

**Testkonzept**

****

|  |  |
| --- | --- |
| **Team:** | **Edge** |
| **Projektleiter:** | **Kevin Mangold** |
| **Teammitglieder:** | **Severin Neuner, Florian Schmidt, Nicholas Dickel, Tessa Haschtschek** |
| **Datum:** | **10.04.17** |

**Bearbeitungsverlauf**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Datum** | **Autor** | **Änderung** |
| 1.00 | 6.4.2017 | Severin Neuner | Erstellung des Konzepts |
| 1.01 | 10.04.2017 | Tessa Haschtschek | Formatierung und Titelblatt |

**Einführung**

Da das vollständige Testen einer Software sehr aufwändig ist und nach jeder Änderung erneut ausgeführt werden muss, werden automatisierte Tests verwendet. Am besten eignen sich für diesen Fall Unittests. Dabei wird jede Einheit der Software weitestgehend unabhängig von den anderen Einheiten getestet. Das hat den Vorteil, dass auftretende Fehler im Programmcode einer Einheit zugeordnet und dadurch schneller gefunden und behoben werden können.   
Diese Software basiert auf Java, weshalb es sich anbietet, JUnit zu verwenden. Dies kann in die Entwicklungsumgebung Eclipse als Plugin integriert werden. Jenkins kann zudem den Quelltext automatisch von Github beziehen, mit Hilfe der JUnittests auf Fehler zu überprüfen und den Quelltext nur im fehlerlosen Fall zu übernehmen. Dadurch ist sichergestellt, dass kein fehlerbehaftetes Programm auf dem Server liegt und vom Endbenutzer verwendet wird.

**JUnit**

JUnit ist ein für Java entwickeltes Testframework für automatisierte Tests. Diese Tests liefern nur zwei Ergebnisse, nämlich entweder das Gelingen oder das Misslingen des Tests. Bei jedem Test werden Methoden mit bestimmten Parametern aufgerufen und die Auswirkung mit der erwarteten Auswirkung, beispielsweise einem erwarteten Rückgabewert verglichen. Stimmen diese nicht überein, schlägt der Test fehl. Ein Test schlägt ebenfalls fehl, wenn eine Exception auftritt, die nicht abgefangen wird.  
Jenkins protokolliert diese Tests und kann deshalb weitere Informationen wie die letzte erfolgreiche Ausführung eines Tests anzeigen.

**Testkonzept**

Das Testkonzept wird in die folgenden vier Phasen aufgeteilt: Komponententests, Integrationstests, Systemtests und Abnahmetest.

**1. Komponententest**

Komponententests sollen möglichst unabhängig von der zu testenden Klasse entwickelt werden. In ihnen werden alle Anforderungen an die Klasse überprüft. Die funktionale Korrektheit einer Klasse kann dann nach Implementierung der Klasse nachgewiesen werden, indem diese alle im Komponententest festgelegten Testszenarien erfüllt. Die Komponententests werden wie oben beschrieben mit JUnit und Jenkins realisiert.

**2. Integrationstest**

Nach erfolgreicher Ausführung der Komponententests kann mit den Integrationstests begonnen werden. Ein Integrationstest testet die Schnittstellen der einzelnen Komponenten, um das Zusammenspiel von mehreren Komponenten zu testen.

**3. Systemtest**

Der Systemtest stellt den Test des Gesamtsystems auf Erfüllung der im Pflichtenheft festgelegten Anforderungen zu prüfen. Deshalb muss dieser Test in einer realistischen Umgebung durchgeführt werden, wie sie auch nach Abnahme durch den Kunden verwendet werden wird. Der Test überprüft funktionale, aber auch nicht funktionale Merkmale des Systems. Der funktionale Systemtest konzentriert sich auf funktionale Qualitätsmerkmale wie Vollständigkeit und Korrektheit der Software, während der nichtfunktionale Systemtest die Prüfung nichtfunktionaler Qualitätsmerkmale wie beispielsweise der Performance und Sicherheit des Systems zur Aufgabe hat.

**4. Abnahmetest**

Der Abnahmetest wird vom Kunden am fertigen Endprodukt durchgeführt. Sein Bestehen ist maßgeblich für die Abnahme der Software durch den Kunden.

**Testablauf**

Bei Tests wird eingeplant, dass Fehler gefunden werden. Deshalb muss auch ausreichend Zeit für die Behebung der gefundenen Fehler eingeplant werden. Anzahl und Komplexität der Testfälle müssen ausreichend sein, damit jede Anforderung getestet wird. Im Idealfall wird durch die Tests 100% des Quelltexts abgedeckt. Mit dem Eclipse-Plugin Eclemma kann die erreichte Codeabdeckung überprüft werden.

Für die Fehlerbehebung der gefundenen Fehler ist das Teammitglied zuständig, in dessen Aufgabenbereich der Fehler aufgetreten ist. Kann ein Fehler nicht unmittelbar behoben werden, soll ein Fehlerprotokoll erstellt und mit den anderen Teammitgliedern geteilt, um Lösungen zu entwickeln. Wird ein Fehler gefunden, soll das zuständige Teammitglied umgehend informiert werden, unter welchen Umständen der Fehler aufgetreten ist und welche Auswirkungen er hatte.