Karlsruher Institut für Technologie

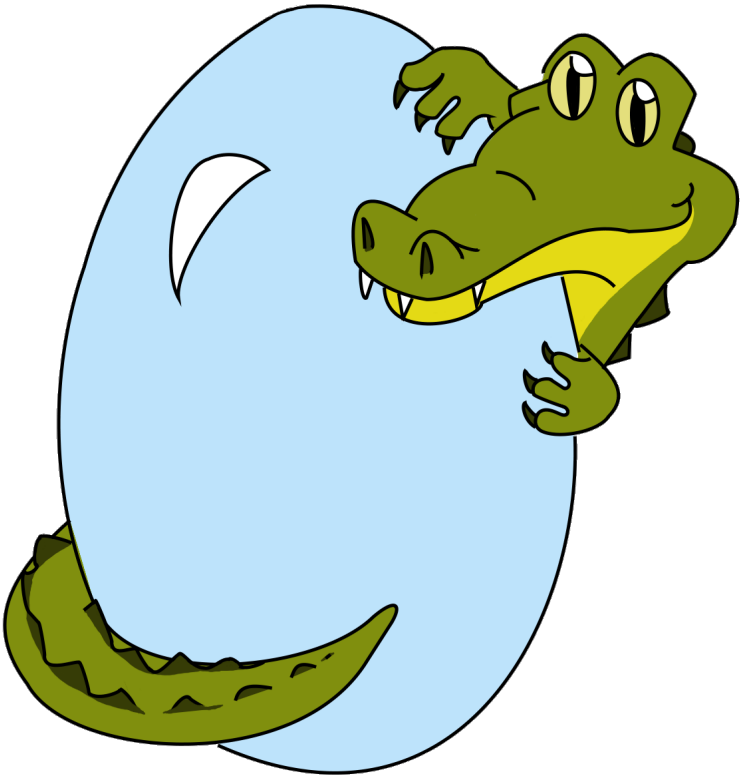
Croggle

Lernanwendung für Grundschüler

Qualitätssicherung

Lukas Böhm, Tobias Hornberger, Jonas Mehlhaus,
Iris Mehrbrodt, Vincent Schüßler, Lena Winter

27. März 2014



Inhaltsverzeichnis

1 Einleitung

Nach dem Abschluss der Implementierungsphase bot Croggle eine Vielfalt an Funktionen, welche bereits zu großen Teilen fehlerfrei zusammenspielten. Trotzdem fühlte sich das Produkt an einigen Stellen noch unfertig an. So kam es etwa zu gelegentlichen Abstürzen oder inkonsistentem Verhalten; auch die Bedienung und die Oberfläche waren noch etwas „kantig“.

In der nun abgeschlossenen Testphase wurde jetzt die Gelegenheit wahrgenommen, möglichst alle Störfaktoren zu beheben und an der ein oder anderen Stelle sogar noch größere Verbesserungen zu integrieren. Alle Änderungen werden im Folgenden im Detail aufgeführt und erläutert. Zudem wurde eine umfangreiche Testbasis geschaffen, um die Qualität der Software nicht nur zu belegen, sondern auch für die Zukunft zu sichern. Bei der Erstellung der Tests wurde sich hierzu am sogenannten „V-Modell“ orientiert, wodurch sich die Testfälle in Kategorien einteilen lassen, wie sie sich als einige der Kapitelüberschriften in diesem Dokument wiederfinden.

2 Integrationstests

2.1 Controller

Für die Überprüfung der fehlerlosen Zusammenarbeit der voneinander abhängigen Systemkomponenten wurde eine Reihe weiterer Testfälle geschrieben. Dazu gehören vor allem Tests, die mittels des TestHelpers eine Instanz der AlligatorApp beziehen, da dieser die komplette Applikation in einem „headless“ Modus (d.h. ohne grafische Ausgabe) aufbaut und bereitstellt. Viele der Controller konnten so bereits in einer nahezu „realen“ Umgebung getestet werden, u.a.

ProfileController der ProfileControllerTest beinhaltet Sequenzen zur Verifizierung der Codebereiche, die Informationen zu Profilen in der Datenbank speichern. Hauptsächlich werden hier Profile angelegt, verändert und gelöscht. Aber auch das Setzen und Auslesen des aktuell aktiven Profils mittels libGdx' ApplicationPreferences wird hier überprüft. Hinzu kommen noch Tests zur Benachrichtigungsübermittlung bei Änderungen am derzeitigen Profil an andere Programmteile und die Validierung von Nutzerdaten.

SettingController wird in SettingControllerTest im Zusammenspiel mit dem ProfileManager und in diesem Zusammenhang auch der Datenbank getestet.

StatisticController auch dieser Controller wird in StatisticControllerTest auf die fehlerfreie Zusammenarbeit mit dem ProfileController getestet

AchievementController der entsprechende Testfall interagiert direkt mit dem entsprechenden Datenbankmanager bzw. über den PersistenceManager

2.2 Simulator

Desweiteren befassen sich mehrere Testfälle in SimulatorTest.java mit der fehlerfreien Funktionsweise des Simulators. Dabei wird unter anderem Folgendes betrachtet:

Validierung & Lastgrenzen Es existiert ein eigenes Unterpaket (operations.validation) um vom Nutzer eingegebene Konstellationen vor dem Simulationsbeginn auf Validität zu prüfen. Dass die darin enthaltenen Validierer unter den Bedingungen während der Simulation funktionieren, wird im SimulatorTest überprüft. Die Einhaltung der Lastgrenzen bei einer langwierigen Simulation wird durch testStackOverflowIssue getestet. Dabei dürfen keine Fehler passieren, abgesehen von jenen, die bei der Überschreitung der 300-Alligatoren Grenze auftreten.

Evaluation Die Validierung von korrekten Ergebnissen der Simulation wird in mehreren Tests durchgeführt und beinhaltet somit auch die Überprüfung einer ganzen Reihe von „board.operations“, die an der Evaluation beteiligt sind. Dazu zählen u.a.

- CopyConstellation
- ExchangeColor
- FindEating
- uvm...

2.3 Renderer

Das wohl größte Modul in Bezug auf Komplexität und Vernetzung mit anderen Subsystemen stellt der BoardActor im „renderer“-Paket dar. Der BoardActorTest prüft in dieser Hinsicht das Verhalten des Gesamtsystems bei einer sehr langen Simulation. Dafür nimmt es den expandierenden Lambda-Term „λg.(λx.g (x x)) (λx.g (x x))“ für eine Simulation und meldet den BoardActor als BoardEventListener an deren Simulator an. Bei der anschließenden Simulation muss der BoardActor zuerst die Ausgangskonstellation darstellen („layouten“) und anschließend auf alle Änderungen an der Kosntellation reagieren. Dies geschieht unter Zuhilfenahme der Funktionen des „layout“-Pakets.

3 Systemtests

Die im Pflichtenheft aufgeführten Funktionssequenzen konnten mithilfe des monkeyrunner Werkzeugs automatisiert werden. Für jede einzelne Sequenz wurde ein eigenes Python Programm erstellt, in dem auch die Vorbedingungen, zum Beispiel das beim Start des Testfalls ein bestimmter Bildschirm angezeigt oder eine bestimmte Anzahl an Benutzter im System gespeichert sind, aufgeführt sind. Bei allen Tests ist vorausgesetzt, dass Croggle auf dem Gerät installiert ist und sich in der Anwendung befindet. Die Benennung der Testszenarien orientiert sich dabei am Pflichtenheft, das Programm zum Testszenario „Ein Benutzerprofile erstellen (T110)“ hat so etwa den Namen „t110.py“. Alle im Pflichtenheft beschriebenen Funktionssequenzen konnten erfolgreich durchlaufen werden, wobei es zu kleinen Anpassungen an die letztendliche Implementierung zu Croggle kam. Diese Veränderungen traten in folgenden Szenarien auf:

- „Ein Benutzerprofil erstellen (T130)“: Im dieser Funktionssequenz wurde beschrieben, dass der Benutzer um seinen Namen zu ändern auf seinen angezeigten Benutzernamen klickt. In der jetzigen Fassung von Croggle wird kein Benutzernamen mehr angezeigt, der Anwender klickt stattdessen auf einen Button mit der Beschriftung „Umbenennen“. Gleiches gilt für das Ändern des Profilbilds, im Pflichtenheft wurde das Fenster zur Änderung über den Klick auf ein angezeigtes Profilbild erreicht. Dieses Bild wurde durch einen Button mit der Aufschrift „Avatar ändern“ ersetzt.

- „Ein Benutzerprofil löschen (T130)“: Hier wurde angenommen, dass man nach dem Löschen eines Profils direkt in das Hauptmenü wechselt. Dies wurde geändert, um Zustände zu vermeiden in denen es kein aktives Profil gibt, da hier zum Beispiel nicht definiert ist, ob und wohin Änderungen an den Einstellungen oder Fortschritte bei Levels gespeichert werden. Nach dem Löschen eines Profils wechselt der Benutzer deshalb nun in die Listenansicht zur Auswahl eines Profils und kann so ein neues Profil erstellen oder sich mit einem bereits bestehenden anmelden.

- „Den Simulationsmodus bedienen (T170)“: Der hier aufgeführte „Schneller“-Button wurde mit dem „Langsamer“-Button durch einen Schieberegler zur Einstellung der Simulationsgeschwindigkeit ersetzt, was die Bedienung vereinfacht hat.

Die im Pflichtenheft beschriebenen Funktionssequenzen wurden durch Folgende ergänzt:

- „Ein MultipleChoice Level verlieren (T200)“: Der Anwender hat Level 4 Paket 2 neu geladen. Die Aktionen des Benutzers bestehen in diesem Szenario aus dem anzeigen lassen des auszuwertenden Ausdrucks, der Wahl mehrere Antwortmöglichkeiten, dem Start und der Ausführung der Simulation und letztendlich dem Neustart des Levels.

- „Ein Edit Level gewinnen (T210)“: Der Spieler hat ein neues Profil erstellt, hat Level 1 Paket 1 geladen und befindet sich so im Platziermodus. Er lässt sich nun den Zielzustand anzeigen, färbt das präsentierte Ei ein, simuliert die Auswertung des erstellten Ausdrucks und gewinnt so das Level. Er schaut sich sein freigeschaltetes Achievement an und startet dann das nächste Level.

- „Freigeschaltet Achievements anschauen (T220)“: Der Benutzer befindet sich im Hauptmenü und wechselt über einen Klick auf den entsprechenden Button in den Achievements-Bildschirm. Hier wählt er mehrere der angezeigten Achievements aus und bekommt so Informationen über diese präsentiert. Anschließend kehrt er wieder in das Hauptmenü zurück.

- „Den Credit Bildschirm aufrufen (T230)“: Vom Hauptmenü aus wechselt der Anwender durch einen Klick auf das Croggle Logo in den Credit Bildschirm, welchen er anschließend wieder verlässt.

- „Die Applikation beenden (T240)“: Im Hauptmenü drückt der Anwender die Android Zurück-Button und bekommt so einen Dialog präsentiert, in dem er gefragt wird, ob er die Applikation verlassen will. Dies bestätigt er und Croggle wird so beendet.

4 Lasttests

Expandierende Ausdrücke Die Auswertung eines immer weiter wachsenden Lambda-Terms wird im Simulator in einer Endlosschleife weitergeführt. Wo zu Beginn noch häufig StackOverflowExceptions auftraten, wird die Simulation nun durch die Einhaltung der Maximalzahl an Elementen (300) begrenzt.

Große Ausdrücke Die Auswertung von Lambda-Termen, welche durch sehr viele Alligatoren und Eier präsentiert wurden, wurde simuliert. Dabei konnte es zu erhebliche Performanceeinbußen kommen, in manchen Fällen dauerte es über eine Minute bis ein einzelner Auswertungsschritt vollständig abgearbeitet und simuliert wurde. In dieser Zeit fror der Bildschirm ein und der Benutzer erhielt keine Rückmeldung über die Gründe für dieses Verhalten. Als Hauptlast stellte sich das Schreiben eines JSON Strings heraus, welcher zum Wiederherstellen des Zustands nach dem Pausieren oder dem Beenden der Applikation verwendet wurde. Die Methoden zum Schreiben des Strings wurde überarbeitet und in einen eigenen Thread ausgelagert, was die Performance deutlich verbesserte.

Willkürliche Bildschirmeingaben Mithilfe des monkey Werkzeugs wurde getestet wie sich die Applikation bei zufälligen und hochfrequenten Bildschirmeingaben verhält. Beim mehrmaligen Testläufen mit unterschiedlichen Event Sequenzen konnten keine Auffälligkeiten festgestellt werden, es kam zu keinen Abstürzen oder sonstigen unerwarteten Verhaltensweisen.

5 Überdeckung

Package	IC	BC	CI	MI	CB	MB
Insgesamt	89%	78%	26612	3235	1378	393
de.croggle.ui.screens	88%	72%	6093	831	152	58
de.croggle.ui.renderer	89%	74%	4208	544	222	80
de.croggle.ui.renderer.layout	75%	55%	1418	476	69	57
de.croggle.util.convert	88%	74%	1992	267	172	60
de.croggle.ui.actors	80%	43%	985	240	18	24
de.croggle.game	88%	78%	1181	162	70	20
de.croggle.game.level	88%	70%	1025	141	57	25
de.croggle.data	82%	66%	477	108	25	13
de.croggle.ui.renderer.objectactors	85%	64%	589	106	14	8
de.croggle.game.sound	80%	67%	320	79	16	8
de.croggle.game.board	91%	97%	586	60	64	2
de.croggle	85%	75%	318	56	6	2
de.croggle.game.board.operations	97%	92%	1898	49	145	13
de.croggle.ui	92%	80%	346	31	8	2
de.croggle.data.persistence	96%	82%	755	28	55	13
de.croggle.backends.sqlite	55%	n/a	30	25	-	-
de.croggle.game.event	87%	89%	147	22	16	2
de.croggle.board.operations.validation	98%	98%	424	10	40	1
de.croggle.game.achievement	100%	100%	1489	0	80	0
de.croggle.data.persistence.manager	100%	98%	1260	0	51	1
de.croggle.util	100%	100%	645	0	50	0
de.croggle.game.profile	100%	92%	393	0	46	4
de.croggle.backends	100%	100%	33	0	2	0

Legende:
IC = Instruction Coverage
BC = Branch Coverage
CI = Covered Instruction
MI = Missed Instruction
CB = Covered Branches
MB = Missed Branches

n/a: In diesem Fall macht eine Branchabdeckung keinen Sinn da keine Abzweigungen innerhalb des Packages existieren.

In dem mit dem Dokument mitgeschickten EclEmma Coverage Report sind diese Daten und mehr in einem interaktiven Format zusammengefasst. Er erlaubt die eingehende Betrachtung der Überdeckung und ermöglicht es die Abdeckung in den einzelnen Paketen, Klassen und Methoden detailliert aufzulisten.

6 Feldtest

Zusätzlich zu den in den vorherigen Kapiteln beschriebenen Tests wurde ein Feldtest durchgeführt. Für diesen wurde die Applikation von 25 Personen im Alter zwischen 13 und 56 Jahren getestet und anschließend jeweils ein Fragebogen ausgefüllt. Unter den Testpersonen befanden sich sowohl Menschen mit, als auch ohne Vorkenntnissen bezüglich des Lambdakalküls und des Umgangs mit Tablets bzw. Smartphones. Im Folgenden werden die Erkenntnisse, die wir durch den Feldtest erhalten haben, zusammengefasst und die daraus resultierenden Änderungen an der Applikation aufgeführt.

6.1 Auswertung der Tests

6.1.1 Erreichen und Bedienen von Statistiken, Achievements und Einstellungen

Sowohl die Statistiken als auch die Achievements und die Einstellungen sind laut den Testpersonen sehr intuitiv zu bedienen.

6.1.2 Verwalten von Benutzerprofilen

Benutzerprofile sind laut den Tests einfach anzulegen und zu wechseln, obwohl es sehr vereinzelt vorkam, dass Testpersonen den Button zum Wechseln eines Profils nicht als solchen erkannten. Das Modifizieren der Benutzerprofile wurde ebenfalls als sehr intuitiv bewertet.

6.1.3 Wartezeiten

Es traten keine Wartezeiten auf, die den Testpersonen zu lange erschienen.

6.1.4 Spielprinzip und Level

Einfärbe- und Einfüge-Level Das Spielprinzip dieser Leveltypen wurde neutral, mit einer leichten Tendenz zum Positiven, bewertet. Als größter Kritikpunkt wurden die Tutorials angeführt. Diese waren, vor allem zu Beginn, unverständlich.

Multiple-Choice-Level Multiple-Choice-Level wurden vom Spielprinzip her etwas schlechter bewertet als die anderen Leveltypen. Kritikpunkte waren auch hier die Tutorials und zusätzlich der unübersichtliche Aufbau des Bildschirms.

Hinweise Die Hinweise wurden von fast allen Testpersonen benutzt und als hilfreich bewertet.

Schwierigkeit der Level Die Schwierigkeit der Level wurde von den Testpersonen als angemessen bewertet.

Farben Es wurde von vielen Testpersonen bemerkt, dass sich manche der verwendeten Farben sehr ähnlich sehen. Dieses Problem trat im Farbenblindmodus nicht auf.

6.2 Resultierende Änderungen

Tutorial Die Tutorials wurden noch einmal von Grund auf überarbeitet.

Multiple-Choice-Screen Es wurden kleinere Änderungen in der Darstellung der Antwortmöglichkeiten vorgenommen. So wurden diese näher zusammengerückt, sodass der Bildschirm nun übersichtlicher erscheint.

Farben Bei den Farben wurde sowohl die Farbpalette als auch die Auswahl der Farben in den Leveln angepasst.

7 Verbesserungen

Drag&Drop: Ermögliche das intuitive Einfügen von neuen Elementen auf einer beliebigen Spielfeldhälfte

Drag&Drop: Ermögliche das Einfügen von neuen Elementen vor und nach jedem bereits vorhandenen Spielfeldobjekt

Zurücktaste: Entferne den bisherigen Stack und führe „logischen“ Vorgänger ein für eine intuitive Nutzung von Androids „zurück“-Knopf

Darstellungseffizienz: Setze sogenanntes „Pooling“ für ActorDeltas (atomare Änderungen an der dargestellten Konstellation) um

Tutorials: Verbessere die bisherigen Tutorials drastisch und ergänze um neue Tutorials

Statistiken: Registriere vollendete Pakete und gewonnene Levels

Statistiken: Registriere platzierte Alligatoren und Eier

Zoom Zeige Zoom Knöpfe automatisch auf nicht multitouch-fähigen Geräten

Multiple Choice Level Stelle die Antwortmöglichkeiten komprimierter dar: Jede Möglichkeit nimmt nur noch so viel Platz ein, wie sie maximal benötigt. Dadurch muss unter anderem weniger gescrollt werden.

Multiple Choice Level Verhindere außerdem, dass mehr als eine Möglichkeit auf einmal ausgewählt ist.

Grafiken Ersetze einige Grafiken, die noch durch Platzhalter dargestellt wurden.

Profilwahl Zeige in der Liste beim Profilwechsel neben dem Benutzernamen außerdem das gewählte Profilbild an. Dies dient der schnelleren Identifikation des Profils.

Level-beendet Menü Passe das Menü für den Fall an, dass ein Level verloren wurde: Der Hauptmenüpunkt ist das Wiederholen des Levels.

Multiple Choice Level Öffne die Startkonfiguration bei Start des Levels. Dies soll den Ablauf logischer gestalten, da nicht mehr die “Frage“ erst nach den “Antworten“ gezeigt wird.

8 Gefundene und behobene Bugs

Drag&Drop: Behebe Fehler, bei dem bereits platzierte Elemente nach Halten und Warten nicht direkt unter der Fingerspitze erschienen

Drag&Drop: Behebe Fehler, bei dem nach Halten und Warten auf bereits platzierten Elemente der Zoom als Rückmeldung nicht rückgängig gemacht wurde

Multiple Choice: Verbiete das Übergehen in die Simulation ohne eine Antwort gewählt zu haben

Alpha-Äquivalenz: Behebe Fehler beim Vergleich zweier Bäume auf Alpha-Äquivalenz

Lastgrenze: Selbst bei Konstellationen, die aus nur 300 Elementen bestanden, kam es durch das Visitor-Pattern an einigen Stellen während der Simulation zu StackOverflowExceptions. Dies wurde durch Modifikation und teilweise Neuimplementierung der folgenden Operationen behoben:

- BoardObject.copy
- GetParentHierarchy
- CountBoardObjects
- ActorLayouter
- CreateWidthMap
- CreateHeightMap

Alligatoren → Json Behebe Fehler bei der Konvertierung, wo ungeprüft auf das Elternelement von Eiern zugegriffen wurde, wodurch NullPointerEcxeptions auftreten konnten.

Json → Alligatoren Korrigiere falsche Annahme zu libGdx' JsonValue.hasChild(String), wo davon ausgegangen wurde, dass „wahr“ zurückgegeben wird, falls ein Kind mit

dem Namen im String existiert. Stattdessen besagt die Dokumentation allerdings, dass zurückgegeben wird, ob ein Kind mit dem gegebenen Namen ein Kind hat.

Zeitgesteuerte Aktionen Es existiert ein Fehler in libGdx, der bei einer unglücklichen Sequenz von „App verlassen“ „warten“ „App wieder starten“ dazu führt, dass das Timer-Werkzeug nicht mehr funktioniert. Dieser Fehler wurde durch einen Workaround umgangen, bis das Problem in libGdx behoben ist. Siehe dazu als Referenz `https://github.com/libgdx/libgdx/issues/1548`.

Laden und Speichern von Benutzernamen Beim Erstellen und Laden von Benutzern, deren Namen ein oder mehrere Apostrophe enthielten, konnte es zu Abstürzen der Applikation kommen. Der Grund hierfür war, dass die Apostrophe die Syntax der SQL SELECT Anweisung, welche zum Laden bestimmter Einträge in der SQL Datenbank verwendet wird, so verändert hatte, dass sie nicht mehr ausgewertet werden konnten und es zu ungefangenen SQLExceptions kam. Der Fehler wurde behoben indem Strings nun nicht mehr direkt in die SELECT Anweisung eingefügt, sondern als Argumente in die „query“-Methode übergeben werden.

Lambdaterm → Alligatoren Behebe Endlosschleife falls in einem Term zu viele Bindungsseparatoren vorkommen.

Lambdaterm → Alligatoren Behebe StringIndexOutOfBoundsException falls in einem Term zu wenige Bindungsseparatoren vorkommen.

Lambdaterm → Alligatoren Gib Fehler zurück, falls in einem Term eine Abstraktion keine Variable bindet.

Farbenblindenmodus Behebe Fehler, der dazu führte, dass nach einer Änderung der Einstellung bezüglich des Farbenblindenmodus Objekte aus der Objektleiste im falschen Modus dargestellt wurden.

Lokalisierung Ersetze Strings, die noch nicht lokalisierbar waren.

Hauptmenü Verändere das Layout dahingehend, dass nicht mehr in manchen Fällen die Titelgrafik zu groß repräsentiert wird, und so den Rest des Menüs staucht.

Simulator Überprüfe das Eingabeboard auch auf fehlende Kinder bei AgedAlligators, nicht wie zuvor nur bei ColoredAlligators.

9 Testwerkzeuge

Um während der Fehlersuche bei auftretenden Problemen schnell das betroffene Systemsegment zu lokalisieren, aber auch um tatsächlich die Qualität des Produkts für die Zukunft zu sichern, wurden während der Phase verschiedene weiter Werkzeuge eingesetzt. Diese werden im Folgenden erklärt und deren Einsatz für etwaige zukünftige Wartungsarbeiten bzw. Produkterweiterungen dokumentiert.

9.1 EclEmma

Mit EclEmma konnte während der Entwicklung der Testfälle in Echtzeit die Überdeckung der einzelnen Codezeilen überprüft werden. Zudem wurden mit dem Werkzeug einige der Überdeckungsreports aufgezeichnet und nach HTML exportiert.

9.2 monkey

monkey eignet sich vor allem dazu, zu testen wie sich eine gerade laufende Applikation bei scheinbar zufälligen und hochfrequenten Benutzereingaben verhält. In unserem Projekt wurde das Werkzeug vor allem für reproduzierbare Stresstest verwendet.

9.3 monkeyrunner

monkeyrunner wurde in der Qualitätssicherungsphase vor allem dazu verwendet, Testszenarien aus dem Pflichtenheft zu automatisieren und ohne die Aufsicht eines Entwicklers zu benötigen nachzustellen. So wurden unter anderem die im Pflichtenheft beschriebenen Funktionssequenzen vollständig durch das Werkzeug automatisiert.

9.4 SQLite Manager

Mithilfe des SQLite Managers konnten die im Pflichtenheft garantierten Datenkonsistenzen anhand des während der Implementierungsphase entstandenen Datenbankschemas überprüft werden. Zudem war es möglich, Probleme bei Inkonsistenzen innerhalb der Datenbank schnell reproduzieren zu können.

10 Glossar

Im Folgenden befindet sich eine Übersicht mit in diesem Doument verwendeten Fachbegriffen und dazugehörigen Erklärungen.

Apk Abkürzung für „Android Package“. Beschreibt das Format, in dem für Android zusammengestellte Programmpakete gepackt und ausgeliefert werden.

V-Modell von Barry Boehm im Jahre 1979 vorgestellte und das Wasserfallmodell erweiternde Herangehensweise an den Softwareentwicklungsprozess, der hauptsächlich die Phasen Implementierung und Test um weitere Ergebnisartefakte ergänzt.

EclEmma kostenloses Eclipse-Plug-In, mithilfe dessen man sehr einfach und schnell die Codeabdeckung von Java-Programmen ermitteln kann. Direkt nach der Installation kann der Benutzter zum Beispiel JUnit-Tests oder lokale Java Applikationen durch eine vorhandene oder neu erstellte Run-Konfiguration als „Coverage“ starten und bekommt so nach deren Terminierung einen Bericht über die Codeabdeckung des Programms präsentiert. In diesem wird ausführlich aufgeführt, wie viel und welcher Code in den jeweiligen Paketen und Klassen überdeckt wurde. Außerdem wird dem Benutzer direkt in der Eclipse-Entwicklungsumgebung farblich angezeigt welche Codeabschnitte durchlaufen und welche nicht erreicht werden konnten. EclEmma bietet desweiteren die Möglichkeit, verschiedene Berichte zusammenzufassen und Berichte direkt als HTML Report oder XML Datei zu exportieren.

monkey Tool ein im Android SDK enthaltenes Kommandozeilenwerkzeug, das dazu genutzt werden kann zufällige Benutzereingaben zu simulieren. Es läuft auf dem Emulator oder einem Android-Gerät und kann eine Sequenz von pseudozufällig erzeugten Events wie Touch- oder Klickeingaben erzeugen. Der Anwender hat dabei die Möglichkeit die Rahmenbedingungen für einen Lauf des monkey-Tools zu setzen. So kann er unter anderem die Anzahl oder den Typ der erzeugten Events über die Kommandozeile bestimmen. Außerdem hat er die Möglichkeit den „Seed“, welcher für die Erzeugung der pseudozufälligen Events verwendet wird, zu modifizieren, wodurch er sicherstellen kann, dass bei jeder Ausführung die gleichen Events ausgelöst werden.

monkeyrunner ein in der Android SDK mitgeliefertes Werkzeug, das genau wie monkey dazu verwendet werden kann Benutzereingaben außerhalb von Android Code

zu simulieren. Im Gegensatz zu monkey sind diese Eingaben aber nicht zufällig, monkeyrunner bietet vielmehr eine API mit der man Python Programme erstellen kann, welche zum Beispiel Click oder Drag and Drop Gesten an gewählten Bildschirmkoordinaten simulieren können. Über die Programme kann der Anwender zum Beispiel auch das Drücken der „Back“ und der „Home“ Taste simulieren, Screenshots vom gerade angezeigten Bildschirm aufnehmen oder eine apk installieren und das installierte Programm dann abschließend ausführen.

SQLite Manager Firefox Add-on, mithilfe dessen man den Inhalt von SQLite Datenbanken einfach visualisieren und bearbeiten kann. Nach der Installation kann der Benutzer über eine grafische Benutzeroberfläche eine Verbindung mit einer bereits bestehenden SQLite Datenbank aufbauen und bekommt so Details zu den einzelnen in der Datenbank enthaltenen Tabellen präsentiert. Veränderungen des Inhalts einzelner Zellen, Spalten oder Zeilen sind dabei ebenso möglich, genauso wie das Erstellen und Löschen von Einträgen in den Tabellen.