ProFormA 2.0 – ein Austauschformat für automatisiert bewertete Programmieraufgaben und für deren Einreichungen und Feedback

Paul Reiser[1], Karin Borm[2], Dominik Feldschnieders[3], Robert Garmann[4], Elmar Ludwig[3], Oliver Müller[5], Uta Priss[2]

Abstract: Das seit 2011 entwickelte ProFormA-Format soll den Austausch von automatisiert bewertbaren Programmieraufgaben zwischen verschiedenen Lernmanagementsystemen und Gradern befördern. Dieser Beitrag stellt die neue Version ProFormA 2.0 vor, die neben der eigentlichen Aufgabe jetzt auch das Format der studentischen Einreichung, des Bewertungsschemas und der Grader-Antwort standardisiert.

Keywords: automatisch bewertete Programmieraufgaben, XML Austauschformat, Interoperabilität, E-Assessment, LMS, Grader

1 Einleitung und Motivation

Das ProFormA-Format wird seit 2011 im Rahmen des eCULT-Projekts entwickelt [St15, St17]. In der ersten Version war es auf die detaillierte Beschreibung von Programmieraufgaben fokussiert einschließlich der Aufgabenbeschreibung, zumindest einer Musterlösung und Tests zur Bewertung der Aufgabe. Dieses Format soll die Kommunikation von Lernmanagementsystemen (LMS), die Aufgaben verwalten und von Studierenden und Lehrenden bedient werden, mit Bewertungssystemen (Gradern) und eventuell Vermittlungssystemen (Middleware) vereinheitlichen. Grader laufen im Hintergrund und werden zur Bewertung eingesetzt, zum Beispiel zur Ausführung von Kompilations- und Unittests und Stilprüfungen. Eine Middleware kann dazu dienen, mehrere verschiedene Grader an ein LMS anzuschließen. Durch die Vereinheitlichung der Kommunikation können unterschiedlichste Systeme (LMS, Grader und Hybrid-Systeme, siehe Abschnitt 4) kombiniert werden, sofern sie das Format unterstützen. Programmieraufgaben, die für ein System geschrieben werden, funktionieren auch auf den anderen Systemen und können mit anderen Lehrenden ausgetauscht und wiederverwendet werden.

[1] ZLB – E-Learning Center, Hochschule Hannover, Expo Plaza 4, 30539 Hannover, paul.reiser@hs-hannover.de
[2] ZeLL, Ostfalia Hochschule, Salzdahlumer Straße 46/48, 38302 Wolfenbüttel, {k.borm,u.priss}@ostfalia.de
[3] Zentrum für Digitale Lehre, Campus-Management und Hochschuldidaktik, Universität Osnabrück, Heger-Tor-Wall 12, 49074 Osnabrück, {dofeldsc,elmar.ludwig}@uos.de
[4] Fakultät IV, Hochschule Hannover, Ricklinger Stadtweg 120, 30459 Hannover, robert.garmann@hs-hannover.de
[5] Institut für Informatik, TU Clausthal, Julius-Albert-Str. 4, 38678 Clausthal-Zellerfeld, oliver.mueller@tu-clausthal.de

doi:10.18420/abp2019-6

Einen Überblick über ProFormA 1.1 und dessen Einsatz in diversen Systemen bietet [Bo17]. ProFormA 2.0, ein Whitepaper und Beispiele sind auf Github[6] verfügbar.

Neben einem überarbeiteten Aufgabenformat wurde in der zweiten, in diesem Artikel vorgestellten Version, auch das Format, in dem das LMS die studentische Lösung an den Grader schickt und die Antwort, die der Grader an das LMS schickt, standardisiert. Dieser Beitrag bietet einen Überblick über ProFormA 2.0.

2 Neue Anforderungen an das Format

Die in [St15] veröffentlichte Erstversion des ProFormA-Formats deckt eine Obermenge der Anforderungen jedes Graders ab. Sie fokussiert auf Aspekte der Aufgabe (*task*): Aufgabentext, Ausfüllvorlagen (Templates), Code-Bibliotheken, Testcode, Konfigurationsdateien und Musterlösungen. Auch ProFormA 2.0 soll möglichst viele Grader sowie zusätzlich LMS und Middlewares unterstützen. Die daraus erwachsenden zusätzlichen Anforderungen werden nun skizziert und nummeriert. Alle Anforderungen adressieren den Dateninhalt. Die Anforderung eines zwischen den Systemen abgestimmten, technischen Kommunikationsprotokolls (REST, SOAP, etc.) ist kein Gegenstand dieses Beitrags.

Es müssen Formate für die Kommunikation zwischen LMS-Frontend und Grader-Backend formuliert werden. Möglichst jedes LMS soll eine studentische Einreichung (*submission*) an jeden ProFormA-kompatiblen Grader senden können **(R1)**. Ein LMS soll das als Antwort (*response*) automatisch generierte Feedback entgegennehmen und Zielgruppenorientiert anzeigen können **(R2)**. Feedback liegt gewöhnlich quantitativ (z. B. als Punktzahl) und qualitativ vor (z. B. Prosa, Bilder, XML-Datei). Da dieses meist in Teilen den ausgeführten Testschritten zugeordnet wird, soll die Zuordnung vom neuen Format unterstützt werden **(R3)**. Besteht der Bewertungsprozess einer Aufgabe aus mehreren Testschritten, so werden die einzelnen Testergebnisse i. d. R. gewichtet und aggregiert. Während viele der in der Erstversion des Formats spezifizierten Teile einer Aufgabe bei einer Wiederverwendung nicht geändert werden müssen, hängt die Gewichtung einzelner Testergebnisse häufig vom Lehrkontext ab. Lehrende sollen die in einer Aufgabe angelegten, durch Tests bewerteten Aspekte individuell gewichten können **(R4)**. Es wurde ein Format für ein Bewertungsschema (*grading-hints*) gesucht, das ein LMS der Lehrkraft editierbar anzeigen kann, und das die Systeme LMS, Middleware oder Grader zur Berechnung des Gesamtergebnisses zu einer Einreichung nutzen können. Die Wiederverwendung von Aufgaben führt zu der Anforderung, das in der Aufgabe spezifizierte Bewertungsschema für jede Einreichung austauschbar zu gestalten **(R5)**. Als letzte und weitere wichtige Anforderung sollen Aufgaben internationalisiert und mit gezielten Zusatzinformationen spezifiziert werden können, um eine Wiederverwendung durch Menschen (insb. Lehrkräfte) und Maschinen (insb. LMS) zu unterstützen **(R6)**.

[6] https://github.com/ProFormA/proformaxml, https://github.com/ProFormA/examples

3 ProFormA Format 2.0

Abb. 1 stellt die Elemente von ProFormA 2.0 dar. Die Neuerungen gegenüber Version 1.1 werden in den folgenden Abschnitten erläutert.

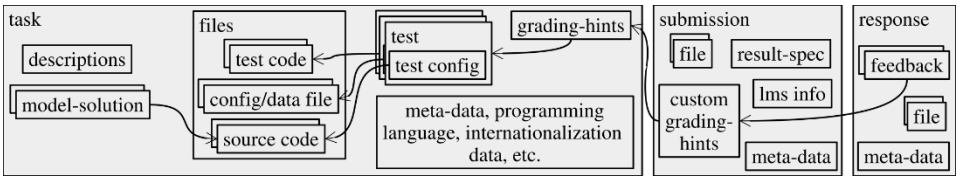

Abb. 1: Elemente einer ProFormA 2.0 Aufgabe, Einreichung und Feedback. Einige Verweise zwischen Elementen sind als Pfeile dargestellt.

3.1 Internationalisierung und allgemeine Datentypen

Bereits in ProFormA 1.1 lässt sich festlegen, ob zu einer Aufgabe gehörende Dateien (Elemente *files/file*, vgl. Abb. 1) bzw. deren jeweiliger Inhalt in einer task.xml eingebettet oder als Teil einer Archivdatei angehängt werden. Für *file*-Elemente, die nun wie in Abb. 1 gezeigt auch innerhalb von *submission*- und *response*-Elementen verwendet werden können, wird in ProFormA 2.0 zusätzlich zwischen Textdateien (Elemente *embedded-txt-file, attached-txt-file*) und Binärdateien (*embedded-bin-file, attached-bin-file*) unterschieden. Während eine eingebettete Textdatei stets UTF-8 kodiert ist und eine eingebettete Binärdatei einer XML base64-kodiert hinzugefügt wird, gibt es die Möglichkeit, für eine angehängte Textdatei über ein Attribut ein zutreffendes Encoding anzugeben. Durch die beschriebenen Änderungen bei *file*-Elementen soll ein LMS beim Anzeigen und ein Grader beim Verarbeiten von Dateien besser unterstützt werden (R1 und R6).

ProFormA 2.0 unterstützt zudem die Möglichkeit für bestimmte Elemente in *task* oder *response* nicht nur eine, sondern beliebig viele natürliche Sprachen zu verwenden (R6, Abb. 1 *internationalization*). Die Hauptsprache einer Aufgabe wird über das optionale *task*-Attribut *lang* angegeben, sofern die Aufgabe in einer einzigen Sprache vorliegt oder nur ein Teil (z. B. die sich an Studierende richtende Aufgabenstellung) in andere Sprachen übersetzt ist. Sind Übersetzungen für *title, description, internal-description* und/oder *content*-Elemente vorhanden, so wird dies im entsprechenden Element gekennzeichnet. Die Kennzeichnung erfolgt durch Angabe der Zeichen @@@ (z. B. <title>@@@title2@@@</title>). Die konkreten Übersetzungen selbst stehen bezugnehmend auf die oben genannten Kennzeichnungen in einer UTF-8 kodierten *strings.txt*-Datei, wobei für jede natürliche Sprache eine separate Datei unter Verwendung entsprechender Ländercodes angelegt werden muss. Dateien lassen sich ebenfalls in mehreren natürlichen Sprachen anbieten und in dem Fall ausschließlich als Anhang beifügen. Im jeweils hierfür vorgesehenen *attached-txt-file*- bzw. *attached-bin-file*-Element wird der anzugebende Dateiname wie oben bereits dargestellt gekennzeichnet (z. B. <attached-bin-file>@@@path-to-pdf-file@@@</attached-bin-file>).

3.2 Bewertungsschema (das *grading-hints*-Element)

Die Gestaltung des Bewertungsschemas (R4) wurde detailliert in [Ga19] beschrieben und kann hier aus Platzgründen nicht ausführlich wiedergegeben werden. Wesentliches Gestaltungsprinzip ist ein Baum, dessen Blätter Verweise auf einzelne Testergebnisse sind (vgl. Abb. 1). An inneren Knoten können verschiedene Aggregationen (Summe, Maximum, Minimum) definiert werden.

3.3 Aufgabe (das *task*-Element)

Das *task*-Element aus Abb. 1 wird aus Platzgründen nicht vollständig beschrieben. Die hier beschriebenen Änderungen gegenüber ProFormA 1.1 ergaben sich – über die in den Abschnitten 3.1 und 3.2 beschriebenen Aspekte hinaus – vor allem aus Anforderung R6.

Das *description*-Element, das die Aufgabenstellung enthält, erlaubt nun flexibel alle gültigen HTML Elemente anstatt einer Teilmenge. Zusätzlich gibt es ein neues *internal-description*-Element, das z. B. die Elemente *model-solution* und *file* mit näheren, nur für Lehrende sichtbaren Informationen ausstattet. Durch das *submission-restrictions*-Element lassen sich Restriktionen für von Studierenden einzureichende Dateien definieren (z. B. Vorgabe von Dateinamen). Diese können unter anderem in Form eines regulären Ausdrucks angegeben werden, für den in ProFormA 2.0 POSIX ERE als Standard gilt.

In ProFormA 1.1 wurde für jede Datei eine Dateiklasse zur Verdeutlichung des Verwendungszwecks angegeben (z. B. *template*, *library*). Da sich die Dateiklassen hierfür zunehmend als unzureichend erwiesen, werden in ProFormA 2.0 stattdessen drei neue Attribute verwendet. Diese geben für jede Datei an, ob sie für die Nutzung durch einen Grader bestimmt ist (*used-by-grader: yes, no*), wie ein LMS mit der Datei umgehen soll (*usage-by-lms: edit, display, download*) und ob sie für Studierende sofort, gar nicht oder erst zu einem späteren Zeitpunkt zur Verfügung gestellt werden soll (*visible: yes, no, delayed*). Im letztgenannten Fall muss eine Lehrkraft den Zeitpunkt (z. B. unmittelbar nach Einreichungsfrist einer Lösung zu einer Aufgabe) über ihr LMS festlegen, das dann zu diesem Zeitpunkt die jeweilige Datei (z. B. eine Musterlösung) den Studierenden bereitstellt. Durch die genannten Attribute lässt sich z. B. je nach Bedarf eine Datei für Studierende als zu nutzende Code-Bibliothek klassifizieren (*used-by-grader*: *yes*, *visible*: *yes*, *usage-by-lms*: *download*), die über ein LMS als Download zur Verfügung steht, oder als Ausfüllvorlage, die Studierende im optimalen Fall direkt im LMS editieren (*used-by-grader*: *no*, *visible*: *yes*, *usage-by-lms*: *edit*).

3.4 Einreichung (das *submission*-Element)

Eine Einreichung (*submission*) setzt sich aus einer studentischen Aufgabenlösung in Form von Dateien sowie aus weiteren, für Grader nützlichen Informationen zusammen. Da Grader eine Lösungsabgabe gegen eine konkrete Aufgabe bewerten, besitzt die Einreichung einen Verweis auf die Aufgabe, der in unterschiedlichen Formen angegeben werden

kann, um den Anforderungen unterschiedlicher Grader zu genügen (R1). Eine Aufgabe kann damit entweder als externe, für einen Grader lokalisierbare Ressource (bspw. in einem Repository im Internet), oder als Bestandteil der Einreichung angegeben werden. Letzteres ermöglicht wahlweise das Einbetten der Aufgabe als XML-Element oder als *file*-Bestandteil der Einreichung gemäß Abschnitt 3.1. Da Aufgaben über eine eindeutige, globale ID verfügen, kann eine Einreichung alternativ nur die Aufgaben-ID referenzieren, falls Grader ein internes Caching von Aufgaben über die ID vornehmen.

Eine Einreichung enthält abgabespezifische Informationen wie Datum und Uhrzeit der Abgabe, optionale Nutzerkennungen der einreichenden Studierenden (bei Einzel- und Gruppenabgaben) sowie LMS-spezifische Daten wie LMS-URL und Kurs-ID. Grader können diese Angaben bspw. dazu nutzen, um Verstöße bei Plagiatsprüfungen auf betroffene Nutzer zurückzuführen. Optional können noch weitere, beliebige Meta-Informationen zu der Einreichung bereitgestellt werden (R6).

Es kann aus technischen Gründen notwendig sein, einem Grader mitzuteilen, in welcher Form ein LMS das Feedback einer Antwort (*response*) verarbeiten und darstellen kann (R2). Das *result-spec*-Element steuert den Detailgrad und die Struktur des erwarteten Feedbacks. Des Weiteren muss das vom LMS verarbeitbare Format der Antwort (als in einer ZIP verpackten oder unverpackten XML-Datei) spezifiziert werden. Anschließend kann die von Studierenden bevorzugte, natürliche Sprache für Feedback-Texte festgelegt werden (Abschnitt 3.1).

Ein weiterer Bestandteil der Einreichung ist das anpassbare Bewertungsschema (*grading-hints*, Abschnitt 3.2). Aufgabenautoren veröffentlichen Aufgaben mit einem vordefinierten Bewertungsschema, das nicht zwangsläufig den Vorstellungen der die Aufgaben einsetzenden Lehrenden entsprechen muss. Das *submission*-Element erlaubt Lehrenden, das voreingestellte Bewertungsschema in einem LMS wunschgemäß anzupassen (R5, Abb. 1). Durch die Anpassung des Bewertungsschemas bildet sich automatisch eine neue Aufgabenkonfiguration, die im Rahmen des jeweiligen LMS-Kurses aktiv wird (R4). Die originale Aufgabe mit dem ursprünglichen vom Aufgabenautor oder von der Aufgabenautorin intendierten Bewertungsschema bleibt unbeeinflusst.

3.5 Antwort (das *response*-Element)

Eine Antwort (*response*) ist ein gradergeneriertes Bewertungsergebnis zu einer studentischen Lösungseinreichung (*submission*). Sie kann die resultierende Gesamtpunktzahl, die Punktzahl für jedes Testergebnis, das textuelle Feedback und Dateien, die für das Verständnis des Bewertungsergebnisses relevant sein können (vgl. Abb. 1) umfassen. In welcher Struktur ein Grader das Feedback bereitstellt, spezifiziert das LMS mit der Einreichung (Abschnitt 3.4, Abb. 1 *submission/result-spec*). Feedback kann damit als vorformatiertes HTML-Fragment strukturiert werden, das bereits alle Informationen (Texte, Punkte und eingebettete Dateien) umfasst und von einem technisch limitierten LMS dargestellt wird. Alternativ wird das Feedback als eine Sammlung von mehreren, einzelnen Testschritten zugeordneten Feedback-Einträgen strukturiert (vgl. R3 und Abb. 1), bei der

die Aufbereitung und Darstellung dem LMS überlassen bleibt. Ein Feedback-Eintrag setzt sich aus einem Titel, dem Text (formatiert als Klartext oder HTML-Fragment) und den für das Feedback relevanten Dateien zusammen, die das LMS als Download-Links darstellen kann. Darüber hinaus gliedern Grader das Feedback nach hierarchischen Log-Leveln (*Debug, Info, Warning, Error*). Das Format sieht vor, dass mehrere Feedback-Einträge einem einzelnen Testschritt zugeordnet werden können, weshalb sich Feedback-Einträge für einen Testschritt mit unterschiedlichen Log-Leveln nicht ausschließen müssen. Sollte ein Testschritt fehlschlagen (bspw. wegen eines in der Einreichung enthaltenen Syntaxfehlers im Quellcode), können diesem Testschritt mehrere Feedback-Einträge zugewiesen werden, bspw. ein *Info*-Eintrag mit der Meldung, dass der Testschritt gescheitert ist, und ein *Error*-Eintrag, der den detaillierten Kompilierfehler aufführt. Lehrende können durch eine entsprechende Konfiguration im LMS steuern, welche Log-Level des Feedbacks Studierende einsehen können (R2). Erkennt ein Grader oder eine Middleware einen Fehler, der auf das Gradingsystem statt auf die studentische Lösungseinreichung zurückzuführen ist, setzen sie ein internes Flag in der Antwort, damit das LMS den jeweiligen Einreichungsversuch invalidiert und betroffene Studierende keinen Einreichungsversuch verlieren.

Feedback für Lehrende und Studierende kann inhaltlich variieren, wenn Grader Feedback mit zusätzlichen, für Lehrende nützlichen Informationen anreichern, die für Studierende keinen didaktischen Nutzen aufweisen. Grader stellen daher das gesamte Feedback einer Antwort in zweifacher Ausführung bereit, einmal für Studierende und einmal für Lehrende (R2). Grader können die beiden Feedback-Teile bspw. inhaltlich disjunkt gestalten. Oder sie erstellen das Lehrendenfeedback als Obermenge des Studierendenfeedbacks. Andere Gestaltungen sind denkbar. Das LMS präsentiert Studierenden nur das studierendenspezifische Feedback, während Lehrende sowohl ihr eigenes, als auch das studierendenspezifische Feedback einsehen können.

4 ProFormA-Kompatibilität

Das ProFormA-Format wird seit mehreren Jahren an verschiedenen Hochschulen eingesetzt. ProFormA 2.0 wird durch die folgenden Software-Projekte zumindest prototypisch unterstützt: Graja, GATE, ein Moodle-Plugin mit Praktomat-Grader an der Ostfalia und Stud.IP. GATE ist ein hybrides System, das sowohl LMS- als auch Grader-Funktionalität beinhaltet und an der TU Clausthal entwickelt wird [Mü17]. Die anderen Systeme verwenden entweder Moodle (ein international bekanntes LMS) oder Stud.IP (ein an deutschen Hochschulen verbreitetes LMS). Graja wird an der Hochschule Hannover (HsH) entwickelt und über die spezielle Middleware Grappa an Moodle angebunden [Ga17]. An der Ostfalia-Hochschule[7] wurde der Praktomat[8] um eine ProFormA-Schnittstelle erweitert, so dass er als reiner Grader nutzbar ist [Ro17]. Eine LMS-seitige Anbindung existiert für Moodle (ProFormA-Moodle-Plugin) und durch das Stud.IP-Aufgaben-

[7] https://github.com/elearning-ostfalia
[8] https://github.com/KITPraktomatTeam/Praktomat

Tool „Vips“ für Stud.IP. Vips bietet Funktionen zur Erstellung und Verwaltung von Programmieraufgaben und war ursprünglich ein eigenständiges Werkzeug, das jetzt aber Auswertungen von ProFormA-kompatiblen Gradern verarbeiten kann. An der Ostfalia wird außerdem ein Editor entwickelt, mit dem man ProFormA-Dateien unabhängig von einem LMS oder Grader bearbeiten kann [Pr17].

Von den verschiedenen Projekten werden zum Teil unterschiedliche Aspekte von ProFormA 2.0 unterstützt. Alle Systeme unterstützen Java-Programmieraufgaben (Kompilierung und JUnit-Tests), bei denen die Test-Dateien in die *task* eingebettet werden, die studentische Einreichung mitgesendet wird und das Feedback in die einzelnen Tests untergliedert als *response* zurückgesendet wird. Alle Systeme außer GATE unterstützen auch Java-Checkstyle. Andere Sprachen (z. B. Python) und Testtypen (z. B. Regexp-Test, Java-PMD) werden nur von einzelnen Systemen bereitgestellt. Ob die Dateien als XML oder als gezipptes XML erstellt und gelesen werden, wird recht unterschiedlich gehandhabt. Auch bezüglich der Darstellung von LMS-Bedingungen (z. B. die maximale Dateigröße der studentischen Einreichung) gibt es Unterschiede. Einige Systeme speichern diese Informationen im ProFormA-Format, andere betrachten dies als nur zum LMS gehörig und bilden sie nicht im Format ab. Auch die Art und Weise, auf die die studentische Einreichung in die *submission* eingebettet wird, ist noch uneinheitlich. Es fällt auch auf, dass die Schwerpunkte in den einzelnen Systemen unterschiedlich gewählt werden. An der Ostfalia wird beispielsweise nur eine sehr rudimentäre Fassung des Bewertungsschemas umgesetzt während an der HsH der volle Umfang implementiert ist. Dagegen fehlt das Bewertungsschema in anderen Systemen komplett.

Da ProFormA 1.1 als Obermenge der Funktionalitäten der verschiedenen Systeme entstanden ist, erlaubt ProFormA 2.0 in einigen Bereichen mehrere sich gegenseitig ausschließende Varianten, aus denen nur mindestens eine implementiert werden muss. Zum Beispiel erwarten einige Systeme, dass Dateien in das XML-Format eingebettet sind, andere verarbeiten sie als Anhänge in einer ZIP-Datei. Bisher müssen Aufgaben im ProFormA-Format daher eventuell überarbeitet werden, nachdem sie von einem System exportiert und bevor sie in ein anderes System importiert werden sollen. Es gibt verschiedene Möglichkeiten, dieses Problem zu lösen, die hier in absteigender Komplexität aufgeführt werden: 1) alle Systeme müssen alle ProFormA 2.0-Aspekte lesen können, auch wenn sie selber nur eine Variante erzeugen. 2) Systeme kommunizieren über eine Middleware, die alle ProFormA 2.0 Aspekte unterstützt und die die XML-Dateien entsprechend konvertiert. 3) eine verpflichtende Untermenge wird definiert, die ProFormA 2.0.-kompatible Systeme unterstützen müssen. 4) Systeme veröffentlichen ein ProFormA-Profil, das beschreibt, was sie lesen und produzieren können. Ob zwei Systeme Aufgaben austauschen können, lässt sich dann anhand der Profile im Voraus bestimmen.

5 Zusammenfassung und Ausblick

In diesem Beitrag wurden Änderungen und Erweiterungen des ProFormA-Aufgabenaustauschformats zur Vereinfachung und zur besseren Unterstützung vor allem von LMS und Middleware vorgestellt, die aufgrund damit einhergehender neuer Anforderungen notwendig waren. Das Format definiert zu diesem Zweck nun Standards für Einreichungen (*submission*) und Antworten/Feedback (*response*) zu studentischen Abgaben, bietet Support für die Nutzung mehrerer Sprachen und enthält die Definition eines komplexen Bewertungsschemas. Zudem wurden Änderungen an Elementen, die eine Datei repräsentieren, vorgenommen. Zum Schluss wurde diskutiert, welche Bestandteile des Formats von einem System unterstützt werden müssen, um kompatibel zum ProFormA-Format zu sein.

Zukünftig soll der Fokus auf dem verstärkten Praxiseinsatz liegen. Zu diesem Zweck muss insbesondere die Anbindung an ein Repository vorangetrieben werden, über das Programmieraufgaben im ProFormA-Format bereitgestellt und ausgetauscht werden können. Hierzu ist es u. a. wichtig festzulegen, welche Metadaten (Abb. 1 *meta-data*) zu einer Aufgabe offiziell vom Format unterstützt werden. In diesem Zusammenhang wird die Nutzung einer Teilmenge des Learning Objects Metadata (LOM) Standards in Betracht gezogen. Zusätzlich muss abschließend definiert werden, ab wann ein System als ProFormA-kompatibel eingestuft werden kann.

Literatur

[Bo17] Bott, O.; Fricke, P.; Priss, U.; Striewe, M. (Hrsg.): Automatisierte Bewertung in der Programmierausbildung. Digitale Medien in der Hochschullehre, ELAN e.V., Waxmann, 2017.

[Ga17] Garmann, R.: Der Grader Graja. In [Bo17].

[Ga19] Garmann, R.: Ein Format für Bewertungsvorschriften in automatisiert bewertbaren Programmieraufgaben. In: Pinkwart, N.; Konert, J. (Hrsg.): DeLFI 2019 - Die 17. Fachtagung Bildungstechnologien, 2019.

[Mü17] Müller, O.; Strickroth, S.: Der Grader GATE. In [Bo17].

[Pr17] Priss, U.; Borm, K.: An Editor for the ProFormA Format for Exchanging Programming Exercises. In: Dritter ABP Workshop, Potsdam 2017.

[Ro17] Rod, O.: Integration mithilfe der Middleware ProFormA-Server. In [Bo17]

[St15] Strickroth, S. et al..: ProFormA: An XML-based exchange format for programming tasks. eleed, 11, 2015. https://eleed.campussource.de/archive/11/4138

[St17] Strickroth, S.; Müller, O.; Priss, U.: Ein XML-Austauschformat für Programmieraufgaben. In [Bo17].