**Pflichtenheft zur Erweiterung der Anwendung KBUnit   
(Knowledge Based Unit Testing Application)**

# 1 Visionen und Ziele

**/V10/ (/LV10/)** Der Kunde soll mit der Anwendung KBUnit in die Lage versetzt werden, in seinen Softwareprojekten Behavior Driven Development um Behavior Driven Testing zu erweitern, um die Softwarequalität zu erhöhen.

**/V20/ (/LV20/)** Der Kunde soll mit KBUnit in die Lage versetzt werden, in seinen Softwareprojekten eine hohe Transparenz über den Stand der Softwarequalität zu erhalten.

**/Z10/ (/LZ10/)** Der Kunde soll mit KBUnit in die Lage versetzt werden, die Backlogs aus agilen Projektmanagement-Tools für die Testfallgenerierung zu verwerten.

**/Z20/ (/LZ20/)** Der Kunde soll mit KBUnit in die Lage versetzt werden, jederzeit den Stand zu Modultests einsehen zu können.

# 2 Rahmenbedingungen

**/R10/ (/LR10/)** KBUnit wird bei einem IT-Dienstleister für die Unterstützung der Software- entwicklung eingesetzt.

**/R20/ (/LR20/)** KBUnit wird von allen Mitgliedern eines Softwareentwicklungsteams und auch von Wissensträgern über die zu entwickelnde Anwendung (beispielsweise Product Owner, Softwaretester, Endanwender) benutzt.

**/R30/** Eingesetzte Software auf der Zielmaschine:

**/R31/** Server: Apache TomCat (Webserver) Version9, MySQL-Server Version 5, JDK 8

**/R32/** Client: KBUnit, Emma JaCoCo, Java Version 8, Webbrowser (z.B. Mozilla Firefox), eclipse ide 2018-12, JRE Version 8

**/R40/** Eingesetzte Hardware auf der Zielmaschine inklusive Konfiguration:

**/R41/** Server: PC (Virtuelles Betriebssystem/Linux System, 16 GB freier Festplattenspeicher, 16 GB Arbeitsspeicher), Monitor, Eingabegeräte (Maus und Tastatur), Netzwerkkarte (NIC) mit Anbindung an das Internet

**/R42/** Client: PC (Betriebssystem Windows 10, 12 GB freier Festplattenspeicher, 8 GB Arbeitsspeicher), Monitor, Eingabegeräte (Maus und Tastatur), Netzwerkkarte (NIC) mit Anbindung an das Internet

**/R50/** Software auf dem Entwicklungssystem:

**/R51/** Server: Apache TomCat (Webserver) Version 9, MySQL-Server Version 5, JDK 8

**/R52/** Client: JDK Version 8, Backlog-Vorgabe (z.B. aus ScrumBO Weboberfläche inkl. Datenbank), eclipse ide version 2018-12

**/R60/** Hardware des Entwicklungssystems: siehe Zielmaschine (/R40/)

**/R70/** Orgware: MID Innovator 13

# 3 Kontext und Überblick

**/K10/ (/LK10/)** siehe UML-Anwendungsdiagramm

**/K20/ (/LK20/) Bisher bereits vorhanden:**

KBUnit ist eine Client-Server-Anwendung und besteht bisher aus den Komponenten KBUnit-Entwickler und KBUnit-Wissensträger. KBUnit-Entwickler liegt in dem Projekt der zu entwickelnden Anwendung. KBUnit-Wissensträger ist eine Stand-Alone-Anwendung.

**/K20/ (/LK20/)** **Zu entwickeln:**

KBUnit soll weiterhin eine Schnittstelle zu dem Testabdeckungstool Emma JaCoCo erhalten.

KBUnit soll eine Komponente erhalten mit einer Schnittstelle zum Einlesen von Backlogs aus einem agilen Projektmanagement-Tool.

# 4 Funktionale Anforderungen

**/F10/ (/LF10/) Stand der JUnit Tests ansehen**

Für den Wissensträger soll eine Webanwendung (Teil A) zur Verfügung stehen, welche die aktuell in der KBUnit-Datenbank vorhandenen Testfälle übersichtlich anzeigt. Es muss aus der Anzeige hervorgehen, zu welchen Methoden welche JUnit Tests mit welchem Ergebnis durchlaufen sind.

**/F11/** Erstellung einer Übersicht inklusive Auswahl/Navigation mittels HTML, CSS und JavaScript als Webseite. Aufbau der Übersicht inklusive Auswahl/Navigation siehe Skizze.

**/F12/** In der Übersicht sollen nicht nur diejenigen JUnit Tests sichtbar sein, die mittels KBUnit verarbeitet werden. In der aktuellen Version von KBUnit-Wissensträger sind auch die übrigen vom Entwickler erstellten JUnit Tests in der Datenbank verfügbar und sollen somit in der Übersicht berücksichtigt werden.

**/F20/ (/LF20/) Testabdeckung durch JUnit Tests ansehen**

Für den Wissensträger soll eine Webanwendung (Teil A) zur Verfügung stehen, welche anzeigt, welche Testabdeckung man mit den aktuell vorhandenen JUnit Tests erreicht. Die Testabdeckung wird mit Emma JaCoCo gemessen.

**/F21/** Der Entwickler startet Emma JaCoCo zur Ansteuerung eines Testlaufs in seiner IDE (Eclipse). Falls Java allein nicht ausreicht, damit der Wissensträger auf, die durch Emma JaCoCo gemessene, Testabdeckung zuzugreifen kann, wird ein Report in Form eines HTML-Outputs generiert. Dieser soll in der Webanwendung eingelesen werden.

**/F22/** Dem Wissensträger soll eine Funktionalität zur Verfügung stehen, mit welcher er die anzuzeigenden Klassen aus Emma JaCoCo auswählt. In der aktuell vorhandenen KBUnit-Datenbank sollen alle, im Rahmen der Testläufe generierten Daten gespeichert werden.

**/F23/** Bezüglich der Testabdeckung sollen sämtliche JUnit Tests, die in den Packages „testKBUnit“ und „test“ vorhanden sind, berücksichtigt werden. Zu diesen JUnit Tests soll die mit Emma JaCoCo gemessene Testabdeckung in die Datenbank geschrieben werden.

**/F30/ (/LF30/) Backlogs zur Generierung von JUnit Tests einlesen (entfällt)**

Den Entwicklern soll eine Anwendung (Teil B) zur Verfügung stehen, mit welcher im JSON-Format vorgegebene SprintBacklogs eingelesen werden können. Eine Userstory kann als JSON abgebildet werden.

**/F31/** Die Anwendung soll die bereits vorhandene Anwendung KBUnit-Entwickler in Form einer Java-Komponente mit JavaFX-Oberfläche ergänzen.

**/F32/** Das SprintBacklog soll im Dateiformat JSON und mit einer entsprechenden package structure vorliegen.

**/F33/** Aus dem SprintBacklog können die zur Testfallgenerierung erforderlichen Java-Klassen direkt eingelesen werden.

**/F40/ (/LF40/) JUnit Tests mittels Backlog generieren (entfällt)**

Den Entwicklern steht die Möglichkeit zur Verfügung, aus den eingelesenen Backlogs JUnit Testfälle zu generieren.

**/F41/** Das SprintBacklog soll im Dateiformat JSON und mit einer entsprechenden package structure vorliegen.

**/F42/** Die Testfallgenerierung wird in KBUnit-Entwickler angestoßen.

# 5 Qualitätsanforderungen

**/QB10/ (/LQB10/)** Da nicht davon ausgegangen werden kann, dass die Wissensträger computeraffin sind, müssen die Vorgaben an das Backlog klar sein.

**/QE10/ (/LQE10/)** Die Übersichten zu der Situation zu den Modul-Testfällen müssen aussagekräftig sein.

**/QW10/ (/LQW10/)** KBUnit wird ständig weiterentwickelt.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Produktqualität | sehr gut | gut | normal | nicht relevant |
| Funktionalität |  | X |  |  |
| Zuverlässigkeit |  | X |  |  |
| Benutzbarkeit | X |  |  |  |
| Effizienz | X |  |  |  |
| Wartbarkeit | X |  |  |  |
| Portabilität |  |  | X |  |

# 6 Abnahmekriterien

**/A10/** Gültiges Abnahmeszenario:

**/A11/** Teil A: Emma JaCoCo zur Ansteuerung eines Testlaufs in IDE starten, eine Java-Klasse für den Testlauf auswählen, den Testlauf starten, die gemessene Testabdeckung in Konsole anzeigen, einen Report im HTML-Format für das Testergebnis erstellen, die Webanwendung im Browser öffnen, ggf. den durch Emma JaCoCo generierten HTML-Report einlesen, die Test-Übersicht in der Navigation aufrufen, eine Methode zur Ergebnisanzeige auswählen, einen zur ausgewählten Methode gehörigen JUnit Test auswählen, den Stand des ausgewählten JUnit Tests anzeigen, zur Test-Übersicht zurückkehren, eine Klasse zur Anzeige der Testabdeckung auswählen, die Testabdeckung der gewählten Klasse anzeigen, die Webanwendung schließen

**/A12/** (entfällt)

Teil B: KBUnit-Entwickler starten, einen Button (choose story) zur Auswahl eines SprintBacklogs anklicken, die entsprechende JSON-Datei für das SprintBacklog inkl. Datei-Pfad auswählen, die im SprintBacklog enthaltene Userstory anzeigen, einen weiteren Button (read story) zur Generierung einer neuen Testklasse anklicken, automatische Generierung dieser Testklasse abwarten, das Package dieser neue Testklasse öffnen, die im richtigen Package angelegte Testklasse anzeigen, KBUnit-Entwickler schließen