Lernportfolio Informatik IV

Mirko Messer

Aktuallisiert:6. Januar 2012

1 Software-Dokumentation

1.1 Auftrag 1.

Da im laufe der Zeit, bei einer Software immer wieder neue Aspekte, Feautures, Sonderheiten usw. dazu kommen, ist es wichtig das die Software/Code sauber und Strukturiert Dokumentiert wird. Verlässt ein Wichtiger Know-How träger ein Software-Projekt und dessen wissen ist nirgends Dokumentiert, heisst dies für die Person welche sich nun mit dieser Software beschäftigen muss, unter Umständen eine sehr mühsame und zeitaufwändige Einarbeitungszeit. Auch komplexer Code kann mithilfe einer guten Dokumentation schnell verstanden und verwendet werden. Die Praxis zeigt zudem, dass es sich mittel- und langfristig auszahlt eine aktuelle Dokumentation zu besitzen. Wenn man Anfängt die Dokumentation als Werkzeug der Software Entwicklung zu sehen, und es wie der Code fortlaufend mit wächst. Ist auch die Nachbearbeitung der Dokumentation auf ein minimales reduziert.

1.2 Wichtige Punkte der Software Dokumentation

- Dokumentversionen sind wichtig, wenn in Etappen an der Dokumentation gearbeitet wird.
- Inhaltsverzeichniss sollte maximal 3 Stufen haben
- Beim Inhalt gilt Qualität vor Quantität, der Leser soll verstehen wie die Software funktioniert.
- keine Wiederholungen sondern Verweise verwenden.
- Quellend immer sauber und komplett Angeben (Bei Websites auch Datum angeben).
- Rechtschreibung, ev. von Dritten Korrektur lesen lassen.

1.3 Auftrag 2.

Da wir in unserem neusten Projekt nach dem agilen[1] Vorgehensmodell Scrum vorgehen. Möchte ich euch einmal die Wichtigsten Punkte für das kommende Software Projekt mitteilen. Wir beginnen mit einer Analyse in dem wir festlegen was in unserer Software umgesetzt werden soll. Dazu verwenden wir UCDD[2]-Methode zusammen mit dem Kunden. Ebenfalls wird ein Grobes Klassendiagramm erstellt um den Überblick der Daten und deren Zusammenhängen zu bekommen. Weiter erstellen wir ein Lastenheft welches von unserem Kunden unterzeichnet wird, in diesem ist der Funktionsumfang komplett abgebildet. In der Design-Phase werden wir einen ersten Entwurf des GUI's erstellen, dies wird nur konzeptionell erfolgen und noch nicht dem definitiven Client entsprechen.

Umsetzung nach Scrum. In einer ersten Phase werden alle Funktionen welche in der Software verwendet werden in den sogenannten "Product Backlog" abgelegt. In einem weiteren schritt werden die Funktionalitäten gruppiert und zu sogenannten Sprints realisiert, so ein "Sprint" dauert maximal einen Monat. Um die Realisierung zu unterstützen beinhaltet der "Sprint Backlog" alle Funktionalitäten welche in diesem "Sprint" umgesetzt werden sollen. Nach Ablauf des "Sprints", werden wir gemeinsam den nächsten "Sprint"

[1] Agil: geschäftigt, betreibsam
[2] UCDD: Use Case Driven Design

planen. Hierzu nehmen wir aus dem "Product Backlog" weitere Funktionen welche noch umzusetzen sind. Jeder "Sprint" stellt eine Iteration dar, Die Analyse, Design Implementierung, Test und Dokumentation beinhaltet. Zudem entsteht nach jedem "Sprint" ein ausführbares Produkt mit neuen Funktionen. Jeder kann also sehen wie unsere Software wächst. Täglich werden wir zudem ein maximal 15-minütiges "Daily Scrum" Meeting führen, in welchem jeder über die fertiggestellten Arbeiten, Probleme usw. Informiert wird. Die Erkenntnisse daraus werden wir im "Burndown Chart" (hier wird der Restaufwand des SSprint"Dokumentiert) und im "Impediment Backlog" (Dokumentation der Probleme, Hindernisse) niederschreiben. Nach beendigung eines "Sprints" werden wir jeweils ein Review machen in welchem das Ergebnis des "Sprints" angeschaut wird.

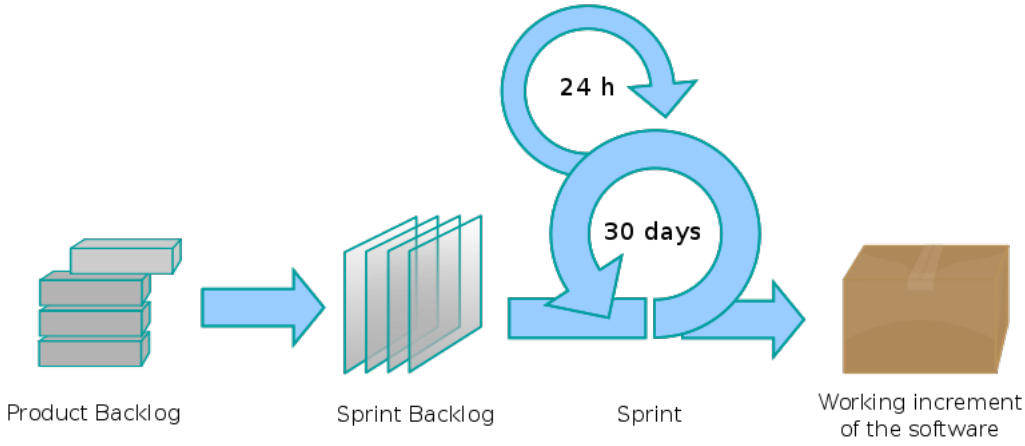

Abbildung 1: Scrum Vorgehensmodell (Quelle: Wikipedia)

2 Java Persistence API