Fachanwendung für digitale Modulkataloge

Eine Untersuchung zu Graph-basierter Daten-Modellierung und Navigation

Vera G. Meister [1], Wenxin Hu [2], Aleksandra Revina [3], Marcel Cikus [4], Johannes Müller[5]

Abstract: Hochschulen nutzen eine Vielzahl von Fachanwendungen für die verschiedensten Prozesse zur Konzeption, Publikation, Verwaltung, Planung, Durchführung und Leistungsabrechnung von Modulen, oftmals mit redundanter Datenhaltung. Fehlende Integration der Anwendungen sorgt für mangelhafte Datenqualität und substanzielle Mehrarbeit. Das Paper untersucht mit Hilfe eines Prototyps, inwiefern eine Graph-basierte Datenhaltung ein Ansatz zur Lösung des Integrationsproblems sein kann. Zugleich wird erforscht, ob eine Graph-Navigation bei den Nutzer*innen auf Akzeptanz stößt und in welchem Maße sie die Orientierung in vernetzten Strukturen fördert.

Keywords: Digitaler Modulkatalog; Wissensgraph-Anwendung; Hochschul-IT; integrierte Datenhaltung; Graph-basierte Navigation; Prototyp; Softwareevaluation

1 Einleitung

Im Ergebnis der Bologna-Reform wurden die Studiengänge an Hochschulen modularisiert, d. h. sie setzen sich aus einzelnen, fachlich differenzierbaren, durch Prüfung abschließbaren Lerneinheiten – den Modulen – zusammen [Ku10]. Diese Lerneinheiten sind zu konzipieren, zu spezifizieren, zu publizieren, mit Ressourcen zu belegen und schließlich durchzuführen und abzuschließen. Ein Modul ist damit zum einen ein ideelles Konzept, zum anderen eine Veranstaltung in Raum und Zeit.

In einer digitalen Hochschullandschaft sollten alle Prozesse rund um diese zentralen Entitäten durch digitale Werkzeuge – sogenannte Fachanwendungen – unterstützt werden. Idealerweise sollten die Daten zu den Modulen und weiteren verbundenen Entitäten integriert sein, d. h. nicht mehrfach in verschiedenen, isolierten Datenbanken verarbeitet und gespeichert werden. Die IT-Landschaften deutscher Hochschulen sind häufig aufgrund verteilter Verantwortlichkeiten in den Instituten mäßig bis gar nicht integriert.

[1] Technische Hochschule Brandenburg, Fachbereich Wirtschaft, Magdeburger Straße 50, 14770 Brandenburg an der Havel, Deutschland, vera.meister@th-brandenburg.de, https://orcid.org/0000-0002-2780-0222
[2] ebenda, hu@th-brandenburg.de, https://orcid.org/0000-0003-3449-5980
[3] ebenda, revina@th-brandenburg.de, https://orcid.org/0000-0002-8405-0018
[4] ebenda, cikus@th-brandenburg.de, https://orcid.org/0000-0001-6715-2146
[5] ebenda, johannes.mueller@th-brandenburg.de

doi:10.18420/inf2020_43

Aufbauend auf Vorarbeiten [MB18; MHP19] soll in diesem Paper die prototypische Implementierung einer Fachanwendung für digitale Modulkataloge untersucht werden. Der Fokus wird auf die integrativen Fähigkeiten einer Graph-basierten Datenhaltung und den Einfluss einer solchen Navigation auf Nutzerakzeptanz gelegt. Dafür werden in Abschnitt 2 Motivation und Forschungsfragen ausgearbeitet sowie in Abschnitt 3 der Stand der Forschung reflektiert. In den Abschnitten 4 Prototypentwicklung und 5 Evaluation werden die Methoden zur Beantwortung der Forschungsfragen eingeführt und die Ergebnisse dargestellt. Das Paper schließt mit einem Fazit und Ausblick in Abschnitt 6.

2 Motivation und Forschungsfragen

Wie in [MHP19] gezeigt, ist die Bandbreite an eingesetzten Fachanwendungen für Modulkataloge an Hochschulen sehr groß. Im nicht seltenen Extremfall gibt es für jeden Prozess rund um Module ein anderes Werkzeug: Office-Tools und/oder Dokumenten-Management-Systeme (DMS) für die Konzeption und Bereitstellung von Modulbeschreibungen, Content-Management-Systeme (CMS) für die Veröffentlichung auf Hochschulwebseiten, komplexe Tabellenkalkulationen für die Planung von Lehrbedarfen und -verpflichtungen, Spezialanwendungen für die Stunden- und Raumplanung, Lernmanagement-Systeme (LMS) für die didaktische und organisatorische Steuerung der Lernprozesse sowie Campus-Management-Systeme (CaMS) für das Prüfungsmanagement im Student Life Cycle. Daten zu Modulen werden in einem solchen Setting vielfach redundant erfasst und gespeichert, was zu mangelhafter Datenqualität und substanziellem Mehraufwand führt.

Um eine integrierte Datenhaltung herzustellen gibt es vier verschiedene Ansätze [Kr20]:

1. Ablösung der einzelnen Fachanwendungen durch ein integriertes komplexes System mit einheitlicher Datenhaltung,
2. Einführung einer integrierenden Middleware basierend auf einem vorkonfigurierten, einheitlichen Datenschema,
3. Programmierung dedizierter Schnittstellen für die Point-to-Point-Integration der verschiedenen Fachanwendungen,
4. Einführung eines flexibel erweiterbaren, Graph-basierten Daten-Hubs, der die Relationen und Interdependenzen zwischen den verschiedenen Datenquellen expliziert und zugänglich macht.

Einige große Hersteller von CaMS versuchen sich an Lösungsansatz 1. Die Praxis zeigt jedoch, dass nicht nur die komplexen Einführungsprozesse, sondern auch die Spezifik der verschiedenen Hochschulen diesem One-Fits-All-Ansatz zuwiderlaufen. Ansatz 2 erfordert ein vollständig ausgearbeitetes Integrationsmodell, das nur mit hohem Aufwand an Änderungen angepasst werden kann. Der wesentliche Nachteil von Ansatz 3 besteht in seinem hohen Pflegeaufwand, der quadratisch zu jeder weiteren zu integrierenden Anwendung ansteigt. Ansatz 4 wiederum ist offen im Hinblick auf die Datenmodelle der zu integrierenden

Anwendungen und zugleich flexibel erweiterbar. Der größte „Nachteil" besteht hier darin, dass es sich um eine Technologie handelt, der unter IT-Verantwortlichen und Softwareentwicklern eine steile Lernkurve bescheinigt wird [Tu20]. Zugleich weist [Tu20] nach, dass im letzten Jahrzehnt Graph-Technologien in einer Reihe von Forschungsfeldern (wie Biomedizin) und Branchen (Finanzwirtschaft, Maschinenbau u. a.) eine weite Verbreitung gefunden haben.

Der Einsatz von Graph-Technologien ist besonders dann angezeigt, wenn es sich um stark vernetzte Daten handelt, die zudem verschiedenen Fachsichten unterliegen, wie es sich auch im vorliegenden Anwendungsfall darstellt. So sind aus Sicht des Studiengangmanagements Module ideelle Konzepte, die mit Lehrenden, Organisationseinheiten, didaktischen Methoden, Ordnungen, Lehr- und Prüfungsformen etc. vernetzt sind. Aus Sicht der Ressourcenplanung handelt es sich um (wiederkehrende) Veranstaltungen mit räumlichen, personellen und weiteren Anforderungen. Beide Sichten sind miteinander verbunden, ergänzen einander und erlauben zudem eine Vielzahl weiterer prozessspezifischer Projektionen. Somit erscheint es grundsätzlich angezeigt, auf der Datenintegrationsebene mit Graph-Technologien zu arbeiten. Davon abzugrenzen ist die Frage, ob diese Graph-Sicht auch auf die Endnutzerebene ausgedehnt werden kann und soll. Die bereits zitierte steile Lernkurve [Tu20] kann sich negativ auf die Akzeptanz auswirken, könnte aber zugleich einen wertvollen Beitrag zur Bekanntheit der Technologie als Ganzes leisten und damit die Hürden für einen breiteren praktischen Einsatz senken.

Aus dem Vorgesagten ergeben sich die folgenden Forschungsfragen:

1) Wie ist ein potenziell erweiterbares Graph-Schema für digitale Modulkataloge zu konzipieren?
2) Mit welchen Front-End-Technologien kann eine Graph-basierte Navigation implementiert werden?
3) Wie wirkt sich die Graph-Navigation auf die Nutzerakzeptanz der Fachanwendung für digitale Modulkataloge aus?

Zur Beantwortung der Forschungsfragen 1) und 2) wird die Methode des Prototyping angewandt. Die Beurteilung der Nutzerakzeptanz gemäß Forschungsfrage 3) erfolgt auf Basis einer qualitativen Untersuchung, die darüber hinaus der Ableitung von Anforderungen für die Weiterentwicklung der Fachanwendung dient.

3 Stand der Forschung

Entlang der drei Forschungsfragen sollen in diesem Abschnitt zunächst die relevanten Forschungsfelder adressiert und der Forschungs- und Entwicklungsstand mit Bezug zur jeweiligen Fragestellung dargestellt werden.

3.1 Graph-basierte Daten-Modellierung

Jede Fachanwendung basiert auf der Konzeption und der technischen Implementierung eines domänenspezifischen Datenmodells. Formal und logisch stringente, fachlich adäquate sowie potenziell erweiterbare Graph-basierte Datenmodelle zu entwickeln, erfordert neben einer tiefen Kenntnis der fachlichen Domäne zugleich ein ganzes Bündel weiterer Kompetenzen: Abstraktionsvermögen, Verständnis von Mustern und Antimustern, Orientierung in bestehenden Vokabularen und Spezifikationen, Beherrschung geeigneter Tools. Zu den letzten drei Feldern gibt es eine Fülle an neueren Forschungs- und Entwicklungsergebnissen.

Der Begriff Ontology Design Pattern (ODP) wurde 2005 von Aldo Gangemi geprägt [Ga05]. Die bis heute aktive Forschungs-Community stellt ihre Ergebnisse auf einem semantischen Wiki der breiten Nutzung zur Verfügung [OD20]. Für den vorliegenden Anwendungsfall sind insbesondere die Content-ODPs und im Hinblick auf Nachnutzung bzw. Erweiterung vorhandener Ontologien (Schemata) die Alignment-ODPs relevant. [Su12] thematisiert die Nutzung von Mustern für die visuelle Darstellung von Graph-Schemata. Eine umfassende Darstellung der Forschung zu ODPs findet sich in [Hi16].

Die Entwicklung von Vokabularen und Spezifikationen für die Graph-basierte Daten-Modellierung hat im letzten Jahrzehnt zahlreiche wertvolle Ergebnisse hervorgebracht. Für den Anwendungsfokus von besonderem Nutzen sind drei mächtige Spezifikationen des W3C: Turtle, SPARQL und SHACL[1]. Auch bei den Vokabularen in der Anwendungsdomäne sind die Ergebnisse zweier vom W3C kuratierter Community-Groups besonders relevant: zum einen schema.org mit Fokus auf `schema:Course` und `schema:CourseInstance` und zum anderen das Verifyable Credentials Data Model.

Bei den Tools für die praktischen Modellierung von Daten-Graphen, lassen sich drei Konstruktions-Paradigmen unterscheiden: (i) Code, (ii) Baum- und (iii) Graph-Struktur. Für alle drei Paradigmen gibt es Editoren bzw. auch komplexere Tools, allerdings folgt die Mehrzahl dem Baumstruktur-Paradigma. Zu diesen Werkzeugen gehört der klassische Open-Source-Editor Protégé [Mu15] ebenso wie das semantische Wiki (ebenfalls Open Source) OntoWiki [FAM16] sowie die leistungsfähigen kommerziellen Tools Top-BraidComposer [To20], Corporate Memory [Br19] und PoolParty [Po20b]. Alle Tools verfügen über Features oder PlugIns, um die Schema-Graphen zu visualisieren sowie über Export-Funktionen, um RDF-Dateien (also reinen Code) auszugeben. Dennoch besteht der größte Nachteil dieser Werkzeuge darin, dass ein sehr starker Fokus auf die Klassenhierarchie gelegt wird, während die Modellierung sämtlicher anderer Relationen zwischen Klassen nur mittelbar unterstützt wird.

Grafische Editoren sind weniger weit verbreitet. OWLGrEd [Ce19] ist ein Open-Source-Editor, der sich in der Visualisierung an UML-Klassendiagrammen orientiert, während Grafo [Da20] (ein kommerzielles Tool) die VOWL-Technologie [Lo16] implementiert. Bei beiden sind individuelle grafische Anpassungen nur bedingt möglich. Allerdings verfügen auch sie über Export-Funktionen. Der Support von Code-Editoren (z. B. rdfEditor

[1] https://www.w3.org/TR/turtle/, https://www.w3.org/TR/turtle/, https://www.w3.org/TR/shacl/

[Do20]) beschränkt sich auf Syntax-Highlighting und Syntaxprüfung für verschiedene Serialisierungsformate. Der größte Vorteil eines Code-Editors besteht darin, dass Struktur und Schlankheit des Codes in der Hand der Entwickler*innen verbleiben. Schließlich sei erwähnt, dass es eine Reihe von Werkzeugen für die verteilte Modellierung gibt. Genannt sei hier nur Quit Store [ARM16], basierend auf Git-Technologien. Dank der Import- und Export-Funktionen bzw. der Einbindung von PlugIns können Entwickler*innen nach Bedarf von einem Präsentationsparadigma zu einem beliebigen anderen wechseln.

3.2 Front-End-Technologien für die Graph-Navigation

Grafische Modelle dienen allgemein der Visualisierung komplexer Zusammenhänge. Sie erleichtern damit die Orientierung und verbessern die Aufnahme zusammenhängender Informationen. Dafür gibt es eine Vielfalt von Anwendungsgebieten: Ablaufdiagramme und Prozessmodelle, Strukturdiagramme, Organigramme, Netzpläne und nicht zuletzt semantische Graph-Schemata. Graphische Editoren verfügen zwangsläufig auch über Elemente einer Graph-Navigation, um Detailinformationen zu Graph-Elementen zu erfassen bzw. wiederzugeben. Aktuelle Beispiele dafür sind der Camunda Modeler [Ca20] für die Modellierung von Geschäftsprozessen und -entscheidungen sowie der bereits erwähnte Ontologie-Editor Grafo [Da20]. Aber auch Informationsangebote und Dokumentationen können von einer Graph-Navigation profitieren, wie z. B. die Process Instance View im Camunda Cockpit [Ca20] oder WebVOWL [Lo16] zur Graph-basierten Dokumentation von Ontologien.

Für die technische Umsetzung in Web-Anwendungen kommen dabei folgende Technologien zum Einsatz: XML, insbesondere SVG eingebettet in HTML5 für die Serialisierung der Graphen selbst, CSS für diverse Style-Definitionen, JavaScript-Bibliotheken für die Generierung und das Event-Handling von Graph-Elementen (z. B. D3.js) eingebettet in ein modernes Entwicklungs-Framework (z. B. Vue.js). Eine idealtypische Kombination dieser Technologien findet sich in einer neueren Arbeit zur multi-disziplinären Design-Optimierung [Al19]. Dort wird der Ansatz zur Automatisierung einer Kette von Design-Tools kritisch diskutiert und nachgewiesen, dass der Aufwand zur Vollautomatisierung zwischen 60-80% der gesamten Projektzeit in Anspruch nimmt. Erschwerend kommt hinzu, dass viele grafische Design-Tools (z. B. CmapTools [Cm20], Camunda Modeler [Ca20]) zwar eine Ausgabe in SVG unterstützen, diese jedoch weit hinter den Spezifikationsmöglichkeiten des Formats zurückbleiben. So werden z. T. Texte oder Kanten als Punktpfade serialisiert, keinerlei IDs für die Graph-Elemente übergeben etc. Letzteres ist für eine eindeutige Adressierung im Web besonders notwendig.

3.3 Graph-basierte Benutzerführung

Nach [Vo09, S. 126] ist die Mehrzahl der klassischen Web-Anwendungen hierarchisch organisiert, um den Nutzer*innen Zugang zu Inhalt und Funktionen der Anwendung bereitzustellen. Typische hierarchische Navigationsmuster sind horizontale und vertikale

Menüs, Listen und Baumstrukturen. Hier sind zwei semantische Ordnungsmuster erkennbar: (1) Differenzierung nach Kategorien und (2) Über- bzw. Unterordnung. Beide sind eher grob und geben nur eingeschränkt Auskunft über Zusammenhänge und Strukturen. Zugleich sind sie dadurch schnell zu erfassen und platzsparend. Letzteres ist insbesondere bei responsiven Webdesigns eine Kernanforderung (vgl. z. B. [Kr19]).

Diese Vorteile büßt eine hierarchische Navigation dann ein, wenn wesentliche semantische Charakteristika der betreffenden Anwendungsdomäne nicht oder nur unzureichend durch die Listen- oder Baumstruktur abbildbar sind. Das ist z. B. offensichtlich bei Anwendungen mit Geo-Lokation der Fall. Hier tritt die (Land-)karte als zentrales Navigationsfeature in den Mittelpunkt. Andere Beispiele finden sich z. B. im Geschäftsprozessmanagement, wenn Fachanwendungen über Prozesslandkarten navigierbar sind.

Aus der Kognitionsforschung ist bekannt, dass die Visualisierung von Verbindungen durch Graph-Kanten zu einer effektiveren Wahrnehmung struktureller Zusammenhänge führt [Wa13, S. 183]. Damit bietet Graph-basierte Navigation nicht nur einen semantisch reicheren Zugang zu Inhalten und Funktionen, sondern auch eine bessere Orientierung in der Domäne und respektive in der Anwendung.

Eine Reihe von Arbeiten beschäftigen sich mit der Gegenüberstellung des Nutzerverhaltens in Graph- und Hierarchie-geleiteten Informationssystemen [Sa16] sowie mit der Visualisierung von Graph-Strukturen generell [Po20a]. Zentrale Erkenntnisse sind hier, dass Graph-Visualisierungen die Orientierung in komplexen Wissensdomänen beschleunigen. Allerdings gehen diese positiven Effekte dann verloren, wenn die Anzahl der durch Kanten verbundenen Knoten zu groß wird.

4 Prototypenentwicklung

Um die Forschungsfragen beantworten zu können, musste der zu entwickelnde Prototyp eines Digitalen Modulkatalogs mit den gewünschten Fähigkeiten zur Datenintegration als vertikaler Prototyp über alle drei Schichten: Datenhaltung, Fachlogik und Benutzeroberfläche konzipiert werden. Bei der Entscheidung für einen „horizontalen“ Fokus des Prototyps fiel die Wahl auf folgende Fachanforderung, die in den Studiengängen zu einem hohen Arbeitsaufwand führt und für die in [MB18] eine mangelhafte Unterstützung durch bestehende IT-Systeme nachgewiesen wurde: „Lehrende sollen dedizierte Editierrechte für die von ihnen verantworteten Module erhalten.“ Als Durchstichlösung wurde zudem der PDF-Download von Modulbeschreibungen implementiert.

4.1 Passfähigkeit und Erweiterbarkeit des Graph-Schemas

In [MB18] wurde bereits ein auf Schema.org basierendes Graph-Schema für Digitale Modulkataloge entwickelt und begründet. [MHP19] berichtet u. a. von ersten Experimenten zur semi-

automatischen Population des Wissensgraphen. Als Schema-Referenz wurden die Modulbeschreibungen im Fachbereich Wirtschaft der Hochschule verwendet. Im Ergebnis eines Workshops mit allen Fachbereichen und zentralen Einrichtungen der Hochschule wurde eine Vielzahl weiterer Anforderungen erhoben und priorisiert. Daraus ergab sich die Notwendigkeit, das Graph-Schema auf Erweiterbarkeit in dreierlei Hinsicht zu prüfen: (i) systematische und klassifizierte Darstellung von Lernzielen statt einer einfachen Auflistung, (ii) Adaptation des semiautomatischen Populationsprozesses auf weitere Studiengänge, (iii) Ergänzung von Planungsgrößen für die Veranstaltungsplanung. Die Erkenntnisse daraus werden im Folgenden dargelegt.

Die kompetenzorientierte Definition von Lernzielen gehört zu den Kernpunkten der Bologna-Reform. Im initialen Schema wurden die Lernziele als ungeordnete, identifizierbare Liste (`schema:ItemList`) über die generische Relation `schema:about` codiert. Jedes einzelne Lernziel erschien ausschließlich als Literal und war somit nicht weiter spezfizierbar. Der Wertebereich der genutzten Property (`schema:itemListElement`) in Schema.org umfasst nicht nur die Datentypklasse `schema:Text`, sondern auch die Objektklasse `schema:ItemList`. Dank dieser Modifikation konnten die Lernziele als Entitäten codiert werden. Über `schema:position` werden jedem Lernziel eine Ordnungsnummer und über `schema:additionalType` die Klassifikation in Kompetenzarten (fachliche, soziale, personale) zugewiesen. Im Fall fachlicher Kompetenzen kann darüber hinaus eine Bloom'sche Taxonomiestufe (z. B. Verstehen) deklariert werden. Dieses partielle Refactoring des Schemas hatte keinen Einfluss auf den Rest der Datenbasis.

Die Modulbeschreibungen liegen als Texte mit schwach standardisierter Tabellenstruktur vor. Die Variationsbreite ist dabei selbst innerhalb eines einzigen Modulkatalogs so hoch, dass manuelle Vor- und Nachbereitungsschritte unabdingbar sind. Getestet wurden Populationsprozesse auf Basis von Python und Excel. Es hat sich gezeigt, dass die Toolchain für jeden Fachbereich und im Detail auch für jeden Modulkatalog etwas angepasst werden muss. Mit Hilfe von Python-Skripten können die Rohdaten geparst und vorstrukturiert werden. Die Transformation dieser tabellarischen Daten in RDF wurde in dieser ersten Forschungsphase mit Excel unter Verwendung komplexer Textfunktionen und Makros umgesetzt. Es ist geplant, bei der anstehenden Erweiterung auf den dritten Fachbereich der Hochschule mit OpenRefine [Op20] zu arbeiten und damit den Anteil manueller Aufgaben weiter zurückzudrängen. Alle verwendeten Skripte und Dokumente finden sich auf GitHub[2]. Die Erfahrung hat gezeigt, dass die Erweiterung der Datenbasis auch mit Schema-Anpassungen einher geht. Im aktuellen Fall betraf das insbesondere die Anbindung der Module an Studiengänge. Aus diesem Grund wurde die ohnehin etwas umständliche Codierung über `schema:AlignmentObject` zugunsten des flexibler nutzbaren `schema:PropertyValue` ersetzt. Die Änderungen an der bestehenden Datenbasis konnten mit Hilfe des SPARQL-Update-Protokolls umgesetzt werden.

Wie bereits erwähnt, ist ein Modul zum einen ein ideelles Konzept (`schema:Course`), zum anderen eine Veranstaltung in Raum und Zeit (`schema:CourseInstance`). Während die bisher betrachteten Refactoring-Schritte die konzeptionelle Seite betrafen,

[2] https://github.com/bmake/modcat-prototyp/wiki

betrifft Anforderung (iii) die Veranstaltungssicht. Änderungen an der Grundstruktur sind hierfür nicht erforderlich. Die notwendigen Ergänzungen beschränken sich auf quantitative Attribute sowie Relationen zu anderen Ressourcen (Räume, Personen). In einem aktuellen Konzeptworkshop wurde die Relevanz dieser Anforderung nochmals bestätigt.

In Summe zeigte sich das Graph-Schema als stabil in seiner Grundstruktur und mit vertretbarem Aufwand erweiterbar. Konkret wurden folgende Ergebnisse erzielt: Die Datenbasis umfasst jetzt sieben vollständige Modulkataloge aus zwei von drei Fachbereichen. Lernziele können nach Kompetenzkategorien differenziert und bei fachlichen Kompetenzen einer Taxonomiestufe nach Bloom zugeordnet werden. Die Datenbasis kann für weitere Planungstools des Studiengangmanagements, wie Lehrkapazitätsplanung und Veranstaltungsplanung, im Verbund genutzt werden.

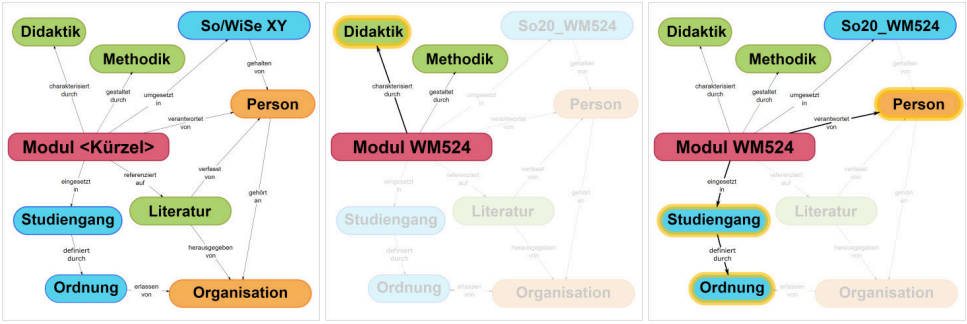
4.2 Implementierung der Graph-basierten Navigation

Die Fachanwendung selbst stellt eine komplette Neuentwicklung dar. Bisherige Vorstufen (vgl. [MHP19]) wurden mit Hilfe des Site-Generators Jekyll-RDF als rein statische Seiten implementiert. Da den Nutzer*innen ein unmittelbares Eingabefeedback und zugleich optimale Orientierung und Eingabesicherheit im Kontext komplexer Daten bereitgestellt werden sollten, fiel die Wahl auf folgende Front-End-Technologien: (i) die auf XML basierende Spezifikation des W3C zur Beschreibung zweidimensionaler Vektorgrafiken SVG in Verbindung mit D3.js, einer JavaScript-Bibliothek zur Erstellung dynamischer, interaktiver Datenvisualisierungen in Webbrowsern sowie (ii) das clientseitige JavaScript-Framework Vuetify mit vorgefertigten Oberflächenkomponenten[3].

Beim Entwurf des Navigationsgraphen war zunächst zu klären, welche Knoten und Kanten auszuwählen sind, um einen Kompromiss zwischen Orientierung in der Wissensdomäne und Übersichtlichkeit zu finden. Im Ergebnis wurden neun Knoten und zwölf Kanten ausgewählt (s. Abb.1a). Zwei der Knoten (Didaktik und Methodik) gruppieren mehrere Entitätstypen, sodass der Navigationsgraph zwölf Entitätsklassen und 15 Relationstypen repräsentiert. Bei der Farbcodierung wurden die folgenden Entscheidungen getroffen: der zentrale Knoten (Modul) hebt sich durch rote Färbung ab. Entitätsdaten, die von Modulverantwortlichen editiert werden können, sind grün eingefärbt. Dazu gehören Didaktik, Methodik und Literatur. Daten, die nur von Studiengangleitungen editiert werden können, sind blau eingefärbt. Für die Orientierung notwendige, in dieser Anwendung jedoch nicht editierbare Entitätstypen (Knoten) werden orange dargestellt.

Das Grundmodell für den Navigationsgraphen wurde mit dem Open-Source-Tool Inkscape erstellt, das ein hochwertiges SVG-Modell erzeugt und zudem spezifische Anpassungen über einen integrierten XML-Editor erlaubt. So sind alle Graph-Elemente mit sprechenden IDs versehen, Texte und grafische Objekte sind als solche spezifiziert. In Abhängigkeit von den Rechten und Interaktionen der Nutzer*innen weisen die Graph-Elemente orientierungsfördernde

[3] https://github.com/AKSW/jekyll-rdf, https://www.w3.org/XML/, https://d3js.org/, https://vuetifyjs.com/



(a) Basis Sicht (b) Modul-Verantwortlicher (c) Studiengang-Leitung

Abb. 1: Sichten des Navigationsgraphen

Effekte auf: Ghosting, Highlighting, Schatten (vgl. Abb.1b und Abb.1c). Diese Effekte werden durch den Einsatz der JavaScript-Bibliothek D3.js unterstützt. Der Navigationsgraph ist als eigenständige Vue-Komponente implementiert (`SVGGraph.vue`) und interagiert mit der übergeordneten Komponente `Starter.vue` und der benachbarten `FormBasisDaten.vue`. Die Auswahl des zu editierenden Moduls wird durch die Komponente `Select.vue` realisiert. `MainFooter.vue` beinhaltet Copyright-Informationen. Abb.2 zeigt das User Interface im Ganzen. Alle Ressourcen sind ebenfalls in GitHub[4] zugänglich.

Im Ergebnis bietet der Prototyp den Nutzer*innen für jedes zu editierende Modul eine auf ihre Bearbeitungsrechte angepasste Sicht auf die Graph-Navigation, die den Editierprozess konsistent unterstützt. Vue.js-Funktionen sorgen dafür, dass alle Eingaben während einer Sitzung auf dem Client zwischengespeichert werden. Der Bearbeitungsstand kann jederzeit in die Datenbank gespeichert werden. Das Rollen-Rechte-Konzept ist vorerst nur simuliert, einen Freigabeworkflow gibt es noch nicht.

5 Evaluation

Zur Ermittlung der Akzeptanz der neuen Fachanwendung für Digitale Modulkataloge durch die Nutzer*innen – in diesem Fall Lehrende verschiedener Studiengänge im Fachbereich Wirtschaft der Technischen Hochschule Brandenburg – wurde ein Softwaretest erarbeitet, durchgeführt und ausgewertet. Der Prototyp befindet sich in einer frühen Entwicklungsphase, daher hat die Evaluation durch Nutzer*innen einen großen Stellenwert für die Entwicklung der Fachanwendung im Hinblick auf User Interface (UI) Designs, Funktionalitäten und dahinter liegenden Technologien.

[4] https://github.com/bmake/modcat-prototyp

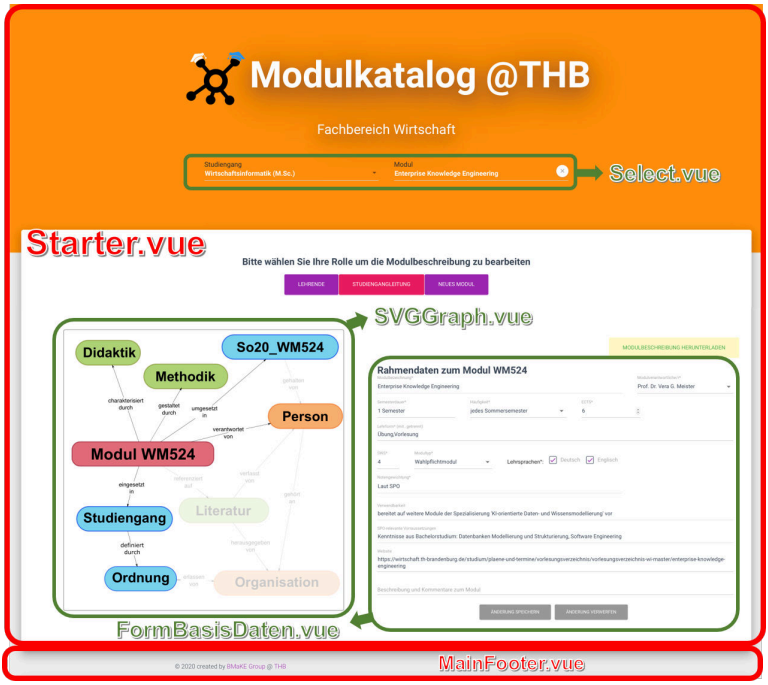


Abb. 2: User Interface mit Hervorhebungen zur Kennzeichnung der Vue-Komponenten

5.1 Untersuchungsdesign und -durchführung

Die Auswahl der Evaluationsmethoden [SB11] war von folgenden zielgebenden und begrenzenden Anforderungen geleitet: (i) Erkenntnisse zum Interaktionsfluss sammeln; (ii) neben positiven Aspekten auch potenzielle Nutzungsprobleme und Ideen zur Erweiterung aufdecken; (iii) den Tester*innen fachlich einschlägige Nutzungsszenarien anbieten; (iv) die Evaluation wegen der COVID-19-Pandemie remote durchführen. Ziel (i) erfordert eine Beobachtung des Nutzerverhaltens. Dabei sollen quantitative Daten zur Dauer von Interaktionen und zu Mausbewegungen gewonnen werden. Um diese Daten beurteilen zu können, sind zwei Teilnehmergruppen zu formieren: Noviz*innen und Expert*innen. Ziel (ii) kann mit der Methode des Lauten Denkens während der Beobachtung ergänzt um ein semistrukturiertes Interview (Online-Fragebogen) erreicht werden. Um Anforderung (iii) zu adressieren, wurden für jede/n Tester*in individuelle, fachlich einschlägige Use Cases entwickelt. Alle Beobachtungsdaten wurden von einer Desktop-Aufzeichnungssoftware gesammelt. Dieser Ansatz findet auch in Vor-Ort-Tests vielfache Anwendung. Aus Anforderung (iv) folgte die Notwendigkeit, den Teilnehmer*innen mediengestützte Anleitungen und eine Datenschutz-konforme Speicherumgebung bereitzustellen. Es wurde die frei nutzbare Software Active Presenter empfohlen und eine Bedienungsanleitung erstellt. Für den Upload der Aufzeichnungsvideos wurde die Lernplattform Moodle genutzt.

Es konnten 15 Lehrende des Fachbereichs Wirtschaft für die Teilnahme am Test gewonnen werden, welche die Gruppe der Noviz*innen bildeten. Als Expert*innen fungierten die fünf Mitglieder des Entwicklungsteams. Die Use Cases bildeten typische Aufgaben beim Editieren einer Modulbeschreibung ab: Lernergebnisse anpassen und spezifizieren, Inhaltselemente ändern, Lehr- und Lernmethoden strukturieren, weitere Daten ergänzen oder anpassen, wie z. B. Lehrsprachen, Prüfungsvoraussetzungen oder Kommentare. Dabei wurden keinen konkreten Änderungen vorgegeben, sondern auf die jeweilige Fachkompetenz der Tester*innen gesetzt, indem jede/r Editieraufgaben in einem ihr/ihm vertrauten Modul vornehmen sollte. Damit konnte der Test einer realen Nutzung der Software weitgehend angenähert werden. Die Teilnehmer*innen wurden Mitte April 2020 mit einer E-Mail zur Teilnahme aufgefordert. Mit der Mail wurden alle notwendigen Anweisungen, Links und Dokumente versandt. Wegen der Remote-Bedingungen musste der Testzeitraum über 4 Wochen ausgedehnt werden.

5.2 Auswertung der Untersuchungsergebnisse

Da die Erhebung der Rohdaten aus den übermittelten Aufzeichnungen und dem Fragebogen subjektiven Interpretationen unterliegt – das betrifft auch die quantitativen Daten – wurde sie durch zwei unabhängige Personen durchgeführt und im Anschluss konsolidiert. Die qualitativen Daten wurden einer thematischen Analyse nach [CBH15] unterzogen. Dafür wurden die Daten nach Themenbereichen gruppiert und nach konsistenten Beobachtungen (von mehr als 3 Teilnehmern bestätigt) und Besonderheiten strukturiert. Schließlich erfolgte eine Differenzierung in positive (Highlights) und negative (Probleme) Beobachtungen. Die aggregierten und konsolidierten Ergebnisse mit Fokus auf Forschungsfrage 3) zeigen Tab.1 (quantitative Daten) und Tab.2 (qualitative Daten).

Tab. 1: Quantitative Beobachtungsdaten zur Graph-Navigation bei Ausführung des Use Cases

Navigation zum Bereich (Knoten)	Zeit in sec.						Anzahl Klicks						Anzahl Scrollen					
	Experten			Novizen			Experten			Novizen			Experten			Novizen		
	$\overline{X}$	$\widetilde{X}$	σ	$\overline{X}$	$\widetilde{X}$	σ	$\overline{X}$	$\widetilde{X}$	σ	$\overline{X}$	$\widetilde{X}$	σ	$\overline{X}$	$\widetilde{X}$	σ	$\overline{X}$	$\widetilde{X}$	σ
Didaktik	4	5	2	31	13	40	1,0	1,0	0,0	2,9	1,0	6,3	0,0	0,0	0,0	2,6	1,0	3,4
Methodik	2	2	1	10	7	10	1,0	1,0	0,0	1,4	1,0	1,4	0,2	0,0	0,4	1,8	1,0	2,4
Rahmendaten	3	3	1	16	8	15	1,0	1,0	0,0	2,4	1,0	3,0	0,0	0,0	0,0	1,7	2,0	1,1

Die deutlichen Unterschiede zwischen der Bearbeitungszeit sowie der Anzahl von Klicks und Scrollen zwischen Noviz*innen und Expert*innen in Tab.1 weisen darauf hin, dass eine Graph-Navigation für die erste Gruppe eher ungewohnt ist. Auffällig ist, dass bei fast allen Items die Medianwerte der Noviz*innen sich wesentlich weniger von denen der Expert*innen unterscheiden als die Mittelwerte. Das lässt darauf schließen, dass nur wenige der Noviz*innen für diese Differenzen verantwortlich sind. Die Daten in Tab.2 machen deutlich, dass während der Graph-Navigation im Prototypen mehr Probleme als Highlights wahrgenommen wurden. Die ersten vier Probleme in Bereich konsistente Beobachtung lassen wiederum auf mangelnde Gewohnheit im Umgang schließen. Es werden Vorschläge in Richtung einer klassischen hierarchischen Navigation unterbreitet. Die anderen problematischen Beobachtungen stellen Verbesserungsvorschläge im Rahmen der Graph-Navigation dar. Einige davon wurden bereits

Tab. 2: Qualitative Beobachtungsdaten mit Fokus auf das Thema Graph-Navigation

Konsistente Beobachtung	Besonderheiten
Highlights	
-	- hilfreich, übersichtlich, optisch attraktiv - passende Anordnung
Probleme	
- nicht als Navigation erkannt - Abfolge unklar - zu viele Informationen - Drop-Down-Menüs verwenden - Zoomfunktion dysfunktional - Mouse-Over-Effekte nutzen	 - irreführendes Farbschema - maximal 3 Farben, mehr Kontrast

implementiert. In Vorbereitung der Evaluation gab es keinerlei Einführung oder Hinweise zum Thema Graph-Navigation. Es muss also konstatiert werden, dass auch in dieser Situation die Graph-Navigation nicht vollständig abgelehnt wurde. Allerdings sollten bei Einführung des Tools in den Regelbetrieb begleitende Einführungen und Tipps bereitgestellt werden. Um die

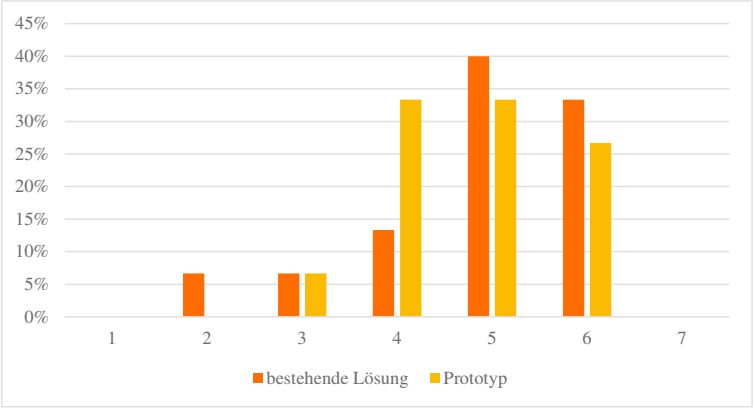


Abb. 3: Beurteilung der Zufriedenheit mit der bestehenden Lösung und dem getesteten Prototyen, N=15, 1 = sehr unzufrieden, 7 = außerordentlich zufrieden

Frage nach der Auswirkung der Graph-Navigation auf die Nutzerakzeptanz umfassend zu beantworten, sollen abschließend zwei strukturierte Items aus dem Online-Interview dargestellt werden. Wie Abb.3 zeigt, ist die Mehrheit der 15 Tester*innen zwar mit der bestehenden Lösung zur Bearbeitung von Modulkatalogen zufrieden, allerdings betrachteten sie zugleich den Prototypen als leistungsfähige Alternative. Die Bewertung sollte jeweils auf einer siebenstufigen Skala angegeben werden. Zusammengefasst kann somit festgestellt werden, dass eine Graph-Navigation als robuste Alternative zu herkömmlichen Navigationsmustern anzusehen ist.

6 Fazit und Ausblick

Aufbauend auf den Vorarbeiten [MB18; MHP19] stellt der hier präsentierte Prototyp einen wichtigen Schritt zu einer integrierten Fachanwendung für Digitale Modulkataloge dar. Die Graph-basierte Datenhaltung hat ihre Flexibilität und Erweiterbarkeit entlang anspruchsvoller Anforderungen bewiesen. Es ist gelungen, eine adaptive Graph-Navigation mit modernen Web-Technologien umzusetzen, die von den Tester*innen des Prototyps akzeptiert wurde. Im Rahmen der Evaluation sowie weiterer Meetings mit Stakeholdern wurden Anforderungen zu einem aufbauenden Entwicklungskonzept zusammengetragen. Neben der noch zu erweiternden Editierfunktion sollen das Browsen und die Dokumentation individueller Modulkataloge als Basisfunktionen implementiert werden. Für die Verknüpfung mit weiteren Fachanwendungen werden alternative Ansätze diskutiert. Eine leichtgewichtige Lösung könnte die Konfiguration einer REST API sein, auf die andere Anwendungen Zugriff haben. Aufwändiger und anspruchsvoller wäre eine Data-Hub-Lösung auf Basis hochwertiger Datenbanktechnologie. Die Kooperation mit anderen Hochschulen auf Open-Source-Basis wird angestrebt.

Literatur

[Al19] Alobaid, A.; Garijo, D.; Poveda-Villalón, M.; Santana-Pérez, I.; Fernández-Izquierdo, A.; Corcho, Ó.: Automating ontology engineering support activities with OnToology. J. Web Semant. 57/100472, 2019.

[ARM16] Arndt, N.; Radtke, N.; Martin, M.: Distributed Collaboration on RDF Datasets Using Git: Towards the Quit Store. In: SEMANTiCS 12th. S. 25–32, 2016.

[Br19] Brockmann, H.-C.: Wie Fair Data über den Erfolg der Digitalen Transformation entscheidet, 2019, URL: it-daily.net, Stand: 27. 11. 2019.

[Ca20] Camunda, 2020, URL: camunda.com/products, Stand: 01. 09. 2020.

[CBH15] Clarke, V.; Braun, V.; Hayfield, N.: Thematic Analysis. In: Qualitative Psychology: A Practical Guide to Research Methods. Sage, 2015.

[Ce19] Cerans, K.; Ovcinnikova, J.; Liepins, R.; Grasmanis, M.: Extensible Visualizations of Ontologies in OWLGrEd. In. Bd. 11762. LNCS, S. 191–196, 2019.

[Cm20] CmapTools: Institute for Human & Machine Cognition, 2020, URL: cmap.ihmc.us, Stand: 01. 09. 2020.

[Da20] Data.world, 2020, URL: gra.fo, Stand: 01. 09. 2020.

[Do20] DotNetRDF, 2020, URL: dotnetrdf.org, Stand: 01. 09. 2020.

[FAM16] Frischmuth, P.; Arndt, N.; Martin, M.: OntoWiki 1.0: 10 Years of Development – What's New in OntoWiki. In: SEMANTiCS 12th. Bd. 1695. CEUR, 2016.

[Ga05] Gangemi, A.: Ontology Design Patterns for Semantic Web Content. In: ISWC 5th. Bd. 3729. LNCS, S. 262–276, 2005.

[Hi16] Hitzler, P.; Gangemi, A.; Janowicz, K.; Krisnadhi, A.; Presutti, V.: Ontology Engineering with Ontology Design Patterns – Foundations and Applications. IOS Press, 2016.

[Kr19] Krijn, M.: Responsives Webdesign: 9 Do's and Don'ts bei der Navigation, 2019, URL: blog.amplexor.com, Stand: 15. 11. 2019.

[Kr20] Krupa, K.: Data Hub Guide for Architects. Marklogic, 2020.

[Ku10] Kultusministerkonferenz: Ländergemeinsame Strukturvorgaben für die Akkreditierung von Bachelor und Masterstudiengängen – Beschluss i. d. F. vom 04.02.2010, 2010.

[Lo16] Lohmann, S.; Negru, S.; Haag, F.; Ertl, T.: Visualizing ontologies with VOWL. Semantic Web 7/4, S. 399–419, 2016.

[MB18] Meister, V. G.; Becker, J.: Konzept und vergleichende Analyse eines Wissensgraph-basierten Modulkatalogs. In: LAiW co-located zur INFORMATIK. S. 14–28, 2018.

[MHP19] Meister, V. G.; Hu, W.; Pottenstein, P.: Wissensgraph-basierter Modulkatalog als Schnittstelle zwischen digitaler Lehre und digitalem Campusmanagement. In: Hochschulen in Zeiten der Digitalisierung. Springer Vieweg, S. 89–105, 2019.

[Mu15] Musen, M. A.: The protégé project: a look back and a look forward. AI Matters 1/4, S. 4–12, 2015.

[OD20] ODP: Semantic Web Portal to Ontology Design Patterns (ODPs), 2020, URL: ontologydesignpatterns.org, Stand: 01. 09. 2020.

[Op20] OpenRefine, 2020, URL: openrefine.org, Stand: 01. 09. 2020.

[Po20a] Po, L.; Bikakis, N.; Desimoni, F.; Papastefanatos, G.: Linked Data Visualization: Techniques, Tools, and Big Data. Morgan & Claypool, 2020.

[Po20b] Poolparty, 2020, URL: poolparty.biz, Stand: 01. 09. 2020.

[Sa16] Sarrafzadeh, B.; Vtyurina, A.; Lank, E.; Vechtomova, O.: Knowledge Graphs versus Hierarchies: An Analysis of User Behaviours and Perspectives in Information Seeking. In: CHIIR. S. 91–100, 2016.

[SB11] Sarodnick, F.; Brau, H.: Methoden der Usability Evaluation – Wissenschaftliche Grundlagen und praktische Anwendungen. Hans Huber, Bern, 2011.

[Su12] Suárez-Figueroa, M. C.; Gómez-Pérez, A.; Motta, E.; Gangemi, A.: Ontology Engineering in a Networked World. Springer, 2012.

[To20] TopQuadrant, 2020, URL: topquadrant.com, Stand: 01. 09. 2020.

[Tu20] Tudorache, T.: Ontology engineering: Current state, challenges, and future directions. Semantic Web 11/1, S. 125–138, 2020.

[Vo09] Vora, P.: Web Application Design Patterns. Morgan Kaufmann, 2009.

[Wa13] Ware, C.: Information Visualization – Perception for Design. Morgan Kaufmann, 2013.