Dokumentationskonzept

Das Dokumentationskonzept dient der Qualitätssicherung, indem es einheitliche Richtlinien für die Erstellung des Quelltextes, der Kommentare und aller dazugehörigen Dokumentationen festlegt. Die Dokumentation muss dabei analog zur Softwareentwicklung erfolgen, da so alle relevanten Fakten verfügbar sind. Sie soll es einerseits dem Auftraggeber und den Benutzern ermöglichen, mit dem Produkt zu arbeiten, andererseits soll sie andere Programmierer in die Lage versetzen, sich in das Programm einzuarbeiten, es weiterzuentwickeln oder zu warten.

Vorgehensweise
Erste Grundlagen des Dokumentationskonzeptes ergeben sich aus dem Lastenheft/Pflichtenheft. Zur Fixierung bestimmter Standards wird außerdem dieses Dokument durch die Entwicklungsgruppe erstellt. Da in verschiedenen Phasen des Projekts unterschiedliche Verantwortlichkeiten entstehen, sind erarbeitete Dokumente und Konzepte per Mailingliste grundsätzlich allen Mitgliedern zugänglich zu machen. 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

1. Interne Dokumentation

Die interne Dokumentation erfolgt innerhalb des Programms und liefert eine Kurzbeschreibung über die Funktionalitäten. Sie dient vor allem dem Entwicklungsprozess und dem Testen.

1.1 Quellcode:

Die Quelltexte sollen der Java-Code-Convention sowie Helmut Balzert's Richtlinien für guten Code entsprechen. Ziel ist es, die Quelltexte übersichtlich zu gestalten, um eine schnelle Einarbeitung zu ermöglichen. Besonders wichtig ist dabei

- die Nutzung selbsterklärender Bezeichner
- Umsetzung der Richtlinien für Bezeichner
- die Einhaltung der Formatierungsvorgaben für Code
- Beachtung des einheitlichen Aufbaus der Klassen

1.2 Quellcodekommentare:

Der Programmierer fügt in seinen Quellcode während der Implementierung Kommentare für die Klassen, Variablen und Methoden entsprechend der Javadoc-Syntax ein, in denen die Funktionen, Werte, Ein- und Ausgaben sowie die Entwicklung dokumentiert werden.

1.3 Javadoc- Dokumentation:

Javadoc erstellt aus speziellen Quelltextkommentaren automatisch eine HTML-Dokumentation die in jedem Browser darstellbar ist und den Aufbau von Paketen und Klassen zeigt. Entsprechende Kommentare müssen unmittelbar vor der Deklaration der Klasse oder Methode stehen. Spezielle Tags werden mit einem @ gekennzeichnet (z.B. @param, @version, @author, @return…)

1.4 Dokumentation der Testfälle:

Eine Beschreibung aller Tests der Komponenten und des gesamten Systems mit Angabe des Zwecks, Ablaufs, und der Ergebnisse. Dafür wird ein spezielles Testkonzept entwickelt und eingesetzt.

2. Externe Dokumentation

Die Externe Dokumentation enthält alle Dokumente, die dem Benutzer zugänglich gemacht werden. Darin können auch Elemente der internen Dokumentation enthalten sein. Sie soll es vor allem den Benutzern ermöglichen, mit dem Produkt effizient und fehlerfrei zu arbeiten.

2.1 Kundendokumentation:

Darin sind alle wichtigen Dokumente enthalten, die während des Projektes erstellt werden, beispielsweise Lastenheft, Glossar, Pflichtenheft und ähnliches. Sie wird dem Kunden währen der Arbeit am Projekt übergeben, um ihn über die Entwicklung des Projekts zu informieren. Sie dient auch zur Kommunikation mit dem Kunden selbst, einen Konsens über Anforderungen und Leistungen des späteren Produktes zu schaffen.

3. Verantwortlichkeit

Die interne, quelltextnahe Dokumentation liegt in der Verantwortung der jeweiligen Programmierer, ebenso die javadocs. Der Verantwortliche für Dokumentation muss den Quellcode in Bezug auf die Einhaltung der festgelegten Richtlinien für das Kommentieren überprüfen. Die Erstellung der externen Dokumentation liegt in der Verantwortung des Verantwortlichen für Dokumentation und Qualitätssicherung. Verantwortlich für den Quellcode einschließlich der Kommentierung und Javadoc ist der Verantwortliche für Implementierung. Verantwortlich für die Dokumentation der Testfälle ist der Verantwortliche für Tests.

4.Richtlinien für guten Code entsprechend java.sun und Helmut Balzert

-Bezeichner:
- natürlichsprachlich, problemnah, verständliche Abkürzungen
- beginnt mit einem Buchstaben,enthält kein _ , ß und kein Leerzeichen
- es ist Groß- und Kleinschreibung zu verwenden, aber Bezeichner sollten sich nicht nur In der Groß-Kleinschreibung unterscheiden
- entweder deutsche oder englische Sprache verwendet. Ausnahme: push, get, set, ...
- bei mehreren Worten wird jedes Wort Großgeschrieben.
- Klassennamen -beginnen mit Großbuchstaben,
  -bestehen aus Substantiv(Singular) ggf mit Adjektiv,
  (+GUI falls GUI-Klasse)
  - wenn Klassen zu einem bestimmten Paket gehören, so ist der Paketname Teil des Klassennamens (Paketname.Klassenname)
  zB. Paketname: Plugin.diagramm.part
  ->Klassenname:PluginCreationWizard

-Objektnamen:
- beginnen mit Kleinbuchstaben , enden mit Klassennamen
-bei anonymen Objekten mit ein, erster... z.B. einRechteck
-GUI-Interaktionselemente: attributnameFachkonzeptklasse z.B nameButton

-Attributname:
-im Englisch beginnt es mit Kleinbuchstaben,im Deutsch mit Großbuchstaben
-sind detailliert zu beschreiben z.B ZeilenZähler

-Operationsnamen:
-beginnen immer mit Kleinbuchstaben, verbSubstantiv z.B. zeigeFigur
-bei Attribut-leseoperation: getAttributname, bei schreibop.: setAttributname
-bei boolschen anfragen: isAttributname

Deklaration:

- nur 1 Deklaration pro Zeile (in der Regel nicht: int wert1, int wert2;)
- nie verschiedene Typen in 1 Zeile deklarieren (int wert, array a1)
- Variablen nur am Blockanfang deklarieren (außer in for-Anweisungen)
- Variablen nur dann public machen, wenn es unbedingt nötig ist
- vermeide das überschreiben von Variablennamen (global vs. lokal)
- Variablen möglichst bei Deklaration gleich initialisieren
- Klassendeklaration: öffnende Klammer auf der Zeile der Deklaration, schließende in eigener Zeile auf Höhe des Anfangs der Deklaration class Sample { ...
  }
- Methodendeklaration : kein Leerzeichen zw Name und Parameterliste,Leerzeile vor jeder neuen Methode methodenName(param,...){...