Verantwortlicher für Qualitätssicherung : Christin Baumberg
Verantwortlicher für Dokumentation : Melanie Fricke
Verantwortlicher für Recherche : Stefan Beyer

Dokumentationskonzept

(Wiki -Portlet)

(Version 2.3)

1. Einleitung

Umfassende und problemadäquate Dokumentation ist integraler Bestandteil eines erfolgreichen Softwareentwicklungsprozesses. Sobald mehrere Personen ein Projekt bearbeiten ist das Setzen und Einhalten von Standards bezüglich Codekonventionen wie auch Quellcode-Dokumentation unerlässlich. Während sich die genannten Punkte vor allem auf Entwicklung, Erweiterung und Wartung, also dem programmiertechnischen Aspekt bzw. die interne Dokumentation beziehen, ist zu beachten, dass bereits ab Beginn des Projekts auch mit Konzeption und Erstellung der externen Dokumentation zu beginnen ist. Diese ist vor Allem auf den Endanwender ausgerichtet.
Die Konzeption beider Dokumentationssysteme ist auch deshalb frühestmöglich anzufertigen, weil bei nachträglicher Erarbeitung wichtige Informationen verloren gehen und bestimmte Prozesse nicht mehr nachzuvollziehen sind.
Alle relevanten anfallenden Informationen sind von daher unverzüglich von demjenigen, der sie erzeugt respektive verarbeitet, festzuhalten.
Obwohl diese Arbeitsweise anfänglich Mehraufwand bedeutet erspart auch hier eine sorgfältige Planung längerfristig Aufwand.

2. Vorgehensweise

Erste Grundlagen des Dokumentationskonzeptes ergeben sich aus dem Lastenheft/Pflichtenheft. Zur Fixierung bestimmter Standards wird außerdem dieses Dokument durch die Entwicklungsgruppe erstellt.
Ein weiterer wichtiger Meilenstein ist der Reviewprozess. Da in verschiedenen Phasen des Projekts unterschiedliche Verantwortlichkeiten entstehen, sind erarbeitete Dokumente und Konzepte per Mailingliste grundsätzlich allen Mitgliedern zugänglich zu machen. Erweitert und intensiviert wird dieses arbeitsteilige Vorgehen durch intensive Nutzung unseres PmWikis, da hier auch mehrere Personen in die Entstehung von Artefakten eingebunden werden können.
Neben der Erstellung eines Testkonzepts durch unseren Verantwortlichen für Tests wird vereinbart, dass Testfälle auf der Basis von Geschäftsprozessen erstellt werden. Für deren Konzeption und ausreichende Dokumentation ist ebenfalls der Testverantwortliche zuständig.

3. Interne Dokumentation

Für alle Kommentare und Bezeichner (Attribute, etc.) im Quellcode wird die englische Sprache verwendet.

3.1 javadoc - Dokumentation

Sämtliche entstehenden Klassen, Interfaces, Konstruktoren sowie Methoden die projektrelevanten Status haben (im Normalfalle sollten das sämtliche als public deklarierten sein), müssen mittels javadoc kommentiert werden. Javadoc ist ein Java-internes System, dass aus nach einer expliziten Syntax kommentiertem Quellcode HTML-Seiten mit Beschreibungen generiert. Nach jeder Änderung an vermittels javadoc kommentierten Code-Fragmenten ist durch Anwendung entsprechender Befehle die Dokumentation zu aktualisieren. Alle Kommentare stehen unmittelbar vor der Deklaration der Programmiereinheit.
Zur Formatierung soll hier folgender Standard vereinbart werden:

- Beginn des Kommentars mit: **/****
- kurze Zusammenfassung (Wiedergabe der Nutzung der Klasse oder Methode)
- Beschreibung der Funktionalität
- **@author:** Name des Autors einer Klasse
- **@version:** Nummer der Version einer Klasse
- **@see:** Verweis auf andere Klassen oder Methoden, falls nicht automatisch erzeugt durch Vererbungs- und Variablentypbeziehungen (d. h. muss nicht immer angegeben werden)
- **@param:** Beschreibung der Übergabewerte einer Methode
- **@return:** Beschreibung der Rückgabewerte einer Methode
- **@exception:** Beschreibung aller abzufangenden Ausnahmen der Methode
- Ende des Kommentars mit: ***/**

Beispiel:

```
/**
 * short discription
 * detailed discription of the functionality of a class or method
 * @author name (only classes)
 * @version 1.3 (only classes)
 * @see reference
 * @param nameOfParameter1 (only methods)
 * @param nameOfParameter2 (discripe all parameters with a @param  statement)
 * @return (discription of returnvalue; only methods)
 * @exception nameOfException
 */
```

3.2 Quelltextkonvertierung und –kommentierung

Durch Code-Konventionen wird eine gute Lesbarkeit erreicht. Diese ist wichtig für ein schnelles Verständnis. Um für Übersichtlichkeit zu garantieren, wird bei größeren Projekten eine Ordnerstruktur festgelegen. Hinzuweisen ist hier insbesondere auf die JavaCodeConventions (siehe http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html).
Zu einem guten Programmierstil gehört zunächst die Einhaltung der richtigen Formatierung. Auf eine korrekte Klammerung ist zu achten. Geschweifte Klammern werden immer in die entsprechende Zeile, das heißt zum Beispiel hinter die einen Klassen- oder Methodennamen, geschrieben. Die folgenden Zeilen innerhalb dieser Klammer werden eingerückt.

Innerhalb einer Klasse ist folgende Reihenfolge einzuhalten, wobei jede public-Klasse in eine eigene Datei zu schreiben ist:

- private Attribute
- Konstruktoren
- public Getter- und Setter-Methoden
- public-Methoden
- protected- und standard-Methoden
- private-Methoden

Auch bei der Deklaration der Bezeichner ist einiges zu beachten. Generell dürfen keine Sonderzeichen verwendet werden. Ein Bezeichner kann aus den Buchstaben von a bis z und wahlweise zusätzlich aus Ziffern von 0 bis 9 bestehen. Am Anfang muss ein Buchstabe stehen.
Bei der Nutzung von Variablen ist darauf zu achten, dass diese durch problemadäquate Datentypen deklariert werden. Attribute sind durch kleine Buchstaben gekennzeichnet. Namen sollten kurz, aber aussagekräftig sein. Zum leichteren Verständnis ist auf einzelne Buchstaben zu verzichten. Wobei hier die Deklaration eines Iterators im Schleifenkopf einer For-Schleife eine Ausnahme darstellt. Die Sichtbarkeit von Attributen ist immer privat. Eine Veränderung von Attributen ist nur durch Getter- und Setter-Methoden möglich.
Mit Ausnahme der Sichtbarkeit gelten für Methodennamen die gleichen Regeln wie für Attributnamen. Klassennamen werden wie Methodennamen gebildet, allerdings beginnen Klassennamen stets mit einem Großbuchstaben.
Damit der Quellcode gut leserlich ist, sind Kommentare notwendig. Diese sind jedoch nur eine Ergänzung zu javadoc (siehe 2.1 javadoc – Dokumentation).
Der Blockkommentar steht an erster Stelle in der Datei, noch vor allen anderen Anweisungen. Er enthält zum Beispiel Informationen darüber welche Methoden noch überarbeitet beziehungsweise welche Fehler noch behoben werden müssen. Im Folgenden ein Beispiel hierfür:

```
/*
 * calculatedArea(int x); does not pass the while-loop
 * writeLenghts(); returns wrong result
 */
```

Einzeilige Kommentare werden genutzt, um schwierige Abschnitte im Code für andere verständlich zu machen. Diese werden vor dem entsprechenden Abschnitt eingefügt. Er kann auch dazu genutzt werden, die Funktion von Attributen kurz näher zu erläutern. Folgend ein Beispiel:

```
int a = 0; //length of base area of a pyramid
int h = 0; //height of pyramid
...
//calculating of volume of a regular hexagonal pyramid
V = ( a * a ) / 2 * h * Math.sqrt(3);
```

3.3 Dokumentation verschiedener Testfälle

Alle Ergebnisse der Komponenten-, Integrations-, System- und Abnahmetests werden dokumentiert. Alle Klassen und Methoden werden getestet. Auch in den Testklassen werden die unter „2.2 Quelltextkonvertierung und –kommentierung“ festgelegten Standards eingehalten.
Das vorliegende Testkonzept wird in die laufenden Projektarbeiten einbezogen. Der Kunde kann alle Aspekte im Bezug auf Tests im Testkonzept nachlesen.

4. Externe Dokumentation – Benutzerhandbuch und Online Hilfesystem

Um dem Endbenutzer bzw. Anwender die Arbeit mit dem Software-Produkt zu erleichtern, ist eine vollständige und fehlerfrei Dokumentation sehr wichtig. Sie muss leicht verständlich sein, damit auch ein Nutzer ohne Hintergrundwissen das Produkt problemlos verwenden kann.
Da im Portletstandard JSR168 bereits eine Hilfefunktion durch den Help-Portletmode definiert ist, erweist sich die Nutzung dieser Funktion als sinnvoll. Daher wird neben einem Benutzerhandbuch auch ein Online Hilfesystem angeboten.
Beides steht dem Benutzer zur Verfügung und soll alle Funktionalitäten verständlich und umfassend beschreiben. Zusätzlich ist im Benutzerhandbuch ein Produktüberblick und detaillierte Informationen über Systemvoraussetzungen, Installation und Handhabung des Produktes beschrieben.

5. Qualitätsstandards

Produktqualität	sehr gut	gut	normal	nicht relevant
Funktionalität			x	
Zuverlässigkeit	x			
Benutzbarkeit		x		
Effizienz			x	
Änderbarkeit		x		
Übertragbarkeit	x			

6. Verantwortlichkeiten

Zweifelsohne sollte der Verantwortliche für Dokumentation und Qualitätssicherung in diesem Bereich federführend sein, auch wenn Beiträge natürlich von allen Entwicklern geleistet werden müssen. Die Einhaltung der Standards zu überwachen ist allerdings dem vorgenannten Gruppenmitglied anzurechnen. Ebenfalls ist er alleine für die Erstellung externer Dokumentationen zuständig. Während javadoc und Quelltextdokumentation innerhalb des Projektes hauptsächlich von Implementierern erstellt werden, ist die Kommentierung und Dokumentation von Testklassen Sache des Testers.