Automatische Bewertung von Python-Anwendungen

Florian Grummel[1] Ayla Brettle[2] David Schuster[3] Rainer Oechsle[4]

Abstract: The Department of Computer Science at the University of Applied Sciences in Trier has been using and further developing a system for automatic software evaluation (ASB) for several years. The ASB system is a web application which allows students to submit solutions to programming exercises within a set time frame. The solutions can be automatically evaluated with the help of various measures. This enables a quick feedback on the correctness of a program with regard to requirements specified by a lecturer. This article describes the general structure of the ASB system and explains the automated testing and evaluation of Python applications within the ASB system using the Container Platform Docker.

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 können durch verschiedene Maßnahmen automatisch bewertet werden, so dass ein schnelles Feedback über die Korrektheit eines Programms hinsichtlich gestellter Anforderungen für die Nutzerin möglich ist. In diesem Beitrag wird der allgemeine Aufbau des ASB-Systems beschrieben und darauf aufbauend das automatisierte Testen und Bewerten von Python-Anwendungen innerhalb des ASB-Systems unter Verwendung der Container-Platform Docker erläutert.

Keywords: automatische Bewertung; Python; Web-Anwendung; Docker; Google Web Toolkit (GWT); WebSocket

1 Einleitung

An der Hochschule Trier nimmt die Programmierausbildung in den Bachelor-Studiengängen des Fachbereichs Informatik einen großen Stellenwert ein. Um bei stetig steigenden Studierendenzahlen dennoch eine individuelle Betreuung anbieten zu können, wird seit dem Jahr 2006 das System zur automatischen Softwarebewertung (ASB) eingesetzt. Es handelt sich dabei um eine Server-Client-Anwendung, die in der Programmiersprache Java implementiert wurde und Programme mithilfe von Testfällen auf die Einhaltung vorgegebener Anforderungen prüfen kann. Testfälle und Lösungen müssen hierzu allerdings nicht in

[1] Hochschule Trier, FB Informatik, Postfach 1826, 54208 Trier, Deutschland F.Grummel@hochschule-trier.de
[2] Hochschule Trier, FB Informatik, Postfach 1826, 54208 Trier, Deutschland A.Brettle@hochschule-trier.de
[3] Hochschule Trier, FB Informatik, Postfach 1826, 54208 Trier, Deutschland Da.Schuster@hochschule-trier.de
[4] Hochschule Trier, FB Informatik, Postfach 1826, 54208 Trier, Deutschland oechsle@hochschule-trier.de

Java implementiert worden sein. Da insbesondere in den Grundlagenfächern des Informatikstudiums algorithmische Problemstellungen überwiegend in der Programmiersprache Python[5] modelliert werden, bietet sich die Möglichkeit, das System um die Bewertung von Python-Programmen zu ergänzen.

Um ein Programm unter der Verwendung von Unit-Tests dynamisch testen zu können, muss dieses zur Ausführung gebracht werden. Die Ausführung von Python-Programmen gestaltet sich jedoch als Herausforderung, da diese von Studierenden angestoßen werden, die absichtlich oder unabsichtlich Schaden am System anrichten können. Dies begründet sich unter anderem durch die mangelnde Restriktierbarkeit in Python, wenn es um den Umgang mit Systemressourcen geht. Im Vergleich werden Java-Programme in einer separaten Umgebung, der „Java Virtual Machine“ ausgeführt, für welche mittels Kommandozeilenargumenten eine Sicherheitsspezifikation definiert werden kann, die beispielsweise die Zugriffsberechtigungen auf das Dateisystem oder die Freigabe und Verwendung von Netzwerkports einschränkt. In Ermangelung dieses Sicherheitskonzeptes wurde für die Ausführung nativer und interpretierter Software wie Python im Kontext des ASB-Systems bislang auf die Verwendung des Linux-Programms `Systrace`[6] zurückgegriffen. Dieses Programm dient dazu, Dateizugriffe und Portfreigaben mithilfe einer Policy-Datei, die Regeln für das auszuführende Programm definiert, einzuschränken.

Um `Systrace` verwenden zu können, musste die unter Windows-Server ausgeführte Vorgängerversion des ASB-Systems bisher auf einen separaten Linux-Server zurückgreifen, dessen Betriebssystem Debian[7] mit `Systrace` kompatibel ist. Es wurde somit sichergestellt, dass Python-Programme nicht von dem ASB-Server selbst ausgeführt werden, sodass an diesem kein Schaden angerichtet, er nicht in einen inkonsistenten Zustand gebracht oder gar heruntergefahren werden kann. Das genaue Vorgehen dieser Variante wird in Kapitel 3.1 detailliert betrachtet.

Die folgenden Kapitel geben zunächst einen Überblick über den Aufbau des ASB-Systems und beschreiben die Umstrukturierung der Ausführung von Python-Programmen. Es wird ein Konzept vorgestellt, das zur isolierten Prüfung von Lösungen keinen externen Server benötigt und auf der sicheren und ressourcenschonenden Verwendung von *Docker*-Containern basiert.

2 Das ASB-System

Das ASB-System bewertet automatisiert studentische Programme und ermöglicht den Nutzern ein individuelles Feedback zu den eingereichten Lösungen. Um nachzuvollziehen, wie die Bewertung von Python-Programmen realisiert werden kann, wird im Folgenden die Serverseite betrachtet und erläutert, wie beim ASB-System sogenannte Ausführungsumgebungen, Plugins und Lösungsbearbeitungskonfigurationen zusammenarbeiten.

[5] https://www.python.org/
[6] http://www.citi.umich.edu/u/provos/systrace/
[7] https://www.debian.org

[HOS17] beschreibt, dass das System zur automatischen Softwarebewertung als Komponentensystem realisiert wurde. Komponentensysteme sind modular aufgebaut und können zur Laufzeit erweitert werden. Der strukturelle Aufbau des ASB-Systems gleicht einem Schichtenmodell, in dessen Mittelpunkt der sogenannte ASB-Kern steht, welcher für die Infrastruktur des Systems verantwortlich ist. Der Kern betreut und verwaltet die Verbindungen zu allen Clients über asynchron agierende Websockets, kommuniziert mit der Datenbank und organisiert die Dateiablage des gesamten Systems. Des Weiteren legt der Kern den Rahmen für die Entwicklung und Ausführung von Komponenten in Form von Schnittstellen fest.

Die Komponenten werden in ASB als Ausfühungsumgebungen bezeichnet und dienen dem Zweck, die Ausführung von Prozessen vorzubereiten und anzustoßen. Allgemein stellt der ASB-Kern den Ausführungsumgebungen ein Arbeitsverzeichnis mit einer obligatorischen Ordnerstruktur zur Verfügung, in das alle zur Ausführung relevanten Daten abgelegt werden. Die Schritte zur Vorbereitung variieren in Abhängigkeit der verwendeten Programmiersprache. Im ersten Schritt werden beispielsweise alle relevanten Bibliotheken zu einem Klassenpfad zusammengetragen oder Source-Dateien kompiliert. Der ASB-Kern stellt den Ausführungsumgebungen außerdem Konfigurationsdateien im XML-Format zur Verfügung, welche unter anderem Einschränkungen definieren, die bei der Bewertung eingehalten werden müssen. Die Ausführungsumgebung muss sicherstellen, dass diese Einschränkungen bei der Ausführung der Softwarebewertung berücksichtigt werden. Die eigentliche Bewertung findet dann durch die Lösungsbearbeitungsplugins statt.

Bei Lösungsbearbeitungsplugins handelt es sich um eigenständige Programme, welche dazu dienen ein bestimmtes Merkmal einer eingereichten Lösung zu untersuchen. Zu den zu untersuchenden Merkmalen gehören neben der Ausführung von Unittests beispielsweise die generelle Kompilierbarkeit des Programms oder die Einhaltung von Code-Konventionen[8]. Plugins werden dabei stets im Kontext eines Arbeitsverzeichnisses des ASB-Systems ausgeführt. Ein Lösungsbearbeitungsplugin hat das Ziel, eine XML-Datei namens `result.xml` zu erzeugen, die eine ASB-konforme Struktur aufweist. Diese Datei spiegelt das Bewertungsergebnis wider und wird vom ASB-System verwendet, um den Studierenden dieses Ergebnis anzuzeigen. [HOS17] liefert weiterführende Informationen zum Aufbau und der Funktionsweise des ASB-Systems.

3 Die Python-Ausführungsumgebung

Während der Ausführung soll sichergestellt sein, dass studentische Python-Programme keinen uneingeschränkten Zugriff auf die Systemressourcen des Servers erhalten, um beabsichtigten oder unbeabsichtigten Schäden am ASB-Server vorzubeugen. Weiterhin muss die Möglichkeit bestehen, Systembefehle einschränken zu können, um beispielsweise

[8] Beim Testen von Java-Anwendungen wird hier auf die Software Checkstyle (s. http://checkstyle.sourceforge.net/) zurückgegriffen

das Herunterfahren des Systems zu unterbinden. Um dieses Ziel zu erreichen, wurde die Bewertung von Python-Code bislang auf einen externen Debian-Server ausgelagert. Das überarbeitete Vorgehen verwendet hingegen Docker-Container auf dem gleichen System, das auch den ASB-Server ausführt.

3.1 Bisheriges Vorgehen

Die Bewertung von Python-Code wurde bislang unter Verwendung des Programms `Systrace` durchgeführt. Im Folgenden sollen die Schritte zur Vorbereitung des zur Verfügung gestellten Arbeitsverzeichnisses beschrieben werden.

In einer ersten Phase wird das Arbeitsverzeichnis vorbereitet. Zur Vorbereitung gehört die Erstellung des Klassenpfades und die Kennzeichnung aller Verzeichnisse, welche als Python-Packages deklariert werden sollen, durch eine Python-Initialisierungsdatei[9]. Im Anschluss wird programmatisch eine ausführbare Datei namens `Main.py` erzeugt, die speziellen Pyhton-Code enthält und als Ausgangspunkt für die Lösungsbewertung dient. In diesem Python-Skript werden dem Systempfad des Programms alle zur Bewertung notwendigen Bibliotheken hinzugefügt und der Aufruf einer vorgeschriebenen Methode eines durch den Dozenten definierten Lösungsbearbeitungsplugins angestoßen.
Damit die eingereichten Programme keinen Schaden anrichten oder den Server gar herunterfahren, wird die Ausführung durch das Programm `Systrace` eingeschränkt. `Systrace` schaltet sich zwischen eine Anwendung und den darunter liegenden Linux-Kernel. Aufrufe der Anwendungen, die auf dem Kernel arbeiten, werden von dem Programm beobachtet und in Abhängigkeit definierter Richtlinien unterbunden. Dies gewährleistet, dass eingereichte Lösungen nur innerhalb festgelegter Grenzen agieren dürfen. Die zur Ausführung relevanten Regeln werden von einem Dozenten über die Benutzeroberfläche des ASB-Systems definiert und liegen zum Zeitpunkt der Ausführung in Form einer XML-Konfiguration im Arbeitsverzeichnis vor. Die Ausführungsumgebung muss in einem weiteren Schritt aus dieser XML-Datei eine Richtlinien-Datei für `Systrace` erzeugen. Dozenten können nicht über alle Einträge der Richtlinien-Datei selbst entscheiden, sondern aus Sicherheitsgründen lediglich einige ausführungsrelevante Einstellungen anpassen.

Da nun alle zur Ausführung notwendigen Dateien vorhanden sind, wird das in einem vorherigen Schritt generierte Python-Skript `Main.py` mithilfe von `Systrace` unter Verwendung der erzeugten Richtlinien-Datei zur Ausführung gebracht. Da dieses Skript das für die Bewertung zuständige Lösungsbearbeitungsplugin ausführt, liegt im Anschluss die Datei `result.xml` vor. Die Bewertung ist somit abgeschlossen und die `result.xml`-Datei kann vom ASB-System verwendet werden, um dem Studierenden das Ergebnis der Bewertung anzuzeigen.

Die ASB-Benutzeroberfläche bietet die Möglichkeit, die maximale Laufzeit eines Programms zu beschränken, sodass dieses nach einer vorgegebenen Zeit durch das System abgebrochen

[9] Diese Datei trägt den Namen __init__.py

wird. Hierzu wird eine separate SSH-Verbindung aufgebaut, um den laufenden *Systrace*-Prozess unter Verwendung seiner Prozess-ID zu beenden.

3.2 Überarbeitetes Vorgehen

Durch die Verwendung von `Systrace` konnte die mangelnden Restriktierbarkeit von Python-Programmen zwar gelöst werden, allerdings beschreibt [Wa07] Sicherheitslücken in der aktuellen Stable-Version von `Systrace`, welche eine *Privilege Escalation* ermöglichen oder die Restriktionen der Richtliniendatei umgehen können. Unter anderem aufgrund dieser Schwachstellen wurde Systrace im Jahr 2007 aus dem NetBSD-Standard 5.0[10] und im Jahr 2016 aus dem OpenBSD-Standard 6.0[11] entfernt.

Um eine sichere Umgebung zur Ausführung von Python-Programmen zu schaffen, wäre es denkbar, mehrere virtuelle Maschinen am Host-System vorzuhalten und die Bewertungen auf diese aufzuteilen. Um eine virtuelle Maschine betreiben zu können, muss für diese in der Regel Hardware simuliert werden. Ein Betriebssystem, das in einer virtuellen Maschine läuft, bringt außerdem einen eigenen Kernel mit. Die Kommunikation zwischen dem Betriebssystem der virtuellen Maschine und der Hardware des Host-Systems erfolgt dann über einen *Hypervisor*. Alle Prozesse, die innerhalb einer virtuellen Maschine ausgeführt werden, sind zwar vom Host-Betriebssystem abgeschottet, dennoch beeinträchtigt eine dauerhaft laufende virtuelle Maschine die Perfomanz des Gesamtsystems und belegt Ressourcen auch dann, wenn sich diese im Leerlauf befindet. Um ressourcenschonend arbeiten zu können, soll deshalb die Verwendung des Container-Systems Docker[12] Anwendung finden.

Im Jahr 2013 wurde die Container-Plattform Docker mit dem Ziel entwickelt, Abhängigkeitsmatrizen bei der Kompilierung und Ausführung von Software umgebungsunabhängig zu sogenannten Containern zusammenzupacken. Entstanden ist ein leichtgewichtiges Konzept, das den Vorteil einer isolierten Ausführung bietet, ohne die Perfomanz des Host-Systems dauerhaft zu beeinträchtigen. Wie in [KM16] beschrieben, eignen sich Docker-Container unter anderem zur Ausführung kurzlebiger Aufgaben und können im Gegensatz zu virtuellen Maschinen bei Bedarf mit wenig Aufwand erzeugt und gelöscht werden.

In der Community-Edition steht Docker für Windows, Mac-Systeme sowie einige Linux-Distributionen, darunter auch Ubuntu, zur Verfügung. Da es sich bei dem ASB-System um ein nicht kommerzielles Projekt handelt, darf Docker in der Community-Edition (Docker CE) kostenfrei verwendet werden.[13]

Die Docker-Plattform stellt nach [Do17] die sogenannte Docker-Engine zur Verfügung, welche als Client-Server-System implementiert zur Verwaltung von Docker-Containern dient und unter anderem die folgenden Komponenten enthält:

[10] https://www.netbsd.org/releases/formal-5/NetBSD-5.0.html
[11] https://www.openbsd.org/60.html
[12] https://www.docker.com/
[13] https://docs.docker.com/engine/faq/

Dockerfile
: Das Dockerfile dient zur Definition der von Docker ausgeführten Container. Es enthält unter anderem Angaben über Portfreigaben und Kommandos, die beim Starten ausgeführt werden sollen.

Docker-Image
: Aus einem Dockerfile erzeugt der Prozess `dockerd` ein Docker-Image. Es handelt sich dabei um eine schreibgeschützte Vorlage, mithilfe derer Docker-Container instanziiert werden können. Oftmals basieren Docker-Images auf anderen Images und erweitern diese um individuelle Anpassungen. Zur Verwaltung von Standard-Images[14] greift Docker auf ein zentrales Ablagesystem, das *Docker-Repository* zurück.

Docker-Container
: Ein Docker-Container bezeichnet eine Instanz eines Docker-Images. Auf Basis eines Images können mehrere gleiche Container erzeugt und parallel ausgeführt werden. Die Container teilen sich dabei den Kernel mit dem Host-Computer. Zustände von Containern werden nicht persistiert, sodass diese immer wieder im gleichen Zustand starten.
Es besteht die Möglichkeit, bei der Ausführung Pfade des Host-Systems zu definieren, die in den Container eingehängt werden. Es ist somit möglich, Verzeichnisse zwischen Containern und dem Host-System zu teilen.

Zur Docker-Engine gehören außerdem der *Docker-Daemon* sowie der *Docker-Client*, welche in [KM16] beschrieben werden.

Während virtuelle Maschinen ein Betriebssystem mit einem eigenen Kernel simulieren, teilen sich Docker-Container einen gemeinsamen Kernel, woraus ein geringerer Aufwand beim Erzeugen und Ausführen einer isolierten Umgebung resultiert. Docker-Images benötigen gegenüber virtuellen Maschinen außerdem weniger Speicherplatz. Das ASB-System kann bei Bedarf einen Docker-Container hochfahren und ihn nach Ausführung des Lösungsbearbeitungsplugins wieder beenden.

Durch die Verwendung von Docker wurde die Python-Ausführungsumgebung umstrukturiert, wobei einige Schritte der vorherigen Implementierung bestehen geblieben sind. Die Ausführungsumgebung erzeugt weiterhin in allen relevanten Verzeichnissen Initialisierungs-Dateien und generiert ebenfalls das Python-Skript `Main.py`, welches die Ausführung des Lösungsbearbeitungsplugins anstößt. Unmittelbar nach der Vorbereitungsphase wird auf Basis eines durch den Dozenten spezifizierten DockerFiles ein Docker-Image erstellt, aus dem dann ein Container erzeugt wird. In diesen Container wird das aktuelle Arbeitsverzeichnis unter einem obligatorischen Verzeichnisnamen eingehängt. Innerhalb des Containers kann dann direkt in dem Verzeichnis gearbeitet werden. Nach Ablauf des Python-Skriptes beendet sich der Docker-Container und die Bearbeitung ist abgeschlossen. Das eingehängte

[14] Images, welche bereits bestimmte Bibliotheken vorinstalliert haben, etwa Python oder Java

Arbeitsverzeichnis wird nach der Ausführung des Lösungsbearbeitungsplugins genutzt, um das Bewertungsergebnis dem ASB-System zur Verfügung zu stellen.

Sollte eine studentische Lösung nicht innerhalb einer definierten Zeitspanne terminieren, so beendet nach deren Ablauf die Ausführungsumgebung den zuvor angestoßenen Docker-Prozess automatisch.

Für das ASB-System wird momentan ein DockerFile definiert, welches auf dem Container `Python:3.6.1-slim` basiert und direkt nach dem Starten das durch die Ausführungsumgebung generierte Python-Skript ausführt.

4 Das ASB-System im Sicherheitskontext

[Fo06] beschreibt Kriterien, die bei der Erstellung und Verwendung von Softwarebewertungssystemen (oder auch Programming-Contest-Systems) beachtet werden sollen. Dieser Abschnitt beschreibt einige der dargestellten Schwachstellen und wie das ASB-System diesen unter Verwendung von Docker-Containern begegnen kann.

Übermäßiger Kompilieraufwand eingereichter Programme

Auch Programme mit vergleichsweise wenig Quelltext können prinzipiell eine übermäßig hohe Kompilierzeit aufweisen, wie in [Fo06] am Beispiel eines *C++*-Programms demonstriert wird.
Um zu verhindern, dass eingereichte Programme zum Kompilieren zu viel Zeit beanspruchen, werden die ausgeführten Docker-Container nach einer benutzerdefinierten Zeitspanne beendet.

Übermäßige Beanspruchung von Systemressourcen

Bedingt durch das Design einer Programmiersprache können Compiler eine unbestimmte Anzahl an Zeichen einlesen, um die syntaktische Korrektheit eines Kommandos zu bestimmen. Abhängig vom verwendeten Compiler werden sogar gesamte Dateien in den Arbeitsspeicher geladen. [Fo06] demonstriert auch hier, wie ein Compiler dazu veranlasst werden kann, so lange Daten einzulesen, bis die Gesamtkapazität des Arbeitsspeicher ausgeschöpft ist.
Das Gesamtsystem ist in diesem Zustand unter Umständen nur eingeschränkt bis gar nicht mehr handlungsfähig, sodass ein Abbrechen des Dockercontainers nicht gewährleistet werden kann.
Diesem Problem begegnet das ASB-System präventiv. Um einen Container davon abzuhalten, den gesamten Arbeitsspeicher des Host-Systems zu beanspruchen, können hierfür beim Containerstart Limitierungen definiert werden. Ein Container selbst kann dann zwar den Zustand einer *Out-Of-Memory-Situation* erreichen, beeinträchtigt hierdurch jedoch nicht die Performanz des Gesamtsystems. Container, die keinen Speicher mehr verwenden können, erhalten eine `Out-Of-Memory`-Ausnahme und

beginnen automatisch speicherintensive Prozesse zu beenden. Wird der Bewertungsprozess beendet, bevor eine Ergebnisdatei geschrieben wurde, wird dies dem Benutzer im ASB-System mitgeteilt.

Eine weitere Ressource, die limitiert werden muss, ist der durch Docker verwendete Festplattenplatz des Host-Systems. Es soll vermieden werden, dass studentische Programme Dateien erzeugen können, die den gesamten Festplattenspeicherplatz einnehmen. In Abhängigkeit des Dateisystems des Host-Rechners beschreibt [Do17] Lösungsansätze, um den Speicherplatz in Docker-Containern zu beschränken.

Mitschnitt und Manipulation des Netzwerkverkehrs

Wie in [Fo06] beschrieben, könnten die von einem Teilnehmer eingereichten Programme versuchen, den Netzwerkverkehr des Systems auszulesen, um auf diesem Wege die Programme anderer Teilnehmer einzusehen oder zu kopieren. Die Testfälle im ASB-System sind prinzipiell nicht von externen Ressourcen abhängig. Das bedeutet, dass jede Bewertung, die auf eine studentische Lösung angewendet wird, eigenständig agiert und zum Beispiel im Umfeld verteilter Anwendungen eigene Server und Clients simulieren kann. Auf dieser Grundlage besteht bislang keine Notwendigkeit, den Docker-Containern eine externe Netzwerkverbindung einzurichten, sodass im Kontext des ASB-Systems die Standardnetzwerkbrücke `none` verwendet wird. Eingereichte Programme können somit keine Verbindung zu externen Computern oder dem Host-System herstellen. In dieser geschützten Ausführungsumgebung ist es den studentischen Programmen erlaubt, alle Ports zu verwenden, ohne dass diese den gesamten Netzwerkverkehr des ASB-Systems mitschneiden oder beispielsweise *DDOS*-Angriffe[15] verüben können.

5 Zusammenfassung und Ausblick

Das ASB-System ist seit einigen Jahren erfolgreich im Einsatz und entlastet die Dozenten in der Lehre. Gleichzeitig bietet es Studierenden individuelles Feedback zur Bearbeitung ihrer Lösungen. In diesem Beitrag wurde beschrieben, wie das ASB-System grundlegend aufgebaut ist und wie Berechtigungen bei der Ausführung und Bewertung studentischer Lösungen eingeschränkt werden können. Es wurde außerdem die Container-Platform Docker vorgestellt und erläutert, welche Vorteile sie dem ASB-System im Vergleich zur Verwendung von virtuellen Maschinen bietet.

Die Python-Ausführungsumgebung wurde umstrukturiert und nutzt anstelle des Programms `Systrace` nun Docker-Container, um Berechtigungen von Python-Programmen bei der Ausführung zu kontrollieren. Das vorgestellte Konzept beschränkt sich in seiner Anwendung jedoch nicht nur auf Python-Programme, sondern soll langfristig als universelles Vorgehen

[15] Abk. Distributed Denial Of Service

auch Einzug in andere Ausführungsumgebungen erhalten. Es besteht dadurch die Möglichkeit, die Ausführung studentischer Lösungen am ASB-System einheitlich restriktieren zu können.

Literaturverzeichnis

[Do17] Docker, Inc.: , Docker overview. https://docs.docker.com/engine/docker-overview/, 2017. [Online; accessed 1-June-2017].

[Fo06] Forisek, Michal: Security of Programming Contest Systems. Informatics in Secondary Schools, Evolution and Perspectives, 2006.

[HOS17] Herres, Britta; Oechsle, Rainer; Schuster, David: Der Grader ASB. In: Automatisierte Bewertung in der Programmierausbildung. Oliver J. Bott and Peter Fricke and Uta Priss and Michael Striewe, 2017. ISBN: 978-3-8309-3606-0.

[KM16] Karl Matthias, Sean P. Kane: Docker Praxiseinstieg. mitp, Frechen, 2016.

[Wa07] Watson, Robert N. M.: Exploiting Concurrency Vulnerabilities in System Call Wrappers. WOOT 7, 2007.