Grading mit Grappa – Ein Werkstattbericht

Peter Fricke[1], Robert Garmann[2], Felix Heine[2], Carsten Kleiner[2], Paul Reiser[2], Immanuel De Vere Peratoner[2], Sören Grzanna[2], Peter Wübbelt[3], Oliver J. Bott[3]

Abstract: „Grappa“ ist eine Middleware, die auf die Anbindung verschiedener Autobewerter an verschiedene E-Learning-Frontends respektive Lernmanagementsysteme (LMS) spezialisiert ist. Ein Prototyp befindet sich seit mehreren Semestern an der Hochschule Hannover mit dem LMS „moodle“ und dem Backend „aSQLg“ im Einsatz und wird regelmäßig evaluiert. Dieser Beitrag stellt den aktuellen Entwicklungsstand von Grappa nach diversen Neu- und Weiterentwicklungen vor. Nach einem Bericht über zuletzt gesammelte Erfahrungen mit der genannten Kombination von Systemen stellen wir wesentliche Neuerungen der moodle-Plugins, welche der Steuerung von Grappa aus moodle heraus dienen, vor. Anschließend stellen wir eine Erweiterung der bisherigen Architektur in Form eines neu entwickelten Grappa-php-Clients zur effizienteren Anbindung von LMS vor. Weiterhin berichten wir über die Anbindung eines weiteren Autobewerters „Graja“ für Programmieraufgaben in Java. Der Bericht zeigt, dass bereits wichtige Schritte für eine einheitliche Darstellung automatisierter Programmbewertung in LMS mit unterschiedlichen Autobewertern für die Studierenden absolviert sind. Die praktischen Erfahrungen zeigen aber auch, dass sowohl bei jeder der Systemkomponenten individuell, wie auch in deren Zusammenspiel via Grappa noch weitere Entwicklungsarbeiten erforderlich sind, um die Akzeptanz und Nutzung bei Studierenden sowie Lehrenden weiter zu steigern.

Keywords: E-Assessment, Programmieraufgabe, Middleware, Grader, Plugin

1 Einleitung

„Grappa“[4] ist eine Middleware, die auf die Anbindung verschiedener Autobewerter (*Grader*) an verschiedene Lernmanagementsysteme (*LMS*) spezialisiert ist ([GHW15]). Die gesamte Kommunikation zwischen dem Frontend und dem Grader wird über Grappa abgewickelt, von der initialen Konfiguration der Aufgabenstellung durch die Lehrkraft bis hin zur Abgabe von Aufgabenlösungen durch die Studierenden. Der Ablauf einer Aufgabenbereitstellung sieht so aus, dass die Lehrkraft Grappa zunächst mit Grader-spezifischen Konfigurationsdaten initialisiert, die u. a. auch die Musterlösung für die Aufgabenstellung

[1] Hochschule Hannover, ZSW – E-Learning Center, Expo Plaza 4, 30539 Hannover, peter.fricke@hs-hannover.de
[2] Hochschule Hannover, Fakultät IV Wirtschaft und Informatik, Ricklinger Stadtweg 120, 30459 Hannover, (robert.garmann|felix.heine|carsten.kleiner)@hs-hannover.de bzw. (paul.reiser|immanuel.de-vere-peratoner|soeren.grzanna)@stud.hs-hannover.de
[3] Hochschule Hannover, Fakultät III – Medien, Information und Design, Expo Plaza 12, 30459 Hannover, (peter.wuebbelt|oliver.bott)@hs-hannover.de
[4] Der Begriff entstand aus dem Kofferwort Grapper (*Gra*der Wra*pper*), welches anschließend mit der Intention, Genuss zu assoziieren, abgewandelt wurde.

enthalten können. Anschließend kann die eigentliche Aufgabe mit Referenzen auf die zuvor spezifizierten Konfigurationsdaten angelegt werden. Sämtliche aufgaben- und konfigurationsspezifischen Daten werden aus Performancegründen von Grappa persistiert und stehen somit für spätere Abfragen zur Verfügung. Grappa arbeitet über das Frontend eingehende Aufgabenlösungen asynchron ab. Dabei kann eine beliebige Anzahl von Abgaben in eine Warteschlange eingereiht werden. Für eine ausführliche Besprechung Grappa-verwandter Ansätze verweisen wir auf [GHW15].

Ein Prototyp befindet sich mit dem Frontend „moodle“ und dem Backend „aSQLg“ seit mehreren Semestern im Einsatz und wurde z.B. in [SBG+14] evaluiert. Abb. 1 zeigt Grappa mit einigen angebundenen Softwarekomponenten. Nach den in Abschnitt 2 beschriebenen praktischen Einsatzerfahrungen werden wir in Abschnitt 3 die Entwicklungen im Detail beschreiben, die in den grau hervor gehobenen Komponenten erfolgten. Abschnitt 4 stellt geplante Weiterentwicklungen dar.

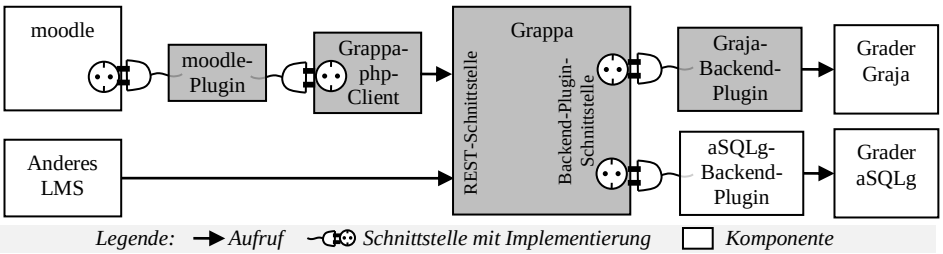


Abb. 1: Architektur

2 Einsatzszenarien

Dieser Abschnitt beschreibt verschiedene Szenarien, die an der Hochschule Hannover durchgeführt und mit moodle und aSQLg zum Einsatz kamen (Überblick in Tab. 1).

Nr.	Studiengang	Termin	Lehrveranstaltung	Teilnehmer/-innen	Anzahl Aufgaben	Bemerkungen
i.	Informations-management	WS 14/15	Relationale Datenbanken	88, davon 25 berufsbegl.	61	freiwillig, übungsbegleitend, max. 10% Klausurbonus
ii.	Medizin. Informations-management	WS 14/15	Datenbanken I (DB1)	44 evaluierende	10	freiwillig, veranstaltungsbegleitend
iii.		SS 15	Datenbanken II (DB2)	39 evaluierende	7	freiwillige Probeklausur am Abschluss der Veranstaltung
iv.	Angewandte Informatik	SS 15	Informationssysteme I (IS1)	80, davon 53 registriert	26	SQL-Vertiefung, freiwillig, übungsbegleitend

Tab. 1: Einsatzszenarien in Bachelor-Studiengängen der Hochschule Hannover

Im Einsatzszenario i. stand der den Aufgaben zugrundeliegende Datenbestand zusätzlich als SQL-Skript zur Verfügung, mit dem eine lokale Datenbank eingerichtet werden konnte. Wegen des in den Jahren zuvor eher umständlichen Uploads der Lösungsvorschläge in andere Bewertungssysteme wurde empfohlen, die Aufgaben vor einer finalen Abgabe zunächst lokal zu testen. Die Evaluation, an der 53 Studierende teilnahmen, zeigte allerdings, dass die Aufgaben überwiegend direkt über das moodle-Interface bearbeitet wurden und Handhabung und Arbeitsgeschwindigkeit dabei mehrheitlich als sehr gut bis gut bewertet wurden. Knapp mehr als 80% der Teilnehmer sprach sich zwar dafür aus, die automatisierte Programmbewertung auch in anderen Fächern benutzen zu wollen, hinsichtlich Feedback zur Aufgabenlösung, leichterer Erkennbarkeit der Fehler im SQL-Code sowie Übersichtlichkeit der Ergebnisdarstellung, zeigte die Evaluation jedoch auch deutliches Entwicklungspotential auf.

Mit den jährlich wiederkehrenden Einsatzszenarien ii. und iii. bestehen bereits Erfahrungen seit dem WS 12/13 (ii.) bzw. seit dem SS 14 (iii.). Im Einsatzszenario ii. haben 77% der Studierenden das freiwillig nutzbare Angebot in Anspruch genommen. Die Übersichtlichkeit der Oberfläche sowie die Qualität des Feedbacks wurde mit gut bewertet (jeweils Schulnote 2,1 bei Noten von 1-5), die Geschwindigkeit mit gut bis sehr gut (1,6). Das Hochladen der Lösungsdateien wurde von einigen Studierenden als schwierig empfunden. Zeichensatzfehler führten zu Irritationen, da Fehlermeldungen dadurch partiell nicht nachvollziehbar waren. Das Feedback des Programms wurde von den Studierenden zum Teil als unklar oder ungenau bewertet. Im Einsatzszenario iii. konzentrierte sich die Nutzung des Programms auf wenige Tage und insbesondere auf das Wochenende vor der Klausur. Lösungen zu den einzelnen Aufgaben konnten erstmalig als Freitexte direkt eingegeben werden (siehe Abschnitt 3.1) und mussten nicht zusammengefasst als Lösungsdatei abgegeben werden. Insgesamt nutzten 31% der Studierenden das freiwillig nutzbare Angebot. Die Übersichtlichkeit wurde ebenso wie die Qualität des Feedbacks und die Geschwindigkeit der Rückmeldung mit Note 2,4 deutlich schlechter als im WS 14/15 bewertet. Als Gründe für diese unerwartete Entwicklung wurde unter anderem der Zeitmangel während der Vorbereitungszeit genannt. Die Studierenden, die das System nutzen konnten, gaben in den Freitextrückmeldungen zum Teil positive Rückmeldungen, zum Teil wünschten sie sich detailliertere Informationen zu den eigenen Fehlern.

Im Einsatzszenario iv. haben sich 53 Studierende für die automatisierte Bewertung registriert; von diesen wiederum haben 39 mindestens eine Aufgabe zur Bewertung hochgeladen. Die prinzipielle Bereitschaft, die automatisierte Bewertung zu nutzen, kann also als gut angesehen werden. Auch die Bearbeitung der Übungsaufgaben lag damit anteilsmäßig höher als bei den danach folgenden Aufgabenblättern. Allerdings wurde die Zahl der Abgaben immer dann deutlich geringer, wenn das sich noch im Teststadium befindliche Bewertungssystem Probleme zeigte. Dies galt, auch wenn der Studierende selbst von diesen Problemen gar nicht betroffen war. Es zeigte sich ferner, dass die Aufbereitung der Ergebnisse der automatischen Bewertung große Sorgfalt erfordert, insbesondere wenn mehrere Teilaufgaben zu einer gemeinsam zu bewertenden Aufgabe zusammengefasst wurden. Die Studierenden verlieren hier leicht den Überblick über die Antworten bzw. Ergebnisse, da die Darstellung in Spalten zu vielen Zeilenumbrüchen führt. Schließlich zeigte sich, dass

die Dauer, bis eine Bewertung vorliegt, eine kritische Größe ist. Bei der hier verwendeten größeren Datenbank und komplexeren Anfragen ist eine Wartezeit nicht zu vermeiden. Die asynchrone Verarbeitung der Bewertungen über Grappa ist in diesem Bereich unverzichtbar. Dennoch sollte die zu erwartende Zeit, bis ein Ergebnis vorliegt, möglichst korrekt und leicht erkennbar für die Studierenden dargestellt werden. Bei den hier verwendeten länger laufenden Anfragen ist eine Einreichung per Dateiupload zu bevorzugen, da die Studierenden zumeist mehrere Iterationen benötigen, bevor sie eine lauffähige oder sogar korrekte Anweisung erstellt haben. Hierfür bietet sich die direkte Nutzung der spezialisierten Werkzeuge zur Erstellung von Anfragen (bspw. Oracle SQL Developer) eher an, als eine Bereitstellung derselben Funktionalität im LMS.

Zusammenfassend zeigen die Erfahrungen des Einsatzes mit moodle und automatisierter Programmbewertung (hier: aSQLg), dass die Handhabung so einfach wie möglich gehalten werden muss. Der nutzenden Person müssen etwaige Fehler verständlich und eindeutig angezeigt werden können. Ebenso sollte eine Alternative zum Dateiupload für die Abgabe von Studierenden möglich sein. Die Darstellung des Feedbacks und die Unterstützung bei der Analyse eigener Fehler sind ebenfalls optimierbar.

3 Entwicklungsfortschritt

Dieser Abschnitt beschreibt den aktuellen Entwicklungsfortschritt der einzelnen Teilkomponenten. Die Gliederung gleicht Abb.1, lesend von links nach rechts. Beginnend beim LMS moodle und dem entwickelten moodle-Plugin, werden in Abschnitt 3.1 die umgesetzten Neuerungen vorgestellt, die u.a. aus den gewonnenen Erkenntnissen der bisherigen Evaluationen hervorgegangen sind. Anschließend folgt eine kurze Erläuterung der neuen Systemkomponente Grappa-php-Client in Abschnitt 3.2. Entwicklungsarbeiten der Middleware Grappa werden in Abschnitt 3.3 vorgestellt, um abschließend den angebundenen Autobewerter Graja und das dazugehörige Backend-Plugin in Abschnitt 3.4 zu beschreiben.

3.1 moodle-Plugin

Die zwei für das LMS moodle entwickelten Grappa-spezifischen Plugins dienen 1) der Verwaltung von Grader-Konfigurationsdateien, 2) der Administration von Programmieraufgaben und 3) der automatischen Bewertung studentischer Abgaben. Die aus bisherigen Einsätzen (vgl. Abschnitt 2) gewonnenen Erkenntnisse fließen in stetige Verbesserungen ein. Um die Konfiguration einer Aufgabe zu beschleunigen, wurden Standardwerte für die Einstellung der Informationsfülle und der Zielgruppe von Feedbackmeldungen realisiert. Die Lehrkraft hat die Möglichkeit, den maximalen Zeit- und Speicherplatzverbrauch pro (Teil-)Aufgabe einzustellen und bei Bedarf die „Manuelle Freigabe“ zu aktivieren. Diese verlangt von der Lehrkraft eine manuelle Bestätigung der von Grappa gelieferten Bewertung und des dazugehörigen Feedbacks, bevor sie für den Studierenden sichtbar werden.

Die Darstellung der Bewertung wurde vollständig überarbeitet inklusive der unterschiedlichen Informationsansichten für Lehrende (Abb. 2, rechts) und Studierende. Zusätzliche Informationen des Graders zu einer Abgabe, die bisher nur in der Logdatei einsehbar waren, werden nun für Lehrende zusammen mit dem Feedback angezeigt. Schließlich wurde gemäß Anforderung sichergestellt, dass die (zuvor bereits vorhandene) Möglichkeit, die Abgaben in ein Freitextfeld direkt einzugeben, genutzt werden kann. Abb. 2 zeigt links den Statusbereich für eine Abgabe mit Freitextfeld. Er enthält u. a. den Status der Bewertung und den Zeitpunkt, zu dem dieser Status (im Hintergrund) zuletzt abgefragt wurde. Über den Button „Status aktualisieren" wird die Seite neu geladen. In den Klammern dahinter steht die Zeit in Sekunden, die verbleibt, bis der Status der Bewertung (im Hintergrund) erneut abgefragt wird.

Abb. 2: Statusbereich zu einer Abgabe (links) und Ausschnitt d. Feedbacks einer automatisierten Bewertung für Lehrkräfte (rechts, aus Platzgründen unten abgeschnitten)

Sobald das Ergebnis der Bewertung verfügbar ist und ggf. vom Lehrenden manuell bestätigt wurde, wird es auf der Abgabeseite angezeigt. Details zu den bewerteten Aufgabenteilen und Bewertungsaspekten lassen sich über „Plus/Minus"-Symbole einzeln oder über den Link „Alle Ergebnisse aufklappen" komplett ein- und ausblenden (siehe Abb.2, rechts und Abb. 3, rechts). Zeichensatzfehler der Texte werden durch die einheitliche Kommunikation und Formatierung der Nachrichten im UTF-8 Format abgefangen.

3.2 Grappa-php-Client

Bei der Anbindung des in Abschnitt 3.1 beschriebenen moodle-Plugins offenbarten sich Funktionen, die jedes LMS-Plugin implementieren muss, um mit Grappa zu kommunizieren. Dazu gehören die REST-konforme Übertragung von XML-Dateien, die Polling-Funktionalität und der dazugehörige Ablauf. Es wurde entschieden, diese Funktionalitäten

in eine php-Client-Bibliothek abzulegen. So konnten die erforderlichen Entwicklungsarbeiten für php-basierte LMS-Plugins deutlich vereinfacht werden. Die konkrete Aufgabe des Clients ist die Bereitstellung von Grappa-spezifischen php-Objekten und deren Übersetzung in entsprechende XML-Daten und umgekehrt. Ebenso werden wiederkehrende Abläufe und die Kommunikation mit Grappa in Form von php-Methoden bereitgestellt. Somit kann sich die Entwicklung des LMS-Plugins auf die Integration in das LMS und die Benutzeroberfläche konzentrieren. Der Grappa-php-Client wird seit dem WS14/15 erfolgreich von dem in Abschnitt 3.1 beschriebenen moodle-Plugin verwendet.

3.3 Grappa

Die Middleware Grappa musste ebenfalls erweitert werden, um die in Abschnitt 2 gewonnenen Erkenntnisse umzusetzen - die asynchrone Verarbeitung und die Bereitstellung einer möglichst präzisen Wartezeit. Der asynchrone Bewertungsvorgang wurde stabilisiert und um detailliertere bzw. strukturiertere Fehlermeldungen erweitert. Um eine präzise Wartezeit für die Dauer der Bewertung zu schätzen, bildet Grappa basierend auf den Laufzeiten bisheriger Bewertungen von typähnlichen Aufgabenstellungen den gleitenden Durchschnitt der Bewertungszeit einer Aufgabe und errechnet die verbleibende Restzeit, um sie auf Anfrage an das LMS zurückmelden zu können. Überschreitet die Laufzeit eines Bewertungsprozesses einen vorkonfigurierten Maximalwert, so wird der Prozess abgebrochen. Die Überwachung (und Durchsetzung) einer Maximallaufzeit ist eine von mehreren Möglichkeiten, den Betriebsmittelverbrauch eines Graders zu steuern. Bewertungsprozesse werden von Grappa persistiert. Ist der Bewertungsprozess abgeschlossen, wird das Ergebnis für eine in Grappa konfigurierbare Zeit vorgehalten und anschließend automatisch gelöscht. Bewertungsergebnisse enthalten u. a. die erreichte Gesamtpunktzahl und Ergebniskommentare. Grappa unterstützt für die Rückmeldung der Bewertungsergebnisse unterschiedliche Formate (HTML, Text, u.a.), die das LMS über die Schnittstelle abfragen kann. Aufgabe des Grader(-Plugin)s ist es, das gewünschte Format zu liefern.

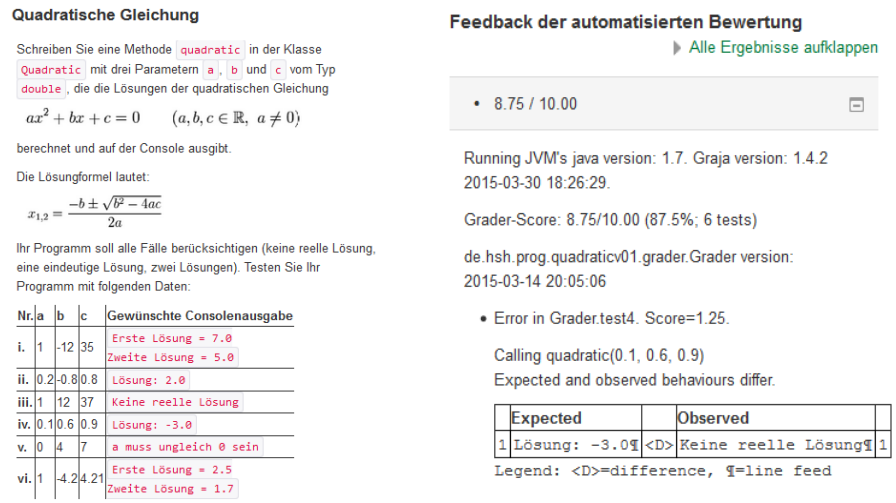
3.4 Graja-Plugin

Graja bewertet studentische Java-Programme unter Einsatz von JUnit 4 sowie einer mitgelieferten Klassenbibliothek. Graja kann über die Kommandozeile in einer eigenen JVM oder über ein Java API direkt aus einer laufenden JVM gestartet werden. In beiden Fällen müssen mehrere Informationen angegeben werden: Aufgabenstruktur, Maximalpunktzahl, Dateipfade sowie weitere Konfigurationsparameter. Auf der Kommandozeile erhält Graja eine entsprechende Zip-Datei im Austauschformat [SSM+14]; am Java API erhält Graja ein entsprechendes Java-Objekt als Eingabe. Durch diese beiden Schnittstellen kann Graja direkt auf dem LMS-Server ausgeführt werden, wenn dort ein JDK installiert ist und wenn das LMS entweder die Ausführung eines Shell Skripts ermöglicht oder das LMS direkt auf die Java API von Graja zugreift. Graja antwortet nach erfolgter Bewertung mit einem

Ergebnis, welches sich i. W. aus erreichten Punkten sowie HTML-Fragmenten mit Bewertungskommentaren zusammensetzt. Graja wird seit Oktober 2012 regelmäßig in Programmieren-Lehrveranstaltungen mit einer LMS-Eigenentwicklung („ppkm“) eingesetzt.

Da Graja keine Webservice-Schnittstelle besitzt, wurde mit dem Ziel, Graja in moodle und ohne zusätzliche Entwicklungsaufwände auch in weiteren LMS einzusetzen, ein Graja-Backend-Plugin unter Nutzung der Backend-Plugin-Schnittstelle von Grappa erstellt. Das Graja-Backend-Plugin bildet hierbei einfach Objekte des Grappa-Domänenmodells auf Transferobjekte der Graja-API ab.

Graja lässt sich nun wie schon zuvor der Grader aSQLg über moodle steuern. Die Benutzeroberfläche ist Lehrkräften und Studierenden, die bereits mit aSQLg Erfahrung gesammelt haben, bekannt, so dass kaum Schulungs- und Einarbeitungsaufwände anfallen. Abb. 3 zeigt ein von Graja geliefertes Bewertungsergebnis, wie es sich für einreichende Studierende in moodle vermittels des Graja-Backend-Plugins darstellt.



Quadratische Gleichung

Schreiben Sie eine Methode `quadratic` in der Klasse `Quadratic` mit drei Parametern `a`, `b` und `c` vom Typ `double`, die die Lösungen der quadratischen Gleichung

$$ax^2 + bx + c = 0 \qquad (a, b, c \in \mathbb{R},\ a \neq 0)$$

berechnet und auf der Console ausgibt.

Die Lösungformel lautet:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Ihr Programm soll alle Fälle berücksichtigen (keine reelle Lösung, eine eindeutige Lösung, zwei Lösungen). Testen Sie Ihr Programm mit folgenden Daten:

Nr.	a	b	c	Gewünschte Consolenausgabe
i.	1	-12	35	`Erste Lösung = 7.0` `Zweite Lösung = 5.0`
ii.	0.2	-0.8	0.8	`Lösung: 2.0`
iii.	1	12	37	`Keine reelle Lösung`
iv.	0.1	0.6	0.9	`Lösung: -3.0`
v.	0	4	7	`a muss ungleich 0 sein`
vi.	1	-4.2	4.21	`Erste Lösung = 2.5` `Zweite Lösung = 1.7`

Feedback der automatisierten Bewertung

Alle Ergebnisse aufklappen

- 8.75 / 10.00

Running JVM's java version: 1.7. Graja version: 1.4.2 2015-03-30 18:26:29.

Grader-Score: 8.75/10.00 (87.5%; 6 tests)

de.hsh.prog.quadraticv01.grader.Grader version: 2015-03-14 20:05:06

- Error in Grader.test4. Score=1.25.

Calling quadratic(0.1, 0.6, 0.9)
Expected and observed behaviours differ.

	Expected		Observed	
1	`Lösung: -3.0¶`	<D>	`Keine reelle Lösung¶`	1

Legend: <D>=difference, ¶=line feed

Abb. 3: Durch Grappa vom Graja-Backend-Plugin an moodle vermitteltes Feedback (rechts) zu einer in moodle eingestellte Programmieraufgabe (links)

Derzeit liegen mit der Neuentwicklung nur praktische Erfahrungen im kleinen Rahmen vor. Graja soll im Wintersemester 2015/16 erstmals in einer Programmieren-Lehrveranstaltung in Kombination mit moodle verwendet werden.

4 Ausblick und Zusammenfassung

In der Vergangenheit ist es gelungen, Grappa in seinen Funktionen zu stabilisieren und hinsichtlich der gewonnen Erfahrungswerte zu verbessern. Weitere wichtige Anforderungen an Grappa konnten so umgesetzt werden: Der Entwicklungsaufwand zur Anbindung

eines LMS wurde reduziert. Die Anbindung des Graders Graja gelang, ohne dass an Grappa selbst oder am Frontend Graja-spezifische Anpassungen erforderlich waren. Mit Hilfe der in 3.2 vorgestellten Client-Bibliothek soll zukünftig das an der HsH entwickelte LMS ppkm an Grappa angebunden werden. Ebenso laufen Entwicklungsarbeiten zur Unterstützung des in [SSM+14] vorgestellten XML-basierten Austauschformates für Programmieraufgaben für die automatische Bewertung. Dadurch sollen in Zukunft die erforderlichen Entwicklungsarbeiten für anzubindende Frontend- und Backend-Plugins für LMS wie z.B. LON-Capa und Grader wie JACK, Praktomat, Gate etc. (eine Zusammenstellung ausgewählter Grader bietet [SSM+14]) deutlich reduziert werden. Im günstigsten Fall entfallen diese ganz.

Während die Grundfunktionen in Grappa und im moodle-Plugin realisiert sind, wird derzeit prioritär daran gearbeitet, die vom moodle-Plugin angelegten Programmieraufgaben und Konfigurationsdateien sichern und wiederherstellen zu können. Zusammen mit weiteren Tests und Fehlerbehebungen wäre damit die Betaphase abgeschlossen und eine erste Version für weitere moodle-Produktivsysteme stünde zur Verfügung. Weiterhin werden Tooltips und Hilfestellungen zum Anlegen von Programmieraufgaben (Lehrenden-Sicht), als auch Abgabeansichten schrittweise verbessert. Geplant ist zudem die Entwicklung weiterer Grader-Backend-Plugins zur Anbindung weiterer Systeme zur automatisierten Bewertung von Programmen (z.B. in C++).

Die gewonnenen Erkenntnisse aus Abschnitt 2 zeigen, dass das Feedback des Graders detaillierter, weniger technisch und damit insgesamt besser auf Studierende zugeschnitten werden muss. Diese Verbesserung ist Teil des eCULT+ Projektes und soll –insofern die Förderung erfolgt– spätestens ab Oktober 2016 begonnen werden.

Literaturverzeichnis

[SBG+14] Stöcker, A.; Becker, S.; Garmann, R.; Heine, F.; Kleiner, C.; Werner, P.; Grzanna, S.; Bott, O.: Die Evaluation generischer Einbettung automatisierter Programmbewertung am Beispiel von Moodle und aSQLg. In: Trahasch, S., Plötzner, R., Schneider, G., Sassiat, D., Gayer, C. & Wöhrle, N. (Hrsg.), DeLFI 2014 - Die 12. e-Learning Fachtagung Informatik. Bonn: Gesellschaft für Informatik. (S. 301-304).

[SSM+14] Strickroth, S.; Striewe, M.; Müller, O.; Priss, U.; Becker, S.; Bott, O.; Pinkwart, N.: Wiederverwendbarkeit von Programmieraufgaben durch Interoperabilität von Programmierlernsystemen. In: Trahasch, S., Plötzner, R., Schneider, G., Sassiat, D., Gayer, C. & Wöhrle, N. (Hrsg.), DeLFI 2014 - Die 12. e-Learning Fachtagung Informatik. Bonn: Gesellschaft für Informatik. (S. 97-108).

[GHW15] Garmann, R.; Heine, F.; Werner, P.: Grappa – die Spinne im Netz der Autobewerter und Lernmanagementsysteme. In Keil, R. (Hrsg.), Pongratz, H. (Hrsg.), DeLFI 2015 - Die 13. E-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V. (S. 169-181).