Dokumentationswerkzeug Doxygen[1]

David Damm

12. Juli 2004

[1]Dieses Dokument basiert auf `[DvH04]`, Dimitri van Heesch, Manual zum Dokumentationswerkzeug *Doxygen* (siehe `www.doxygen.org`).

Inhaltsverzeichnis

Kapitel 1

Einleitung

Doxygen ist ein Open-Source-Dokumentationswerkzeug, um innerhalb von Quelltexten zu dokumentieren. Mit dem Programm lassen sich alle auf *C* basierenden Pogrammiersprachen verwenden, also im Falle des *XCTL*-Projektes *C++*. Im Gegensatz zu *JavaDoc* unterstützt *Doxygen* neben einer *HTML*-Dokumentation zusätzlich die Erzeugung von LaTeX-Code, *RTF*-Dateien oder Manual-Seiten.

Doxygen läßt sich mit Hilfe einer Konfigurationsdatei, in der man eine Vielzahl von Optionen einstellen kann, anpassen. Diese Konfigurationsdatei kann mit einem beliebigen Texteditor modifiziert werden, oder man verwendet das mitgelieferte Programm *Doxywizard*.

Kapitel 2

Installation

2.1 Windows

Für die Installation von *Doxygen* unter *Windows* kann man zwischen zwei Varianten wählen. Entweder man lädt sich die Datei `doxygen-v.v.v-setup.exe` von `[www.doxygen.org]` herunter, wobei `v.v.v` die Versionsnummer angibt, und startet das sich selbst extrahierende Archiv mittels Doppelklick, oder man wählt die ZIP-Datei `doxygen-v.v.v.windows.bin.zip` und entpackt die Dateien in das gewünschte Installationsverzeichnis (z.B. `C:\Programme\Doxygen`).

In der EXE-Datei ist auch der *Doxywizard* zum Modifizieren der Einstellungen für *Doxygen* enthalten.

2.2 Linux

Um *Doxygen* unter *Linux* zu installieren, benötigt man die *Linux*-Distribution von `[www.doxygen.org]`, die man sich im Bereich Download herunterladen kann.

Das Dokumentationswerkzeug befindet sich in dem Archiv `doxygen-v.v.v.linux.bin.tar.gz`, wobei `v.v.v` die Versionsnummer angibt. Möchte man zusätzlich den *Doxywizard* benutzen, so muß man ein weiteres Archiv `doxywizard-v.v.v.linux.bin.tar.gz`, in dem nur dieser enthalten ist, herunterladen.

Um *Doxygen* in einem lokalen Nutzerverzeichnis zu installieren, muß man nun die Dateien im Home-Verzeichnis entpacken. Dazu entzippt man die Dateien zunächst und entpackt die Archive mit

```
shell> tar -xzf doxygen-v.v.v.linux.bin.tar.gz
shell> tar -xzf doxywizard-v.v.v.linux.bin.tar.gz
```

in das automatisch erstellte Verzeichnis `doxygen-v.v.v`. (Die Dateien lassen sich natürlich auch in einen beliebigen anderen Ordner kopieren.)

Nun befinden sich in dem Ordner `doxygen-v.v.v/bin/` die Pogramme

- `doxygen`, zum Erstellen der Dokumentation,
- `doxywizard`, zum komfortablen Einstellen der Optionen für *Doxygen* und
- `doxytag`, zum Zusammenstellen verschiedener Dokumentationen.

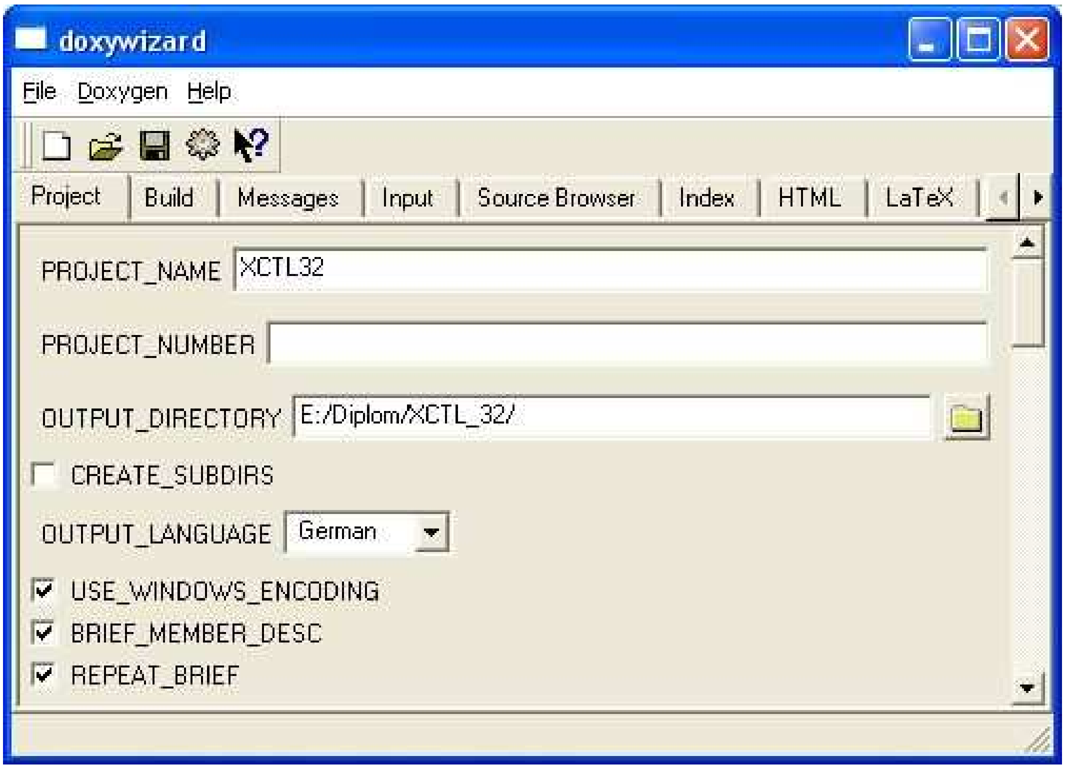

Abbildung 2.1: *Doxywizard* unter Windows

Für das *XCTL*-Projekt genügen die ersten beiden Tools, da eine Dokumentation für das gesamte Projekt erzeugt werden soll.

Doxygen kann nun mit Hilfe einer Shell

```
shell> doxygen-v.v.v/bin/doxygen
```

oder innerhalb von *Doxywizard* (mit dem Zahnrad-Button aus der Menüleiste) aufgerufen werden, nachdem die Konfigurationsdatei erstellt und den eigenen Wünschen angepasst wurde.

Kapitel 3

Konfiguration

Um *Doxygen* verwenden zu können, muß zuvor eine Konfigurationsdatei erzeugt werden. Das geschieht mit dem Aufruf

```
doxygen -g setup.dox
```

innerhalb einer Shell (unter *Linux*) oder in der Eingabeaufforderung (unter *Windows*). Man kann auch den *Doxywizard* starten und die Konfigurationsdatei unter dem Namen `setup.dox` (oder einem beliebigen anderen Namen) speichern.

Die generierte Datei `setup.dox` enthält nun alle standardmäßig gesetzten Werte für *Doxygen*. Diese Werte lassen sich nun entweder mit einem Texteditor oder mit dem mitgelieferten Werkzeug *Doxywizard* und dem Aufruf

```
doxywizard setup.dox
```

verändern.

Dann sollte man alle Dateien `*.dox` mit dem *Doxywizard* verknüpfen, so dass alle Konfigurationsdateien für *Doxygen* mit einem Doppelklick (unter Windows) oder einfachem Klick (unter Linux) mit dem *Doxywizard* geöffnet werden.

In den nächsten Abschnitten werden nur die wichtigsten Parameter erwähnt (für eine ausführliche Beschreibung der Parameter siehe [`www.doxygen.org`]).

3.1 Projekteinstellungen

Startet man den *Doxywizard*, so ist zunächst die Kartei `Project` aktiviert. In dieser lassen sich allgemeine Einstellungen für das Projekt vornehmen.

Dort findet sich ein Textfeld, zum Festlegen des Projektnamens (`PROJEKT_NAME`). Im Falle des *XCTL*-Projektes sollte dort `XCTL32` eingetragen werden.

Im Feld `OUTPUT_DIRECTORY` wird der Pfad für die erzeugten Dokumentationen angegeben, so dass alle von *Doxygen* erstellten Dateien in diesem Verzeichnis gespeichert werden. Aktiviert man zusätzlich die Option `CREATE_SUBDIRS`, so werden die Dateien in Unterverzeichnisse gegliedert. Für das *XCTL*-Projekt sollte diese Option deaktiviert bleiben.

Weiterhin läßt sich die Sprache für die Dokumentation mit `OUTPUT_LANGUAGE` festlegen. Da im *XCTL*-Projekt vorwiegend Deutsch verwendet wird, sollte hier `German` eingestellt werden.

Die letzte Projektoption, die noch erwähnt werden soll, ist `JAVADOC_AUTOBRIEF`. Das aktivieren dieser Option führt dazu, das jeder Kommentarblock stets mit einer kurzen Beschreibung beginnt (siehe Kapitel 4 auf Seite 8). Diese Option wird für das *XCTL*-Projekt aktiviert.

3.2 Optionen zum Erstellen der Dokumentation

Aktiviert man die zweite Kartei `Build` im *Doxywizard*, so kann man zahlreiche Optionen, die das Erzeugen der Dokumentation beeinflussen, verändern. Hier betrachten wir nur die Option `EXTRACT_ALL`. Sofern diese Option aktiviert ist, werden alle Dokumente im angegebenen Pfad geparst und somit für alle Klassen, Funktionen oder Variablen Dokumentationen erstellt, auch wenn diese noch nicht mit *Doxygen* dokumentiert wurden. Deaktiviert man die Option, so werden nur jene Dateien geparst, in denen tatsächlich *Doxygen*-Anweisungen auftauchen.

Für das *XCTL*-Projekt eignen sich beide Varianten, je nachdem, ob man einen vollständigen Überblick über alle Klassen haben möchte (`EXTRACT_ALL` aktiviert) oder, ob man Herausfinden möchte, welche Klassen noch dokumentiert werden müssen (`EXTRACT_ALL` deaktiviert).

3.3 Eingabedateien festlegen

In der Kartei `Input` legt man die Einstellungen für die Eingabedateien fest. Unter dem Punkt `INPUT` können verschiedene Quellen angegeben werden, die dann später beim Erstellen der Dokumentation von *Doxygen* geparst werden. Die einfachste Lösung für das *XCTL*-Projekt ist, das Wurzelverzeichnis `XCTL32` anzugeben, und dann zusätzlich die Option `RECURSIVE` zu aktivieren, damit auch alle Unterverzeichnisse im Verzeichnis `XCTL32` durchsucht werden.

Dann müssen noch die Dateiendungen (`FILE_PATTERNS`) von den Dateien angegeben werden, die geparst werden sollen, d.h. `*.cpp` und `*.h` im Falle des *XCTL*-Projektes. (Dateien mit diesen Endungen werden aber auch automatisch geparst, sofern kein Pattern angegeben wurde.)

Möchte man einige Dateien oder Verzeichniss nicht parsen lassen, so kann man diese explizit bei `EXCLUDE` angeben.

3.4 Dokumentation erzeugen

Die Einstellungen zum Erstellen der Dokumentation lassen sich dann für jeden einzelnen Dokumentationstyp (*HTML*, LaTeX, *RTF*, *MAN*, *XML*) separat festlegen.

Für das *XCTL*-Projekt sind nur die *HTML*- und LaTeX-Dokumentation von Bedeutung. Deshalb sollte `GENERATE_HTML` und `GENERATE_LATEX` aktiviert werden. Möchte man zusätzlich eine *RTF*-Dokumentation oder Manual-Seiten erzeugen, so muß `GENERATE_RTF` bzw. `GENERATE_MAN` in den jeweiligen Karteifenstern aktiviert werden.

Die generierte Dokumentation wird im Verzeichnis, das bei `HTML_OUTPUT` oder `LATEX_OUTPUT` angegeben ist, gespeichert (und zwar als Unterverzeichnis im Verzeichnis `OUTPUT_DIRECTORY`, siehe Projekteinstellungen).

Mit `HTML_FILE_EXTENSION` kann die Dateiendung für die erzeugten *HTML*-Dateien angegeben werden (z.B. `*.html` oder `*.htm`).

3.5 Konfigurationsbeispiel

Der folgende Auszug stammt aus der Beispieldatei `Setup.dox`, die zuerst mit *Doxygen* generiert und dann mit *Doxywizard* modifiziert und dem *XCTL*-Projekt angepasst wurde. Die besonders wichtigen Einstellungen wurden hervorgehoben.

```
# Doxyfile 1.3.7
#---------------------------------------------------------------------------
# Project related configuration options
#---------------------------------------------------------------------------
PROJECT_NAME = XCTL32
PROJECT_NUMBER =
OUTPUT_DIRECTORY = /u/ddamm/disk/doxygen/doxygenbeispiel/
CREATE_SUBDIRS = NO
OUTPUT_LANGUAGE = German
...
JAVADOC_AUTOBRIEF = YES
...
#---------------------------------------------------------------------------
# Build related configuration options
#---------------------------------------------------------------------------
EXTRACT_ALL = NO
...
#---------------------------------------------------------------------------
# configuration options related to the input files
#---------------------------------------------------------------------------
INPUT = /u/ddamm/disk/doxygen/doxygenbeispiel/
FILE_PATTERNS =
RECURSIVE = YES
EXCLUDE =
...
#---------------------------------------------------------------------------
# configuration options related to the HTML output
#---------------------------------------------------------------------------
GENERATE_HTML = YES
HTML_OUTPUT = html
HTML_FILE_EXTENSION = .html
HTML_HEADER =
HTML_FOOTER =
...
#---------------------------------------------------------------------------
# configuration options related to the LaTeX output
#---------------------------------------------------------------------------
GENERATE_LATEX = YES
LATEX_OUTPUT = latex
LATEX_CMD_NAME = latex
...
#---------------------------------------------------------------------------
# Configuration::additions related to the search engine
#---------------------------------------------------------------------------
SEARCHENGINE = NO
```

Kapitel 4

Dokumentation

Der Dokumentationsstil für das *XCTL32*-Projekt soll sich an *JavaDoc* orientieren, d.h. alle Kommentare werden im *JavaDoc*-Stil geschrieben.

Kommentare, die von *Doxygen* verarbeitet sollen, müssen dabei folgende Gestalt haben: es muss ein *C++* Kommentarblock mit zwei Sternchen zu Beginn verwendet werden.

```
/** Eine Dokumentation.
 *  Ausführliche Dokumentation ...
 *  ...
 */
```

In der ersten Zeile des Kommentares wird eine kurze Beschreibung der Klasse (oder Methode, Variable, ...) angegeben. Diese kurze Beschreibung endet bei dem ersten Punkt, nach dem ein Leerzeichen oder das Zeilenende folgt (sofern die Variable `JAVADOC_AUTOBRIEF` gesetzt wurde). Danach wird die ausführliche Dokumentation angegeben.

Möchte man nur einen kurzen Kommentar in einer Zeile hinzufügen, so genügt

```
//* kurzer Kommentar
```

um diesen zu erstellen. Der Kommentar geht dann bis zum Zeilenende und wird als ein neuer Absatz interpretiert. Fügt man also mehrere Kurzkommentare hintereinander ein, so wird jede Zeile als ein neuer Absatz in der Dokumentation dargestellt.

Alle Kommentare dürfen *HTML*-Code enthalten, der ebenfalls von *Doxygen* entsprechend interpretiert wird. So lassen sich zum Beispiel auch *HTML*-Kommentare einfügen, die dann in der erzeugten Dokumentation zwar im Quellcode stehen, aber nicht in der Dokumentation selbst angezeigt werden.

```
//* <!-- HTML-Kommentar --> Doxygen-Kommentar
```

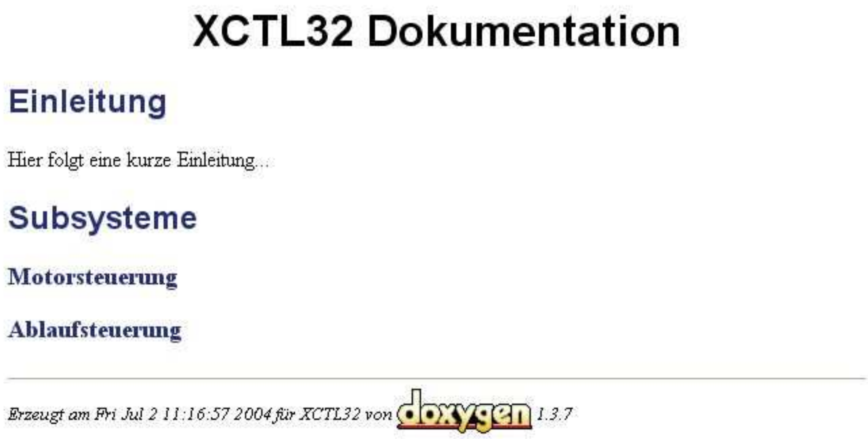

Abbildung 4.1: Dokumentationsstartseite mit *Doxygen*

4.1 Hauptseite

Die Hauptseite wird als erstes angezeigt, wenn man die erstellte Dokumentation (`html/index.html`) öffnet. Sie soll einen Überblick über das gesamte Projekt vermitteln. Im Falle des *XCTL*-Projektes eignet sich eine allgemeine Beschreibung des Projektes und jeweils eine kurze Beschreibung der Subsysteme.

Um den Text formatiert darzustellen, gibt es die folgenden Befehle.

```
@mainpage <titel>
@section <name> <titel>
@subsection <name> <titel>
```

Der erste Befehl signalisiert, dass der Kommentarblock eine Beschreibung für die Haupseite des Projektes enthält. Normalerweise erfolgt diese Beschreibung in einer von der Implementation getrennten Datei (z.B. `Project.h`) im Wurzelverzeichnis des Projektes. Die Überschrift der Startseite wird im `<titel>` festgelegt.

Mittels `@section` und `@subsection` lassen sich nun Kapitel festlegen, wobei `<titel>` den Titel des Kapitels angibt, und `<name>` als Anker für einen Verweis dient. So kann mit `@see` oder `@link` an anderer Stelle auf das entsprechende Kapitel verwiesen werden.

```
/** @mainpage XCTL
 *
 *  @section sec1 Einleitung
 *  Hier folgt eine kurze Einleitung...
 *
 *  @section sec2 Subsysteme
 *  @subsection sec2_1 Motorsteuerung
 *  @subsection sec2_2 Ablaufsteuerung
 */
```

Die von *Doxygen* erzeugte zugehörige Dokumentation wird in Abbildung 4.1 dargestellt.

4.2 Klassen und Dateien

Es besteht die Möglichkeit, mit dem Befehl

```
@file <name>
```

den Inhalt der mit `<name>` spezifizierten Datei zu beschreiben. Sinnvollerweise gehört dieser Befehl an den Anfang einer jeden dokumentierten Datei.

Die Beschreibung der Datei erfolgt in einem extra Kommentarblock, wobei die erste Zeile bis zum Punkt für eine Kurzbeschreibung reserviert ist, falls `JAVADOC_AUTOBRIEF` eingeschaltet ist. Danach folgt die ausführliche Beschreibung des Dateiinhaltes. *Doxygen* generiert dann eine Tabelle, in der alle Dateien mit der zugehörigen Kurzbeschreibung aufgelistet werden.

Jede Klasse sollte im Normalfall in einem eigenen Dokument (`*.h`, `*.cpp`) stehen. Die allgemeine Dokumentation der Klasse gehört dann in die Headerdatei (`*.h`), während spezielle Implementationsdetails an den entsprechenden Stellen im Quelltext (`*.cpp`) beschrieben werden sollten.

In den Kopf einer jeden Headerdatei sollten die Autoren des Quelltextes, das Datum der letzten Änderung und die Dokumentversion geschrieben werden.

Dafür stellt *Doxygen* die folgenden Kommandos zur Verfügung.

Mittels

```
@author <name>
```

können die Autoren des Dokumentes festgelegt werden. Es können mehrere Autoren angegeben werden, indem weitere `@author` hinzugefügt werden. Die Namen erscheinen dann beim Erstellen der Dokumentation in den jeweiligen Klassen.

Das Kommando

```
@date <datum>
```

legt das Datum fest, an dem zuletzt an der Datei gearbeitet wurde. Es genügt, wenn das Datum nur einmal im Kopf der Datei verwendet wird und nicht bei jeder einzelnen Methode, da man sonst den Überblick verliert.

Mit der Anweisung

```
@version <versionsnummer>
```

wird die Version des Dokumentes festgehalten. Nach dieser Anweisung sollte dann auch noch knapp beschrieben werden, was sich zur vorherigen Version geändert hat, um die Entwicklung des Quelltextes nachvollziehen zu können.

Eine Headerdatei könnte damit folgenden Kopf enthalten:

```
/** Klasse realisiert Beispiel2.
 *
 *  @author David Damm
 *  @date 2.7.2004
 *
 *  @version 0.2
```

Beispiel2 Klassenreferenz

Klasse realisiert Beispiel2. Mehr...

`#include <Beispiel2.h>`

Ausführliche Beschreibung

Klasse realisiert Beispiel2.

Autor:
David Damm

Datum:
2.7.2004

Version:
0.2 Toten Code entfernt.

0.1 Kommentare hinzugefügt (Doxygen).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- **Beispiel2.h**

Erzeugt am Fri Jul 2 11:25:57 2004 für XCTL32 von doxygen 1.3.7

Abbildung 4.2: Klassenbeschreibung mit *Doxygen*

```
 *  Toten Code entfernt.
 *
 *  @version 0.1
 *  Kommentare hinzugefügt (Doxygen).
 */
class Beispiel2
{
}
```

Doxygen erzeugt dann für die Klasse `Beispiel2` die in Abbildung 4.2 dargestellte Dokumentation. Dabei fällt auf, dass die beiden `@version`-Kommandos zu einer Gruppe zusammengefaßt wurden. Die Versionsnummern erscheinen jeweils als ein neuer Absatz.

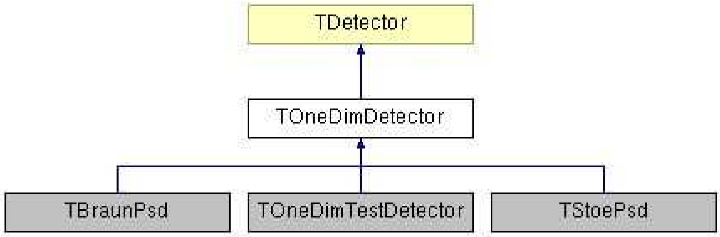

Abbildung 4.3: Klassendiagramm in *Doxygen*

4.3 Funktionen und Variablen

Es sollten auf jeden Fall alle Variablen und Funktionen dokumentiert werden, egal ob sie `private`, `protected` oder `public` sind. Doxygen erzeugt zwar für die geschützten Attribute oder Funktionen keine Dokumentation, doch ist die Beschreibung nützlich für das Gesamtverständnis für die Entwickler, die später Änderungen an einem Subsystem (oder einer Klasse) vornehmen oder Neuerungen hinzufügen wollen.

Außerdem wird automatisch zu jeder Klasse ein Klassendiagramm erstellt, sofern diese Klasse von einer anderen abgeleitet wurde oder selbst Ableitungen besitzt. Die im Diagramm dargestellten Klassen werden dann auf die jeweilige Dokumentation der Klasse verlinkt, so daß ein leichtes und schnelles Navigieren möglich ist.
In der Abbildung 4.3 wird ein solches Klassendiagramm am Beispiel der Detektoren gezeigt. Die aktuell angezeigte Klassendokumentation verbirgt sich hinter dem weißen Kästchen. Gelbe Kästchen signalisieren mittels *Doxygen* dokumentierte Klassen, während grau hinterlegte Klassen nicht dokumentiert (und damit auch nicht verlinkt) sind.
Ein durchgehender dunkelblauer Pfeil gibt an, dass es sich um eine öffentliche (`public`) Vererbung handelt. Ist der Pfeil gestrichelt und dunkelgrün, so handelt es sich um eine geschützte (`protected`) Vererbung, und ist er gepunktet und dunkelgrün, so erfolgt eine private (`private`) Vererbung.

Um eine Funktion zu dokumentieren, stellt *Doxygen* verschiedene Befehle zur Verfügung. Mit

```
@param[in,out] <bezeichner> <beschreibung>
```

lassen sich alle Parameter einer Funktion beschreiben. Mit `[in]` gibt man an, dass es sich um eine Eingabe handelt und mit `[out]`, dass es ein Ausgabeparameter ist. Wird der Parameter für Ein- und Ausgabe verwendet, so muß man `[in,out]` angeben. Dabei entspricht der `<bezeichner>` dem Namen des Parameters im Funktionskopf. In der `<beschreibung>` wird angegeben, wofür der Parameter verwendet wird.

Der Rückgabewert einer Funktion wird mittels

```
@return <beschreibung>
```

dokumentiert. Sollte die Funktion keinen Rückgabewert besitzen, so kann dieses Kommando weggelassen werden.

Weitere nützliche Kommandos, um Funktionen oder Variablen zu beschreiben sind im folgenden aufgeführt.

```
@see <name>
@todo <beschreibung>
@bug <fehler>
@warning <warnung>
@test <testfall>
```

Dokumentation der Elementfunktionen

**int Beispiel3::Addiere(int *a*,
int *b*
)**

Addiert zwei ganze Zahlen.

Parameter:
[in] *a* Die erste Zahl.
[in] *b* Die zweite Zahl.

Rückgabe:
Die Summe der beiden Zahlen a und b.

Siehe auch:
Addiere(double,double)

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- **Beispiel3.h**
- Beispiel3.cpp

Erzeugt am Fri Jul 2 11:43:35 2004 für XCTL32 von doxygen *1.3.7*

Abbildung 4.4: Dokumentation von Elementfunktionen

Mit dem ersten Befehl lassen sich Verweise auf andere Funktionen oder Variablen, die mit der soeben beschriebenen Funktion oder Variable in gewissem Zusammenhang stehen, erstellen (siehe Beispiel 3).

Mit `@todo` können noch zu erledigende Tätigkeiten festgehalten werden. Das Kommando `@bug` dient zum dokumentieren erkannter Fehler. *Doxygen* generiert daraus jeweils eine Todo-Liste und eine Fehler-Liste.

Mittels `@warning` kann auf eine fehlerhafte Benutzung der Funktion oder Variable hingewiesen werden. Bei dem `@test`-Kommando kann man schon spätere Testfälle spezifizieren.

```
/** Klasse realisiert Beispiel3.
 *
 *  @author David Damm
 *  @date 2.7.2004
 */
class Beispiel3
{
  public:
  /** Addiert zwei ganze Zahlen.
   *  @param[in] a Die erste Zahl.
   *  @param[in] b Die zweite Zahl.
   *  @return Die Summe der beiden Zahlen a und b.
   *  @see Addiere(double,double)
   */
  int Addiere(int a, int b);
  double Addiere(double a, double b);
}
```

Ein Auszug aus der generierten Dokumentation wird in Abbildung 4.4 gezeigt.

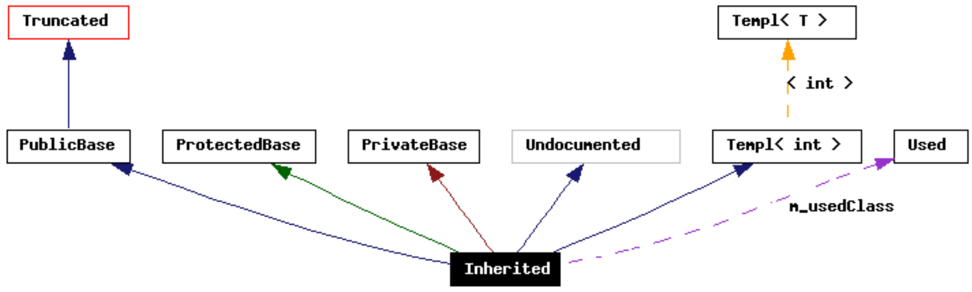

Abbildung 4.5: Graphen und Diagramme mit *Dot*

4.4 Graphen und Diagramme

Doxygen kann für *C++*-Klassen automatisch Klassendiagramme erstellen. Möchte man aber auf weitere Graphen und Diagramme zugreifen, so benötigt man zusätzlich das Werkzeug *Dot*. *Dot* ist ebenfalls ein Open-Source-Werkzeug und kann unter der Adresse `[www.research.att.com/sw/tools/graphviz]` heruntergeladen werden.

Für *Linux* kann man den Quellcode `graphviz-v.v.tar.gz` herunterladen und einem lokalen Verzeichnis installieren. Dazu entpackt man die Datei

```
shell> tar -xzf graphviz-v.v.tar.gz
shell> configure --prefix=<pfad>
shell> make install
```

und konfiguriert das Programm zunächst, indem man bei `<pfad>` das Installationsverzeichnis angibt. Danach muß noch *Make* aufgerufen werden, um das Programm zu installieren. *Dot* befindet sich dann im Verzeichnis `<pfad>/bin/`.

Für die *Windows*-Installation lädt man `graphviz-v.v.exe` herunter und installiert das Programm im gewünschten Verzeichnis (z.B. `C:\Programme\Graphviz`).

Wenn man nun das Werkzeug *Dot* installiert hat, kann man dieses auch in *Doxygen* verwenden. Dazu startet man den *Doxywizard* und aktiviert das Karteifenster `Dot`. Dort kann man nun die Option `HAVE_DOT` aktivieren und muß bei `DOT_PATH` den Pfad zu dem Programm *Dot* angeben.

Neben einem Klassendiagramm (Option `CLASS_GRAPH`) lassen sich nun mit *Doxygen* auch ein Kollaborationsdiagramm (Option `COLLABORATION_GRAPH`), eine grafische Veranschaulichung der Klassenhierarchie (Option `GRAPHICAL_HIERARCHY`), eine Darstellung der `include`-Abhängigkeiten (Option `INCLUDE_GRAPH`) und ein Graph, der die Aufrufstruktur (Option `CALL_GRAPH`) veranschaulicht, erstellen.

Die Rechtecke in der Abbildung 4.5 haben dabei folgende Bedeutung:

- Ein schwarz gefülltes Rechteck stellt die Struktur oder Klasse dar, für die der Graph erzeugt wurde.
- Ein Rechteck mit schwarzem Rahmen kennzeichnet eine dokumentierte Struktur oder Klasse.
- Ein Rechteck mit grauem Rahmen kennzeichnet eine undokumentierte Struktur oder Klasse.
- Ein Rechteck mit rotem Rahmen kennzeichnet eine dokumentierte Struktur oder Klasse, für die nicht alle Vererbungs-/Enthaltenseinsbeziehungen dargestellt werden. Ein Graph wird gekürzt, wenn er nicht in die angegebenen Schranken paßt.

Diese Schranken können unter der Angabe der Breite bei `MAX_DOT_GRAPH_WIDTH` sowie der Höhe bei `MAX_DOT_GRAPH_HEIGHT` verändert werden. Beide Werte sind auf 1024 Pixel voreingestellt und sollten je nach Monitorauflösung angepaßt werden.

Die Pfeile in der Abbildung 4.5 bedeuten:

- Ein dunkelblauer Pfeil stellt eine öffentliche Vererbungsbeziehung zwischen zwei Klassen dar.
- Ein dunkelgrüner Pfeil stellt eine geschützte Vererbung dar.
- Ein dunkelroter Pfeil stellt eine private Vererbung dar.
- Ein gestrichelter violetter Pfeil bedeutet, dass eine Klasse in einer anderen enthalten ist oder von einer anderen benutzt wird. Am Pfeil stehen die Variablen, mit deren Hilfe auf die Struktur oder Klasse an der Pfeilspitze zugegriffen werden kann.
- Ein gestrichelter gelber Pfeil kennzeichnet eine Verknüpfung zwischen einer Template-Instanz und der Template-Klasse von welcher es abstammt. Neben dem Pfeil sind die Template-Parameter ausgeführt.

4.5 Listen

Einfache Listen lassen sich durch das Kommando

```
@li <listenelement>
```

erstellen. Das Kommando hat ein Argument und wird entweder durch eine leere Zeile beendet oder durch ein weiteres `@li`. Mit Hilfe dieses Kommandos lassen sich nur einfache Listen erstellen. Möchte man aber trotzdem verschachtelte Listen verwenden, so kann man auf den *HTML*-Standard zurückgreifen oder man erzeugt die Listen in der folgenden Art und Weise.

Ein einfacher Listeneintrag wird durch ein Minus (`-`) gekennzeichnet. Möchte man eine nummerierte Liste erzeugen, so muß man hinter dem Minus noch eine Raute angeben (`-#`). Um eine untergeordnete Liste zu erstellen, müssen die jeweiligen Anstriche um eine feste Anzahl Leerzeichen eingerückt werden. Alle Anstriche, die dann in der selben Spalte beginnen, gehören zur selben Liste.

```
/**
 *  - Liste 1a
 *    -# Liste 1a.1
 *    -# Liste 1a.2
 *  - Liste 1b
 */
```

Das obige Beispiel erzeugt außen eine Liste ohne Nummerierung. Das erste Listenelement enthölt wiederum eine nummerierte unterliste mit zwei Einträgen.

Im folgenden ausführlicheren Beispiel sind zwei nacheinanderfolgende einfache Listen und eine verschachtelte Liste mittels *HTML*-Code implementiert. *Doxygen* erzeugt dann für diese Listen den in Abbildung 4.6 dargestellten Dokumentationstext.

```
/** Darstellung von Listen.
 *
 *  Einfache Listen:
 *  @li Liste 1a
```

Ausführliche Beschreibung

Darstellung von Listen.

Einfache Listen:

- Liste 1a
- Liste 1b

- Liste 2a
- Liste 2b

Verschachtelte Listen:

- Liste 3a
  1. Liste 3a.1
  2. Liste 3a.2
- Liste 3b

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- **Beispiel4.h**

Erzeugt am Fri Jul 2 11:51:55 2004 für XCTL32 von doxygen 1.3.7

Abbildung 4.6: Erstellung von Listen mit *Doxygen*

```
 *  @li Liste 1b
 *
 *  @li Liste 2a
 *  @li Liste 2b
 *
 *  Verschachtelte Listen:
 *  <ul>
 *      <li>Liste 3a</li>
 *      <ol>
 *        <li>Liste 3a.1</li>
 *        <li>Liste 3a.2</li>
 *      </ol>
 *      <li>Liste 3b</li>
 *  </ul>
 */
class Beispiel4
{
}
```

4.6 Links

Doxygen generiert für alle in der Dokumentation vorkommenden Dateien, Klassen, Funktionen ... automatisch Links, so dass zum Beispiel aus der Angabe eines Klassennamens ein Link zu dieser Klasse generiert wird.

HTML	Bedeutung
`<a href="..."></a>`	Erstellt einen Hyperlink.
`<p></p>`	Kennzeichnet einen Absatz.
`<b></b>`	Text wird fett dargestellt.
`<em></em>` oder `<i></i>`	Text wird kursiv dargestellt.
`<code></code>`	Verwendet als Schriftart Schreibmaschine.
` `	Ein Zeilenumbruch.
`<hr>`	Erzeugt eine horizontale Linie.
`<center></center>`	Richtet den Text zentriert aus.
`<ul></ul>`	Erzeugt eine ungeordnete Liste.
`<ol></ol>`	Erzeugt eine durchnummerierte Liste.
`<li></li>`	Ein Listenelement.

Tabelle 4.1: *HTML*-Kommandos

Um andere Elemente untereinander zu verlinken, kann man die Anweisung

```
@link <link-objekt> <link-name> @endlink
```

benutzen. Dabei muß das zu verlinkende Ziel `<link-objekt>` angegeben werden. Der Name des Link-Objektes `<link-name>` wird für den Hyperlink in der Dokumentation verwendet. Ein Link endet mit `@endlink`.

Es besteht auch die Möglichkeit, das *HTML*-Kommando zu verwenden, um einen Link zu erzeugen (`<a href="..."></a>`).

4.7 HTML-Kommandos

Doxygen versteht auch *HTML*-Kommandos und interpriert diese entsprechend. Die Tabelle 4.1 gibt einen kurzen Überblick über häufig verwendete *HTML*-Tags. Für eine vollständige Auflistung der Kommandos sei auf das Manual [DvH04] verwiesen.

4.8 Sonderzeichen

Da einige Zeichen eine besondere Bedeutung haben, müssen diese, sofern sie in der Dokumentation als einzelnes Zeichen (z.B. `'#'`) verwendet werden sollen, durch einen zusätzlichen Backslash gekennzeichnet werden (siehe Tabelle 4.2).

4.9 Mathematische Formeln

Doxygen bietet die Möglichkeit, LaTeX-Formeln in die Dokumentation zu integrieren. Dieses funktioniert aber nur für die *HTML*- oder LaTeX-Ausgabe und nicht für *RTF*-Dateien oder Manuals.

Um Formeln in Form von Bildern in die *HTML*-Dokumentation einzubauen, müssen die folgenden drei Werkzeuge verfügbar sein:

- `latex`, zum Parsen der Formeln.

Doxygen	Zeichen
`'\$'`	`'$'`
`'\@'`	`'@'`
`'\\'`	`'\'`
`'\&'`	`'&'`
`'\~'`	`'~'`
`'\<'`	`'<'`
`'\>'`	`'>'`
`'\#'`	`'#'`
`'\%'`	`'%'`

Tabelle 4.2: Sonderzeichen in *Doxygen*

- `dvips`, um DVI-Dateien nach *PostScript* zu konvertieren
- `gs`, um *PostScript*-Dateien in *Bitmaps* umzuwandeln.

Es gibt zwei Möglichkeiten, um Formeln in die Dokumentation einzubinden, und zwar

- innerhalb eines Textes (zwischen `\f$ ... \f$`) oder
- als abgesetzte Formel (zwischen `\f[ ... \f]`).

Alle Formeln müssen gültige Kommandos im Mathematik-Modus von LaTeXsein.

```
/**
 *  Der Abstand zwischen \f$(x_1,y_1)\f$ und \f$(x_2,y_2)\f$ ist
 *  \f[\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}.\f]
 */
```

Das Ergebnis des obigen Beispiels hat dann folgende Gestalt:

Der Abstand zwischen (x_1,y_1) und (x_2,y_2) ist

$$\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}.$$

4.10 Zusammenfassung

Die Tabelle soll eine Zusammenfassung über alle hier besprochenen Befehle und deren Bedeutung geben.

Befehl	Verwendung
`@mainpage <titel>`	Beschreibung der Haupseite der Dokumentation.
`@section <name> <titel>`	Beginnt ein Kapitel.
`@subsection <name> <titel>`	Beginnt ein Unterkapitel.
`@file <name>`	Beschreibung einer Datei.
`@author <name>`	Name des Autors.
`@date <datum>`	Datum der letzten Änderung.
`@version <versionsnummer>`	Versionsnummer des Dokumentes.
`@param[in,out] <bezeichner> <beschreibung>`	Beschreibung von Funktionsparametern.
`@return <beschreibung>`	Rückgabewert einer Funktion.
`@see <name>`	Verweis an eine andere Stelle.
`@todo <beschreibung>`	Auflistung von zu erledigenden Tätigkeiten.
`@bug <fehler>`	Auflistung von Fehlern.
`@warning <warnung>`	Warnungen zur Benutzung einer Funktion o.ä.
`@test <testfall>`	Spezifikation von Testfällen.
`@li <listenelement>`	Erzeugen einer Liste.
`@link <link-objekt> <link-name> @endlink`	Erstellen eines Links.

Tabelle 4.3: Übersicht einiger *Doxygen*-Befehle

Index