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

18 Integration automatisierter Programmbewertung in Moodle

Peter Fricke und Michael Striewe

Zusammenfassung
Das OpenSource-Lernmanagementsystem MOODLE *wurde 1999 von Martin Dougiamas entwickelt [DT03]. Es wurde seither stetig erweitert und ist an vielen Hochschulen im Einsatz. Die Standardinstallation von Moodle bietet jedoch keine Möglichkeit der automatischen Programmbewertung. Dieses Kapitel gibt einen groben Überblick über die Möglichkeiten, externe Grader mit Moodle zu verbinden.*

18.1 Einleitung

Das Lernmanagementsystem Moodle ist weit verbreitet und bietet eigene Features zur Durchführung einfacher Aufgabenformate wie Multiple Choice und Fill-In an, jedoch keine Funktionen zur automatischen Bewertung von Programmieraufgaben. Sollen Lernende oder auch Lehrende in einer entsprechenden Veranstaltung, in der Moodle als Lernmanagementsystem zum Einsatz kommt, nicht dazu gezwungen werden, sich in die Bedienung eines separaten Werkzeugs für die automatische Bewertung von Programmieraufgaben einzuarbeiten, ist eine Integration solcher Werkzeuge in Moodle notwendig. Neben der Frage der Bedienung sprechen auch die Bequemlichkeit des Logins (Single Sign-On in Moodle statt separates Login im externen Werkzeug) sowie die übersichtliche Zusammenführung von Ergebnissen (alle Leistung in Moodle statt verteilt über mehrere Systeme) für eine Integration.

Moodle bietet verschiedene Schnittstellen an, über die zusätzliche Features in eine bestehende Installation integriert werden können. Abschnitt 18.2 diskutiert die Nutzung der LTI-Schnittstelle, über die externe Werkzeuge über ein standardisiertes Verfahren aufgerufen werden können. Abschnitt 18.3 befasst sich mit der Nutzung von Plugins zur Bereitstellung zusätzlicher Features.

18.2 Integration durch IMS-LTI

Der von IMS-Global veröffentlichte LTI-Standard (Learning Tools Integration) beschreibt einen generellen Mechanismus, mit dessen Hilfe eine allgemeine Lernplattform und ein spezialisiertes Lernwerkzeug auf Basis von HTTP-Anfragen Daten austauschen können [Ims]. Moodle stellt eine Implementierung der Version 1.1.1 des LTI-Standards bereit, die von Autoren genutzt werden kann, indem in den Kursraum eine Aktivität des Typs „Externes Tool“ eingebunden wird. Innerhalb dieser Aktivität muss dann eine URL eingetragen werden, unter der das einzubindende Werkzeug angesprochen werden kann. Die tatsächliche Interaktion der Lernenden geschieht dann ausschließlich mit dem externen Werkzeug. Dies bedeutet insbesondere, dass weder die Aufgaben durch Lehrende direkt in Moodle eingetragen werden, noch dass die Einreichung der Lösungen der Lernenden direkt in Moodle erfolgt.

18.2.1 Technik

Moodle implementiert durch den Aufruf der URL das minimale Verfahren, das im Standard als „Basic Launch“ bezeichnet wird. Die Rolle der Lernplattform ist darin die des sogenannten „Tool Consumer“ (TC), der die Dienste eines spezialisierten Werkzeuges nutzen möchte, indem er Aufrufe an dieses Werkzeug delegiert. Dazu werden dem aufgerufenen Werkzeug mehrere Parameter übertragen, die ihm Auskunft über den eingeloggten Benutzer, den aufrufenden Kursraum, den vom Benutzer angeklickten Link usw. geben. Es können auch zusätzliche Parameter definiert werden, die von den Autoren individuell in der Aktivität belegt werden können.

Das aufgerufene externe Werkzeug wird im Standard als Tool Provider (TP) bezeichnet und stellt dem Tool Consumer seine Dienste zur Verfügung. Es ist dabei Sache des Tool Consumers zu entscheiden, ob das Werkzeug nach einem Klick auf den Link in einem neuen Browserfenster oder eingebettet in die Lernplattform angezeigt wird. Der Tool Provider stellt in jedem Fall seine eigene Weboberfläche bereit und arbeitet mit den Daten, die ihm im Rahmen des Aufrufs übertragen wurden. Im Gegenzug kann der Tool Provider den sogenannten „Basic Outcome Service“ nutzen, der vom Tool Consumer zur Verfügung gestellt wird, um in einer einzelnen HTTP-Anfrage eine Punktzahl zurückzuliefern, die im Tool Consumer als Bewertung für die Durchführung der Aktivität verbucht wird.

Die Kommunikation zwischen Tool Provider und Tool Consumer kann über SSL-verschlüsselte Verbindungen (HTTPS) durchgeführt werden und wird in je-

dem Fall über OAuth signiert, um die nötige Sicherheit zu gewährleisten. Dabei wird im Standard davon ausgegangen, dass ein Vertrauensverhältnis zwischen Tool Consumer und Tool Provider etabliert wird. Der Tool Provider muss dem Tool Consumer dazu einen Anwendungsschlüssel und ein zugehöriges Passwort zuweisen, die der Tool Consumer nutzt, um sich beim Aufruf des Links gegenüber dem Tool Provider zu authentifizieren. Der Tool Provider nutzt dieselben Daten, um seine eigenen Anfragen zum Setzen von Bewertungen zu signieren. Dieses Vertrauensverhältnis bedeutet insbesondere, dass zwischen dem Tool Consumer und dem Tool Provider keine Passwörter der Benutzer ausgetauscht werden. Der Tool Consumer kann jedoch bei Ausführung des Links Daten wie den Benutzernamen oder eine Benutzer-ID übertragen, so dass der Tool Provider diese nutzen kann, wenn sie für die Durchführung des angebotenen Dienstes notwendig sind.

Eine exemplarische Konfiguration für die Nutzung des Graders JACK in Moodle ist in Abbildung 18.1 dargestellt. Alle o. g. Einstellungen wie die URL des Tool Providers oder die Art der Anzeige im Kurs (Einstellung „Startcontainer“) können hier vorgenommen werden.

Neben dem LTI-Standard für die Integration von Werkzeugen und Plattformen gibt es von IMS Global auch den LIS-Standard (Learning Information Services), der sich um eine systematische Beschreibung der ausgetauschten Informationen bemüht und der im Rahmen des LTI-Standards genau an dieser Stelle zum Einsatz kommt. Dazu gehört auch, dass der Tool Consumer in jedem Fall eine eindeutige ID des Aufrufs generiert, auf die sich der Tool Provider bei der Rückmeldung der Bewertung beziehen muss. Die tatsächliche Zuweisung der gesendeten Punktzahl zum Benutzer ist damit wiederum Sache des Tool Consumers und es wird somit verhindert, dass der Tool Provider beliebige Daten im Datenbestand des Tool Consumers verändern kann.

18.2.2 Vorteile

Mit dem LTI-Standard steht eine einfache Möglichkeit bereit, durch die ein externes Werkzeug wie beispielsweise ein Grader zur automatischen Bewertung von Programmieraufgaben schnell und mit sehr geringem Aufwand in Moodle integriert werden kann. Einzige Voraussetzung dazu ist, dass dieses Werkzeug die passende Schnittstelle implementiert. Insbesondere müssen keine zusätzlichen Installationen durchgeführt, sondern lediglich einige Einträge in der Konfiguration vorgenommen werden. Ferner sind keine Kompromisse oder Anpassungen bei bestehenden Aufgabensammlungen für das externe Werkzeug notwendig, da die

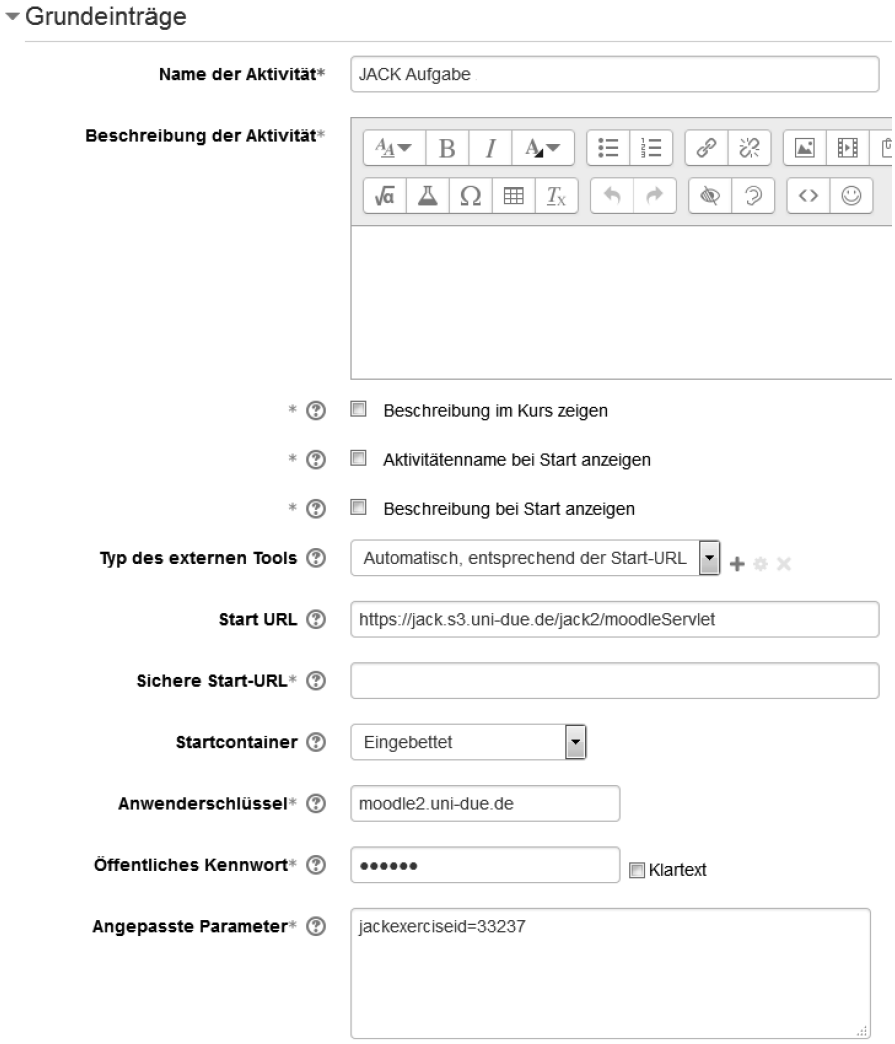

Abbildung 18.1: Beispielhafte Konfiguration einer Aufgabe vom Typ „Externes Tool“. Relevant sind insbesondere die Start-URL des externen Werkzeugs sowie der Anwenderschlüssel und das Kennwort, mit dem sich der Tool Consumer bei diesem Werkzeug anmeldet. Als angepasste Parameter können spezifische Informationen übertragen werden. In diesem Beispiel ist es die ID der Aufgabe, die innerhalb des externen Werkzeugs angezeigt werden soll.

Inhalte nicht nach Moodle importiert, sondern ausschließlich im externen Werkzeug verwaltet werden.

Auch den Entwicklern von Gradern bietet diese einfache Schnittstelle Vorteile, denn sie müssen nicht Plugins (siehe Abschnitt 18.3) für verschiedene LMS in den jeweiligen Programmiersprachen dieser Systeme entwickeln, sondern sie können mit der einmaligen Implementierung der LTI-Schnittstelle Interoperabilität mit verschiedenen LMS herstellen.

18.2.3 Nachteile

Die Möglichkeiten des „Basic Outcome Service“ zur Rückmeldung eines Ergebnisses in Moodle beschränken sich auf eine Punktzahl im Interval von 0.0 bis 1.0. Dies ist zwar ausreichend, um den erfolgreichen Abschluss einer Aktivität zu signalisieren und beispielsweise in Moodle-Lernpfaden entsprechend auf das Ergebnis zu reagieren, aber es bietet keinen Platz für weitere Informationen. Jegliches weiteres Feedback muss daher durch das externe Werkzeug und dessen Oberfläche erfolgen. Dies ist insbesondere bei Programmieraufgaben nachteilig, da die Auswertung solcher Aufgaben eine gewisse Zeitspanne in Anspruch nehmen kann. Das externe Werkzeug muss also ggf. sicherstellen, dass bei einem späteren erneuten Aufruf des entsprechenden Links in Moodle auch eine Einsicht in frühere Lösungen möglich ist. Ferner muss sichergestellt sein, dass Moodle eine späte Rückmeldung überhaupt noch entgegen nimmt, da jeder Aufruf einer externen Aktivität mit einem individuellen Token von begrenzter Gültigkeitsdauer versehen ist. Ist diese verstrichen, ist keine Rückmeldung mehr zu diesem Aufruf möglich.

Für Lehrende muss zudem eine Möglichkeit bestehen, sich am externen Werkzeug anzumelden, sofern sie dort eigene Inhalte bereitstellen oder Bewertungen einsehen wollen. Das „Basic Launch“-Verfahren von LTI ist ausschließlich für die Sicht der Lernenden gedacht und bietet somit keinerlei Unterstützung für Lehrende.

18.3 Integration durch Plugins

Moodle (ursprünglich Modular Object-Oriented Dynamic Learning Environment) ist, wie der Name schon sagt, modular aufgebaut und bietet mehrere Schnittstellen zur Integration verschiedener Arten von Plugins. Dabei ist Moodle in der Programmiersprache PHP entwickelt und bedient sich verschiedener Webtechnolo-

gien wie zum Beispiel JavaScript und Ajax und ist unter der GNU GPLv3[1] lizenziert. Somit ist eine Erweiterung und auch die Veränderung des bestehenden Codes von Moodle selbst und den angebotenen Plugins möglich.

Die Plugins können genutzt werden um die Seiten- und Kursansicht z. B. durch „Blöcke“ zu verändern, Funktionen über „Aktivitäten“ hinzuzufügen, oder die Administration durch „Admin-Tools“ zu verbessern. Die Community hat sich in den letzten Jahren stark vergrößert, so dass es eine große Auswahl dieser Plugins gibt. Für die Entwicklung eines Plugins stehen ebenfalls eine Reihe von APIs online zur Verfügung.

Möchte ein Entwickler sein Plugin der Community zur Verfügung stellen, muss er sich zudem an eine Vielzahl von Programmierrichtlinien halten, damit sein Plugin der automatischen und manuellen Validierung genügt. Erst nach erfolgreichem Bestehen wird dieses Plugin in das Moodle-Plugin-Directory aufgenommen.

Sollte eine lokale Anpassung ausreichend sein, können die Programmierrichtlinien ignoriert werden. Eine Aufnahme in das Moodle-Plugin-Directory ist somit ausgeschlossen, jedoch hat man somit alle Freiheiten ggf. sogar Änderungen am Moodle-Core vorzunehmen um seine Ziele zu erreichen.

Durch die stetige Weiterentwicklung von Moodle ist es allerdings auch nötig, dass selbst entwickelte Plugins angepasst und auf dem neuesten Stand gehalten werden, damit es keine Kompatibilitätsprobleme gibt.

18.3.1 Integration durch Anbindung von Grappa

Für die in Kapitel 23 vorgestellte Middleware Grappa wurde ein Moodle-Plugin entwickelt. Es basiert auf der Hauptaktivität „Aufgabe“[2] und wurde für die Anbindung an Grappa angepasst. Somit sind die Grundfunktionalitäten und auch die Ansicht innerhalb von Moodle den Lehrenden und Studierenden bekannt. Das Moodle-Plugin nutzt die in Kapitel 23.4.2 vorgestellte Clientbibliothek um mit Grappa zu kommunizieren.

Der Erweiterungsmechanismus wurde auch innerhalb der *Programmieraufgaben*-Plugins beibehalten, so dass je nach Unterstützung des Graders weitere Mini-Plugins für Abgaben[3] und Feedback[4] möglich sind. Derzeit wird die normale Dateiabgabe und Textfeldabgabe unterstützt. Das Feedback kann ebenfalls über die Rückmeldung des Graders hinaus erweitert werden, indem ein Kommentar-

1 https://www.gnu.org/licenses/gpl-3.0.en.html
2 https://docs.moodle.org/dev/Assignment
3 https://docs.moodle.org/dev/Assign_submission_plugins
4 https://docs.moodle.org/dev/Assign_feedback_plugins

feld oder eine Datei genutzt werden. Der Moodle-Administrator kann mehrere Grappa-Grader-Instanzen (siehe auch Abbildung 18.2) hinzufügen. Der Lehrende braucht grundlegende Kenntnisse, welche Konfigurationen und Dateien der Grader benötigt. Ebenso müssen Studierende instruiert werden, wie eine Lösung eingereicht werden muss.

18.3.1.1 Installation

Für die Installation des Moodle-Plugins sollte zunächst eine Grappa-Instanz mit mindestens einem Grader eingerichtet werden. Für die aktuellen Moodle-Versionen wurde eine neue Plugin-Struktur gewählt. Vorher bestand die Anbindung an Grappa aus zwei *Activity*-Plugins – eines für *Problems* und eines für *GrdCfgs*. Dies hatte den Nachteil, dass für eine Aufgabe immer mindestens zwei Aktivitäten angelegt werden mussten. Da Aktivitäten innerhalb des Kurses erscheinen, wurde die Übersichtlichkeit des Kurses für den Lehrenden gestört. Seit Moodle 3.0 besteht die Möglichkeit der Manipulation der Navigationsleiste, so dass die Aktivität für *GrdCfgs* mit dem Haupt-Plugin *mod_grappa* zur Verwaltung von Programmieraufgaben (*Problems*) verschmolzen wurde. Es gibt nun eine extra Übersicht für die angelegten Konfigurationsdateien (siehe Abbildung 18.3) und lediglich die Programmieraufgabe selbst ist im Kurs zu sehen.

Nach der Installation des Moodle-Plugins müssen zunächst für jeden Grader die dazugehörigen Grappa-Serverinstanzen angelegt werden. Abbildung 18.2 zeigt eine beispielhafte Konfiguration mit den Gradern *aSQLg* (Kapitel 12) und *Graja* (Kapitel 11). Es können beliebig viele Instanzen angelegt werden. Dabei müssen die unterstützten Rollen der Konfigurationsdateien (vergleiche *roles* in Kapitel 23.2.1) des Graders bzw. BackendPlugins hinterlegt werden.

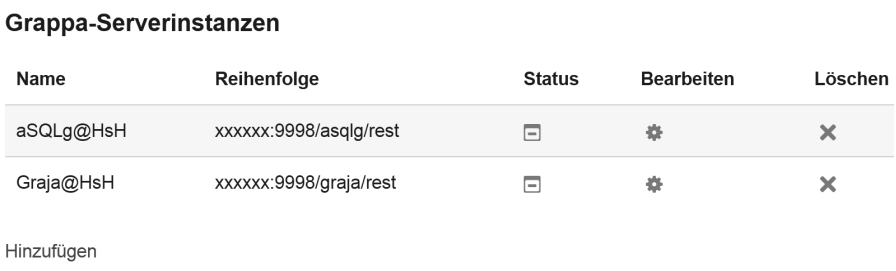

Abbildung 18.2: Übersicht der Konfigurationsseite für Grappa-Instanzen

18.3.1.2 Konfiguration

Nach erfolgreicher Installation des Moodle-Plugins und dem Anlegen von Grappa-Serverinstanzen kann der Lehrende innerhalb seiner Kurse *GrdCfgs* und *Programmieraufgaben* anlegen. Durch die Abhängigkeit von *GrdCfgs* innerhalb der *Problem*-Struktur muss der Lehrende zwingend folgende Reihenfolge beachten:

1. Lokales Erstellen/Konfigurieren von *GrdCfgs* (bspw. task.zip-Dateien, properties-Dateien, Musterlösungen).
2. Hochladen dieser *GrdCfgs* unter Angabe der Rolle und eines Namens in der *GrdCfg-Verwaltung*.
3. Erstellen der Aktivität *Programmieraufgabe* innerhalb des Kurses.

In Abbildung 18.3 ist eine beispielhafte Übersicht von *GrdCfgs*. Dabei muss zu jeder *GrdCfg* eine Grappa-Server-Instanz, ein beliebiger Name, eine Rolle und eine Datei angegeben werden. Der Grappa-Server generiert eine *GrdCfg-ID*, welche vom Moodle-Plugin hinterlegt wird. *GrdCfgs* können auf dieser Seite angelegt, geändert oder gelöscht werden und werden in Zukunft ebenfalls für Quizfragen innerhalb von Moodle genutzt. Im einfachsten Fall, wie bspw. in Graja (Kapitel 11), reicht eine im Austauschformat (siehe Abschnitt 24) vorliegende *task.zip*-Datei.

Sind alle benötigten *GrdCfgs* angelegt, kann innerhalb des Kurses die Aktivität „Programmieraufgabe“ anlegt werden. Diese gleicht in den Grundeinstellungen der normalen Moodle-„Aufgabe“. Lediglich der Unterabschnitt *Grader Ein-*

Abbildung 18.3: Übersicht der Konfigurationsseite für GrdCfgs

stellungen und *Teilaufgaben o. -aspekte* sind zusätzlich zu sehen. Lehrende können wie gewohnt Aufgabenname und Beschreibung angeben und grundlegende Aufgabenkonfiguration wie bspw. Aufgabeneröffnung, Abgabezeitpunkt, Voraussetzungen und Bewertungsrichtlinien konfigurieren. Detaillierte Informationen zu diesen Attributen einer Aufgabe sind nicht Teil dieses Artikels.

Innerhalb der zusätzlichen Abschnitte (Abbildung 18.4) muss der Lehrende zunächst die *Aufgabenwurzel* konfigurieren, indem die Informationen wie *Grader*, *Knotenschlüssel*, *GrdCfgs* und maximale Punktzahl angegeben werden müssen. Andere Einstellmöglichkeiten werden ebenfalls unterstützt und in Kapitel 23.2.2 erläutert.

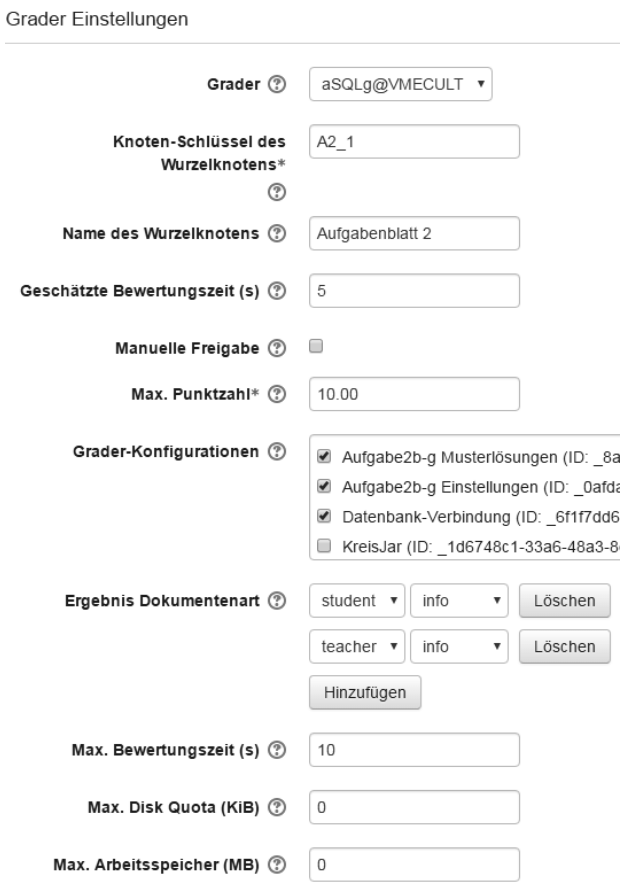

Abbildung 18.4: Ansicht der Lehrenden zum Anlegen einer Programmieraufgabe

Zusätzlich können der *Aufgabenwurzel* Teilaufgaben oder Teilaspekte in einem weiteren Abschnitt angehängt werden. Diese Einstellungen sind identisch und spiegeln die Baumstruktur eines Grappa *Problems* wider.

Erstellte Aufgaben können beliebig kopiert und in andere Kurse innerhalb von Moodle übernommen werden. Ein Im- und Export in das Austauschformat (Kapitel 24) ist in Planung.

18.3.1.3 Bewertung studentischer Abgaben

Die studentische Einreichung kann ebenfalls bei der Programmieraufgabenerstellung konfiguriert werden. Eine Auswahl zwischen Datei und Freitext ist derzeit möglich. Sobald ein Studierender seine Abgabe bestätigt hat, wird diese als Datei und unter Angabe der zu der Programmieraufgabe zugeordneten *problem-id* an Grappa übermittelt. Dabei bekommt das Moodle-Plugin eine erste Abschätzung der Dauer der Bewertung und zeigt diese Zeit als Countdown-Timer auf einem Button an. In Abbildung 18.5 ist der Abgabebildschirm eines Studierenden zu sehen. Sobald der Countdown-Timer abgelaufen ist, kann der Studierende auf „Status aktualisieren“ klicken. Im Hintergrund wurde bereits das Ergebnis von Grappa abgefragt und zur Verfügung gestellt.

Die Darstellung des Grader-Feedbacks innerhalb von Moodle wurde von der Hochschule Hannover mehrfach evaluiert [Stö+13; Stö+14; Fri+15]. Die Studierenden legten dabei großen Wert auf die Übersicht der Darstellung und die inhaltliche Verständlichkeit. Das Moodle-Plugin selbst kann auf den Inhalt des Feedbacks keinen Einfluss nehmen, dies obliegt der Rückmeldung des Graders und ggf. des *BackendPlugins*. Die Darstellung innerhalb von Moodle wurde jedoch stetig verbessert. Die mögliche Baumstruktur der von Grappa gelieferten Ergebnisdokumente (vergleiche Kapitel 23.2.3) wurde als auf- und zuklappbare DIV-Container, wie in Abbildung 18.6 umgesetzt. Dabei ist das Moodle-Plugin in der Lage die Rolle der angemeldeten Person zu unterscheiden und die dementsprechenden Informationen zur Verfügung zu stellen. Abbildung 18.6 zeigt die Sicht eines Lehrenden. Der Grader „aSQLg“ lieferte in diesem Beispiel die Musterlösung als zusätzliche Information zurück. Andere Hinweise die vom Grader selbst oder aus der Kommandozeile stammen, sind ebenso denkbar.

Der Lehrende hat zusätzlich die Möglichkeit, in die Bewertung einzugreifen oder sie manuell nach Bestätigung freizuschalten. Ein Freitextfeedback steht ebenso zur Verfügung.

18.3.1.4 Fazit

Durch die Anbindung von Grappa innerhalb von Moodle ist die Möglichkeit gegeben, eine große Menge von Gradern anzusprechen. Die Unterstützung komplexer Aufgaben- und Feedbackstrukturen ist dabei ebenfalls gegeben. Der Lehrende muss jedoch das Wissen über die Funktionsweise der einzelnen Grader besitzen und die Aufgaben dahingehend konfigurieren können.

Eine Unterstützung des Austauschformates befindet sich derzeit in Entwicklung und soll diese Hürde beseitigen, indem eine fertige Aufgabe im *task.xml*-Format einfach in den Kurs geschoben werden kann. Ebenso ist eine grafische Benutzeroberfläche für das in der ProFormA-Gruppe in eCULT+ geplante Repository denkbar. Der Austausch von Aufgaben durch Im- und Export soll Lehrende zusätzlich entlasten.

Durch die baldige Möglichkeit der Erstellung von *Fragetypen* für Grappa sind zusätzlich Prüfungen und Tests im Rahmen des summativen Assessments denkbar.

Abbildung 18.5: Ansicht der Dateiabgabe eines Studierenden

Abbildung 18.6: Darstellung des Grader-Feedbacks in Moodle im Grappa-Format aus Sicht des Lehrenden

18.3.2 Weitere Plugins im Moodle-Plugin-Directory

Das Moodle-Plugin-Directory bietet derzeit nur eine Aktivität zur automatischen Programmbewertung an. Das „Virtual Programming Lab“ (VPL) wurde 2009 von der University of Las Palmas de Gran Canaria (ULPGC) entwickelt und seither stetig vorangetrieben. Mittlerweile liegt VPL in Version 3 vor und zeichnet sich durch die Unterstützung vieler Programmiersprachen mit mehr oder weniger Funktionen aus. [Thi15] zeigt eine ausführliche Beschreibung der Funktionsweise und einer Evaluation dieser Aktivität.

VPL besteht aus zwei Teilen. Zum einen muss das Moodle-Plugin installiert werden und zum anderen der *jail-server*. Die gewünschten unterstützen Program-

miersprachen müssen auf diesem *jail-server* installiert und ggf. konfiguriert werden. Unterstützt werden laut Dokumentation bekannte Sprache wie beispielsweise C, C++, C#, Fortran, Haskell, Java, Pascal, Perl, Php, Prolog, Python und Ruby.

Auf dem *jail-server* werden die Skripte der studentischen Abgaben ausgeführt, was einen enormen Sicherheitsaspekt birgt. Schädlicher oder schlicht falscher Code von Abgaben können dem Moodle-Server keinen Schaden zufügen. Sollte durch Fehlverhalten der studentischen Lösungen der *jail-server* abstürzen, bleibt die Moodle-Instanz davon unberührt und ein Weiterarbeiten in anderen Kursen ist gewährleistet.

Die Kommunikation zwischen dem Moodle-Plugin und dem *jail-server* geschieht über XMLRPC, während Moodle-Server und Client (Browser des Studierenden oder Lehrenden) über Ajax kommunizieren. Beim Aufruf einer Konsole wird ebenfalls ein Websocket (WS/WSS) zwischen Client und *jail-server* verbunden.

18.3.2.1 Fazit

VPL ist ein mächtiges Werkzeug, welches viele verschiedene Programmiersprachen unterstützt. Die Funktionen innerhalb von Moodle und die gesicherte Anbindung an einen Jail-Server bieten einen optimalen Komfort und Sicherheit, da die Studierenden und Lehrenden mit dem Browser arbeiten können (Syntax-Highlighting) und diese Programme nicht auf dem Moodle-Server ausgeführt werden. Der Lehrende kann verschiedene Tests vorbereiten um den studentischen Code auf Input und Output zu analysieren und auch VPL-eigene Skripte erstellen um ggf. externe Tools aufzurufen. Jedoch sieht VPL den Lehrenden ebenfalls als Aufgabenersteller an. Für kleinere Programme oder Skripte kann man davon ausgehen, dass diese Anforderung vom Lehrenden umgesetzt werden kann. Größere Grading-Systeme wie bspw. Graja und JACK (siehe Kapitel 11 und 9) bieten jedoch detaillierte Einstellungsmöglichkeiten und auch dementsprechendes Feedback an. Lehrende sollten komplexere Programmieraufgaben von Programmieraufgabenautoren zur Verfügung gestellt bekommen und lediglich Gewichtungen und Punkte für Bewertungsaspekte einstellen können. So ist ein minimaler Aufwand für den Lehrenden gesichert. [Gar16] stellte diese These auf und beschrieb die Parallelen zur Softwareentwicklung. Mit dem VPL könnte man zwar die Aktivitäten innerhalb von Moodle kopieren und ggf. anderen Lehrenden zur Verfügung stellen, jedoch bedarf es für Anpassungen immer der Kenntnis des Aufgabenaufbaus, der Programmiersprache und der VPL-Syntax.

Literatur für dieses Kapitel

[DT03] Martin Dougiamas und Peter Taylor. „Moodle: Using Learning Communities to Create an Open Source Course Management System“. In: *Proceedings of EdMedia: World Conference on Educational Media and Technology 2003*. Hrsg. von David Lassner und Carmel McNaught. Honolulu, Hawaii, USA: Association for the Advancement of Computing in Education (AACE), 2003, S. 171–178. URL: https://www.learntechlib.org/p/13739.

[Fri+15] Peter Fricke u. a. „Grading mit Grappa – Ein Werkstattbericht.“ In: *Workshop „Automatische Bewertung von Programmieraufgaben“ (ABP 2015)*. Bd. 1496. CEUR Workshop Proceedings. 2015.

[Gar16] Robert Garmann. *Graja – Autobewerter für Java-Programme*. Bericht (SerWisS) 941. Hochschule Hannover, 2016. URL: http://serwiss.bib.hs-hannover.de/frontdoor/index/index/docId/941.

[Ims] *IMS Learning Tools Integration*. IMS Global Learning Consortium, 2012. URL: https://www.imsglobal.org/specs/ltiv1p1p1.

[Stö+13] Andreas Stöcker u. a. „Evaluation automatisierter Programmbewertung bei der Vermittlung der Sprachen Java und SQL mit den Gradern *aSQLg* und *Graja* aus studentischer Perspektive.“ In: *DeLFI 2013 – Die 11. E-Learning Fachtagung Informatik*. Bd. 218. LNI. GI, 2013, S. 233–238.

[Stö+14] Andreas Stöcker u. a. „Die Evaluation generischer Einbettung automatisierter Programmbewertung am Beispiel von Moodle und aSQLg.“ In: *DeLFI 2014 – Die 12. e-Learning Fachtagung Informatik*. Bd. 233. LNI. GI, 2014, S. 301–304.

[Thi15] Dominique Thiébaut. „Automatic Evaluation of Computer Programs Using Moodle’s Virtual Programming Lab (VPL) Plug-in“. In: *J. Comput. Sci. Coll.* 30.6 (Juni 2015), S. 145–151. ISSN: 1937-4771. URL: http://dl.acm.org/citation.cfm?id=2753024.2753053.