Dokumentationskonzept

Ein sehr wichtiger Teil der Softwareentwicklung ist die Dokumentation des Produkts. Das Dokumentationskonzept legt Qualitätsstandards fest, die während der Herstellung und nach der Fertigstellung, während der Pflege- und Wartungsphase, den Umgang mit der Software erleichtern, denn alle Beteiligten können schnell und einfach die logischen Zusammenhänge und Strukturen erkennen.

I. Interne Dokumentation
  a) Codekonventionen
  b) Kommentierter Quellcode
  c) Dokumentation mittels `javadoc`

II. Externe Dokumentation
  a) Designbeschreibung
  b) Testdokumentation
  c) Handbuch

I. Interne Dokumentation

a) Codekonventionen

Ein Großteil – etwa 80% – der Entwicklungszeit einer Software werden für Wartung und Pflege benötigt. Selten kümmern sich aber die Originalprogrammierer darum. Deshalb ist es wichtig sich an gewisse Regeln zu halten, um die Lesbarkeit des Codes zu erhöhen und ihn so für die spätere Arbeit daran leicht verständlich zu machen.
Wesentliche Mittel zu Übersicht sind:

- Einrücken untergeordneter Zeilen (gewöhnlich 4 Leerzeichen)
- Mehr als 80 Zeichen pro Zeile vermeiden
- Ausreichend Absätze bzw. Zeilenzwischenräume, um logisch zusammenhängende Abschnitte von anderen zu trennen
- Namen für Klassen, Interfaces, Methoden, Attribute u.ä. möglichst aussagekräftig und „selbsterklärend“ wählen
- Für Klassennamen sollten Substantive, für Methoden Verben und für Interfaces Adjektive gewählt werden
- Klassennamen beginnen mit einem Großbuchstaben, Variablen und Attribute werden kleingeschrieben; besteht ein Name aus mehreren Worten, werden sie durch Großschreibung getrennt; Konstanten werden komplett in Großbuchstaben geschrieben und evtl. mehrere Worte durch Unterstrich getrennt
- Nur eine Zuweisung oder Operation pro Zeile

b) Kommentierter Quellcode

Damit leicht erkennbar ist, was zum Beispiel eine Variablen bezwecken soll oder was in einer Schleife passiert, ist es wichtig bestimmt Bereiche zu kommentieren. Es gibt verschiedene Möglichkeiten den Code zu kommentieren:

Für `java`-Dateien gibt es folgende Arten:

- Block-Comments

```
/*
 *   Kommentar
 */
```

  Sie beschreiben ausführlicher, was im nachfolgenden Block passiert. Sie stehen am Anfang jeder Klasse um einen Überblick über ihre Funktionen zu schaffen.

- Single-Line Comments

```
/* Kommentar */
```

  Sie beschreiben den Inhalt oder die Funktion der nachfolgenden Zeile(n)

- End-Of-Line Comments

```
// Kommentar
```

  Sie werden meistens verwendet, wenn eine (kleine) Anmerkung zu einer Zeile nötig ist, indem sie hinter den Inhalt der Zeile angehangen werden.

Um `jsp`-Dateien zu kommentieren gibt es zwei Möglichkeiten. Beide haben die selbe Aufgabe wie die Single-Line Kommentare bei Java. Aber fügt man in eine `jsp`-Datei einen Kommentar der Form `<!-- Kommentar -->` ein, dann erscheint dieser auch in der Ausgabe der Datei (d.h. zum Beispiel im Code der erzeugten `html`-Datei). Kommentare der Form `<%-- Kommentar --%>` sind nur in der `jsp`-Datei sichtbar.

c) Dokumentation mittels `javadoc`

Die Programmiersprache Java bietet die Möglichkeit eine Dokumentation eines Programms oder Pakets erstellen zu lassen. Mittels eines bestimmten Syntax wird aus dem Code alle wichtigen Informationen und Aufgaben des Programms herausgezogen und daraus automatisch eine API erstellt, die im `html`-Format vorliegt. Sie bietet dann für alle Beteiligten eine Übersicht über alle Produktfunktionen.
Damit die Informationen in die API übernommen werden, muss vor jeder Klasse, Methode, Exception, jedem Interface und Konstruktor ein Blockkommentar folgender Form stehen:

```
/**
 *   Kommentar
 */
```

Zusätzlich können mit Hilfe von sog. Tags spezielle Informationen hinzugefügt werden. Sie stehen auch im Blockkommentar und beginnen jeweils mit einem „@“. Die wichtigsten sind:

`@author`	(gibt den/die Autoren wieder)
`@version`	(gibt die Versionsnummer wieder)
`@param`	(erklärt einzelne Parameter einer Methode)
`@returns`	(erklärt den Rückgabewert einer Methode)
`@throws`	(erklärt Fehlerbehandlung einer Methode)

Für beide Formen der Kommentierung ist es wichtig, dass sie parallel zur Implementierung geschehen. Hinterher ist es viel aufwändiger und wahrscheinlicher, dass es dabei zu Fehlern kommt. Akteure sind die Verantwortlichen für Implementierung, Dokumentation und Qualitätssicherung.

II. Externe Dokumentation

a) Designbeschreibung

Die Design-Beschreibung enthält alle wichtigen Informationen über die Funktionen und die Architektur der Software. Die innere Struktur des Programms wird mit UML-Diagrammen dargestellt. Sie erklärt außerdem welche Anforderungen an die Plattform(en) gestellt werden. Eventuelle neue Beteiligte können sich so schnell mit dem Programm vertraut machen und vor allem bietet es dem Kunden im frühen Entwicklungsstadium eine gute Übersicht. Verantwortlich sind der Projektleiter, der Implemtentierer, der Modellierer und der Dokumentierer.

b) Testdokumentation

Vor dem eigentlichen Implementieren werden Testläufe durchgeführt. Sie dienen dazu Fehlerquellen zu finden und zu beheben und müssen nach jeder Änderung des Systems erneut durchgearbeitet werden. Der Verantwortliche für die Tests muss diese auch dokumentieren, d.h. auflisten, wann welcher Test durchgeführt wurde und ob er erfolgreich war oder nicht. Für Komponententest einzelner Softwareeinheiten soll das JUnit-Framework eingesetzt werden.

c) Handbuch

Das Handbuch wird im pdf-Format zur Verfügung gestellt. Es beschreibt die Bedienung des Programms aus der Sicht des Anwenders. Alle wichtigen Informationen zum Umgang mit dem Produkt werden beschrieben und häufig gestellte Fragen werden schon im Vorfeld beantwortet. Dadurch ist es ein unverzichtbarer Bestandteil der Software. Da es meist die einzige Unterstützung für den Benutzer ist, muss es sorgfältig erstellt und gestaltet werden.