Qualitätssicherungskonzept

Projektleitung: Otto Bittner

15.01.17

GBAW-17

Inhalt

1 Dokumentationskonzept

Um neuen Entwicklern, Kunden und Endbenutzern die Möglichkeit zu geben sich schnell in das Produkt einzuarbeiten ist es wichtig während des Entwicklungsprozess eine Dokumentation für die verschiedenen Aspekte und Schichten des Projekts anzufertigen und zu pflegen. Im Folgenden werden einige Standards festgelegt, denen diese Dokumentation genügen soll.

1.1 Code Dokumentation

Um Entwicklern die nach längerer Zeit zum Projekt zurückkehren oder sich zum ersten mal damit beschäftigen den Einstieg zu erleichtern wird der Quellcode in zwei Formen dokumentiert.
Um das Verständnis von konkreten Konstrukten im Code zu fördern werden Inline Kommentare benutzt. Die Verantwortung hierfür liegt stets beim entsprechenden Autor.
Damit eine einheitliche, strukturierte Möglichkeit zum Nachschlagen von Klassen und Funktionen vorliegt wird außerdem eine Dokumentation mithilfe von JSDoc (JavaScript) bzw. pydoc (Python) erstellt. Auch hier ist es Aufgabe des Autors sich um die Erstellung dieser Dokumente zu kümmern.

Kontrollierende Instantz ist der Qualitätssicherungsbeauftragte. Dieser überprüft wöchentlich das Vorhandensein der entsprechenden Dokumente und weißt bei Mängeln im wöchtenlichen Meeting darauf hin.

1.2 Coding Standard

Um die Qualität und Lesbarkeit des geschriebenen Codes zu erhöhen sollen für die benutzten Sprachen Vorgaben eines Codestyles genutzt werden und automatisch durch Jenkins Tools durchgesetzt werden.
Für JavaScript soll JSLint genutzt werden. Auch wenn hier eine sehr strenge Kontrolle des Codes stattfindet und teilweise eigentlich funktionierender Quellcode als fehlerhaft identifiziert wird ist es eine gute Grundlage um besser lesbaren Code zu erzeugen und für den unerfahrenen JS-Entwickler nicht absehbare Fehler zu verhindern.
Für Python soll der Styleguide PEP 8 genutzt werden.

1.3 Designdokumentation

Um einen Überblick über die implementierte Architektur und Designentscheidungen zu geben wird über den Verlauf des Projekt eine Entwurfbeschreibung geschrieben. Hier wird auch eine aktuelle Übersicht über die Funktionen des Produkts, Entwurfsprinzipien einzelner Module sowie spezieller Begrifflichkeiten gegeben. Es ist Aufgabe des jeweiligen Entwicklers die Entwurfsprinzipien und Informationen die ggf. darüber hinaus gehen zu seinem Modul festzuhalten.

1.4 Prozessdokumentation

1.4.1 Entwicklungsbeiträge

Im genutzten Versionierungssystem git werden Beiträge zum Projekt als Commits zusammengefasst. Diese sind stets möglichst klein zu halten und mit einer aussagekräftigen Commit-Message zu versehen. Dies ist die Aufgabe des jeweiligen Entwicklers. Hierdurch können induzierte Fehler schneller lokalisiert werden.

1.4.2 Testergebnisse & Bugtracking

Jeder Entwickler ist für die Erstellung von Unittests zu seinem Code verantwortlich. Sollten zu irgendeinem Zeitpunkt diese Tests fehlschlagen wird ein Issue dazu angelegt und der Entwickler des betroffenen Codes als Assignee zugewiesen. Die Priorisierung ist Aufgabe des Projektleiters. Fehler die bei manuellen Tests zu Tage treten werden genauso behandelt.

1.5 Dokumentation für Endbenutzer

Um einem Endbenutzer einen Leitfaden zur Benutzung der Software zu geben soll das von GitLab zur Verfügung gestellte Wiki benutzt werden. Jedoch ist es Ziel der Software möglichst selbsterklärend zu sein.

2 Testkonzept

Fehler sind in Softwareprojekten ab einer gewissen Größe unausweichlich. Um mit diesen Fehlern umgehen zu können haben sich über die Zeit einige Konzepte etabliert, mit deren Hilfe Schwachstellen möglichst früh identifiziert und behandelt werden können. Welche diese sind und wie sie in diesem Projekt zum Einsatz kommen sollen wird im Folgenden beschrieben.

2.1 Komponentent Tests

Komponententest (engl. Unittest) sind möglichst klein zu haltende Testeinheiten, welche nicht auf gespeicherte Daten zurückgreifen und eine sehr kurze Laufzeit besitzen sollen(< 2 min). Ein Unittest erstreckt sich über eine funktional geschlossene Codeeinheit, typischerweise einzelne Funktionen. Hierzu wird die zu testende Funktion mit verschiedenen Parametern aufgerufen und mit einem erwarteten, vom Entwickler festgelegten Ergebnis verglichen. Stimmen die Ergebnisse nicht überein schlägt der Test fehl.
Wichtig bei der Entwicklung eines Tests ist dass der Entwickler die zu modelierenden Funktionalität ausreichend verstanden hat und somit passende Aussagen über die Berrechnungsergebnisse treffen kann. Somit bietet es sich an den Unittest stets vor dem eigentlich funktionalen Quellcode zu schreiben. Jeder Entwickler ist stets für die Erstellung von Unittests zu seinen Beiträgen verantwortlich.

Bei der Entwicklung der Unittests für JavaScript soll JSUnit zum Einsatz kommen. JS Unit ist eine Portierung des JUnit Test Framework (Testing Framework für Java) für Javascript.
Für die Tests von Python Code eignet sich die Bibliothek Unittest. Außerdem besitzt Django ein eigenes Testframework welches u.a. auf Unittest aufbaut und die Entwicklung von Integrationtests erleichtert. Es soll an passenden Stellen eingesetzt werden.

2.2 Integration Tests

Integrationtests prüfen hauptsächlich die Interaktion der verschiedenen Komponenten. Werden für die einzelnen Komponenten zuverlässig Unittests geschrieben müssen deutlich weniger Integrations Tests geschrieben werden. Als Faustregel soll hier gelten, dass für alle fünf Unittests ein Integration Tests geschrieben werden muss. Stellt sich bei der Entwicklung herraus, dass ein bestimmtes Modul eine hohe Komplexität aber niedrige Kompliziertheit besitzt kann das Verhältnis zugunsten von mehr Integration Tests verschoben werden. Diese Entscheidung obligt dem jeweiligen Entwickler.

2.3 System Tests

Im Rahmen dieser Tests wird die Software als Ganzes betrachtet. Das Programm wird ausgeführt und vom Entwickler, möglicherweise aber auch externen Dritten, durch Benutzung getestet.
Hier soll nocheinmal das Zusammenspiel aller Komponenten überprüft werden, aber auch die Nutzbarkeit im Sinne der äußerlichen Gestaltung und Menüführung. Diese Tests finden soweit möglich während der Entwicklung einer neuen Komponente statt. Ist dies aufgrund von fehlenden Abhänigkeiten nicht möglich wird dies in den internen, wöchentlichen Treffen gemacht.

3 Organisatorische Festlegungen

3.1 Treffen

Das Team trifft sich intern wöchentlich um aktuelle Tätigkeiten und anstehende Aufgaben zu besprechen. Ziel dieses Treffens ist, dass alle Teammitglieder über den aktuellen Entwicklungsfortschritt informiert sind und schnell auf Probleme reagiert werden kann.
Desweitern gibt es ein weiteres wöchentliches Treffen mit dem Tutor bzw Betreuern um enstandene Fragen zu klären und Feedback zu den jeweiligen Abgaben zu erhalten.

3.2 Kommunikation

Zur Kommunikation nutzt das Team Slack um kurzzeitig auftretende Probleme und Fragen untereinander zu klären.

3.3 Arbeitsergebnisse

Ist eine Aufgabe von einer Person alleine zu erledigen gilt die offizielle Abgabefrist. Der Zuständige muss sich selbst um etwaig nötige Hilfe oder Feedback kümmern (auf andere Teammitglieder zugehen).
Wird eine Aufgabe intern aufgeteilt existiert stets ein Verantwortlicher für die Aufgabe. Diesem müssen die Teilergebnisse zur Koordination und Zusammenführung drei Tage vor der offiziellen Abgabe zugestellt werden. Der Zeitpuffer dient dazu nötige Änderungen, welche erst beim Zusammenführen erkenntlich werden, einarbeiten zu können.