Achim Köhler

Der C/C++-Projektbegleiter

C/C++ Projekte planen, dokumentieren, bauen und testen

dpunkt.verlag

Lektorat: Dr. Michael Barabas
Copy-Editing: Annette Schwarz, Ditzingen
Satz: Achim Köhler
Herstellung: Nadine Berthel
Umschlaggestaltung: Helmut Kraus, www.exclam.de
Druck und Bindung: Koninklijke Wöhrmann B.V., Zutphen, Niederlande

Bibliografische Information Der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.ddb.de> abrufbar.

ISBN 978-3-89864-470-9

1. Auflage 2007

Ringstraße 19
69115 Heidelberg

5 4 3 2 1 0

Vorwort

Problemstellung und Gliederung

Die Anforderungen an den Funktionsumfang, die Leistung und die Qualität moderner Softwareprodukte nehmen nicht erst seit gestern beständig zu. Das Entwicklungsbudget und die terminlichen Rahmenbedingungen hingegen entwickeln sich zumeist in die entgegengesetzte Richtung.

Um diese Herausforderungen zu bewältigen, sind Maßnahmen auf breiter Front erforderlich. Der Software-Entwicklungsprozess an sich muss genauso betrachtet werden wie die einzelnen Aktivitäten der Softwareentwicklung. Dazu gehört das Projektmanagement, die Anforderungsanalyse, Design, Implementierung, Qualitätssicherung, Qualitätskontrolle, Versionsverwaltung, Dokumentation, Konfigurationsmanagement und das Software-Deployment. Es ist festzustellen, dass zur Unterstützung bei vielen dieser Aktivitäten inzwischen eine ganze Reihe freier Tools verfügbar ist.

In diesem Buch werden Vorgehensweisen und Werkzeuge vorgestellt, die schwerpunktmäßig in den Bereichen Implementierung, Qualitätssicherung, Qualitätskontrolle und Dokumentation zur Anwendung kommen. Im Vordergrund stehen nicht die theoretischen Grundlagen und Konzepte. Vielmehr wird versucht, die Themen mit klarem Bezug zur täglichen Praxis und so »nah am Code« wie möglich zu behandeln. Mit »Code« ist dabei *C++*- und C-Code gemeint. Das bedeutet, dass ausschließlich solche Methoden und Werkzeuge betrachtet werden, die für Projekte auf der Basis der Programmiersprachen *C* und *C++* von Interesse sind.

Abbildung 1 verdeutlicht, welche Bereiche der Softwareentwicklung durch die einzelnen Kapitel des Buches adressiert werden.

Kapitel 1 des Buches befasst sich mit Programmierrichtlinien für *C*- und *C++*-Projekte und gibt Hinweise zur Planung und Organisation. Behandelt werden Themen wie Codierstil, Organisation von Projektverzeichnissen und Dateien, Struktur von Include- und Quelltextdateien sowie bewährte Methoden der Programmierung.

In **Kapitel 2** wird das Dokumentationswerkzeug *Doxygen* vorgestellt. *Doxygen* wertet speziell formatierte Kommentare in Quelltextdateien aus

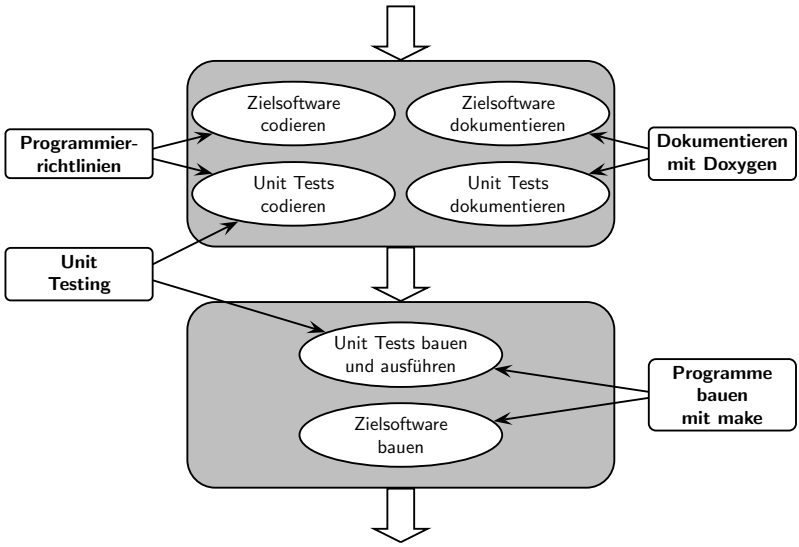

Abbildung 1
Adressierte Anwendungsgebiete

und kann daraus eine Dokumentation in verschiedenen Ausgabeformaten erstellen. Der Funktionsumfang und die vielfältigen Einstellmöglichkeiten von *Doxygen* werden mit zahlreichen Beispielen und Abbildungen eingehend beschrieben. Thema ist auch die Integration des Tools in die Entwicklungsumgebungen Eclipse und Microsoft Visual Studio.

In **Kapitel 3** geht es um das Build-Werkzeug *make* zum Bauen von Programmen. Nach einem einführenden Beispiel werden die vielfältigen Möglichkeiten dieses Klassikers unter den Build-Werkzeugen ausführlich dargelegt. Der Einsatz von *make* mit Eclipse/CDT und eine kurze Vorstellung anderer Build-Werkzeuge runden das Kapitel ab.

Kapitel 4 widmet sich ausführlich dem Thema Unit Testing. Nach einer allgemeinen Einführung in das Unit Testing und die testgetriebene Entwicklung werden verschiedene Unit Test Frameworks vorgestellt: *CUnit* für das Unit Testing in *C*-Projekten, *CPPUnit* und das Boost Unit Test Framework für Projekte mit der Programmiersprache *C++*. Die Erstellung von Unit Tests wird anhand praktischer Beispiele genauso behandelt wie textbasierte und grafische Testoberflächen und die Möglichkeiten zur Anpassung und Erweiterung der Pakete.

Im **Anhang** schließlich sind anwendungsbezogene Kurzanleitungen zu *make* und *Doxygen* und Kurzreferenzen zu den vorgestellten Unit Test Frameworks zusammengefasst.

Versionen

Die in diesem Buch erwähnten Softwarepakete werden alle mehr oder weniger intensiv weiterentwickelt. Alle Ausführungen beziehen sich auf die zum Zeitpunkt der Erstellung freigegebenen Softwareversionen. Es wurde bewusst darauf verzichtet, neue Merkmale des aktuellen Entwicklungsstandes der jeweiligen Pakete zu erläutern, da keinesfalls gesichert ist, dass diese in der nächsten freigegebenen Version tatsächlich auch verfügbar sind.
Im Einzelnen bezieht sich das Buch auf die folgenden Softwareversionen:

Boost Test Library	1.33.1
CPPUnit	1.12.0
CUnit	2.1.0
Graphviz (*dot*)	2.12
Doxygen	1.5.2
Eclipse	3.2.1
Eclipse CDT	3.1.1
GNU make	3.81
Microsoft Visual Studio	7.1
gcc (MinGW)	3.4.2
MinGW Runtime	3.9
MinGW Utils	0.3

Bildschirmfotos im Buch wurden zum Teil noch mit Vorgängerversionen erstellt. Eventuelle Abweichungen in der Darstellung sind jedoch minimal.

Notation und Konventionen

Soweit es der Verständlichkeit und der Lesbarkeit nicht abträglich ist, wird in diesem Buch eine durchgängige Notation benutzt.

Codefragmente und Programmbeispiele sind durchgängig in `Schreibmaschinenschrift` gesetzt. Die Syntax einer Programmiersprache wird nicht besonders hervorgehoben. Die einzige Ausnahme ist Kapitel 2, in dem *Doxygen*-Schlüsselworte in *`geneigter Maschinenschrift`* gesetzt werden.

```
#include <stdio.h>
/** @brief Das klassische Programm */
int main(int argc, char *argv[]) {
        printf("Hello world!\n");
        return 0;
}
```

Soweit Leerzeichen und Tabulatorzeichen von Bedeutung sind, werden diese durch die Zeichen ␣ beziehungsweise → dargestellt.

```
#␣make␣Beispiel
all:␣$(TARGETS)
⟶$(CC)␣$(CFLAGS)␣$<
```

Wenn in Beispielen nur aufgrund der beschränkten Seitenbreite ein Zeilenumbruch erforderlich ist, dann wird dies durch das Zeichen ↪ signalisiert.

```
const char *s = "Eine lange Zeichenkette, die sich hier↪
                 nur aus drucktechnischen Gründen ↪
                 über mehrere Zeilen erstreckt."
```

In der Darstellung von Benutzerinteraktionen werden Benutzereingaben in *geneigter Schreibmaschinenschrift* und Ausgaben in normaler Schreibmaschinenschrift gesetzt. Eine Eingabeaufforderung eines Systems wird durch das Zeichen $ dargestellt.

```
$whoami
root@arthur
```

Abschließend sei darauf hingewiesen, dass im Rahmen des vorliegenden Werkes auf eine geschlechtsspezifisch korrekte Schreibweise der besseren Lesbarkeit wegen verzichtet wurde. Es sind aber unter der gängigen männlichen Schreibweise beziehungsweise unter neutralen Formulierungen immer beide Geschlechter subsumiert.

Webseite zum Buch

Eine Webseite zum Buch gibt es unter http://www.macht-publik.de/ccpp/. Viele Beispiele aus dem Inhalt, eine Linksammlung und weitere Informationen zum Buch sind dort verfügbar.