Moodle, Grappa, aSQLg, Graja - Neue Entwicklungen bei der Grading-Software der Hochschule Hannover

Robert Garmann[1], Peter Fricke[2], Paul Reiser[2], Christopher Bersuch[2], Oliver J. Bott[3]

Abstract: For several years the grading software developed at the Hanover University of Applied Sciences and Arts has increased the quality of various university courses. Increased quality had been confirmed by several evaluations. In this paper, we present improvements to the software that have been achieved in the past two years. We take a look at all software layers involved: the frontend "Moodle" with its Moodle plugin, the graders "Graja" and "aSQLg" in the backend, as well as the middleware "Grappa" mediating between these components. We describe some of the improvements in detail and illustrate them with diagrams and screen photos.

Abstract: Die an der Hochschule Hannover entwickelte Grading-Software steigert seit mehreren Jahren die Qualität der Lehrveranstaltungen verschiedener Studiengänge der Hochschule. Qualitätsverbesserungen konnten durch mehrere Evaluierungen belegt werden. Im vorliegenden Beitrag stellen wir Verbesserungen der Software vor, die in den letzten beiden Jahren erreicht wurden. Wir werfen einen Blick auf alle beteiligten Softwareschichten: das Frontend „Moodle“ und das hier eingesetzte Moodle-Plugin, die Grader „Graja“ und „aSQLg“ im Backend, sowie die zwischen diesen Komponenten vermittelnde Middleware „Grappa“. Wir beschreiben einige der erreichten Verbesserungen im Detail und illustrieren diese durch Diagramme und Bildschirmfotos.

Keywords: Programmieraufgabe, Grader, Moodle, SQL, Java, E-Assessment

1 Einleitung

Seit mehreren Jahren nutzen Lehrende der Hochschule Hannover (HsH) das Lernmanagementsystem (LMS) *Moodle* zur Verwaltung und Durchführung von Lehrveranstaltungen. Durch die an der HsH entwickelte Middleware *Grappa* können Lehrende Programmieraufgaben für verschiedene Programmiersprachen mit einer automatischen Bewertung versehen. Grappa kann prinzipiell jedes LMS mit jedem Autobewerter (*Grader*) vernetzen [GHW15]. Für die Einbindung von Grappa in Moodle entstand ein spezielles Moodle-Plugin, welches mit Grappa über einen auch in anderen LMS-Anbindungen einsetzbaren PHP-Client asynchron verbunden ist (s. Abb. 1). Mit den an Grappa über *Backend-Plugins* angebundenen Gradern *Graja* und *aSQLg* bietet die HsH den Studierenden derzeit automatisiert bewertete Aufgaben für Java und SQL an.

[1] Hochschule Hannover, Fakultät IV Wirtschaft und Informatik, Ricklinger Stadtweg 120, 30459 Hannover, robert.garmann@hs-hannover.de
[2] Hochschule Hannover, ZLB – E-Learning-Center, Expo Plaza 12, 30539 Hannover, peter.fricke@hs-hannover.de, paul.reiser@stud.hs-hannover.de, christopher.bersuch@stud.hs-hannover.de
[3] Hochschule Hannover, Fakultät III Medien, Information und Design, Expo Plaza 12, 30539 Hannover, oliver.bott@hs-hannover.de

Ausgehend von Erfahrungen in früheren Einsatzszenarien [Fri15] wurden in den letzten beiden Jahren einige Neuerungen erreicht. Die Neuerungen führten einerseits zu einer verbesserten Usability, andererseits wurden nicht zuletzt durch technische Optimierungen zusätzliche Funktionen ermöglicht. Beispielsweise kann die Konfiguration von Programmieraufgaben nun zentral und damit benutzerfreundlicher erfolgen. Ein weiteres Beispiel ist das sog. *Gradebook* in Moodle, in dem nun auch Programmieraufgaben zusammen mit dem Feedback aufgelistet werden. Als drittes Beispiel seien SQL-Aufgaben genannt, die nun auch für andere Datenbanksysteme (z. B. H2) realisiert werden können. Diese und weitere Beispiele für Neuerungen beschreiben die Abschnitte 2 bis 4 dieses Aufsatzes. Wir beschließen den Aufsatz in Abschnitt 5 mit einer Beschreibung der zukünftig geplanten und teilweise schon begonnenen Verbesserungen.

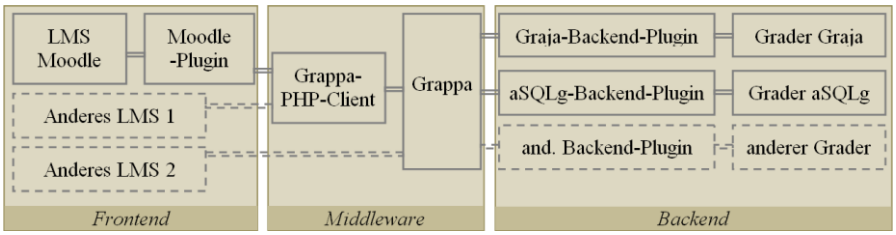

Abb. 1: Softwarekomponenten zur Einbettung von Gradern in Moodle. Die gestrichelt dargestellten Komponenten und Verbindungen sind in Planung.

2 Neuerungen im Frontend

In diesem Abschnitt beschreiben wir Verbesserungen des Moodle-Plugins, die sich insb. auf die Benutzungsschnittstelle auswirken.

2.1 Zentrale Konfiguration von Programmieraufgaben

Bei Verwendung des Moodle-Plugins legt die Lehrkraft in einem Kurs zwei Arten von Artefakten an: *Konfigurationsdateien für Grader* (kurz Grader-Configs resp. *GrdCfg*) und *Programmieraufgaben*. Programmieraufgaben legen Grader-unabhängige Eigenschaften fest, wie Abgabezeitpunkt, Benachrichtigungseinstellungen, etc. Zudem referenzieren Programmieraufgaben jeweils ein oder mehrere GrdCfgs. Eine GrdCfg kann in mehreren Aufgaben eingesetzt werden. Bspw. wird mit Graja i. d. R. pro Kurs eine einzige Properties-Datei mit globalen Einstellungen für alle Aufgaben genutzt. Diese Datei ist eine GrdCfg mit der Graja-spezifischen Rolle *props*. Jede Graja-Aufgabe hat zusätzlich eine eigene GrdCfg, die die eigentliche Aufgabe im ProFormA-Format enthält (Graja-spezifische Rolle *task*).

Grappa-Server-Instanz	Name	Rolle	Datei	GrdCfg-ID	Hinweise/Fehler	Aktionen
Graja-1-7-0@HsH	Aufgabe "Bruch"	task	de.hsh.prog.bruchv02.zip	_5e0d7e34-7447-4207-8354-dcbb8a3f82e8	-	
Graja-1-7-0@HsH	Graja-properties	props	debug.props	_217140d7-faa9-4096-9d4b-838ce20a3e4a	-	

Abb. 2: Liste der im Kurs gespeicherten Grader-Configs (Ausschnitt)

Die Benutzerführung in Moodle wurde inzwischen insofern verbessert, dass GrdCfgs gemäß ihrem referenzierbaren, zentralen Charakter als Teil der Kurs-Administration gepflegt werden können (s. Abb. 3). Von einer Liste aller im Kurs gespeicherten GrdCfgs (s. Abb. 2) gelangt man zu den Einstellungen einer einzelnen GrdCfg (s. Abb. 4).

Abb. 3: Zentraler Zugang zu Grader-Konfigurationen

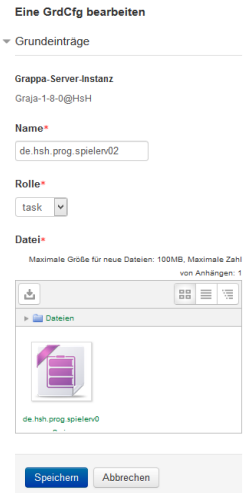

Abb. 4: Einstellungen einer GrdCfg

Diese Art der Verwaltung von GrdCfgs ist wesentlich intuitiver als die bisherige Realisierung als Moodle-Aktivitäten. Auch wurden durch diese Realisierung die technisch notwendigen Voraussetzungen für summatives Assessment mit Moodle-Tests geschaffen (vgl. auch Abschnitt 5).

2.2 Programmieraufgaben im „Gradebook“

Die vom Moodle-Plugin angebotene *Programmieraufgabe* entstand als Spezialisierung des zum Moodle-Standardumfang gehörenden *assignments*. Dadurch befinden sich Programmieraufgaben und „normale“ Aufgaben gleichberechtigt im sog. *Gradebook* eines Studenten. Üblicherweise werden semesterbegleitend sowohl automatisiert bewertete Aufgaben als auch einige von Tutoren bewertete Aufgaben gestellt.

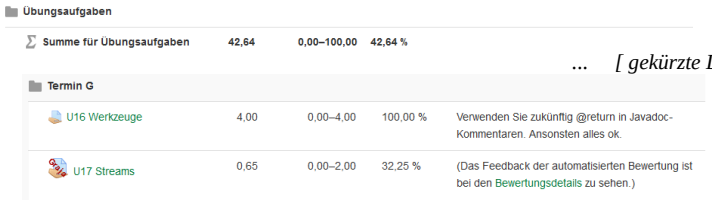

Abb. 5: Ausschnitt eines Gradebooks mit einer „normalen“ und einer automatisiert bewerteten Aufgabe. Punktzahl und Feedback sind integriert bzw. durch einen Mausklick auf *Bewertungsdetails* erreichbar.

Am Ende muss als Teil der Prüfungsleistung ein vorgegebener Prozentsatz aller Aufgaben erreicht werden. Die Studierenden haben während des Semesters stets einen Überblick über die bisher erreichten Punkte im Gradebook (s. Abb. 5).

2.3 Programmieraufgaben und normale Moodle-Aufgaben

Die o.g. *Programmieraufgabe*n (programming assignment) basieren ursprünglich auf den standardmäßig in Moodle integrierten *Aufgabe*n (assignments) einer früheren Moodle-Version. Letztere wurden in Moodle-Folgeversionen weitgehend verändert. Damit Lehrende, die die Erstellung der normalen Moodle-Aufgaben bereits kennen, sich weiterhin auf der Benutzeroberfläche zurechtfinden, wurde die Programmieraufgabe der Aktivität *Aufgabe* der neuesten Moodle-Version wieder angeglichen. Neue Zusatzfunktionen der Aufgabe sollen bald ebenfalls in der Programmieraufgabe nachprogrammiert werden. Lediglich in Einstellungsoptionen, die für Programmieraufgaben spezifisch sind, wie z. B. die Auswahl der unterschiedlichen Grader sowie ihrer GrdCfgs, weicht die Programmieraufgabe weiterhin von einer Aufgabe ab. Eine weitere hilfreiche Neuerung der Programmieraufgabe ist, dass nur noch die GrdCfgs zur Auswahl per Checkbox angeboten werden, die für den ausgewählten Grader zur Verfügung stehen.

2.4 Technische Neuerungen „unter der Haube“

Zu den Grundfunktionen für sämtliche Aktivitäten innerhalb eines Kurses in Moodle gehört das Kopieren der Aktivität. Dazu bedient sich Moodle einer Sicherungs- und Wie-

derherstellungs-Engine[4], welche ebenfalls für die Kurssicherung genutzt wird. In Moodle werden Elemente des Grappa-Domänenmodells (*Problem* und *Result*, siehe [GHW15]) für Informations- und Editierzwecke angezeigt und persistiert. Wegen der durchaus komplexen Baumstruktur des Domänenmodells wurden die zuvor genannten Sicherungs- und Wiederherstellungsfunktionen in den ersten Versionen der *Programmieraufgabe* außer Acht gelassen. Die Integration dieser Features wurde vorangetrieben, so dass es nun möglich ist, eine *Programmieraufgabe* innerhalb eines Kurses zu kopieren. Ebenfalls kann diese jetzt zwischen Kursen und Moodle-Instanzen ausgetauscht werden. Für den Austausch zwischen verschiedenen Moodle-Instanzen muss auf beiden Seiten allerdings eine Grappa-Server-Instanz mit dem gleichen Namen vorhanden sein. Der Name dient der eindeutigen Identifizierung und Zuordnung von *GrdCfg*s und *Problem*s zu einer Grappa-Server-Instanz innerhalb von Moodle. Ein Dialog, der die Grappa-Server-Instanz abfragt, ist nicht möglich, denn die Moodle interne Restore-API[5] muss aufgrund der automatischen Kurssicherung ohne Dialoge auskommen (diese wird per Batchverarbeitung ausgeführt).

Innerhalb von Moodle werden alle Aktionen des Systems und der Nutzerinnen und Nutzer in *Berichten* geloggt. Seit Moodle Version 2.7 wurde die Event-Api[6] grundlegend überarbeitet. Die *Programmieraufgabe* (Abschnitt 2.3) entstand vor dieser Moodle-Version, so dass eine Anpassung notwendig war. Das Logging der *Programmieraufgabe* wurde komplett neu entwickelt und erweitert. Zur Vielzahl verschiedener Events, die für einen Administrator sichtbar sind, gehören bspw. das Anlegen, Ändern, Löschen und Bewerten einer Programmieraufgabe, sowie das Abrufen und manuelle Ändern der Aufgabenbewertung mit den dazugehörigen Benutzerdaten.

3 Neuerungen in der Middleware

Dieser Abschnitt beschreibt Verbesserungen im Grappa-PHP-Client und im Grappa-Kern. Da Grappa unabhängig vom LMS konzipiert wurde, stehen die hier beschriebenen Verbesserungen neben Moodle auch anderen LMS zur Verfügung.

3.1 ProFormA-Format

Grappa entstand vor Entwicklung des ProFormA-Austauschformats für Programmieraufgaben [St15]. Das hiervon unabhängig entwickelte Fachdatenmodell von Grappa besitzt jedoch hinreichend Ähnlichkeiten zum ProFormA-Format, die es erlauben, Grappa-Aufgaben in ProFormA-Aufgaben zu konvertieren und umgekehrt. Abb. 6 zeigt Ausschnitte der beiden Fachdatenmodelle. Ein Grappa-*ProblemNode* entspricht weitgehend einem ProFormA-*test*. Die Hierarchie von Grappa-*subproblem*s, die eine hierarchische

[4] https://docs.moodle.org/33/en/Backup_and_restore_FAQ
[5] https://docs.moodle.org/dev/Restore_API
[6] https://docs.moodle.org/dev/Event_2

Kategorisierung von Bewertungsaspekten und deren Gewichtung darstellt, lässt sich in ProFormA-*grading-hints* als Grappa-spezifische Erweiterung beschreiben. Grappa-*GrdCfg*s entsprechen der ProForma-*test-configuration* und deren *filerefs*.

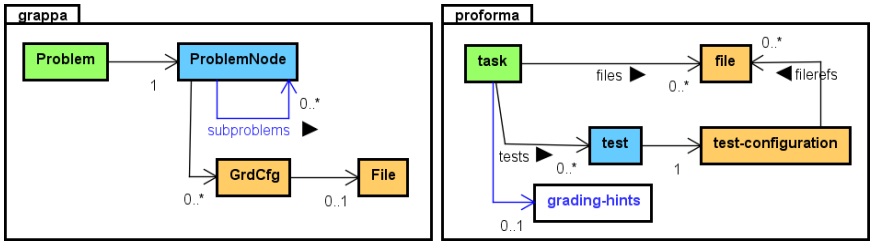

Abb. 6: Ausschnitt aus den Fachdatenmodellen des Grappa- und des ProFormA-Formats. Entsprechungen sind gleichfarbig markiert.

Die Entwicklungsarbeiten zur Konvertierung zwischen den beiden Formaten sind derzeit noch nicht abgeschlossen. Es gibt jedoch eine einfache Möglichkeit, bereits jetzt Grappa mit ProFormA-Aufgaben zu nutzen. Ein Grappa-*Problem* besteht bei diesem Ansatz aus einem einzigen *ProblemNode* ohne *subproblem*s. Eines der dem *ProblemNode* zugeordneten *GrdCfg*s enthält eine vollständige ProFormA-*task*. Der Vorteil dieses Ansatzes ist, dass jedes Grappa-kompatible LMS dadurch bereits heute eine Benutzungsschnittstelle zu ProFormA-kompatiblen Gradern anbieten kann.

Wir binden auf diese Weise derzeit Graja und dessen ProFormA-kompatible Java-Aufgaben via Grappa an das Moodle-Plugin an. Der Nachteil dieses vorläufigen Ansatzes ist, dass Lehrende die Gewichtung von Bewertungsaspekten bei der Einrichtung einer Aufgabe nicht wie gewohnt auf der Weboberfläche des LMS, sondern nur durch die etwas umständlichere Editierung der ProFormA-*task* vornehmen können.

3.2 Technische Neuerungen „unter der Haube“

Die REST-Schnittstelle zu Grappa wurde HTTP-konform angepasst. Ein LMS, das an Grappa direkt oder über den PHP-Client anbindet, sendet HTTP-Anfragen und erhält HTTP-Antworten, die mit standardisierten HTTP-Status-Codes versehen sind. Auf Basis dieser Codes kann das LMS HTTP-Antworten auswerten und auf unterschiedliche Ereignisse geeignet reagieren.

Das Logging von Ereignissen wurde in Grappa umstrukturiert. Lognachrichten werden pro Grappa- bzw. Grader-Instanz[7] gruppiert und in separate Dateien geschrieben. Log-Nachrichten werden mit Kontextinformationen versehen, um das Auffinden, Analysieren und Beseitigen von Fehlern zu optimieren. Sämtliche Aktionen laufen im Kontext eines LMS ab, dessen ID in die Log-Nachrichten aufgenommen wird. Abhängig von der Art

[7] Die in Abb. 1 dargestellte Grappa-Komponente wird zur Laufzeit für jedes angeschlossene Grader-Backend als separate Instanz gestartet. Jede Instanz erhält einen eigenen Zugriffspfad und eine eigene Konfiguration.

der Aktion kommen weitere Informationen wie GrdCfg- und Problem-ID, sowie eine ID des jeweiligen Bewertungsprozesses hinzu.

Programmieraufgaben, bei denen die Dateinamen eingereichter Dateien lösungsrelevant sind, werden bisher in Moodle als ZIP-Datei abgegeben. Um das manuelle „Zippen“ von Abgaben, die nur aus einer einzelnen Quellcode-Datei bestehen, zu vermeiden, wurde das interne Domänenmodell von Moodle und Grappa (inkl. PHP-Client) so erweitert, dass auch einzelne Quellcode-Dateien zusammen mit ihrem Dateinamen von Moodle über Grappa an angeschlossene Grader durchgereicht werden können.

Eine weitere Verbesserung betrifft das Management des für den Betrieb eines Grappa-Servers erforderlichen Datenbanksystems zur Speicherung relevanter Daten und Statusinformationen. Bei Programmstart gestartete Update Scripts aktualisieren nun automatisch die verwendete Datenbank nach einem Grappa-Versionswechsel (getestet mit MySQL, Oracle und PostgreSQL).

4 Neuerungen im Backend

Die Grader aSQLg und Graja sowie ihre jeweiligen Backend-Plugins für Grappa wurden verbessert. Die Auswirkungen auf die Benutzungsschnittstelle und die unterstützten Funktionen werden nun beschrieben. Da beide Grader LMS-unabhängig sind, stehen die hier beschriebenen Verbesserungen neben Moodle auch anderen LMS zur Verfügung.

4.1 SQL-Aufgaben für weitere Datenbanksysteme

Der Grader aSQLg wurde für das DBMS Oracle entwickelt. Aufgrund von Nachfragen, ob aSQLg auch einfachere und kleinere Datenbanken unterstützen könne wie bspw. H2[8], wurde aSQLg erweitert. Es ist nun möglich, über die Konfigurationsdateien zwischen Oracle- und H2-Datenbanken zu wechseln. Ein simultaner Betrieb beider Datenbanksysteme ist ebenfalls möglich und wurde im Sommersemester 2017 erstmals im Lehrbetrieb genutzt.

4.2 Graja-Feedback interaktiv einblendbar

Graja kategorisiert wie viele Grader sein Feedback nach verschiedenen Bewertungsaspekten. Technisch liefert Graja ein einziges HTML-Fragment zurück, in dem die hierarchische Bewertungsstruktur durch geschachtelte HTML-Listen ausgedrückt wird. Ein solches HTML-Fragment lässt sich in beliebige web-basierte LMS einfach integrieren.

[8] http://www.h2database.com

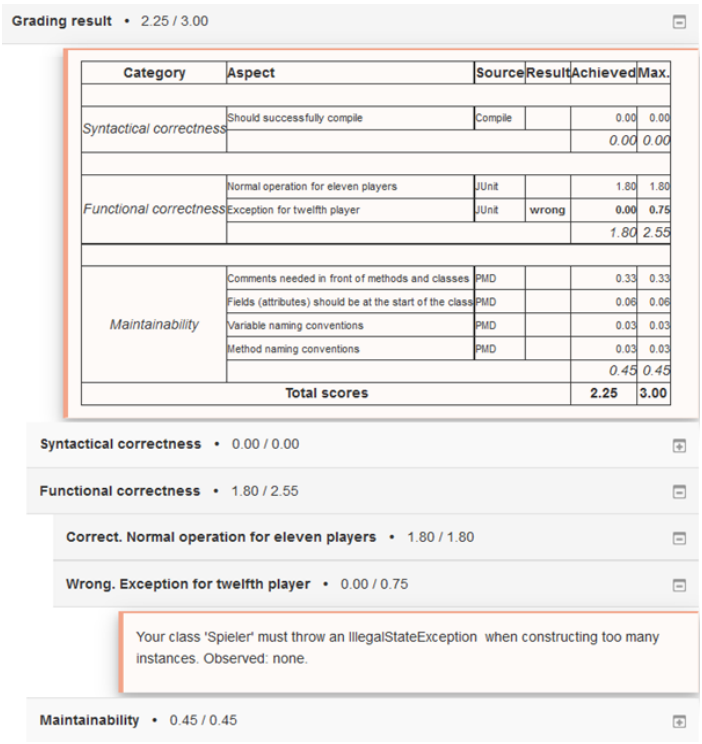

Abb. 7: Feedback mit eingeblendeten Details der Kategorie *Functional correctness* und ihrer zugehörigen Bewertungsaspekte (gekürzt).

Bei komplexen Aufgaben kann das Feedback sehr umfangreich ausfallen, so dass einreichende Studierende den Überblick verlieren können. Damit Studierende interaktiv diejenigen Teile des Feedbacks ausblenden können, die nicht im Fokus des Interesses stehen, wurde das Moodle-Plugin entsprechend erweitert. Genutzt wurde hierfür die Fähigkeit der Middleware Grappa, Bewertungsergebnisse als einen Baum von Ergebnisknoten an das LMS zu liefern.

Graja und das Backend-Plugin wurden nun dahingehend erweitert, dass sie eine Hierarchie von Ergebnisknoten liefern, wobei jeder Ergebnisknoten einen Teil des ursprünglichen HTML-Fragments enthält. Abb. 7 zeigt ein Beispiel, bei dem interaktiv nur ein Teil des Gesamtfeedbacks eingeblendet wurde.

5 Zusammenfassung und Ausblick

Die Grappa-Middleware und das Moodle-Plugin hat mit den beschriebenen Neuerungen nun einen Stand erreicht, der es ermöglichte, dessen bisherige Bereitstellung auf einem

Moodle-Testsystem für nur einen begrenzten Teilnehmerkreis von Lehrenden dahingehend zu erweitern, das Plugin und damit die Grader aSQLg und Graja auch im Moodle-Produktivbetrieb der HsH interessierten Nutzern anzubieten. Dennoch sind weitere Entwicklungen erforderlich. Zwar bietet die Bereitstellung von Moodle-Programmieraufgaben die Möglichkeit, insbesondere im Kontext formativen Assessments auch komplexere Aufgaben zur automatisierten Bewertung anzubieten. Um aber auch bei summativen Assessments, z. B. bei einer E-Klausur, Programmieraufgaben integriert in einen Moodle-Test bestehend aus mehreren Aufgaben auch anderen Typs (z. B. Lückentext oder MC-Frage) anbieten zu können, ist die Entwicklung eines eigenen „Fragetyps“ für Programmieraufgaben erforderlich. Erstellung und Bearbeitung einer derartigen „Programmierfrage“ sind ähnlich zur Programmieraufgabe, allerdings werden Feedback und Bepunktung anders gehandhabt. Die entsprechende Weiterentwicklung ist für das zweite Halbjahr 2017 geplant. Weitere erforderliche Erweiterungen betreffen die Entwicklung zusätzlicher Backend-Plugins für Grader wie Jack [SZG15] oder Praktomat [Gi16], sowie die vollständige Unterstützung des ProFormA-Austauschformates.

Literaturverzeichnis

[Fr15] Fricke, P. et al.: Grading mit Grappa - Ein Werkstattbericht. In: Automatische Bewertung von Programmieraufgaben, Wolfenbüttel, CEUR-WS.org, 2015.

[GHW15] Garmann, R.; Heine, F.; Werner, P.: Grappa – Die Spinne im Netz der Autobewerter und Lernmanagementsysteme. In: DeLFI 2015, München, Bd. 247, LNI, GI, 169-181.

[Gi16] Giffhorn D. u. a.: Praktomat, http://pp.info.uni-karlsruhe.de/projects/praktomat /praktomat.php, Stand: 16.06.2017.

[St15] Strickroth, S. et al.: ProFormA: An XML-based exchange format for programming tasks. eleed e-learning & education, 11(1), 2015.

[SZG15] Striewe, M.; Zurmaar, B.; Goedicke, M.: Evolution of the E-Assessment Framework JACK. In: Software Engineering Workshops 2015, Dresden, CEUR-WS.org, 2015.