Universität Basel FS 22 25.März 2022
Programmierprojekt: Night Train to Budapest
Alexandr Sazonov, Jonas Biedermann, Sebastian Lenzlinger, Seraina Schöb

Software-Qualitätssicherungs-Konzept: Ein Entwurf

Was ist das Ziel des QA-Konzeptes:

Um die Qualität des Endproduktes sicherzustellen, soll der Entwicklungsprozess des Spiels durch ein Qualitätssicherungskonzept begleitet werden. Dieses Konzept wird sich dabei auf das Coden konzentrieren in dem es folgendes sicherstellt:

(i) Der Code ist stilistisch einheitlich.
(ii) Der Code lässt sich testen.
(iii) Der Code ist (möglichst) einfach verständlich.
(iv) Der Code tut was er tun soll.

Welche Massnahmen werden ergriffen um diese Ziele zu erreichen?

Um (i) umzusetzen wird für das Projekt der *Google styleguide* verwendet. Dafür sollen alle Beteiligten den *Google styleguide* in ihre Entwicklungsumgebung importieren und sich während der gesamten Entwicklungsphase daran halten.

Um (ii) und (iii) sicherzustellen müssen mehrere Massnahmen ergriffen werden. Dabei werden die meisten Massnahmen sowohl (ii) als auch (iii) befördern, da ein einfach verständlicher Code offensichtlicher Weise zumeist die Testung vereinfacht. Die Massnahmen lauten wie folgt:

- Regelmässige Messungen verschiedener Code Metriken (wöchentlich)
  Dabei sollen vor Allem die Metriken *Lines per Code per Method/Class*, *(Cyclomatic) Complexity* und *Dependency* untersucht werden. Diese quantifizieren die Lesbarkeit und Testbarkeit eines Codes. So ist ein hohes *Lines per Code per Method* – Ergebnis ein Indikator für ein zu niedriges Abstraktionslevel, was die Lesbarkeit mindert oder es lässt sich aus der *(Cyclomatic) Complexity* ablesen wie viele Test für die Prüfung notwendig sind.
- Die konsistente Verwendung eines Loggers (Apache Log4J)[1]
- Regelmässige Code Reviews
  Hierbei sollen drei Techniken Verwendung finden
  (a) Selbstprüfung mittels gemeinsamer Checkliste, immer sofort nach der eigenen Codingsession
  (b) Informale Reviews in Sitzungstechnik (2 mal wöchentlich)
  (c) Buddy – System für sofortige gegenseitige Rückmeldung

Schlussendlich soll (iv) mittels *Unit-Testing* (JUnit) und *Code Coverage* der Tests (JaCoCo) sichergestellt werden. *Unit-Tests* sollen dabei fortlaufend wo sinnvoll eingeführt werden und sogleich auf *Code Coverage* geprüft werden. Ebenso soll es gegen Ende des Projekts eine längere Testphase geben, in der Tests erweitert und neue erstellt werden sollen um alle relevanten *Edge-Cases* zu testen.

[1] Unsere Definition der Logger-Levels und ihrer Anwendung findet sich im Appendix

Appendix[2]

Group-Conventions Log4J Levels meaning:

FATAL	Is fatal fort the Application (i.e. crashes), needs to be addressed **right now.**
ERROR	Harmful but not fatal, needs to be addressed ASAP, but not everything needs to be stopped right away.
WARN	Needs to get looked at at some point but is not the highest priority.
INFO	What is the Code doing here (i.e. give a return value)
DEBUG	Where ever you would put a *System.out.println()* statements to test your code, instead log with debug.
TRACE	No assigned use so far.

Checkliste Code Review:

- Kann ich diesen Code verstehen?
- Hält sich der Code an den *Google styleguide*?
- Ist alles nötige in der *Documentation* vermerkt?
- Ist dieser Code sinnvoll platziert?
- Ist dieser Code redundant?
- Wie lässt sich dieser Code testen?
- Wie kann dieser Code noch verständlicher Gemacht werden?
- …

[2] Es wird hier mit Foliensatz 6 *Software-Qualitätssicherung* Folie 65 davon ausgegangen, dass der Appendix mit Checkliste nicht zum Seitenlimit zählt.