok.)
- ▶ **Qualitätsdokumentation**
 - Test-, Audit- bzw. Review-Dokumentationen
 - Nachweis der Qualitätsparameter
- ▶ **Prozessdokumentation**
 - Dokumente der Prozessgestaltung (Vorgehensmodelle, Standards, Richtlinien)

Prof. U. Alßmann, Softwareentwicklungswerkzeuge (SEW)

Quelle: [24 S. 245 ff.]



46.2 Generative, Templategesteuerte Dokumentationswerkzeuge

6

.. funktionieren mit templategesteuerter
Codegenerierung



Dokumentationswerkzeug *JavaDoc*

7

- ▶ Erstellt aus dem mit Kommentaren versehenem Java-Quelltext automatisch HTML-Dokumentationsdateien
 - aus HTML-Schablonen (templates)
- ▶ Das Werkzeug verwendet die öffentlichen Klassen-, Interface- und Methodendeklarationen und
 - fügt zusätzliche Informationen aus evtl. vorhandenen Dokumentationskommentaren hinzu.
 - Zu jeder Klassendatei xyz.java wird eine HTML-Seite xyz.html generiert.
- ▶ Über zusätzlich generierte Index- und Hilfsdateien wird das Navigieren über verschiedene Links und Querverweise mit anderen Seiten der Dokumentation unterstützt.
- ▶ Über die Steuerung mittels Tags (@) können Metadaten zur Aufbereitung der Dokumente verwendet werden.
- ▶ Das Layout kann nur schwer beeinflusst werden. Inhalt und Struktur der Dokumente werden 1:1 vom Programmquelltext übernommen.
 - Über Doclets ist eine weitere Konvertierung auch in RTF, XML, PDF, Framemaker und Windows Help möglich.
- ▶ JavaDoc gibt es mittlerweile für alle anderen Programmiersprachen

Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)



Quelle: JavaDoc; aus Wikipedia URL: <http://de.wikipedia.org/wiki/JavaDoc>

Document Generator *SELECT*

8

- ▶ Zusammenarbeit mit MS Word-Templates
 - Hardcopy von Diagrammen mit OLE in ein Word-Dokument
 - Der SELECT Document Generator kann ein Dokumentationsgerüst in MS Word generieren
- ▶ Inhalt der generierten Dokumentation:
 - Diagramme des SELECT-Modells, genaue Objektbeschreibungen, leere Abschnitte für Analysen, Berichte oder Zusammenfassungen
 - Auswahl aus vorgefertigten Dokumentvorlagen
 - Auswahl der einzufügenden Abschnitte aus der Vorlage
 - Nachträgliches Einfügen und Entfernen von Abschnitten möglich
 - Aktualisierung des Inhalts durch selektive Synchronisierung mit dem SELECT-Modell
- ▶ Das Layout lässt sich anpassen durch Bearbeiten der Dokumentvorlagen oder des generierten Dokumentes
- ▶ Navigation in den generierten Abschnitten der Dokumentation direkt aus dem Document Generator heraus

Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)



Dokumentationswerkzeuge Innovator

9

- ▶ Aus den Modellspezifikationen sowie den weiteren Informationen (Textdateien, Grafiken) wird eine Modelldokumentation generiert.
- ▶ Modellspezifische Dokumentationsvorlagen
 - Struktur und Layout nach Wunsch bestimmbar.
- ▶ Die im Modellbrowser selektierten Dokumentationsinhalte werden über einen Dokumentationsgenerator erzeugt und in die eigentliche Dokumentation überführt.
- ▶ Ausgabearart:
 - Voranzeigefenster (Standardeinstellung in eigenem Format);
 - PostScript-Dokument;
 - Word für Windows Dokument;
 - ASCII-Text Dokument;
 - XML-Datei
 - Druck-Repository für Druckaufbereitung.

Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)



Beispiel Dokumentationsvorlage

10

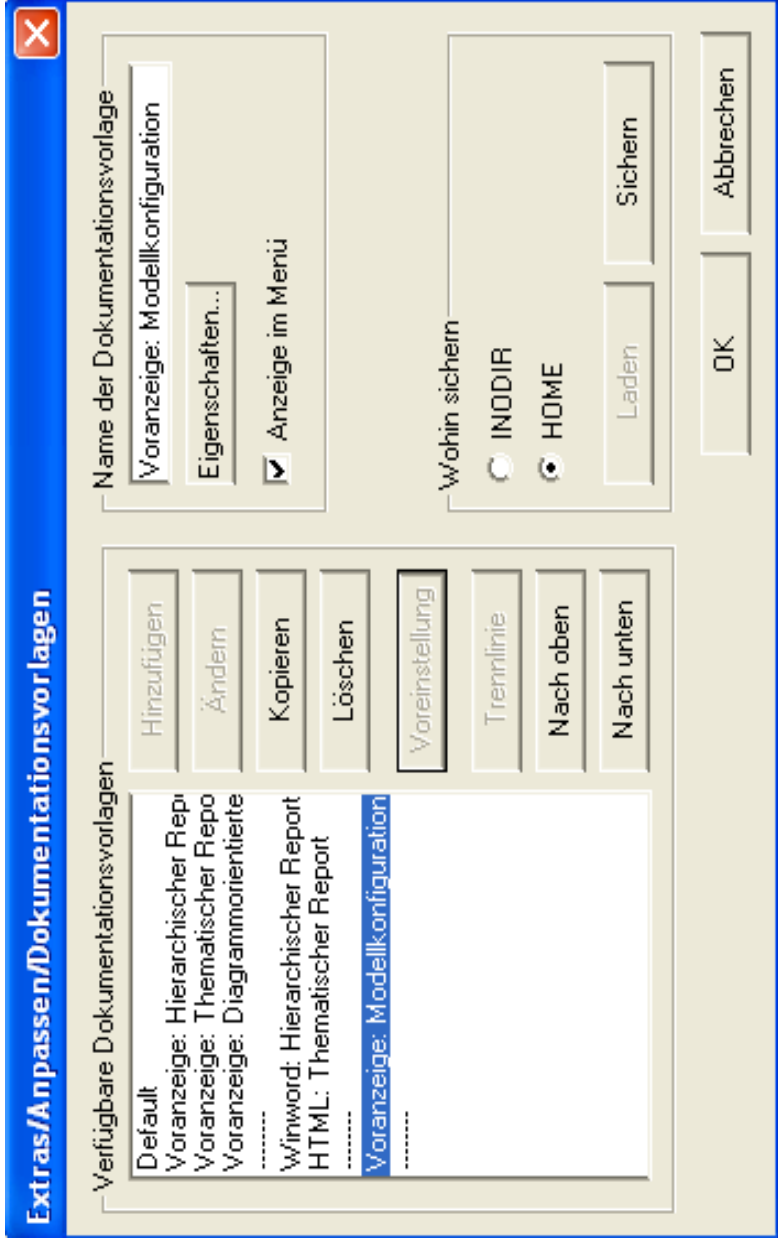


Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)



Anpassen Dokumentationsvorlagen

11

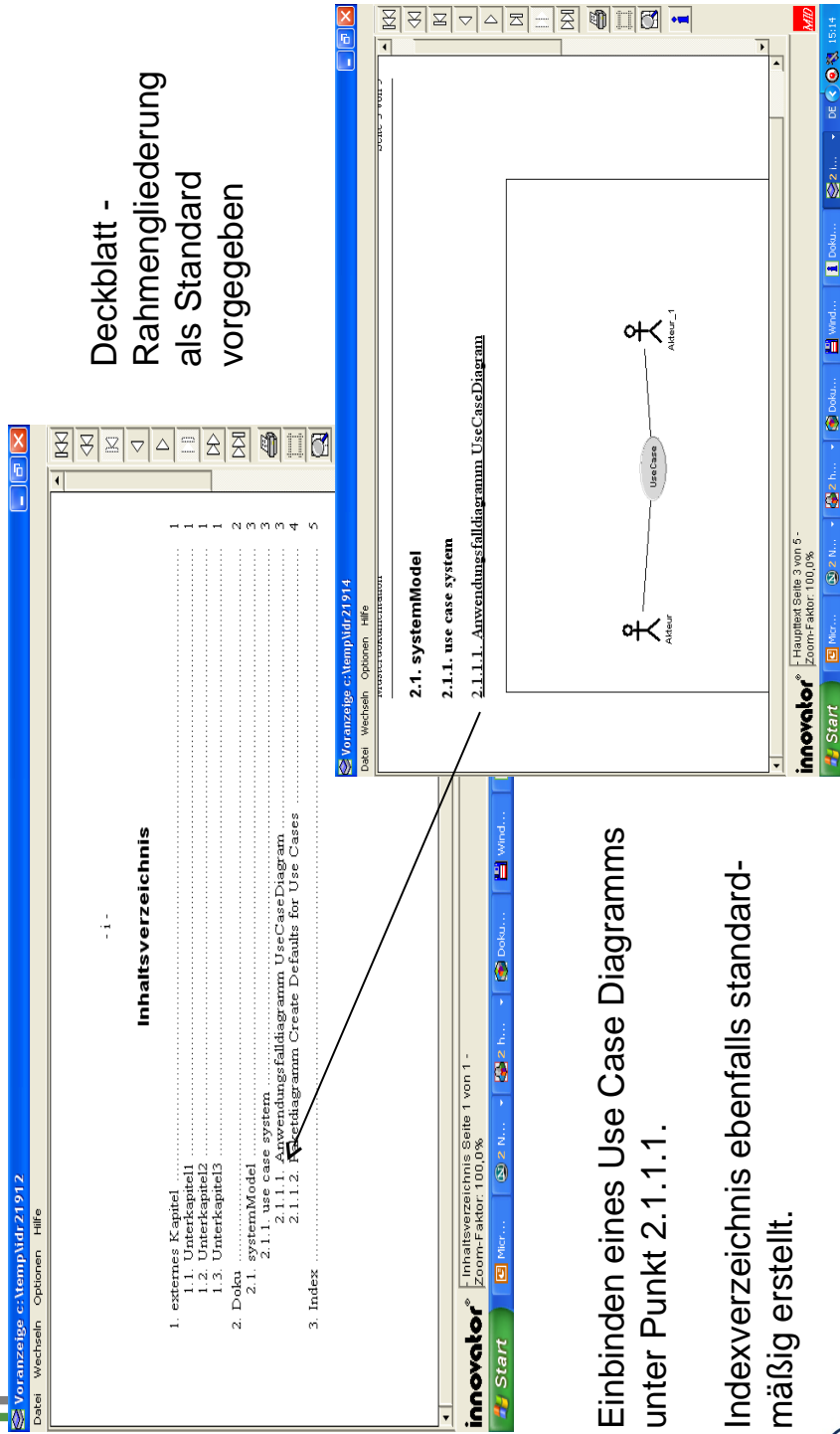


Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)



Beispieldokumente Innovator

12



Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)



Einbinden eines Use Case Diagramms unter Punkt 2.1.1.1.

Indexverzeichnis ebenfalls standardmäßig erstellt.

46.3 Elucidative Dokumentationswerkzeuge

13

- They link code, models and documentation by model mapping



Softwareentwicklungswerkzeuge (SEW) © Prof. Uwe Alßmann

Tutorial: Beispiel

14

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 `startObservationOutputStream()` wird versucht, alle laufenden Prozesse zu beenden und das Shopfenster zu schließen. Hatte der 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 java.io.*;
import java.io.*; //IOException, FileOutputStream
public class SalesPointApplication extends Shop {
    public SalesPointApplication() {
        super();
    }

    public void start() {
        super.start();
    }

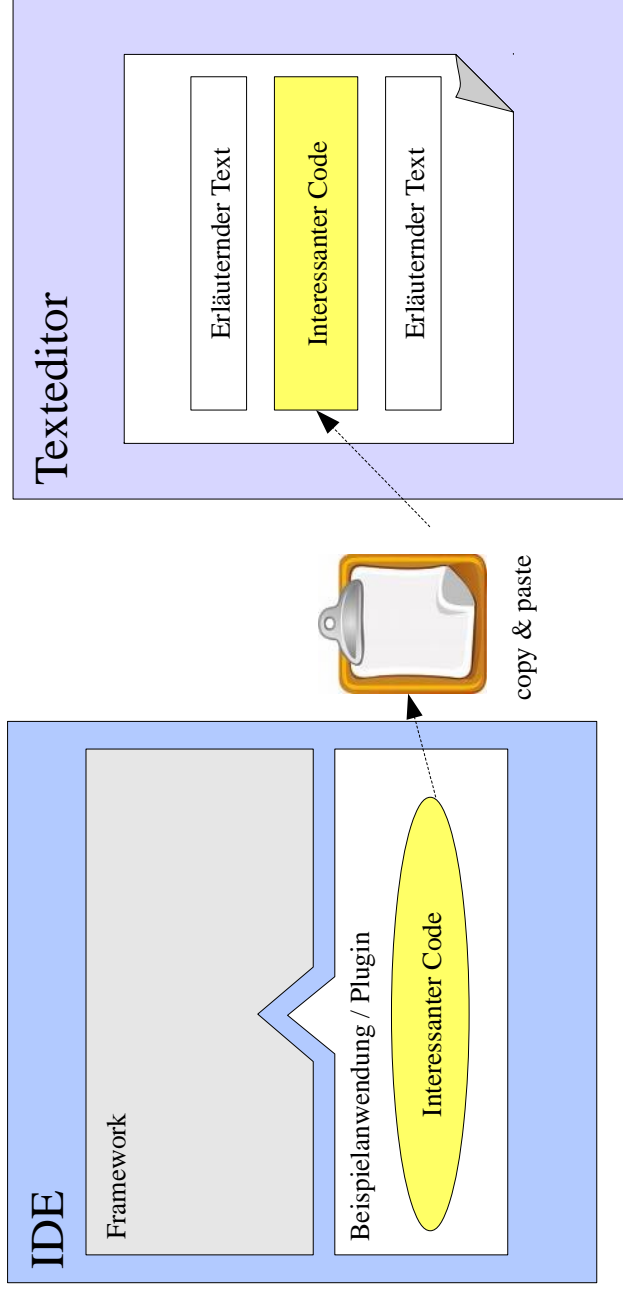
    try {
        startObservationOutputStream(new FileOutputStream("logfile.log", true));
    } catch (IOException ioex) {
        System.err.println("Unable to create log file.");
    }
}

import java.io.*;
import java.io.*; //IOException, FileOutputStream
public class SalesPointApplication extends Shop {
    public SalesPointApplication() {
        super();
    }

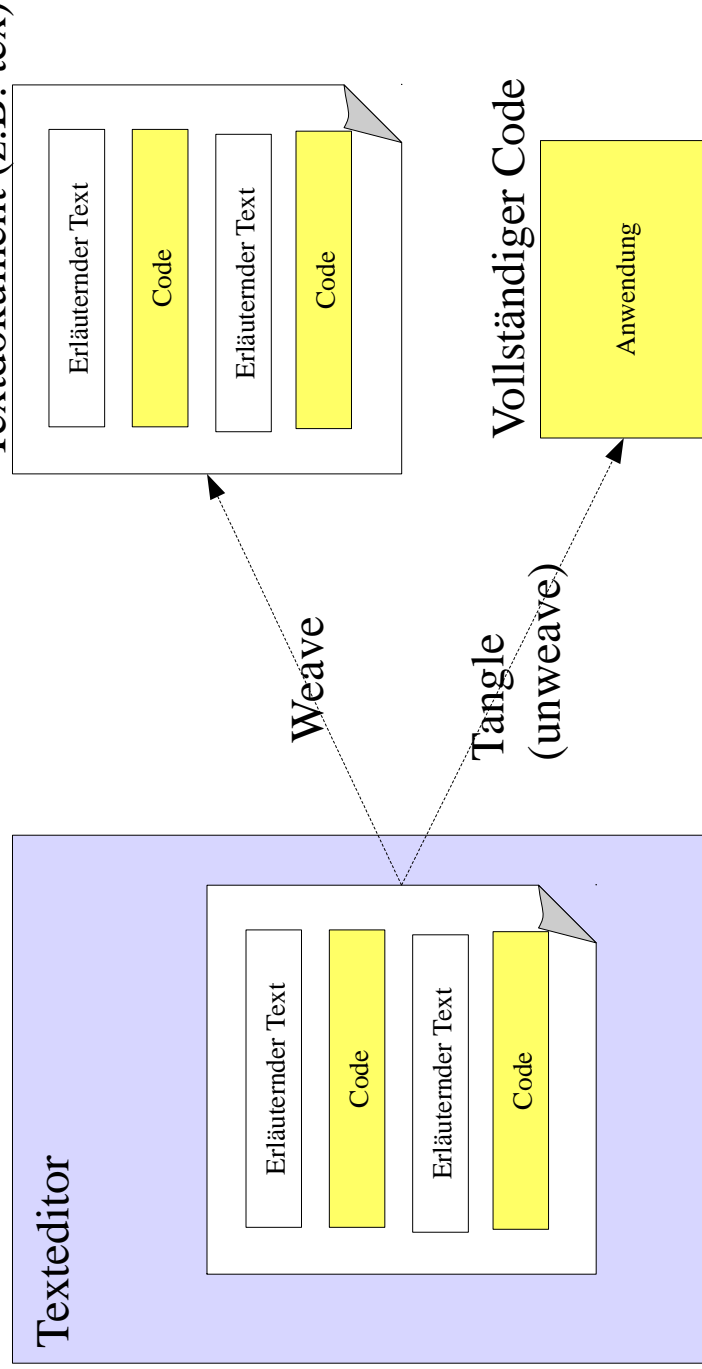
    public void start() {
```



Manuelles Tutorialschreiben



Literate Programming by Code Unweaving



[[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.
```

aus Literate Programming
von Donald E. Knuth

- ▶ Überblick: <http://www.literateprogramming.com/>
- ▶ OMNotebook/DrModelica: <http://www.modelica.org/tools>

OMNotebook - DrModelica

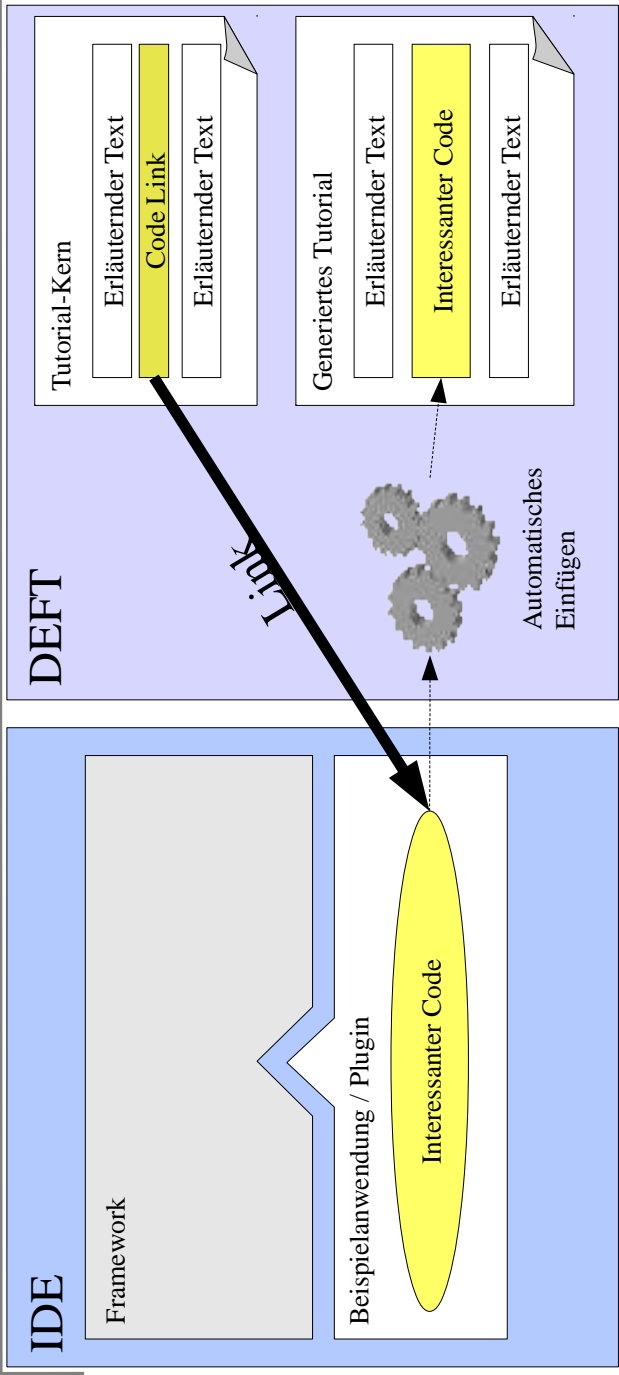
```
class Dog
  constant Real legs = 4;
  parameter String name = "Dummy";
end dog;

Create an instance of the class Dog

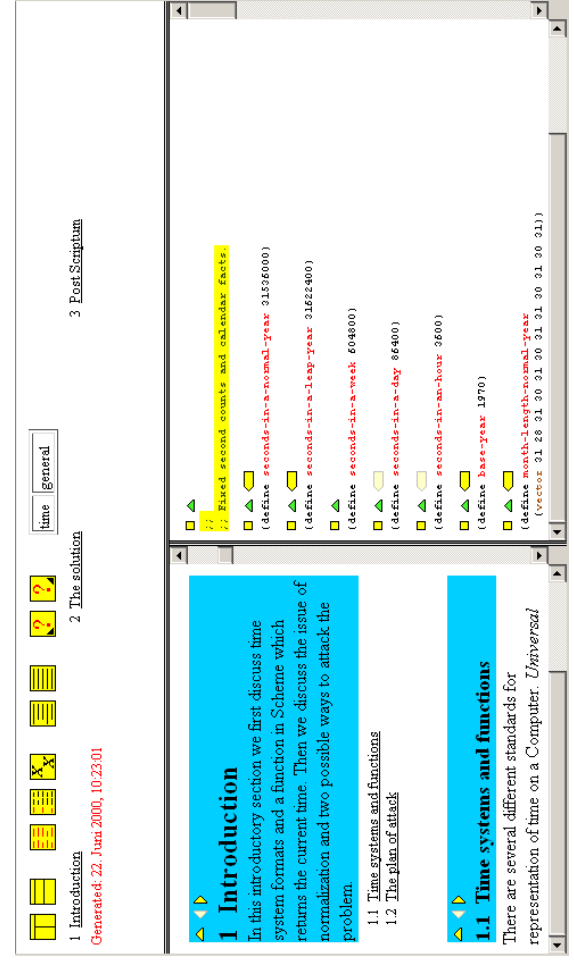
Create another instance and give the dog the name "Im".
```

- ▶ Verlinkte Dokumente mit interaktiven Übungen, teilweise ausführbar
- ▶ Inspiriert von DrScheme und DrJava, Lernwerkzeugen für Scheme bzw. Java

Elucidative Programming Links Documentation with Queries to Code



Elucidative Programming

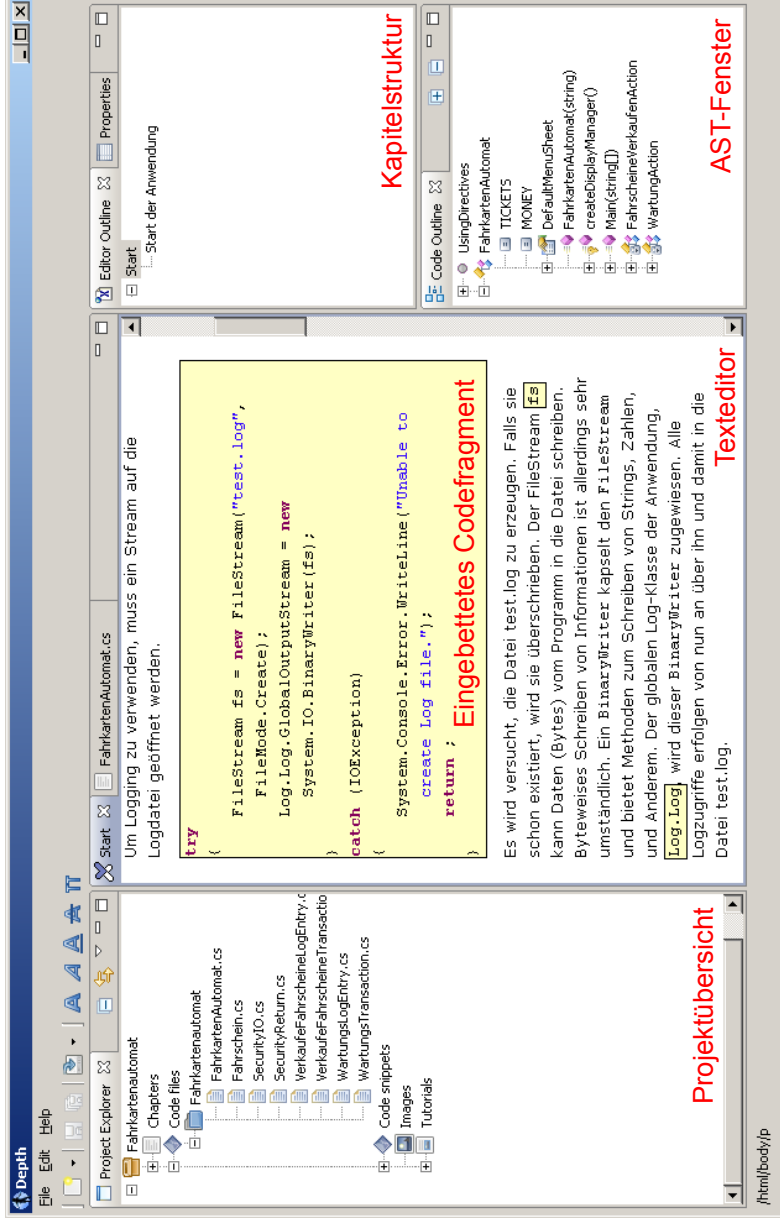


- ▶ <http://www.cs.aau.dk/~normark/elucidative-programming/>
- ▶ <http://deftproject.org>

Development Environment For Tutorials (DEFT)

www.deftproject.org

21



Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)



Development Environment For Tutorials

22

- ▲ Eclipse RCP-Anwendung
- ▲ Sprachunabhängig
- ▲ Verwaltung von Text und Code unter einem Dach
- ▲ Automatisches Prettyprinting von Codefragmenten
- ▲ Hilfe bei der Aktualisierung
 - Automatische Aktualisierung von eingebetteten Codefragmenten
 - Benachrichtigung bei veränderten Codefragmenten

Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)



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 `LokaleDatei` 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 test.");
    return ;
}
```

Es wird versucht, die Datei `test.log` zu erstellen. Falls sie schon existiert, wird sie überschrieben. Der `FileStream` ist kompakt (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, Inten, 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 tempLux = new Point((d.Width - 100) / 2, (d.Height - 80) / 2);
    return new AirDisplayManager(this, ref tempLux, ref tempLux2);
}

[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
    Catalog cTickets = Catalog.forName(TICKETS);
    cTickets.addItem(new Fahrerschein("Einzelfahrt", 300));
    cTickets.addItem(new Fahrerschein("Sammelfahrerschein", 1500));
    cTickets.addItem(new Fahrerschein("erwässigte Einzelfahrt", 150));
}
```