Automatische Bewertung von Android-Apps

Mathis Heimann[1], Patrick Fries[2], Britta Herres[3], Rainer Oechsle[4] und Christian Schmal[5]

Abstract: Im Fachbereich Informatik der Hochschule Trier wird seit mehreren Jahren ein System zur automatischen Software-Bewertung (ASB) eingesetzt und weiterentwickelt. Das ASB-System ist eine Web-Anwendung, die es Studierenden ermöglicht, Lösungen zu Programmieraufgaben innerhalb eines festgelegten Zeitraums einzureichen. Die Lösungen werden durch zuvor installierte Programme automatisch bewertet. In diesem Artikel beschreiben wir, wie das ASB-System erweitert wurde, damit von den Studierenden nun auch Android-Apps auf den ASB-Server geladen und automatisch bewertet werden können.

Keywords: automatische Bewertung, Java, Web-Anwendung, JUnit, Checkstyle, FindBugs, Android, Android-Emulator, Google Web Toolkit (GWT), WebSocket

1 Einleitung

Im Fachbereich Informatik der Hochschule Trier wird seit 2006 ein webbasiertes System namens ASB (Automatische Software-Bewertung) eingesetzt. Das ASB-System ermöglicht Dozenten, Programmieraufgaben zu ihren Vorlesungen zu erstellen, zu denen Studierende Lösungen hochladen können. Die von den Studierenden programmierten Lösungen werden durch mehrere Bewertungsmaßnahmen automatisch überprüft. Neben einer Überprüfung der Einhaltung von Programmierkonventionen (z.B. Einrückungen) werden die studentischen Programme ausgeführt. Dabei wird getestet, ob die Programme sich so wie vorgegeben verhalten. Die Ergebnisse der Bewertungen werden den Studierenden angezeigt. Studierende können ihre Programme daraufhin ändern und erneut zur Überprüfung einreichen, solange die Abgabefrist noch nicht abgelaufen ist.

Das System wird für die Übungen mehrerer Module sowohl im Präsenz- als auch im Fernstudium eingesetzt. Neben „Einführung in die objektorientierte Programmierung“ handelt es sich dabei u.a. um die Module „Grafische Benutzeroberflächen“, „Parallele Programmierung“ und „Entwicklung verteilter Anwendungen“. Seit Sommersemester 2013 enthält das Wahlpflichtangebot der Bachelor-Studiengänge zusätzlich das Modul „Entwicklung mobiler Anwendungen mit Android“. Zu diesem Zweck wurde das ASB-System so erweitert, dass nun auch Studierende ihre in den Übungen entwickelten Apps auf das ASB-System hochladen können und entsprechendes Feedback dazu bekommen.

[1] Hochschule Trier, Fachbereich Informatik, Postleitzahl 1826, 54208 Trier, heimannm@hochschule-trier.de
[2] Hochschule Trier, Fachbereich Informatik, Postleitzahl 1826, 54208 Trier, p.fries@hochschule-trier.de
[3] Hochschule Trier, Fachbereich Informatik, Postleitzahl 1826, 54208 Trier, herresb@hochschule-trier.de
[4] Hochschule Trier, Fachbereich Informatik, Postleitzahl 1826, 54208 Trier, oechsle@hochschule-trier.de
[5] Hochschule Trier, Fachbereich Informatik, Postleitzahl 1826, 54208 Trier, schmalc@hochschule-trier.de

In diesem Workshop-Beitrag beschreiben wir im Wesentlichen, welche Erweiterungen am ASB-System der Hochschule Trier vorgenommen wurden, damit Android-Apps automatisch überprüft werden können (Kapitel 3). Als Basis wird in Kapitel 2 zunächst das ASB-System vorgestellt. Kapitel 4 geht dann noch in aller Kürze auf die Benutzerschnittstelle ein, bevor in Kapitel 5 der Beitrag mit einer Zusammenfassung und einem Ausblick abgeschlossen wird.

2 Das ASB-System

Die Entwicklung des ASB-Systems begann im Wintersemester 2005/2006 mit einer Bachelor-Abschlussarbeit. Inzwischen wurde das System mehrfach geändert und erweitert, dabei auch komplett neu implementiert (ein relativ früher Entwicklungsstand wurde in [Mo07] präsentiert). Im Folgenden stellen wir die grundlegenden Konzepte des ASB-Systems (Stand 2015) vor.

2.1 Lehrveranstaltung, Übungsblatt und Aufgabe

Zu den grundlegenden Konzepten des ASB-Servers gehören die Begriffe Lehrveranstaltung, Übungsblatt und Aufgabe, die jeweils in einer 1:n-Beziehung zueinander stehen: Zu einer Lehrveranstaltung können mehrere Übungsblätter angelegt werden, und jedem Übungsblatt können wiederum mehrere Aufgaben zugeordnet werden. Für ein Übungsblatt kann eine Abgabefrist definiert werden. Diese Frist hat den Effekt, dass nach Überschreitung dieser Frist keine Lösungen zu den Aufgaben dieses Übungsblatts mehr eingereicht werden können. Sowohl eine Lehrveranstaltung als auch eine Aufgabe kann mit einer oder mehreren Bewertungsmaßnahmen verknüpft werden. Auf jede zu einer Aufgabe eingereichte studentische Lösung werden dann alle Bewertungsmaßnahmen angewendet, die sowohl für diese Lehrveranstaltung als auch für diese spezielle Aufgabe definiert wurden. Abbildung 1 fasst den Sachverhalt in Form eines UML-Diagramms noch einmal zusammen.

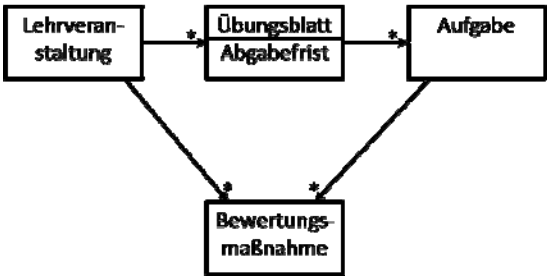

Abbildung 1: Grundbegriffe des ASB-Systems und ihre Beziehungen

Die Zuordnung von Bewertungsmaßnahmen zu Lehrveranstaltungen und Aufgaben existiert deshalb, weil wir davon ausgehen, dass gewisse Bewertungsmaßnahmen in

einer bestimmten Lehrveranstaltung immer wieder dieselben sind. Dazu gehören bei uns an der Hochschule Trier im Wesentlichen eine Syntaxprüfung durch einen Compiler mit einer gewissen Warnungsstufe, eine Überprüfung der für diese Lehrveranstaltung definierten Codierungskonventionen durch Checkstyle und das Suchen nach typischen Fehlermustern mit Hilfe von FindBugs. Aufgabenspezifische Bewertungsmaßnahmen sind Tests, die prüfen, ob die studentischen Lösungen die in dieser Aufgabe definierten Vorgaben erfüllen oder nicht.

2.2 Bewertungsmaßnahmen

Eine Bewertungsmaßnahme ist im Wesentlichen ein Programm, das auf dem ASB-Server als Bewertungs-Plugin von einem Dozenten über eine webbasierte Schnittstelle installiert werden muss. Ursprünglich war das ASB-System auf die Ausführung von Java-Programmen beschränkt. Da aber der Wunsch aufkam, auch Programme, die in anderen Programmiersprachen (C++, Python) geschrieben sind, testen zu können, wurde das Konzept der Ausführungsumgebung eingeführt. Eine Ausführungsumgebung startet ein Bewertungs-Plugin in einer bestimmten Weise (Java-Anwendungen müssen anders gestartet werden als Python-Anwendungen). Ein Bewertungs-Plugin ist immer mit einer bestimmten Ausführungsumgebung assoziiert. Bei der „naiven“ Ausführung eines Bewertungs-Plugins kann es allerdings zu Problemen kommen, z.B. dann, wenn ein Test eine Methode eines studentischen Programms ausführt und diese Methode (gewollt oder ungewollt) Schäden auf dem ASB-Server verursacht. Deshalb kann man für eine Ausführungsumgebung Rechte definieren, welche einem auszuführenden Bewertungs-Plugin eingeräumt werden. Wenn bei der Bewertung eine Aktion durchgeführt wird, ohne dass das dazu benötigte Recht vorhanden ist (z.B. ein Zugriff auf das Dateisystem), so wird die Bewertungsmaßnahme mit einem Fehler abgebrochen. Auch Ausführungsumgebungen sind Plugins des ASB-Servers, die über eine webbasierte Schnittstelle installiert werden.

Bei der Einrichtung einer Bewertungsmaßnahme für eine Lehrveranstaltung oder eine Aufgabe muss neben dem Bewertungs-Plugin noch eine Konfiguration angegeben werden, wobei der Inhalt einer Konfiguration abhängig vom konkreten Bewertungs-Plugin ist. Für das Checkstyle-Plugin z.B. enthält die Konfiguration die Codierungskonventionen, die von Checkstyle überprüft werden. Im Allgemeinen ist die Konfiguration eine Datei, die das Bewertungs-Plugin, das von der ihm zugeordneten Ausführungsumgebung aufgerufen wird, neben der studentischen Lösung als Eingabe erhält. Das Bewertungs-Plugin erzeugt eine XML-Datei in einem ASB-spezifischen Dialekt. Ferner werden alle Standard-Ausgaben und Standard-Fehlerausgaben als Ergebnisdateien dem ASB-Server zur Verfügung gestellt, der diese zur Anzeige weiterverarbeitet. Da jedes Bewertungs-Plugin eine spezielle XML-Datei erzeugen muss, ist beispielsweise das „rohe“ Checkstyle-Programm nicht das Bewertungs-Plugin, sondern das Bewertungs-Plugin besitzt zusätzlich noch ein Programmgerüst um Checkstyle herum, damit es im Rahmen des ASB-Servers verwendet werden kann (dies gilt für alle anderen Bewertungs-Plugins in gleicher Weise). Das Einhalten von Vorgaben ist bei Komponen-

ten-Software, um die es sich hier handelt, durchaus üblich; Komponenten müssen in der Regel die Vorgaben eines Kompontenten-Frameworks erfüllen, damit sie als Komponenten in dieses Frameworks installiert werden können. Eine Konfiguration ist immer mit genau einem Bewertungs-Plugin verbunden (und dieses mit genau einer Ausführungsumgebung). Deshalb genügt die Angabe einer Konfiguration, um eine Bewertungsmaßnahme zu definieren. In Abbildung 2 ist das Gesagte noch einmal zusammenfassend dargestellt. Insbesondere sollte beachtet werden, dass eine studentische Lösung durch Maßnahmen bewertet werden kann, die durchaus in unterschiedlichen Ausführungsumgebungen laufen können.

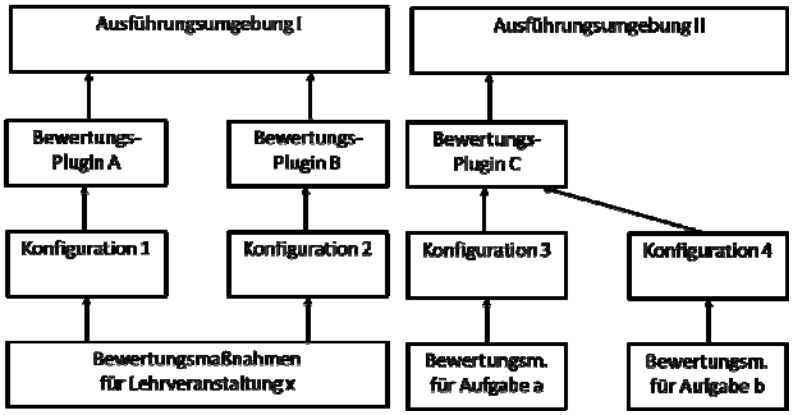

Abbildung 2: Bewertungsmaßnahmen des ASB-Systems

Bei Tests für diejenigen Lehrveranstaltungen, in denen Java verwendet wird, werden alle Bewertungsmaßnahmen mit der Java-Ausführungsumgebung durchgeführt. Dies liegt daran, dass natürlich die Tests Java-Programme sind, aber auch Checkstyle und der Java-Compiler sind selbst Java-Programme. Bei den Tests ist übrigens nicht der Test-Code das Bewertungs-Plugin, sondern das Bewertungs-Plugin ist der JUnit-TestRunner, der von uns erweitert wurde, damit er als Bewertungs-Plugin des ASB-Systems verwendet werden kann. Eine Konfiguration besteht aus den übersetzten Java-Tests, die vom Test-Runner eingelesen und ausgeführt werden. In Abbildung 2.2 könnte das Bewertungs-Plugin C beispielsweise der modifizierte JUnit-TestRunner sein. Für Aufgabe a würde man dann als Konfiguration 3 noch den aufgabenspezifischen Testcode bereitstellen, während der Testcode für Aufgabe b als Konfiguration 4 zu sehen ist.

Diese Struktur des ASB-Servers, insbesondere das Konzept der Ausführungsumgebung, ermöglichte eine problemlose Erweiterung des ASB-Systems zum Testen von Android, die im folgenden Kapitel 3 ausführlicher beschrieben wird.

3 Die Android-Ausführungsumgebung

Zum Testen von Android-Apps wurde eine Android-Ausführungsumgebung entwickelt und auf dem ASB-Server installiert. Das Bewertungs-Plugin, das in dieser neuen Ausführungsumgebung läuft, ist – ähnlich wie zuvor bei „normalen“ Java-Tests – ein Android-TestRunner, der wie der JUnit-TestRunner von uns erweitert wurde. Auch in diesem Fall sind die eigentlichen Tests wieder Konfigurationen für das Bewertungs-Plugin des Android-TestRunners. Wenn der Quellcode der Android-Apps ebenfalls mit Checkstyle und einem Java-Compiler überprüft wird, dann laufen diese Bewertungen in der Java-Ausführungsumgebung, während das Testen in der Android-Ausführungs-umgebung durchgeführt wird. Mit dieser Vorstellung wurde Abbildung 2.2 gezeichnet; die Ausführungsumgebung I könnte die für Java und die Ausführungsumgebung II die für Android sein.

Bevor wir näher auf die Android-Ausführungsumgebung eingehen, folgen zunächst einige Erläuterungen zu Android-Apps und dem Testen dieser Apps.

3.1 Android-Apps und das Testen von Apps

Eine Android-App ist eine Anwendung für Android-Geräte wie z.B. Smartphones und Tablets. Eine App muss in eine APK-Datei gepackt werden, damit sie ausgeführt werden kann. Dazu benötigt man reale Android-Geräte oder Android-Emulatoren, auf denen die App vor der Ausführung installiert werden muss. Eine APK-Datei ist eine ZIP-Datei, in der sich nicht nur der Programmcode befindet, sondern auch zusätzliche Dateien wie z.B. eine XML-Manifest-Datei, in der u.a. die einzelnen Software-Komponenten, aus denen die App besteht, deklariert sind, wobei eine Komponente als Startkomponente gekennzeichnet ist, die dann aktiv wird, wenn die App gestartet wird. Weitere Dateien in einer APK-Datei sind XML-Dateien, die das Layout und den Inhalt der Oberfläche einer Komponente festlegen, sowie Ton- und Bilddateien (z.B. für Icons oder den Fenster-hintergrund). Um also eine App auf dem ASB-Server ausführen zu können, muss man von den Studierenden verlangen, dass sie entweder direkt eine APK-Datei auf dem ASB-Server einreichen sollen oder alternativ so viele Dateien, dass eine APK-Datei daraus generiert werden kann (lediglich die Quellcode-Dateien in Form einer ZIP-Datei wie bisher für die anderen Lehrveranstaltungen hochzuladen, reicht also in diesem Fall nicht aus). Da die APK-Datei aber den Quellcode nicht mehr enthält und so keine Prüfungen wie Checkstyle durchgeführt werden können, haben wir uns entschieden, dass der komplette Eclipse-Projektordner einer App als ZIP-Datei hochgeladen werden muss. Somit kann auf dem ASB-Server sowohl der Quellcode untersucht als auch eine APK-Datei daraus generiert werden.

Das Testen einer Android-App funktioniert in der Regel so, dass eine Test-App generiert wird, die mit der zu testenden App verschmolzen wird, sodass beide Apps in einem einzigen Prozess (d.h. in einer einzigen virtuellen Maschine) ausgeführt werden. Die Test-App muss sich ebenfalls in einer APK-Datei befinden. Eclipse bietet zur Unter-

stützung hierfür an, ein Android-Testprojekt zu erzeugen, das alle nötigen Einstellungen bzgl. der Manifest-Datei schon enthält. Zur Unterstützung der Programmierung der Tests steht den Testentwicklerinnen eine Erweiterung des JUnit-Test-Frameworks für Android zur Verfügung. Wie bei JUnit gibt es auch für Android-Tests einen Testrunner, dessen Klasse in der XML-Manifest-Datei der Test-App als Instrumentation angegeben werden muss. Wie schon für die JUnit-Tests haben wir den Android-Testrunner in einer eigenen Klasse erweitert, damit zum Beispiel am Ende die Testergebnisse in eine XML-Datei geschrieben werden. In den Android-Tests, die auf dem ASB-Server ausgeführt werden, wird in den Test-Apps als Instrumentation immer unsere eigene Android-Testrunner-Klasse angegeben.

Bei der Programmierung der Tests hat man über eine Programmierschnittstelle beispielsweise die Möglichkeit, einzelne Komponenten der zu testenden App zu starten und zu beenden, sich bei einer Komponente der zu testenden App eine Referenz auf ein Oberflächenelement wie einen Button zu beschaffen, das Berühren dieses Buttons zu simulieren sowie in einem Textausgabefeld den dort vorhandenen Text auszulesen, um diesen danach mit einem Solltext zu vergleichen. Dies ist sehr ähnlich wie das Testen von „normalen“ Programmen mit grafischer Benutzeroberfläche.

Zur Ausführung eines Tests muss sowohl die zu testende App als auch die Test-App in einem Android-System (echtes Android-Gerät oder Android-Emulator) installiert werden. Dazu verwendet man ADB (Android Debug Bridge), ein Programm, das im Android Development Kit enthalten ist. Wenn man mit Eclipse entwickelt, wird ADB beim probeweisen Ausführen seiner App unbemerkt vom Entwickler verwendet. Es gibt allerdings auch eine Kommandozeilenschnittstelle, mit der über ADB entsprechende Kommandos an einen Emulator oder ein reales Gerät übermittelt werden können. Über ADB kann zum Beispiel auch auf das Dateisystem des emulierten oder realen Android-Geräts zugegriffen werden, um Dateien auf das Gerät oder vom Gerät zu übertragen. Auf diese Weise kann ADB von der Android-Ausführungsumgebung genutzt werden.

3.2 Implementierung der Android-Ausführungsumgebung

Wie oben schon erklärt wurde, soll die APK-Datei der zu testenden App auf dem ASB-Server erzeugt werden. Ferner wurde entschieden, dass aus Sicherheitsgründen ein Emulator nach einem Testlauf nicht erneut verwendet, sondern heruntergefahren und erneut im Original-Zustand hochgefahren werden soll. Sowohl das Erzeugen einer APK-Datei als auch das Starten eines Emulators ist relativ ressourcenintensiv. Dies kann insbesondere dann kritisch werden, wenn eine größere Anzahl an Studierenden gleichzeitig ihre Lösungen auf den ASB-Server laden. Deshalb wurde entschieden, dass die Android-Apps und ihre Test-Apps nicht auf dem Rechner, auf dem der ASB-Server läuft, ausgeführt werden sollen, sondern auf einem separaten Rechner. Die Android-Ausführungsumgebung auf dem ASB-Server ist somit lediglich ein schlanker Client, der die Kommunikation mit dem Android-Server übernimmt. In Abbildung 3 sind die einzelnen Instanzen gezeigt, die für die Durchführung der App-Tests verantwortlich sind. Die

eingekreisten Nummern weisen auf die einzelnen Schritte hin, die bei der Ausführung eines Tests durchgeführt werden und im Folgenden genauer beschrieben sind.

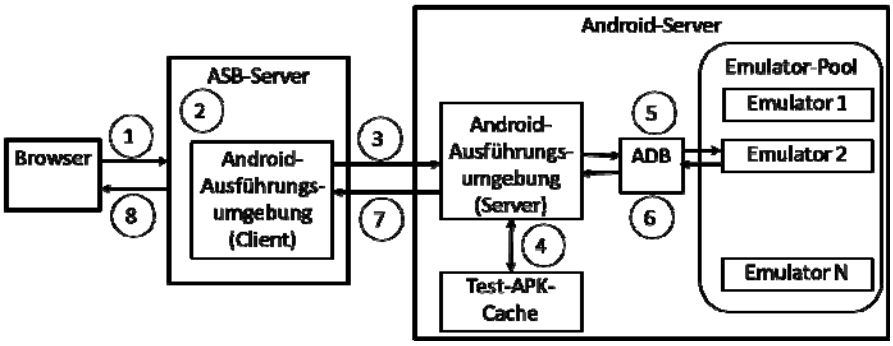

Abbildung 3: Ablauf eines Android-App-Tests auf dem ASB-Server

Die Ausführung des Tests einer Android-App läuft folgendermaßen ab:

1. Eine Studierende lädt ihre App in Form eines gezippten Eclipse-Projektordners auf den ASB-Server.
2. Der ASB-Server schaut nach, welche Bewertungsmaßnahmen für Lösungen zu dieser Aufgabe festgelegt wurden. Wir gehen davon aus, dass es eine Bewertungsmaßnahme gibt, die in der Android-Ausführungsumgebung laufen soll.
3. Die Android-Ausführungsumgebung übermittelt die studentische Lösung zusammen mit der Test-App (ebenfalls als Eclipse-Projekt) an den Android-Server.
4. Der Android-Server erzeugt aus der studentischen Lösung eine APK-Datei. Da zum Testen einer Lösung für eine Aufgabe immer dieselbe Test-App benötigt wird, und da die Erzeugung einer APK-Datei durchaus Ressourcen kostet, werden Test-APK-Dateien auf dem Android-Server in einem Cache gehalten. Beim ersten Mal wird die APK-Datei im Cache gespeichert. In den folgenden Fällen wird, falls die Test-App nicht verändert wurde, die APK-Datei aus dem Cache gelesen.
5. Da das Hochfahren eines Emulators relativ lange dauert, werden mehrere Android-Emulatoren in einem Pool betriebsbereit vorgehalten. Es wird darüber Buch geführt, welche Emulatoren benutzt werden und welche im Moment frei sind. Es wird nun ein freier Emulator ausgewählt, falls vorhanden (ansonsten wird der Auftrag in einer Auftragswarteschlange gestaut). Über ADB werden die beiden Apps (die studentische App und die Test-App) auf dem gewählten Emulator installiert und der Test wird gestartet.
6. Nach Ausführung des Tests wird die Ergebnisdatei über ADB vom Emulator heruntergeladen, der Emulator wird beendet und es wird ein neuer Emulator hochgefahren.

7. Der Android-Server sendet die Ergebnisdatei zurück an den ASB-Server.
8. Der ASB-Server erzeugt daraus eine Web-Seite, die auf dem Browser angezeigt wird.

Diese Darstellung ist stark vereinfacht. Eine ausführliche Beschreibung der Android-Ausführungsumgebung findet man in [He13].

4 Benutzerschnittstelle zum ASB-Server

Die Benutzerschnittstelle zum ASB-Server hat sich durch die Android-Ausführungsumgebung in keiner Weise geändert. Die Client-Schnittstelle wurde mit GWT (Google Web Toolkit) realisiert; sie wird also in Java programmiert und durch einen Übersetzer in JavaScript transformiert. Der Browser (d.h. der JavaScript-Code) und der ASB-Server kommunizieren über WebSockets. Damit muss der Browser bei einer länger dauernden Testausführung nicht wiederholt beim Server nachfragen, ob schon Ergebnisse vorliegen, sondern der Server kann über den WebSocket selbst die Initiative ergreifen und die Ergebnisse zum Browser übertragen, sobald sie vorliegen.

5 Zusammenfassung und Ausblick

In diesem Artikel wurde beschrieben, wie der ASB-Server, der seit mehreren Jahren an der Hochschule Trier zur automatischen Bewertung studentischer Programme eingesetzt wird, so erweitert wurde, dass er nun auch in der Lage ist, Android-Apps, die von Studierenden entwickelt und auf den ASB-Server geladen werden, automatisch zu überprüfen.

Momentan liegen noch keine größeren Erfahrungen über die Nutzung dieser Erweiterung vor. Nachdem wir aber seit mehreren Jahren überwiegend positive Erfahrungen mit dem ASB-System gemacht haben, gehen wir im Moment davon aus, dass die Studierenden es begrüßen werden, wenn sie zukünftig auch ihre Apps automatisch überprüfen lassen können.

Literaturverzeichnis

[He13] Heimann, M.: Erweiterung des Servers zur automatischen Software-Bewertung um eine Testumgebung für Android-Programme. Master-Abschlussarbeit, Hochschule Trier, Dezember 2013.

[Mo07] Morth, T.; Oechsle, R.; Schloß, H.; Schwinn, M.: Automatische Bewertung studentischer Software. Workshop „Rechnerunterstütztes Selbststudium in der Informatik“, Universität Siegen, 17. September 2007 (im Rahmen der 5. e-Learning Fachtagung Informatik DeLFI 2007, 17.-20. September, Universität Siegen).