72. Dokumentationswerkzeuge

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

46.1 Aufgaben der Dokumentationswerkzeuge

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

Ziel der Softwaredokumentation

- 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 (Terminerfüllung, 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 Progamm 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)

Quelle: [24 S. 241 ff.]

Aufgaben der Dokumentationswerkzeuge

- 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, Quell-programme 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

Taxonomie der Dokumente

- **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)

Quelle: [24 S. 245 ff.]

46.2 Generative, Templategesteuerte Dokumentationswerkzeuge

.. funktionieren mit templategesteuerter Codegenerierung

Dokumentationswerkzeug *JavaDoc*

- Erstellt aus dem mit Kommentaren versehenem Java-Quelltext automatisch HTML-Dokumentationsdateien
  - aus HTML-Schablonen (templates)
- Das Werkzeug verwendet die öffentlichen Klassen-, Interface- und Methodendeklara-tionen und
  - fügt zusätzliche Informationen aus evtl. vorhandenen Dokumentationskom-mentaren hinzu.
  - Zu jeder Klassendatei xyz.java wird eine HTML-Seite xyz.html generiert.
- Über zusätzlich generierte Index- und Hilfsdateien wird das Navigieren über verschie-dene 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

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

Document Generator *SELECT*

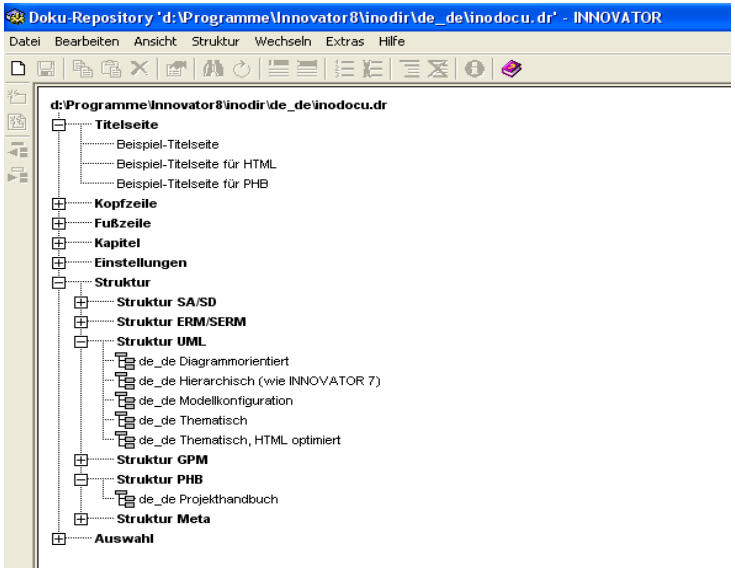
- Zusammenarbeit mit MS Word-Templates
  - Hardcopy von Diagrammen mit OLE in ein Word-Dokument
  - Der SELECT Document Generator kann ein Dokumentationsgerüst in MS Word generieren
- Inhalt der generierten Dokumentation:
  - Diagramme des SELECT-Modells, genaue Objektbeschreibungen, leere Abschnitte 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

Dokumentationswerkzeuge Innovator

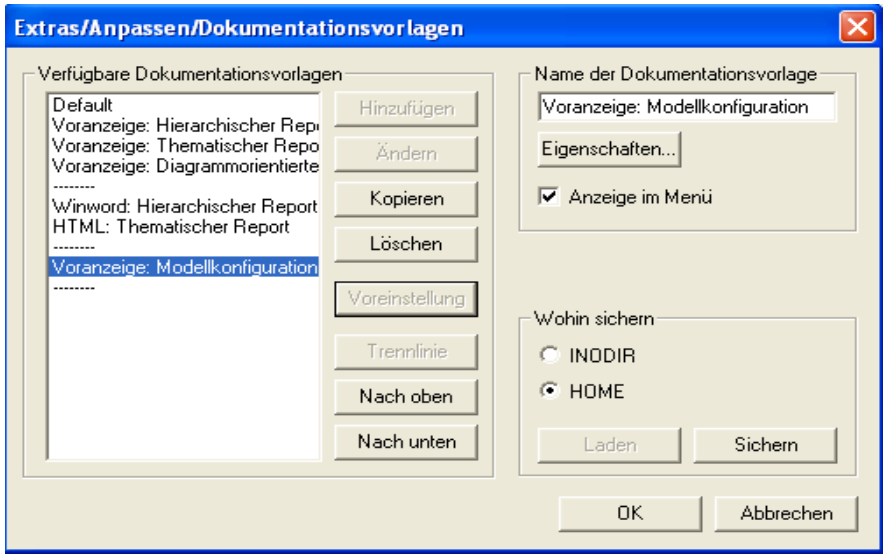
- Aus den Modellspezifikationen sowie den weiteren Informationen (Textdateien, Grafiken) wird eine Modelldokumentation generiert.
- Modellspezifische Dokumentationsvorlagen
  - Struktur und Layout nach Wunsch bestimmbar.
- Die im Modellbrowser selektierten Dokumentationsinhalte werden über einen Dokumentationsgenerator erzeugt und in die eigentliche Dokumentation überführt.

- Ausgabeart:
  - Voranzeigefenster (Standardeinstellung in eigenem Format);
  - PostScript-Dokument;
  - Word für Windows Dokument;
  - ASCII-Text Dokument;
  - XML-Datei
  - Druck-Repository für Druckaufbereitung.

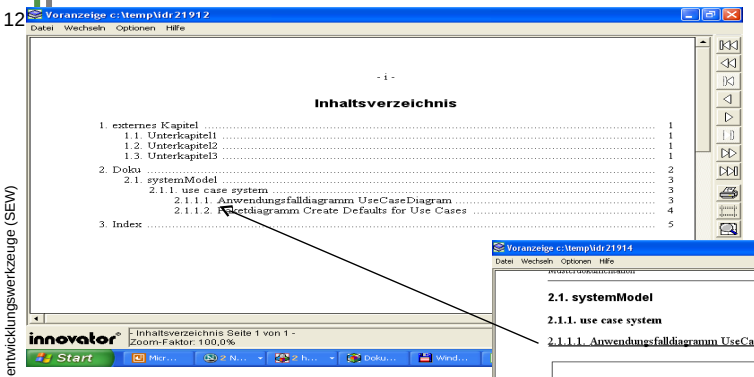
Beispiel Dokumentationsvorlage

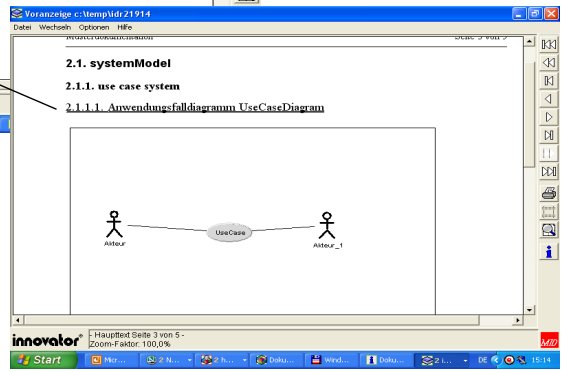


Anpassen Dokumentationsvorlagen



Beispieldokumente Innovator





Deckblatt - Rahmengliederung als Standard vorgegeben

Einbinden eines Use Case Diagramms unter Punkt 2.1.1.1.

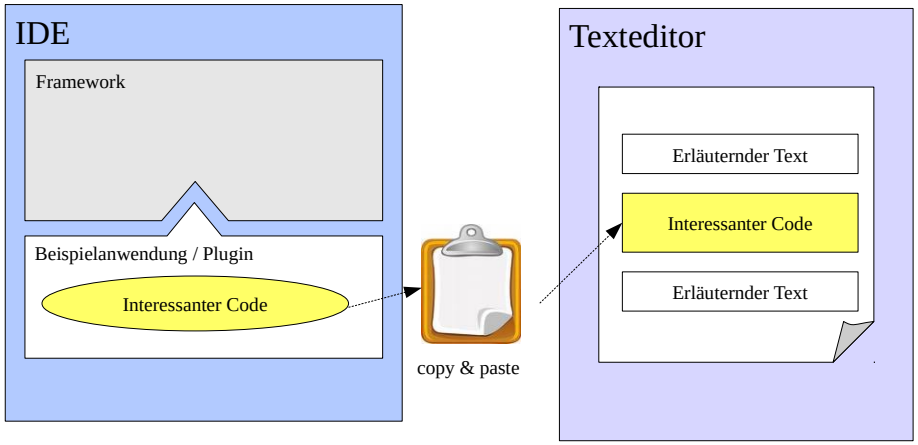
Indexverzeichnis ebenfalls standard-mäßig erstellt.

46.3 Elucidative Dokumentationswerkzeuge

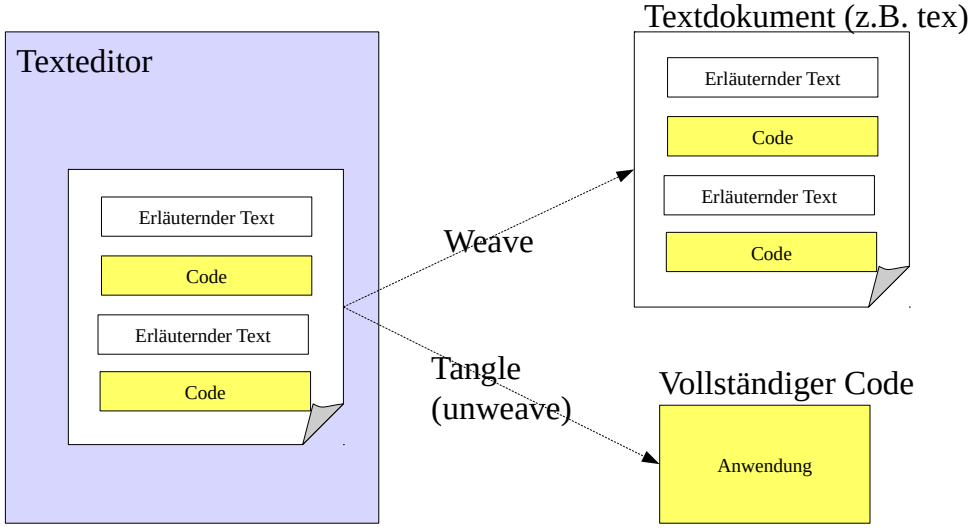
- They link code, models and documentationi by **model mapping**

Tutorial: Beispiel

Manuelles Tutorialschreiben



Literate Programming by Code Unweaving



Literate Programming

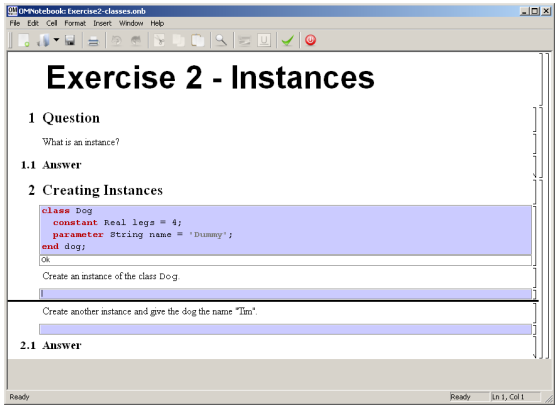
⟦The program text below specifies the "expanded meaning" of ⟨Program to print ... numbers 2⟩; notice that it involves the top-level descriptions of three other sections. When those top-level descriptions are replaced by their expanded meanings, a syntactically correct PASCAL program will be obtained.⟧

```
⟨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.
```

aus Literate Programming
von Donald E. Knuth

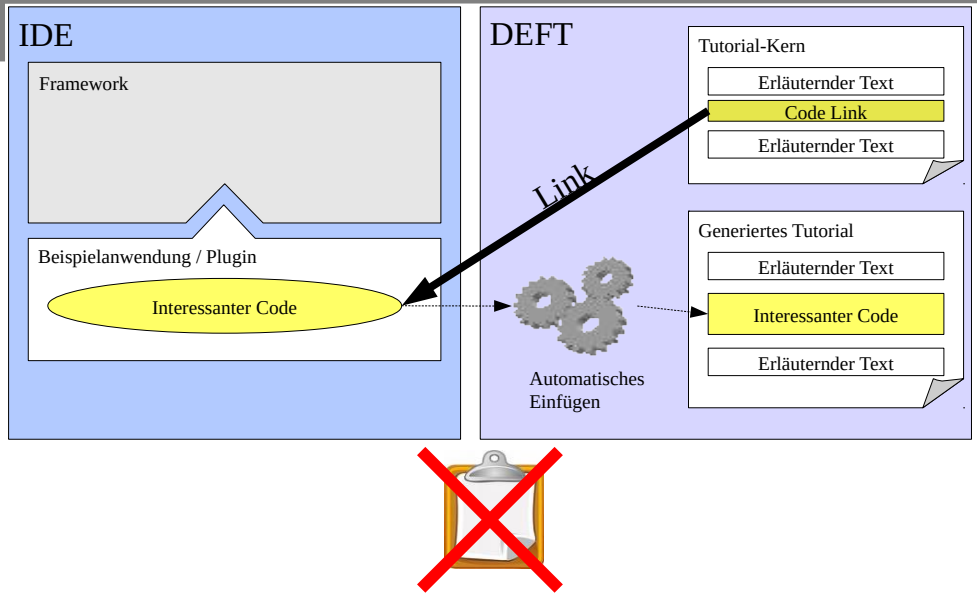
- Überblick: http://www.literateprogramming.com/
- OMNotebook/DrModelica: http://www.modelica.org/tools

OMNotebook - DrModelica

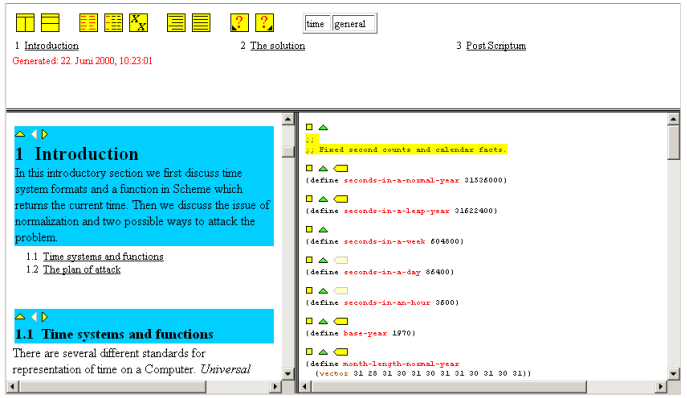


- Verlinkte Dokumente mit interaktiven Übungen, teilweise ausführbar
- Inspiriert von DrScheme und DrJava, Lernwerkzeugen für Scheme bzw. Java

Elucidative Programming Links Documentation with Queries to Code

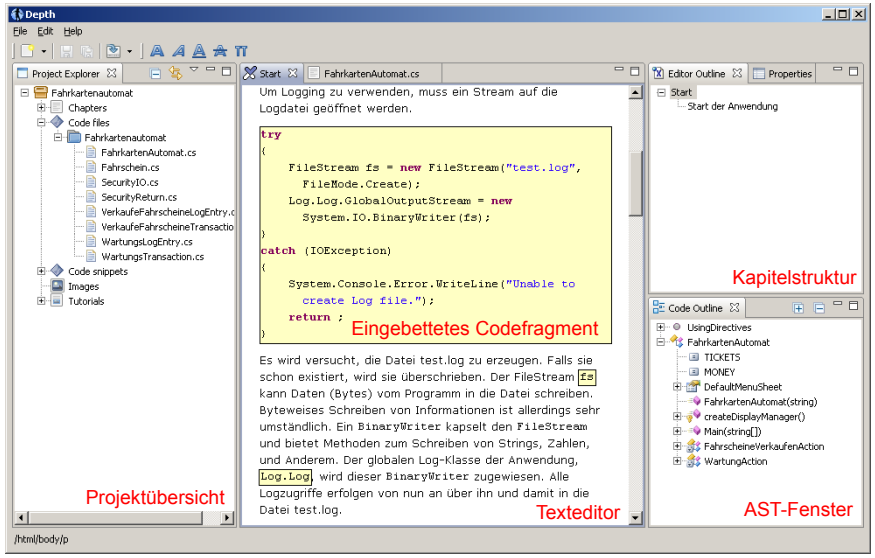


Elucidative Programming



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

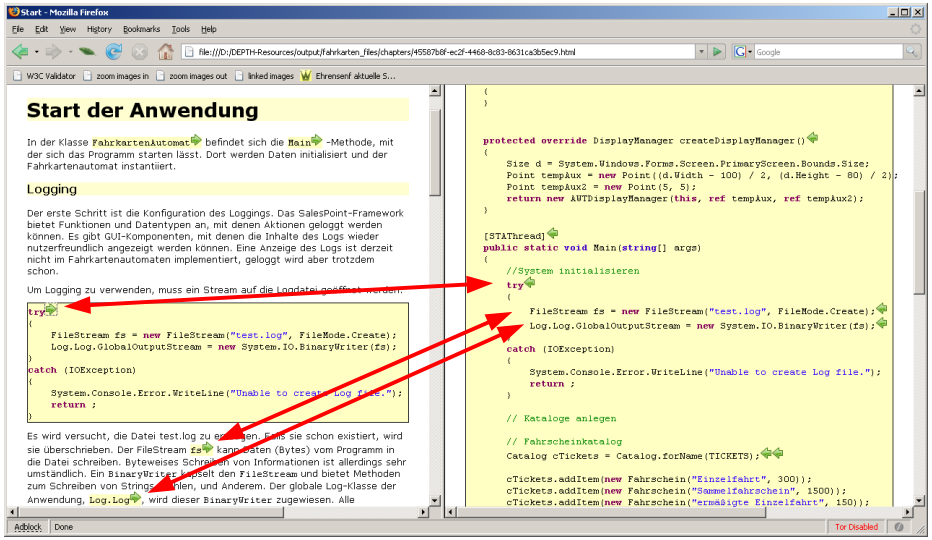
Development Environment For Tutorials (DEFT www.deftproject.org)



Development Environment For Tutorials

- 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

Generiertes HTML Tutorial



The End