aus

Oliver J. Bott, Peter Fricke, Uta Priss, Michael Striewe (Hrsg.)

Automatisierte Bewertung in der Programmierausbildung

Digitale Medien in der Hochschullehre Band 6
2017, 420 Seiten, br., 42,90 €, ISBN 978-3-8309-3606-0

Waxmann Verlag GmbH
www.waxmann.com info@waxmann.com

16 Der Grader ASB

Britta Herres, Rainer Oechsle und David Schuster

Zusammenfassung
Im Fachbereich Informatik der Hochschule Trier wird seit 2006 das ASB-System (Automatische Softwarebewertung) entwickelt und eingesetzt. Die webbasierte Client-Server-Anwendung ermöglicht die automatische Bewertung von studentischen Lösungen zu Programmieraufgaben innerhalb eines festgelegten Zeitraums. In diesem Beitrag wird das ASB-System aus technischer Sicht beschrieben. Das ASB-System ist ein Komponentensystem, dessen Infrastruktur von einem verteilten Ereignisbus gebildet wird, über den die Komponenten innerhalb eines Rechners, aber auch über Rechnergrenzen hinweg, kommunizieren können. Die einzelnen Bestandteile bilden ein Schichtensystem bestehend aus Ausführungsumgebungen, Bewertungsmaßnahmen (Plugins) und Konfigurationseinstellungen für die Bewertungsmaßnahmen. Es werden statische und dynamische Bewertungsmaßnahmen unterstützt. Eine besondere Herausforderung stellt die dynamische Bewertung von Android-Apps dar. Auf die Integration der Bewertungsmaßnahme für Android-Apps in das ASB-System wird in diesem Beitrag speziell eingegangen.

16.1 Einleitung

Das System zur automatischen Bewertung von Software (ASB) ist eine webbasierte Client-Server-Anwendung, die seit 2006 im Fachbereich Informatik der Hochschule Trier entwickelt und eingesetzt wird. Ziel dieses Systems ist die automatisierte Bewertung studentischer Lösungen von Programmieraufgaben. Durch die Vielzahl von Studierenden in programmierlastigen Lehrveranstaltungen soll das ASB-System den Aufwand des händischen Korrigierens von Lösungen zu Programmieraufgaben deutlich reduzieren und Studierenden sofortiges Feedback über die Korrektheit ihrer erstellten Lösungen geben.

Im ASB-System werden Lehrveranstaltungen und die dazugehörigen Programmieraufgaben abgebildet. Zu Aufgaben können Lösungen eingereicht werden. Eine Lösung besteht in der Regel aus einer oder mehreren Quellcodedateien, die als ZIP-Datei gepackt und hochgeladen werden. Für Java-Programme muss eine ZIP-Datei so strukturiert sein, dass sich nach dem Entpacken eine Verzeichnisstruktur ergibt, die den Packages entspricht, in denen sich die Klassen aus den Quellcodedateien befinden. Eine eingereichte Lösung wird in der Regel mehreren Bewertungsmaßnahmen unterzogen, wobei man zwischen statischen und dynamischen Bewertungsmaßnahmen unterscheiden kann. Bei statischen Bewertungsmaßnahmen wird der Quellcode der eingereichten Programme analysiert. Bei dynamischen Bewertungsmaßnahmen wird das zu bewertende Programm ausgeführt und sein Verhalten geprüft.

Es lassen sich drei Nutzergruppen des ASB-Systems unterscheiden:

- eine eher technisch orientierte Nutzergruppe: Personen dieser Gruppe legen die Bewertungsmaßnahmen auf dem ASB-System an, konfigurieren, ändern und löschen diese gegebenenfalls auch wieder, erzeugen Lehrveranstaltungen und Aufgaben, ordnen existierende Bewertungsmaßnahmen den Lehrveranstaltungen und Aufgaben zu und legen für die Aufgaben Zeiträume fest, in denen Lösungen eingereicht werden können;
- die Gruppe der Lehrenden: die Lehrenden sehen sich die von den Studierenden eingereichten Lösungen und deren Bewertungen an;
- die Gruppe der Studierenden: die Studierenden laden ihre Lösungen auf den ASB-Server, betrachten ihre Bewertungsergebnisse und laden im Fehlerfall ihre veränderten Lösungen erneut hoch.

Das ASB-System wurde im Laufe der Jahre ständig weiterentwickelt, zum einen, weil neue Anforderungen dazu gekommen sind, zum anderen, weil die Architektur des Systems und die technologische Basis geändert wurden. Die Software wurde von Assistenten, Assistentinnen und Studierenden entwickelt. Eine frühe Version des ASB-Systems wurde in [Mor+07] beschrieben. Momentan arbeiten wir mit der Version 4, auf der dieser Beitrag basiert.

In diesem Kapitel wird das ASB-System aus technischer Sicht dargestellt, wobei wir insbesondere auf die Implementierung zur Bewertung von Android-Apps eingehen werden. Der Einsatz des ASB-Systems im Hochschulkontext ist in Kapitel 7 beschrieben. Hier haben wir auch erwähnt, dass wir ASB nicht zur vollautomatischen Bewertung, sondern lediglich als Assistenzsystem nutzen, das den Dozenten anzeigt, ob die Bewertungsmaßnahmen alle erfolgreich waren bzw. welche Probleme festgestellt wurden. Wie mit diesen Bewertungsergebnissen umgegangen wird, kann jeder Dozent für sich entscheiden.

16.2 Anforderungen

Basierend auf den Erfahrungen, die wir mit früheren Versionen des ASB-Systems gesammelt haben, und den im Laufe der Jahre aufgekommenen Wünschen der Nutzenden wurden die nachfolgend aufgeführten Anforderungen an die aktuelle Version gestellt. Diese sollten insbesondere die Modularität und Flexibilität des Systems sicherstellen:

- A1: Das System soll Programme bewerten, die in unterschiedlichen Programmiersprachen geschrieben sind.

- A2: Auch die auf die eingereichten Programme angewendeten Bewertungsmaßnahmen sollen in unterschiedlichen Programmiersprachen implementiert werden können.

- A3: Das System soll sowohl statische als auch dynamische Bewertungsmaßnahmen unterstützen.

- A4: Die im Kontext von dynamischen Bewertungsmaßnahmen ausgeführten Programme, die zur Bewertung eingereicht wurden, sollen beschränkte Zugriffsrechte besitzen. Welche konkreten Rechte solche Programme haben, soll von der jeweiligen Bewertungsmaßnahme abhängig sein. So soll es beispielsweise den eingereichten Programmen in den allermeisten Fällen nicht möglich sein, Netzverbindungen aufzubauen und darüber Daten zu senden. Für Bewertungsmaßnahmen von Programmen, die einen Client einer Client-Server-Anwendung realisieren, soll dies aber sehr wohl möglich sein. Allerdings sollen diese Programme dann nur eine Verbindung zum Testprogramm aufbauen können, welches die Serverseite nachahmt und dabei überprüft, ob sich das zu bewertende Client-Programm so verhält, wie es soll.

- A5: Bereits vorhandene Werkzeuge wie zum Beipsiel Checkstyle und Findbugs sollen als Bewertungsmaßnahmen in das System integriert werden können.

- A6: Die Bewertungsmaßnahmen sollen individuell je nach Anwendungsfall konfigurierbar sein. Damit ist zum Beispiel gemeint, dass für unterschiedliche Lehrveranstaltungen unterschiedliche Kodierungskonventionen von der Bewertungsmaßnahme Checkstyle überprüft werden.

- A7: Die Installation neuer Bewertungsmaßnahmen soll zur Laufzeit erfolgen. Ferner sollen bestehenden Bewertungsmaßnahmen zur Laufzeit verändert und gelöscht werden können.

16.3 Grundsätzliche Entwurfsentscheidungen

Wie die Vorgängerversionen ist auch das aktuelle ASB-System eine webbasierte Anwendung. Das heißt, die Nutzenden des Systems können über einen Browser auf den ASB-Server zugreifen. Die Nutzung des ASB-Systems ist von überall aus möglich. Allerdings können sich nur registrierte Personen durch Angabe einer Kennung und eines Passworts am System anmelden, wobei die Nutzenden hierfür in der Regel ihre Kennung und ihr Passwort der Hochschule Trier verwenden, deren Korrektheit bei jedem Login mit Hilfe der Server des Rechenzentrums der Hochschule Trier überprüft wird.

Um das ASB-System mit einer möglichst großen Modularität und Flexibilität auszustatten, wurde es als Komponenten-Framework mit austauschbaren Komponenten realisiert [Oec13]. Insbesondere Anforderung A7 legt eine solche Entwurfsentscheidung nahe. Das Komponenten-Framework, das wir als ASB-Kern bezeichnen, besteht im Wesentlichen aus einem selbst implementierten Ereignisbus, an den die Komponenten angeschlossen werden und über den die Komponenten miteinander kommunizieren können. Dieser Ereignisbus wurde nicht nur auf der Serverseite realisiert, sondern auf die Clients ausgedehnt. Das heißt, dass die Ereignisbusse der Clients mit dem Ereignisbus des Servers gekoppelt sind. Eine direkte Kommunikation zwischen den Ereignisbussen der Clients ist nicht möglich. Da es sich um eine webbasierte Anwendung handelt, bedeutet dies zum einen, dass die Browser nicht nur reine HTML-Seiten darstellen, sondern dass auch in den Browsern Programmcode ausgeführt werden muss. Zum anderen bedeutet eine webbasierte Anwendung, dass zur Kommunikation zwischen Client und Server das HTTP-Protokoll benutzt wird, welches ursprünglich ein reines Request-Response-Protokoll war. Damit ist gemeint, dass die Initiative für eine Kommunikation immer vom Client ausgeht und der Server nur auf Anfragen eines Clients antwortet, nicht aber von sich aus Nachrichten zu einem Client senden kann. Dies hat gewisse Nachteile, die im Kontext ASB anhand eines Beispiels erläutert werden sollen: Nach dem Hochladen eines Programms werden die Bewertungsmaßahmen angestoßen, deren Ausführung auch länger dauern kann. Da der Server das Ergebnis nach Beendigung von einer oder aller Bewertungsmaßnahmen nicht von sich aus an den Client mitteilen kann, muss der Client erst wieder eine Anfrage an den Server senden, um die Ergebnisse anzuzeigen. Dies kann

im einfachsten Fall manuell durch die Nutzerin erfolgen, indem sie immer wieder klickt um die Seite neu zu laden und nachzusehen, ob die Ergebnisse schon verfügbar sind. Es gibt auch unterschiedliche Techniken, die ohne das Zutun der Nutzenden funktionieren. Letztlich handelt es sich dabei aber immer um eine Form des Pollings, was bedeutet, dass der Client immer wieder Anfragen sendet. Durch das Konzept von sogenannten WebSockets, das es seit einigen Jahren gibt, ist es nun inzwischen möglich geworden, dass zwischen Client und Server eine länger dauernde Verbindung eingerichtet wird, über die sowohl der Server als auch der Client zu jeder Zeit von sich aus etwas senden kann. Diese WebSockets werden im ASB-System verwendet, um den Ereignisbus des Servers mit den Ereignisbussen der Clients zu koppeln. Die Grundstruktur unseres ASB-Systems ist somit ein verteiltes Ereignisbussystem, das in Abbildung 16.1 schematisch dargestellt ist. Das komplette System ist ausschließlich in Java programmiert. Clientseitig wird das Webanwendungsframework *Google Web Toolkit (GWT)*[1] verwendet. Dies ermöglicht, dass die grafische Benutzeroberfläche, die vom Browser dargestellt wird, in Java entwickelt werden kann. Ein von GWT mitgelieferter Compiler kompiliert den Java-Quellcode nach JavaScript, der von allen aktuellen Browsern interpretiert werden kann. Für die WebSocket-Kommunikation zwischen den Ereignisbussen werden *JSON*[2]-formatierte Nachrichten benutzt. Damit können Java-Objekte als Zeichenketten serialisiert werden. Auf der Empfängerseite kann die Zeichenkette wieder deserialisiert werden, was bedeutet, dass daraus wieder ein Objekt erzeugt wird, das die gleichen Werte wie das auf der Senderseite serialisierte Ob-

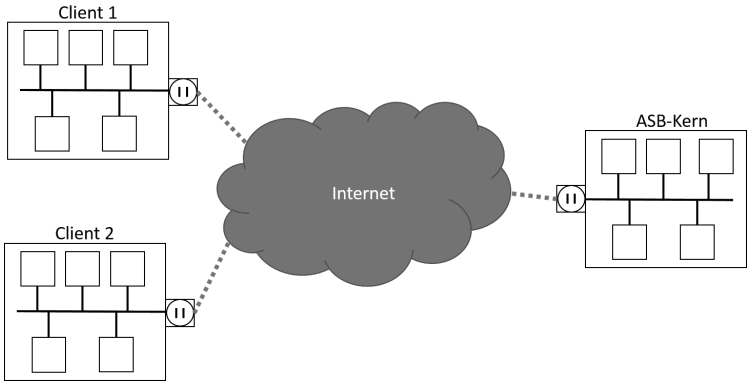

Abbildung 16.1: Client-Server-Kommunikation im ASB-System

1 http://www.gwtproject.org/?hl=de
2 http://www.json.org/json-de.html

jekt besitzt. Umgangssprachlich kann man dies so ausdrücken, dass damit Objekte über das Netz gesendet werden können.

Die Verwendung von GWT, den Ereignisbussen und den WebSockets ermöglicht sowohl die strikte Trennung zwischen Client und Server als auch die vollständige Entwicklung des Systems in der Programmiersprache Java. Somit ist es möglich, client- und serverseitig mit den gleichen Java-Objekten zu arbeiten und diese als JSON-formatierte Zeichenketten zu übertragen.

Im Folgenden gehen wir auf die Architektur des ASB-Servers ein.

16.4 Architektur des ASB-Servers

Wie zuvor erläutert wurde, handelt es sich beim ASB-System um ein Komponenten-Framework mit austauschbaren Komponenten. Das Framework wird ASB-Kern genannt. Die restlichen Teile bilden ein dreischichtiges System, das sich aus den Anforderungen ergibt:

- Schicht 1: Ausführungsumgebungen: Aufgrund der Anforderungen A1 und A2, dass sowohl die zu bewertenden Programme als auch die Bewertungsmaßnahmen in unterschiedlichen Programmiersprachen geschrieben sein können, stellt das ASB-System Ausführungsumgebungen zur Verfügung, die die Ausführung von Programmen unterschiedlicher Programmiersprachen erlauben. Auch die Beschränkung der Rechte, die die eingereichten Programme während der Ausführung haben, wird durch die Ausführungsumgebungen realisiert (Anforderung A4).

- Schicht 2: Plugins bzw. Bewertungsmaßnahmen: Alle Bewertungsmaßnahmen werden in einheitlicher Weise im System als Plugins verwaltet. Dies gilt sowohl für statische als auch für dynamische Bewertungsmaßnahmen (Anforderung A3). Das ASB-System unterscheidet nicht zwischen statischen und dynamischen Bewertungsmaßnahmen. Durch Programmierung eines kleinen Adapters können bereits existierende Werkzeuge gekapselt und als Bewertungsmaßnahmen in das ASB-System integriert werden (Anforderung A5). Bewertungsmaßnahmen können auch im laufenden Betrieb hinzugefügt, geändert und gelöscht werden (Anforderung A7).

- Schicht 3: Konfigurationen: Ein Plugin allein definiert eine Bewertungsmaßnahme in der Regel nicht vollständig. So müssen die Parameter von Kodierungskonventionen, die von der statischen Bewertungsmaßnahme Checkstyle geprüft werden, in einer Konfigurationsdatei angegeben werden. Zum

Beispiel überprüft Checkstyle, ob nach einer öffnenden Klammer korrekt eingerückt wird. Ob man mit Tabs oder Leerzeichen einrücken soll und gegebenenfalls mit vielen Leerzeichen, wird in einer Konfigurationsdatei festgelegt. Unterschiedliche Dozenten bevorzugen unterschiedliche Konventionen. In einem solchen Fall muss Checkstyle als Plugin nur ein Mal im ASB-System installiert werden. Es kann dann aber mit unterschiedlichen Konfigurationsdateien verwendet werden (Anforderung A6). Da statische und dynamische Bewertungsmaßnahmen einheitlich im System behandelt werden, wurde das Prinzip der Konfiguration auf die dynamischen Bewertungsmaßnahmen übertragen. Für das Testen von Java-Programmen ist die Bewertungsmaßnahme (Schicht 2) ein kleines Rahmenprogramm namens Unit-Test-Runner, das JUnit-Tests im Rahmen des ASB-Systems ausführt. Der konkrete Testcode wird hier als Konfiguration bezeichnet und vom Unit-Test-Runner ausgeführt.

Abbildung 16.2 zeigt den strukturellen Aufbau des ASB-Servers als Schichtensystem. Im Folgenden gehen wir näher auf den ASB-Kern (Schicht 0), die Ausfüh-

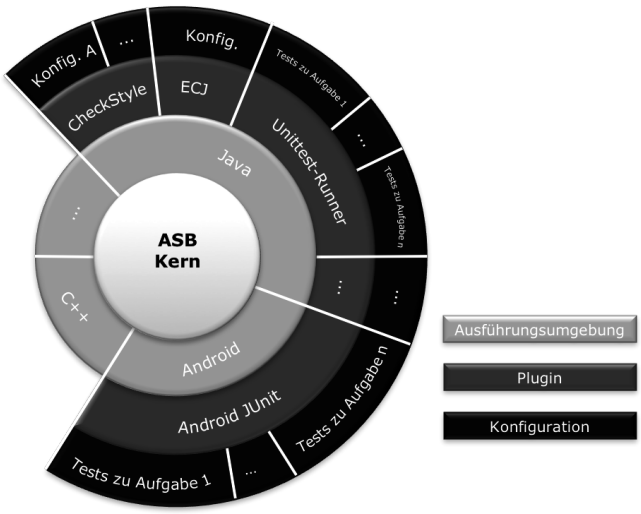

Abbildung 16.2: Struktureller Aufbau des ASB-Systems

rungsumgebungen (Schicht 1), die Plugins (Schicht 2) und die Konfigurationen (Schicht 3) ein.

Der ASB-Kern

Der *ASB-Kern* stellt die Infrastruktur des Serversystems dar. Sie wird durch einen *Ereignisbus* realisiert. Der Ereignisbus stellt Methoden bereit, um *Event*-Objekte (Ereignismeldungen) über den Bus zu senden. Diese Methoden können von den Komponenten des Serversystems benutzt werden, wobei es Methoden gibt zum Senden einer Nachricht nur auf den lokalen Bus des Servers, aber auch zum Senden auf den Serverbus mit Weitergabe an die Busse der Clients. Trifft über eine WebSocket-Verbindung eine Anfrage eines Clients in Form eines Event-Objekts auf dem Server ein, wird dieses ebenfalls über den Serverbus verteilt. Um auf den Bus gesendete Nachrichten empfangen zu können, müssen sich sogenannte *Ereignisbehandler* am Bus registrieren. Um zu entscheiden, welcher Behandler welche Nachrichtenarten verarbeitet, muss dieser eine entsprechend annotierte Methode bereitstellen, die das empfangene Ereignis als Parameter enthält. Wie schon zuvor beschrieben wurde, haben WebSockets den Vorteil, dass sie über einen sogenannten *Push*-Mechanismus verfügen. Das heißt, sobald sich eine Information serverseitig geändert hat, beispielsweise das Anlegen einer neuen Aufgabe in einer Lehrveranstaltung, ergreift der Server die Initiative und übermittelt den aktualisierten Datensatz an alle Clients, die von dem entsprechenden Ereignis betroffen sind. In diesem Beispiel beträfe das alle Studierenden, die sich für die betreffende Lehrveranstaltung im ASB-System angemeldet haben und gerade eingeloggt sind.

Der ASB-Kern enthält bereits Ereignisbehandler, um lesende und schreibende Zugriffe auf eine Datenbank durchzuführen. Die Datenbank und ihre Schnittstelle über das Bussystem bilden einen Teil des ASB-Kerns. Darüber hinaus verwaltet der ASB-Kern die im weiteren Verlauf dieses Kapitels vorgestellten Komponenten. Der Kern wurde als Komponenten-Framework implementiert, das ermöglicht, einzelne Komponenten zur Laufzeit zu installieren, zu aktualisieren und zu entfernen. Dadurch kann das ASB-System modular erweitert und angepasst werden.

Die Ausführungsumgebungen

Ausführungsumgebungen (AU) dienen dazu, Programme einer beliebigen Programmiersprache zur Ausführung zu bringen. Im Kontext des ASB-Systems bedeutet dies, dass durch eine Ausführungsumgebung ein Plugin mit der dazu gehörigen Konfiguration gestartet werden kann, um studentische Lösungen sowohl statisch als auch dynamisch bewerten zu können. Die Programme werden in einem eigenen Prozess oder auf einem anderen Rechner (siehe Kapitel 16.5) ausgeführt. Dabei haben sie minimale Ausführungsrechte. Unter anderem wird ihnen

kein Zugriff auf das Dateisystem gewährt. Über sogenannte *Umgebungsparameter* können jedoch Zugriffsrechte definiert werden. Ein Beispiel dafür sind die *Android-Permissions* für die Android-AU. Zu einer Ausführungsumgebung kann es beliebig viele Konfigurationen geben, bei denen die Werte der Umgebungsparameter je nach Kontext variieren. Programme für verteilte Anwendungen dürfen zum Beispiel Socket-Verbindungen aufbauen, wobei die erlaubten Portnummern auf einen bestimmten einstellbaren Bereich beschränkt sind. Programmen zu einer Lehrveranstaltung über grafische Benutzeroberflächen ist im Gegensatz dazu jegliche Netzkommunikation verboten. Weiterhin kann eine Ausführungsumgebung *Ausführungsparameter* verlangen. Innerhalb der Java-AU ist es beispielsweise nötig, die Einstiegsklasse des zu startenden Programms anzugeben. Die Android-AU hingegen verlangt keine Ausführungsparameter. Darüberhinaus definiert eine Ausführungsumgebung das Dateiformat, in dem zu analysierende studentische Lösungen eingereicht werden müssen.

Eine Ausführungsumgebung besitzt eine Programmierschnittstelle mit der zentralen Methode `execute()`, der als Parameter ein *File*-Objekt übergeben wird. Dieses repräsentiert einen Ordner, der alle Dateien enthält, die zur Ausführung eines Programms benötigt werden. Dazu gehören die eingereichten Dateien der studentischen Lösung, das anzuwendende Plugin und deren Konfigurationsdateien sowie zusätzlich benötigte Bibliotheken. In diesen Ordner werden nach durchgeführter Bewertung auch die Bewertungsergebnisse als XML-Datei abgelegt, die den Studierenden in aufbereiteter Form im Browser angezeigt werden. Die Ausführungsumgebung bereitet alle vorliegenden Daten auf, setzt die Parameter und bringt das Programm zur Ausführung. Jede Konfiguration eines Plugins enthält eine Maximalzeit, die die Ausführung des Plugins nicht überschreiten darf. Wird diese Zeit erreicht, wird die Ausführung abgebrochen, indem die von jeder Ausführungsumgebung bereitgestellte Methode `cancel()` automatisch aufgerufen wird. Diese Methode kümmert sich um die korrekte Terminierung der Bewertung und generiert eine entsprechende Fehlermeldung, die dem Studierenden, dessen Programm gerade bewertet wurde, als Feedback angezeigt wird.

Bisher wurden Ausführungsumgebungen für Java, Python, C++ und Android implementiert.

Die Plugins

Programme, die Bewertungsmaßnahmen auf studentischen Lösungen durchführen, werden *Plugins* genannt. Diese werden für Ausführungsumgebungen definiert und durch Konfigurationsdateien konkretisiert. Innerhalb eines Plugins wird

der Ablauf der Bewertungsmaßnahme festgelegt und benötigte Funktionalität wie beispielsweise das Schreiben der Bewertungsergebnisse gekapselt. Durch den modularen Aufbau des ASB-Systems können Plugins über die Benutzeroberfläche sogar zur Laufzeit angelegt, konfiguriert, erweitert und gelöscht werden.

Die wichtigsten Plugins der Java- und Android-Ausführungsumgebung sind:

- *Checkstyle*[3]: Das Plugin Checkstyle untersucht den an das System übertragenen Java-Quellcode auf das Einhalten von Programmierkonventionen. Die Konventionen werden mit Hilfe der dazugehörigen Konfigurationsdateien eingestellt. Dadurch können verschiedenste Prüfungen, wie beispielsweise das korrekte Klammern von Codeblöcken oder das Vorhandensein von doppeltem Code, vorgenommen werden.

- *Eclipse Compiler for Java (ECJ)*[4]: Dieses Plugin ist für die Kompilierung der Java-Dateien verantwortlich. Ein Vorteil des ECJ gegenüber des herkömmlichen Java-Compilers[5] ist die Konfigurierbarkeit. Mit Hilfe einer Konfigurationsdatei lassen sich verschiedene Fehler- bzw. Warnmeldungen an- und ausschalten. Falls beispielsweise ein Parameter einer Methode ungenutzt ist, informiert der ECJ in der aktuellen Konfiguration den Nutzer darüber. Im Gegensatz dazu würde der Java-Compiler von Oracle diese Tatsache ignorieren.

- *JMaat*: Das eigens entwickelte Plugin dient der Plagiatserkennung der eingereichten studentischen Lösungen [Mü07]. Bewertungsmaßnahmen können auf eine einzige Lösung oder auf alle Lösungen zu einer Aufgabe angewendet werden. Die Plagiatserkennung ist eine Maßnahme, die offensichtlich auf alle Lösungen angewendet werden muss. In einem solchen Fall wird die Bewertungsmaßnahme erst angestoßen, nachdem die Einreichungsfrist abgelaufen ist und alle eingereichten Lösungen in ihrer finalen Version vorliegen. Zur Plagiatserkennung wird der eingereichte Quellcode mittels eines Parsers in eine Folge von Symbolen umgewandelt. Diese Folge von Symbolen stellt eine abstrakte Programmstruktur dar. Diese Programmstrukturen werden durch Plagiaterkennungsalgorithmen bezüglich ihrer Ähnlichkeit untersucht.

- *Unit-Test-Runner*: Der Unit-Test-Runner bringt kompilierte Java-Programme zur Ausführung. Jede Konfiguration für dieses Plugin definiert für jede

3 http://checkstyle.sourceforge.net/
4 http://mvnrepository.com/artifact/org.eclipse.jdt.core.compiler/ecj/4.3.1
5 http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

einzelne Aufgabe eine Menge von Testfällen, die nach erfolgreicher Kompilierung durchgeführt und ausgewertet werden. Dazu wird unter anderem das Testframework *JUnit*[6] verwendet, welches von uns um mehrere Annotationsmöglichkeiten erweitert wurde. Mit Hilfe dieser Annotationen lässt sich der Inhalt eines automatisch generierten Testberichts steuern. Der Testbericht beschreibt u. a., welche Tests durchgeführt wurden. Mit Hilfe einer Annotation lässt sich jeder Testmethode eine Beschreibung des durchgeführten Tests zuordnen. Je nach Ausgang der Testmethode wird der Testfall dann als erfolgreich oder fehlgeschlagen im Testbericht gekennzeichnet. Für den Fall des Scheiterns eines Tests können längere Fehlermeldungen und Hinweise zur Behebung des Problems auch in Annotationen angegeben werden. Schließlich können der besseren Übersicht wegen Testfälle für den Testbericht gruppiert und ihre Reihenfolge im Testbericht festgelegt werden.

- *Android-JUnit*: Das Plugin Android-JUnit dient der Ausführung von Testfällen für Android-Applikationen. Durch dieses Plugin lassen sich Tests für die Oberflächen von Android-Apps durchführen. Android-JUnit ist momentan das einzige Plugin der Android-AU. Kapitel 16.5 geht noch ausführlicher auf das Testen von Android-Apps ein.

Die möglichen Ergebnisse jeder einzelnen Bewertungsmaßnahme sind (aufsteigend geordnet nach dem Grad der Problematik): erfolgreich, mit Warnung abgeschlossen, Fehler entdeckt oder Ausführung der Bewertungsmaßnahme gescheitert. Das Gesamtbewertungsergebnis zu einer eingereichten Lösung ist das problematischste Ergebnis aller Einzelbewertungen. Es gibt weder eine Gewichtung der Bewertungsmaßnahmen noch werden den Ergebnissen Punktezahlen zugeordnet.

Wie im Abschnitt über Ausführungsumgebungen schon erwähnt wurde, erzeugt jedes Plugin außerdem eine XML-Ergebnisdatei, in der Details zur Ausführung der Bewertungsmaßnahme zu finden sind. Wenn in ASB ein anderes Programm wie z. B. Checkstyle oder JUnit zur Bewertung benutzt wird, so ist immer ein entsprechendes Plugin nötig, das das benutzte Programm kapselt. Das Plugin führt sowohl vorbereitende Schritte durch, um die Ausführung des genutzten Programms zu ermöglichen, als auch nachbereitende Schritte, um das Ergebnis des Programms in ein ASB-eigenes XML-Format zu transformieren. Diese Transformation ist natürlich stark davon abhängig, in welcher Form das benutzte Programm seine Ergebnisse hinterlässt. Das eigentliche Checkstyle z. B. erzeugt bereits eine XML-Datei, die nach Ausführung vom Checkstyle-Plugin geparst und in das ASB-eigene XML-Format transformiert wird. Beim Unit-Test-Runner wird

6 http://junit.org/junit4/

die XML-Ergebisdatei aus den Ergebnissen der JUnit-Ausführung und den Texten der oben beschriebenen Annotationen erzeugt. In allen Fällen wird die Ergebnisdatei später aus dem ASB-eigenen XML-Format zur Anzeige im Browser in HTML gewandelt.

Die Plugin-Konfigurationen

Zu jedem Plugin kann es verschiedene *Konfigurationen* geben. Art und Umfang der Konfigurationsdateien unterscheiden sich je nach Plugin. Das Checkstyle-Plugin verlangt eine XML-Datei, in der die Codierungskonventionen definiert werden. Der *Unit-Test-Runner* und *Android-JUnit* hingegen benötigen die durchzuführenden Tests zu einer gestellten Aufgabe. Tests für Aufgaben zur Lehrveranstaltung „Grafische Benutzeroberflächen“ funktionieren zum Beispiel so: Ein studentisches Programm wird gestartet, auf der Oberfläche werden bestimmte Interaktionselemente identifiziert, die von den Studierenden mit vorgegebenen Identifikatoren gekennzeichnet sein müssen, auf diesen Elementen werden bestimmte Aktionen wie Texteingaben oder Mausklicks über eine Programmierschnittstelle ausgelöst und schließlich wird überprüft, ob bestimmte Interaktionselemente die erwarteten Beschriftungen aufweisen. Die Tests für Android-Apps arbeiten nach demselben Prinzip. Beim Testen paralleler Programme werden die studentischen Programme instrumentiert, um bestimmte zeitliche Abläufe zu erzwingen und zu prüfen, ob sich die studentischen Programme dabei so verhalten wie erwartet. Ein spezielles Prüfverfahren, um die Ausgaben parallel laufender Threads zu überprüfen, wurde in [OB07] präsentiert.

Es ist möglich, Plugin-Konfigurationen zur Vor- oder Nachbereitung für Plugins oder deren Konfigurationen anzulegen. So wird beispielsweise eine Konfiguration des Java-Compilers als Vorbereitung des Unit-Test-Runners genutzt, da der von den Studierenden eingereichte Quellcode erst übersetzt werden muss, bevor er ausgeführt werden kann. Weiterhin werden mit den Konfigurationen die maximale Laufzeit der Bewertung und die von der Ausführungsumgebung geforderten Ausführungsparameter festgelegt.

Plugin-Konfigurationen können wie Plugins ebenfalls zur Laufzeit im System angelegt werden. Eine Plugin-Konfiguration bezieht sich immer auf genau ein Plugin. Das heißt, durch die Angabe einer Konfiguration wird eine ganz konkrete Bewertungsmaßnahme definiert. Plugin-Konfigurationen können global definiert werden. Damit sind sie für jede Lehrveranstaltung nutzbar. Konfigurationen können aber auch nur für einzelne Lehrveranstaltungen angelegt werden. Damit können sie nur für die betreffende Lehrveranstaltung verwendet werden.

Lehrveranstaltungen und Aufgaben

Das ASB-System unterscheidet zwischen der Konfiguration von Bewertungsmaßnahmen und dem Administrieren von Lehrveranstaltungen. Lehrveranstaltungen und ihre Aufgaben, zu denen Lösungen hochgeladen werden können, sind der zentrale Kontext, in dem sowohl Dozenten als auch Studierende das ASB-System innerhalb eines Semesters nutzen.

Die Dozenten bzw. die sie unterstützenden Assistenten können Lehrveranstaltungen sowie dazugehörige Aufgaben anlegen und diesen die durchzuführenden Bewertungsmaßnahmen in Form von Plugin-Konfigurationen zuordnen. Bewertungsmaßnahmen können der gesamten Lehrveranstaltung oder aber nur einzelnen Aufgaben zugeordnet werden. Der Lehrveranstaltung zugeordnete Bewertungsmaßnahmen gelten automatisch für jede Aufgabe dieser Lehrveranstaltung. Dies bietet sich für Checkstyle und das Kompilieren an, denn in der Regel werden in einer Lehrveranstaltung über das gesamte Semester dieselbe Programmiersprache und dieselben Programmierkonventionen verwendet. Will man eine Plagiatserkennung immer für alle Aufgaben durchführen, so würde man die entsprechende Konfiguration auch der Lehrveranstaltung zuordnen. Plugin-Konfigurationen, die Testfälle zu einer konkreten Aufgabe beinhalten, werden direkt einer Aufgabe zugeordnet. Einmal konfigurierte Bewertungsmaßnahmen müssen nur im Falle einer Änderung angepasst werden. Gibt es keine Änderungen, so können einmal angelegte Aufgaben über mehrere Semester verwendet werden. Beim Anlegen von Aufgaben müssen die Dozenten einen Zeitraum angeben, in dem Lösungen zu dieser Aufgabe hochgeladen werden können. Lediglich dieser Zeitraum muss in jedem Semester angepasst werden.

Studierende melden sich zu den Lehrveranstaltungen im System an und haben somit Zugriff auf alle offenen Aufgaben. Zu erstellten Aufgaben können die Studierenden innerhalb des festgelegten Zeitraumes Lösungen einreichen. Beim Einreichen einer Lösung werden die Konfigurationen der anzuwendenden Bewertungsmaßnahmen ausgelesen, die oben vorgestellte Ordnerstruktur aufgebaut und der Ausführungsumgebung übergeben. Nach Beenden einer Bewertung wird die Bewertungsdatei ausgelesen, aufbereitet und dem Client zur Darstellung übermittelt.

16.5 Testen von Android-Apps

Das Testen von Android-Applikationen stellt für das ASB-System eine besondere Herausforderung dar. Da Android-Apps in Java implementiert werden, kön-

nen diese auch durch die bereits vorhandenen statischen Codeanalysen für Java-Programme (zum Beispiel Checkstyle und JMaat) evaluiert werden. Jedoch ist es nicht möglich, den Unit-Test-Runner für Android-Applikationen zu nutzen, da diese innerhalb eines Android-Betriebssystems ausgeführt werden müssen. Im Folgenden wird der Ablauf einer solchen Bewertung durch das Android-JUnit-Plugin innerhalb der Android-AU nach [Hei+15] erläutert.

Aufgrund benötigter Speicher- und Rechenressourcen werden Android-Programme nicht auf dem eigentlichen ASB-Server, sondern auf einem eigens dafür bereitgestellten Server ausgeführt. Das ASB-System fungiert hierbei als Client, der die Anfragen weiterleitet. Abbildung 16.3 veranschaulicht den Bewertungsablauf einer von einer Studierenden eingereichten Lösung.

Die Bewertung von Android-Apps verläuft wie folgt:

1. Eine Studierende lädt ihr zur Aufgabe erstelltes Android-Projekt (Quelldateien, XML-Dateien, Manifest-Datei, ...) als ZIP-Archiv auf den ASB-Server.

2. Der ASB-Server nimmt das Archiv entgegen. Die zur Bewertung der Aufgabe benötigten Testklassen werden zusammen mit dem studentischen Programmcode erneut gepackt.

3. Im nächsten Schritt schickt der ASB-Server diese Dateien mittels SSH an den Android-Server.

4. Zum Ausführen einer Applikation auf realen Android-Geräten oder -Emulatoren wird diese in einer APK-Datei verpackt. In diese Form wird nun die studentische Lösung vom Android-Server gebracht. Zum Testen von Android-Apps wird ebenfalls eine eigene Test-App in Form einer APK-Datei benötigt. Diese bleibt für alle Bewertungen dieser Aufgabe gleich. Da

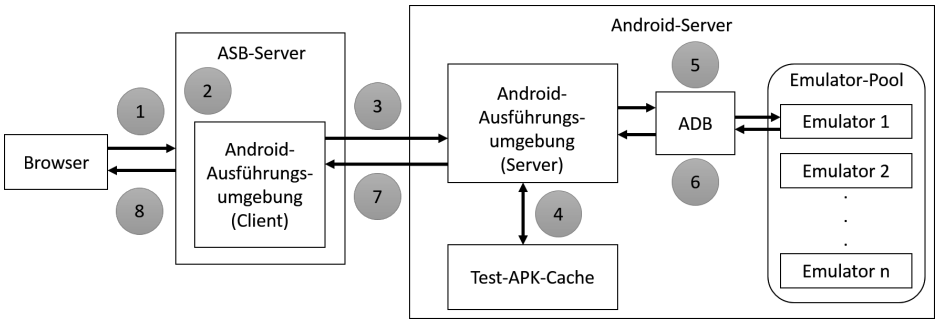

Abbildung 16.3: Ablauf

die Erstellung einer APK-Datei ressourcenintensiv ist, werden Test-APK-Dateien in einem Cache vorgehalten. Falls noch keine solche APK vorhanden ist, wird diese aus den übergebenen Testklassen erzeugt und im Cache gespeichert.

5. Android-Apps können nicht direkt auf dem Server ausgeführt werden, sondern benötigen Android-Emulatoren. Diese werden in einem Pool vorgehalten. Der Server wählt einen freien Emulator aus und übergibt diesem sowohl die studentische APK als auch die Test-APK über die *Android Debug Bridge (ADB)*. Über die ADB wird das Installieren, Starten und Testen der Applikationen veranlasst. Sollte kein Emulator verfügbar sein, wird der Auftrag in einer Auftragswarteschlange gestaut.

6. Nach Ausführung der Tests wird die erstellte Ergebnisdatei über die ADB vom Emulator geladen. Der Emulator wird heruntergefahren und ein neuer Emulator im Pool bereitgestellt. Dies garantiert gleiche Bedingungen für jeden Bewertungsvorgang und verhindert mögliche Seiteneffekte vorheriger Bewertungen.

7. Der ASB-Server empfängt nun die Ergebnisdatei vom Android-Server via SSH.

8. Dieser bereitet die Ergebnisdatei des Testlaufes so auf, dass sie der Studierenden im Browser angezeigt werden kann.

16.6 Einsatz des Systems

Das ASB-System befindet sich derzeit bei vier Modulen des Präsenz- und zwei Modulen des Fernstudiums produktiv im Einsatz. In den diesen Modulen entsprechenden Lehrveranstaltungen auf dem ASB-Server werden pro Jahr ca. 400-500 Studierende registriert, wobei bei dieser Zahl manche Studierende auch mehrfach gezählt werden können, nämlich dann, wenn sie im Laufe eines Jahres mehrere Module belegen, in denen ASB eingesetzt wird. Die Ausführungsumgebungen für C++ und Python wurden realisiert, weil entsprechender Bedarf angemeldet wurde. Tatsächlich wurden sie aber bislang nicht genutzt. Auch die Bewertungsmaßnahmen haben sich im Laufe der Zeit geändert und wurden weiterentwickelt. So hatten wir eine Zeit lang das Tool Findbugs sowie ein Tool zur Bestimmung von Softwaremetriken im Einsatz. Da die Bewertungsergebnisse aber doch nicht so sehr aussagekräftig waren, wurden diese Werkzeuge in letzter Zeit nicht mehr eingesetzt.

Auch müssen Tests für neue Aufgaben zusätzlich entwickelt und bei einer Änderung von Aufgaben angepasst werden. Als das Modul „Grafische Benutzeroberflächen“ beispielsweise von Swing auf JavaFX umgestellt wurde, waren alle bisher vorhandenen Tests nicht mehr nutzbar. Sie müssen nun für die neue Plattform JavaFX neu entwickelt werden.

16.7 Verfügbarkeit des Systems

Das ASB-System ist unter der asb.hochschule-trier.de erreichbar. Neben der Hochschulkennung ist es nach Absprache möglich, eine *ASB-Kennung* anzulegen. Dadurch ist das System auch für Nutzer außerhalb der Hochschule Trier zum Testen nutzbar. Der Quellcode des Systems kann auf Anfrage bereitgestellt werden. Zur Nutzung des Systems an einer anderen Hochschule muss die Instanz dort verwaltet werden. Ein Hosting auf einem Server der Hochschule Trier ist nicht möglich.

Danksagung

Mehrere Personen haben im Laufe der Jahre zur Entwicklung des ASB-Systems beigetragen. Bei allen möchten wir uns an dieser Stelle für ihre Arbeit bedanken, insbesondere bei Patrick Fries, Mathis Heimann, Thiemo Morth, Klaus Müller und Christian Schmal. Ohne sie wäre dieser Beitrag nicht möglich gewesen.

Literatur für dieses Kapitel

[Hei+15] Mathis Heimann u. a. „Automatische Bewertung von Android-Apps“. In: *Workshop „Automatische Bewertung von Programmieraufgaben“ (ABP 2015)*. Bd. 1496. CEUR Workshop Proceedings. 2015.

[Mor+07] Thiemo Morth u. a. „Automatische Bewertung studentischer Software“. In: *Workshop „Rechnerunterstütztes Selbststudium in der Informatik“*. Universität Siegen, 2007.

[Mü07] Klaus Müller. *Vergleich und Implementierung von Verfahren zur Plagiaterkennung von Software*. Abschlussarbeit. 2007.

[OB07] Rainer Oechsle und Kay Barzen. „Checking Automatically the Output of Concurrent Threads“. In: *11th Annual SIGCSE Conference on In-*

novation and Technology in Computer Science Education. Dundee, Scotland, 2007.

[Oec13] Rainer Oechsle. *Java-Komponenten*. München: Carl Hanser Verlag GmbH & Co. KG, 2013.