Bachelorarbeit

Erweiterung eines E-Learning-Moduls zum Assessment von B-Baum-Datenstrukturen

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

„E-Assessment ist eines der Schlagworte der letzten Jahre“ [Rue10, S. 11]. Unter dieser Form des Assessments versteht man das Nutzen von Hardware und Software im Prüfungsprozess. So wurden im deutschsprachigem Raum für E-Assessment Synonyme wie Online-Prüfungen oder computergestützte Prüfungen verwendet [Rue10, S. 13]. Zwei Gründe sprechen für diese Assessmentform als Mittel des Leistungsnachweises im Prüfungsalltag. Durch Effizienzsteigerung kann knappen Raum- und Personalressourcen vorbeugen. Außerdem können neue didaktische Konzepte wie bessere Visualisierung oder neue Aufgabenformate, die Computer erst ermöglichen, angeboten werden [Rue10, S. 11 ff.]. Die meisten Systeme, die E-Assessments anbieten, stellen jedoch nur einfache Aufgabentypen wie Multiple-Choice-Aufgaben zur Verfügung. Da insbesondere technische Disziplinen wie die Informatik auf Aufgabentypen zur Erhebung von kognitiven Fähigkeiten angewiesen sind [KG10, S. 24], entwickelte der Lehrstuhl für Praktische Informatik in der Wirtschaft der WWU Münster das E-Assessment-Tool EASy-DSBuilder zum Assessment von AVL-Baum-Datenstrukturen für die Moodle-Distribution Learnweb [Use14, S. 1]. Bei Moodle handelt es sich um ein weltweit stark verbreitetes Lernmanagement-System [Ger07, S. 33], das Lehrenden, Administratoren und Lernenden eine robuste, sichere und integrierte Plattform bereitstellen soll [Moo15a]. Das System bietet die Möglichkeit der individuellen Anpassung an spezifische Anwendungssituationen [Moo15j].

Das Ziel dieser Arbeit ist die Entwicklung und Vorstellung eines E-Learning-Moduls zum Assessment von B-Baum-Strukturen. Im Zentrum steht hierbei das Moodle-Modul EASy-DSBuilder. Insbesondere stellt diese Arbeit die Erweiterung dieses Moduls um des Assessments von B-Baum-Datenstrukturen vor. Grundlagen dieser Arbeit (Kapitel 2) sind eine Einführung in die Lernplattform Moodle, die Formen des E-Assessments, und schließlich die Definition einer B-Baum-Datenstruktur. Das darauf folgende Kapitel 3 stellt das Moodle-Modul EASy-DSBuilder vor. Hierbei wird auf die Funktionalität aus Benutzersicht und auf die Struktur aus technischer Sicht eingegangen. Im Kapitel 4 werden die Anforderungen an ein Tool zum Assessment von B-Baum-Datenstrukturen vorgestellt. Insbesondere werden im Kontext des EASy-DSBuilder Änderungsanforderung an dieses Modul zur Umsetzung des Assessments von B-Baum-Datenstrukturen dargelegt. Anschließend wird die Umsetzung der Anforderungen betrachtet. Hierbei werden die entwickelten Funktionalitäten beleuchtet und es wird ein Einblick auf ausgewählte Implementierungsdetails gegeben. Abschließend wir ein Fazit des Ergebnisses dieser Arbeit betrachtet und ein Ausblick auf mögliche weitere Entwicklungen des Moduls gegeben.

2 Grundlagen

Im folgenden Kapitel werden die für diese Arbeit wichtigen Grundlagen vorgestellt. Insbesondere handelt es sich um Einführungen in die Konzeption der Lernplattform Moodle, das E-Assessment und die B-Baum-Datenstruktur.

2.1 Die Lernplattform Moodle

Der folgende Abschnitt erläutert, um welche Art System es sich bei der Lernplattform Moodle handelt und stellt weitere wichtige Eigenschaften dieser Plattform vor.

2.1.1 Das Lernmanagement-System Moodle

Bei Moodle handelt es sich um ein weltweit verbreitetes Lernmanagement-System [Ger07, S. 33], das Lehrenden, Administratoren und Lernenden eine robuste, sichere und integrierte Plattform bereitstellen soll [Moo15a]. 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 darüber hinaus ein 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 [Moo15j]. 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, die Support und Beratung leisten [SH09, S. 12].

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. Ferner 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 die Rolle *Trainer* hat, ist in sämtlichen Kursen dieses Kontextes als *Trainer* eingetragen und kann diese bearbeiten.

- *Trainer ohne Editorrecht*: Diese Rolle ist Trainer in sämtlichen Kursen dieses Kontextes, kann diese aber nicht bearbeiten.

- *Teilnehmer/in*: Diese Rolle ist Teilnehmer in sämtlichen Kursen dieses Kontextes, kann also auch Kurse mit Zugriffsschlüssel betreten.

Die Kommunikationsmethoden, die von Moodle zur Verfügung gestellt werden, sind Chats [HK11, S. 47 ff.] und Foren [HK11, S. 51 ff.]. Des weiteren können noch externe Werkzeuge wie GeoGebra mit Moodle verlinkt werden [HK11, S. 63 f.].

Abbildung 1 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, Benutzerdaten und andere Daten gehalten werden. Die mittlere Schicht stellt Schnittstellen zur Verfügung. Die oberste Schicht stellt die Sicht bereit, über die seitens der Administratoren, Dozenten oder Studierenden auf Inhalte zugegriffen werden kann [Sch05, S. 11].

Im Kontext dieser Arbeit wird verstärkt auf die Schnittstellenschicht eingegangen. Das Kapitel 2.1.2 wird eine Möglichkeit zur Einbindung von Modulen in Moodle erläutern. Kapitel 2.2 wird sich indes auch mit den Teilbereichen Aufgaben und Tests aus dem Bereich Authoring der Ansichtsschicht auseinandersetzen: So wird das Kapitel 2.2 zeigen, dass Aufgaben und Test für E-Assessments eine wesentliche Rolle spielen und diese in diesem Kontext über ein LMS abgewickelt werden können.

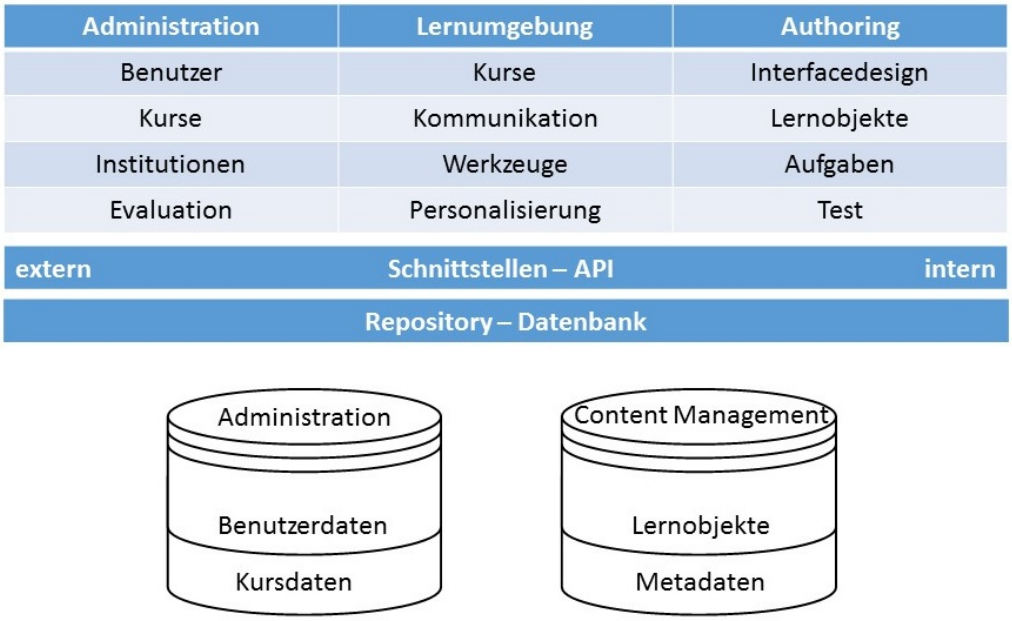

Abbildung 1: Idealtypische Architektur eines LMS [Sch05, S. 12]

2.1.2 Modularten

Moodle verfügt über eine umfangreiche Anzahl an Modularten, von denen eine Auswahl in diesem Abschnitt näher beschrieben wird. Da es sich bei dem EASy-DSBuilder um ein Activity-Modul handelt, wird die Struktur dieses Moduls tiefgehender vorgestellt.

Blöcke

Bei Blöcken handelt es sich um Module, welche über eine einfache Art Interface verfügen, sodass sie auf jeder Moodle-Seite hinzugefügt werden können. Das Layout einer Moodle-Seite bietet links und rechts von der Hauptinhaltsspalte jeweils Spalten zum Einbinden von Blöcken. Hauptaufgabe von Block-Modulen ist die Darstellung zusätzlicher Inhalte oder Daten, die über verschiedene Sichten aufgearbeitet werden können. Weiterhin wird die Bearbeitung von Daten über ein Interface ermöglicht [Moo15g].

Themem

Mit diesem Modul lässt sich die Darstellung einer Moodle-Distribution bearbeiten. Hierbei können Themen-Module auf verschiedene Ebenen angewandt werden. Ebenen sind beispielsweise die gesamte Moodle-Seite, Kurskategorien oder einzelne Kursseiten. Über diese Module schafft Moodle eine klare Trennung zwischen Präsentationsschicht und funktionaler Schicht [Moo15g].

Kursformate

Dieses Modul dient zur Anpassung der Struktur von Kursen. In erster Linie wird in diesen Modulen definiert, wie die Hauptseite eines Kurses aufgebaut ist, wie der Navigationsbaum innerhalb eines Kurses erstellt wird und wie die einzelnen Kursbereiche strukturiert sind [Moo15d].

Lokale Module

Bei lokalen Modulen handelt es sich um Module, die eingesetzt werden können, wenn kein anderes Modul passend erscheint. Typische Anwendungsfälle für diese Art von Modul sind das Einbinden externer Systeme wie Webserver oder das Erweitern des Navigationsblocks um eigene Menüpunkte [Moo15f].

Activity-Module

Activity-Module stellen in erster Linie Kursaktivitäten wie Foren, Hausarbeiten, Quizze und Workshops zur Verfügung [Moo15b]. Weiterhin bietet Moodle eine Schnittstelle, sodass Activity-Module selbst entwickelt werden können. Es muss jedoch eine bestimmte Datenstruktur implementiert werden. Diese besteht aus separaten Unterverzeichnissen und verpflichtenden Dateien. Des Weiteren haben Entwickler die Möglichkeit, weitere Dateien selbst zu gestalten [Moo15c]. Die verpflichtende Struktur sieht wie folgt aus:

`/<modname>/backup`

Dieser Ordner dient zur Ablage aller Dateien, welche definieren, wie sich das Modul bei einem Backup oder einer Wiederherstellung verhalten soll [Moo15c].

`/<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 dieses Moduls zu einem Kurs [Moo15c].
- **`/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 [Moo15c].
- **`upgrade.php`** Auf Grund dessen, dass die Datei **`install.xml`** nur einmal während der Installation aufgeführt wird, braucht es eine Methode, um die Daten-

bank nachträglich um Tabellen oder Spalten zu erweitern. Diese Funktionalität wird von dieser Datei bereitgestellt und kommt bei einem Update des Moduls zum Einsatz [Moo15c].

/<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 [Moo15c].

/<modname>/pix

Dieser Ordner dient zum Speichern des Logos des Moduls, welches in der Modulauswahl neben dem Modulnamen erscheint. Der Name des Logos muss **icon.gif** lauten. Ferner besteht die Möglichkeit weitere Bilder, die im Modul verwendet werden sollen, in diesem Ordner zu speichern [Moo15c].

/<modname>

- **/lib.php** Diese Datei bietet eine Schnittstelle für die zu implementierenden Kernfunktionen. Diese werden 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 einzuhalten, da die Datei **/lib.php** keine Klasse definiert, welche Namenskonflikte verhindert. Wird sie nicht eingehalten, kann es zu Uneindeutigkeiten kommen. Es wird geraten, Funktionalitäten, die einen hohen Codeumfang haben, der Übersichtlichkeit halber in eine Datei namens **locallib.php** auszulagern [Moo15h].

- **/mod_form.php** Diese Datei wird beim Hinzufügen oder Bearbeiten eines Kurses genutzt. Es enthält die Elemente, die 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, auf Aktivitäten bei

allen Instanzen dieses Moduls, welche einem bestimmten Kurs übergeben wurden, zu hören. Diese Datei ist spezifisch für diese Modulart.

- **`/view.php`** Diese Datei wird bei der Erzeugung der Anzeige benötigt. Beim Aufrufen eines Moduls über die Kurssicht wird auf sie verwiesen. Dabei wird ihr die Instanz-ID übergeben, anhand welcher die Daten der Instanz ausgewählt und angezeigt werden können. Diese Datei ist spezifisch für diese Modulart.
- **`/version.php`** Diese Datei enthält die aktuelle Versionsnummer dieses Moduls genauso wie weitere Attribute, beispielsweise die Mindestanforderungen hinsichtlich der Moodle-Plattform.

2.2 E-Assessment

Dieses Kapitel erläutert, worum es sich bei E-Assessments handelt. Hierzu wird zuerst eine allgemeine Einleitung gegeben. Anschließend werden die beiden Kategorien, E-Assessment als Teilbereich des Lernprozesses und E-Assessment als Methode des Leistungsnachweises, tiefgehender erläutert.

2.2.1 Einleitung

Aufgrund 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, Beurteilung, Bewertung, Dokumentation und Rückmeldung der jeweiligen Lernvoraussetzungen, des aktuellen Lernstandes oder der erreichten Lernergebnisse/ -leistungen, 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 neun 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 anhand der zuvor gegebenen Definition hinsichtlich ihrer Hauptaufgabe in drei Typen unterteilt werden [CJ10, S. 8]:

- **Diagnostische Assessments** finden normalerweise zu Beginn einer Lehrveranstaltung statt. Sie dienen der Aufdeckung möglicher Wissenslücken bei Teilnehmern und bieten die Möglichkeit, gegebenenfalls ein Nachbesserungsangebot zu schaffen.
- **Formative Assessments** ermöglichen Studierenden und Lehrenden, einen Überblick über den aktuellen Lernstand zu erhalten. Eine Einschätzung wird Studierenen insbesondere über ein zeitnahes Feedback gegeben.
- **Summative Assessments** bieten eine Bewertungsgrundlage über den Lernfortschritt eines Studierenden. Bei diesem Typen steht im Gegensatz zum Feedback die Notengebung im Vordergrund.

Es wird in den folgenden Abschnitten auf das formative Assessment eingegangen, indem E-Assessments als Teilbereich des Lernprozesses vorgestellt werden. Anschließend werden summative Assessments behandelt, indem tiefgehender auf den Aspekt des Leistungsnachweises im Zusammenhang mit E-Assessment eingegangen wird.

2.2.2 E-Assessment als Teilbereich des Lernprozesses

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 und inhaltliche Abhängigkeiten haben [KG10, S. 25]. Abbildung 2 zeigt den Lehr-Lern-Prozess eines Übungsbetriebs.

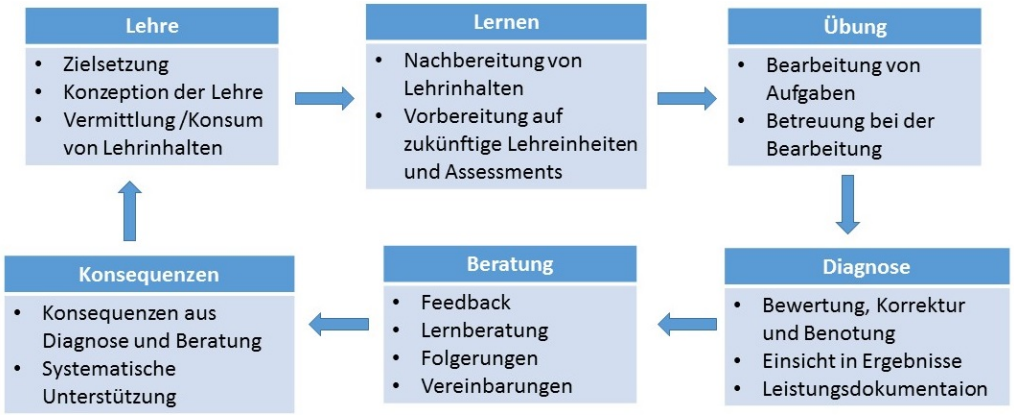

Abbildung 2: Iterative Lehr-Lern-Prozesse eines Übungsbetriebs [KG10, S. 25]

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.].

Weiterhin sind Leistungsüberprüfungen ein integraler Bestandteil der Lehr- und Lernprozesse. So sollen wöchentliche Übungszettel dazu beitragen, dass das von Studierenden erlernte theoretische Wissen durch Bearbeitung geeigneter Aufgaben reflektiert und verinnerlicht wird. Insbesondere gestaltet sich das Halten von klassischen Präsenzübungen bei knappen finanziellen Ressourcen schwer. Eine Möglichkeit, trotz Reduktion des Aufwands eine langfristige hohe Qualität des Übungsbetriebs zu gewährleisten, ist in E-Assessments zu sehen. [KG10, S. 24]. In den meisten Systemen, welche E-Assessments anbieten, werden jedoch nur einfache Aufgabentypen wie Multiple-Choice-Aufgaben oder Lückentexte unterstützt. Über solche Aufgabentypen können Wissenszuwächse gut geprüft werden, kognitive Fähigkeiten und Methodenwissen jedoch nur bedingt. Insbesondere in naturwissenschaftlichen oder technischen Disziplinen wie der Informatik ist die Kontrolle und Überprüfung von Aufgaben, bei denen die Anwendung analytischer, kreativer und konstruktiver Fähigkeiten gefordert ist, besonders wichtig. So entwickelte beispielsweise die WWU Münster das E-Assessment-System EASy für das Informatikstudium, welches einfache Aufgabentypen wie dem Multiple-Choice, aber auch anspruchsvolle Aufgabentypen zu mathematischen Beweisen und Programmierung bereitstellt [KG10, S. 24]. Die Evaluation dieses Systems im Praxiseinsatz zeigte, dass der Lernprozess unterstützt wurde und insbesondere der Übungsbetrieb von diesem System profitiert hat. Weiterhin stellte die Evaluation in Ausblick, dass solche Systeme auch auf eine Vielzahl von anderen Bildungseinrichtungen übertragen werden könnten [KG10, S. 34].

2.2.3 E-Assessment als Methode des Leistungsnachweises

Neben Leistungsüberprüfungen als Bestandteil des Lehr- und Lernprozesses kann E-Assessment auch für einen Leistungsnachweis herangezogen werden. Als Leistungs-

nachweis wird die abschließende Leistungserbringung zum Bestehen einer Lehrveranstaltung oder eines Modul angesehen. So werden Leistungsnachweise klassisch durch Klausuren, Referate oder Studienarbeiten repräsentiert. Wird also E-Assessment als Leistungsnachweis eingesetzt, kann man davon sprechen, dass statt einer traditionellen Prüfungsform eine vergleichbare elektronische Durchführungsvariante gewählt wurde [Rue10, S. 18]. Der folgende Abschnitt beschäftigt sich mit der Gegenüberstellung von traditionellen Prüfungsformen und elektronischen Äquivalenten.

Ein klassisches Beispiel für einen Leistungsnachweis ist die schriftliche Prüfung. Ihr elektronisches Äquivalent ist die E-Prüfung bzw. E-Klausur, welche offene und geschlossene Fragen verwendet, um Lehrinhalte zu prüfen. Oft wird sie als computerunterstützende Prüfung mit Multiple-Choice-Fragen verstanden, jedoch sind auch offene Fragen sehr effektiv. So kann beispielsweise durch die verbesserte Lesbarkeit die Korrekturzeit verkürzt werden. Darüber hinaus kann unter Elektronic Submission der automatisierte elektronische Abgabeprozess von schriftlichen Arbeiten wie Protokollen, Seminar- und Hausarbeiten verstanden werden. Hierbei können z.B. in Lernplattformen Ablagemöglichkeiten zur fristgerechten Abgabe bereitgestellt werden. Daneben können Simulationen am Computer komplexe wissenschaftspraktische Tätigkeiten übernehmen. Derweise können Simulationen die Bewertung typischer Labortätigkeiten wie beispielsweise die Handhabung von Mikroskopen oder die Interpretation von Röntgenbildern übernehmen. Außerdem können Foren als Bewertungsmöglichkeit für mündliche Mitarbeit dienen. Ebenso kann ein Wiki als Präsentationsmöglichkeit einer erbrachten Gruppenleistung dienen. So kann abschließend E-Assessment als Methode der Leistungserbringung als Substitutionsmodell verschiedener klassischer Beurteilungsmethoden verstanden werden [Rue10, S. 18 ff.].

2.3 Die Datenstruktur B-Baum

Der B-Baum ist eine von R. Bayer und E. McCreight entwickelte ausgeglichene Baumstruktur. Hierbei steht der Name *B* für balanciert, breit, buschig ober Bayer, nicht jedoch für binär. Im Gegensatz zu Binärbäumen ist die Grundidee eines B-Baums der variierende Verzweigungsgrad, während die Baumhöhe vollständig ausgeglichen ist. Dies bedeutet, dass alle Pfade von der Wurzel zu den Blättern gleich lang sind, Knoten jedoch mehrere Kanten enthalten können. Deswegen fallen B-Bäume auch in die Kategorie Mehrwegbäume [SS14, S. 386]. Überdies verfügen Knoten eines B-Baums über mehrere Datensätze [SS14, S. 386 f.]. Ihm Rahmen dieser Arbeit werden diese Datensätze als Schlüssel bezeichnet. Abbildung 3 zeigt alle Elemente

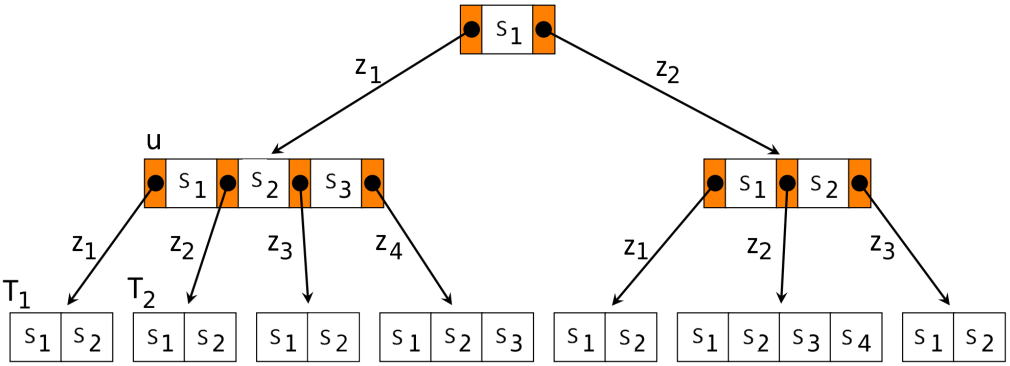

Abbildung 3: Struktur eines B-Baums

eines B-Baums. Dargestellt ist beispielsweise der Knoten u mit seinen Schlüsseln s_1, s_2 und s_3. Von ihm führen die Kanten z_1 bis z_4 ab. Im Folgenden wird ein B-Baum definiert.

Ein Baum T ist ein *B-Baum der Ordnung* $m, m \in \mathbb{N}$, falls

1. alle Knoten maximal $2m$ Elemente,
2. die Wurzel mindestens 1 und alle anderen innere Konten mindestens m Schlüssel,
3. alle inneren Knoten maximal $2m + 1$ Nachfolger und
4. alle Blätter dieselbe Tiefe

haben [S. 387]Saake2014. Die maximale Verzweigung $2m + 1$ wird in dieser Arbeit mit t bezeichnet.

Für einen sortierten Baum muss Folgendes gelten: Sei u ein innerer Knoten mit l Söhnen. Bezeichne $T_i, 1 \leqslant i \leqslant l$ den i-ten Unterbaum des Teilbaumes mit Wurzel u. Der Knoten u enthält $l - 1$ Schlüssel $s_1, s_2, ..., s_{l-1}$ und l Zeiger $z_1, z_2, ..., z_l$. Der Zeiger z_i zeigt auf den i-ten Sohn von u. Für alle Knoten v in T_i gilt für alle Schlüssel s in v [Blu13, S. 22]:

$$\begin{cases} s \leqslant s_i & \text{,falls } i = 1 \\ s_{i-1} < s \leqslant s_i & \text{,falls } 1 < i < l \\ s_{l-1} < s & \text{,falls } i = l \end{cases}$$

3 Vorstellung des Moodlemoduls 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. Dieses Kapitel wird das Tool EASy-DSBuilder vorstellen. Hierbei wird zuerst in Kapitel 3.1 auf die Funktionalität aus Benutzersicht eingegangen. Anschließend wird eine Einordnung als E-Assessment-Tool durchgeführt (Kapitel 3.2). Abschließend erfolgt eine Erläuterung der technischen Umsetzung (Kapitel 3.3).

3.1 Funktionalität aus Benutzersicht

Im folgenden Kapitel wird die Funktionalität des Moodle-Moduls EASy-DSBuilder vorgestellt. Hierbei wird auf die beiden Sichten Studierender und Lehrender eingegangen.

Lehrende

Der Lehrende hat zwei grundlegende Aufgaben. Zum einen ist er dafür verantwortlich, dass eine Aufgabe erstellt wird, zum anderen hat er die Möglichkeit, die Abgaben einzusehen, um sie beispielsweise zu bewerten oder 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 Moodle-Modul hinzufügen. Hierzu muss er einen Wrapper auf Basis des Interfaces *DataStructure-WrapperInterface<K extends Comparable<K>, D>* implementieren (vgl. Anhang, Quellcode 10). Diese Wrapperklasse muss anschließend vom Lehrenden als Hauptklasse eingestellt werden. Auf die Funktionalität der Wrapperklasse aus technischer Sicht wird im Kapitel 3.3.1 näher eingegangen. Des Weiteren kann der Lehrende ein Feedback aktivieren. Die genaue Funktionalität des Feedbacks wird im Abschnitt über die Funktionalität aus Sicht des Studierenden erläutert.

Studierende

Der Studierende verfügt über zwei Ansichten. Zum einen gibt es die Übersichtsansicht, zum anderen die Bearbeitungsansicht. Nachdem der Studierende sich in das Modul eingewählt hat, ist die 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. 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

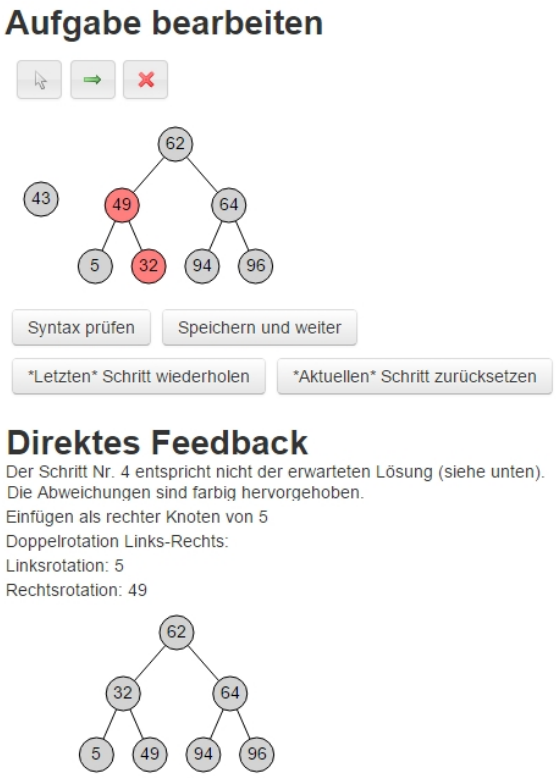

Abbildung 4: DSBuilder Editor

dem der Studierende die Aufgabe bearbeiten kann (vgl. Abb. 4). Der Drag-and-Drop-Grafikeditor enthält zwei bearbeitbare Elemente, die Knoten und die Kanten. Im oberen linken Bereich des Editor befinden sich drei Knöpfe, über welche der Editiermodus ausgewählt werden kann. Der erste 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 Kanten.

Über Manipulation dieser Elemente sollen Studierende den Umgang mit Datenstrukturoperationen erlernen. Hierbei kann der Studierende Operationen wie das Einfügen eines Elements in oder das Löschen eines Elements aus einer Datenstruktur praktizieren. In der aktuellen Version des EASyDSBuilders kann der Studierende nur das Einfügen praktizieren. Weiterhin beginnt in dieser Version jeder Schritt mit

dem Ergebnisbaum des zuvor eingereichten Schritts, oder einem Initiierngsbaum, falls es sich um den ersten Schritt handelt. Auf der linken Seite des Editors wird der einzufügende Knoten bereitgestellt. Die Aufgabe des Studierenden ist es nun, diesen Knoten an der richtigen Stelle in den Baum einzufügen. Hierbei kann er sich dem Verschieben der Knoten wie dem Löschen oder neu Ziehen von Kanten bedienen. 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 Binärbaumes, bekommt der Studierende eine Fehlermeldung mit einem Hinweis über die Fehlerquelle. Ein Baum wird in diesem Sinne als fehlerhaft angesehen, wenn ein Knoten nicht Teil des Baums ist oder der Baum zyklisch ist.

Hat der Lehrende bei der Einrichtung des DSBuilders die Option *direktes Feedback* eingestellt, erscheint im Falle einer falschen Eingabe ein Feedbackfeld unterhalb des Editors. Dieser ist der dritte grundlegende Teil der Bearbeitungsansicht. In diesem Feedbackfeld wird zuerst ein Informationstext angezeigt, welcher das richtige Vorgehen in dem zuvor eingereichten Schritt beschreibt. Unterhalb dieses Informationstextes ist der korrekte Baum zu sehen. Die falsch eingeordneten Knoten sind im Ausgangsbaum rot markiert (vergl. Abb. 4).

3.2 Einordnung als E-Assessment-Tool

Datenstrukturen wie Binärbäume sind elementare Bestandteile von Informatikvorlesungen. Insbesondere der Umgang mit Operationen wie dem Einfügen oder Löschen von Elementen innerhalb solcher Strukturen erfordert kognitive Fähigkeiten, sodass Frageformen wie Multiple-Choice keine adäquaten Formen des Assessments sind. Der EASy-DSBuilder soll beim Erlernen dieser Fähigkeiten innerhalb der Informatikvorlesung unterstützen [Use14, S. 1]. In welchen Typ E-Assessment der DSBuilder einzuordnen ist, wird im folgenden Abschnitt erläutert.

In Kapitel 2.2.1 wurden bereits die drei Assessmenttypen diagnostisches, formatives und summatives Assessment vorgestellt. Diagnostisches Assessment hat die Aufgabe, die Fähigkeiten von Teilnehmern vor Beginn einer Lehrveranstaltung zu ermitteln. Die Funktionalität des DSBuilders ermöglicht diesen Einsatzzweck, wogegen der beabsichtigte Anwendungsfall der Einsatz während des Lehr- und Lernbetriebs ist [Use14, S. 1]. Dies wiederum spricht dafür, dass der DSBuilder als formatives Assessment eingestuft wird. Das Tool stellt für diese Kategorisierung alle geforderten Funktionalitäten bereit. So kann eingestellt werden, dass der Studierende ein

direktes Feedback erhält, sodass er sich selbst einschätzen kann. Des Weiteren kann der Lehrende die Ergebnisse einsehen und somit den Lernstand innerhalb des Kurses einschätzen. Wenn die Feedbackfunktion hingegen abgestellt wird, bietet der DSBuilder die Funktionalität eines summativen Assessments, da der Lehrende neben der Möglichkeit der Einsichtnahme in die Abgaben auch die Möglichkeit der Bewertung der Abgaben hat.

3.3 Umsetzung aus technischer Sicht

Das gesamte System um den EASy-DSBuilder besteht backendseitig aus zwei separaten Systemen. Zum einen gibt es das eigentliche Moodle-Modul, welches in eine bestehende Moodel-Plattform integriert werden kann, zum anderen gibt es einen Datenstruktur-Verarbeitungsservice, welcher als Webservice implementiert ist. Das Moodlemodul hat die Möglichkeit, über die Moodle-API Daten in einer SQL-Datenbank - beispielsweise einem MySQL-Server - zu hinterlegen. Die Kommunikation zwischen dem Moodlemodul 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 5 ist dargestellt, wie die unterschiedlichen Technologien in einander greifen.

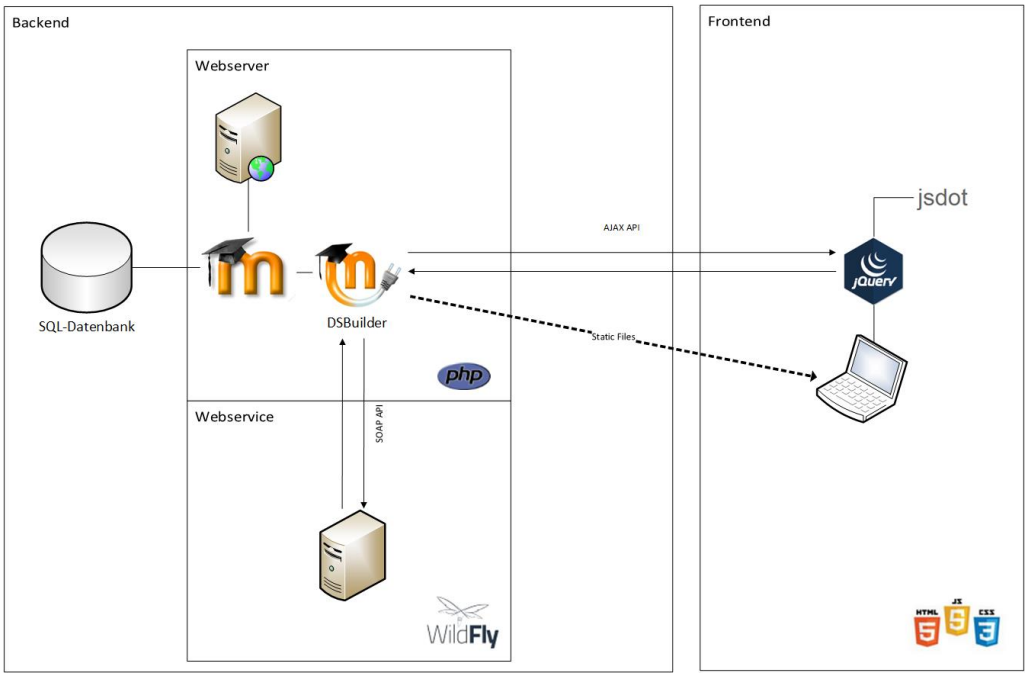

Abbildung 5: Technischer Überblick

Auf Clientseite wird HTML mit CSS und JavaScript verwendet, um das Modul 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 Moodle-Modul über das AJAX-Protokoll organisiert. Jsdot dient als Grapheneditor.

3.3.1 Datenstruktur-Verarbeitungsservice

Der Datenstruktur-Verarbeitungsservice hat die Aufgabe, Datenstrukturen in Form eines JSON Strings für das Moodle-Modul bereitzustellen. Hierfür muss der Lehrende zuvor eine Wrapperdatei auf Basis eines vorgegebenen Interfaces (vgl. Anhang, Codebeispiel 10) in Java implementieren und der Anwendung zur Verfügung stellen. Diese Wrapperklasse dient als Schnittstelle zwischen dem Webservice und der Implementierung der Datenstruktur. Die wichtigen Funktionen, die durch die Implementierung dieses Interfaces zur Verfügung gestellt werden sollen, sind das Hinzufügen und Löschen von Schlüsseln, sowie das Erzeugen eines serialisierten JSON Strings aus der Datenstruktur und das Deserialisieren eines JSON Strings in die Datenstruktur.

Um funktionsfähig zu sein, muss der Datenstruktur-Verarbeitungsservice den vom Lehrenden bereitgestellten Code kompilieren und ausführen. Der Verarbeitungsservice ist als separater Webservice implementiert und wird somit von einem anderen Server aus bereitgestellt. Die Separierung des Systems erfolgt aus den Risiken, dass der Code schädlich sein oder eine schlechte Ausführungsleistung aufweisen 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 verhindert 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].

Der Webservice verfügt über die SOAP-Schnittstellen *analyzeSourceCode*, *assignmentInitialize* und *assignmentIsIntialized*, die zur Initialisierung einer neuen DSBuilder-Instanz benötigt werden. Ebenso verfügt der Webservice über die SOAP-Schnittstellen *submissionGetNextInput* und *submissionGetInitialDS*, die zur Verarbeitung einer Datenstruktur benötigt werden. Um den Webservice nutzen zu können müssen ihm Codedateien zu Verfügung gestellt werden. Über den SOAP-Aufruf *analyzeSourceCode* wird die Lesbarkeit der Dateien geprüft. Anschließend kann über den SOAP-Aufruf *assignmentInitialize* der Code compiliert und bereitgestellt werden. Die Ausführung des Codes ist vor jedem Einfügen, das von einem Studierenden durchgeführt wird, notwendig. So kann über den SOAP-Aufruf *submissionGetNex-*

tInput der der nächste Übungsschritt für den Studierenden bereitgestellt werden. Bei der Initialisierung der Übung wird der SOAP-Aufruf *submissionGetInitialDS* benötigt. Die Antwort auf diese beiden Aufrufe beinhaltet den neuen Schlüssel, die serialisierte Datenstruktur und zusätzliches Feedback.

3.3.2 Moodlemodul backendseitig

Das backendseitige Moodlemodul besitzt die grundlegende Struktur eines Moodlemoduls, wie sie in Kapitel 2.1.2 dargestellt wurde. Die Weiteren, für die Funktionalität des Moduls wichtigen Dateien, sind die Dateien **`renderer.php`** und **`renderable.php`**, welche den 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 Funktionalität 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 für die Generierung des DOM-Codes verantwortliche Datei ist, ist die **`renderer.php`**. Die Datei **`renderable.php`** implemeniert hingegen ein Interface, welches für die Verwendung des moodleinternen Renderers notwendig ist [Moo15i]. 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. Hierbei

```
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

übernimmt die Initialisierung des Grapheneditors, welche im Quellcode 1 dargestellt wird, eine zentrale Rolle. Es wird eine Hilfsfunktion des Moodle-API [Moo15e] verwendet, welche die frontendseitige JavaScript-Funktion zur Initialisierung des Grapheneditors anstößt. Die frontendseitige Funktionalität wird im Kapitel 3.3.3 vertieft.

Modulinterne AJAX-API

Zur asynchronen Datenübertragung zwischen Browser und Server stellt das Moodle-Modul eine AJAX-API zur Verfügung. Der Quellcode 2 zeigt einen Codeausschnitt

```
1 try {
2   if ($action === DSBUILDER_AJAX_ACTION_CHECK) {
3     $result = $dsbuilder_ajax->action_check_valid_graph(
          $jsdot_graph_raw, $additional_info);
4   } elseif ($action === DSBUILDER_AJAX_ACTION_NEXT_STEP) {
5     $result = $dsbuilder_ajax->action_submit_current_step(
          $jsdot_graph_raw, $additional_info);
6   } elseif ($action === DSBUILDER_AJAX_DELETE_LAST_STEP) {
7     $result = $dsbuilder_ajax->action_delete_last_step();
8   }
9 }
```

Quellcode 2: Ausschnitt AJAX API

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 (vgl. Kapitel 3.1, 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 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 6 zeigt das Datenmodell des Moduls.

Nachdem das Modul neu in 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.

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 aus dem Webservice resultierenden Datenstruktur bietet. Des Weiteren repräsentiert diese Klasse die Datenstruktur, die im jsdot-Editor zur

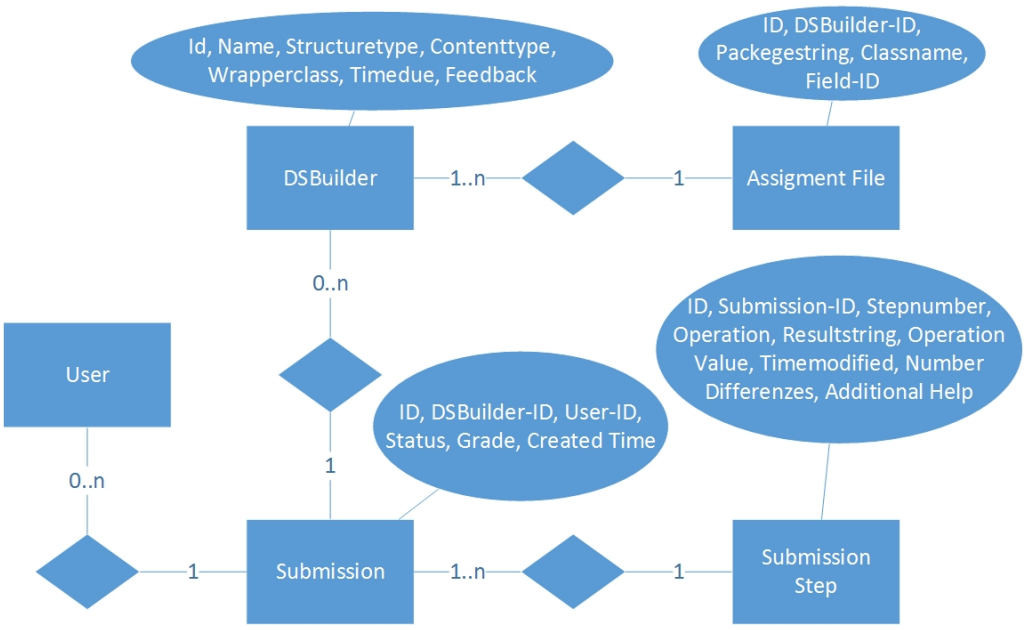

Abbildung 6: Datenmodell des DSBuilders

Präsentation eines Baumes gebraucht wird. Da ein jsdot-Graph über Knoten und Kanten verfügt, werden diese Elemente in der Datei **`lib/js_dot_con- vert.php`** durch die Klassen **`jsdot_edge`** und **`jsdot_node`** repräsentiert. Spezifischere Erläuterungen zu den jsdot-Elemente sind in Kapitel 3.3.3 zu finden. Abbildung 7 zeigt ein UML-Klassendiagramm der vier Klassen. Die Funktionalität zur Umwandlung zwischen Datenstrukturen für den jsdot-Editor und den Webservice stellt die Klasse **`dsbuilder_binary_tree`** bereit.

Die Datenstrukturverarbeitung

In der Datei **`lib/js_dot_convert.php`** liegt die Funktionalität der Datenstrukturverarbeitung.

Die drei öffentlichen Funktionen der Klasse **`dsbuilder_binary_tree`**, welche die wichtigen Funktionalitäten zur Verarbeitung der Datenstruktur bereitstellen, sind die statischen Funktionen *initialize_with_binary_tree_data_structure(...)*, *get_binary_tree_data_ structure(...)* und *compare_sample_and_student_graph(...)*. Hierbei ermöglicht die erste Funktion die Umwandlung der Datenstruktur, die der Webservice liefert, in die Datenstruktur für den jsdot-Editor. Die zweite Funktion ermöglicht die Umwandlung in die umgekehrte Richtung. Die dritte Funktion vergleicht zwei Graphen und markiert die sich unterscheidenden Knoten. Sie wird für die Feedbackfunktion genutzt.

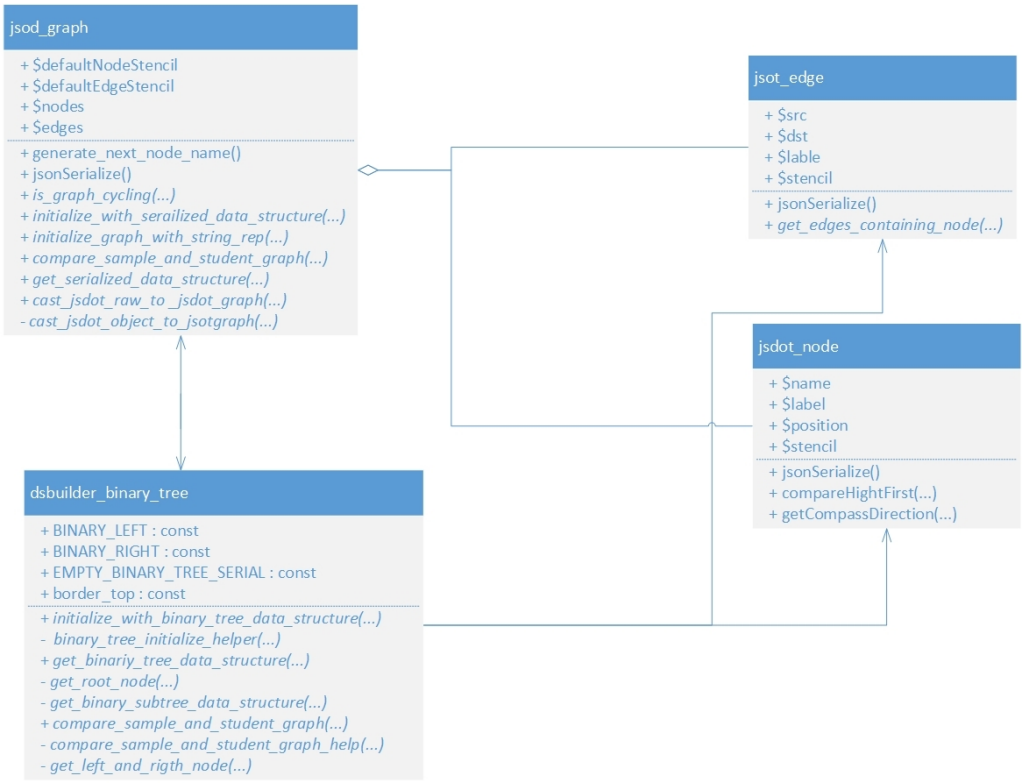

Abbildung 7: Klassendiagramm der **`js_dot_convert.php`**

3.3.3 Moodlemodul frontendseitig

Die im Vordergrund stehende Funktionalität des Fronends ist der von der JavaScript-Applikation jsdot bereitgestellte Grapheneditor. In diesem Grapheneditor werden Elemente zur Verfügung gestellt, die vom Studierenden bearbeitet werden können. Die Funktionalität zur Verarbeitung der Nutzereingaben wird über die Datei **`dsbuilder.js`** bereitgestellt. Ferner stellt diese Datei die AJAX-Kommunikation zur Verfügung.

Der Grapheneditor

Der Grapheneditor wird durch die JavaScript-Applikation jsdot bereitgestellt. So kann jsdot über das HTML-Element *svg* Knoten und Kanten bereitstellen, die editierbar sind. Aus diesen Elementen können Strukturen wie Graphen, Bäume oder Listen erstellt werden. Der Codeauszug 3 zeigt die Initialisierung eines neuen jsdot-Graphen. Weiterhin stellt jsdot eine API zur Verfügung. Über diese API können neue Elemente zu einem Graphen hinzugefügt werden. Ebenso können bestehende Elemente geändert oder ausgelesen werden. Außerdem kann bei der Initialisierung eines Graphen eingestellt werden, ob der Graph im Bearbeitungsmodus oder im Anzeigemodus bereitgestellt wird. Codebeispiel 3 zeigt die Initialisierung eines Gra-

```
init_jsdot_edit : function(e, divname, jsongraph) {
        this.jsdot_graphs[divname] = new JSDot(divname, {
                mode : "editor",
                json : jsongraph
        });
},
```

Quellcode 3: Initiierung eines JsDot-Graphs

phen im Bearbeitungsmodus. Der Bearbeitungsmodus wird verwendet, um einen editierbaren Graphen für die Aufgabenbearbeitung zur Verfügung zu stellen (vgl. Kapitel 3.1, Abb. 4 oben). Der Anzeigemodus wird hingegen für das Feedback verwendet (vgl. Kapitel 3.1, Abb. 4 unten). Im Frontend findet keine Verarbeitung der Graphen-Datenstruktur statt. Die Graphen-Datenstruktur wird über die im folgenden Abschnitt erläuterte AJAX-API an das Backend gesendet und dort verarbeitet (vgl. Kapitel 3.3.2, Abschnitt *Datenstrukturverarbeitung*).

AJAX-Api des Frontends

Das Frontend stellt eine AJAX-API zur Echtzeitkommunikation mit dem Moodle-Modul bereit. Über die Knöpfe unterhalb des Grapheneditors (vgl. Kapitel 3.1, Abschnitt *Studierender*) können die Funktionen der API angestoßen werden. Die Verarbeitung der AJAX-Anfragen wurde bereits in Kapitel 3.3.2 im Abschnitt *Modulinterne AJAX-API* erläutert. Über AJAX-Anfragen werden Informationen wie die Graphen-Datenstruktur, das Feedback und mögliche Fehler übertragen.

4 Anforderungen an ein E-Learning-Modul zum Assessment von B-Baum-Datenstrukturen

In diesem Kapitel werden die bestehenden Anforderungen an ein E-Learning-Modul zum Assessment von B-Baum-Datenstrukturen vorgestellt und näher erläutert. Das im Rahmen dieser Arbeit vorgestellte Assessment-Konzept bezieht sich auf die Erweiterung des Moodle-Moduls EASy-DSBuilder, welches um die Funktion des Assessments von B-Baum-Datenstrukturen erweitert werden muss. Diesbezüglich stellt das Kapitel 4.1 die Anforderungen an das B-Baum-Assessment im EASy-DSBuilder vor. Anschließend wird in Kapitel 4.2 tiefgehender auf die Anforderungen des Grapheneditors eingegangen.

4.1 Anforderungen an das B-Baum-Assessment im EASy-DSBuilder

In diesem Kapitel werden die Anforderungen an die Erweiterungen des EASy-DS-Builders um das B-Baum-Assessment vorgestellt und näher erläutert. Die Anforderungen wurden im Gespräch mit dem Verantwortlichen für das Moodle-Modul EASy-DSBuilder ermittelt. Hierbei wurden die Anforderungen in die Kategorien *Funktionale* und *Nicht-Funktionale Anforderungen* unterteilt. *Funktionale Anforderungen* beschreiben Fähigkeiten des Systems, die von einem Anwender erwartet werden, damit dieser mit Hilfe dieses Systems ein fachliches Problem gelöst werden kann. *Nicht-Funktionale Anforderungen* definieren hingegen die grundlegenden Eigenschaften des Systems [Bal14, S. 109]. Über Userstories werden die Anforderungen veranschaulicht.

4.1.1 Funktionale Anforderungen

Im Folgenden werden die *Funktionalen Anforderungen* für ein Modul zum Assessment von B-Baumstrukturen beschrieben.

(1) Didaktische Qualität soll erhalten bleiben

> Das E-Assessment soll sich im Schwierigkeitsgrad qualitativ nicht vom papierbasiertem Assessment unterscheiden.

Das Assessment soll so aufgebaut sein, dass es der Struktur eines papierbasierten Assessments entspricht. Insbesondere bedeutet dies, dass die Bearbeitungsmöglichkeiten aus didaktischer Sicht nicht eingeschränkt werden dürfen. Eingeschränkt aus

didaktischer Sicht bedeutet in diesem Kontext, dass den Studierenden während der soeben beschriebenen Aufgabenbearbeitung im Verglich zur papierbasierten Aufgabenbearbeitung Bearbeitungsvariationen bei der Lösungsfindung vorenthalten werden.

(2) Das Assessment soll zehnschrittig sein

> Das Assessment soll über 10 Schritte gehen, in denen der Baumstruktur pro Schritt jeweils ein neuer Schlüssel hinzugefügt wird.

Ebenso wie beim Assessment des AVL-Baums soll das Assessment des B-Baums über zehn Schritte gehen. Pro Schritt wird dem Studierenden hierbei ein neuer Schlüssel zur Verfügung gestellt, die er in den bereits vorhanden B-Baum einfügen können soll. Nachdem die zehn Schritte absolviert sind, soll der Studierende darüber informiert werden, dass er die gesamte Aufgabe absolviert hat.

(3) Der Studierende soll einen Überblick über den aktuellen Aufgabenstand erhalten

> Der Studierende soll die Möglichkeit, erhalten einen Überblick über den aktuellen Stand und den Fortschritt der Bearbeitung seiner Aufgabe zu bekommen.

Ebenso wie beim Assessment des AVL-Baums soll das Assessment des B-Baums eine Übersicht über dein aktuellen Stand der Aufgabe bereitstellen. Auf dieser Übersicht erhält der Studierende Informationen darüber, wie viel Prozent er bereits absolviert hat, in welchem Schritt er sich befindet und welcher Schlüssel in diesem Schritt zum Einfügen bereitgestellt wurde.

(4) Der Studierende soll einen Editor zum Assessment von B-Baum-Datenstrukturen erhalten

> Der Studierende soll die Möglichkeit bekommen, eine B-Baum-Datenstruktur innerhalb eines Editors so zu bearbeiten, sodass er die Aufgabenstellung jedes Schrittes lösen kann.

Ebenso wie beim Assessment des AVL-Baums soll das Assessment des B-Baums über einen Editor verfügen, in dem man eine B-Baum-Datenstruktur bearbeiten kann. Die Elemente, die durch den Editor bereitgestellt werden sollen, sind Kanten, Knoten und Schlüssel. Es soll möglich sein, aus diesen Elementen einen syntaktisch

korrekten B-Baum zu erstellen. Des Weiteren muss der Editor Funktionalitäten bereitstellen, die das Einfügen eines neuen Schlüssels in einen bereits bestehenden B-Baum ermöglichen.

(5) Es soll die Möglichkeit des Feedbacks gegeben werden

> Je nach Einstellung des Lehrenden soll der Studierende bei falscher Eingabe ein Feedback bekommen.

Momentan besteht für den Lehrenden die Möglichkeit, ein Feedback für den Studierenden einzustellen. Hat dieser das Feedback aktiviert, so soll während des Assessments von B-Baum-Datenstrukturen bei falscher Eingabe seitens des Studierendem diesem unterhalb des Editors der B-Baum mit richtig eingefügtem Schlüssel angezeigt werden. Weiterhin sollen die sich in Editor und Feedback unterscheidenden Knoten rot markiert werden.

(6) Navigationsmöglichkeiten innerhalb einer Aufgabe müssen vorhanden sein

> Der Studierende soll die Möglichkeit haben, innerhalb der einzelnen Schritte einer Aufgabe zu navigieren. Zur Navigation gehört die Prüfung der Syntax, das Bestätigen des aktuellen Schritts und das Zurücksetzen des vorherigen und des aktuellen Schritts.

Bisher hat der Studierende die Möglichkeit, sich über die Knöpfe *Syntax prüfen*, *Speichern und weiter*, **Letzten* Schritt wiederholen* und **Aktuellen* Schritt zurücksetzen* durch eine Aufgabe zu navigieren. Für das Assessment von B-Baum-Datenstrukturen soll diese Art der Navigation beibehalten werden.

(7) Aufgabenschritte müssen auf der vorherigen Eingabe basieren

> Bis auf den Initialisierungsschritt müssen alle Aufgabenschritte einer Aufgabe auf der letzten Eingabe des Studierenden basieren.

Ebenso wie beim Assessment des AVL-Baums soll beim Assessment des B-Baums jeder Aufgabenschritt bis auf den Initialisierungsschritt auf der Abgabe des vorherigen Schritts basieren. Dies bedeutet, dass der Studierende, falls er eine fehlerhafte Abgabe getätigt hat, auch mit dieser Abgabe weiterarbeitet.

(8) Der Lehrende soll eine Auswahlmöglichkeit erhalten, zwischen Datenstrukturtypen wählen zu können

> Der Lehrende soll bei Einrichtung einer neuen Instanz des DSBuilders aus einer Liste mit allen verfügbaren Datenstrukturtypen die Auswahl treffen können, mit welcher Datenstruktur die Anwendung dem Studierenden bereitgestellt wird.

In der momentanen Version des EASy-DSBuilders wird als Datenstrukturtypen *Tree* defaultmäßig als Datenstruktur übergeben. Dem Lehrenden soll die Möglichkeit gegeben werden, bei der initialen Einstellung einer Instanz des Moduls einen Datenstrukturtypen auszuwählen.

4.1.2 Nicht-Funktionale Anforderungen

Im Folgenden werden die *Nicht-Funktionalen Anforderungen* beschrieben.

(1) Die Laufzeit einer Nutzereingabenverarbeitung soll in vertretbarer Zeit stattfinden

> Das Modul soll die Eingaben des Studieren in vertretbarer Zeit erkennen und verarbeiten. Den wichtigsten Teil der Verarbeitung nimmt der Umgang mit der Datenstruktur ein.

Das Modul soll aus der Eingabe des Studieren eine B-Baum-Datenstruktur erkennen und sie in eine Datenstruktur umwandeln können, die vom Webservice erkannt und weiterverarbeitet werden kann. Dies soll ein vertretbarer Zeit stattfinden.

4.2 Spezifizierung der konzeptionellen Anforderungen des Grapheneditors

Werden die Anforderungen aus Kapitel 4.1 auf eine Kernanforderung reduziert, dann kann diese Kernanforderung darin verstanden werden, dass dem Studierenden ein graphischer Editor zur Verfügung gestellt werden soll, über den ein B-Baum erweitert werden kann. Vergleicht man einen B-Baum mit dem bereits implementierten AVL-Baum, so verfügt der B-Baum über eine weitere editierbare Komponente. Beim AVL-Baum sind Knoten und Schüssel durch ein eins-zu-eins-Mapping unveränderbar miteinander verbunden. Beim B-Baum kann ein Knoten jedoch mehrere Schlüssel enthalten, welche aber auch nicht fest an diesen Knoten gebunden sind. Die daraus

resultierende Herausforderung ist, die drei Komponenten Kante, Knoten und Schlüssel dem Studierenden so zur Verfügung zu stellen, dass der Studierende die Komponenten möglichst intuitiv nutzen kann. Die JavaScript-Applikation jsdot soll dabei beibehalten werden. Es ergaben sich die Möglichkeiten, einen B-Baum komplett aus editierbaren jsdot-Elementen zu implementieren, oder eine B-Baumstruktur in den Hintergrund des Editors zu legen, auf dem die Schlüssel richtig eingeordnet werden müssen. Im Folgenden werden Vor- und Nachteile beider Möglichkeiten vorgestellt.

4.2.1 B-Baum aus jsdot-Elementen

Jsdot bietet die Möglichkeit, neben runden auch rechteckige Editor-Knoten zu erstellen. Aus diesen könnten Baum-Knoten generiert werden. Auf diese Weise könnten Studierende eine B-Baumstruktur auf der Basis dieser Editor-Knoten und -Kanten erstellen, um darauf die jeweiligen Schlüsselelemente richtig zu positionieren.

Jedoch ergeben sich daraus mehrere Probleme. So ist die Umsetzung der Positionierung der abwärtsgerichteten Kanten ein Problem. Idealtypisch ist die Positionierung der abwärtsgerichteten Kanten am Anfang oder am Ende eines Baum-Knotens oder zwischen zwei Schlüsseln (vgl. Kapitel 2.3, Abb. 3). Dies ist jedoch ohne einen größeren Eingriff in die jsdot-Implementierung nicht möglich. Alternativ können die abwärtsgerichteten Kanten an die Schlüssel gehängt werden, wobei der erste oder letzte Schlüssel eines Knotens über zwei abwärtsgerichtete Kanten verfügt. Die Kanten können dann wie zuvor an Knoten enden. Jedoch kann nicht unterbunden werden, dass Schlüssel oder Kanten untereinander im nicht beabsichtigten Sinn verbunden werden. Daraus resultiert, dass eine Großzahl an möglichen falschen Eingabemöglichkeiten überprüft werden müsste. Auch könnte diese Vorgabe die Usability beeinträchtigen, da sich die geforderte Benutzung der einzelnen Elemente seitens des Studierenden als nicht eindeutig nachvollziehbar herausstellen könnte. Des Weiteren bereitet die Höhenhierarchie der einzelnen jsdot-Elemente Probleme. So ist es auf Grund fehlender Dokumentation bezüglich der Höhenhierarchie einzelner Elemente schwer umsetzbar Schlüssel so zu definieren, dass sie Knoten durchgehend überlappen.

4.2.2 B-Baumstruktur im Hintergrund

Die Baumstruktur im Hintergrund bietet den Vorteil, dass nur noch die einzelnen Schlüssel editierbar sein müssen, der Studierende folglich nur noch die Schlüssel an die richtige Position bringen muss. Dies bietet den Vorteil, dass die Richtigkeit über die absolute Position, und nicht die Position relativ zu anderen Schlüsseln, geprüft

werden kann. Es ergeben sich zwei Möglichkeiten eines Hintergrundgraphen. Es kann entweder ein statischer Baum, der durchgehend die gesamte Struktur aufweist, oder ein dynamischer Baum, der sich der Eingabe des Studierenden anpasst, implementiert werden. Ein statischer Baum erfordert weniger Implementierungsarbeit, ein dynamischer Baum erhöht jedoch die Übersichtlichkeit, da nur benötigte Teile des Baums angezeigt werden. Außerdem kann ein dynamischer Baum die Chance bieten, dem Studierenden die Entwicklung eines B-Baums besser zu veranschaulichen. Diesen Vorteilen steht jedoch gegenüber, dass der Studierende aus didaktischer Sicht in seinen Bearbeitungsmöglichkeiten eingeschränkt sein würde. Insbesondere hier kann der Studierende die Baumstruktur nicht mehr falsch aufbauen, sodass die Hintergrundstruktur des B-Baums zur Reduktion möglicher Fehlerquellen führt.

4.2.3 Ergebnis

Insbesondere der B-Baum aus jsdot-Elementen bringt verschiedene Herausforderungen mit sich. So ist es in dieser Variante auf Grund der Vielzahl von Elementen schwierig eine einfach zu verstehende und zu benutzende graphische Oberfläche zu bieten. Dem gegenüber steht die einfach zu bedienende graphische Oberfläche mit einer B-Baumstruktur im Hintergrund, die dem Studierenden jedoch striktere Vorgaben in der Benutzung vorgibt und somit Fehlerquellen reduziert. Die hier vorliegende Situation beschreibt einen Trade-off zwischen geringer Usability mit einem hohen Maß an Bearbeitungsvariationen und einer hohen Usabiltiy mit einem geringeren Maß an Bearbeitungsvariationen. Wie die Auswertung der Fallstudie über die Nutzung des EASy-DSBuilders zeigte, sorgte besonders die hohe Usability für hohe Akzeptanz bei den Studierenden[Use14, S. 5]. Auch in der neuen Anwendung soll der Akzeptanz eine hohe Priorität zugeschrieben werden, sodass trotz der Vereinfachung der Aufgabe durch die Reduktion möglicher Fehlerquellen der Editor mit Hintergrundstruktur dem Editor, in dem alle Elemente bewegbar sind, vorzuziehen ist.

5 Umsetzungsdetails der neuen Anforderungen

In dieses Kapitel werden die Stellen vorgestellt, an denen Änderungen vorgenommen wurden und erläutert die Hintergründe, auf Grund derer diese Änderungen vorgenommen werden mussten. Hierzu wird zuerst in Abschnitt 5.1 ein Überblick über die notwendigen Änderungen und Erweiterungen gegeben. In den restlichen Abschnitten wird tiefgehender auf die wichtigen Änderungen und Erweiterungen eingegangen. Dabei wird zuerst die Funktionalität erläutert. Zur Verdeutlichung werden anschließend exemplarisch Ausschnitte aus der Implementierung vorgestellt.

5.1 Überblick über Änderungen

In diesem Abschnitt wird ein grundlegender Überblick über alle Änderungen gegeben, die in dem Moodle-Modul EASy-DSBuilder vorgenommen wurden, um die Anforderungen aus Kapitel 4.1 über Funktionalität des Assessments von B-Baum-Datenstrukturen umzusetzen.

Für die Verwendung des EASy-DSBuilders ist eine Initialisierung seitens eines Lehrenden nötig. In der alten Version konnten die in Kapitel 3.1 beschriebenen Eigenschaften übergeben werden. Es konnte jedoch nicht zwischen Datenstrukturen gewählt werden. So wird in der alten Version als Datenstrukturtyp defaultmäßig *Tree* übergeben. An dieser Stelle ist eine Auswahlmöglichkeit für Datenstrukturen in der Erstellungsmaske des Moduls implementiert worden. Die Auswahlmöglichkeit ist als Dropdown-Menü umgesetzt worden, wobei die in der Datei `dsbaStructureType.php` definierten Konstanten als Datenstrukturtypen zur Auswahl gestellt sind.

Nachdem eine Instanz des EASy-DSBuilders erzeugt ist, muss dem Modul jedoch zur Durchführung eines Assessments eine Datenstruktur seitens des Datenstrukturverarbeitungs-Webservices bereitgestellt werden. Die zur Kommunikation zwischen Moodle-Modul und Webservice verwendete Datenstruktur wird in Abschnitt 5.2 vorgestellt. An der Funktionalität des Datenstrukturverarbeitungs-Webservices ist nichts geändert worden. Fehler, die während der Entwicklung auftraten, wurden behoben. Zur Endwicklung des Moodle-Moduls musste jedoch die Datenstruktur eines B-Baums in Java implementiert werden, damit die Funktionen des Webservices bereitgestellt werden konnten.

Die wichtigen Erweiterungen zur Bereitstellung der Funktionalität eines Assessments von B-Baum-Datenstrukturen wurde im Moodle-Modul fronend- und backendseitig implementiert. Hierbei steht frontendseitig die Erweiterung des Grapheneditors im Vordergrund, welche ausführlich im Abschnitt 5.3 erläutert wird.

Des Weiteren war die Implementierung einer Verarbeitung der Datenstruktur im Backend nötig, sodass nun zwischen der vom Webservice stammenden Datenstruktur und der für den Editor gebrauchten Datenstruktur gewandelt werden konnte. Diese Funktionalität wird ausführlicher in Abschnitt 5.4 erläutert. Darüber hinaus wurden noch folgende Veränderungen vorgenommen. In der Datei **`renderer.php`**, in der die Ansicht generiert wird, wurde eine Fallentscheidung für die Anzeige des Grapheneditors implementiert. Dies ist damit begründet, dass der B-Baum-Editor, wie in Kapitel 4.2 begründet, zwei übereinander liegende jsdot-Editoren benötigt. Die Überlagerung der beiden Editoren wurde mit *CSS* umgesetzt. Weiterhin wurde die AJAX-Api angepasst. Die minimale Änderung bestand darin, dass eine weitere Information übergeben wird, auf die der B-Baum-Editor aufbaut.

5.2 Datenstruktur für die Moodle-Modul-Webservice-Kommunikation

Die Datenstruktur, die in der Kommunikation zwischen Moodle-Modul und Webservice eingesetzt wird, spielt eine fundamentale Rolle in der Gesamtanwendung EASy-DSBuilder. Diese Datenstruktur muss in einem JSON-kompatiblen Format alle notwendigen Informationen über die zu verarbeitende Baumstruktur beinhalten. Im Falle des B-Baums muss die Datenstruktur als zentrales Element den Knoten mit der jeweiligen Anzahl an Schlüsseln wiedergeben. Des Weiteren muss die Datenstruktur noch die Kindsknoten der innere Knoten beinhalten, die wiederum richtig positioniert werden müssen. So können die Kindsnoten als linkes oder rechtes Kind des inneren Knotens positioniert werden. Außerdem gibt es noch Kindsknoten, die zwischen den einzelnen Schlüsseln des Wurzelknotens positioniert werden (vgl. Kapitel 2.3).

Auf dieser Grundlage wurde zur Abbildung eines B-Baums eine mehrdimensionale Arraystruktur als Datenstruktur ausgewählt. Diese Arraystruktur ist bei einem B-Baum T mit einer Höhe größer eins wie folgt aufgebaut. Die erste Dimension der Arraystruktur spiegelt die Wurzel des B-Baums T wieder. Erstes Element des Arrays ist der linke Kindsknoten der Wurzel. Anschließend folgt der erste Schlüssel der Wurzel, dem ein weiterer Kindsknoten nachfolgt. Diese Abfolge aus Schüssel und Kindsknoten erfolgt so lange, bis das Array mit dem rechten Kindsknoten der Wurzel abschließt. Die jeweiligen Kindsknoten sind nach derselben Struktur des Wurzelknotens aufgebaut. Eine mögliche Repräsentation eines B-Baums durch diese Datenstruktur kann wie folgt aussehen:

[[[”16”, ”19”], ”31”, [”37”, ”41”]], ”50”, [[”56”], ”86”, [”96”]]]

Die so eben gezeigte Datenstruktur zeigt einen B-Baum der Höhe drei. Der oberste Wurzelknoten enthält den Schüssel ”3”. Die Wurzel hat das linke Kind [[”16”, ”19”], ”31”, [”37”, ”41”]] und das rechte Kind [[”56”], ”86”, [”96”]], welche jeweils separat betrachte einzelne B-Bäume darstellen. Diese Beispiel verdeutlicht anschaulich den rekursiven Aufbau dieser Datenstruktur. Abgeschlossenen wird die Struktur durch kinderlose Knoten, wie sie im vorgestellten B-Baum beispielsweise durch den Knoten [”16”, ”19”] repräsentiert werden. Diese Datenstruktur steht wiederum in einem weiteren Array mit der Länge zwei an erster Position. An zweiter Position wird die Ordnungszahl m (vgl. Kapitel 2.3) abgelegt.

5.3 Grapheneditor

Im Folgenden wird die neu entwickelte Funktionalität des Frontends erläutert. Insbesondere bedeutet dies, dass auf die Umsetzung des Grapheneditors für die B-Baumstruktur eingegangen wird. Hierzu wird zuerst die Umsetzung des in Kapitel 4.2 vorgestellten Konzepts des Grapheneditors in Kapitel 5.3.2 vorgestellt. Anschließend wird in Kapitel 5.3.3 tiefgehender auf einzelne, in Kapitel 5.3.2 vorgestellten Funktionen eingegangen.

5.3.1 Positionierung der B-Baum-Elemente

Dieser Abschnitt erläutert das Prinzip, nach dem die B-Baum-Elemente positioniert werden. Die zentrale Datenstruktur für die Positionierung der Elemente liefert das mehrdimensionale Array `nodesPerTier`. Aus der obersten Dimension des Arrays kann man die Knoten einer Schicht entnehmen, wobei die einzelnen Knoten wiederum über ihre jeweiligen Schlüssel verfügen. Alle Knoten einer Schicht meint in diesem Zusammenhang alle Knoten einer bestimmten Höhe. Der Einfachheit halber werden den Dimensionen des Arrays Buchstaben zugeordnet. Der ersten Dimension, welche die Schichten enthält, wird i zugeordnet. Der zweiten Dimension, welche die Knoten enthält, wird j zugeordnet. Der dritten Dimension, welche die Schlüssel enthält, wird k zugeordnet. Somit kann über das Array `nodesPerTier[i][j][k]` auf den k-ten Schüssel des j-sten Knotens in der i-ten Schicht zugegriffen werden. Über die Variablen i, j und k können Schlüssel somit eindeutig identifiziert werden. Wie die Schlüssel in die Datenstruktur `nodesPerTier[i][j][k]` eingeordnet werden, wird in Kapitel 5.3.3 erläutert.

Über das Array `nodesPerTier` erfolgt die Positionierung der einzelnen Schlüssel. Über diese Struktur wird jedem Schlüssel eine bestimmte Position auf dem Editor zugeordnet. Die Position auf der y-Achse errechnet sich über die Formel $50 * i + 25$.

Die Höhendifferenz pro Schicht beträgt somit 50 Pixel. Weiterhin erhält der B-Baum einen Abstand zum oberen Rand des Editors von 25 Pixeln. Die Position auf der x-Achse berechnet sich über die Formel $spacing\_x + x\_start + (place\_per\_array * j) + (k * 30)$. Die Variable $spacing\_x$ beschreibt den Abstand, den der B-Baum vom linken Rand des Editors hat. Der Betrag liegt bei 50 Pixeln, sodass der neu einzufügende Knoten in diesem Bereich platziert werden kann. Die Variable x_start beschriebt den Betrag, um welchen die Knoten je Schicht eingerückt werden, damit der B-Baum eine gleichmäßige Struktur erhält. Über den Ausdruck $place\_per\_array * j$ werden die Knoten auf ihre jeweilige Position gebracht. $Place\_per\_array$ beschreibt den Platz, den ein Knoten zur Verfügung hat. Dieser Platz hängt von der Höhe des Knotens ab. Über den Index j wird der Knoten um den richtigen Betrag verschoben. Zuletzt wird noch der Schlüssel durch den Ausdruck $(k * 30)$ richtig positioniert. Es werden jedem Schlüssel hierbei 30 Pixel zur Verfügung gestellt.

Abbildung 8 zeigt eine B-Baum-Struktur im Editor. Die kreisförmigen Elemente stellen die Schlüssel dar. Die rechteckigen Elemente repräsentieren in ihrer Gesamtheit die Knoten des Baums, wobei sie einzeln die Stellen widerspiegeln, auf denen Schlüssel abgelegt werden können. Dem Studierenden steht es frei die Schlüssel zu

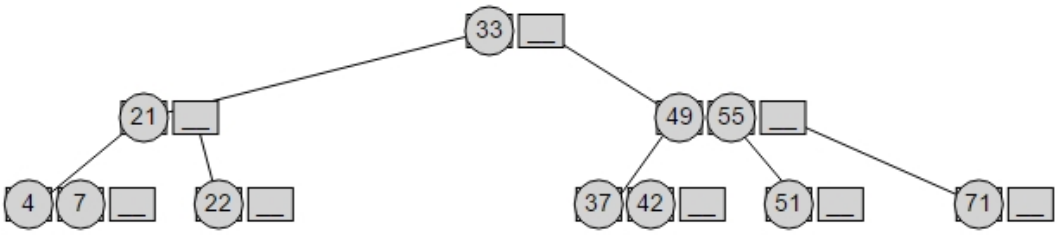

Abbildung 8: B-Baum-Struktur im Editor

bewegen. Bewegt der Studierende einen Schlüssel von einem Knotenelement, so reduziert sich die Baumstruktur. Legt der Studierende hingegen einen Schlüssel auf einem Knoten ab, so erkennt das System dies und ordnet den Schlüssel automatisch in den Knoten ein. Die Baumstruktur erweitert sich währenddessen. Die Funktionalität der Nutzereingabenverarbeitung wird tiefgehender im folgenden Kapitel 5.3.2 erläutert.

5.3.2 Funktionalität

Aus dem im Kapitel 4.2 bereits erläuterten Gründen soll der Grapheneditor aus zwei separaten Elementen bestehen. Diese beiden Elemente sind zwei jsdot-Editoren, welche übereinander liegen. Der im Hintergrund liegende Editor ist nicht bearbeitbar und und liefert die B-Baumstruktur, auf der die Schlüssel des B-Baums angeordnet werden können. Die Schlüssel werden von einem im Vordergrund liegendem jsdot-Editor zur Verfügung gestellt und sind verschiebbar. Um eine höhere Interaktivität bereitzustellen, entwickelt sich die im Hintergrund liegend Baumstruktur dynamisch. Das bedeutet, dass sobald ein Schlüssel zu der Baumstruktur hinzugefügt wird, sich die Baumstruktur automatisch um neue Knoten vergrößert. Die Anpassung beinhaltet die Erweiterung des Knotens, in den eingefügt wurde, sowie die Kindsknoten, die der erweiterte Knoten bereitgestellt bekommt. Ebenso reduziert sich die Baumstruktur nach Entnahme eines Schlüssels um die jeweiligen Knoten. Sobald ein Schlüssel auf einen Knoten abgelegt wird, erkennt das System dies und richtet den gesamten Knoten mit dem neu eingefügten Schlüssel in Echtzeit aus. Durch die automatische Neuausrichtung der Schlüssel kann der Studierende erkennen, ob das System seine Eingabe erkannt hat.

Abbildung 9 zeigt ein Petri-Netz, welches die Funktionalität der Echtzeitverarbeitung beschreibt. Das Petri-Netz beinhaltet drei Arten von Transaktionen. Zur ersten Art von Transaktionen gehören Referenzen auf Funktionen, die eine Kommunikation zwischen dem GUI und dem Nutzer ermöglichen. Es wird hierfür auf die jQuerey-Funktionen *mousemove()*, *mousedown()* und *mouseup* verwiesen. Zur besseren Erkennbarkeit sind auf diese Funktionen verweisenden Referenzen im Petri-Netz kursiv abgebildet. Bei der zweiten Art von Transaktionen handelt es sich um Ergebnisse von Verzweigungen. Die dritte Art von Transaktionen beschreibt umfangreichere Funktionalitäten, welche in Kapitel 5.3.3 tiefgehender beschreiben werden.

Ausgangspunkt ist die Funktion *b_tree_helper()*, welche die Echtzeitverarbeitung initialisiert. Aus der auf diese Funktion referierenden Transaktion heraus entstehen zwei Stellen, welche auf Eingaben des Nutzers warten. Es handelt sich hierbei um die Funktionen *mousemove()* und *mousedown()*. *Mousemove()* wird ausgeführt, wenn der Mauszeiger über den Editor fährt. Aus der Stelle *Mouse selected* heraus gibt es zwei mögliche Folgezustände. Wurde kein Schlüssel bewegt (*No key moved*), wird der nächste Zustand mit der erneuten Belegung der Stelle *Waiting for mousemove()* erreicht. Für die Transaktion *One key is moved* muss jedoch die Stelle *Key is selected* belegt sein. Für die Belegung dieser Stelle muss zuerst die Transaktion *mousedown()* durchlaufen sein. Anschließend können die Transaktionen *Clicked on*

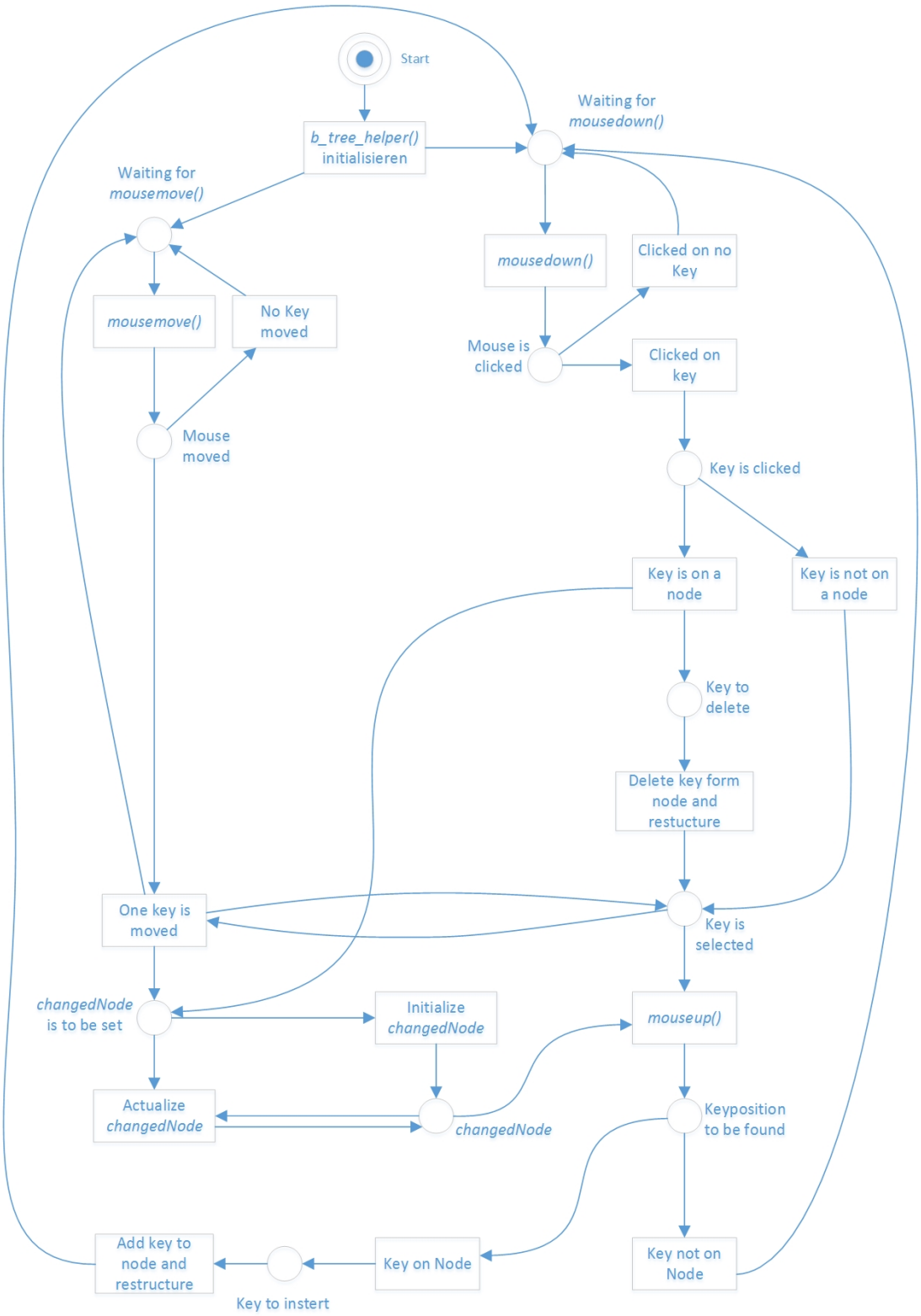

Abbildung 9: Petri-Netz des Grapheneditors

no key oder *Clicked on Key* ausgeführt. Eine Erläuterung des Vorgehens für den direkten Rücksprung zur Stelle *Waiting for mousedown()* über die Transaktion *Clicked on no key* erfolgt gegen Ende des Kapitels. Wurde jedoch auf einen Schlüssel des B-Baums gedrückt (*Clicked on key*), wird im Anschluss die Verzweigung durchlaufen, die zwischen dem Schlüssel auf (*Key is on node*) oder nicht auf einem Knoten (*Key is not on node*) unterscheidet. Liegt der Schlüssel nicht über einem Knoten, wird direkt die Stelle *Key is selected* markiert. Liegt der Schlüssel hingegen über einem Knoten, wird zuerst die Stelle *changeNode is to be set* markiert. Währenddessen wird die Transaktion *Delete key from node and restructure* durchlaufen, sodass die Stelle *Key is selected* schließlich auch über diese Verzweigung markiert wird.

Im folgenden Absatz wird das Teilnetz um die Stellen *changedNode is to be set* und *changedNode* vorgestellt. Bei *changedNode* handelt es sich um eine Variable, die den zuletzt geänderten Schlüssel beinhaltet. Enthält diese Variable einen Wert, dann ist die gleichnamige Stelle im Petri-Netz markiert. Um die Stelle zu markieren, können zwei Transaktionen durchlaufen werden. Zum einen gibt es die Transaktion *Initialize changedNode*, zum anderen die Transaktion *Actualize changedNode*, wobei bei dieser *changedNode* bereits markiert sein muss. Die für beide Transaktionen ausgehende Stelle ist *changedNode is to be set*. Die zur Markierung dieser Stelle notwendigen Transaktionen sind *One key is moved* und *Key is on a node*. Dies bedeutet, dass die Variabel *changeNode* gesetzt wird, sobald ein Schlüssel, der auf einem Knoten liegt, angeklickt, bzw. ausgewählt wird. Des Weiteren wird die Variable gesetzt, sobald ein ausgewählter Knoten – siehe Abhängigkeit zwischen *One key is moved* und *Key is selected* – auf dem Editor bewegt wird.

Eine von der Stelle *Key is selected* nächstmögliche Transaktion ist die Transaktion *mouseup()*, welche auf die jQuery-Funktion *mouseup()* referiert. Diese Funktion wird ausgelöst, sobald der Nutzer die Maustaste löst. Das Petri-Netz zeigt, dass die beiden Stellen *changedNode* und *Key is selected* markiert sein müssen, damit die Transaktion *mouseup* durchlaufen werden kann. Insbesondere muss in diesem Fall zwischen der durch die Transaktion *mouseup()* repräsentierten Funktionalität und der jQuery-Funkion *mouseup()* differenziert werden. Die jQuery-Funktion *mouseup()* wird ausgelöst, sobald der Nutzer die Maustaste löst, folglich auch, wenn kein Schlüssel ausgewählt ist. Für die Transaktion *mouseup()* müssen jedoch die eben beschriebenen Stellten markiert sein. Der auftretende Widerspruch, der durch die Forderung des Petri-Netzes nach einer Belegung dieser Stellen für die Ausführung der Transaktion auftritt, wird durch Betrachtung der in der Funktion *mouseup()* steckenden, durch die Transaktion beschriebenen, Funktionalität aufgelöst. Die der Transaktion *mouseup()* folgende Funktionalität enthält die notwendige Be-

dingung, dass ein veränderter Schlüssel vorhanden ist. Wird diese Bedingung nicht erfüllt, wird die Funktion abgebrochen. Auf Basis dieser Argumentation ist die der Transaktion *mouseup()* folgende Funktionalität nur mit dem Vorhandensein einer *changedNode* verfügbar, obwohl die jQuery-Funktion *mousemove()* ausgelöst worden ist. Hiermit ist ebenso das Enden der Transaktion *Clicked on no Key* in der Stelle *Waiting for mousedown()* begründet. Hiernach wird nach dem Lösen der Maustaste zwar die jQuery-Funktion *mouseup()* ausgelöst, die aus der Funktion resultierende Funktionalität wird jedoch nicht angesprochen. Der Transaktion *mouseup()* folgt eine Fallunterscheidung. Ist der Schlüssel auf einem Knoten abgelegt worden, so wird er zu diesem Knoten hinzugefügt und der Knoten wird neu strukturiert. Anschließend wird die Marke zu der Stelle *Waiting for mousedown()* geleitet. Ist der Schlüssel nicht auf einem Knoten abgelegt worden, so wird die Marke direkt an die Stelle *Waiting for mousedown()* geleitet. Somit befindet sich der Editor wieder im Ausgangszustand.

5.3.3 Implementierungsdetails

In diesem Kapitel wird die Umsetzung der in Kapitel 5.3.2 beschriebenen Funktionalität der Echtzeitverarbeitung des Grapheneditors vorgestellt. Hierfür wird zuerst auf die wichtigen Datenstrukturen eingegangen. Anschließend werden die für die Gesamtfunktionalität wichtigen Teilfunktionalitäten veranschaulicht. Zu diesem Zweck wird insbesondere auf die bereits in Kapitel 5.3.2 vorgestellten Teilfunktionalitäten eingegangen. Insbesondere wird vorgestellt, wie das System erkennt, dass ein Knoten ausgewählt, bewegt und wieder abgelegt wird. Außerdem wird gezeigt, wie die Verwaltung der Schlüssel in einem Knoten funktioniert, wie also erkannt wird, ob der Studierende einen Schlüssel zu einem Knoten hinzugefügt hat.

Neben der Datenstruktur `nodesPerTier` (vgl. 5.3.1) ist eine weitere wichtige Datenstruktur das Array `nodeListBounds`. Diese Datenstruktur enthält die Grenzen der x-Achse, in denen ein Schlüssel als Schlüssel innerhalb eines Knotens erkannt wird. Das Array `nodeListBounds` ist dem Array `nodesPerTier` stark nachempfunden. So handelt es sich ebenfalls um ein dreidimensionales Array, wobei auch hier die Schichten auf der ersten und die Knoten auf der zweiten Dimension abgelegt werden. In der dritten Dimension sind jedoch anstatt von Schlüsseln die Grenzen der einzelnen Knoten gespeichert. Quellcode 4 zeigt, wie über die übergebene Position einem Schlüssel die aus Kapitel 5.3.1 bekannten Variablen i, j und k eindeutig zugeordnet werden können. In der Funktion `getTierAndList` wird jede Schicht durchlaufen (Z. 2) und es wird geprüft, ob die übergebene Position einer Schicht zugeordnet

```
var getTierAndList = function (position) {
  for(var i = 0; i < tier; ++i){
    if (position && 50 * i - 15 < position[1] - 25 &&  position[1]
        - 25 < 50 * i + 15) {
      for(var j = 0; j < Math.pow(t, i); ++j){
        if (nodeListBounds[i][j][0] < position[0] && position[0] <
            nodeListBounds[i][j][1]) {
          for (var k = 0; k <= t; ++k) {
            if ((nodeListBounds[i][j][0]+k*30+15) < position[0] &&
                position[0] < (nodeListBounds[i][j][0]+(k+1)*30+15))
                 {
              return {i: i, j: j, k: k};
            }
          }
          return {i: i, j: j};
        }
      }
    }
  }
  return undefined;
}
```

Quellcode 4: Ausschnitt *b_tree_initialize_helper(...)*

werden kann (Z. 3). Ist dies der Fall, werden die einzelnen Knoten einer Schicht durchlaufen (Z. 4). Nun wird geprüft, ob die übergebene Position einem Knoten zuordenbar ist (Z. 5). Hierfür bedient man sich der zuvor vorgestellten Datenstruktur **nodeListBounds**. Ebendies geschieht anschließend für die Zuordenbarkeit der Position zu einem Schlüssel. Ist eine Zuordnung bestimmbar, so wird diese zurückgegeben. Nach diesem Prinzip funktioniert auch die Einordnung von Schlüsseln in die Datenstruktur **nodesPerTier**.

Eine weitere wichtige Funktion ist die Neustrukturierung eines Knotens. Quellcode 5 zeigt die Neustrukturierung nach dem Hinzufügen eines weiteren Schlüssels. Zuerst wird der geänderte Schlüssel dem entsprechenden Knoten der Datenstruk-

```
nodesPerTier[n.i][n.j].push(changedNode);
nodesPerTier[n.i][n.j].sort(function(a, b) {
        var x = a.position[0];
        var y = b.position[0];
        return ((x < y) ? -1 : ((x > y) ? 1 : 0));
});
for (var k = 0; k < nodesPerTier[n.i][n.j].length; ++k) {
        nodesPerTier[n.i][n.j][k].setPosition(/* ... */);
}
```

Quellcode 5: Ausschnitt *b_tree_initialize_helper(...)*

tur **nodesPerTier** übergeben (Z. 1). Die Position des Knotens im Array wird über die so eben vorgestellte Methode **getTierAndList** bestimmt. Anschließend werden die Schlüssel im Knoten der x-Position nach sortiert (Z. 2 ff.). Abschließend wird den sortierten Schlüsseln nach der in Kapitel 5.3.1 vorgestellten Methode ihre neue Position auf der x-Achse zugewiesen.

5.4 Datenstrukturverarbeitung backendseitig

Die in diesem Abschnitt vorgestellten Änderungen beziehen sich auf die in Kapitel 3.3.2 unter Abschnitt *Datenverarbeitung* vorgestellte Funktionalität. Bei der Datei, in der Änderungen vorgenommen wurden handelt es sich um die **`lib/js_dot_convert.php`**. In ihr befinden sich in der alten Version des EASy-DSBuilders die Klassen **`jsdot_graph`**, **`jsdot_edge`**, **`jsdot_node`** und **`dsbuilder_binary_tree`**. Die Klasse **`jsdot_graph`** dient als Schnittstelle zum restlichen Modul und verwaltet den jsdot-Graph. Zum Umgang mit übergebenen Datenstrukturen bedient die Klasse sich im Fall von Binärbäumen der Klasse **`dsbuilder_binary_tree`**. Die nun vorgenommene Erweiterung enthält die Implementierung der Klasse **`dsbuilder_b_tree`**. Diese neu implementierte Klasse orientiert sich stark an der Klasse **`dsbuilder_binary_tree`**. Sie stellt dieselben Funktionen bereit, nur dass die Funktionen in der Klasse **`dsbuilder_b_tree`** dem Umgang mit B-Baum-Datenstrukturen dienen.

Die drei öffentlichen Funktionen der Klasse **`dsbuilder_b_tree`** sind die statischen Funktionen *initialize_with_b_tree_data_structure(...)*, *get_b_tree_data_structure(...)* und *compare_sample_and_student_graph(...)*. Hierbei ermöglichen die ersten beiden Funktionen die Umwandlung der Datenstruktur, sodass mit der Datenstruktur in den beiden Anwendungsfällen Grapheneditor und Webservices gearbeitet werden kann. Die dritte Funktion vergleicht zwei Graphen und markiert die sich unterscheidenden Knoten. Sie wird für die Feedbackfunktion genutzt. Im Folgenden werden die ersten beiden Funktionen tiefgehender vorgestellt.

Die Funktion *initialize_with_b_tree_data_structure(...)* initialisiert einen neuen **`jsdot_graph`** anhand der in Kapitel 5.2 vorgestellten serialisierten Datenstruktur. Der Funktion können drei Parameter übergeben werden. Die Übergabe einer serialisierte Datenstruktur ist verpflichtend. Ansonsten kann übergeben werden, wie viel Abstand der Baum im Editor auf der x-Achse zum linken Rand und auf der y-Achse zum oberen Rand haben soll. Standardmäßig werden 50 Pixel für die x-Achse und 25 Pixel für die y-Achse übergeben. Jedoch handelt es sich bei der Funktion *initialize_with_b_tree_data_structure(...)* nur um eine Wrapperfunktion. Die eigentliche Umwandlung einer serialisierten Datenstruktur in einen **`jsdot_graph`** geschieht in der Hilfsfunktion *b_tree_initialize_helper(...)*. Quellcode 6 zeigt einen Ausschnitt der Hilfsfunktion. Da es sich bei der serialisierten Datenstruktur um eine rekursiv aufgebaute Datenstruktur handelt ist, die Verarbeitung dieser Struktur ebenfalls rekursiv aufgebaut. Der Hilfsfunktion wird eine serialisierte B-Baum-Datenstruktur übergeben. Wie in Kapitel 5.2 beschrieben, besteht die Datenstruktur aus einem Array, welches Schlüssel und Kindsknoten-Arrays des Wurzelknotens enthält. Auf

```
1  private static function b_tree_initialize_helper(jsdot_graph
       $jsdot_graph, $json_row, $height, $index_array, $spacing_x,
       $spacing_y, $t) {
2    /* ... */
3    $index_node = 0;
4    foreach($json_row as $e){
5      if(gettype($e) == "string"){
6            $node = new jsdot_node (/* ... */);
7            $jsdot_graph->nodes [] = $node;
8            $index_node++;
9      } else {
10           $left = self::b_tree_initialize_helper (/* ... */);
11     }
12   }
13 }
```

Quellcode 6: Ausschnitt *b_tree_initialize_helper(...)*

Grund dieser Struktur wird jedes Element des Wurzelknoten-Arrays durchlaufen (Z. 4). Hierbei werden die einzelnen Knoten-Arrays des übergebenen B-Baums durch die Variable **$json_row** widergespiegelt. Für jedes Element in dem jeweiligen Array erfolgt eine Fallunterscheidung, in der geprüft wird, ob es sich bei dem jeweiligen Element um ein *String* handelt. Ist das jeweilige Element ein *String* (Z. 5), so muss es sich um einen Schlüssel im Array handeln. In diesem Fall wird ein neues **jsdot_node** initialisiert und den Knoten des **$jsdot_graph** übergeben (Z. 6 f.). Es werden hierbei jeweils Name, Schlüssel und Position an den Knoten übergeben. Wie die Position bestimmt wird, wurde in Kapitel 5.3.1 erläutert. Handelt es sich nicht um einen *String*, so muss es sich um ein Kindsknoten-Array handeln. In diesem Fall ruft die Hilfsfunktion sich selber auf, damit die Kindsknoten-Arrays auf die selbe Weise verarbeitet werden können. Quellcode 7 zeigt den rekursiven Aufruf in seiner Gesamtheit. Der erste übergebene Parameter ist wieder der in Funktion

```
1 self::b_tree_initialize_helper ( $jsdot_graph, $e, ($height + 1),
    $index_node + $index_array * $t, $spacing_x, $spacing_y, $t)
```

Quellcode 7: rekursiver Aufruf aus Quellcode 6 in Z. 10

initialize_with_b_tree_data_structure(...) initialisierte **jsdot_graph**. Der zweite Parameter ist das Kindsknoten-Array. Die nächsten beiden Parameter beschreiben die Position des Arrays. Als erstes wird die Höhe übergeben, die eine größer als die momentane Höhe ist. Danach wird die Position dieses Kindknotens innerhalb einer Schicht weitergegeben. **$index_node** sagt dabei aus, um den wievielten Kindsknoten es sich handelt. **$index_array** zeigt die Position des Wurzelknotens innerhalb der Schicht auf. Diese muss noch mit der maximalen Verzweigung **$t** multipliziert werden, damit die richtige Position für den Kindsknoten gefunden wird. Die Summe bildet schließlich die Finale Position. Die restlichen Parameter werden unverändert

weitergegeben. Zum Abbruch der Funktion dienen die kinderlosen Knoten, da in ihnen keine Kindsknoten-Arrays zum weiteren rekursiven Aufruf vorhanden sind.

Die Funktion *get_b_tree_data_ structure(...)* wandelt wiederum eine **jsdot_graph**-Datenstruktur in eine serialisierte Datenstruktur für den Webservice um. Quellcode 8 zeigt die gesamte Funktion. So wird der **jsdot_graph** als erstes in

```
1 public static function get_b_tree_data_structure(jsdot_graph
      $jsdot_graph, $t) {
2         $nodesPerTier = self::get_b_tree_nodes_per_tier_structure(
              $jsdot_graph, $t);
3         $result = self::b_tree_structure_helper($nodesPerTier
              [0][0], $nodesPerTier, 0, 0, $t);
4         return json_encode ( [$result, $t] );
5 }
```

Quellcode 8: *get_b_tree_data_ structure(...)*

die Form „Knoten pro Schicht“ umgewandelt (Z. 2). Da diese Form bereits aus Kapitel 5.3.1 bekannt ist, und die Umwandlung hier äquivalent verläuft, wird an dieser Stelle nicht weiter darauf eingegangen. Wichtig für diese Funktion ist jedoch noch, dass in ihr die Richtigkeit der Syntax des B-Baums geprüft wird. Es wird geprüft, ob alle Schlüssel Teil des Baums sind, ob alle Knoten die maximale Anzahl an Schlüsseln nicht überschreiten und ob der Baum ausbalanciert ist. Anschließend wird die rekursive Hilfsfunktion *b_tree_structure_helper(...)* aufgerufen, die die eigentliche Umwandlung umsetzt. Die Hilfsfunktion wird mit dem Wurzelknoten des B-Baums initialisiert (**$nodesPerArray[0][0]**). Weiterhin wird noch der gesamte B-Baum, die Position des Wurzelknotens und die maximale Verzweigung **$t** übergeben. Quellcode 9 zeigt die Hilfsfunktion. In der Funktion wird jeder Schlüssel des übergebenden

```
1  private static function b_tree_structure_helper($nodeList,
       $nodesPerTier, $i, $j, $t){
2      $result = array();
3      $index = 0;
4      foreach ($nodeList as $node){
5          if ($nodesPerTier[$i+1][$j*$t + $index]){
6              $r = self::b_tree_structure_helper(/* ... */);
7              array_push($result, $r);
8          }
9          array_push($result,   $node->label->value);
10         if ($nodesPerTier[$i+1][$j*$t + $index+1]  && sizeof(
               $nodeList)-1 == $index){
11             $r = self::b_tree_structure_helper(/* ... */);
12             array_push($result, $r);
13         }
14         $index++;
15     }
16     return $result;
17 }
```

Quellcode 9: *b_tree_structure_helper(...)*

Knotens (**`$nodeList`**) durchlaufen (Z. 4). Für jedes Element wird zuerst geprüft, ob ein linker Kindsknoten existiert (Z. 5). Ist dies der Fall, ruft die Funktion sich selbst auf, wobei der Kindsknoten als zu bearbeitendes Element übergeben wird (Z. 6). Anschließend wird das Ergebnis dieses rekursiven Aufrufs zum Ergebnis hinzugefügt (Z. 7). Auch jeder Schlüssel wird dem Ergebnis übergeben (Z. 9). Die zweite if-Abfrage (Z. 10) prüft, ob es sich um den letzten Schlüssel eines Knotens handelt, und ob dieser einen rechten Kindsknoten hat. Ist dies der Fall, wird für diesen Kindsknoten ein weiterer rekursiver Aufruf initiiert (Z. 11). Als abbrechende Bedingung für die Rekursion dienen die Kinderlosen Knoten.

6 Fazit und Ausblick

In dieser Arbeit wurde die Erweiterung des Moodle-Moduls EASy-DSBuilders um das Assessment von B-Baum-Datenstrukturen dokumentiert. Hierzu wurde eingangs der EASy-DSBuilder in seiner Struktur und Funktionalität vorgestellt. Anschließend folge eine Anforderungsanalyse zur Erweiterung des Moduls um die Funktionalität eines Assessments von B-Baum-Datenstrukturen. Darauf folgend wurden die Stellen im Modul vorgestellt, an denen Modifikation und Erweiterungen durchzuführen waren, um die auf Basis der Analyse erarbeiteten Anforderungen umsetzen zu können. Als Ergebnis kann ein funktionsfähiges Modul in seiner zweiten Version präsentiert werden, welches das Assessment von B-Baum-Datenstrukturen unterstützt.

So kann der Lehrende in dieser zweiten Version des EASy-DSBuilders bei der Initialisierung einer neuen Instanz dieses Moduls zwischen den zu prüfenden Datenstrukturen wählen. Dem Studierenden wurde hingegen ein erweiterter Editor bereitgestellt, der für das Assessment von B-Bäumen optimiert ist. Dem Studierenden wird die Möglichkeit geboten, die auf dem Editor befindlichen Schlüssel intuitiv zu bewegen und auf einer B-Baum-Grundstruktur im Hintergrund, welche sich weiterhin automatisch anpasst, abzulegen. Dabei wurden sämtliche Funktionalitäten des Moduls der ersten Version beibehalten und auf das neue Assessment übertragen. Hierzu gehört die Möglichkeit, im zehnschrittigen Assessment durch die einzelnen Schritte zu navigieren. Eine weitere wichtige Funktion ist die des Feedbacks. So kann der Lehrende auch für das Assessment von B-Bäumen ein Feedback einstellen, welches dem Studierenden Informationen darüber gibt, welche Fehler er bei seiner Eingabe gemacht hat.

Eine Möglichkeit der Verbesserung des Moduls liegt in der Optimierung der Positionierung der Elemente eines B-Baums im Editor. In der gegenwärtigen Version wird jedem Element ein fester Platz zugeordnet. So bleiben auf der horizontalen Achse Freiräume zwischen einzelnen Elementen, wenn eine Schicht nicht voll belegt ist. Auf Grund des exponentiellen Wachstums und der vollen Ausgeglichenheit der Höhe braucht der B-Baum insbesondere bei höherer Ordnung m so viel Platz auf der x-Achse, dass die Usability eingeschränkt wird. Hier sind die unnötigen Zwischenräume insbesondere in der Blattschicht zu eliminieren, um den Gesamtbaum zu Verschlanken.

A Anhang

A.1 Weiterer Quellcode

```
1  package de.wwu.pi.treeBuilder.api;
2
3  public interface DataStructureWrapperInterface<K extends Comparable
       <K>, D> {
4          /**
5           * Initializes the data structure and returns an empty data
                structure object.
6           *
7           * @param random
8           *              a "pseudo" random number that can be used to
                 randomly initialize the data structure.
9           * @return returns the data structure object
10          */
11         public DataStructureWrapperInterface<K, D> initializeEmpty(
               double config);
12
13         /**
14          * adds an element (key + data) to the data structure.
15          *
16          * @param key
17          *              key to be added to the DS
18          * @param data
19          *              data to be added to the DS
20          */
21         public void add(K key, D data);
22
23         /**
24          * removes the element with the key 'key' from the DS
25          *
26          * @param key
27          *              key to be removed from DS
28          */
29         public void remove(K key);
30
31         /**
32          * Deserializes data structure and returns a new data
                structure object. The serialized object was created
                with the
33          * serialize method of this class. Important recompute
                hidden values (e.g. balance of nodes within AVL tree)
                after
34          * deserialization of an object.
35          *
36          * @param serialDataStructure
37          *              data structure as JSONString, created with
                the serialize method of this class.
38          * @return data structure object, to be manipulated later.
39          */
40         public DataStructureWrapperInterface<K, D> deserialize(
               String serialDataStructure);
41
42         /**
43          * Serializes the data structure as a JSON String. Hidden
                data e.g. balance flag of a node within an AVL tree
                should
44          * be recomputed after deserialization instead of parsing
                it to the serialized string.
45          *
46          * @return JSON string representing the data structure.
47          */
```

```
48          public String serialize();
49
50          /**
51           * A generator method to be called to generate the next key
                , that should be inserted by the student.
52           *
53           * @param random
54           *              random value, that should be used to generate
                 the next key. Same random values should lead to the
                same
55           *              key.
56           * @return key to be inserted
57           */
58          public K generateKeyToInsert(double random);
59
60          /**
61           * A generator method to generate Data for a key.
62           *
63           * @param key
64           * @return
65           */
66          public D generateDataToKey(K key);
67
68          /**
69           * serializes a given key to String
70           *
71           * @param key
72           *              key to be serialized
73           * @return serialized String
74           */
75          public String serializeKey(K key);
76
77          /**
78           * serializes a given data element to String
79           *
80           * @param data
81           *              data object to be serialized.
82           * @return serialized data object
83           */
84          public String serializeData(D data);
85
86          /**
87           * deserializes a given string representation of a key
                object
88           *
89           * @param serialKey
90           *              string representation of the key
91           * @return key object
92           */
93          public K deserializeKey(String serialKey);
94
95          /**
96           * deserializes a given string representation of a data
                object
97           *
98           * @param serialData
99           *              string representation of the data
100          * @return data object
101          */
102         public D deserializeData(String serialData);
103
104         /**
105          * Info Text for a student.
106          * @return helptext that can be used to give detailed info
                on how what to to in an action/ Might be empty.
107          */
```

```
108         public String getAdditionalHelpText();
109 }
```

Quellcode 10: **DataStructureWrapperInterface.java**

A.2 Inhalte der beiliegenden CD

Dieser Bachelorarbeit liegt eine CD mit folgendem Inhalt bei:

- Die Abgabeversion der Bachelorarbeit als PDF-Datei.
- Alle LaTeX-Dateien, die zur Kompilierung der Abgabeversion der Bachelorarbeit benötigt werden.
- Das in dieser Arbeit weiterentwickelte Moodle-Modul EASy-DSBuilder
- Der Verlauf (GIT-Repository) der Programmierung

Literatur

[Bal14] Helmut Balzert. *Lehrbuch der Softwaretechnik: Entwurf, Implementierung, Installation und Betrieb*. Spektrum Verlag, 2014.

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

[Blu13] Norbert Blum. *Algorithmen und Datenstrukturen*. Oldenburg Verlag München, 2013.

[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. `http://wildfly.org/news/2014/02/11/WildFly8-Final-Released/`. Aufgerufen am: 2015-02-28.

[HK11] Key Hoeksem and Markus Kuhn. *Unterricht mit Moodle 2*. 2011.

[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.

[Moo15a] MoodleDocs. About Moodle. `https://docs.moodle.org/28/en/About_Moodle`, 2015. Aufgerufen am: 2015-03-04.

[Moo15b] MoodleDocs. Activities. `https://moodle.org/plugins/browse.php?list=category&id=1`, 2015. Aufgerufen am: 2015-03-12.

[Moo15c] MoodleDocs. Activity Modules. `https://docs.moodle.org/dev/Activity_modules`, 2015. Aufgerufen am: 2015-03-04.

[Moo15d] MoodleDocs. Course formats. `https://docs.moodle.org/dev/Course_formats`, 2015. Aufgerufen am: 2015-05-12.

[Moo15e] MoodleDocs. JavaScript usage guide. `https://docs.moodle.org/dev/JavaScript_usage_guide`, 2015. Aufgerufen am: 2015-03-06.

[Moo15f] MoodleDocs. Local plugins. https://docs.moodle.org/dev/Local_plugins, 2015. Aufgerufen am: 2015-03-11.

[Moo15g] MoodleDocs. Moodle architecture. https://docs.moodle.org/dev/Moodle_architecture, 2015. Aufgerufen am: 2015-03-09.

[Moo15h] MoodleDocs. NEWMODULE Documentation. https://docs.moodle.org/dev/NEWMODULE_Documentation#lib.php, 2015. Aufgerufen am: 2015-03-04.

[Moo15i] MoodleDocs. Output renderers. https://docs.moodle.org/dev/Output_renderers, 2015. Aufgerufen am: 2015-03-06.

[Moo15j] MoodleDocs. Was ist Moodle. https://docs.moodle.org/28/de/Was_ist_Moodle, 2015. Aufgerufen am: 2015-02-26.

[Rue10] Cornelia Ruedel. *E-Assessment*. Waxmann Verlag GmbH, 2010.

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

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

[SS14] Gunter Saake and Kai-Uwe Sattler. *Algorithmen und Datenstrukturen*. dpunkt.verlag, 2014.

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

Ich versichere hiermit, dass ich meine Bachelor-Abschlussarbeit „*Erweiterung eines E-Learning-Moduls zum Assessment von B-Baum-Datenstrukturen*“ 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 30.03.2015 _____

David Bujok