Bachelorarbeit

Thema der Arbeit

David Bujok

Themensteller: Prof. Dr. Herbert Kuchen
Betreuer: Dipl.-Wirt.Inform. Claus Alexander Usener
Institut für Wirtschaftsinformatik
Praktische Informatik in der Wirtschaft

Inhaltsverzeichnis

1 Einleitung

Viele bekannte Lernplattformen unterstützen den Übungsbetrieb durch Funktionen zu Bereitstellung von Aufgabenblättern und zur Organisation studentischer Lösungen. Einige Systeme bieten darüber hinaus eine Elektronische Überprüfung der Lösung an [KG10].

Bei Moodle handelt es sich um ein Softwarepaket, welches einen konstruktivistischen Lehr- und Lernansatz unterstützt. [moof] Weltweit in 231 Ländern über 53.000 Seiten registriert [moof]

Moodle ist international die am weitesten verbreitete Lernplattform [See].

Die Westfälische Wilhelms-Universität Münster stellt zur Verbesserung des Lehrbetriebs eine Moodledistribution unter dem Namen Learnweb zur Verfügung.

Für die Vorlesung *Informatik I* wurde bereits ein Moodlemodul implementiert, welches die Möglichkeit bietet ????

Die Arbeit wird durch ein Grundlagenkapitel (Kapitel **??**) eingeleitet, in die zentralen Ideen von E-Assessment vorgestellt und die wesentlichen Merkmale der Lernplattform Moodle hervorgehoben werden. Bei der Vorstellung von Moodle wird auf die Pluginstruktur der Plattform eingegangen.

Im darauffolgenden Kapitel (Kapitel 4) wird das Modul EASy-DSBuilder vorgestellt. Hierbei wird auf die Funktionalität aus Benutzersicht und auf die Struktur aus technischer Sicht eingegangen.

2 E-Assessment

2.1 Einleitung

In der Hochschullehre nehmen Leistungsüberprüfungen einen integralen Bestandteil in Lehr- und Lernprozessen ein. Ein Augenmerk liegt hierbei auf der Identifizierung und Bewertung individueller Lernfortschritte [KG10, S. 23 f.]. Darunter wird insbesondere das Prüfen und Bewerten einzelner Übungen als Teil einer pädagogischen Handlungseinheit verstanden. Dabei können die verschiedenen Übungsphasen und -iterationen unterschiedliche Längen haben und inhaltliche Abhängigkeiten haben [KG10, S. 25]. Abbildung 1 zeigt den Lehr-Lern-Prozess eines Übungsbetriebs.

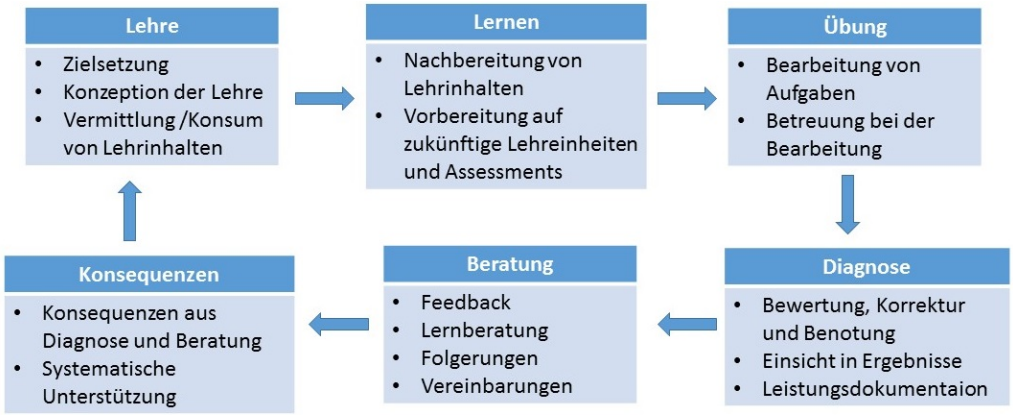

Abbildung 1: Iterative Lehr-Lern-Prozesse eines Übungsbetriebs

Im Regelfall beginnt ein Übungszyklus mit der Phase der Lehre. In dieser Phase vermittelt der Lehrende dem Studierenden die entsprechenden Inhalte. Anschließend hat der Studierende in der Lernphase die Möglichkeit das Vermittelte Wissen im Selbststudium zu vertiefen. In einer darauf folgenden Übung kann der Studierende sein theoretisches Wissen anhand geeigneter praktischer Übungen ausprobieren und festigen. Im Anschluss folgt die Diagnose der Übung, welche Korrektur und Benotung beinhaltet. Dies geschieht im Regelfall durch DozentInnen oder TutorInnen, kann aber auch durch KommilitonInnen erfolgen (Peer Review). Es sollte ein Feedback über die erbrachte Leistung und Ratschläge folgen. Optional kann aus den Ratschlägen eine Leistungsvereinbarung abgeleitet werden [KG10, S. 25 f.].

Auf Grund des Bologna-Prozesses, Sparmaßnahmen, Diskussionen über die Verwendung von Studiengebühren und steigende Studierendenzahlen wird im Bereich der Übungsangebote der Einsatz von E-Assessments vermehrt in Betracht gezogen [KP10, S. 9]. Unter e-Assessment versteht man das Spektrum der auf den neuen (elektronischen) IKT basierenden Verfahren der lehrzielbezogenen Bestimmung, Be-

urteilung, Bewertung, Dokumentation und Rückmeldung der jeweiligen Lernvoraussetzungen, des aktuellen Lernstandes oder der erreichten Lernergebnisse/ -leistungen vor, während (?Assessment für das Lernen?) oder nach Abschluss (?Assessment des Lernens?) einer spezifischen Lehr-Lernperiode [Blo06, S. 6]. E-Assessment-Systeme können des Weiteren Nutzen für Lehrende und Lernende bieten. Cook und Jenkins identifizierten neuen Vorteile. Bei den Wichtigsten handelt es sich um die Möglichkeiten des direkten Feedbacks und der sofortigen und objektiven Benotung. Außerdem bieten E-Assessment-Systeme einfache Skalierbarkeit und Wiederverwendbarkeit [CJ10, S. 3]. E-Assessments können hinsichtlich ihrer Hauptaufgabe in drei Typen unterteilt werden [CJ10, S. 8]:

- **Diagnostische Assessments** finden normalerweise zu beginn einer Lehrveranstaltung statt um mögliche Wissenslücken bei Teilnehmern aufzudecken und gegebenenfalls ein Nachbesserungsangebot zu schaffen.
- **Formative Assessments** ermöglicht Studierenden und Lehrenden einen Überblick über den aktuellen Lernstand zu erhalten. E-Assessment ermöglicht Studierenden weiterhin ein direktes Feedback.
- **Summative Assessments** bieten eine Bewertungsgrundlage über den Lernfortschritt eines Studierenden. Bei diesem Typen steht im Gegensatz des Feedbacks die Notengebung im Vordergrund.

Leistungüberprüfungen sind ein integraler Bestandteil der Lehr- und Lernprozesse. [KG10].

Wöchentliche Übungszettel –> theoretisch erlerntes Wissen durch Bearbeitung geeigneter Aufgaben reflektieren und verinnerlichen S.24

2.2 Das E-Assessment-System EASy der WWU Müsnter

3 Die Lernplattform Moodle

3.1 Was ist Moodle

Bei Moodle handelt es sich um ein weltweit anerkanntes Lernmanagement-System [Ger07, S. 33], das Lehrenden, Administratoren und Lernenden eine robuste, sichere und integrierte Plattform bereitstellen soll [mooa]. Der Name Moodle leitet sich von der Akronymisierung des Ausdrucks ***M****odular* ***O****bject* ***O****riented* ***D****ynamic* ***L****earning* ***E****nvironment* ab [Ger07, S. 33]. Moodle ist weiterhin eine frei verfügbares Softwarepaket, da es der GNU Public Lizenz unterliegt [SH09]. Software, welche unter einer GNU Public License vertrieben wird, darf kopiert, benutzt und weiterentwickelt werden. Eine einschränkende Bedingung ist, dass Änderungen oder Weiterentwicklungen den eben genannten Pflichten unterliegen, sie folglich auch veröffentlicht und Dritten zur Verfügung gestellt werden müssen [moof]. Die Plattform wird von einer weltweiten Gemeinschaft und von der Moodle Pty. Ltd. laufend weiterentwickelt. Vom australischen Moodle Erfinder Marign Dougiamas wird das Projekt zielgerichtet geleitet. Des weiteren gibt es ein Netzwerk professioneller Partnerunternehmen, welche Support und Beratung leisten [SH09, S. 12].

3.2 Moodle als Lernmanagement-System

Unter einem Lernmanagement-System (LMS) versteht man im wesentlichen ein Management-System für die Automatisierung und die Administration von Ausbildung. Insbesondere sollten LMS über folgende Funktionen verfügen [Sch05, S. 14]:

- Eine Benutzerverwaltung (Anmeldung mit Verschlüsselung)
- Eine Kursverwaltung (Kurse, Verwaltung der Inhalte, Dateiverwaltung)
- Eine Rollen- und Rechtevergabe mit differenzierten Rechten
- Kommunikationsmethoden (Chat, Foren) und Werkzeuge für das Lernen (Whiteboard, Notizbuch, Annotationen, Kalender etc.)

Moodle stellt diese Funktionen zur Verfügung. So besteht über die Website-Administration die Möglichkeit der Benutzerverwaltung [Ger07, S. 563 2 ff.] und der Kursverwaltung [Ger07, S. 588 ff.]. Bei der Rollen - und Rechtevergabe bietet Moodle flexible Möglichkeiten der Administration. So verfügt Moodle über vorgefertigte Basisrollen mit bestimmten Rechten, die einen Großteil der Anwendungsfälle abdecken. Für bestimmte Situationen können Rollen jedoch editiert oder neue Rollen erstellt werden [Ger07, S. 191]. Die Basisrollen des Systems sind [Ger07, S. 193]:

- *Kursverwalter:* Wer in einem Kontext *Kursverwalter* ist, kann einen *neuen Kurs erstellen* und in diesem unterrichten, weil er automatisch als *Trainer* eingetragen wird. Zu anderen Kursen im gleichen Kontext hat er aber keinen Zugriff.
- *Trainer*: Wer in einem Kontext *Trainer* ist, ist in sämtlichen Kursen dieses Kontextes als *Trainer* eingetragen und kann diese Bearbeiten
- *Trainer ohne Editorrecht*: ist Trainer in sämtlichen Kursen dieses Kontextes.
- *Teilnehmer/in*: ist Teilnehmer in sämtlichen Kursen dieses Kontextes, kann also auch Kurse mit Zugriffsschlüssel betreten.

Auf die Kommunikationsmethoden, die Moodle zur Verfügung stellt, wird in Kapitel 3.3 eingegangen.

Abbildung 2 zeigt die idealtypische Architektur eines LMS. Zu sehen ist, dass ein LMS über drei Schichten verfügt. Bei der untersten Schicht handelt es sich um die Datenbankschicht, in der alle Lernobjekte,Benuterdaten und andere gehalten werden. Die mittlere Schicht stellt Schnittstellen zur Verfügung. Die oberste Schicht stellt die Sicht bereit, über die über die seitens von Administratoren, Dozenten oder Studierenden auf Inhalte zugegriffen werden kann [Sch05, S. 11]. Im Kontext die-

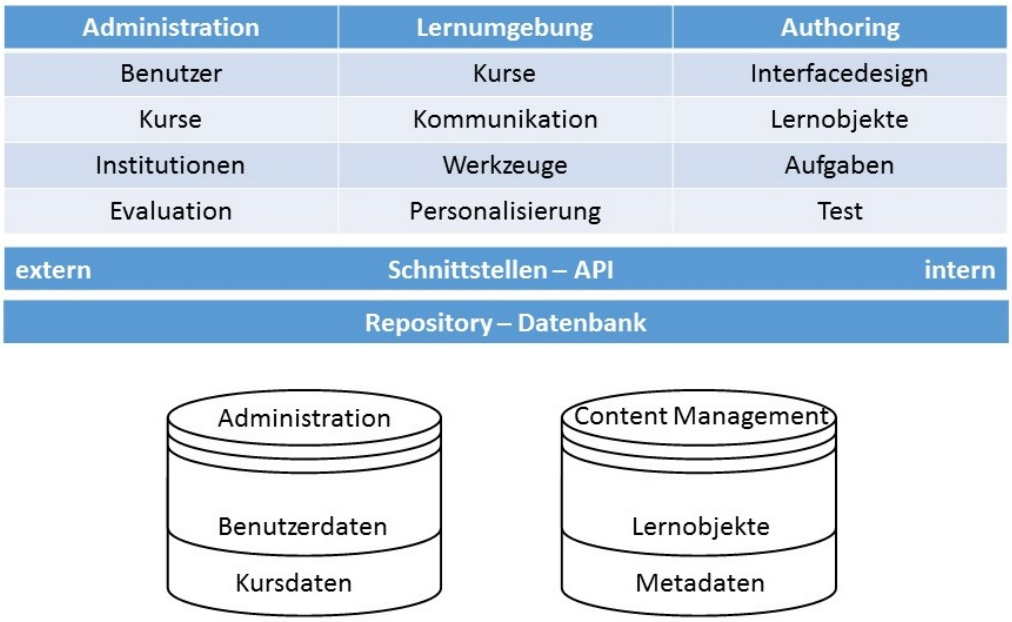

Abbildung 2: Idealtypische Architektur eines LMS [Sch05, S. 12]

ser Arbeit wird insbesondere verstärkt auf die Schnittstellenschicht eingegangen.

Das Kapitel 3.4 wird die Möglichkeit Einbindung von Modulen in Moodle erläutern. Das Kapitel **??** wird hingegen den Teilbereichen Aufgeben und Tests aus dem Bereich Authoring der Ansichtsschicht auseinandersetzen. *Es wird der Forschungsbereich E-Assessment vorgestellt, welcher sich mit Überprüfungen über Onlinemedien auseinandersetzt.*

3.3 Kommunikationsmethoden in Moodle

3.4 Modularität in Moodle

3.4.1 Pluginarten

3.4.2 Aufbau eines Plugins

Für jedes Plugin in Moodle muss eine bestimmte Datenstruktur implementiert werden. Diese besteht aus separaten Unterverzeichnissen und verpflichtenden Dateien. Des weiteren haben Entwickler die Möglichkeit weitere Dateien selbst zu gestalten [moob].

`/<modname>/backup`
Dieser Ordner dient zur Ablage aller Dateien, welche definieren, wie sich das Modul bei einem Backup oder einer Wiederherstellung verhalten soll [moob].

`/<modname>/db`

- **`/access.php`** In dieser Datei werden die so genannten *capabilities* für das Plugin definiert. *capabilities* beschreiben die Berechtigungen, welche eine Rolle in diesem Plugin zugeordnet bekommt. Eine Berechtigung ist beispielsweise das hinzufügen einer neuen Instanz diese Plugins zu einem Kurs [moob].
- **`/install.xml`** Diese Datei wird bei der Installation des Moduls benutzt. Sie definiert, welche Datenbanktabellen und -felder erstellt werden. Hierfür wird das XML-Format verwendet. Braucht das Modul keine weiteren Tabellen oder Spalten, so kann auf diese Datei verzichtet werden [moob].
- **`upgrade.php`** Auf Grund dessen, dass die Datei **`install.xml`** nur einmal während der Installation aufgeführt wird, braucht es eine Methode um die Datenbank nachträglich um Tabellen oder Spalten zu erweitern. Diese Funktionalität wird von dieser Datei bereitgestellt und kommt bei einem Update des Moduls zum Einsatz [moob].

/<modname>/lang

In diesem Ordner können alle *Strings* gespeichert werden, die im Modul benutzt werden sollen. Jede Sprache hat hierbei einen spezifischen Ordnernamen ('**/lang/de**' beispielsweise für die Sprache Deutsch). Die in diesem Ordner gespeicherte Datei muss in der Form **<modname>.php** benannt sein [moob].

/<modname>/pix

Dieser Ordner dient dazu das Logo des Moduls zu speichern, welches neben dem Modulname erscheint. Der Name des Logos muss **icon.gif** lauten. Weiterhin besteht die Möglichkeit weitere Bilder in diesem Ordner zu speichern [moob].

/<modname>

- **/lib.php** Diese Datei bietet eine Schnittstelle für die zu implementierenden Kernfunktionen. Kernfunktionen werden dazu benötigt, damit das Modul in Moodle integriert arbeiten kann. Diese Schnittstellen-Funktionen werden von Moodle nach einem bestimmten Ereignis im Prozessablauf aufgerufen, sofern diese vom Modul in der Datei **/lib.php** definiert wurden. Dabei ist jeder dieser Funktionen zunächst der Name des Moduls vorangestellt, gefolgt von einem Unterstrich und dem Funktionsnamen (**<pluginname>_core_function**). Diese Konvention ist deshalb so wichtig, da die Datei **/lib.php** keine Klasse definiert, welche Namenskonflikte verhindern. Es wird geraten Funktionalitäten, welche einen hohen Codeumfang haben, der Übersichtlichkeit halber in eine Datei namens **locallib.php** auszulagern. würde. [mood]

- **/mod_form.php** Diese Datei wird beim Hinzufügen oder Bearbeiten eines Kurses genutzt. Es enthält die Elemente welche im Editiermenü des Moduls zu sehen sind. Die in dieser Datei enthaltende Klasse muss der Namenskonvention nach in der Form **mod_<modname>_mod_form** benannt sein.

- **/index.php** Diese Datei wird von Moodle dazu genutzt, um auf Aktivitäten bei allen Instanzen dieses Moduls, welche einem bestimmten Kurs übergeben wurden, zu hören. Diese Datei ist spezifisch für diese Modulart *Activity Module*.

- **/view.php** Diese Datei wird bei der Erzeugung der Anzeige benötigt. Beim Aufrufen eines Moduls über die Kurssicht wird auf diese Datei verwiesen. Dabei wird dieser Datei die Instanz-ID übergeben, anhand welcher die Daten der

Instanz ausgewählt und angezeigt werden können. Diese Datei ist spezifisch für diese Modulart *Activity Module*.

- **`/version.php`** Diese Datei enthält die aktuelle Versionsnummer dieses Moduls. Außerdem enthält diese Datei weitere Attribute wie beispielsweise die Mindestanforderungen hinsichtlich der Moodleplattform.

4 Vorstellung des Moodleplugins EASy-DSBuilder

Der EASy-DSBuilder ist ein E-Assessent Tool, welches der Evaluation grundlegender Konzepte über Operationen (z.B. Suchen, Einfügen, und Entfernen) innerhalb der Datenstruktur *Binärbaum* dient [Use14].

Das Tool wurde speziell für die Lernplattform Moodle implementiert.

Diese Kapitel wird das Tool EASy-DSBuilder vorstellen. Hierbei wird zu erst in Kapitel 4.1 auf die Funktionalität aus Benutzersicht eingegangen. Anschließend erfolgt eine Erläuterung der Implementation (Kapitel

4.1 Funktionalität aus Benutzersicht

Im folgenden Kapitel wird die Funktionalität des Moodleplugins EASy-DSBuilder vorgestellt. Hierbei wird auf die beiden Sichten Student und Lehrender eingegangen.

Lehrender

Der Lehrende hat zwei Grundlegend Aufgaben. Zum einen ist er dafür verantwortlich, dass eine Aufgabe erstellt wird, zum anderen hat er die Möglichkeit, die Ergebnisse einzusehen, um beispielsweise Indikatoren zur Verbesserung der Lehre zu finden [Use14]. Wird eine neue Aufgabe erstellt, hat der Lehrende die Möglichkeit allgemeine Informationen wie den *Titel*, die *Beschreibung* und das *Fälligkeitsdatum* anzugeben. Unter *Source Files* kann der Lehrende über Drag-and-Drop seine eigene Implementierung einer Datenstruktur zu dem Moodleplugin hinzufügen. Hierzu muss er jedoch eine Wrapper auf Basis eines Interfaces implementieren, welches die verlangten Voraussetzungen erfüllt. Diese Wrapperklasse muss anschließend vom Lehrenden als Hauptklasse eingestellt werden. Auf die Funktionalität der Wrapperklasse aus technischer Sicht wird im Kapitel 4.2.1 näher eingegangen. Des weiteren kann der Lehrende eine Feedback aktivieren. Die genau Funktionalität des Feedbacks wird im Absatz der Studentensicht erläutert.

Studierender

Der Studierende verfügt über zwei Ansichten. Zum einen die Übersichtsansicht, zum anderen die Bearbeitungsansicht. Nachdem der Studierende sich in das Plugin eingewählt hat, ist ist Übersichtsansicht über den bisherigen Verlauf des Assessments zu sehen. In dieser Übersicht ist der Abgabestatus, der Bewertungsstand, der Abgabezeitpunkt und die verbliebene Zeit zu sehen (vergl. Abb. **??**). Über den Button *Aufgabe bearbeiten* gelangt der Studierende zum Editor, in dem die Aufgabe bearbeitet werden kann.

Die Bearbeitungsansicht ist in drei grundlegende Abschnitte unterteilt. Den oberen Teil der Ansicht bildet ein Überblick über den aktuellen Schritt. Dieser Überblick beinhaltet den Fortschritt der Aufgabe, die Nummer des aktuellen Schritts und den aktuellen Arbeitsauftrag. Im mittleren Teil der Sicht befindet sich der Editor, in dem der Studierende die Aufgabe bearbeiten kann. Im oberen linken Bereich des Editor befinden sich drei Knöpfe (vergl. Abb. **??**.1), über welche der Editiermodus ausgewählt werden kann. Der 1. Knopf ermöglicht das verschieben von Konten im Editor, der zweite Knopf ermöglicht das Ziehen von Kanten zwischen zwei Knoten, und der dritte Knopf ermöglicht das Entfernen von Konten.

Der DragandDropGrafikeditor enthält zwei bearbeitbare Elemente, die Knoten und die Kanten. Über Manipulation dieser Elemente sollen Studierende den Umgang mit Datenstrukturoperationen erlernen. Hierbei kann der Studierende Operationen wie das Einfügen in oder das Löschen aus einer Datenstruktur praktizieren. In der **momentanen** Version des EASyDSBuilders beginnt jeder Schritt mit dem Ergebnisbaum des zuvor eingereichten Schrittes oder einem Initiierngsbaum wenn, es sich um den ersten Schritt handelt. Auf der linken Seite des Editors wird der einzufügende Knoten bereitgestellt. Die Aufgabe des Studierenden ist es, diesen Knoten an der richtigen Stelle in den Baum einzufügen. **Erläuterung der Möglichkeiten von Manipulationen** Nachdem der Studierende seine Veränderungen vorgenommen hat, kann er über den Knopf *Syntax prüfen* den Baum ausbalanciert anzeigen lassen. Auf diese Weise kann der Studierende überprüfen, ob die Anwendung den Baum im Sinne des Studierenden verarbeitet hat. Entspricht die überprüfte Struktur nicht der Struktur eines Baumes, **genauere Definiton** bekommt der Studierende eine Fehlermeldung mit Hinweis über die Fehlerquelle.

Hat der Lehrende bei der Einrichtung des DSBuilders die Option *direktes Feedback* eingestellt, erscheint im Falle einer falschen Eingabe ein Feedbackfeld unterhalb des Editors. In diesem Feedbackfeld wird zu erst ein Informationstext angezeigt, welches das richtige Vorgehen in dem zuvor eingereichtem Schritt beschreibt. Unterhalb dieses Informationstextes ist der korrekte Baum zu sehen. Die falsch eingeordneten Knoten sind rot markiert.

4.2 Umsetzung aus technischer Sicht

Das gesamte System um den EASy-DSBuilder besteht backendseitig aus zwei separaten Systemen. Zum einen gibt es das eigentliche Moodleplugin, welches in eine bestehenden Moodelplattform integriert werden kann, zum anderen gibt es einen Datenstruktur-Verarbeitungsservice, welcher als Webservice implementiert ist.

Das Moodleplugin hat die Möglichkeit über die Moodle-API Daten in einer SQL-Datenbank - beispielsweise einem MySQL-Server - zu hinterlegen. Die Kommunikation zwischen dem Moodleplugin und dem Webservice läuft über das SOAP-Protokoll. Der Webservice ist als WildFly Application Server implementiert und unterliegt somit dem Java-EE7-Standard [Gre]. In Abbildung 3 ist dargestellt, wie die unterschiedlichen Technologien in einander greifen.

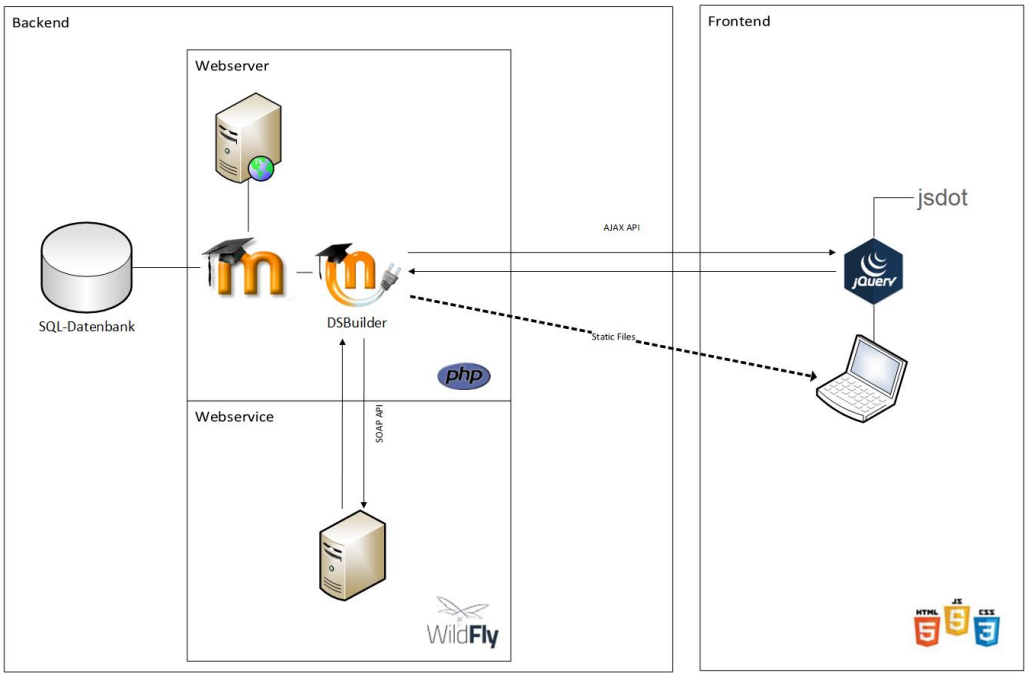

Abbildung 3: Technischer Überblick

Der Datenstruktur-Verarbeitungsservice hat die Aufgabe Datenstrukturen mit Hilfe von Die Separierung des Systems erfolgt aus den Risiken, dass der Code schädlich sein oder eine schlechte Ausführungsleistung aufweist kann. Durch die Trennung der beiden Systeme kann in beiden Fällen Zusammen- oder Performanceeinbrüchen der gesamten E-Learning-Plattform vorgebeugt werden. Weiterhin kann so Datendiebstahl vorgebeugt werden, da in der Verarbeitungsumgebung keine nutzerbezogenen Daten verarbeitet werden. Bei Ausfall des Verarbeitungsservices ist jedoch das Aufrufen eines nächsten Schrittes nicht mehr möglich [Use14].

Auf Clientseite wird HTML mit CSS und JavaScript verwendet, um das Plugin für den Benutzer darstellen zu können. Als JavaScript-Frameworks wird jQuery und und als JavaSrcipt-Applikation wird jsdot eingesetzt. Über jQuery ist die Kommunikation mit dem Moodleplugin über das AJAX-Protokoll organisiert. Jsdot dient als Grapheditor.

4.2.1 Datenstruktur-Verarbeitungsservice

Der Datenstruktur-Verarbeitungsservice kompiliert und führt den vom Lehrenden bereitgestellten Code aus. Er ist als Webservice implementiert und kann somit von einem anderen Server aus bereitgestellt werden. Die Ausführung des Codes ist vor jedem Einfügen oder Löschen, das von einem Studierenden durchgeführt wird, notwendig.

Auf der Grundlage des bisherigen, eingereichten Schritts berechnet die Ausführungsumgebung den nächsten die Ausführungsumgebung der nächsten Betriebswert (Taste, die eingefügt oder gelöscht wird), die erwartete Lösung und die entsprechende detaillierte Rückmeldungen.

4.2.2 Moodleplugin backendseitig

Das backendseitige Moodleplugin besitzt die grundlegende Struktur eines Moodleplugins, wie sie in Kapitel 3.4.2 dargestellt wurde. Die weiteren, für die Funktionalität des Moduls wichtigen Dateien sind die Dateien **`renderer.php`** und **`renderable.php`**, welchen DOM-Code generieren, die Dateien **`ajax_request.php`** und **`ajax_helper.php`**, welche das Handling von AJAX-Anfragen übernehmen und die Datei **`lib/js_dot_convert.php`**, welche die Funktionen zur Verarbeitung der Datenstrukturen zur Verfügung stellt. Im weiteren Verlauf dieses Abschnittes werden diese drei Funktionalitäten tiefgehender erläutert und Codeausschnitte exemplarisch vorgestellt.

Generierung des DOM-Codes

Die Datei, welche für die Generierung des DOM-Codes verantwortlich ist, ist die **`renderer.php`**. Die Datei **`renderable.php`** implemeniert hingegen ein Interface, welches für die Verwendung des moodleinternen Renderers notwendig ist [mooe]. Die in der Datei **`renderer.php`** enthaltende Klasse **`mod_dsbuilder_renderer`** enthält Funktionen zum erstellen der für den DSBuilder benötigten Ansichten. So können Übersichten über laufende oder eingereichte Abgaben oder Notentabellen generiert werden. Ebenso können Ansichten zur Aufgabenbearbeitung generiert werden. Hier-

```
1  // call initialize graph function
2 $this->page->requires->js_init_call('M.mod_dsbuilder.
     init_jsdot_show', array(
3                 $div_id_graph_2,
4                 json_encode($com_object->graph)
5 ), false, self::get_jsdot_module_info());
```

Quellcode 1: Aufruf zur Initialisierung eines JSDot-Graphs

bei übernimmt die Initiierung des Grapheneditors, welche im Quellcode 1 dargestellt wird, eine zentrale Rolle. Es wird eine Hilfsfunktion des Moodle-API [mooc] verwendet, welche die frontendseitige JavaScript-Funktion zur Initialisierung des Grapheneditors anstößt. Die frontendseitige Funktionalität wird im Kapitel 4.2.3 vertieft.

Modulinterne AJAX-API

Zur asynchronen Datenübertragung zwischen Browser und Server stellt das Moodleplugin eine AJAX Api zur Verfügung. Der Quellcode 2 zeigt einen Codeausschnitt aus der **`ajax_request.php`**, in dem die möglichen Aktionen definiert sind, die nach einer AJAX-Anfrage durchgeführt werden können. Hierbei handelt es sich um die Funktionen, welche über die Knöpfe unterhalb des Grapheneditors (vergl. Kapitel 4.1, Abschnitt Studierender) angestoßen werden können. Explizit handelt es sich um die Funktionen *Syntax prüfen* (Quellcode 2, Z. 2), *Speichern und weiter* (Quellcode 2, Z. 6) und *Letzten Schritt wiederholen* (Quellcode 2, Z. 10). Die jeweils angestoßenen Funktionen sind in der **`ajax_helper.php`** definiert. Von dort aus werden weitere

```
1  try {
2      if ($action === DSBUILDER_AJAX_ACTION_CHECK) {
3          $jsdot_graph_raw = required_param('jsdot_graph', PARAM_TEXT
               );
4          $result = $dsbuilder_ajax->action_check_valid_graph(
               $jsdot_graph_raw);
5          $dsbuilder_ajax->add_to_log($action, $step_no);
6      } elseif ($action === DSBUILDER_AJAX_ACTION_NEXT_STEP) {
7          $jsdot_graph_raw = required_param('jsdot_graph', PARAM_TEXT
               );
8          $result = $dsbuilder_ajax->action_submit_current_step(
               $jsdot_graph_raw);
9          $dsbuilder_ajax->add_to_log($action, $step_no);
10     } elseif ($action === DSBUILDER_AJAX_DELETE_LAST_STEP) {
11         $result = $dsbuilder_ajax->action_delete_last_step();
12         $dsbuilder_ajax->add_to_log($action, $step_no);
13     }
14 }
```

Quellcode 2: AJAX API

Funktionen zur Datenstrukturverarbeitung in der Klasse **`jsdot_graph`** angestoßen. Diese Funktionen werden im nächsten Abschnitt vertiefender behandelt.

Die Datenhaltung

Das Modul DSBuilders benötigt vier Entitäten für seine Datenhaltung. Es handelt sich um die Entitäten *DSBuilder*, *Assigment File*, *Submission* und *Submission Step*. Die Abbildung 4 zeigt das Datenmodell des Moduls. Nachdem das Modul neu in

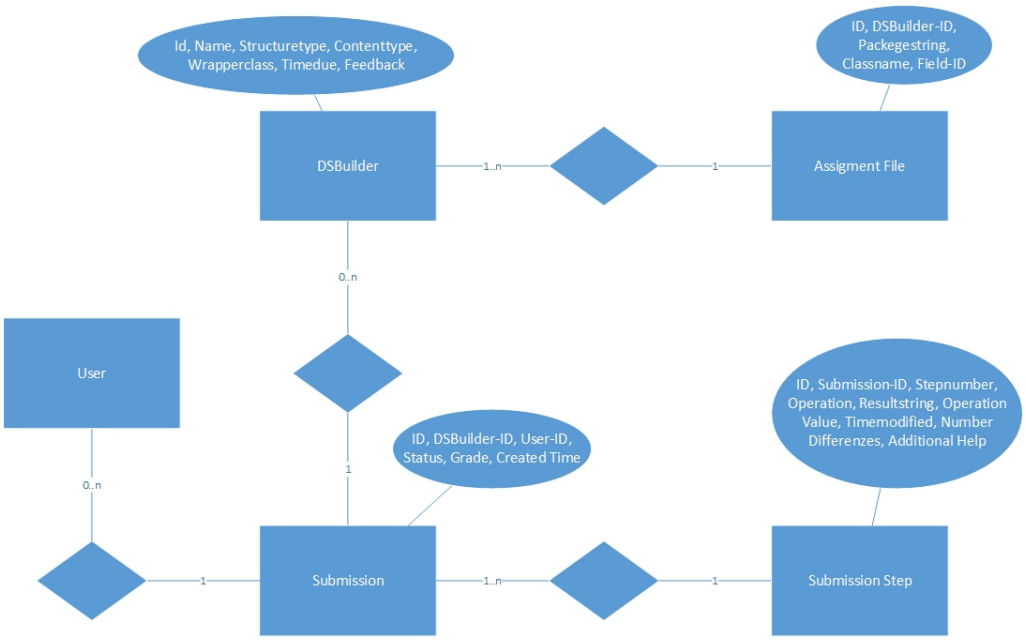

Abbildung 4: Datenmodell des DSBuilders

einem Kurs initialisiert worden ist, wird ein neues Datum der Entität *DSBuilder* angelegt. Die der neuen Instanz des Moduls vom Lehrenden zugewiesenen Javadateien werden als Datum der Entität *Assigment File* gespeichert. Sobald Studierende die Instanz nutzen, wird für jeden Studierenden mit Vermerk auf die Instanz ein neues Datum der Entität *Submission* angelegt. Jeder *Submission* werden *Submission Steps* zugeordnet. Sie beinhalten Informationen über die jeweiligen ausgeführten Schritte.

Die Datenstrukturverarbeitung

In der Datei **`lib/js_dot_convert.php`** liegt die Funktionalität der Datenstrukturverarbeitung. In ihr sind vier Klassen implementiert, von denen die Klasse **`jsdot_graph`** eine Schnittstelle zur Konvertierung einer Datenstruktur zwischen dem Grapheneditor im Frontend und der Datenhaltung im Backend bietet. Abbildung 5 zeigt ein UML-Klassendiagramm der vier Klassen.

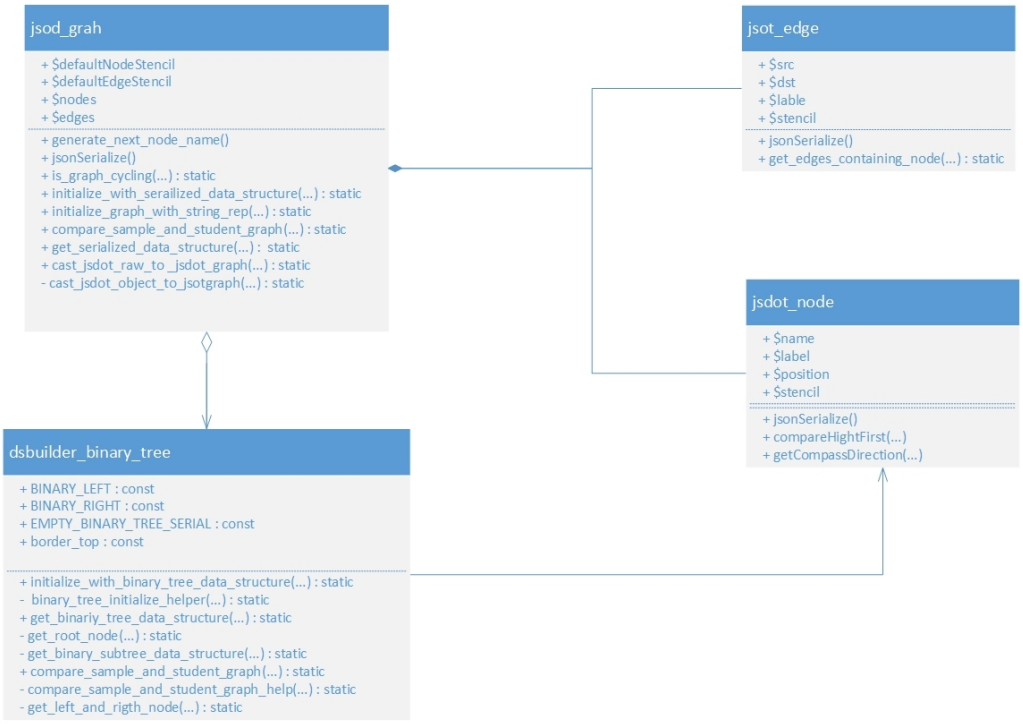

Abbildung 5: UML js_dot_convert

4.2.3 Moodleplugin frontendseitig

Die zentrale Datei im Frontend ist die **`dsbuilder.js`**. Sie organisiert die Kommunikation mit dem Grapheneditor. Der Codeauszug 3 zeigt die Initialisierung eines neuen jsdot-Graphen. Des weiteren stellt diese Datei die AJAX-Kommunikation zur

```
1 init_jsdot_edit : function(e, divname, jsongraph, structureType) {
2         this.jsdot_graphs[divname] = new JSDot(divname, {
3                 mode : "editor",
4                 json : jsongraph
5         });
6 },
```

Quellcode 3: Initiierung eines JsDot-Graphs

Verfügung.

5 Änderungsanforderungen des DSBuilders

In diesem Kapitel werden die Anforderungen an die Erweiterungen des EASy-DSBuilders vorgestellt und näher erläutert. Die Hauptanforderung lautet:

> Der EASy DSBuilder soll um die Datenstruktur B-Baum erweitert werden.

Aus dieser Hauptanforderung lassen sich weitere Unteranforderungen Ableiten.

- editirbare graphische Oberfläche zur Erstellung von B-Bäumen
- Funktionalität soll beibehalten werden:
  - Kommunikation Moodle <—> Web-Server
  - Schritte werden gespeichert
  - Eingabe überprüfen
  - Feedback
  - Schritt zurück
- Lehrender: Auswahl zwischen Typ

6 Umsetzung der Änderungsanforderungen

In den folgenden Abschnitten wird die Umsetzung der Änderungsanforderungen beschreiben. Dies bedeutet, dass erläutert wird, an welchen Stellen Erweiterungen vorgenommen werden mussten, und beschrieben wird, wie diese Änderungen in der Implementierung umgesetzt wurden.

6.1 Umbau des EASy-DSBuilders

Dieses Kapitel stellt die Stellen vor, an denen Änderungen vorgenommen werden mussten und erläutert die Ursachen, auf Grund derer die Änderungen vorgenommen werden mussten. In Kapitel 6.2 wir auf die Implementierungsdetails eingegangen. Dieses Kapitel ist nach Schichten strukturiert.

6.1.1 Datenstrukturverarbeitungs-Webservice

An der Funktionalität des Datenstrukturverarbeitungs-Webservice sollte nichts geändert werden. Fehler, die während der Entwicklung auftraten, wurden behoben. Zur Endwicklung des Moodlemoduls musste jedoch die Datenstruktur B-Baum in Java implementiert werden, damit die Funktionen des Webservices bereitgestellt werden konnten.

6.1.2 Moodlemodul backendseitig

Die für das backendseitige Moodlemodul wichtigen Funktionalitäten ist die in Kapitel 4.2.2 beschriebene Generierung des DOM-Codes, das Handling von AJAX-Anfragen und das Verarbeiten und zur Verfügung Stellen der Datenstruktur. Die weitere Gliederung dieses Abschnitts ist an die Gliederung des Kapitels 4.2.2 angelehnt.

Generierung des DOM-Codes

6.2 Ausgewählte Implementierungsdetails

Literatur

[Blo06] E. Bloh. Methodische Formen des E-/Online-Assessment. *Unveröffentlichtes Manuskript*, 2006.

[CJ10] Julian Cook and Vic Jenkins. Getting started with e-assessment. *University of Bath*, 2010.

[Ger07] Fredi Gertrsch. *Das Moodle 1.8 Praxisbuch.* Addison-Wesely Verlag, 2007.

[Gre] Jason Greene. WildFly News.

[KG10] Herbert Kuchen and Susanne Gruttmann. Computerunterstützter Übungsbetrieb im Informatikstudium. *Zeitschrift für e-lerning*, 1:23–35, 2010.

[KP10] Gerd Kortemeyer and Riegler; Peter. Large-Scale E-Assessments, Prüfungsvor- und Nachbearbeitung. *Zeitschrift für e-lerning*, 1:8–22, 2010.

[mooa] About Moodle.

[moob] Activity Modules.

[mooc] JavaScript usage guide.

[mood] NEWMODULE Documentation.

[mooe] Output renderers.

[moof] Was ist Moodle.

[Sch05] Rolf Schulmeister. *Lernplattformen für das virtuelle Lernen.* Oldenburg Verlag München Wien, 2. edition, 2005.

[See] Robert Seetzen. Die freie Lernplattform Moodle.

[SH09] Christoph Scheb and Ralf Hilgenstock. *moodle einführen.* DIALOGE Verlag, 2009.

[Use14] Claus A Usener. EASy-DSBuilder : Automated Assessment of Tree Data Structures in Computer Science Teaching. 2014.

Ich versichere hiermit, dass ich meine Diplomhausarbeit „*Thema der Arbeit*“ selbstständig und ohne fremde Hilfe angefertigt habe, und dass ich alle von anderen Autoren wörtlich übernommenen Stellen wie auch die sich an die Gedankengänge anderer Autoren eng anlehnenden Ausführungen meiner Arbeit besonders gekennzeichnet und die Quellen zitiert habe.

Münster, den (Abgabedatum)

David Bujok