1. Übungsblatt zu Software Qualität

Michel Meyer, Manuel Schwarz

25. Oktober 2012

Aufgabe 1.1

(a)

- **Portabilität**: Plattformunabhängigkeit durch JAVA gegeben.
- **Vollständigkeit**: Während der Programmierung wurde sichergestellt, dass alle in der Aufgabe genannten Punkte implementiert wurden.
- **Benutzbarkeit**: Intuitive, traditionelle und schlichte GUI, die die Benutzung schnell und einfach hält.

(b)

- **Sicherheit**: Es könnten Dialog-Abfragen implementiert werden, sodass nicht jede beliebige Datei (versehentlich) überschrieben werden kann.
- **Wartbarkeit**: Javadoc erhöht die Wartbarkeit sowohl für den/die ursprünglichen Programmierer als auch für weitere Leute, die sich in den Code einarbeiten müssen.
- **Benutzbarkeit/Robustheit**: Hinsichtlich eines Text-Editors sollten verschiedene Text-Kodierungen unterstützt werden.
- **Effizienz**: Steigerung der Effizienz durch Optimierung der Methoden einzelner Aktionen (laden, speichern, usw.).

(c)

- **Syntaxfehler:** Das Programm ließ sich nicht kompilieren, weil die JAVA-Syntax nicht eingehalten wurde.
- **Semantischer Fehler**: Diverse `NullPointerException`s, wenn beim `JFileChooser` auf „Abbrechen“ geklickt wurde, wodurch `null` an verschiedene Methoden weitergeleitet wurde.
- **Semantischer Fehler**: Es wurde beim Beenden/Laden/Neu nicht immer darauf hingewiesen, dass das aktuelle Dokument geändert wurde, weil die Listener nicht korrekt konfiguriert wurden.

- **Optimierungsfehler**: Beim Versuch eine Methode kürzer und schneller (genauer: if-Bedingungen zusammenzufassen) zu machen, machte die Methode in den meisten Fällen gar nichts mehr.

(d)

Die Fehlerklassifikation dient zum einen einer Prioritätseinteilung. So sind beispielsweise Syntaxfehler eher zu beheben als semantische, weil bei fehlerhafter Syntax in der Regel gar nichts funktioniert (Beispiel in JAVA: Der Code lässt sich nicht kompilieren).
Zum anderen dient die Klassifikation der Gruppenzuweisung. So können Optimierungsfehler von Teams behoben werden, die sich in der Optimierung auskennen, während Spezifikationsfehler oder Parallelitätsfehler an entsprechende andere Teams weitergeleitet werden können. Des Weiteren können die QS-Prozesse bei folgenden Produkten verbessert werden, da man mehr Kontrolle an häufigen Fehlerquellen erwirken kann.

(e)

Mit Hilfe der expliziten Qualitätssicherungsmaßnahmen werden die vom Programmierer gemachten impliziten Maßnahmen sichergestellt, unterstützt und eventuell besser umgesetzt. Zum Beispiel könnten die expliziten Maßnahmen zur erhöhten Sicherheit (Einbauen von Dialog-Abfragen) die impliziten Überlegungen und Umsetzungen zur Benutzbarkeit verbessern (kein versehentliches Überschrieben von Dateien).
Des Weiteren können durch implizite, die Programmiersprache betreffende Maßnahmen bereits Vorteile entstehen (z.B. Plattformunabhängigkeit bei JAVA oder ordentlich formatierter Code bei Python). Explizite Maßnahmen kosten zu Beginn eventuell mehr Zeit, zahlen sich am Ende jedoch oft aus (bessere Wartbarkeit durch guten Code / gute Dokumentation, besseres Endprodukt mit höherer Qualität).

1 Aufgabe 1.2

(a) 1. **Qualitätsmaß:** meantime between failures (MTFB)
2. **Qualitätsmerkmal**: Zuverlässigkeit
3. **Ausprägung:** > 500s

(b) 1. **Qualitätsmerkmal:** Benutzbarkeit (Usability)
2. **Qualitätsteilmerkmal:** Verständlichkeit
3. **Qualitätsmaß:** Befragung von Testnutzern
4. **Ausprägung:** Skala von 1 - 10 (wobei 1 = unverständlich und 10 = intuitiv)

(c) **Verfügbarkeit** $= \frac{MTBF}{MTBF+MTTR}$
mit $MTBF = \frac{Betriebsdauer}{Fehleranzahl}$

$$MTBF = \frac{(t2-t1)+(t4-t3)+(t6-t5)}{3}$$

$$MTBF = \frac{23}{3} = 7.67$$

daraus folgt:

$$Verfügbarkeit = \frac{7.67}{7.67+((t3-t2)+(t5-t4))}$$

$$Verfügbarkeit = \frac{7.67}{12.67} = 0.605$$