Karlsruher Institut für Technologie

Lamb.da - Das Spiel

Testbericht

Farid El-Haddad, Florian Fervers, Kai Fieger,
Robert Hochweiß, Kay Schmitteckert

26. März 2015

Inhaltsverzeichnis

1 Einleitung

2 Verbesserungen

3 Änderungen am Pflichtenheft

4 Testwerkzeuge

4.1 Statische Werkzeuge zur Codeanalyse

- Checkstyle
  - Checkstyle ist ein freies statisches Codeanalyse-Werkzeuge, welches verwendet wird, um die Java-Codequalität zu verbessern. Dieses statische Analyse-Werkzeug lässt sich mit einer geeigneten Datei konfigurieren, um zu entscheiden nach welchen Kriterien die Qualität verbessert wird. Der von uns verwendete Checkstyle ist im Repository vorzufinden und prüft unter anderem die Methodenlänge, Dateilänge, Anzahl an Methodenparameter, richtige Formatierung des Codes (Whitespaces etc.), Namenkonventionen

- FindBugs
  - FindBugs ist ein freies statisches Codeanalyse-Werkzeug, welches verwendet wird, um insbesondere Fehlermuster in Java-Code bzw. im Java-Bytecode zu finden. Diese Fehlermuster können unter anderem auf wirkliche Fehler hindeuten, müssen sie aber nicht. Wir haben dieses Werkzeug verwendet, um die eben angesprochenen Fehlermuster aufzudecken und zu überprüfen, ob diese zu Fehlern führen oder nicht. Alle gefundenen Bugs waren recht klein und führten demnach nicht zu Fehlern, weshalb eine Beseitigung nicht nötig war.

4.2 Dynamische Werkzeuge zur Codeanalyse

- Code Coverage mit EMMA
  - EMMA ist ein dynamische Codeanalyse-Werkzeug, welches verwendet wird, um die Code Coverage (Testüberdeckung) von JUnit-Tests zu messen. Mit dieser Testüberdeckung lässt sich oft Code herausfiltern, welcher niemals ausgeführt wird und weiterhin die Abdeckung von diversen Modultests. EMMA basiert auf einer partiellen Zeilenüberdeckung, in der teilweise ausgeführte Zeilen registriert werden, z.B. bei verzweigenden Anweisungen. Dieses dynamische Werkzeug wurde von uns verwendet, um die Testüberdeckung der JUnit-Model-Tests zu beobachten und zu analysieren.

- MonkeyRunner
  - MonkeyRunner ist ein dynamisches Werkzeug, welches mit Python-Programmen ausgeführt wird. Das Werkzeug funktioniert für Android-Geräte oder solche Emulatoren. Dieses dynamische Werkzeug führt vorgeschriebene Python-Dateien aus und klickt automatisiert an die in der Python-Datei definierten Stellen, wobei es immer wieder Screenshots aufnimmst. MonkeyRunner wurde von uns verwendet, um unsere definierten globalen Testfälle und Systemtests nachzustellen, um diese nicht immer wieder per Hand durchzugehen.

- Monkey
  - Monkey ist ein dynamisches Werkzeug, welches gerne für Belastungs- und Stresstests für Android-Applikationen verwendet wird. Dieses Tool läuft entweder direkt auf Android-Geräten oder auf solchen Emulatoren. Dieses dynamische Werkzeug generiert zufällige und hochfrequentierte Eingaben und prüft somit, ob irgendwelche unerwarteten Exceptions oder Fehler auftreten bzw. ab welcher Frequenz von Eingaben die Applikation abstürzt. Dieses Tool wurde von uns verwendet, um die eben beschriebenen Belastungs- und Stresstests durchzuführen.

5 Unittests

6 Integrationstests

7 Systemtests

8 Statistik

9 Lasttests

Expandierende Lambda-Terme Zunächst wurde die Auswertung eines immer weiter wachsenden Lambda-Terms im Reduktionsmodus in einer Endlosschleife durchgeführt. Dies führte dann häufig zu StackOverflowExceptions. Dieses Problem behoben wir, indem wir eine Maximalzahl an Knoten (100) in den Lambda-Termen und damit in den Elementen einführten, die die Reduktion der Lambda-Terme begrenzt.

Große Lambda-Terme Die Auswertung von Lambda-Termen, welche aus sehr vielen Lämmern und Edelsteinen bestehen, wurde ebenfalls getestet. Dabei kam es zu keinen größeren Performanceeinbußen, jedoch führten sehr große Lambda-Terme teilweise ebenfalls zu StackoverflowExceptions, was so wie bei expandierenden Ausdrücken durch Einführung einer Maximalanzahl Von Elementen behoben wurde.

Willkürliche Bildschirmeingaben Mithilfe des monkey-Werkzeugs wurde getestet wie sich die Applikation bei zufälligen und hochfrequenten Bildschirmeingaben verhält. Das monkey-Werkzeug wurde also für Stress-Tests verwendet. Beim mehrmaligen Testläufen mit unterschiedlichen Event Sequenzen konnten unter normalen Bedingungen keine Auffälligkeiten festgestellt werden, es kam zu keinen Abstürzen oder sonstigen unerwarteten Verhaltensweisen.

10 Behobene Bugs

10.1 Shop

- Anzeige von mehreren aktivierten Items pro Kategorie
  - Der Bug im Shop, wenn man zwei verschiedene Items einer Kategorie aktivierte, dass beide als aktiviert angezeigt werden, wurde behoben und somit wird der Status jedes Items nun richtig angezeigt.
- Fehlende Anzeige aktivierter Items nach Programmneustart
  - Der Bug im Shop, dass ein bereits aktiviertes Item nach einem Neustart des Programms nicht mehr als aktiviert angezeigt wird, wurde behoben und nun werden beim Laden eines Profils, wechseln des Profils sowie beim Neustarten alle Zustände der Items richtig angezeigt.
- Schließen offener Kategorien nach Item-Einkauf
  - Der Bug im Shop, wenn ein Item gekauft wurde und direkt nach dem Kauf alle offenen Kategorien geschlossen wurden, wurde behoben und der aktuelle Zustand einer Kategorie (ob offen oder geschlossen) wird nach einem Kauf nun beibehalten.

11 Glossar