188.925 Wissenschaftliches Arbeiten

Seminararbeit "Who is Who @ TU"
Stefan Biffl - Software Engineering

von:
Christian Kletzander (1125210)
Kögler Alexander (1125544)

Inhaltsverzeichnis

Kurzfassung (Abstract)

Anforderungsdokumente stehen nicht nur am Beginn der Softwareentwicklung, sondern sind auch der Grundbaustein für den Softwareentwickler und Designer. Die Blickwinkel und Verständnisse der Thematik aller an einem Softwareprojekt teilhabenden Stakeholder sind unterschiedlich und die primären Vorstellungen gehen mitunter auseinander. Die Standish Group nennt in der Chaos-Studie als Hauptgründe für Misserfolge in Softwareprojekten unvollständige oder unklare Anforderungen, mangelnde Einbindung der Stakeholder und letztendlich übermäßige Anforderungsänderungen[1]. An dieser Stelle setzt die Software-Qualitätssicherung ein. Ziel ist es, Missverständnisse zu erkennen und eine für die beteiligten Akteure zufriedenstellende Lösung zu finden. Nach dem Prozess der Anforderungsanalyse steht eine Softwareanforderungsspezifikation zur Verfügung die einen „Kompromiss aller beteiligten Stakeholder im Hinblick auf das zu erstellende Produkt aus der aktuellen Sicht aller Beteiligten“[2] darstellt. In dieser Seminararbeit werden Methoden und Strategien thematisiert die helfen den Übergang zwischen Anforderungsanalyse und Softwareimplementierung reibungslos zu gestalten um das Auftreten von Fehlern zu minimieren und die Qualität des Projekts zu steigern.

Quellen (Abstract)

[1] http://de.wikipedia.org/wiki/Chaos-Studie

[2] Best Practice Software Engineering

1 Einführung in Software Engineering

Als Projekt bezeichnet man einen Gegenstand, der mit einer gewissen Anforderung und begrenzten Ressourcen zu einem möglichst qualitativen Ziel führen soll. Dieses Ziel durchstreift alle Bereiche des täglichen Lebens beginnend beim Bau eines Hauses bis zum Waschen der täglichen Wäsche. Näher betrachtet können Projekte im Großen sowie im Kleinen stattfinden und auch höchst komplex werden. Am Beispiel der Wäsche gibt es verschiedenste Faktoren die Einfluss darauf nehmen, ob ein Waschgang erfolgreich wird. Hierzu zählen zum Beispiel die Dosierung des Waschmittels, die richtige Temperaturwahl, die perfekte Drehzahl für ein optimales Ergebnis, den Einsatz von Weichspüler usw. Wir sehen also, dass die kleinsten Projekte oft schon bei der Anforderungsanalyse sehr komplex werden können.

Ein wesentlicher Unterschied zwischen Wäsche waschen und Software Projekten liegt darin, dass in den meisten Fällen mehrere Personen am letztgenannten Projekt beteiligt sind und es einen wesentlichen Unterschied gibt, ob man alleine oder im Team arbeitet. Aufgrund dessen benötigen wir Software Engineering. Es ist notwendig von der Anforderung bis zum Ziel diese Personen zu kontrollieren. Software Engineering befasst sich mit der Entwicklung der Software, der Organisation des Teams, der Modellierung und Abstraktion der Umwelt mittels geeigneter (Daten-)Strukturen, sowie der Inbetriebnahme und letztendlich der fortwährenden Wartung des Softwaresystems.

Todo:
Erläuterung der einzelnen Teilbereiche des Software Engineering, Vergleich zwischen Wäsche waschen und Programmierprojekten durchziehen, Wäsche waschen Showstopper - Wasser vergessen aufzudrehen! - Vergleich zu Programmierprojekten finden, Komplexität aufzeigen bezüglich dessen, dass viele Leute mitmischen, Anforderungen herausstreichen, verschiedenste Implementierungsprozesse kurz erläutern, Dokumentation erwähnen, Ungeahnte Einflüsse (Mitarbeiterfluktuation), Motivationsfindung der Mitarbeiter, ...

2 Software Inspection (2 1/2 Seiten)

Mit dem Fortschritt der Mikroprozessortechnologie wuchs auch der Umfang des Programmcodes eines Softwareprojekts. Umfassten Programme mit Beginn der Digitalisierung einige Lochkarten, so sind mittlerweile Softwareprojekte mit mehreren Millionen Codezeilen (quote needed) kein Ausnahmefall. (http://spectrum.ieee.org/computing/software/why-software-fails)

Selbst in besonders sorgfälltig getesteten kritischen Systemen ist mit Fehlern in einem Prozent des Codeumfangs zu rechnen. Trotz umfangreichen Audits und Tests die zu Kosten von 1000$ je Codezeile führten, konnte nach dem Challenger Unglück am Programmcode des NASA Spaceshuttle ein Fehlerumfang von 0.1 Promille nachgewiesen werden. (http://history.nasa.gov/sts1/pages/computer.html)

Es ist mittlerweile unabdingbar auf computergestützte Fehlererkennung oder

Softwaretests zu setzen. Könnten aber Computer den Menschen bei der Fehlererkennung vollständig ersetzen, so wäre die Menschheit in der Lage garantiert fehlerfreie Programme zu erstellen. Als klassisches Kapitel der Informatik konnte Alan Turing mit dem "Halteproblem" nachweisen, dass es nicht möglich ist eine Maschine zu entwickeln, die die endliche Ausführung von jeglichem Programmcode beweisen kann. Somit spielt der Mensch als Gegenpol zur künstlichen Intelligenz auch in Zukunft eine wichtige Rolle, bei der fehlerfreien Generierung von Programmcode.

Todo:
Code Review, Coding Richtlinien, Kompilereinstellungen/warnungen, merkwürdiges TIL Zeug und Sat oder so??, Projektplanungsmodelle (V-Model, Wasserfall, Agile Entwicklung) ganz kurz erwähnt und auf PRaxisuntauglichkeit von Wasserfall hinweisen), (http://smartbear.com/SmartBear/media/pdfs/best-kept-secrets-of-peer-code-review.pdf) das einarbeiten und vorher durchackern

3 Ausgewählte Beiträge

3.1 Beitrag Eins (2 1/2 Seiten)

3.2 Beitrag Zwei (2 1/2 Seiten)

3.3 Beitrag Drei (2 1/2 Seiten)

3.4 Beitrag Vier (2 1/2 Seiten)

4 Zusammenfassung (ca. 1/2 Seite)

5 Literaturverzeichnis (ca. 1/2 - 1 Seite)