Malte Blattmann
Benedikt Elßmann
Semen Gaidenbrik
Edwin Knese
Dustin Kröger
René Lindenberg
Markus Reinisch
Jonathan Schlue

Betreuer: Robert Schädlich
Abgabedatum: 02.05.2017

Qualitätssicherungsbericht

Inhaltsverzeichnis

1 Dokumentationskonzept **2**
1.1 Allgemeines . . . 2
1.2 Sprache . . . 2
1.3 Code Dokumentation . . . 2
1.3.1 Changelog . . . 2
1.3.2 Quelltextnahe strukturierte Dokumentation . . . 2
1.4 Ruby on Rails . . . 2

2 Testkonzept **3**
2.1 Allgemeines . . . 3
2.1.1 Komponententest . . . 3
2.1.2 Integrations- und Systemtest . . . 3
2.2 Instandhaltung . . . 3

3 Organisatorisches **4**
3.1 Konsultation . . . 4
3.1.1 Treffen mit Stakeholdern . . . 4
3.1.2 Teamintern . . . 4
3.2 Verständigung und Zusammenarbeit . . . 4
3.3 Git-Repository . . . 4
3.4 Definition of Done . . . 5

1 Dokumentationskonzept

1.1 Allgemeines

Zum Gelingen des Projektes ist eine ausführliche Dokumentation unserer Arbeitsweise notwendig.
Die Dokumentation trägt zum Gesamtverständnis des Projektes beitragen, da durch sie ein besseres Verständnis innerhalb des Teams und beim Auftraggeber entsteht.
Ein von uns gewähltes Mittel der Dokumentation ist die Protokollierung der teaminternen Treffen, sowie der Rücksprachen mit den Auftraggebern.
Diese Dokumentation ist sowohl intern, als auch öffentlich auf unserer Projektseite zu finden:

`http://pcai042.informatik.uni-leipzig.de/~aso17/jekyll/consultation`

1.2 Sprache

Die Dokumentation innerhalb des Quellcodes erfolgt in Englisch und die externe Dokumentation auf Deutsch.

1.3 Code Dokumentation

1.3.1 Changelog

Um einen Überblick über Entwicklungsprozesse zu erhalten wurde ein Changelog angelegt. Das hierfür vereinbarte Schema ist:

<Abgabenr> <Datum> (Name) <Grobe Beschreibung Aktivität>

1.3.2 Quelltextnahe strukturierte Dokumentation

Zur automatischen Dokumentationsgenerierung des Frontend-Quelltextes in Ruby verwenden wir das Tool YARD.

1.4 Ruby on Rails

Um die Qualität der Software sicherzustellen, führten wir einen kurz gehaltenen Code Style Guide ein und orientieren uns an den grundlegenden Prinzipien von Ruby on Rails:

"Don't repeat yourself."

Redundanz wird wenn möglich im Quellcode vermieden.

"Convention over Configuration"

Solange wir uns an übliche Konventionen (beispielsweise gleichartige Bezeichner) in allen Bereichen der Software halten, müssen diese nicht konfiguriert werden.

2 Testkonzept

2.1 Allgemeines

Zur Entwicklung einer Software, die ihre Funktionen erfüllt und ihren Anforderungen gerecht wird, lassen sich unsere Tests in folgende Untergruppen einteilen, die aufeinander aufbauen:

2.1.1 Komponententest

Wir testen die einzelnen funktionalen Bestandteile der Software durch Unittests mit dem Paket *test/unit* für Ruby.

Beim Backend werden wir uns aufgrund des Einbindens ganzer externer Systeme wie Sparqlify auf Integrationstests fokussieren.

2.1.2 Integrations- und Systemtest

Durch ein selbst gefertigtes Skript werden wir die konsistente Datenbereitstellung am SPARQL-Endpunkt durch Validation Tests überprüfen.
Bei erfolgreichem Testverlauf ist die ordnungsgemäße Funktion eines Großteils des Systems sichergestellt.
Ein Scheitern des Systemtests führt zu einer erneuten Überprüfung der einzelnen Komponenten.

2.2 Instandhaltung

Falls es bei der Aktualisierung der RDF-Daten durch einen aktualisierten SQL-Dump zu Komplikationen kommt, erfolgt die Wiederherstellung einer früheren Version.

3 Organisatorisches

3.1 Konsultation

3.1.1 Treffen mit Stakeholdern

Die Treffen (Montag 15:00 Uhr) helfen dem Team den Anforderungen der Betreuer gerecht zu werden und eine gemeinsame Vorstellung der Software zu verwirklichen.
Weiterhin können durch eine Konsulation mit unserem Betreuer offene Fragen und Probleme geklärt werden.
Der Projektleiter vertritt hierbei die Interessen des Teams und steht in engem Kontakt mit den Stakeholdern.

3.1.2 Teamintern

Zur internen Verständigung haben wir uns auf einen weiteren wöchentlichen Termin (Mittwoch 15:00 Uhr) geeinigt.
Hierbei wird mithilfe von Scrum das Vorgehen koordiniert, die weiteren Aufgaben verteilt und die vorherige Arbeit kritisch hinterfragt.
Weiterhin kann bei diesem Treffen Kritik geäußert und konstruktiv im Team diskutiert werden.
Abschließend wird eine Aufwandsschätzung für die kommenden Aufgaben von jedem Teammitglied abgegeben und die vorherigen Aufwandsschätzungen werden teamintern überprüft.

3.2 Verständigung und Zusammenarbeit

Um eine schnelle Kommunikation zu gewährleisten verwenden wir den webbasierten Instant-Messaging-Dienst Slack.
Die Verständigung findet in einem teaminternen und einem offiziellen Channel mit Betreuern statt.
Mit diesem Kommunikationmittel werden individuelle und teamübergreifende Probleme gemeinsam gelöst.

3.3 Git-Repository

Mithilfe von Git werden die gemeinsamen Artefakte versioniert und für das gesamte Team bereitgestellt.
Zur Sicherung der Konsistenz der Artefakte verwirklichen wir das Prinzip der Continuous Integration. Somit wird die Qualität derselbigen sichergestellt.

3.4 Definition of Done

Folgende Kriterien müssen für den Abschluss einer Aufgabe erfüllt sein:

- Bestehen aller oben aufgeführten Testebenen
- Code muss den vereinbarten Standards entsprechen
- Dokumentation muss den vereinbarten Standards entsprechen
- Es wurde ein teaminternes Code-Review durchgeführt
- Artefakt erfüllt die Anforderungen des Auftraggebers