Softwaretesten

Exercise 2

Anselm Brachmann Marcus Janz René Perschon Martin Schulze

August 24, 2012

Aufgabe 1: Qualitätsicherung

Folgende Gründe tragen dazu bei, dass Qualitätssicherung erschwert wird:

Testen Tester kennen möglicherweise nur die zu testenden Interfaces, wissen aber nichts über interne Realisierung. Somit verallgemeinern diese ggf. falsch und haben keine Chance, bestimmte Fehler der Software zu finden.

Reviews Heutige Software setzt zum Teil immer noch uralte Module ein, welche einfach in neue Produkte integriert werden. Diese durchzusehen wirft Probleme auf, da auch Sprachen und Compiler sowie alle Werkzeuge weiterentwickelt wurden und nicht immer abwärtkompatibel sein müssen.

Kommunikation Große Projekte, welche von vielen Menschen, Gruppen oder Firmen gemeinsam realsisiert werden, erfordern dass Missverständnisse beispielsweise bei der Auslegung der Spezifikation ausgeschlossen sind. Das ist sehr schwer zu erreichen.

Prozesse Heutige Softwareprojekte, welche von verschiedenen Firmen gleichzeitig realisiert werden, erfordern auch eine Anpassung der Vorgehensmodelle. So lässt das *Wasserfallmodell* schwer mit *Extreme Programming* vereinen und es kommt zu Reibungen, wenn jede Firma auf ein eigenes Modell festgelegt ist.

Standards Heutige Entwicklungen werden oft von der Industrie vorangetrieben und erst später offiziell standardisiert (siehe bsp. Javascript). Software ist also niemals fertig, sondern muss ständig an aktuelle Standarts angepasst werden.

Sicherheit Teilsweise werden sehr alte aber simple Sicherheitslücken (bsp. Formatstrings) erst jetzt bekannt und erfordern, weit verbreitete und viel benutzte Software zu verbessern um diese nicht angreifbar zu machen. Das ist eine Herkulesaufgabe.

Allgemein stellt vor allem die heutige Komplexität von Software sowie die zunehmende Vernetzung der Systeme die Produzenten vor Probleme bezüglich der Qualitätssicherung.

Aufgabe 2: Innere und äußere Qualität

Äußeres Merkmal Benutzbarkeit und inneres Merkmal Übertragbarkeit

Innere Qualität einer Software bezieht sich auf den Code, dieser sollte beispielsweise leicht änderbar oder anpassbar sein und somit ermöglichen, sie Software für verschiedene Systeme verfügbar zu machen. Äußere Qualität ist, was der Benutzer letzten Endes sieht: Er will, dass die Software zuverlässig und leicht zu benutzen ist. Nehmen wir das Beispiel, dass eine Software für MacOS, also ein Unix-System auf ein Windows-System portiert werden soll. Ist der Code gut durchdacht, sieht er für diesen Fall evtl vor, dass an einer zentralen Stelle die Verwendung von Pfaden organisiert ist. Während Windows einen Backslash (’) benutzt, ist auf Unix-Systemen

der normale Slash ('/') in der Verwendung. Im schlechten Fall muss das in jeder Datei des Sourcecodes einzeln ersetzt werden - ein beispiel für mangelhafte innere Qualität. Bezüglich der äußeren Qualität will der Nutzer, dass er sich für die Software bei der Übertragung nicht auf ein fremdes System einstellen muss, sondern die Software sich auf das System einstellt. Ein Beispiel: Bei MacOS ist das Menü immer in der Leiste am oberen Ende des Bildschirmes, Software wie MATLAB jedoch implementieren auch unter MacOS ein eigenes Menu am oberen Ende des Fensters, mit dem Resultat, dass der gewohnte Mac-User dieses jedes mal suchen muss. Das ist ein Beispiel für schlechte äußere Qualität. Beide Beispiele betreffen hier die Portierung von Software.