Handbücher und Dokumentation

Softwaretechnik

Andreas Zeller
Lehrstuhl für Softwaretechnik
Universität des Saarlandes, Saarbrücken

2005-10-31

Gestaltung von Benutzer-Handbüchern

Das *Benutzer-Handbuch* ist in der Regel der erste Kontakt eines Benutzers mit einem neuen Software-System.

Das Handbuch ist sozusagen die Visitenkarte des Systems; entsprechend sorgfältig sollte es gestaltet sein.

Das Benutzer-Handbuch soll die Handhabung und das Verhalten des Produkts möglichst vollständig und fehlerfrei beschreiben.

Es muss möglich sein, das Produkt nur anhand des Handbuchs zu bedienen.

Ökonomie von Benutzer-Handbüchern

Warum lohnt sich die Investition in gute *gedruckte* Handbücher?

- Die Benutzbarkeit von gedruckten Büchern übersteigt die von Online-Handbüchern.
- Gute Handbücher sind ein guter Kopierschutz.
- Klare Handbücher ersparen Folgekosten bei der Benutzerberatung.
- Handbücher vermitteln die Konzepte hinter einer Software, die sich aus der GUI nicht erschliessen lassen. Beispiele: Adobe Photoshop, Quark XPress, Programmiersprachen.

Lesenswert:
http://www.asktog.com/columns/017ManualWriting.html

Benutzer-Kategorien

Anordnung und Aufbau eines Handbuchs werden im wesentlichen durch die *Adressaten* bestimmt. Man unterscheidet drei Kategorien:

- *Anfänger* haben keine oder wenig Erfahrungen mit Computersystemen im allgemeinen und dem Produkt im speziellen.
- *Experten* haben viel Erfahrungen mit Computersystemen im allgemeinen, aber wenig Erfahrungen mit dem speziellen Produkt.
- *Fortgeschrittene* liegen irgendwo zwischen Anfängern und Experten; sie haben vielleicht schon ähnliche Produkte benutzt.

Handbuchtypen

Für jede Benutzer-Kategorie gibt es eigene Handbuch-Typen:

- Trainings-Handbuch (*Tutorial*)
- Referenz-Handbuch
- Referenzkarte
- Benutzer-Leitfaden (*user guide*)

Trainings-Handbuch (*Tutorial*)

Zielgruppe: *Anfänger*

Ein Trainings-Handbuch

- ist als Kurs oder Trainingsprogramm organisiert
- ist nach *Aufgaben* gegliedert:
  - Es beschreibt *Wege zur Erreichung von Arbeitszielen.* Beispiel: *So buchen Sie eine Anmeldung*
  - Die nötigen Arbeitsabläufe bestimmen die Gliederung.
- muss von Anfang bis Ende vollständig durchgearbeitet werden
- erfordert direkte Arbeit mit dem Produkt, d.h. der Leser führt die Anweisungen des Trainings-Handbuchs aus
- vermittelt rasch Erfolgserlebnisse
- ist für fortgeschrittene Benutzer und Experten zu elementar

Referenz-Handbuch

Zielgruppe: *Experten*

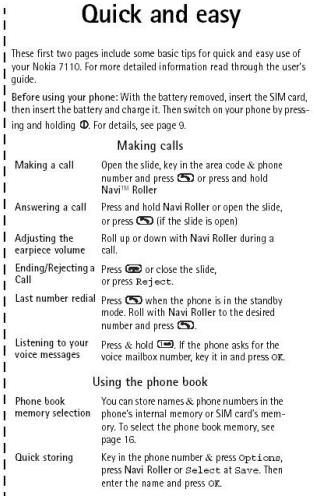
Ein Referenz-Handbuch

- ermöglicht schnellen Zugriff auf spezifische Information
- ist nach *Produktaufbau* gegliedert:
  - Alle Funktionen und Bestandteile werden in der Reihenfolge beschrieben, die sich aus der Struktur des Produktes ergibt. Beispiel: *Die Funktion „Anmeldungen bearbeiten*“
  - Es wird beschrieben, was das Produkt kann…
  - …und nicht, wie man Arbeitsziele mit dem Produkt erreicht
- bietet vollständige Informationen zu allen Produktfunktionen
- unterstellt, dass der Benutzer mehr weiß, als er tun will

Referenzkarte

Zielgruppe: Ebenfalls *Experten*
Eine Referenzkarte

- fasst die wesentlichen Informationen der wichtigsten oder häufigsten Funktionen zusammen.
- ist möglichst kompakt (nicht mehr als eine A4-Seite)
- Beispiel rechts aus Handbuch für Nokia 7110 Mobiltelefon.



Quick and easy

These first two pages include some basic tips for quick and easy use of your Nokia 7110. For more detailed information read through the user's guide.

Before using your phone: With the battery removed, insert the SIM card, then insert the battery and charge it. Then switch on your phone by pressing and holding ⏻. For details, see page 9.

Making calls

Making a call	Open the slide, key in the area code & phone number and press 📞 or press and hold Navi™ Roller
Answering a call	Press and hold Navi Roller or open the slide, or press 📞 (if the slide is open)
Adjusting the earpiece volume	Roll up or down with Navi Roller during a call.
Ending/Rejecting a Call	Press ⏻ or close the slide, or press Reject.
Last number redial	Press 📞 when the phone is in the standby mode. Roll with Navi Roller to the desired number and press 📞.
Listening to your voice messages	Press & hold 1. If the phone asks for the voice mailbox number, key it in and press OK.

Using the phone book

Phone book memory selection	You can store names & phone numbers in the phone's internal memory or SIM card's memory. To select the phone book memory, see page 16.
Quick storing	Key in the phone number & press Options, press Navi Roller or Select at Save. Then enter the name and press OK.

Benutzer-Leitfaden (*user guide*)

Zielgruppe: *fortgeschrittene Benutzer*

Ein Leitfaden

- ist eine Mischform aus Trainings- und Referenzhandbuch
- muss simultan die Bedürfnisse beider Benutzergruppen, Anfänger und Experten, erfüllen.
- besteht typischerweise aus zwei Teilen:
  - Die aufgabenorientierte *Arbeitsanleitung* dient zum Einarbeiten.
    * nicht vollständig
    * erlaubt das Arbeiten mit Grundfunktionen.
  - Der *Referenzteil* dient zum späteren Nachschlagen.
    * erlaubt es, sich die weiteren Funktionen zu erschließen.
- erlaubt es, bereits bekannte Informationen zu überschlagen
- darf aber nicht davon ausgehen, dass der Leser Systemdetails bereits kennt
- ist bei einfachen Systemen das einzige Handbuch

Bei umfangreichen Systemen gibt es oft alle vier Handbuch-Typen!

Beispiel: TEXbook

Aufgaben vertiefen das Verständnis, Passagen für Experten sind besonders gekennzeichent.

take up the visual slack. The standard rule of thumb is to use \/ just before switching from slanted or italic to roman or bold, unless the next character is a period or comma. For example, type

```
{\it italics\/} for {\it emphasis}.
```

Old manuals of style say that the punctuation after a word should be in the *same* font as that *word;* but an italic semicolon often looks wrong, so this convention is changing. When an italicized word occurs just before a semicolon, the author recommends typing '`{\it word\/};`'.

▸ **EXERCISE 4.2**

Explain how to typeset a roman *word in the midst of an italicized sentence.*

Every letter of every font has an italic correction, which you can bring to life by typing \/. The correction is usually zero in unslanted styles, but there are exceptions: To typeset a bold '**f**' in quotes, you should say `a bold '{\bf f\/}'`, lest you get a bold '**f**'.

▸ **EXERCISE 4.3**

Define a control sequence `\ic` such that '`\ic c`' puts the italic correction of character *c* into TEX's register `\dimen0`.

Aufbau eines Benutzer-Handbuchs

Für Benutzer-Handbücher lässt sich kein generelles Gliederungsschema wie etwa Pflichtenhefte vorgeben. Neben dem sachlichen Inhalt gehören aber bestimmte Teile in jedes Handbuch:

Vorwort

Enthält Adressatenkreis, Anwendungsbereich, Änderungen gegenüber Vorversionen.

Ziel: der Leser muss nach den ersten Seiten wissen, ob Produkt und Handbuch für ihn und seine Anwendung bestimmt sind.

Manche Leser überblättern das Vorwort; deshalb gehören unverzichtbare Informationen in die *Einleitung*.

Inhaltsverzeichnis

Handbuch-Aufbau (2)

Einführung

Dient in der Regel als *Gebrauchsanleitung*, die über den Aufbau und den Umgang mit dem Handbuch informiert.

Beschreibt oft auch

- die Konfiguration, die für das Produkt erforderlich ist (wenn nicht in eigenem Kapitel).
- Zielgruppe
- Vorkenntnisse zum Verstehen des Handbuchs

Insgesamt soll die Einführung kurz sein.

Installation

Beschreibt, was der Benutzer tun muss, um das Produkt in Betrieb zu nehmen.

Wird nur selten gelesen, kann deshalb in den Anhang (dann muss aber in der Einführung darauf verwiesen werden).

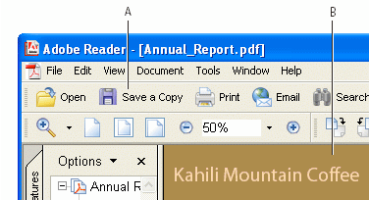
Handbuch-Aufbau (3)

Benutzungsoberfläche

Verdeutlicht den prinzipiellen Aufbau der Benutzungsoberfläche.

Insbesondere

- Tasten- und Mausbelegung
- grundsätzlicher Bildschirmaufbau (z.B. Fenstertypen)
- Dialogstrategie (z.B. erst Objekt, dann Funktion auswählen)



Produktstruktur

Gibt einen Überblick über die einzelnen Bestandteile des Produkts.

Handbuch-Aufbau (4)

Trainingsteil

(im Benutzer-Leitfaden oder Trainings-Handbuch)

Beschreibt *typische Arbeitsabläufe* mit Beispielen und Übungen, mit besonderem Wert auf *Routinearbeiten*.

Anordnung:

- häufigste Arbeitsabläufe und Routinearbeiten
- seltene Arbeitsabläufe
- Initialisieren und Löschen von Systemen

Beispiele müssen aus dem geplanten Anwendungsbereich stammen

Abstraktionen (also *var-1*, *var-2* statt *zins* und *kapital*) sind zu vermeiden

Übungen sind so zu gestalten, dass der Benutzer mit dem Produkt arbeitet – und dabei Erfolgserlebnisse hat.

Handbuch-Aufbau (5)

Referenzteil

(im Benutzer-Leitfaden oder Referenz-Handbuch)

Enthält eine vollständige Beschreibung der einzelnen Objekte und Funktionen.

Die Anordnung orientiert sich meistens an der Menüstruktur.

Kommandos werden alphabetisch angeordnet.

Behandlung von Problemen

Enthält eine Liste von Fehlermeldungen einschließlich detaillierter Erklärungen und Vorschläge

Handbuch-Aufbau (6)

Literaturverzeichnis

Abkürzungsverzeichnis

Glossar

Stichwortverzeichnis / Index / Register

Wichtigstes Hilfsmittel für einen schnellen Zugriff

Enthält Benutzerziele (z.B. *Anmeldung vornehmen*) und Funktionsnamen (z.B. *Buchung erfassen*)

Erstellt der Handbuch-Autor ein gutes Handbuch, dann sparen hunderte oder tausende Benutzer eine Menge Zeit und Nerven!

Hilfesysteme

Ein *Hilfesystem* unterstützt den Benutzer während der Benutzung durch explizite Erklärungen und Auskünfte.

Schnelle und vom Kontext abhängige Ergänzung zum Benutzer-Handbuch

Hilfesysteme erläutern typischerweise folgende Punkte:

- Aktuell wählbare Objekte, Funktionen, und Kommandos (und deren Optionen)
- Bedeutung von Funktionstasten und Mausknöpfen
- Hinweise zu Eingaben
- Erläuterungen zu Funktionsergebnissen
- Spezifische Fehlererklärungen (einschließlich Hilfen zur Korrektur)

Hilfesysteme (2)

Vorteile gegenüber dem Handbuch:

+ Texte in Hilfesystemen sind schneller und leichter zu aktualisieren.
+ Die Information in Hilfesystemen kann nicht verlorengehen.

Nachteile gegenüber dem Handbuch:

- Der Text auf dem Bildschirm wird *langsamer gelesen* als in einem Handbuch (kürzer fassen!)
- Notizen und Markierungen wie auf Papier sind nicht möglich.

Hilfesysteme (3)

Es gibt auch *Mischformen* aus Hilfesystemen und Handbüchern:

- Es gibt Programme, die *ausschließlich per Hilfesystem* dokumentiert sind; lediglich die Installationsanleitung kommt in gedruckter Form.
- Bei manchen Programmen (z.B. *Netscape*) ist die gesamte Dokumentation als WWW-Hypertext organisiert, auf den über das Netz zugegriffen wird.

Dynamische Hilfesysteme

Ein *statisches Hilfesystem* liefert stets dieselbe Information, unabhängig vom Kontext, z.B.:

> in einem Fenster wird immer die gleiche Erklärung gegeben.
>
> der UNIX-Editor ed gibt bei Fehlern aller Art die Meldung "*?*" aus.

Ein *dynamisches Hilfesystem* berücksichtigt den Kontext zum Zeitpunkt der Hilfeanforderung, z.B.:

> in jedem Eingabefeld eines Fensters wird eine spezifische Hilfe gegeben.
>
> eine Fehlermeldung lautet *Der eingegebene Wert 0.005 ist zu klein*

Dynamische Hilfesysteme sind stets vorzuziehen.

Individuelle Hilfesysteme

Ein *uniformes Hilfesystem* gibt jedem Benutzer dieselbe Hilfe unabhängig von seinem Kenntnisstand.

Ein *individuelles Hilfesystem* unterscheidet nach verschiedenen Benutzergruppen (Anfänger, Experte).

Die Einstufung kann auch automatisch erfolgen.

> Beispiel: Nachdem Sachbearbeiter Müller nur noch selten das Hilfesystem aufruft, wird er als vom Hilfesystem als *Experte* eingestuft und erhält nur noch eine Kurzform der Erklärungstexte.

Individuelle Hilfesysteme sind stets vorzuziehen.

Passive und aktive Hilfesysteme

Nur 40% der Funktionalität komplexer Systeme werden benutzt.

Manche Funktionen werden nur selten benutzt, da der Benutzer

- die Funktionen nicht im Detail kennt
- sich unsicher über ihre Details und Wirkungen ist

Das ist der Einsatzbereich von *passiven Hilfesystemen*.

Ein passives Hilfesystem erwartet, dass der Benutzer von sich aus eine Hilfeleistung anfordert, z.B.

- durch Drücken einer Taste (*F1* oder *Help*)
- durch Eingabe eines Kommandonamens ("*man ls*")
- durch Anfragen in natürlicher Sprache („*Warum kann hier kein Text eingefügt werden?*")
- durch Anwahl eines Bedienelements (*balloon help*)

Problem: Oft sind Konzepte realisiert, die der Benutzer nicht vermutet.

Ein *aktives Hilfesystem* beobachtet das Benutzungsverhalten und wird von sich aus aktiv, um dem Benutzer eine Hilfe zu geben.

> Beispiel: Um sich von einem Wort zum anderen zu bewegen, wurden einzeln Pfeiltasten benutzt. Der Word-Assistent erkennt diesen Vorgang und weist den Benutzer darauf hin, dass mit Ctrl+Pfeiltasten Wörter übersprungen werden können.

In der Regel werden heute passive und uniforme Hilfssysteme eingesetzt, die statische und dynamische Hilfeleistungen mischen.

Interne Dokumentation

Wichtig für besseres Verständnis + besseres Design

> The designer of a new system must not only be the implementor and the first large-scale user; the designer should also write the first user manual. [...] If I had not participated fully in all these activities, literally hundreds of improvements would never have been made, because I would never have thought of them or perceived why they were important

[Donald Knuth zu dem Design von TEX]

Die präzise Dokumentation eines Systems für andere zwingt zu einem anderen Blickwinkel:

- Weg von den internen Details
- Hin zu der dem Benutzer oder Programmierer bereitgestellten Funktionalität

Arten der Dokumentation

Design-Dokument

- Leser: Neue Programmierer
- Inhalt: Übersicht, Abstraktionen, High-Level Funktionalität, Design-Entscheidungen, Anwendungsbereiche, Einschränkungen

Tutorial

- Leser: Neue Programmierer
- Inhalt: Typische und einfache Nutzung, Beispiele

Arten der Dokumentation (2)

Referenz-Handbuch

- Leser: Programmierer
- Inhalt: Nutzung, Schnittstellen-Beschreibungen, genaue Semantik, Einschränkungen

Gleiches für den *Benutzer* eines Systems

Anforderungen an Dokumentation

Der *Code* ist die beste Dokumentation:

- per Definition akkurat und aktuell
- Semantik ist schwer zu extrahieren
- Dokumentation der Schnittstellen, nicht der Implementierung

Dokumentation gehört in die *Sprache:*

- Code und Dokumentation laufen nicht auseinander
- Compiler und Laufzeitsystem können Konsistenz prüfen

z.B. Vor- und Nachbedingungen, Invarianten

Dokumentation muss *formal* sein

- ermöglicht Verifikation

Ziele der Dokumentation

- Abstraktion von der formalen Beschreibung
- Schnelles Verständnis des Systems
  Hintergründe, Entwurf, typische Nutzung
- Zusammenfassung der Funktionalität
  Dokumentation der Schnittstellen
- Dokumentation der Absicht des Programmierers
  Details der Implementierung
- Wiederverwendung von Code
  durch Suche in der Dokumentation

Kommentare

Kommentare können das Lesen von Programmen erheblich erleichtern.

Aber: Der Compiler kann nicht prüfen,

- ob der Kommentar Sinn macht
- ob sich der Kommentar auf das Programm bezieht
- ob der Kommentar auf dem neuesten Stand ist

Schlechte Kommentare sind schlimmer als gar keine Kommentare!

Wie man *nicht* kommentiert

Grundregel: *Kommentare knapp und präzise halten.*

Statt

```
/*******************************************************
* Start of function S Q R T  (= "square root")         *
* SQRT takes an argument:                              *
* - double X: the number to be squared (must be >= 0)  *
* SQRT returns:                                        *
* - the square root of X                               *
* (c) 2002 H. Becker, Ottweiler - all rights reserved  *
*******************************************************/
```

besser

```
// Return the square root of X
double sqrt(double x) { … }
```

Wie man *nicht* kommentiert (2)

Statt

```
n += 1;   // Increment n by one
a *= n;   // Multiply a with n
return a; // Return the value of a
```

besser

```
n += 1;
a *= n;
return a;
```

Wie man *nicht* kommentiert (3)

Grundregel: *Kann ich etwas im Programmcode ausdrücken, brauche ich keinen Kommentar.*

Statt

```
int d;           // Current number of daffodils
d = num_d();     // Compute number of daffodils
assert (d == 2); // We expect two daffodils
```

besser

```
int current_daffodils = number_of_daffodils();
int expected_daffodils = 2;
assert (current_daffodils == expected_daffodils);
```

Wie man *nicht* kommentiert (4)

Statt

```
// Invariant: x > 0 and y > 0
```

besser

```
assert(x > 0 && y > 0);
```

Wie man kommentiert

Stroustrup (C++) wünscht sich:

- Einen Kommentar für jede Quelldatei
- Einen Kommentar für jede Klasse
- Einen Kommentar für jede nichttrivale Funktion,
  - der den Zweck der Funktion ausdrückt
  - der ggf. die Vorgehensweise (= den Algorithmus) beschreibt
- Einen Kommentar für jede (globale) Variable und Konstante
- Kommentare für nicht-offensichtliche oder nicht-portable Stellen
- “Very little else”

Literate Programming

Motivation:

- Eine Dokumentation muss mit dem Programm zusammen entstehen
- Das Programm ist ein Artefakt einer gute Dokumentation des Lösungsweges

Idee:

- Schreibe die Dokumentation des Systems
- Bette Codefragmente so ein, dass sie extrahiert und zu einem Programm zusammengestellt werden können

Erste Implementierung: WEB [Knuth]

- TeX und Pascal, Tools: Weave und Tangle

Literate Programming (2)

Beispiel von Knuth (wc.w im CWEB-System):

```
...
@ Most \.{CWEB} programs share a common structure. It's probably
...
@c
@<Header files to include@>@/
@<Functions@>@/
@<The main program@>

@ We must include the standard I/O definitions, since we
@ want to send formatted output to |stdout| and |stderr|.

@<Header files...@>=
#include <stdio.h>

@ The |status| variable ....
```

Doxygen

Umgekehrter Ansatz zum Literate Programming:
Integriere Dokumentation in den Quellcode:

- Einfache paralleles Bearbeiten
- Einfache Integration mit CVS

Extraktion der Dokumentation

- Header- und Implementierungsdateien (C, C++, IDL)
- Querverweise Dokumentation ↔ Quellcode

Ausgabe

- Online: HTML und Man-Pages
- Offline: LaTeX und PDF mit Querverweisen

Doxygen (2)

Graphische Darstellungen:

- Vererbung
- Kollaborations-Diagramme, ...(ähnlich UML)

Formatierungsanweisungen:

- Formeln, Aufzählungen, Links, ...

Doxygen: Umgang

Schritt 1: Erzeugen einer Konfigurationsdatei

```
$ doxygen -g
```

Schritt 2: Anpassen der Konfigurationsdatei

```
$ xemacs Doxyfile
```

Anpassen insbesondere von `INPUT`, `FILE_PATTERNS`, `RECURSIVE`

Schritt 3: Doxygen laufen lassen

```
$ doxygen
```

Schritt 4: Dokumentation betrachten

```
$ netscape html/index.html
```

Kommentare in Doxygen

Per Vorgabe nimmt Doxygen nur *spezielle* Kommentare in die Dokumentation auf.

Doxygen unterscheidet

Kurzkommentare (1 Zeile lang)

Detaillierte Kommentare (> 1 Zeile)

Kurzkommentare

Kurzkommentare stehen meist *vor* dem zu beschreibenden Element

```
//! Berechne die Wurzel aus x.
double sqrt(double x) { … }
```

Alternative: Beschreibung *nach* dem Element

```
const double pi = 3.14159…;  //!< Die Zahl pi
```

Diese Form funktioniert *nur* für Konstanten, Attribute, Variablen und Argumente.

Detaillierte Kommentare

Detaillierte Kommentare stehen immer *vor* dem zu beschreibenden Element.

```
/**
 * Berechne die Wurzel aus x mit Hilfe der
 * Newton-Iteration.
 */
double sqrt(double x) { … }
```

Statt „/**“ (Javadoc-Stil) geht auch „/*!“ (Qt-Stil)

Kurz- und Detaillierte Kommentare können gemischt werden:

```
/**
 * \brief berechne die Wurzel aus x.
 *
 * Benutzt Newton-Iteration.
 */
double sqrt(double x) { … }
```

Freie Kommentare

Um ein Element an beliebiger Stelle dokumentieren zu können, muss ein spezielles Kommando eingefügt werden:

```
/** \fn sqrt
 * \brief berechne die Wurzel aus x.
 *
 * Benutzt Newton-Iteration.
 */
```

Dies wird regelmässig benutzt, um die *aktuelle Datei* zu dokumentieren:

```
/** \file
 * Alle Wurzel-Funktionen.
 */
```

Funktionsdokumentation

```
/*! \fn int read(int fd, char *buf, size_t count)
    \brief Read bytes from a file descriptor.
    \param fd The descriptor to read from.
    \param buf The buffer to read into.
    \param count The number of bytes to read.
    \retval The number of bytes read.
*/
```

generiert

> `int read(int fd, char *buf, size_t count)`
> Read bytes from a file descriptor.
>
> **Parameters:**
>
> **fd** The descriptor to read from.
>
> **buf** The buffer to read into.
>
> **count** The number of bytes to read.
>
> **Returns:** the number of bytes read.

Dokumentation für die Hauptseite

Die Gesamt-Dokumentation wird meist in einer eigenen Datei untergebracht

```
/*!
  \mainpage Titel
  ...
  \section sec1 Erstes Kapitel
  ...
  \section sec2 Zweites Kapitel
  ...
 */
```

Querverweise

Automatische Querverweise für alle dokumentierten Objekte

- Wird mit #Objektname vermieden

Manuelle Querverweise

- Referenz auf ein Kapitel: `\ref refname`
- Setzen eines Ankers `\anchor anchorname`
- Setzen eines Links zu einem Anker oder Objekt: `\link anchorname/Objektname ... \endlink`

Formatierung von Text

Schriftarten mit Kommandos

- *Kursiv* – `\e wort` oder `<e>text</e>`
- **Fett** – `\b wort` oder `<b>text</b>`
- Schreibmaschine – `\c wort` oder `<c>text</c>`

Überschriften mit HTML-Formatierungen

```
<h1>Titel</h1>
<h2>Subtitel</h2>
```

Quellcode in \code …\endcode

Formeln

Formeln werden in TEX-Syntax geschrieben und in \f$ …\f$ eingeklammert:

```
//! berechnet \f$\sqrt{x}\f$
double sqrt(double x) { … }
```

erzeugt: „berechnet $\sqrt{x}$“

Einige weitere TEX-Formeln:

`x^n`	x^n
`x_{i'+1}`	$x_{i'+1}$
`\forall\sigma\left(\sigma < \sum_{i = 1}^n\right)`	$\forall s \left(s < \sum_{i=1}^{n}\right)$
`0 \le n \land\lnot\left(n \ge \frac{1}{2}\right)`	$0 \le n \land \neg \left(n \geq \frac{1}{2}\right)$

Formeln (2)

Mit \f[…\f] werden Formeln abgesetzt:

```
\f[
  |I_2|=\left| \int_{0}^T \psi(t)
           \left\{
              u(a,t)-
              \int_{\gamma(t)}^a
              \frac{d\theta}{k(\theta,t)}
              \int_{a}^\theta c(\xi)u_t(\xi,t)\,d\xi
           \right\} dt
        \right|
\f]
```

$$|I_2| = \left| \int_0^T \psi(t) \left\{ u(a,t) - \int_{\gamma(t)}^a \frac{d\theta}{k(\theta,t)} \int_a^\theta c(\xi) u_t(\xi,t)\, d\xi \right\} dt \right|$$

Aufzählungen

Aufzählungen werden mit Spiegelstrichen realisiert;

-# numeriert die einzelnen Punkte durch

```
/*!
 *  A list of events:
 *    - mouse events
 *         -# mouse move event
 *         -# mouse click event\n
 *            More info about the click event.
 *         -# mouse double click event
 *    - keyboard events
 *         -# key down event
 *         -# key up event
 *
 *  More text here.
 */
```

Bilder

Auch Bilder können eingefügt werden. Hier wird zwischen HTML- und LaTeX-Ausgabe unterschieden:

- `\image html` *dateiname* – fügt *dateiname* in HTML-Dokumentation ein
- `\image latex` *dateiname* – fügt *dateiname* in LaTeX-Dokumentation ein

Zusätzlich können noch Titel und Grössenangaben untergebracht werden:

```
/*! Here is a screen shot:
 *  \image html screenshot.jpg
 *  \image latex screenshot.eps "Screen Shot" width=10cm
 */
```

Spezialzeichen

Einige Zeichen haben besondere Bedeutung in Doxygen.
Um sie als Zeichen darzustellen, muss ein Backslash „\“ davorgestellt werden:

\ @ & $ # < >

Die wichtigsten Kommandos

Ein Kommando endet mit Leerzeile oder nächstem Kommando

Allgemein

- `\author`, `\date`, `\version`, `\todo`, `\warning`, `\bug`, `\deprecated`, `\remarks`, `\sa` (see also)

Funktionen

- `\param parametername`, `\retval return_value`, `\exception`
- `\pre`, `\post`, `\invariant`

Kapitel

- `\section refname title`, `\subsection refname title`
- `\par [partitle]`

Konzepte

Benutzerhandbücher müssen mit Blick auf die *Adressaten* verfasst werden.

Mögliche Typen:

- Trainings-Handbuch (⇒ Anfänger)
- Referenz-Handbuch (⇒ Fortgeschrittene)
- Benutzer-Leitfaden (⇒ alle)

Handbücher werden zunehmend durch *Hilfesysteme* ergänzt oder ersetzt.

Konzepte (2)

Wichtige Regeln des Kommentierens:

- *Schlechte Kommentare sind schlimmer als gar keine Kommentare!*
- *Kommentare knapp und präzise halten.*
- *Kann ich etwas im Programmcode ausdrücken, brauche ich keinen Kommentar.*

Literate Programming extrahiert Quellcode aus der Dokumentation.

Doxygen extrahiert Dokumentation aus Quellcode und nutzt hierfür spezielle Kommentare.

http://www.doxygen.org/