Softwarequalitätssicherungskonzept JDogs

Im Folgenden werden einige Massnahmen beschrieben, wie wir sicherstellen wollen, dass sich unsere Software unseren Vorstellungen entsprechend verhält und so wenig fehlerhaftes Verhalten wie möglich zeigt.

Konstruktives Qualitätsmanagement

- Wir treffen uns zusätzlich zur Übungsstunde jede Woche am Dienstag um 15:30 Uhr, um erledigte sowie anstehende Arbeiten zu besprechen. Wir erscheinen pünktlich zu den vereinbarten Treffen. Bei kurzfristigen Änderungen informieren wir im Gruppenchat auf WhatsApp.
- Wir verwenden – wie verlangt – Javadoc, um unseren Code zu dokumentieren. Dabei möchten wir uns an «Design by Contract» halten.
- Exceptions sollten, wenn möglich, sofort behandelt werden, oder es sollte zumindest vermerkt werden, von wem diese Fehlermeldung behandelt werden sollte.
- Wir verwenden einen WhatsApp-Chat, um uns über den Stand der Dinge auf dem Laufenden zu halten und um Probleme mitzuteilen.
- Wir definieren die Anforderungen, die wir an unser Spiel stellen und halten diese in Checklisten fest, welche schliesslich auch zur Überprüfung der Software dienen. Damit es zu so wenig Missverständnissen wie möglich kommt, ist es zentral, dass die unterschiedlichen Vorstellungen kommuniziert werden und wir uns auf eine Version einigen.

Analytisches Qualitätsmanagement

- Wenn eine Person eine grössere Arbeit erledigt hat, erklärt sie den anderen ihren Code, sodass danach alle mit diesem Code weiterarbeiten können (Structured Walkthrough).
- Zusätzlich führen wir Code Reviews durch, wenn eine Komponente fertig gestellt wurde. Dabei stützen wir uns auf die Checkliste aus der Vorlesung (siehe Anhang).
- Um den Black-Box- und den White-Box-Test durchzuführen, helfen die Listen mit den Anforderungen, die zuvor erstellt wurden. Zudem überlegt sich die zuständige Person entsprechende Äquivalenzklassen und erstellt eine Checkliste, die von den anderen Gruppenmitgliedern ggf. ergänzt werden kann. Der Black-Box- und der White-Box-Test sollte jeweils von einer Person durchgeführt werden, die den zu testenden Code nicht geschrieben hat.
- Bugs werden auf Github mittels des Issue Trackers den anderen mitgeteilt. Dabei sollte der gefundene Fehler so genau wie möglich beschrieben werden (Voraussetzungen, Schritte zur Reproduktion, Fehlermeldungen, etc.).
- Wir wollen so früh wie möglich mit dem Erstellen von JUnit-Tests beginnen.
- Zudem möchten wir Pair Programming (auch zu dritt) bei Gelegenheit einsetzen.
- Die Teammitglieder sollten ihren Code regelmässig einem Quietschentchen o.ä. erklären, um selbst auf allfällige Fehler aufmerksam zu werden.

Messungen

Wir wollen folgende Messungen durchführen:

- Anzahl Logging-Statements mit Logger Apache Log4j 2
- Code Coverage für Tests mit Jacoco
- Folgende 3 Metrics (min, max, average) mit Metrics Reloaded
    - Lines of Code pro Klasse
    - Lines of Javadoc / Lines of Code pro Klasse
    - Lines of Code pro Methode

Anhang

Checkliste Code Review

1. Verstehe ich diesen Code?
2. Hält sich der Code an den vereinbarten Programmierstil?
3. Würde ich diesen Code an dieser Stelle im Projekt erwarten?
4. Hat es bereits Code mit ähnlicher Funktionalität an anderer Stelle?
5. Lässt sich die Lesbarkeit dieses Codes erhöhen?
6. Könne ich diesen Code warten und Änderungen vornehmen?
7. Ist der Code ausreichend dokumentiert?