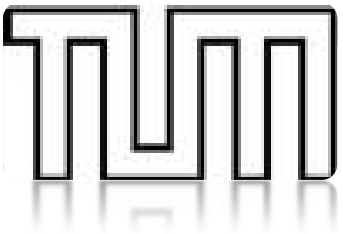


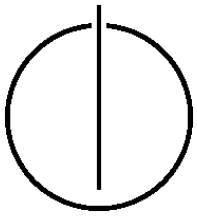
Fakultät für Informatik

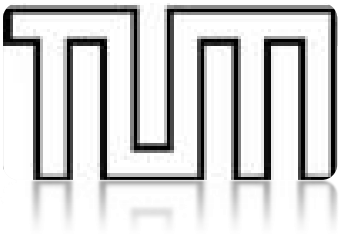
Der Technischen Universität München

Bachelorarbeit in Informatik

Design und Implementierung eines Backends für ein Online-Werkzeug zur Erhebung von Concept Maps

Manuel Schmidt





Fakultät für Informatik

Der Technischen Universität München

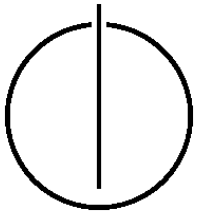
Bachelorarbeit in Informatik

Design und Implementierung eines Backends für ein Online-Werkzeug zur Erhebung von Concept Maps

Design and implementation of a backend for an online tool for surveying concept maps

Autor:	Manuel Schmidt
Aufgabensteller:	Prof. Dr. Peter Hubwieser
Betreuer:	Andreas Mühling

Abgabedatum:	16.01.2012



Erklärung

Ich versichere, dass ich diese Bachelorarbeit selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

In der Arbeit wird ausschließlich die männliche Form wie zum Beispiel „Probant“ genutzt. Diese soll keinen Rückschluss auf das Tatsächliche Geschlecht der genannten Person zulassen und keinesfalls das weibliche Geschlecht diskiminieren.

Ort: Garching b. München
Datum: 14. Januar, 2012 . . . . . .
Manuel Schmidt

Abstract

Concept Maps sind ein Werkzeug, welches sich neben der Einleitung von Unterrichtsstunden oder der Nutzung als Lernstrategie unter anderem zur Evaluierung eignet. Auch wenn sie nicht als ausschließliches Prüfungswerkzeug geeignet sind, stellen Concept Maps doch einen Mittelweg im Spannungsfeld von Objektivität und Reliabilität gegenüber der Validität dar. Aus diesem Grund wurde am im Fachgebiet Didaktik der Informatik an der Technischen Universität München ein Online-Werkzeug namens „CoMapEd“ entwickelt. Dieses ermöglicht es Nutzern eine Concept Map zu einem vorgegebenen Themengebiet zu zeichnen, die im Anschluss für eine Evaluierung online gespeichert wird.

Das manuelle Auswerten dieser Daten ist ein sehr zeitintensiver Prozess. Aus diesem Grund sollte die Evaluierung automatisiert werden. Hierzu wurde „Backend for CoMapEd“ entwickelt. Zur Entwicklung des Backends wurden verschieden Software Engineering Prozesse angewandt. Das Programm „Backend for CoMapEd“ ermöglicht es einem Benutzer die Datensätze der Datenbank zu verwalten und außerdem die Concept Maps der Probanden nach vorgegebenen Bewertungsmethoden evaluieren zu lassen und zu exportieren.

Inhaltsverzeichnis

1. Einleitung

„Bildungsstudie: Lehrer benoten privilegierte Schüler besser“ [Deutschlandradio Kultur –14.12.2011] – dieser oder ähnliche Artikel wurden in den letzten Jahren immer häufiger in den Medien thematisiert. Auch die Studie „Kevin - Studie“ vom September 2009 die im Rahmen einer Masterarbeit an der Universität Oldenburg angefertigt wurde, zeigt dass die Bewertung der Schüler durch den Lehrer nicht nur von deren Leistungen abhängt. So kann der Name, sozialer Hintergrund oder der Charakter in seine Note einfließen.
Beispielsweise kann es auch vorkommen dass ein Schüler eine bessere Note bekommt, wenn er es schafft, seine fachlichen Schwächen mit einer guten Orthografie in wohl formulierten Sätzen zu verpacken gegenüber einem anderen Schüler, der sich mich Ausdruck und Rechtschreibung schwerer tut.
Aus diesem Grund wird stetig nach objektiven Bewertungsmodellen zur Überprüfung des Wissenstandes von Personen in Prüfungssituationen gesucht. Dazu existieren zahlreiche Untersuchungen. Dabei wird nach(McClure, et al., 1998) versucht, eine reliable, objektive und valide Methode zu finden. Allerdings zeigt sich, dass diese zwei Eigenschaften, Reliabilität und Validität, im Gegensatz zueinander stehen. Auf der einen Seite möchte man den Einfluss äußerer Umstände, wie zum Beispiel die körperliche und geistige Verfassung des Probanden (Reliabilität), ausschließen und die Bewertung objektiv halten, also die subjektive Meinung des Korrektors nicht mit einzubeziehen. Auf der anderen Seite möchte man allerdings die Probanden nicht zu stark an einen gewissen Kontext binden, da auf diese Weise die Antworten wichtige Unterschiede im Wissen der Studenten maskieren könnte.(McClure, et al., 1998)
Je mehr Freiraum man den Probanden in der Prüfungssituation lassen möchte, desto eher kommen Prüfungsmethoden wie Aufsätze, Berichte und Präsentationen in Frage. Diese lassen sich allerdings nicht gut objektiv bewerten, denn es spielen bei der Bewertung subjektive Faktoren des Prüflings und des Prüfers eine Rolle. So kann der Proband beispielsweise den subjektiven Eindruck des Korrektors durch seine gute Vortrags- und Ausdrucksweise beeinflussen und die mentale gute oder schlechte Stimmung des Prüfers könnte unbewusst erheblichen Einfluss auf eine Bewertung haben.
Auch die Reliabilität ist nicht gegeben, da ein Schüler beispielsweise in zwei Prüfungen zum gleichen Thema zwei nicht identische Aufsätze verfassen würde. Versucht man dagegen mit Multiple-Choice-Tests oder kurzen offenen Tests die Probanden zu prüfen, so steigt einerseits die Objektivität, andererseits sind die Antworten stark an den Kontext gebunden. Dies könnte starke individuelle Unterschiede im Wissen der Prüflinge überdecken.(McClure, et al., 1998) So kommt es gerade bei dieser Art der Wissensüberprüfung stark darauf an, wie gut die Fragen den gesamten geprüften Themenbereich abfragen. Es ist also die Validität beeinflusst.

So ist die Grundlage für diese Arbeit die Idee aus(McClure, et al., 1998), die einen Mittelweg darstellt. Sogenannte Concept Maps bieten ein Gleichgewicht zwischen einer objektiven, reliablen und einer validen Bewertung.

Dazu ist im Fachgebiet Didaktik an der TU München bereits ein Tool in der Entwicklung, welches auf diese Art und Weise das Wissen der Studenten abfragen soll. Der Name des Tools lautet „CoMapEd“ und steht für „Concept Maps in

Education“. Es gibt bereits eine erste Version, in der die grundlegenden Funktionen implementiert sind. Weitere Funktionen sollen dann im Laufe der nächsten Zeit dazu entwickelt werden. Dieses Programm wird Abschnitt 2.1.2.2 erklärt.
Die Aufgabe dieser Bachelorarbeit ist es zu „CoMapEd“ ein Backend zu entwickeln, welches es zum einen ermöglicht, die durch das Programm gesammelten Daten zu verwalten und zum anderen die gewonnen Informationen auszuwerten. So soll es dem Benutzer des Backends zum Beispiel möglich sein „Musterlösungen“ zur Bewertung der Maps der Probanden einzufügen.

Infolge dessen umfasst diese Arbeit die allgemeine Beschreibung des theoretischen Bereichs von Concept Maps und die Nutzung von Concept Maps als Evaluierungswerkzeug. In diesem Kontext wird ebenfalls auf den theoretischen Hintergrund von Graphen eingegangen. Als letzten theoretischen Punkt wird das Software Engineering behandelt. Von der Anforderungsermittlung bis hin zum abschließenden Testen wird sich der praktische Teil mit der Entwicklung des Backends auseinandersetzen. Schließlich wird das Produkt diskutiert und es werden Empfehlungen zur Weiterentwicklung gegeben.

2. Theorie

2.1. Concept Maps

2.1.1. Was sind Concept Maps

Erstmals tauchten Concept Maps am Ende der 60er und Anfang der 70er Jahre auf, als ein 12 Jahre dauerndes Experiment von Joseph D. Novak mit dem Ziel, das Verständnis von Kindern über naturwissenschaftliche Konzepte zu untersuchen.(McClure, et al., 1998) Laut (Novak, et al., 2008) bildete allerdings David Ausubel mit seiner Arbeit über Lernpsychologie (Ausubel, 1963) die Grundlage zu der Forschungsarbeit von Novak.
In Novaks Experiment wurde eine Befragung von einer großen Zahl von Kindern durchgeführt, die es im Anschluss auszuwerten galt. Allerdings empfanden die Forscher es als schwierig, die speziellen Änderungen des Verständnisses zu wissenschaftlichen Konzepten anhand der Befragungsprotokolle herauszufiltern. Aus dieser Not heraus entstand die Idee, das Wissen der Kinder durch Concept Maps zu repräsentieren. Concept Maps wurden allerdings im Folgenden auch außerhalb der Forschung eingesetzt.
Als Beispiel hierfür sei (Surber, et al., 1981) genannt, welches Concept Maps als Repräsentation für Missverständnisse - im Originaltext als „misunderstanding“(Surber, et al., 1981) – von Studenten bezeichnet.

„Concept Maps sind ein graphisches Werkzeug zur Organisation und Repräsentation von Wissen.“(Novak, et al., 2008) Es gibt Konzepte und Beziehungen zwischen diesen. Bei einer graphischen Darstellung werden verschiedene Konzepte durch Knoten repräsentiert. Beziehungen zwischen den Konzepten werden durch Verbindungslinien dargestellt. (vgl. Kapitel 2.2) Diese Beziehungen werden typischerweise benannt, wobei die Benennung auf die Verbindungslinie geschrieben wird. Nach (Novak, et al., 2008) haben Concept Maps weitere Charakteristika:

- Hierarchische Präsentation der Konzepte:
  - Das allgemeinste Konzept wird ganz oben positioniert und weniger allgemeine Konzepte folgen hierarchisch in absteigender Reihenfolge. Dabei hängt die Struktur vom Kontext ab. Aus diesem Grund sollten Concept Maps nach bestimmten Fragen konstruiert sein, die man gerne beantwortet haben möchte. Diese Fragen nennt man „focus question“. Der Hintergrund („context“) einer Concept Map wird dann durch die Situation oder das Event, welches man unter Umständen auf diese Art und Weise zu verstehen versucht, gebildet.
- Erlauben von Querverweisen („cross-links“):
  - Querverweise sind Verbindungen zwischen Konzepten aus unterschiedlichen Segmenten oder Bereichen der Concept Map. Dieses hilft dem Leser beim Verstehen der Beziehung zwischen den Konzepten unterschiedlicher Bereiche. Außerdem präsentieren solche Querverweise oft kreative Sprünge bei der Erzeugung neuen Wissens. Denn es gehört zur Förderung des kreativen Denkens, solche Querverweise zu suchen und zu charakterisieren.

Ein letztes Feature, welches Concept Maps haben können, ist ein spezielles Beispiel von Objekten oder Ereignissen, welches hilft, die eindeutige Bedeutung eines

Konzepts zu verstehen. Diese werden dann nach (Novak, et al., 2008) normalerweise nicht durch Knoten repräsentiert, da sie selbst keine Konzepte sind.

Es gibt allerdings auch andere Auffassungen, nach denen Concept Maps diese Charakteristika nicht besitzen müssen. Ein Beispiel hierfür wäre (Steiner, et al., 2008), in der Concept Maps lediglich aus einer Menge endlicher, nicht leerer Konzepte und einer Menge von endlicher, nicht leerer, geordneter Paare von Konzepten bestehen. Hierbei wird nicht verlangt, dass diese eine hierarchische Ordnung haben und dem zur Folge sind auch keine Querverweise notwendig.

In (Novak, et al., 2008) wird außerdem erläutert, dass Lernen nach Konzepten stattfindet. Das heißt, gelerntes wird im Gehirn nach Konzepten, also nach Themengebieten, geordnet. So kann man im Kurzzeitgedächtnis zum Beispiel nicht mit mehr als mit zwei bis drei Konzepten gleichzeitig arbeiten. Probanden, denen man eine Liste von 10 bis 12 Buchstaben oder Nummern vorlegt, sind meist nur in der Lage, fünf bis neun Items im Nachhinein abzurufen. Falls diese Buchstaben oder Nummern aber zu einem bekannten Wort oder einer Telefonnummer oder ähnlichem gruppiert werden können - also in Konzepte eingeordnet werden - sind die meisten in der Lage, 10 oder mehr abzurufen.(Novak, et al., 2008)

Von Probanden erstellte Concept Maps lassen sich nach „Struktur-Typen“ unterscheiden. Je nachdem was die Probanden in den Konzepten erkennen, bauen sie zum Beispiel lineare, baumartige oder kreisförmige Maps.(Vanides, et al., 2005) Beispiele hierfür sind auf Abbildung 1 zu sehen.

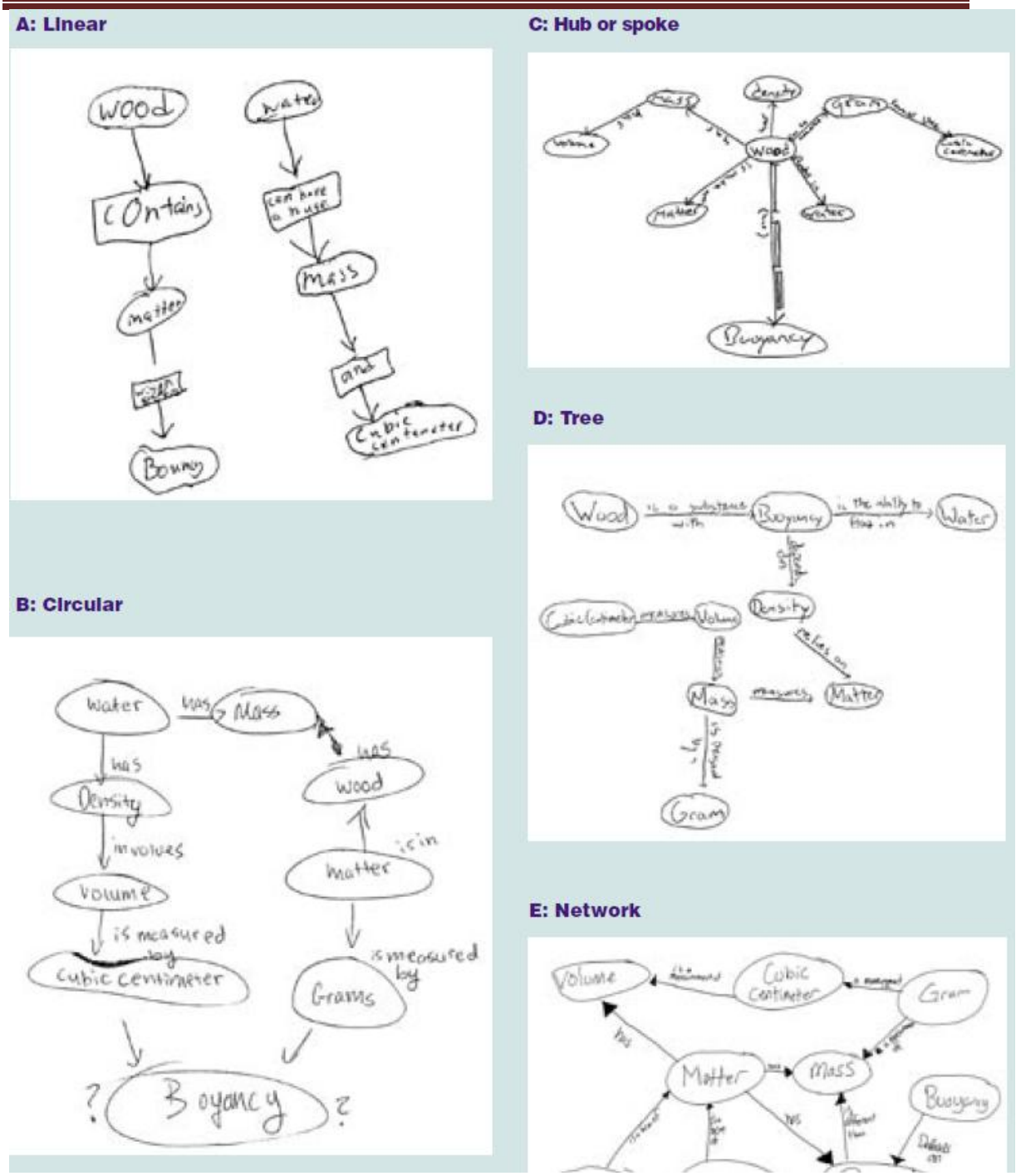


Abbildung 1: Struktur-Typen von Concept Maps(Vanides, et al., 2005)

2.1.2. Einsatzzwecke

2.1.2.1. Einsatzzwecke allgemein

Wie schon in 2.1.1 angedeutet, werden Concept Maps für unterschiedliche Zwecke herangezogen. Zum Beispiel als Lernstrategie, als Themeneinleitung, zur Planung von Unterrichtsstunden oder zum Messen des Wissens der Studenten. (McClure, et al., 1998) Zum Beispiel beschreibt (Novak, et al., 2008), dass Concept Maps ein sehr mächtiges Werkzeug zum Lernen sind.

Auch zum Überprüfen von Schülerwissen sind Concept Maps geeignet. So gäbe es diverse Vorteile, wenn man Concept Maps zur Benotung in Schulen benutzen würde. Zum Beispiel ist es ein natürlicher Weg, um ein Arbeitsgebiet auszudrücken und zu präsentieren.(Steiner, et al., 2008) Außerdem sind die benötigten Darstellungsfähigkeiten, die von Schülern verlangt werden, im Vergleich zu anderen Beurteilungsmethoden relativ gering.(McClure, et al., 1998) Allerdings müssen die Vorteile, die Concept Maps als Benotung mit sich bringen, auch die anfallenden Schwierigkeiten aufwiegen. Da das Lehren ein komplexes Unterfangen ist, bei dem die Lehrer am Ende mehrere Ziele erreichen wollen, müssen sie abwägen, inwieweit das Einführen einer neuen Technik die zu erreichenden Ziele beeinflusst. So wird zum Beispiel das Lehren von Concept Maps eine gewisse Zeit in Anspruch nehmen, in der die Lehrer ihren anderen Zielen nicht direkt näher kommen.(McClure, et al., 1998) Nach (McClure, et al., 1998) stellen Concept Maps Lehrer vor drei Probleme:

1) Es muss den Lehrern möglich sein, den Schülern Concept Maps zu lehren,(McClure, et al., 1998) denn die Schüler müssen ein gewisses Maß an Fertigkeiten mit Concept Maps entwickeln, um reliable Ergebnisse zu liefern. (Hubwieser, et al., 2011)
2) Lehrer müssen bedenken, wie viel Zeitaufwand es im Vergleich zu traditionellen Prüfungsaufgaben darstellt, Concept Maps zu entwerfen.(McClure, et al., 1998)
3) Zum Schluss müssen sich Lehrer im Klaren darüber sein, wie viel Zeit es benötigt, die Maps zu bewerten.(McClure, et al., 1998)

(Barenholz, et al.) und (Trowbridge, et al., 1990) bieten Beispiele dafür, dass Concept Maps auch zum Bewerten des Effekts von Anweisungen auf den Probanden genutzt werden können. Dagegen benutzt (Hegarty-Hazel, et al., 1991) Concept Maps, um die Beziehung zwischen konzeptuellem Verständnis und der Nützlichkeit von Lernstrategien bei naturwissenschaftlichen Studenten zu bewerten.

Eine ganz andere Möglichkeit zur Nutzung von Concept Maps wird von Tillman Weyde und Jens Wissmann in (Weyde, et al., 2004) beschrieben. Hier nutzen sie dynamische Concept Maps für Musik. So werden dem Nutzer zu einem gegeben Musikausschnitt relevante Konzepte gezeigt.(Weyde, et al., 2004) Zu erkennen ist dieses auf Abbildung 2

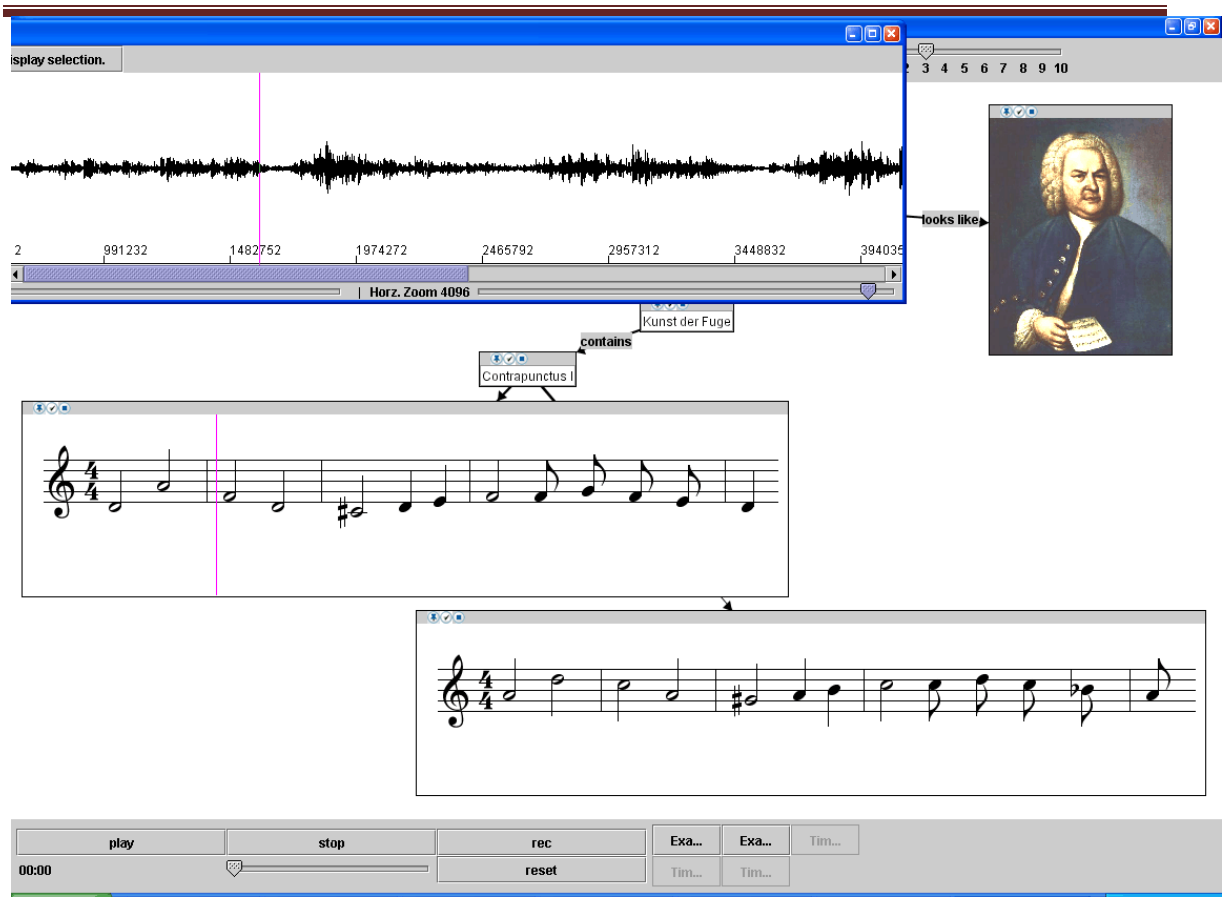


Abbildung 2: Concept Maps zur Nutzung mit Musik(Weyde, et al., 2004)

Dynamische Concept Maps sind solche, die sich im Laufe der Zeit verändern. Sie können zum Beispiel herangezogen werden, um die Funktion eines Geräts zu beschreiben, dessen Zustand und Aktionen sich zeitlich ändern.(Weyde, et al., 2004)

Da „CoMapEd“ allerdings ein Tool zur Evaluierung von Wissen darstellt, werden sich die folgenden Abschnitte ausschließlich mit Concept Maps zum Messen von Wissen beschäftigen.

2.1.2.2. Einsatzzwecke im Fachgebiet Didaktik der Informatik an der TUM / CoMapEd

Das Fachgebiet Didaktik der Informatik an der Technischen Universität München beschäftigt sich aktuell unter anderem mit der Validierung von Concept Maps. Es sind zum Beispiel (Hubwieser, et al., 2011) und (Mühling, 2011) zu nennen, die sich mit Concept Maps im Zusammenhang mit objektorientiertem Programmieren auseinandersetzen. Es wird mit Concept Maps untersucht, wie gut die Studenten das Prinzip der Objektorientierung (aus der Informatik) verstanden haben. Dabei stellte sich heraus, dass die Qualität einer Concept Map in direktem Zusammenhang mit der in der Klausur erreichten Note des Fachs „Einführung in die Informatik für Ingenieure“ steht. (Mühling, 2011)

Im Zuge dieser Untersuchungen wird am Lehrstuhl das Programm CoMapEd („Concept Maps in Education“) entwickelt. Dieses ist ein Tool, mit dem man Concept Maps zeichnen kann. Es bietet eine Menge von Konzepten an, die man von außerhalb der Zeichenfläche heranziehen und anschließend miteinander verbinden kann. Es ist außerdem möglich, die Verbindungen zu benennen. Auf diese Weise soll den Probanden die Freiheit gelassen werden, die Konzepte selbst frei im Raum zu justieren und zu verbinden. Es ist aber ebenfalls möglich, einen vorgegebenen Vorrat an Kanten zu benutzen, um die Validierung einfacher zu gestalten. Über einen Speicherbutton kann die Concept Map anschließend in einer Datenbank gespeichert werden.

Ein Entwicklungsgrund für dieses Tool sind die Vorteile, die ein solches computerbasiertes System gegenüber herkömmlichen Methoden bietet. Ohne dieses müssen die Probanden ihre Concept Maps auf Papier zeichnen. Das bietet nach (Ifenthaler, 2006) zum Beispiel folgende Nachteile: Das Verbinden von Konzepten kann nicht so leicht rückgängig gemacht werden. Noch viel schwieriger ist es allerdings, einmal falsch angeordnete Konzepte wieder in eine übersichtliche Lage zu bringen.(Ifenthaler, 2006; Novak, et al., 2008; Mandl, et al., 2000) Auch die „Archivierung“, „Weitergabe“ und „Publikation der Ergebnisse“(Ifenthaler, 2006) wird durch ein computerbasiertes System erleichtert. Für die Auswertung gilt, dass der Korrektor nicht alle Concept Maps einzeln betrachten und manuell auswerten muss.
Es gibt zahlreiche computerbasierte Systeme dieser Art, die sich „hinsichtlich ihrer experimentellen Realisierung“(Ifenthaler, 2006) unterscheiden.
So ist zum Beispiel „CmapTools“, welches am Institute for Human and Machine Cognition in Florida entwickelt wurde, darauf ausgelegt, die Stärken von Concept Maps, Technologie und Internet zusammenzuführen. (Novak, et al., 2008) Es ermöglicht dem Nutzer, Concept Maps zu konstruieren und zu editieren. Außerdem hat der Benutzer die Möglichkeit, die Concept Maps zu publizieren und somit für jeden zur Verfügung zu stellen. Zusätzlich ermöglicht es die Software, Quellen zu verlinken. (Fotos, Bilder, Graphen, Videos, Diagramme, Tabellen, Texte, Internetseiten oder andere Concept Maps)(Novak, et al., 2008)

Allerdings muss diese Software „CmapTool“ erst auf den Rechnern installiert werden. Dieses kann gerade bei großen Rechensystemen viel administrativen Aufwand

bedeuten. Somit wurde das „CoMapEd“ so entwickelt, dass es webbasiert ist. Der Benutzer benötigt hier lediglich einen Internetzugang und einen Browser, der von den meisten Betriebssystemen mitgeliefert wird. Bei Adressierung der richtigen Internetseite landet der Proband direkt auf der Seite, auf der er seine Concept Maps bauen kann. Dieses bietet gerade bei der Nutzung von Concept Maps als Validierungswerkzeug große Vorteile, da die gesammelten Daten so direkt zentral auf der Datenbank gespeichert werden. Es ermöglicht dem Nutzer außerdem, von verschiedenen Rechnern an seinen Concept Maps weiterzuarbeiten.
Ein weiterer Vorteil den „CoMapEd“ bietet war, dass man annimmt, dass ein Proband eher bereit ist eine Concept Map gewissenhaft zu zeichnen, wenn er eigenen Nutzen davon hat. Das heißt, dass Proband hinterher auch privat zur Wissensstrukturierung nutzen kann. Dazu muss er die Map natürlich auch noch nach dem Zeichnen laden oder exportieren können.

2.1.3. Validierung / Standard Maße

2.1.3.1. Allgemein

Beim Validieren von Concept Maps gibt es zwei Arten von Bewertungen, die „content validity“ oder „application validity“ genannt werden, wobei man sich zunächst für eine entscheiden muss. (Steiner, et al., 2008). Während sich „content validity“ damit auseinander setzt, ob Concept Maps das aktuelle Wissen der Probanden gültig darstellen (Albert, et al., 2005), beschäftigt sich „application validity“ mit der Frage, ob Concept Maps auch den Zweck erfüllen, für den sie gemacht wurden.(Steiner, et al., 2008)

Aus vorhandenen oder fehlenden Verbindungslinien einer Concept Map kann abgelesen werden, ob der Proband eine Beziehung zwischen den Konzepten sieht oder nicht. So kann das Verständnis der Materie überprüft werden.(McClure, et al., 1998) Allerdings muss man hier zwischen dem, was der Proband weiß und was er gezeichnet hat, unterscheiden. Concept Maps repräsentieren nicht direkt das Wissen des Studenten, sondern vielmehr das, was er bereit war zu präsentieren. (Hubwieser, et al., 2010) Diese Gedanken beruhen auf dem Konzept der Externalisierung. Es besagt, dass der Proband von seiner Motivation, Aufmerksamkeit, Zeit und vielen anderen Faktoren beeinflusst werden könnte.(Norman, 1983) Concept Maps repräsentieren folglich nur das Wissen eines Probanden, welches er in der gegebenen Zeit mit der vorhandenen Lust und Motivation bereit war zu zeigen. Auch die Prüfungsmethode kann hierauf Einfluss üben. Bei der Nutzung von Concept Maps muss es also das Ziel sein, diese Störfaktoren beim Probanden zu eleminieren und seine Aufmerksamkeit auf die Aufgabe zu fokussieren.

In (Novak, et al., 2008) wird die Verwendung von Concept Maps zur Evaluierung in der Schule als ein „Henne und Ei“ Problem beschrieben. Auf der einen Seite können Concept Maps nicht zur Bewertung herangezogen werden, solange die meisten Schüler nicht gelernt haben, es als Wissensrepräsentationswerkzeug zu benutzen. Auf der anderen Seite wäre die Bereitschaft von Lehrern und Schülern erst dann sehr groß, zu lernen, sie zu benutzen, falls Concept Maps eingeführt werden würden.(Novak, et al., 2008)

Nach (Ifenthaler, 2006) gibt es drei Varianten der Datenanalyse von Concept Maps, und zwar die graphische, die strukturelle und die Korrespondenzanalyse. In (McClure, et al., 1998) lassen sich dagegen noch andere Methoden finden, so dass davon ausgegangen werden kann, dass dieses nur ein Teil der möglichen Varianten ist. Während die graphische Analyse dem optischen Vergleich mehrerer Concept Maps dient, werden bei der strukturellen Analyse die Concept Maps als gerichtete und benannte Graphen aufgefasst. So können graphentheoretische Indizes, die sonst zur Beschreibung der Struktur von mathematischen Graphen dienen, sinnvoll für die Auswertung von Concept Maps eingesetzt werden. Die Korrespondenzanalyse vergleicht dagegen zwei Concept Maps miteinander und gibt anschließend Aufschluss über die Übereinstimmung zwischen den Graphen. (Stracke, 2003) Im Folgenden wird dieses in Abschnitt 2.1.3.2 näher erläutert. In (McClure, et al., 1998) werden allerdings noch zwei weitere Verfahren, das sogenannte „holistic scoring“ und das „relational scoring“, beschrieben. Auch unterscheidet sich das dort erläuterte

„structural scoring“ in der Berechnung von der strukturellen Analyse aus (Stracke, 2003) (siehe Kapitel 2.1.3.5). „Holistic scoring“ ist eine Bewertungsform, bei der die logische Form der Concept Map betrachtet und mit Punkten von 1 bis 10 bewertet wird. Hierbei gibt es kaum eine Bewertungsstruktur, so dass es die Aufgabe des Korrektors ist, die logischen Zusammenhänge der Concept Map zu erfassen, zu verstehen und anschließend zu bewerten. Dagegen werden beim „relational scoring“ die Relationen zwischen den Konzepten bewertet und daraus eine Punktzahl berechnet.(McClure, et al., 1998)

2.1.3.2. Bsp: Mastermap

Diese Art der Bewertung von Concept Maps beinhaltet das Abgleichen der Usermap mit einer sogenannten Mastermap. Dabei wird nach Gleichheiten und Unterschieden zwischen den beiden Maps gesucht. Diese Bewertung wird nach (Goldsmith, et al., 1989) durch Vergleichen der beiden Maps durchgeführt. Es ist also eine Art der Korrespondenzanalyse, mit dem speziellen Fall, dass die Concept Map einer Testperson (Usermap) mit der Concept Map eines Experten (Mastermap) verglichen wird. Dieses wird auch als interindividuelle Korrespondenzanalyse bezeichnet (Stracke, 2003).
Eine Mastermap kann zum Beispiel von einem Lehrer oder Professor erstellt werden, der sich in der Materie sehr gut auskennt.(McClure, et al., 1998) Sie kann auch aus mehreren Mastermaps generiert werden, die jeweils von Leuten mit sehr guten Kenntnissen der Materie erstellt wurden. (Albert, et al., 2005)
In (Hubwieser, et al., 2011) wird beschrieben, dass es auf lange Sicht ein Ziel sein sollte, solche Mastermaps automatisch nach einem festen Prinzip aus dem gesammelten und gespeicherten Material zur Erhöhung der Objektivität zu generieren. Natürlich ergibt sich daraus auch eine große Zeitersparnis.

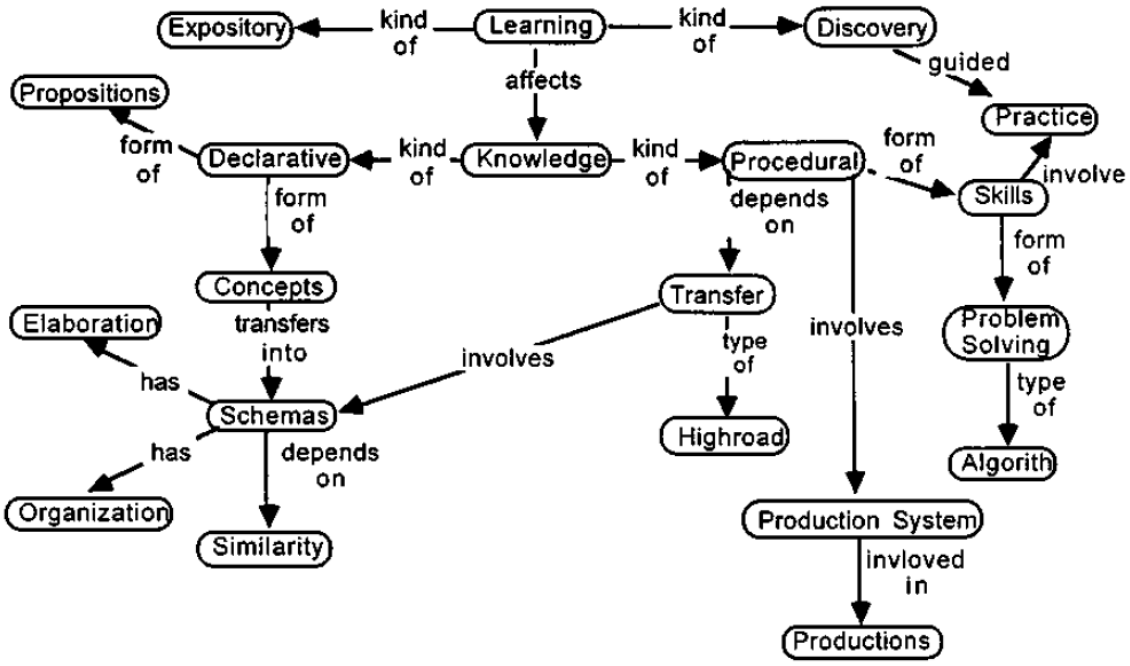


Abbildung 3: Concept Map - Beispiel(McClure, et al., 1998)

Laut (McClure, et al., 1998) findet die Bewertung (BW) nach folgender Formel statt:

Formel 1: Knoten eines Konzepts

$$K(c) = \{\forall\, x \in K \mid \{x, c\} \in E\}$$

Formel 2: Bewertungswert nach Mastermap

$$BW = \frac{1}{|K|} \sum_{c \in K} |K(c)_M \cap K(c)_U| \div |K(c)_M \cup K(c)_U|$$

Erläuterung: $K(c)$ *ist die Menge aller Konzepte die mit dem Konzept c benachbart sind.* $K(c)_M$ *ist die Menge* $K(c)$ *der Mastermap und* $K(c)_U$ *analog für die Usermap.*
Für zusätzliche Erläuterungen vgl. Graphentheorie in Kapitel 2.2.1.

Dieser Übereinstimmungswert oder auch Bewertungswert (BW) liegt zwischen 0 und 1.

Betrachtet man zum Beispiel in Abbildung 3 das Konzept „Learning“ und vergleicht dieses mit der Mastermap aus Abbildung 4, so ergibt sich:

$$K(Learing)_U = \{Expository, Discovery, Knowledge\}$$
$$\Rightarrow |K(Learning)_U| = 3$$
$$K(Learing)_M = \begin{Bmatrix} Expository, Discovery, Knowledge, \\ Transfer, Organization, Elobaration, Practice \end{Bmatrix}$$
$$\Rightarrow |K(Learning)_M| = 7$$

Somit ergibt sich für den Übereinstimmungswert BW:

$$BW = |\{Expository, Discovery, Knowledge\}| \div \left|\begin{Bmatrix} Expository, Discovery, Knowledge, \\ Tranfer, Organiszation, Elobaration, Practice \end{Bmatrix}\right| = \frac{3}{7} \approx 0{,}43$$

Wenn man dieses nun für alle Konzepte durchführt, kommt man laut (McClure, et al., 1998) auf einen Mittelwert von 0,3679.

Anmerkung: In (McClure, et al., 1998) *wurde hier übrigens ein Fehler begangen, da diese mit* $K(Learning)_M = 6$ *rechnen.*

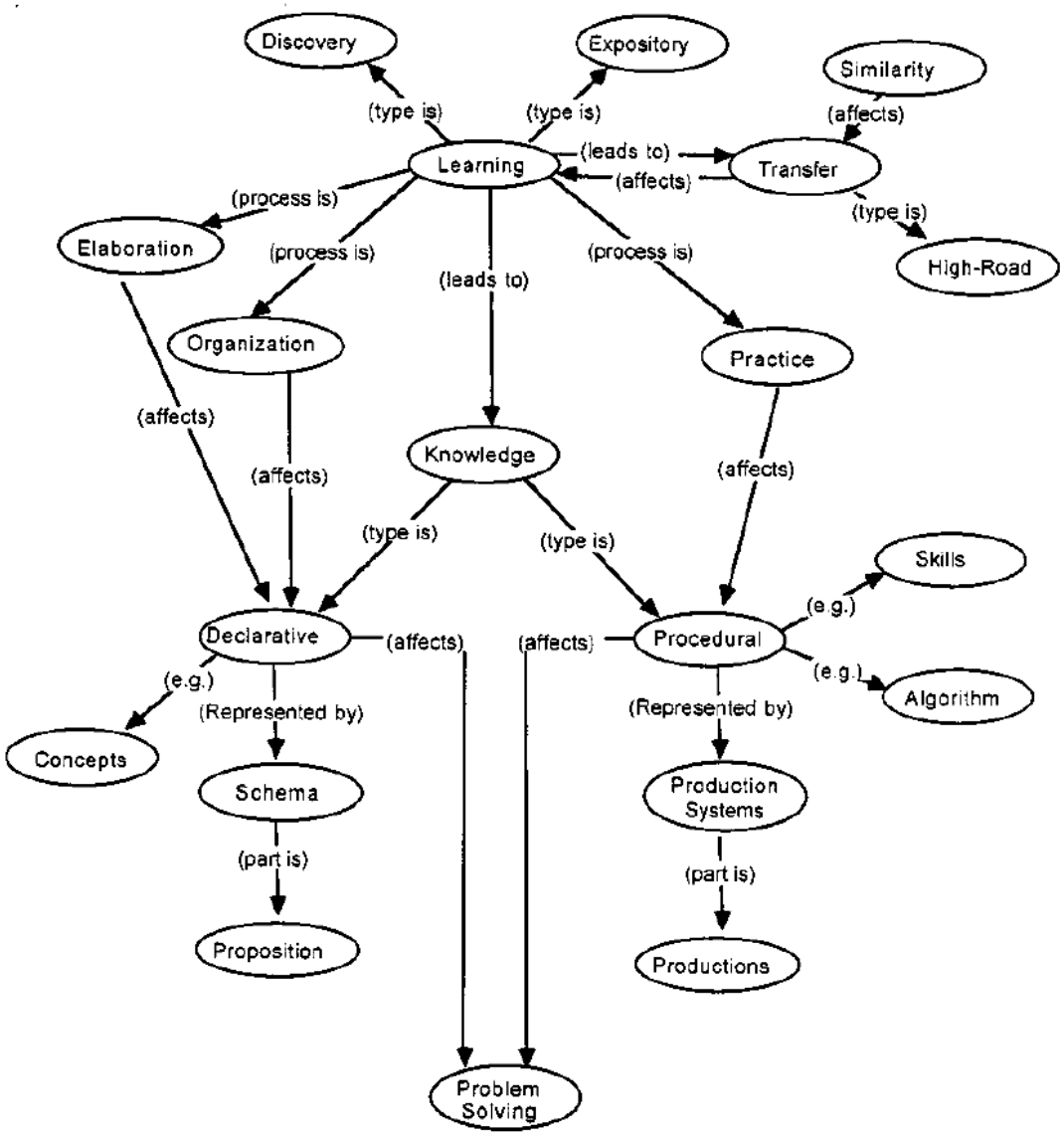


Abbildung 4: Concept Map - Mastermap(McClure, et al., 1998)

Eine etwas einfachere Berechnung wird von (Stracke, 2003) erläutert. Hierbei wird der Korrespondenzkoeffizient C nach folgender Formel berechnet:

Formel 3: Korrespondenzkoeffizient I

$$C = \left(\sum Übereinstimmungen - \sum Unterscheide \right) \div U_{max}$$

Formaler und etwas anders lässt sich auch schreiben:

Formel 4 Korrespondenzkoeffizient II

$$C = [(p_{00} + p_{11})(p_{01} + p_{10})] \div U_{max}$$

p_{00}: *Die Summe der korrespondierenden Begriffspaare, die in beiden Netzen nicht verknüpft sind.*

p_{11}: *Die Summe der korrespondierenden Begriffspaare, die in beiden Netzen verknüpft sind.*

p_{01}: *Die Summe der korrespondierenden Begriffspaare, die in Netz 1 nicht verknüpft sind, aber in Netz 2 verknüpft sind.*

p_{10}: *Die Summe der korrespondierenden Begriffspaare, die in Netz 2 nicht verknüpft sind, aber in Netz 1 verknüpft sind.*

Hierbei liegen die berechneten Werte zwischen -1 bis +1.(Stracke, 2003)

Diese Art der Bewertung, das Bewerten mithilfe einer Mastermap, funktioniert nur auf Grund der Tatsache, dass alle Maps die gleiche Menge an Konzepten benutzen. Dabei gilt es zu beachten, dass der berechnete Gleichheitswert in einer prognostizierbaren Art und Weise mit der Instruktion und traditionellen Messmethoden korrespondiert. (Goldsmith, et al., 1989) Dies lässt den Rückschluss zu, dass der auf diese Art und Weise berechnete Messwert ein gültiger Indikator für das Wissen der Probanden ist. (McClure, et al., 1998)

2.1.3.3. Bsp: Graphische Analyse

Die graphische Analyse kann als eine manuelle Analyse aufgefasst werden. Hierbei werden mehrere Graphen gleichzeitig angezeigt, die es so zu vergleichen und analysieren gilt. Um das Verfahren ein wenig zu vereinfachen, versucht man, die Konzepte oftmals an den gleichen Stellen auszurichten. Auf diese Weise sind fehlende oder überschüssige Kanten sehr schnell zu erkennen. Zusätzlich wird oftmals eine tabellarische Ansicht erzeugt, die die Concept Map, bzw. deren Relationen in einer so genannten Adjazenzmatrix präsentiert (vgl. Kap. 2.2.2). Hier ist zu erkennen welches Konzept mit wie vielen anderen Konzepten verbunden ist. So lässt sich auch schnell erkennen, welche Konzepte gar nicht integriert wurden.(Stracke, 2003)

2.1.3.4. Bsp: Relationale Bewertung

Dem „relational scoring“(Novak, et al., 2008) liegt die entwickelte Technik von (McClure, et al., 1990) zugrunde.

Bei der relationalen Bewertung wird den Kanten zwischen Konzepten, je nach Sinnhaftigkeit und Gehalt der Relation, Punkte von 0 bis 3 gegeben. Dieses wird für jede unterschiedliche Kante zwischen den Konzepten durchgeführt. Im Anschluss werden die Punkte über eine gesamte Map zusammenaddiert und ergeben auf diese Weise eine Gesamtpunktzahl. (Novak, et al., 2008)

Die Concept Map aus Abbildung 3 bekam von einem Korrektor nach (Novak, et al., 2008) zum Beispiel 17x drei Punkte und 4x zwei Punkte für seine Kanten. Somit ergab sich eine Gesamtpunktzahl von 59 Punkten.

2.1.3.5. Bsp: Strukturelle Analyse

Wie schon in 2.1.3.1 angedeutet, gibt es hierzu mindestens zwei Analyseverfahren von Concept Maps, die sich strukturelle Analyse bzw „structural scoring“ nennen. (Stracke, 2003) beruft sich auf (Eckert, 2000) und (Friege, et al., 2000), während das von (Novak, et al., 2008) entwickelte Verfahren auf (Novak, et al., 1984) beruht.

Bei (Stracke, 2003) werden folgende Indizes für die Auswertung der Maps herangezogen: Umfang, Verknüpfungsdichte und Zerklüftetheit.

- Umfang U: Als Umfang wird die Anzahl der Relationen innerhalb der Concept Map bezeichnet. Dabei gilt, dass ein großer Umfang für viele Verknüpfungen spricht und somit für „einen hohen Grad an Vernetzungen von Wissensinhalten“[16]. Der maximale Umfang U_{max}, bei dem jedes Konzept mit jedem anderen Konzept verknüpft ist, wird wie folgt berechnet:

Formel 5: Umfang

$$U_{max} = \frac{n \times (n-1)}{2}$$

- Alternativ zum Umfang lässt sich auch von der Verknüpfungsdichte V sprechen, welche sich durch $V = {}^{U}/_{U_{max}}$ berechnen lässt.
- Zerklüftetheit R: Der Buchstabe R stammt aus dem Englischen von „ruggedness“. Zerklüftetheit, das in anderen Publikationen auch als Anzahl der Komponenten bezeichnet wird, beschreibt die Anzahl der nicht miteinander verknüpften Teilmaps.(Stracke, 2003) Hierbei gilt nach (Bonato, 1990) und (Eckert, 1998) ein großer Grad an Zerklüftetheit als ein „Indiz für wenig integrierte Wissensstruktur“(Stracke, 2003). Dabei läuft der Wert von 1 (eine große Map) bis zu einem Maximum von n (Maximale Anzahl an kleinen Maps).

Anders als beim „structural scoring“ wird bei (Stracke, 2003) allerdings kein Verfahren gezeigt, mit dem man die berechneten Werte U und R zusammenfassen kann, um einen einzigen Wert zu bekommen, der insgesamt als Analysewert dienen kann, so dass eine Bewertung nach diesem Verfahren schwierig ist.

Beim „structural scoring“ aus (McClure, et al., 1998) wird nicht nur die Korrektheit von Relationen überprüft, sondern auch die Präsenz einer hierarchischen Ordnung bewertet. Es werden also Punkte für die Anzahl der hierarchischen Level und identifizierten Querverweise verteilt. Wie in Abschnitt 2.1.1 erwähnt, sind hierarchische Strukturen solche Zweige, die untergeordnete bzw. übergeordnete Kategorien von Konzepten verbinden. Querverweise sind dann Kanten zwischen Konzepten unterschiedlicher hierarchischer Zweige.

Somit hat ein Korrektor in (McClure, et al., 1998) in Abbildung 3 17 gültige Verbindungen, 3 hierarchische Ebenen, 2 Querverweise sowie 2 gültige Beispiele identifiziert. Dieses führt zu einem Gesamtergebnis von 36 Punkten.

2.1.4. Kritik

Nach (Weinert, 2002) gelten Kompetenzen als „die bei Individuen verfügbaren oder durch sie erlernten kognitiven Fähigkeiten und Fertigkeiten, um bestimmte Probleme zu lösen, sowie die damit verbundenen motivationalen, volitionalen und sozialen Bereitschaften und Fähigkeiten, um die Problemlösung in variablen Situationen erfolgreich und verantwortungsvoll nutzen zu können“(Weinert, 2002). Es lässt sich erkennen, dass es ein weitgefächerter Begriff ist, der sich nicht kurz umschreiben lässt. Zu beachten ist, dass es sich hierbei sowohl um kognitive Fähigkeiten als auch um die Motivation und die soziale Bereitschaft handelt. Auch in (Klieme, 2004) werden Kompetenzen und Möglichkeiten zu deren Messung beschrieben. Hier wird erwähnt, dass Kompetenzen unter anderem auch darin bestehen, vorhandenes Wissen auf neue Aufgaben anzuwenden, um anspruchsvolle Aufgaben zu bewältigen. Ein Beispiel für eine Kompetenz wäre: „mathematisches Problemlösen und Argumentieren“(Klieme, 2004). Auch diese Definition zeigt, dass der Begriff Kompetenz über das inhaltliche Wissen und kognitive Fähigkeiten hinausgeht.(Christiansen, 2007)

Nach Pisa gilt, dass die naturwissenschaftliche Grundbildung als „die Fähigkeit, naturwissenschaftliches Wissen anzuwenden, naturwissenschaftliche Fragen zu erkennen und aus Belegen Schlussfolgerungen zu ziehen, um Entscheidungen zu verstehen und zu treffen, die die natürliche Welt und durch menschliches Handeln an ihr vorgenommenen Veränderungen betreffen“(Deutsches PISA Konsortium, 2007) verstanden wird.(Christiansen, 2007) Bei der internationalen PISA-Rahmenkonzeption 2003 wird von fünf Prozessen ausgegangen:

1) „naturwissenschaftliche Fragestellungen erkennen
2) naturwissenschaftliche Nachweise identifizieren
3) Schlussfolgerungen ziehen und bewerten
4) gültige Schlussfolgerungen kommunizieren
5) Verständnis naturwissenschaftlicher Konzepte zeigen“(Christiansen, 2007)

(Riuz-Primo, et al., 1996) untersucht, welche Probleme und Fragen mit Concept Maps im Zusammenhang mit wissenschaftlicher Bewertung bestehen. Dabei wird festgestellt, dass, unabhängig davon, auf welche Art und Weise gemessen wird, Concept Maps immer nur kognitive Strukturen messen.(Riuz-Primo, et al., 1996) Es wird also nur auf einen der fünf Prozesse der PISA-Rahmenkonzeption eingegangen, nämlich den fünften Punkt: „Verständnis naturwissenschaftlicher Konzepte zeigen“(Christiansen, 2007). Concept Maps sind also nicht in der Lage, alle Kompetenzen zu messen.
Zeigen lässt sich dieses außerdem an der Taxonomie von Anderson. In Tabelle 1 sind die „Knowledge Dimension“ und die „Cognitive Process Dimension“(Anderson, et al., 2000) zu erkennen. Es gibt hierzu eine deutsche Übersetzung (BLK-Projekt, 2004), in der sie als „Wissens-Dimension“ und „Kognitive Prozess-Dimension“ bezeichnet werden.
Für die verschiedenen Bereiche gibt es unterschiedliche Prüfmethoden.
Concept Maps befinden sich, wie der Name schon sagt, in der 2. Wissens-Dimension, dem „Conceptual Knowledge“(Anderson, et al., 2000) oder auch „Begriffliches Wissen“(BLK-Projekt, 2004). Wenn man vom reproduzieren des Wissens, wie eine

Concept Map gezeichnet wird, usw. absieht, kommt ein Proband beim Zeichnen einer Concept Map allerdings nicht über die dritte Dimension des kognitiven Prozesses hinaus. Also führt er beim zeichnen einer Concept Map keine „Analyse“, „Bewertung“ oder „Erschaffung“ durch. Somit sind Concept Maps nicht als ausschließliches Mittel zur Lernüberprüfung geeignet. Es wäre natürlich denkbar zum Beispiel die „Analyse“ anzusprechen, indem man zum Beispiel den Probanden die Concept Map einer anderen Person bewerten lässt.

DIE WISSENS-DIMENSIONEN	DIE KOGNITIVE PROZESS-DIMENSION					
	1. **Erinnern**	**2.** **Verstehen**	**3.** **Anwenden**	**4.** **Analysieren**	**5.** **Bewerten**	**6.** **(Er)schaffen**
A. Faktenwissen						
B. Begriffliches Wissen						
C. Verfahrensorientiertes Wissen						
D. Metakognitives Wis-sen						

Tabelle 1: Taxonomie Tabelle(BLK-Projekt, 2004)

Außerdem wird in (Riuz-Primo, et al., 1996) festgestellt, dass es noch weitere Untersuchungen geben muss, bevor Concept Maps für Klassen oder groß angelegte Bewertungen herangezogen werden können. Bevor die Ergebnisse aus Concept Maps Lehrern, Studenten oder der Öffentlichkeit präsentiert werden können, muss es reliable und valide Überprüfungen geben, die die Effekte von unterschiedlichen Techniken auf die Bewertung der kognitiven Struktur des Probanden prüfen.

2.2. Graphen

2.2.1. Graphentheorie

„Ein Graph ist ein Paar $G = (V, E)$ disjunkter Mengen mit $E \subseteq [V]^2$“ (Diestel, 2006) Somit sind die Elemente aus E bei einem gerichteten Graphen also Tupel von Elementen aus V. Dabei bezeichnet man die Elemente aus E als Kanten und die Elemente aus V als Knoten. Bei der graphischen Darstellung kann man die Knoten als Punkte zeichnen, welche anschließend durch Linien verbunden werden. Diese Linien repräsentieren die Kanten.
Es gibt gerichtete und ungerichtete Graphen. (Gellert, et al., 1977) Ein Beispiel hierfür ist in Abbildung 5 zu sehen. Beim gerichteten Graphen ist die Reihenfolge der Knoten von Bedeutung. Bei $(x, y) \in E$ verläuft die Kante nur von Knoten x zu Knoten y und nicht von y nach x. In einer bildlichen Darstellung würde man dieses durch einen Pfeil statt einer Linie repräsentieren. Dem zur Folge ist die Reihenfolge der Knoten $\{x, y\} \in E$ bei ungerichteten Graphen also ohne Bedeutung. Um diesen Unterschied deutlich zu machen, schreibt man eine gerichtete Kante in runde Klammern: „(“ und ungerichtete Kanten in geschweifte Klammern: „{“

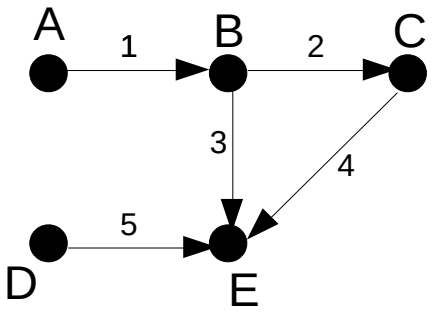


Abbildung 5: Beispielgraph

Beim Zeichnen der Kanten ist es unwichtig, ob die Linien geschwungen oder gerade, disjunkt oder überkreuzt verlaufen. Dieses sind nur Dinge der „Zweckmäßigkeit und der Ästhetik“(Diestel, 2006). Für die formale Definition eines Graphen spielt die bildliche Repräsentation keine Rolle.(Diestel, 2006)

Ein Graph ist endlich oder unendlich, je nachdem ob die Menge V endlich oder unendlich ist. $|V|$ bezeichnet die Kardinalität von V, also die Anzahl der Elemente in V. Sie wird auch als die Ordnung des Graphen bezeichnet. Graphen der Ordnung 0 oder 1 sind triviale Graphen. Den leeren Graphen: $G = (\emptyset, \emptyset)$bezeichnet man als $\emptyset$.(Diestel, 2006)

Zwei Knoten $x, y \in V$ des Graphen $G = (V, E)$ heißen adjazent oder benachbart, wenn es eine Kante $\{x, y\} \in E$ gibt. Zwei Kanten $x, y \in E \; mit \; x \neq y$ heißen benachbart, falls sie einen gemeinsamen Knoten besitzen. Wenn zwei Kanten oder Knoten allerdings nicht benachbart sind, heißen sie unabhängig.(Diestel, 2006)

Beim vergleichen der obigen Definition eines Graphen und der Darstellung von Concept Maps aus Abschnitt 2.1.1 wird offensichtlich, dass eine Concept Map ein gerichteter Graph mit Kantennamen ist.

2.2.2. Implementierung von Graphen

Graphen können auf verschiedene Art und Weisen implementiert werden. Alternative Implementierungsmethoden sind beispielsweise Adjazenzmatrizen, Adjazenzlisten, Inzidenzmatrizen und Inzidenzlisten die im Folgenden erläutert werden. (Saake, et al., 2004; Leiserson, et al., 2007; Wackersreuther, 2007)

Eine Adjazenzmatrix ist eine Matrix, in die Zeilen und Spalten je einen Knoten repräsentieren. Es ist also eine $|V| \times |V|$ –Matrix mit den Elementen

Formel 6: Adjazenzmatrixeintrag

$$a_{ij} = \begin{cases} 1 \; falls \; (i,j) \in E \\ 0 \; sonst. \end{cases}$$

(Leiserson, et al., 2007; Hachenberger, 2008) Der in Abbildung 5 gezeichnete Graph wird in Abbildung 6 als Adjazenzmatrix dargestellt.

	A	B	C	D	E
A	0	1	0	0	0
B	0	0	1	0	1
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

Abbildung 6: Adjazenzmatrix

Die Adjazenzliste bestehen dagegen aus $|V|$ Listen. Dabei steht jede Liste für einen Knoten $v \in V$. Jede Liste $Adj[v]$ enthält für jeden Knoten $k \in V$ einen Eintrag, für die es eine Kante $\{k, v\} \in E$ gibt. (Leiserson, et al., 2007) Abbildung 7 zeigt wiederum den in Abbildung 5 dargestellten Graph als Adjazenzliste.

$$A: \{B\}$$
$$B: \{C, E\}$$
$$C: \{E\}$$
$$D: \{E\}$$
$$E: \{\ \}$$

Abbildung 7: Adjazenzliste

Die Adjazenzliste wird in der Regel vorgezogen, da sie für dünn besetzte Graphen, das heißt Graphen für die $|E| \ll |V^2|$ gilt, kompakter ist. Dementsprechend wird bei einem dichten Graphen die Adjazenzmatrix gewählt. Der Speicheraufwand von Adjazenzlisten ist allerdings linear mit $\Theta(V + E)$ ist, während Adjazenzmatrizen quadratischen Speicheraufwand bezüglich der Knoten benötigen: $\Theta(V^2)$. (Leiserson, et al., 2007)

Die Inzidenzmatrix bezieht sich im Gegensatz zur Adjazenzmatrix und Adjazenzliste nicht auf die Knoten, sondern auf die Kanten. Somit ergibt sich für den Graphen $G = (V, E)$ eine $|V| \times |E|$ Matrix. Jede Spalte stellt also eine Kante dar, und enthält somit genau zwei von 0 verschiedene Einträge. Während die 1 den Beginn einer Kante zeigt, stellt die -1 den Endknoten der Kante dar. Bei einem ungerichtetem Graphen gibt es hierbei natürlich keine Unterscheidung. In dem Fall wird zweimal die 1 gewählt. (Leiserson, et al., 2007) Abbildung 8 zeigt wiederum den aus Abbildung 5 bekannten Graphen als Inzidenzmatrix.

	1	2	3	4	5
A	1	0	0	0	0
B	-1	1	1	0	0
C	0	-1	0	1	0
D	0	0	0	0	1
E	0	0	-1	-1	-1

Abbildung 8: Inzidenzmatrix

Äquivalent gibt es auch noch Inzidenzlisten. Diese sind deutlich Speichersparender. Denn wie deutlich zu erkennen ist, ist die Inzidenzmatrix immer nur sehr dünn

besetzt. Bei einer Inzidenzliste werden die vorhanden Kanten als Liste abgespeichert. Allerdings gibt es für jeden Knoten $v \in K$ eine eigene Liste, mit allen Kanten $(k, x) \in E$.(Wackersreuther, 2007) Die Abbildung 9 verdeutlicht dieses für den Graphen aus Abbildung 5

$$A: \{1\}$$
$$B: \{2, 3\}$$
$$C: \{4\}$$
$$D: \{5\}$$
$$E: \{\ \}$$

Abbildung 9: Inzidenzliste

In Computersystemen gibt es nun verschiedene Möglichkeiten der Speicherung von Graphen. Diese unterscheiden sich durch ihr Format, wobei die Knoten und Kanten in unterschiedlicher Reihenfolge und mit unterschiedlichen Trennzeichen von einander in einer Datei gespeichert werden.

Ein Format, das zur Speicherung von Graphen Anwendung findet, ist das einfach gehaltene TGF-Format. TGF steht für „Trivial Graph Format“. Hierbei werden zunächst alle Kanten durchnummeriert und aufgeschrieben. Nach einem Trennzeichen (#) werden die Kanten inklusive Kantennamen aufgelistet.(yWorks, Betrachtungsdatem: 14.01.2012) Abbildung 10 zeigt, wie der Graph aus Abbildung 5 im TGF aussieht.

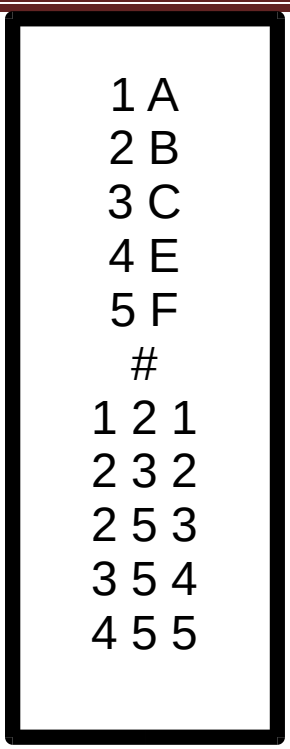


Abbildung 10: TGF

Ein weiteres Format zur Speicherung von Graphen auf Computersystemen ist GraphML. Dieses Format ist ein XML basiertes Graphenformat. Aus diesem Grund ist es besonders gut möglich es im Zusammenhang mit anderen auf XML basierenden Formaten nutzbar.(Brandes, et al., 2004) In Abbildung 11 ist ein Beispiel für das Format gegeben.

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
     http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
  <graph edgedefault="directed">
    <node id="v1"/>
    <node id="v2"/>
    <node id="v3"/>
    <node id="v4"/>

    <edge source="v1" target="v2"/>
    <edge source="v1" target="v3"/>
    <edge source="v2" target="v4"/>
    <edge source="v2" target="v4" directed="false"/>
  </graph>
</graphml>
```

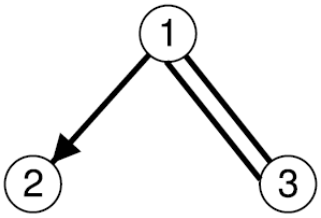


Abbildung 11: GraphXML(Brandes, et al., 2004)

Zur Darstellung von Graphen auf Computersystemen wird also ein vorgegebenes Format benötigt. Mit dessen Hilfe kann der Computer den Graphen zeichnen und speichern, oder auf eine andere Art und Weise repräsentieren und weitergeben.

2.3. Software Engineering

2.3.1. Begriffsklärung Software Engineering

Software Engineering bezeichnet das ingenieursmäßige Vorgehen beim Entwickeln umfangreicher Softwaresysteme, zielt auf die termingetreue, kostengerechte Entwicklung funktionierender, nutzergerechter Software ab. Es findet im Spannungsfeld zwischen Kosten, Qualität und Terminen statt und umfasst viele unterschiedliche Aspekte aus Fachlichkeit, Softwaretechnik sowie Projektorganisation und Management. (Balzert, 2000)
Diese Beschreibung bringt die Problematik der Softwareentwicklung auf den Punkt, denn bei der heutigen Entwicklung von Software muss man viele Umstände und Probleme beachten, um ein Projekt erfolgreich zum Abschluss zu bringen. Einige Beispiele hierfür wären:

a) Die Software wird über langen Zeitraum mit ständiger Veränderung des technischen Standes, eingesetzt.
b) Moderne Softwaresysteme sind nicht mehr autark sondern kommunizieren mit einer Fülle von anderen Systemen (Servern, Datenbanken, anderen Softwaresystemen).
c) Oft wird Software nicht neu entwickelt, sondern weiterentwickelt oder an bestehende Systeme angeschlossen und angepasst.
d) Softwareentwicklung ist ein Markt; man entwickelt meist für einen Kunden, der gewisse Vorstellungen vom Endprodukt hat.
e) Mittlere bis größere Softwareprojekte bringen eine immer größer werdende Komplexität mit sich.
f) Das System wird eventuell selbst auch in der Zukunft weiter entwickelt, verändert oder angepasst.
g) An einem Softwaresystem entwickeln oft mehrere Entwickler parallel.
h) Es gibt viele unterschiedliche Interessengruppen mit unterschiedlichen, meist komplementären Absichten, die zusammengeführt werden müssen.(Brügge, et al., 2004)

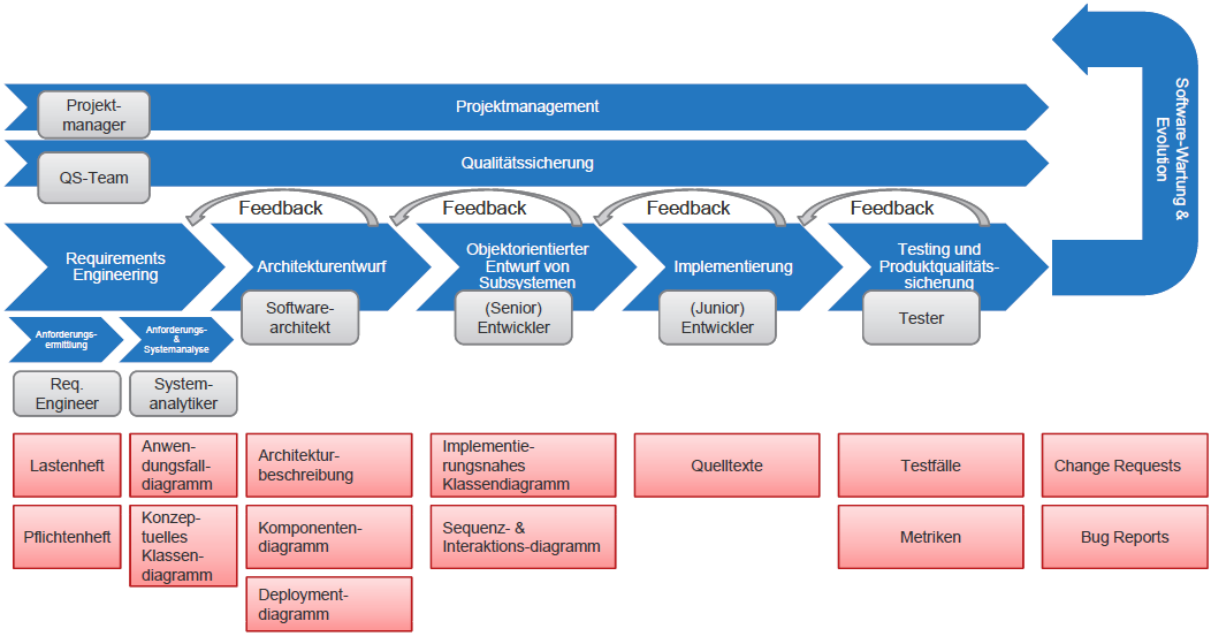


Abbildung 12: Softwareentwicklungsprozess aus (Matthes, 2009)

Aus diesen Anforderungen entstehen zwangsläufig Probleme, die gelöst werden müssen. Dafür sollte ein Softwareprojekt einen gewissen Rahmen haben, an dem sich alle Beteiligten orientieren können, sowohl der Auftraggeber, als auch die Entwickler und alle anderen beteiligten Personengruppen. (Brügge, et al., 2004) Dieser Rahmen wird, wie in der anfänglichen Beschreibung schon erwähnt, durch das Software Engineering geboten und im Folgenden näher beleuchtet.

2.3.2. Projektmanagement

„Projektmanagement soll die Lieferung eines Systems mit hoher Qualität in einen gegebenen Zeit-und Kostenrahmen ermöglichen.“(Brügge, et al., 2004) Ein Projekt besteht aus mehreren Komponenten: dem abzuliefernden Ergebnis, welches aus mehreren zu erreichenden Meilensteinen bestehen kann; der durchzuführenden Arbeit, welche man in kleinere Pakete zerlegen kann; einem Zeitplan, der die zu erreichenden Ergebnisse in eine Zeitachse einordnet; die zur Verfügung stehenden Ressourcen, zu denen sowohl Betriebsmittel als auch Menschen zählen.(Brügge, et al., 2004)

Die Aufgabe des Projektmanagements besteht nun darin, eine umfangreiche Projektplanung durchzuführen, die diese Komponenten berücksichtigt. Dies beinhaltet, Projektorganisation und die Identifikation von Verantwortlichkeiten und Rollen, die Steuerung des Projekts und das Festhalten wichtiger Erfahrungen am Ende eines Teilprojekts oder des Gesamtprojekts, um sie dem nächsten (Teil-)Projekt zur Verfügung zu stellen.(Brügge, et al., 2004)

Ein Softwareprojekt kann in 5 Phasen untergliedert werden:

a. Konzeptionsphase

In dieser Phase entsteht die Projektidee. Dies geschieht meist durch einen Kunden, der einen neuen Auftrag erteilt. Bei großen Projekten muss zunächst geprüft werden, ob ein Auftrag lohnenswert ist (Kosten-Nutzen-Analyse) und ob er mit den vorhandenen Mitteln umsetzbar ist bzw. ob benötigte, aber noch nicht vorhandene Ressourcen beschafft werden können (Machbarkeitsstudie). Wird sich für das Projekt entschieden, wird es für die nächsten Phasen freigegeben und personelle Entscheidungen hinsichtlich eines Projektleiters und eines Softwarearchitekten getroffen. Natürlich können dem Projekt, abhängig von dessen Größe, noch weitere Ressourcen zugeteilt werden. (Brügge, et al., 2004)

b. Definitionsphase

In dieser Phase werden mit dem Kunden und dem Softwarearchitekten die Rahmenbedingungen des Projekts besprochen. Dabei werden Probleme definiert, ein vorläufiger Projektplan ausgehandelt, über eine vorläufige Systemarchitektur gesprochen und Projektvereinbarungen des Kunden und des Projektleiters festgehalten. Außerdem wird ein formelles Lieferdatum festgelegt.(Brügge, et al., 2004)

c. Startphase

In der dritten Phase wird die Infrastruktur des Projekts festgelegt, sowie personelle Entscheidungen hinsichtlich der Projektmitarbeit getroffen. Beispielsweise kann mit Hilfe einer Fähigkeitsmatrix der Projektleiter die Fähigkeiten seiner Mitarbeiter festhalten und in passende Arbeitsgruppen einteilen.(Brügge, et al., 2004)

d. Stationäre Phase

In dieser Phase übernimmt der Projektleiter fast alle Managementfunktionen. Er diskutiert und vereinbart mit dem Kunden die funktionellen und nicht-funktionellen Anforderungen (vgl. Kapitel 2.3.4.1) und kontrolliert das Projekt durch Meetings und Statusberichte von seinen eingeteilten Gruppenleitern. Außerdem ist er für ein umfangreiches Risikomanagement zuständig um Probleme und Verzögerungen, die das Projekt verlangsamen und/oder gefährden könnten, zu identifizieren und zu vermeiden oder diesen frühzeitig entgegen zu wirken.(Brügge, et al., 2004)

e. Terminierung

Bei der Terminierung wird das Projektergebnis an den Kunden ausgeliefert und die Projekthistorie gesichert und archiviert. Hierfür wird das Projekt von den Hauptentwicklern, Dokumentationserstellern und Gruppenleitern für die Übergabe fertiggestellt. Für die Wiederverwendung bei zukünftigen Projekten wird die Projekthistorie gesammelt.(Brügge, et al., 2004)

All diese Schritte sollten in einem Softwaremanagement-Plan festgehalten werden.

2.3.3. Software-Lebenszyklus

„Es gibt immer eine Diskrepanz zwischen einem Konzept und der Realität, da Ersteres statisch und Letzteres dynamisch und fließend ist.“(Pirsig, 2006)

Ein Softwareprojekt enthält viele signifikante Prozesse und Aktivitäten, die in einem Modell abgebildet werden können. Dieses Modell soll den Softwareentwicklungsprozess möglichst realitätsnah abbilden. Er wird vom Projektmanager erstellt. Im Laufe der Zeit haben sich daraus Standardvorgehensmodelle gebildet. „Jede Beziehung [eines solchen Modells] repräsentiert sowohl ein von einem Prozess erzeugtes Arbeitspaket und von einem anderen verbrauchtes Arbeitsprodukt als auch einen formalen Kommunikationskanal zwischen Projektmitgliedern, über den Dokumente, Modelle und Quelltext ausgetauscht werden .“ (Brügge, et al., 2004)

2.3.3.1. Sequentielle, aktivitätsgesteuerte Modelle

a. Wasserfallmodell

Der wichtigste und meist verwendete Vertreter (Heinrich, et al., 2008) dieser Modelle ist das Wasserfall-Modell, dessen Aktivitäten sequentiell nacheinander ausgeführt werden. Ist ein Arbeitsschritt ausgeführt, so ist er abgeschlossen und man geht in die nächste Aktivität über ohne in einen vorherigen Arbeitsschritt zurückkehren zu müssen.

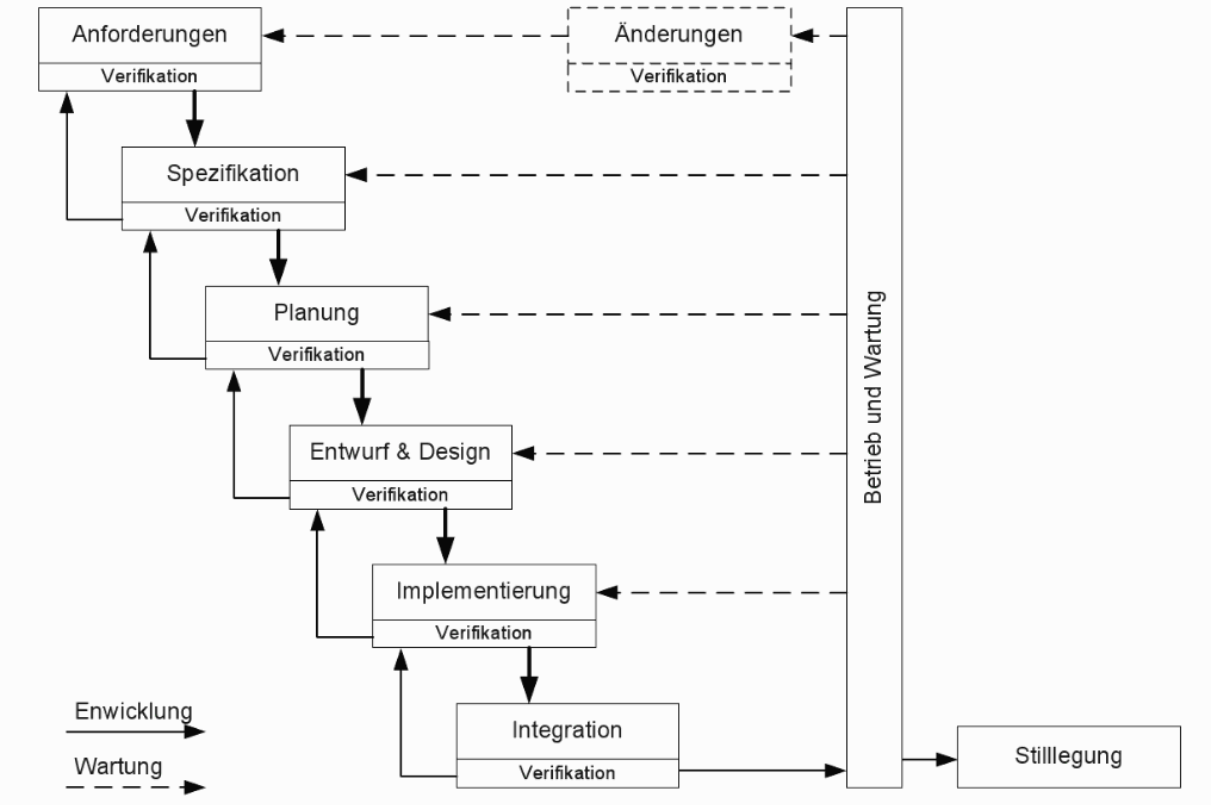


Abbildung 13: iteratives Wasserfallmodell (Schatten, et al., 2010)

Kritik an diesem Modell ist allerdings, dass es unflexibel, starr, veraltet und risikobehaftet sei (Heinrich, et al., 2008; Schatten, et al., 2010). Begründet wird dieses nach (Heinrich, et al., 2008) und (Schatten, et al., 2010) folgendermaßen:

- Unflexibel:
  - Es ist nicht immer möglich, eine Einteilung in solche Phasen vorzunehmen.
  - Es wird bei sehr großen Systemen unübersichtlich.
  - Es gibt keine langfristige Weiterentwicklungsstrategie.

- Starr:
  - Es müssen alle Wünsche des Auftraggebers zu Beginn bekannt sein.
  - Es gibt keinen Zugriff auf vergangene Phasen.
  - Eine Fehlersuche ist schwierig, da die Teams schon aufgelöst wurden.
  - Das Testen findet sehr spät statt.
  - Der Auslieferungszeitpunkt ist sehr spät.
  - Es ist sehr unflexibel im Bezug auf spätere Änderungen.
  - Nachbesserungen sind oft teuer.
  - Der Anwender sieht erst das fertige Produkt.

- Veraltet:
  - Es gibt Kommunikationsprobleme.
  - Es ist kein evolutionärer Prozess.
  - Es ist kein in sich geschlossener Kreislauf.

- Risikobehaftet
  - Fehler werden erst spät gefunden und sind teuer.

b. <u>V-Modell</u>

Das V-Modell ist eine Weiterentwicklung des Wasserfallmodells „welche die Gegenüberstellung der Entwicklungs- und Verifikationsaktivitäten hervorhebt“ (Brügge, et al., 2004). Der linke Teil des V-Modells beschäftigt sich mit einer immer detaillierteren Systembetrachtung, während sich der rechte Teil mit der Systemvalidierung beschäftigt und das System als Ganzes betrachtet.(Brügge, et al., 2004)

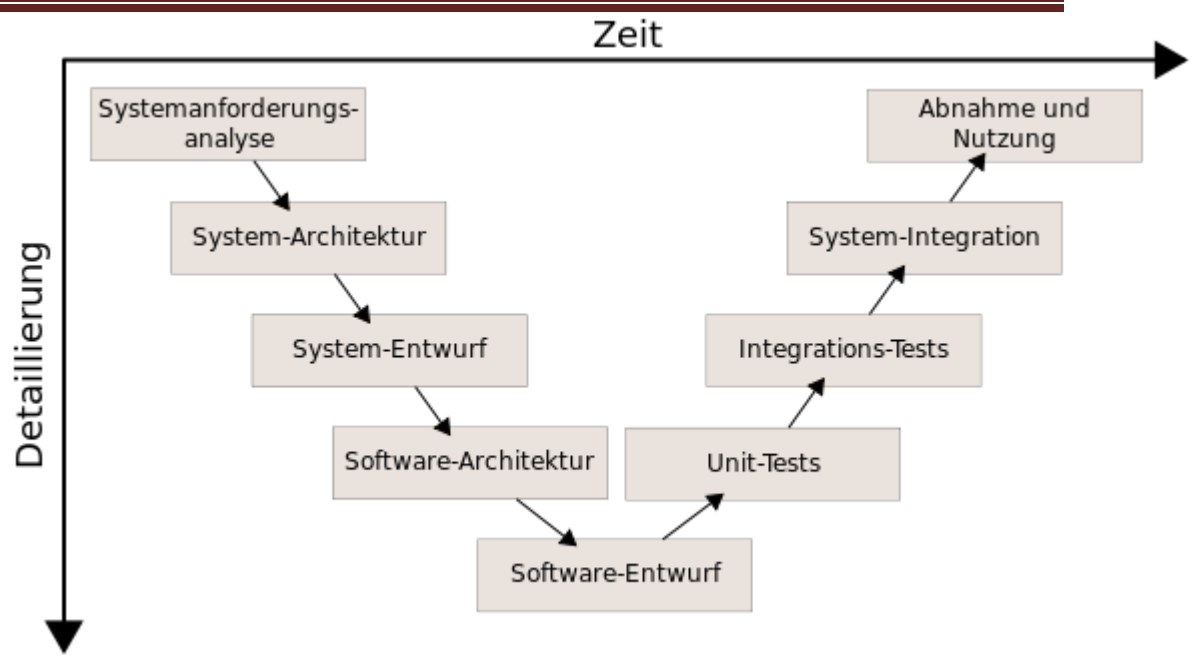


Abbildung 14: V-Modell [wikipedia.de]

Diese Art von Modellen hat sich als unpraktisch herausgestellt, da die Starrheit und Inflexibilität nicht auf die Realität anwendbar ist und eine Rückkehr zu vorherigen Arbeitsschritten bei vor allem mittleren und größeren Projekten fast unabdingbar ist. Außerdem werden in beiden Modellen viele wichtige Aspekte eines Softwareprojekts vernachlässigt.(Brügge, et al., 2004) Um den beschriebenen Nachteilen des V- und Wasserfallmodells entgegenzuwirken, existiert für beide Modelle eine iterative Alternative bei der es Möglich ist in vorherige Arbeitsschritte zurückzukehren. (vgl. Abbildung 13) Beim V-Modell wird dieses zum Beispiel als V-Model XP bezeichnet.(Grande, 2011; C., et al., 2008)

2.3.3.2. Iterative, aktivitätsgesteuerte Modelle

a. Inkrementelles Vorgehensmodell

Inkrementelle Vorgehensmodelle werde auch oftmals als evolutionäre Modelle bezeichnet. Dabei wird versucht möglichst früh erste einsetzbare Ergebnisse zu produzieren. Dabei wird das System zwar als ganzen geplant, jedoch in Teilschritten realisiert. Dabei werden zu Beginn des Projekts die Anforderungen an das spätere System möglichst vollständig erhoben, und dann „in mehreren Ausbaustufen entworfen, entwickelt, eingeführt und genutzt“(Wieczorrek, et al., 2011). Dabei wird jeder Teilabschnitt zunächst getestet, bevor ein neuer begonnen wird. Nicht jedes Projekt eignet sich, um es mit dem inkrementellen Vorgehensmodell umzusetzen.(Wieczorrek, et al., 2011)

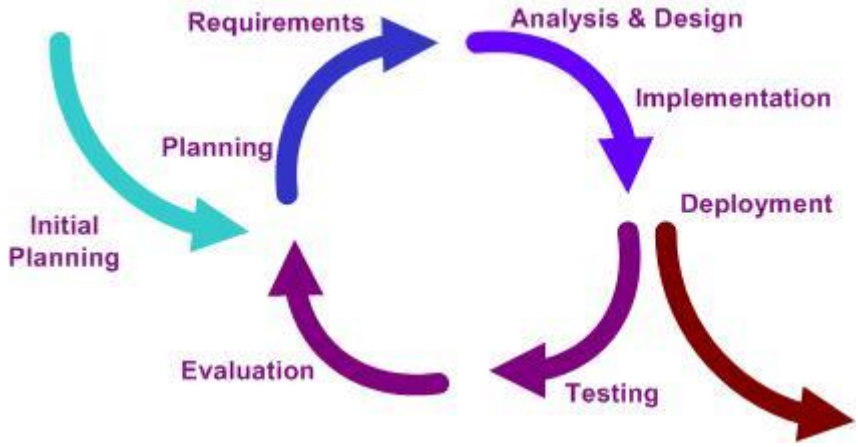


Abbildung 15: inkrementelles Vorgehensmodell [wikipedia.de]

b. Das RUP-Modell

Dieses Modell (in anderen Quellen auch als USP- Modell bezeichnet (Brügge, et al., 2004)) besteht aus den folgenden vier Meilensteinen und versucht sich an bewährte Erfahrungswerte aus der Praxis zu orientieren:(Essigkrug, et al., 2007)

- Inception (Konzeptualisierungsphase)
- Elaboration (Entwurfsphase)
- Construction (Konstruktionsphase)
- Transition (Übergangsphase)

„Diese vier Phasen geben den zeitlichen Verlauf des Vorhabens wieder.“(Schienmann, 2002)

c. Spiralmodell nach Bohem

Das Spiralmodell versucht die Schwächen des Wasserfallmodells zu neutralisieren. Es unterscheidet für jede Aktivität mehrere Teilschritte, die mehrmals iterativ durchgeführt werden können. Diese „Runden“ beinhalten nun auch andere Schritte der Softwarenentwicklung, die das Wasserfall/V-Modell nicht enthielten wie zum Beispiel das Risikomanagement oder die Wieder-verwendung.

Eine Runde besteht aus folgenden Schritten:(Brügge, et al., 2004)

a) Zielbestimmung
b) Spezifikation der Rahmenbedingungen
c) Erzeugung von Alternativen
d) Identifikation von Risiken
e) Behebung von Risiken
f) Entwicklung und Verifikation des Produkts
g) Planung

Abbildung 16: Spiralmodell (Matthes, 2009)

Es gibt noch viele andere Arten von Modellen, wie zum Beispiel entitätsgesteuerte Modelle, die sich besonders gut eignen, „wenn die Zeitabstände zwischen Änderungen die Dauer einer Iteration deutlich unterschreiten und wenn Änderungen sowohl in der Anwendungs- als auch in der Lösungsdomäne auftreten können“(Brügge, et al., 2004). Diese werden an dieser Stelle allerdings nicht weiter erläutert, da sie für die Arbeit nicht weiter relevant sind.

2.3.4. Projektentwicklung

2.3.4.1. Anforderungsermittlung

Die Anforderungsermittlung (Requirements Engineering) ist der erste und doch zugleich einer der wichtigsten Schritte im Softwareentwicklungsprozess. Wie das 1. Gesetz von Boehm besagt: „Fehler passieren am häufigsten während des Requirements Engineerings und des Designs; je später sie entdeckt und entfernt werden, desto teurer sind sie.“(Brügge, et al., 2004)
Es muss also zunächst mit dem Auftraggeber, der im Weiteren auch als Kunde bezeichnet wird, abgeklärt werden, was das System konkret leisten soll. Dabei muss darauf geachtet werden, dass möglichst alle Interessensgruppen vertreten sind, um möglichst alle Anforderungen zu identifizieren und zu berücksichtigen. Dazu gehört nicht nur, welche Eigenschaften das zu entwickelnde System hat, sondern auch in welchem Rahmen es eingesetzt wird, welche Abhängigkeiten zu anderen Systemen bestehen und ob ein System neu- oder weiterentwickelt wird. Dazu versucht man zunächst grobe Anforderungen zu ermitteln um anschließend Feinere von diesen abzuleiten. Nachdem die Anforderungen identifiziert und erfasst wurden, müssen sie abgestimmt, priorisiert und bewertet werden, denn nicht jede Anforderung ist gleich gewichtig. Da ein Projekt auch einen gewissen zeitlichen oder finanziellen Rahmen hat, sind die Anforderungen, die später tatsächlich umgesetzt werden können, begrenzt. Zuletzt sollten die Anforderungen noch schriftlich dokumentiert werden.(Sommerville, et al., 1997)

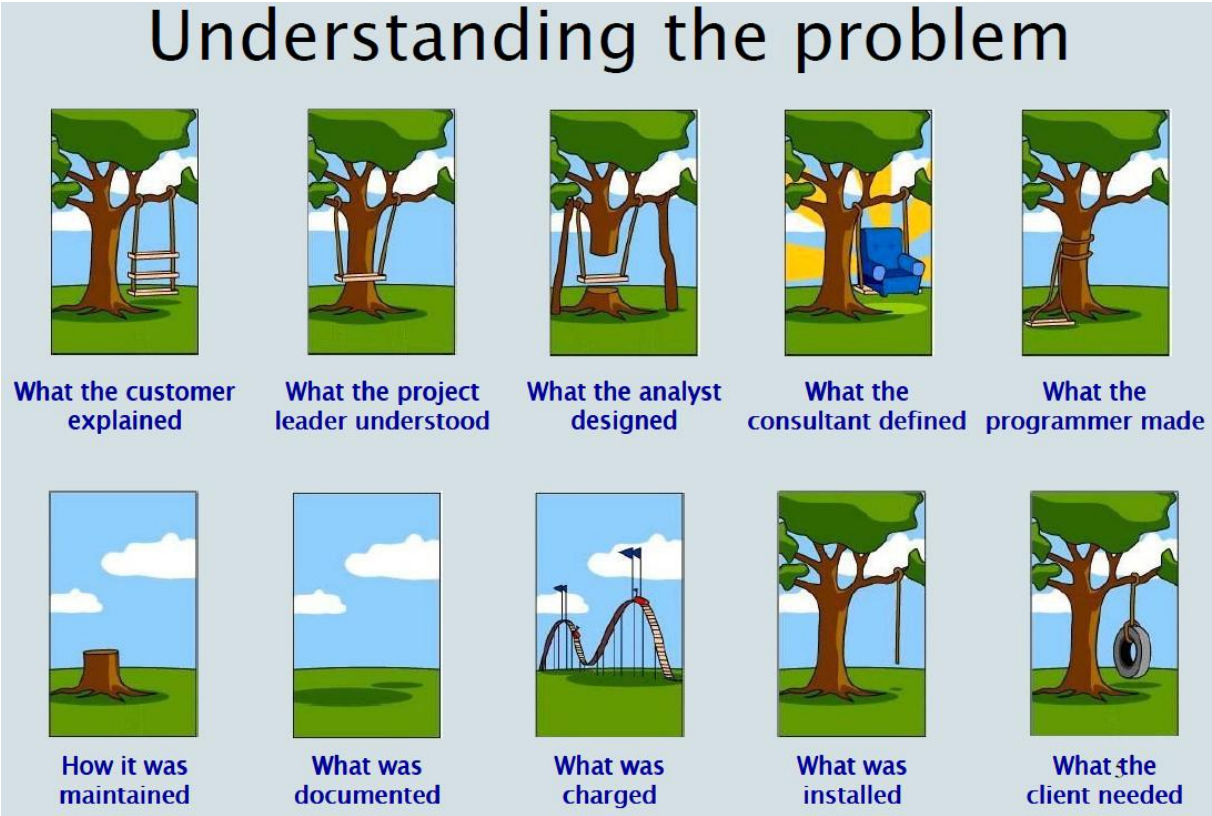


Abbildung 17: Understanding the problem (Chaudron, 2008)

Funktionale und nicht funktionale Anforderungen

Die gesammelten Anforderungen lassen sich klassifizieren in funktionale und nicht-funktionale Anforderungen. „Funktionale Anforderungen beschreiben die Interaktion des Systems mit seiner Umgebung unabhängig von seiner Implementierung. Die Umgebung beinhaltet den Anwender und alle anderen externen Systeme, die auf das System einwirken:“ (Brügge, et al., 2004) Nicht funktionale Anforderungen beschreiben hingegen „[...]Aspekte des Systems, die nicht unmittelbar in Bezug zu seiner Funktionalität stehen.“(Brügge, et al., 2004). Nicht-funktionale Anforderungen lassen sich nach dem FURPS Modell gliedern: Dazu gehören die Bedienbarkeit des Systems und seine Zuverlässigkeit. Als zuverlässig gilt ein System, wenn ein System oder eine Komponente „die erforderlichen Funktionen unter vorgegeben Bedingungen während eines vorgegebenen Zeitrahmens erfüllt:“(Brügge, et al., 2004) Die zeitliche Komponente ist dabei wichtig zu erwähnen, da sie zum Beispiel bei Echtzeitsystemen entscheidend ist. Zur Kategorie "Zuverlässigkeit" zählen auch die Kategorien Robustheit und Sicherheit. Eine weitere nicht-funktionale Anforderung ist die Leistungsanforderung - hierzu zählen Antwortzeit, Durchsatz und Verfügbarkeit des Systems. Die letzte Kategorie bildet die Unterstützbarkeit, welche beinhaltet wie leicht das System an spätere konzeptuelle Änderungen anzupassen ist, wie wartungsfreundlich und wie portabel es ist. Als portabel bezeichnet man ein System, das leicht von einer Hardware- oder Softwareumgebung in eine andere transferiert werden kann.(Brügge, et al., 2004)

Die Anforderungen werden nach ihrer Zusammenstellung noch Priorisiert. Dabei unterscheidet man zwischen Muss/Essenziell, Soll/„bedingt notwendigen“ und optionalen Anforderungen. Dabei gilt, dass eine Software nicht akzeptiert wird, wenn nicht alle essenziellen Anforderungen erfüllt wurden. Erfüllte „bedingt notwendige“ Anforderungen dagegen werten eine Software auf. Aber falls sie nicht realisiert wurden führt dieses nicht zu einer Ablehnung des Projekts. Die optionalen Requirements ermöglichen es dem Auftragnehmer über die die vorhandenen Anforderungen hinaus zu gehen. (Balzert, 2009; Matthes, 2009)

Das FURPS+ Modell unterscheidet zusätzlich nach sogenannten Constrains. Dazu zählen Implementierungsanforderungen ("Vorgaben zum Gebrauch spezieller Werkzeuge, Programmiersprachen oder Hardwareanforderungen" (Brügge, et al., 2004)), Schnittstellenanforderungen, betriebliche Anforderungen (hinsichtlich Administration und Management des Systems, welche durch den Betrieb vorgegeben sind), Verpackungsanforderungen und rechtliche Anforderungen (zum Beispiel Lizenzbestimmungen).(Brügge, et al., 2004)

Lastenheft

Das Lastenheft enthält alle projektbezogenen Forderungen des Auftraggebers gegenüber dem Auftragsnehmer und dokumentiert somit einen Teil der Anforderungen, die in der Anforderungsermittlung identifiziert wurden. Es beschreibt die Ziele, die durch das Produkt erreicht werden sollen, den Produkteinsatz, die Produktumgebung und die Produktfunktionen. Die Produktfunktionen decken sich dabei mit den funktionalen Anforderungen der Anforderungsermittlung. Desweiteren werden die Produktdaten, die Produktleistungen und die Qualitätsanforderungen festgehalten. Als Produktdaten bezeichnet man die Kerndaten des Systems, die eine

Abschätzung der zu bewältigenden Datenmenge liefern sollen. (Hood, et al., 2008; Sommerville, et al., 1997; Balzert, 2009; Balzert, 2000)

Pflichtenheft

Als Reaktion auf das Lastenheft erstellt der Auftragnehmer ein Pflichtenheft. Dieses bietet die Möglichkeit, die ermittelten Anforderungen der Anforderungsermittlung geeignet zu dokumentieren. Es enthält die Projektziele, eine allgemeine Beschreibung über die geplante Systemumgebung, geplante Funktionen und voraussichtliche Benutzer. Anschließend kann das momentane System (falls ein solches existiert) und danach das geplante System beschrieben werden. Bei der Beschreibung des geplanten Systems werden funktionale und nicht-funktionale Anforderungen, Constrains, Skizzen zum Entwicklungszyklus oder der Projektorganisation, Lieferumfang und Abnahmekriterien beschrieben [49]. Zuletzt kann man versuchen, die Zukunft des Systems hinsichtlich voraussichtlicher Erweiterungen, Anpassungen oder der Betriebsdauer abzuschätzen. (Hood, et al., 2008; Sommerville, et al., 1997; Balzert, 2009; Balzert, 2000)

Anforderungs- und Systemanalyse

Aus den ermittelten Anforderungen der Anforderungsermittlung können Akteure, Szenarien und Use Cases identifiziert werden. Dies hilft sowohl dem Auftraggeber als auch dem Auftragnehmer das System besser zu verstehen und einen weniger abstrakten Blick darauf zu werfen. Dies trägt auch dazu bei, frühzeitig aufzudecken, ob es bereits zu Missverständnissen zwischen dem Kunden und dem Auftragnehmer bezüglich des zu entwickelnden Systems gekommen ist.(Brügge, et al., 2004)

Akteure

Als Akteur bezeichnet man "Entitäten, die mit dem zu entwickelnden System interagieren. Bei einem Akteur kann es sich sowohl um einen Menschen als auch um ein anderes System handeln.“ (Brügge, et al., 2004) Um einen Akteur zu identifizieren, muss zunächst die Systemgrenze des zukünftigen Systems definiert werden. Akteure befinden sich immer außerhalb dieser Systemgrenze und tauschen Informationen mit dem System über Schnittstellen aus. Alle Objekte innerhalb der definierten Systemgrenze sind Objekte oder Subsysteme. Man kann zwischen "initiating" und "participating" Akteuren unterscheiden. Initierende Akteure stoßen eine Aktion im System an; partizipierende Akteure nehmen an einer angestoßenen Aktion teil. Akteure sind stets als abstrakte Rollen zu verstehen. Der gleiche Akteur kann also zu unterschiedlichen Zeitpunkten dem System in verschiedenen Rollen begegnen. (Brügge, et al., 2004)

Szenarien

„Ein Szenario ist eine konkrete, fokussierte und informelle Beschreibung eines einzelnen Systemmerkmales aus der Sicht eines einzelnen Nutzers.“ (Brügge, et al., 2004) Dabei können Kernszenarien unterschieden werden. Diese beschreiben die Hauptfunktionalitäten des Systems. Ein Szenario bezieht sich immer auf konkrete Situationen, die im System auftreten können und wird von konkreten Instanzen eines Akteurs angestoßen.(Brügge, et al., 2004)

Use Case

Ein Szenario stellt eine konkrete Instanz eines Anwendungsfalls (Use Case) dar. Ein oder mehreren Szenarien können also zu einem Use Case zusammengefasst werden. Ein Anwendungsfall beinhaltet die beteiligten Akteure, den Ereignisfluss, der durch den initiierenden Akteur angestoßen wird, und die Reaktion des Systems. Außerdem werden die Bedingungen, unter dem der Anwendungsfall stattfinden kann und der Zustand des Systems, nachdem der Anwendungsfall eingetreten ist, beschrieben. Zusätzlich können alternative Verläufe des Anwendungsfalls und Qualitätsmerkmale festgehalten werden. Durch die Sammlung der Anwendungsfälle werden die Systemgrenze und die Rollen, die die einzelnen Akteure gegenüber dem System einnehmen, festgeschrieben.
Übersichtlich darstellen kann man die gesammelten Use Cases in einem Anwendungsfalldiagramm. (Brügge, et al., 2004)

2.3.4.2. Systemanalyse

In der Systemanalyse müssen aus den bekannten Anwendungsfällen Objekte identifiziert werden. Dazu gibt es mehrere Ansätze, die sich auf unterschiedlichen Sichten auf das System konzentrieren. (Brügge, et al., 2004)

Das Objektmodell
Bei der Objektmodellierung geht es darum, Klassen, Attribute, Methoden und Beziehungen zwischen den Klassen zu identifizieren und durch ein UML-Klassendiagramm darzustellen. Durch Generalisierung und Spezialisierung können Hierarchien in Form von Vererbung dargestellt werden.

Die identifizierten Objekte können noch einmal nach Entitäts-, Steuerungs- und Grenzobjekten gegliedert werden. Grenzobjekte sind jene, die eine Schnittstelle zu den Akteuren des Systems herstellen. Steuerungsobjekte koordinieren Grenz-und Entitätsobjekte. Alle anderen Objekte bilden die Klasse der Entitätsobjekte. Um eine Momentaufnahme des Systems zu modellieren, können die UML Objektdiagramme benutzt werden, welche eine konkrete Instanz eines Klassendiagramms bilden. (Brügge, et al., 2004)

Dynamische Modelle
Dynamische Modelle beschreiben das Verhalten des Systems. Es wird durch Sequenzdiagramme und Zustandsdiagramme repräsentiert. „Sequenzdiagramme repräsentieren die Interaktionen innerhalb eines Satzes von Objekten während eines einzelnen Anwendungsfalls. Zustandsdiagramme repräsentieren das Verhalten eines einzelnen Objekts (oder einer Gruppe sehr eng verbundener Objekte).“(Brügge, et al., 2004) Andere wichtige dynamische Modelle sind die Aktivitätsdiagramme, die Kommunikationsdiagramme und die Zeitdiagramme, auf die an dieser Stelle nicht weiter eingegangen wird. (Brügge, et al., 2004; Matthes, 2009)

2.3.4.3. System- und Architekturentwurf

Gegenüber der Systemanalyse beschreibt der Systementwurf wie das System letztendlich realisiert wird. Diese Realisierung kann noch einmal in mehrere Teilschritte gegliedert werden, welche im Folgenden näher erläutert werden. (Brügge, et al., 2004)

Entwurfsziele

Es müssen zunächst die Entwurfsziele, die im Wesentlichen aus den nicht-funktionalen Anforderungen abgeleitet werden können, definiert werden. Diese sollen vor allem dabei helfen, den Entwickler „[…]bei der Lösung von Zielkonflikten [zu] leiten[…]".(Brügge, et al., 2004)

Systemzerlegung

Im zweiten Schritt der Systemanalyse steht die Zerlegung des Systems in Subsysteme und der Ableitung der Systemarchitektur aus diesen Subsystemen im Fokus. Es müssen dabei „Steuerungsabläufe, Zugriffskontrolle und Datenspeicherung" beachtet werden. (Brügge, et al., 2004) Bei der Zerlegung in Subsysteme ist eine hohe Kohäsion der Komponenten innerhalb eines Subsystems und eine lose Kopplung der Subsysteme zu einander das Ziel. Dabei ist eine hierarchische Zerlegung des Systems in Schichten möglich. Eine Schicht stellt eine Gruppe von Subsystemen dar, die nur auf hierarchisch niedrigere Schichten über Schnittstellen zugreifen können. Bei einer geschlossen Architektur kann nur auf die direkt darunter befindliche Schicht zugegriffen werden. Hingegen kann bei einer offenen Architektur mit allen darunter liegenden Schichten kommuniziert werden.
Dargestellt werden kann die entwickelte Zerlegung zum Beispiel durch Paketdiagramme, Komponentendiagramme, Entwurfs-Klassendiagramme und Verteilungsdiagramme. (Matthes, 2009)

Architekturstile

Da in der Softwareentwicklung Systeme mit ähnlichen Anforderungen entwickelt werden, wurden sogenannte Architekturstile dokumentiert, die bei der Systemzerlegung helfen sollen und eine Art Muster vorgeben. Die wichtigsten hierbei sind der Depot-Architektur-Stil, der Model-View-Control-Stil, Client-Server-Architektur, Peer-to-Peer Stil und die 3 bzw. 4 Säulen Architektur. Es ist dabei auch möglich, mehrere Stile miteinander zu verbinden. (Brügge, et al., 2004; Matthes, 2009)

Sonstige Anforderungen der Systemanalyse

Zusätzlich müssen die Entwickler festlegen, wie das System bei Ausnahmefehlern reagiert, auf welche Weise es terminieren soll, wann persistente Objekte erzeugt und zerstört werden sollen. (Brügge, et al., 2004)

Wiederverwendungskonzept: Patterns

Muster (engl. *patterns*) sind bewährte Lösungsschablonen für wiederkehrende Probleme in Softwarearchitektur und Softwareentwicklung. Sie stellen damit eine wiederverwendbare Vorlage zur Problemlösung dar, die in einem bestimmten Zusammenhang einsetzbar ist. (Quibeldey-Cirkel, 1999)

a) Architektur-Patterns:

Diese Muster beziehen sich auf die strukturellen Eigenschaften eines Systems als Ganzes und beeinflussen so die Architektur des Systems. (Brügge, et al., 2004) Die wichtigsten Architekturmuster wurden bereits im vorherigen Abschnitt Architekturstile genannt.

b) Entwurfsmuster

Entwurfsmuster werden auf einer niedrigeren Abstraktionsebene benutzt und beeinflussen die Architektur von Teilsystemen, wobei sie (meistens) keine Forderung an eine konkrete Programmiersprache stellen. Entwurfsmuster lassen sich gliedern in erzeugende Muster (z.B. Singleton und Abstract Factory Muster), strukturelle Muster (z.B. Adaptermuster, Composite-Muster, Proxy-Muster) und Verhaltensmuster (Observer-Muster, Strategy-Muster).(Quibeldey-Cirkel, 1999)

Es existieren noch viele andere Arten von Mustern wie Idiome, Kommunikationsmuster, Organisationsmuster oder Analysemuster,(Quibeldey-Cirkel, 1999) die an dieser Stelle allerdings nicht weiter ausgeführt werden.

2.3.4.4. Testen

Der Testprozess lässt sich nach (Brügge, et al., 2004) in mehrere Testaktivitäten gliedern. Hier sind ein paar Beispiele:

- Testplanung:
  - Bei der Planung muss ein Zeitplan aufgestellt und verfügbare Ressourcen eingeteilt werden. Es werden Testkonzepte und Testfälle festgelegt und erstellt, die sich an der gewählten Teststrategie orientieren. Die Spezifikation legt dabei das gewünschte Verhalten fest. Die Testergebnisse werden dokumentiert und in einer separaten Fehlerbehebung gegebenenfalls beseitigt.(Brügge, et al., 2004; Matthes, 2009)

Da ein vollständiger Test eines Systems in der Praxis nicht möglich ist, kann man zunächst einmal zwei Gruppen von Testverfahren unterscheiden und zwar den Blackbox-Test (welcher nur die Ausgabe des Systems bezüglich einer gegebenen Eingabe untersucht und dokumentiert) und der Whitebox-Test (die interne Struktur des Systems ist bekannt und wird getestet.) Blackbox-Tests werden auch als funktionsorientierte und White-Box-Tests als strukturorientierte Tests bezeichnet. (Beizer, 1990) Das Ziel der Blackbox-Tests ist es, eine Funktions-, Ausgabe- und Ausnahmeüberdeckung zu erreichen. Das Ziel der Whitebox-Tests ist es, eine Abdeckung des Kontrollflusses zu erzielen. Dies kann mit Hilfe eines Kontrollflussgraphen überprüft werden.(Brügge, et al., 2004)
Nun kann das System nach (Brügge, et al., 2004) schrittweise durch folgende unterschiedliche Testaktivitäten getestet werden:

- Komponententests /Modultest:
  - Dabei werden einzelne Subsysteme unabhängig von anderen Modulen getestet.

- Integrationstests:
  - Die einzelnen entwickelten Komponenten ergeben zusammen das System, welches nun als Ganzes getestet wird (Big Bang Strategie). Ist ein Modultest vorrausgegangen, ist bereits bekannt, dass entstandene Fehler erst durch das Zusammenspiel der Komponenten entstanden sind und nicht einen Modulfehler als Ursache haben. Beim Komponententest können auch nur Gruppen von Subsystemen getestet werden (inkrementelle Bottom-Up Integration). In manchen Fällen müssen Dummies eingesetzt werden, um eine schrittweise Integration zu ermöglichen.

Es gibt noch viele weitere Möglichkeiten der Art der Komponentenzusammensetzung, auf die hier allerdings nicht weiter eingegangen werden kann.

- Systemtests:
  - Bei den Modul-und Komponenten Tests steht vor allem das Finden von Defekten im Vordergrund, wohingegen bei Systemtests die Anforderungen aus der Spezifikation an das System getestet werden sollen. Systemtests kann man nach Funktionstests, Leistungstests, Feldtests, Akzeptanztests und Installationstests gliedern. Dabei kann man noch einmal nach Alpha und Betatests unterscheiden. Diese Tests werden nicht mehr durch das Testteam durchgeführt, sondern durch den Kunden. Bei einem Alphatest testet der Kunde das System in der Entwicklungsumgebung und die Entwickler stehen für eventuelle Fehlerbehebungen zur Verfügung. Beim Betatest testet der Anwender das System in seiner Umgebung.

- Akzeptanztest
  - Der Akzeptanztest ist dann bestanden, wenn die im Vorhinein festgelegten Akzeptanzkriterien erfüllt sind.

3. Entwicklung

3.1. Software Engineering am „Backend for CoMapEd“

Im Abschnitt 1.3 wurde erläutert wie der Software Engineering Prozess in der Theorie stattfindet. Dabei wurden bereits viele Schritte vom Projektmanagement bis hin zum Software Engineering erklärt. Da bei der Entwicklung des „Backend für CoMapEd“ nur ein Entwickler beteiligt ist, ist innerhalb des Entwicklerteams keine Absprache notwendig. Genauso besteht kein Bedarf für einen Projektmanager, Protokollführer, usw.. Alle diese Rollen werden vom Entwickler des Programmes eingenommen. Auch viele der Modelle, die im Abschnitt 2.3 erläutert wurden, sind für dieses Projekt nicht notwendig.

Eine Absprache mit dem Kunden hingegen ist unverzichtbar, da das Programm in Zukunft produktiv eingesetzt werden soll. Da allerdings auch hier die Kommunikation in einem kleinen Kreis von drei Personen stattfand, wurden bei der im Folgenden beschriebenen Projektentwicklung nicht alle Möglichkeiten, die das Software Engineering bietet, voll ausgeschöpft.

3.2. Anforderungsermittlung

3.2.1. nicht funktionale und funktionale Anforderungen

Die Anforderungsermittlung ergibt folgende Anforderungen:

	funktional	nicht funktional
Muss	Datenanalyse	Java GWT
	Exportieren d. Analyse	MySQL
	Exportieren v. gesammelten Informationen	Anpassungsfähigkeit mit geringem Arbeitsaufwand
	Anlegen neuer Projekte	
	Anlegen neuer Erhebungen	
Soll	Datenverwaltung	Hibernate
	Anlegen neuer Labelsets	Schnelle Serveranfragen
	Erstellen neuer Label	intuitive Bedienbarkeit
		Robustheit
Optional	Sprachumstellung Deutsch/ Englisch	Einlogg-Möglichkeit
		Protokoll

Tabelle 2: Anforderungen

Da die funktionalen im Gegensatz zu den non-funktionalen Anforderungen vom Auftraggeber vorgegeben wurden, sind diese nicht im Rahmen des Entwicklungsprozesses vom „Backend for CoMapEd“ ermittelt worden. Deshalb werden im Folgenden nur die nicht-funktionalen Anforderungen näher erläutert.

1) Java GWT: Das Programm muss in Java GWT geschrieben werden, da das Frontend bereits in Java GWT geschrieben wurde. Auf diese Weise sollen die beiden Komponenten besser zusammenarbeiten.
2) MySQL: Das vorhandene Frontend „CoMapEd“ kommuniziert mit einer MySQL Datenbank. Da „Backend for CoMapEd“ auf diese Datensätze zugreifen muss, muss auch die gleiche Datenbank angesprochen werden.
3) Anpassungsfähigkeit mit geringem Arbeitsaufwand: Es ist wichtig, dass der Code übersichtlich, verständlich und leicht zu verändern ist, damit es zukünftig möglich ist, den Quellcode leicht anzupassen, wenn eine Änderung im Frontend dieses erfordert.
4) Hibernate: Der Datenbankzugriff sollte nach Möglichkeit mit Hibernate geregelt werden, da dieses bei den Entwicklern des Frontend und den Nutzern des Backend ein bekanntes Framework ist. Dadurch wird keine Einarbeitungszeit notwendig, falls ein Eingriff in den Code des Backends erforderlich ist.
5) Schnelle Serveranfragen: Da das Programm zur Analyse bestimmt ist, sollten während der Arbeitszeit nur Wartezeiten im Sekundenbereich anfallen, in denen es nicht möglich ist, weiterzuarbeiten.
6) Intuitive Bedienbarkeit: Um den Einarbeitungsaufwand in das Backend möglichst gering zu halten, soll die Bedienbarkeit auch für einen nicht-technisch versierten User intuitiv sein.
7) Robustheit: Da bei längerer Dateneingabe oft ein Tippfehler und somit schnell eine fehlerhafte Eingabe entsteht, sollte das Programm tolerant gegenüber Falscheingaben sein und hierbei nicht abstürzen oder womöglich die Datenbank verändern.
8) Einlogg-Möglichkeit: In der späteren Weiterentwicklung sollten sich die Nutzer des Backends einloggen müssen.
9) Protokoll: Es sollte in der Weiterentwicklung ein Protokoll geführt werden.

3.2.2. Lastenheft / Pflichtenheft / Spezifikation

Neben den Vorgaben von einigen nicht-funktionalen Anforderungen, wird „Backend for CoMapEd“ im Datenbankaufbau durch „CoMapEd“ beschränkt. Das in Abbildung 18 gezeigte Netz ist die bereits vorhandene Datenbankstruktur.

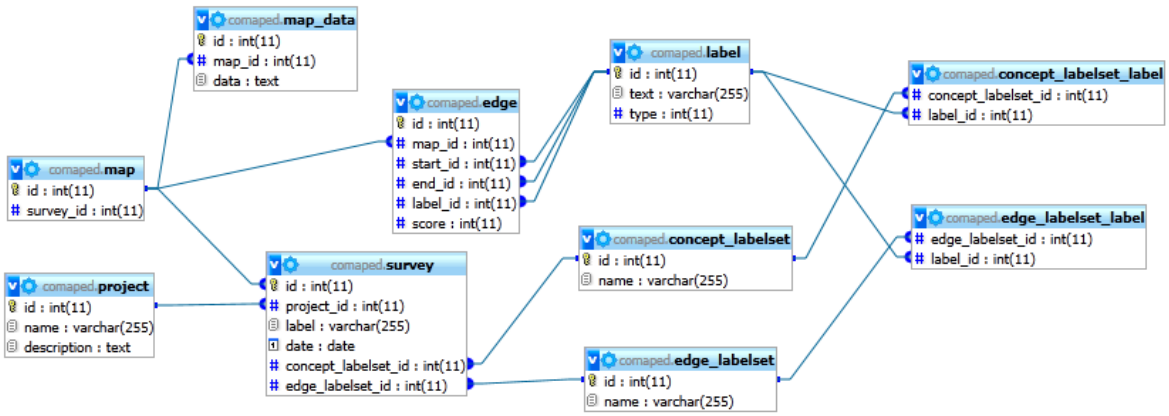


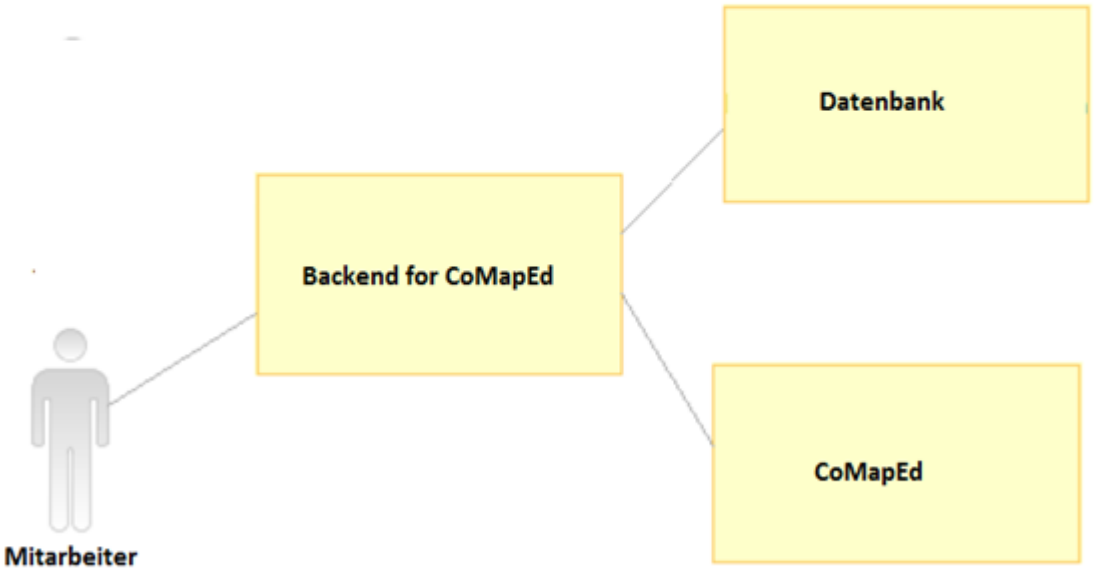
Abbildung 18: Datenbankschema

Aus diesen Anforderungen wurde im Anschluss ein Lastenheft erstellt. Abbildung 19 zeigt lediglich einen kleinen Ausschnitt aus diesem. Hier ist zu erkennen, dass „Backend for CoMapEd“ neben der Kommunikation mit Server und Mitarbeiter auch noch mit „CoMapEd“ kommuniziert. Dieses ist bislang allerdings nur für zukünftige Weiterentwicklung gedacht und wird in den folgenden Implementierungen nicht weiter beachtet.

2. Produkteinsatz

„Backend for CoMapEd“ soll als webbasierte Lösung zum Einsatz kommen, auf das der Mitarbeiter von externen Rechnern über das Internet zugreifen kann, um die Datenverwaltung und Filterung von jedem beliebigen Ort durchführen zu können. Die Zielgruppe von „Backend for CoMapEd“ sind die Verantwortlichen von Erhebungen durch „CoMapEd“.

3. Produktumgebung



- Mitarbeiter: Ein Mitarbeiter ist eine Person, die Zugriff auf die von „CoMapEd“ gesammelten Daten haben soll.

Abbildung 19: Ausschnitt aus Lastenheft

Auf ein Pflichtenheft und Spezifikation wurde im Rahmen dieser Entwicklung verzichtet, da die wichtigen Anforderungen bereits gesammelt wurden und weitere eine schriftliche Dokumentation nicht benötigt wurde

3.3. Analyse / Systementwurf / Objektentwurf

3.3.1. Use Cases und Kernszenarien

In 3.2.1. wurden die Anforderungen an das Programm „CoMapEd – Backend“ beschrieben. Dem zu Folge sind die Datenanalyse und das Exportieren dieser Analyse die wichtigsten Fähigkeiten des Programms. Somit werden diese als Kernszenarien herangezogen, und ausschließlich diese im Use Case-Diagramm Abbildung 20 dargestellt.

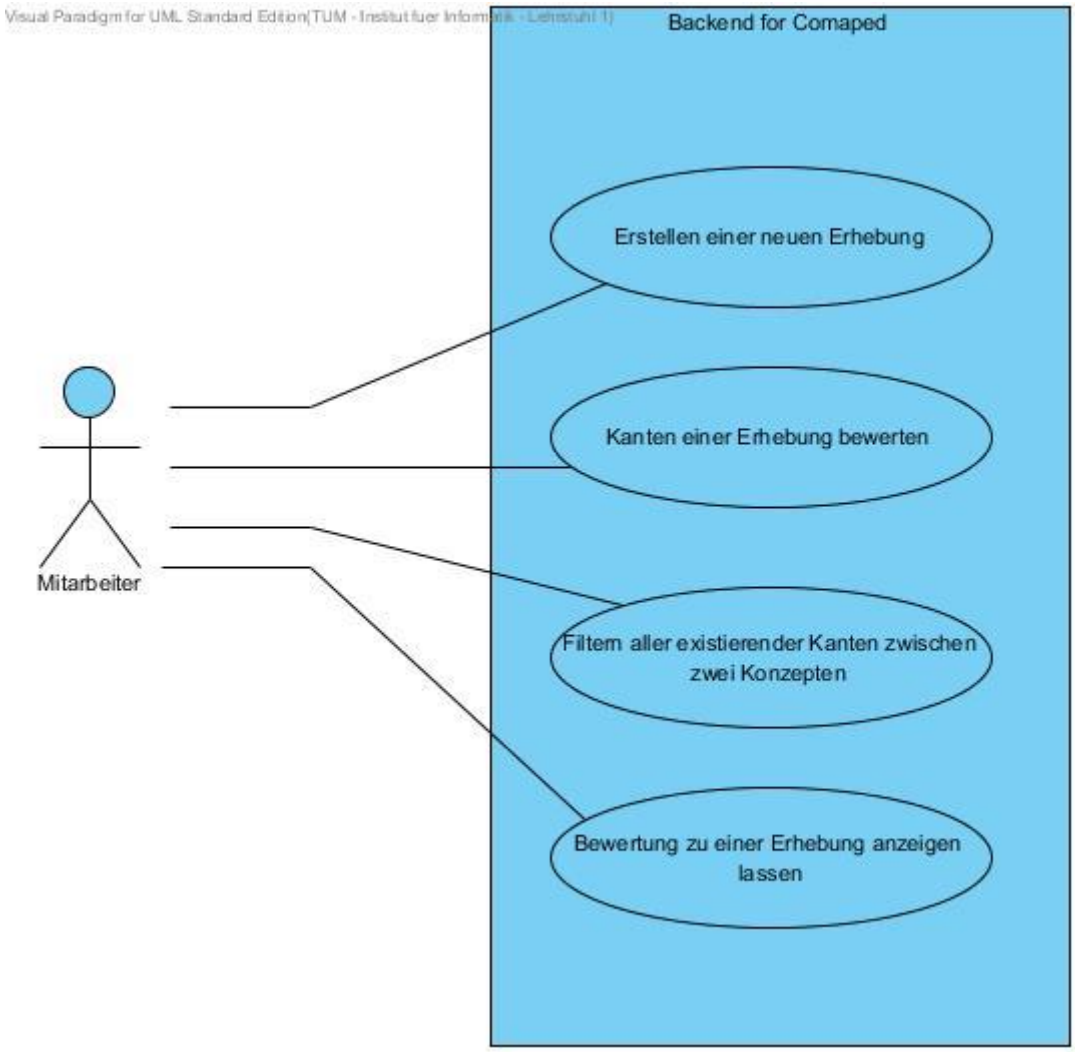


Abbildung 20: Use Case- Diagramm

3.3.2. Klassendiagramm / Sequenzdiagramm

Die Systemanalyse hat zu folgendem Klassendiagramm geführt. Dabei sei hier erwähnt, dass das verwendete Vorgehensmodell dazu geführt hat, dass sich das Klassendiagramm ständig verändert hat. Mehrfaches Refaktorisieren des Codes war notwendig, um den Programmcode übersichtlich zu halten. Im Folgenden wird nur das endgültige Klassendiagramm präsentiert.
Auf Abbildung 21 ist die Struktur des Programms deutlich zu erkennen. Für die Kommunikation zwischen Server und Client wird das RPC Verfahren eingesetzt. Beim RPC (Remote Procedure Call) Modell arbeiten drei Klassen zusammen (rot markiert): ControllerImpl, Controller, ControllerAsync. Diese sorgen für eine asynchrone Kommunikation zwischen Server und Client. [51] Desweiteren ist zu erkennen, dass es eine Schnittstellen-Logikklasse „CoMapEd“ gibt. Diese kommuniziert mit dem Server und gibt anschließend an den GUIcontroller weiter, welche GUI geladen werden muss.

Das GUI Package lässt sich in drei Bereiche unterteilen. GUIcontroller ist die Klasse, die den GUI Bereich kontrolliert und die entsprechenden Klassen anspricht. Sämtliche gui Klassen kennen sich nicht untereinander, sondern nur den GUIcontroller. Der grün umrandete Bereich (Admin,Popup,FEilter) bildet einen Teil des GUI-Bereich des Programms, wobei immer nur einer gleichzeitig, je nach Nutzung des Programms durch den User, aktiv ist. GUIinit wird lediglich für die Initialisierung benötigt. Sie sorgt für den Aufbau der Seite und übergibt die Objekte, die in der späteren Benutzung noch benötigt werden, an GUIcontroller. Der letzte Bereich wird von ButtonsGUI gebildet. Diese Klasse organisiert die zahlreichen Buttons, die vom Programm genutzt werden.

Der blaue Bereich, das Handler Package, beinhaltet die Handler zu den unterschiedlichen Buttons, die benötigt werden.

Außerdem wird auf Basis der Use Cases aus Abbildung 20 und dem Klassendiagramm von Abbildung 21 das auf ABB. gezeigte Sequenzdiagramme abgeleitet.

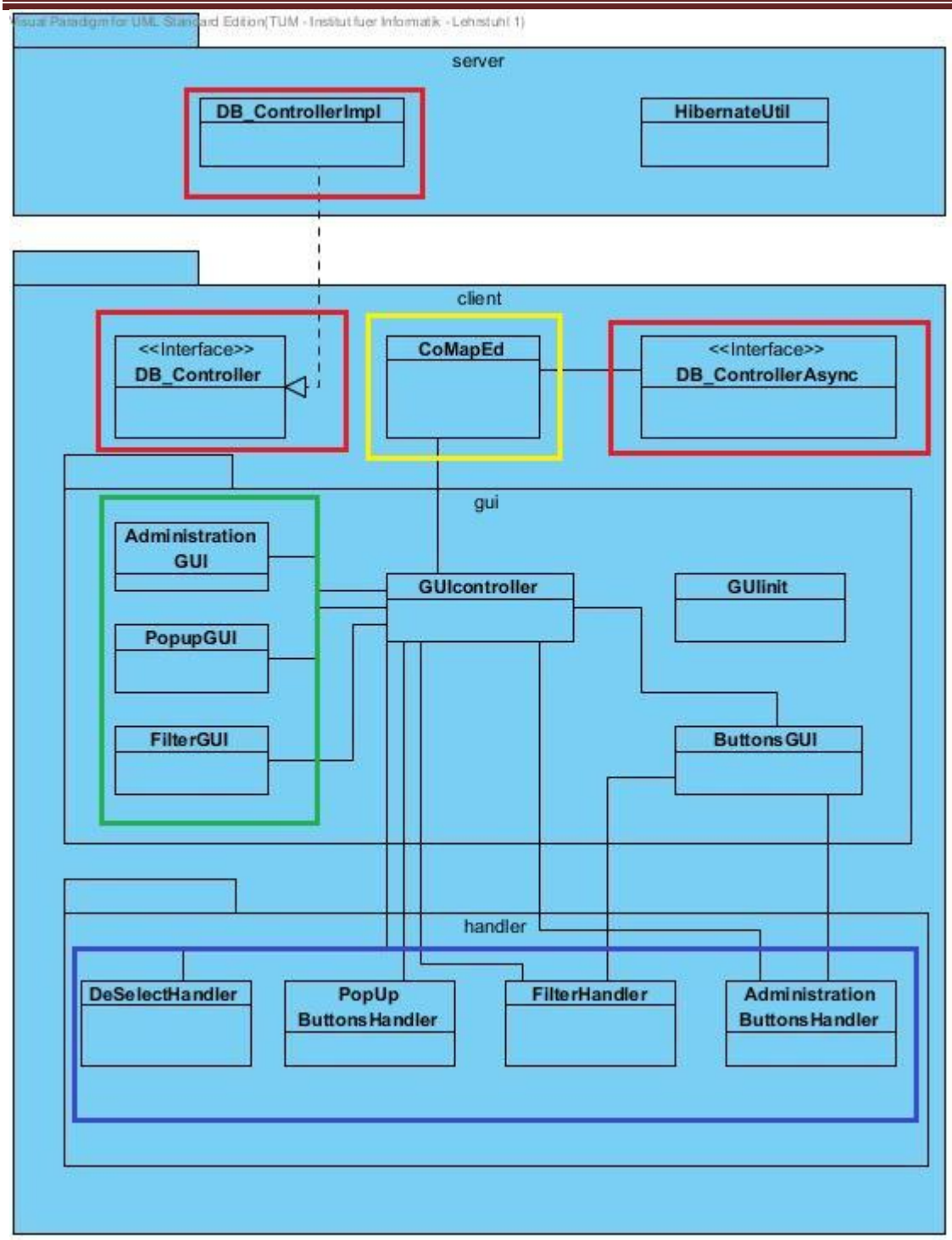


Abbildung 21: konzeptuelles Klassendiagramm

Aus Gründen der Übersichtlichkeit wurde an dieser Stelle ein konzeptuelles Klassendiagramm zur Erläuterung der allgemeinen Struktur gewählt. Im Folgenden wird es noch einmal ein implementierungsnahes Klassendiagramm geben, welches nach Packages aufgeteilt ist, um die Übersichtlichkeit zu erhöhen.

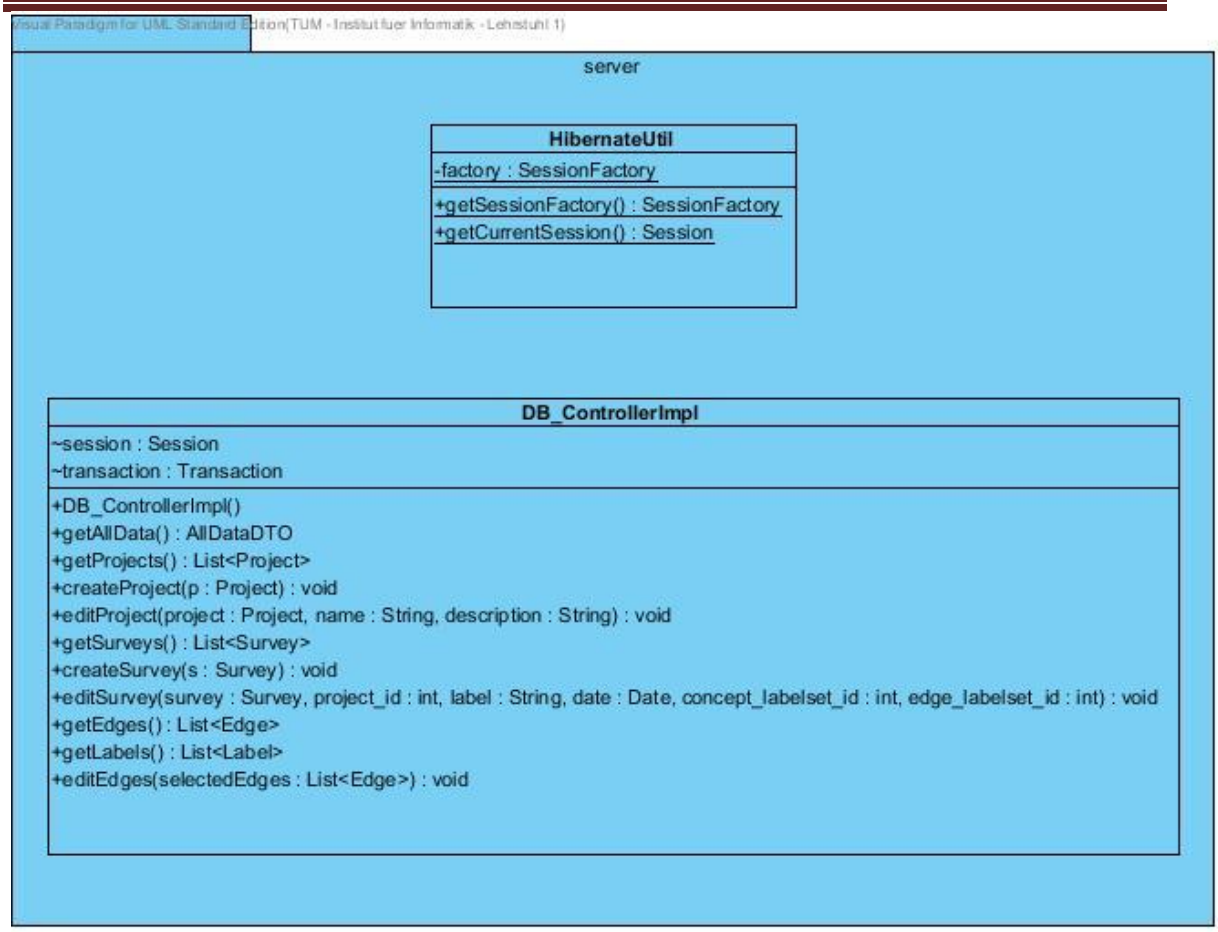


Abbildung 22: Klassendiagramm – Package server

In Abbildung 22, Abbildung 24, Abbildung 25 und Abbildung 26 ist zu erkennen, welche Methoden die einzelnen Klassen mitbringen. Dieses soll das Verständnis des Sequenzdiagrammes aus Abbildung 23 erhöhen.

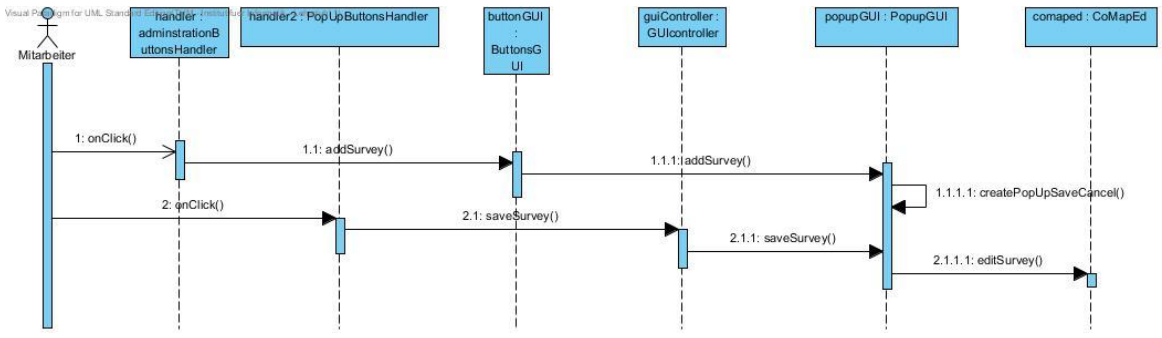


Abbildung 23: Sequenzdiagramm

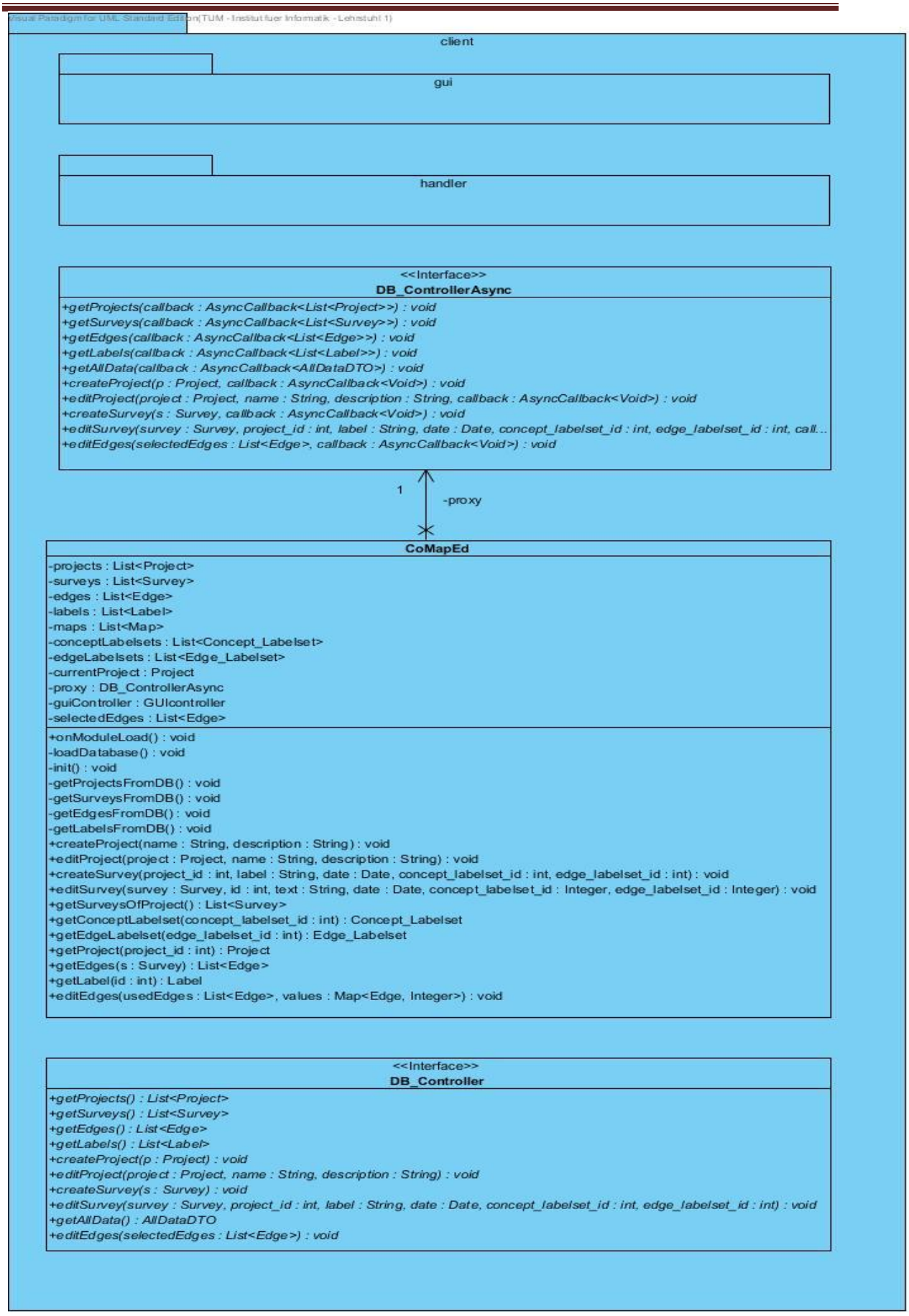


Abbildung 24: Klassendiagramm – Package client

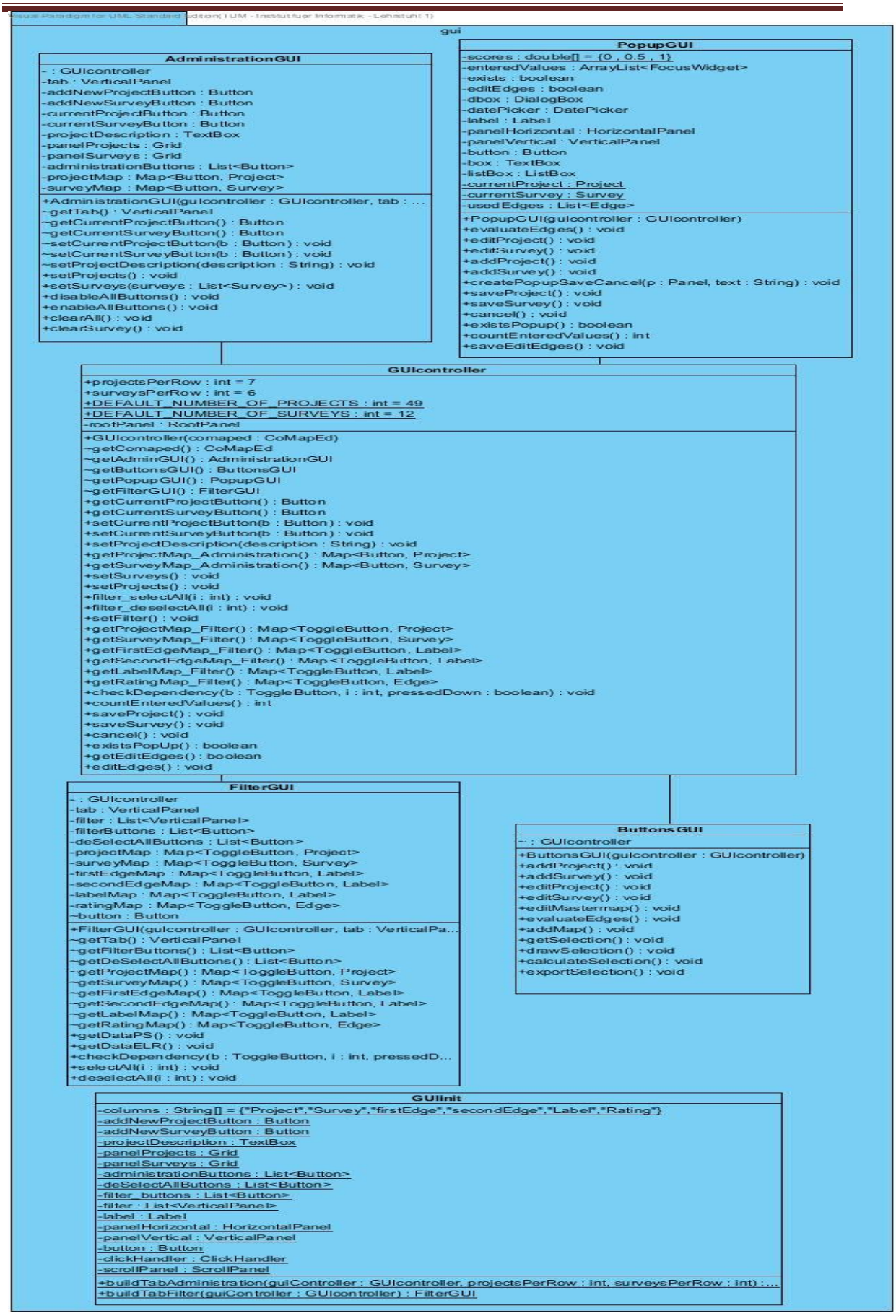


Abbildung 25: Klassendiagramm – Package GUI

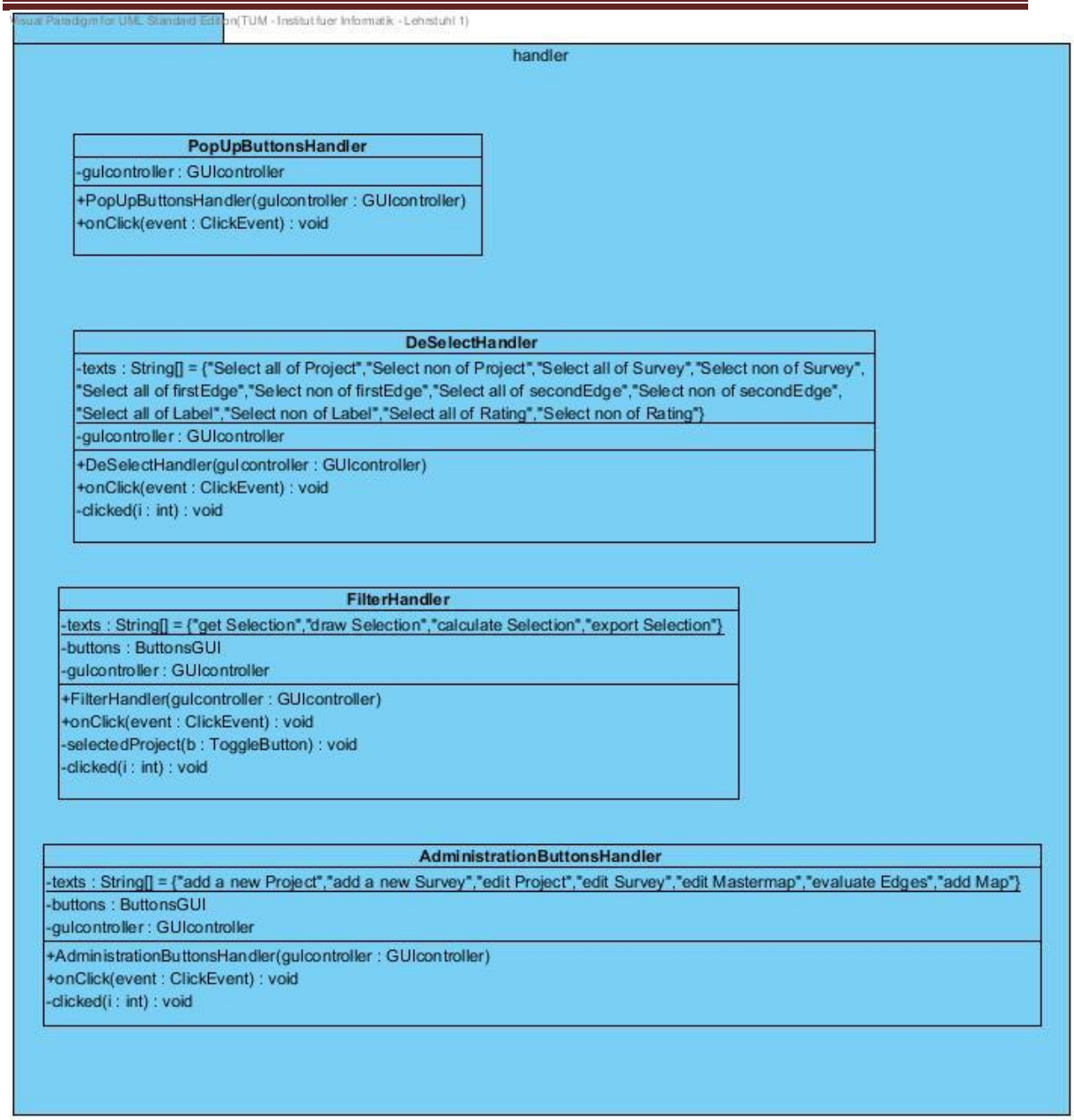


Abbildung 26: Klassendiagramm – Package handler

In Abbildung 23 ist ein Sequenzdiagramm zu dem Szenario der Erhebungserstellung dargestellt. Dieses unterteilt sich in zwei Schritte. Im ersten Schritt wählt der User den Button, der zur Erhebungserstellung genutzt wird. Im zweiten Schritt werden die Daten aus dem in Schritt eins geöffneten Fenster an den Server geschickt.

3.4. Implementierung

Die Implementierung hat viele Fragen und Probleme mit sich gebracht, die im Laufe der Entwicklung aber vollständig gelöst werden konnten. Im Folgenden werden nun einige dieser Probleme benannt und die erarbeitete Lösung erläutert.

Versionskompabilität: Ein entstandenes Problem war die Kompatibilität der GWT- und Hibernate-Version, die nicht für alle Versionskombination gegeben ist. Die Entscheidung ist durch umfangreiche Tests und Recherche auf Hibernate 3 mit GWT 2.3 gefallen. Diese beiden Versionen arbeiten am zuverlässigsten miteinander, so dass es möglich ist, ein funktionierendes Programm zu schreiben. Natürlich ist es denkbar die gewählten Versionen im Nachhinein zu ändern, doch wird es sehr Wahrscheinlich zu Problemen kommen, die zum Beispiel Änderungen verschiedener angebundener Dateien oder des Quellcodes selbst notwendig machen.

Eventbus: Das Frontend „CoMapEd“ arbeitet mit einem Eventbus, auf den die verschiedenen Ereignisse gesendet werden und von den entsprechenden Handlern gelesen werden. „CoMapEd – Backend“ arbeitet allerdings nicht mit einem solchen Bus, da ausschließlich Buttons verwendet wurden, denen man ihren Handler direkt zugeweisen kann, so dass ein solcher Eventbus nicht notwendig ist.

Datenbankschema: Es musste eine Entscheidung getroffen werden, ob es besser ist, eine sehr große Tabelle in der Datenbank zu haben oder eine many-to-many Beziehung einzufügen. Dies ist bei der Zuordnung der Labelsets mit den Labeln und der Edgesets mit den Labeln relevant. Obwohl es inzwischen aufgrund der zur Verfügung stehenden Ressourcen oft vorkommt, dass die Datenbank die Normalform missachtet und sehr große Tabellen führt, anstelle von many-to-many Beziehungen, arbeitet „CoMapEd – Backend“ mit many-to-many Beziehungen. Grundlage für diese Entscheidung war zum einen, dass CoMapEd bereits damit arbeitet und der Mehrwert durch das Umschreiben der Datenbank als zu gering eingeschätzt wird.

Clientseitige vs. Serverseitige Datenselektion: Eine letzte Entscheidung, die in dieser Arbeit erwähnt werden soll, ist ob die Datenselektion auf Server- oder Clientseite durchgeführt werden soll. Dafür ist entscheidend, ob die Nutzung des Programms sich auf eine einzige Funktion beschränkt oder ob viele Funktionalitäten oftmals hintereinander genutzt werden.
Es ist möglich eine Auswahl der angeforderten Daten auf Server- oder Clientseite durchzuführen. Bei einer serverseitigen Selektion werden nur die grade benötigten Daten vom Server an den Client geschickt. Die Selektion auf Clientseite erfordert dagegen das einmalige Senden aller Datensätze vom Server an den Client. Im Anschluss speichert der Client diese Daten und filtert die geforderten Daten heraus. Auf diese Art entsteht ein großer Datenverkehr und somit eine längere Wartezeit zu Beginn einer jeden Programmausführung, welche aber beim weiteren Arbeiten fast vollständig entfällt.
Die Selektion auf Clientseite muss durch in Java geschriebenen Filter durchgeführt werden. Die Datenauswahl auf Serverseite dagegen kann entweder ebenfalls mit in Java geschriebenen Filtern oder über entsprechende SQL-Statements getätigt werden.

Es wurde sich dafür entschieden, dass die Sessions des „Backend for CoMapEd“ hauptsächlich sehr lang sind, weshalb an dieser Stelle eine clientseitige Datenselektion gewählt wurde.

Probleme, die während der Entwicklung des Backends entstanden sind und im weiteren Verlauf gelöst wurden, sind zum Beispiel Versionsprobleme von GWT 2.2 und 2.3

Codewachstum: Da das Programm nach dem inkrementellen Vorgehensmodell entwickelt wurde, wurden die Funktionen nacheinander implementiert. Dabei ging durch den Projektumfang die Übersichtigkeit des Programmcodes verloren. Da die Code-Übersichtlichkeit allerdings eine essentielle Anforderung an das Projekt war, ist eine umfassende Refaktorisierung notwendig gewesen.

Datenbankzugriffe: Auch die Datenbankzugriffe stellten viele Herausforderungen dar. Hier gab es Probleme von Konfigurationseinstellungen, nicht kompatiblen Versionen bis hin zu dem einfachen Problem, dass die Datenbank nicht ohne das Uni SVN erreichbar ist.

3.5. Testing

Die ersten Tests fanden schon während der Entwicklung statt. Dadurch, dass das Programm nach dem inkrementellen Vorgehensmodell entwickelt wurde, testet der Entwickler im Entwicklungsprozess immer wieder die Funktionalität der neu implementierten Funktionen, indem Blackbox Tests zur implementierten Funktion durchgeführt wurden. Nach der Implementierung wurden noch einmal alle Funktionalitäten mit eigens erstellten Daten vom Entwickler getestet.

Nachdem diese Tests erfolgreich waren und angefallene Fehler korrigiert wurden, fand das Testen des „CoMapEd – Backend“ mit neuen Datenbanksätzen statt. Dafür wurde das Programm auf einem Tomcat Server deployed.
Da es nur wenige mögliche Fehlerquellen gibt, wurden diese durch manuelle Ausnahmeeingaben getestet. Als erster Schritt wurde hierfür wieder vom Entwickler selbst getestet, ob mit den Testdaten Probleme auftauchten. Es wurden dabei keine Programmfehler gefunden.

Im Anschluss darauf wurde ein β-Test durchgeführt. Dafür wurden Personen zum Testen herangezogen, für die das Programm fremd war. Sie sollten das Programm nutzen und ebenfalls alle erdenklichen Eingaben austesten um die Robustheit auf die Probe zu stellen. Auf diese Art sollte neben der Funktionalität auch die intuitive Bedienbarkeit getestet werden. Die Benutzer fanden sich sehr schnell in dem Programm zurecht und konnten ebenfalls keine Fehler feststellen.

4. Diskussion des Produkts

4.1. Funktionen des Produkts

„CoMapEd – Backend“ erfüllt alle geforderten Requirements. Es wurde in Englisch implementiert.

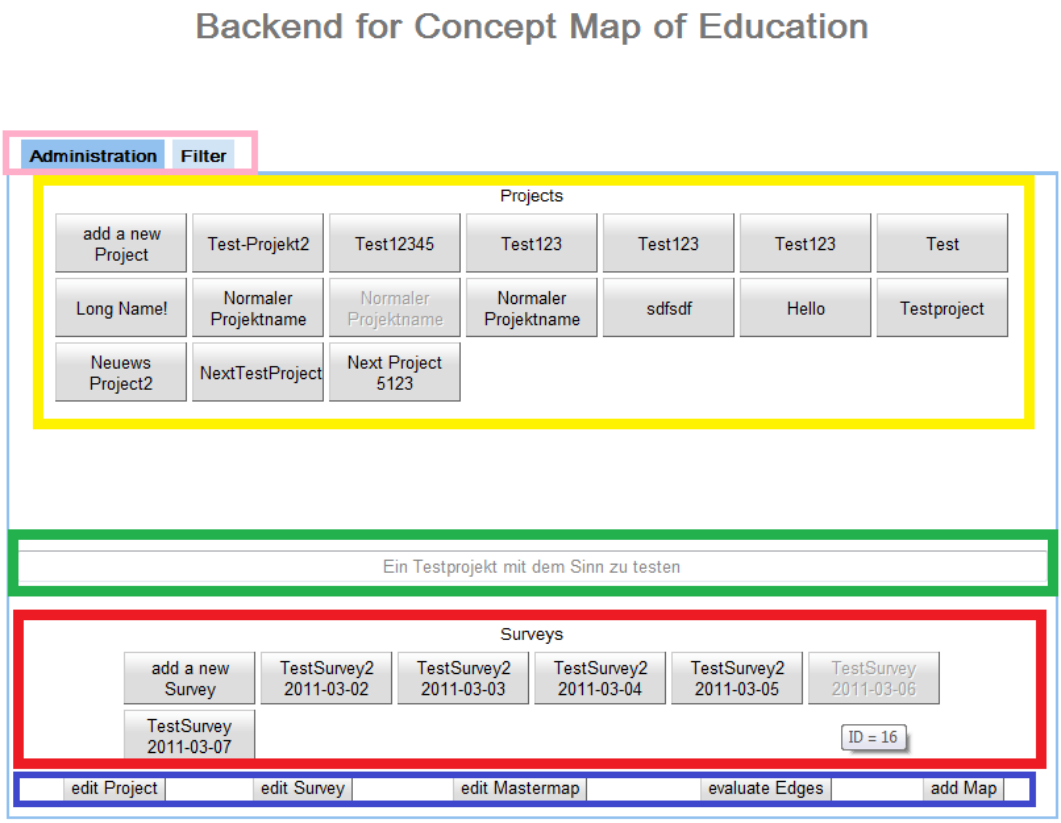


Abbildung 27: „Backend for Comaped“ – Tab Administration

Im rosafarbenen Bereich wählt der User den Tab entsprechend der Funktion, die er nutzen möchte.
Im Tab „Administration“, der auf Abbildung 27 abgebildet ist, wählt er zunächst im gelben Bereich das Projekt, das er betrachten möchte. Die Projektbeschreibung ist dann im grünen Bereich zu sehen, während die dazugehörigen Erhebungen im roten Bereich zu sehen sind. Auch bei den Erhebungen kann er im Anschluss eine Auswahl treffen.
Im blauen Bereich kann nun das ausgewählte Projekt oder die ausgewählte Erhebung bearbeitet werden. Mit „evaluate Edges“ ist es möglich, die in 2.1.3.5 vorgestellte Bewertung für eine Erhebung durchzuführen. „edit Mastermap“ ermöglicht es, die Map auszuwählen, die als Mastermap herangezogen werden soll, wenn nach 2.1.3.2 bewertet werden soll.

Abbildung 28 zeigt dagegen den Tab „Filter“. Im roten Bereich hat der Benutzer zunächst die Möglichkeit, nach Projekten, Erhebungen und Bewertungen zu filtern. Wenn er hier eine Auswahl durch Anklicken der entsprechenden Projekte und Erhebungen oder die rosa markierten Schnellwahltasten getroffen hat, fordert er die entsprechenden Knoten und Kanten durch den gelb markierten Button an.

Anschließend kann noch einmal eine Auswahl der gewollten Kanten und Knoten stattfinden, bevor mit dem grünen Bereich die eigentlichen Funktionen von „CoMapEd – Backend“ durchgeführt werden.

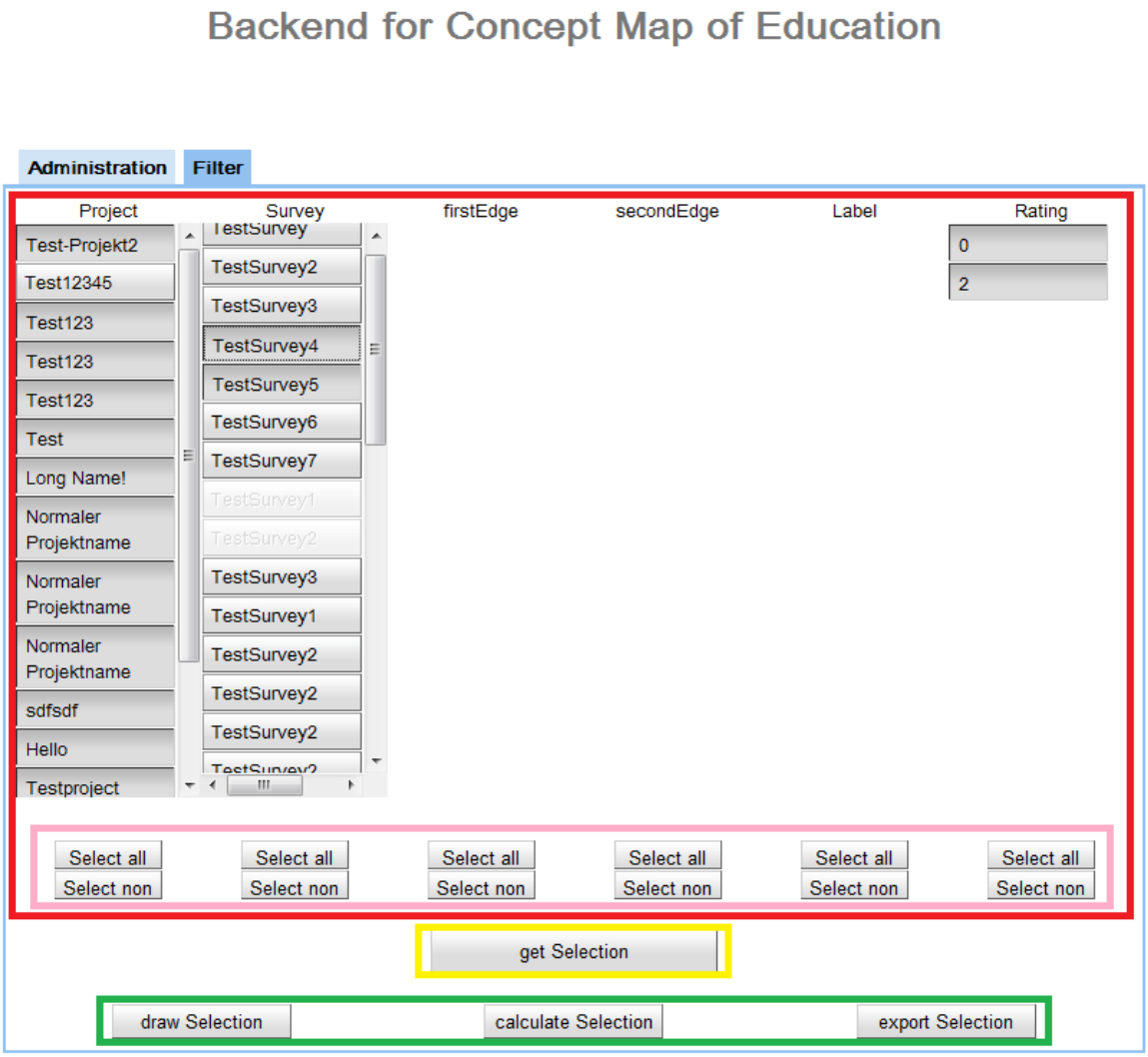


Abbildung 28: „Backend for Comaped“ – Tab Filter

„calculate Selection“ berechnet nach den in 2.3 vorgestellten Verfahren eine Bewertung der Maps. Hierbei ist die Bewertung nach einer Mastermap und die Bewertung nach der Kantengewichtung, die im Tab „Administration“ durchgeführt wurde, implementiert.

Durch Drücken der „export Selection“ -Funktion, wird eine Map exportiert, die nur die ausgewählten Knoten und Kanten beinhaltet.

4.2. Mögliche Weiterentwicklung des Programms

In der Erläuterung im Abschnitt 3.1.1 wurden zwei Buttons, jeweils einer im „Administrations“-Tab und einer im „Filter“-Tab nicht erklärt. Dies liegt daran, dass die Funktion dieser beiden Buttons noch nicht implementiert wurde. Es ist also möglich, zukünftig die Funktion „add Map“ und „draw Selection“ noch zu implementieren. Es soll die Schritte, die ohne diese Buttons dafür notwendig wären, verkürzen. Implementiert wurden sie bisher allerdings noch nicht, da es alternative Wege gibt um Beides durchzuführen.
Eine weitere Funktion, die zukünftig noch implementiert werden könnte, ist die Umstellung der Sprache zwischen Deutsch und Englisch. Dazu sollten die Projekte und Erhebungen am besten einen deutschen und englischen Namen bekommen, um es einheitlich zu halten. Da das Tool aber zur Benutzung vom Fachgebiet Didaktik der Informatik an der TUM entwickelt wurde, wo die Nutzer ein ausreichendes Maß an Englischkenntnissen besitzen, ist es bisher nicht implementiert worden.

Während der Benutzung des Programms wird sich außerdem herausstellen, ob die Annahme aus Kapitel 3.4, dass „CoMapEd – Backend“ hauptsächlich für lange Sessions benutzt wird, korrekt ist. Ansonsten wäre eine Umgestaltung der Datenbankanfragen denkbar.

5. Fazit und Ausblick

Bei der kritischen Betrachtung der in der Einleitung erwähnten Studien rückt immer wieder die Frage nach deren Repräsentativität in den Mittelpunkt. Auch die Objektivität dieser Studien ist oft fragwürdig. Dennoch sind Trends zu erkennen, die es nötig machen objektivere Bewertungsmethoden zu entwickeln, um subjektiven Bewertungenstendenzen entgegenzuwirken. Damit wurde vom Fachgebiet Didaktik in der Informatik an der Technischen Universität München mit dem Werkzeug „CoMapEd“ ein wichtiger Beitrag geleistet. Die Entwicklung von „Backend for CoMapEd“ erweitert diesen Beitrag und sorgt bei Schülern, Studenten und sonstigen Personen durch eine automatisierte Beurteilung für mehr Objektivität.
Auch wenn es durchaus Kritik an Concept Maps als Evaluierungswerkzeug gibt, so wurde deutlich herausgestellt, dass es starke Vorteile bietet. Das Spannungsfeld zwischen Validität und Objektivität / Reliabilität wird wahrscheinlich in einem gewissen Maß immer vorhanden sein, weshalb es auch zukünftig erforderlich ist unterschiedliche Prüfungsmethoden zu benutzen und weiterhin in diese Richtung zu forschen. Ob Concept Maps in naher Zukunft auch in Schulen als Evaluierungswerkzeug eingesetzt werden ist abzuwarten.
Genauso wird sich auch „CoMapEd“ zukünftig beweisen müssen und zeigen dass es ein geeignetes Werkzeug zur Evaluierung ist. Dazu muss es allerdings zunächst fertig gestellt werden und im Anschluss viele Testerhebungen durchlaufen.
Der Funktionsumfang von „Backend for CoMapEd“ bietet ebenfalls noch sehr viele Erweiterungsmöglichkeiten. Hier gibt es neben den bereits vorgestellten noch zu implementierenden Funktionen, die Möglichkeit es als Verwaltung des Frontend zu nutzen. Die Studenten die „CoMapEd“ benutzen könnten hiermit also ebenfalls verwaltet werden. So ist es zum Beispiel vorstellbar, dass im „Backend for CoMapEd“ festgelegt wird, welcher Student welche Knotenmenge für die Erhebung zur Verfügung gestellt bekommt. Daraus würden sich neue Herausforderungen und mögliche weitere Funktionen entwickeln, die dann in Zukunft wieder gelöst werden müssen.

6. Literaturverzeichnis

Albert D. und Steiner C. M. Empirical Validation of Concept maps: Preliminary Methodological Considerations. - [s.l.] : International conference on Advanced leraning Technologies (ICALT'05), 2005.

Albert D. und Steiner C. M. Representing domain knowledge by concept caps: how to validate them?. - [s.l.] : On T. Okamoto, D. Albert, T. Honda & F.W. Hesse (Eds.), The 2nd Joint Workshop of Cognition and Learning through Media-Communication for Advanced e-Learning (pp.169-174) Tokyo, 2005.

Anderson L. W., Krathwohl D. R. und Bloom B. S. A taxonomy for learning, teaching, and assessing: a revsion of Bloom's taxonomy of educational objectives [Buch]. - [s.l.] : Longman, 2000. - 978-0321084057.

Ausubel P.D. The psychology of meaningful verbal learning. - New York : [s.n.], 1963.

Balzert H. Lehrbuch der Softwaretechnik - Basiskonzepte und Requirements Engeneering [Buch]. - Heidelberg : Spektrum, 2009. - 987-3-8274-1705-3.

Balzert H. Lehrbuch der Software-Technik, Bd.1 Software Engeneering [Buch]. - Heidelberg : Spektrum, 2000.

Barenholz H. und Tamir P. A comprehensive use of concept mapping in design, in struction an assessment. Research in Science and Technological Education, 10, 37-52 // 1992.

Beizer B. Software Testing Techniques [Buch]. - New York : Van Nostrand Reinhold Co., 1990.

BLK-Projekt BLK- Projekt Leistungspunktsysteme // Anhang I: Erläuterungen zur Beschreibung und Abstrahierung von intendierten Lernzielen. - Hannover : [s.n.], 2004.

Bonato M. Wissensstrukturierung mittels Struktur-Lage-Technicken. Eine graphentheoretische Analyse von Wissensnetzen. - Frankfurt am Main : [s.n.], 1990.

Brandes U. [et al.] Graph Markup Language (GraphML). - [s.l.] : CRC Press, 2004.

Brügge B. und Dutoit A. H. Objektorientierte Softwaretechnik mit UML, Entwurfsmustern und Java [Buch]. - [s.l.] : Pearson Studium, 2004. - 3-8573-7082-5.

C. Bunse und von Knethen A. Vorgehensmodelle kompakt [Buch]. - [s.l.] : Spektrum Akademischer Verlag, 2008.

Chaudron M. Requirements Engineering. - Leiden : [s.n.], 2008.

Christiansen D. Entwicklung und Erprobung von Aufgaben zur Erfassung zentraler Kompetenzen im Chemieunterrciht am Beispiel Säuren und Basen. - Kiel : [s.n.], 2007.

Deutsches PISA Konsortium Internationales und nationales Rahmenkonzept für ide Erfassung von naturwissenschatlicher Grundbildung in PISA. - 2007.

Diestel R. Graphentheorie [Buch]. - Berlin Heidelberg : Springer, 2006. - 3-540-67656-2.

Eckert A. Die Netwerk-Eloborierungs-Technik (NET) - Ein computerunerstütztes Verfahren zur Diagnose komplexer Wissensstrukturen. - Göttingen : In H. Mandl & F. Fischer (Eds.), Wissen sichtbar machen - Wissensmanagement mit Mapping-Techniken (pp. 138-157), 2000.

Eckert A. Kognition und Wissensdiagnose. Die Entwicklung und empirische Überprüfung des computerunterstützen wissensdiagnostischen instrumentariums Netwerk-Elaborierungs-Technik. - Lengerich : NET, 1998.

Essigkrug A. und Mey T. Rational Unified Process kompakt [Buch]. - [s.l.] : Spektrum, 2007. - 978-3827418364.

Friege G. und Lind G. Begriffsnetze und Expertise.. - [s.l.] : In H. Fischler & J. Peukert (Hrsg.), 2000. - Bde. Concept Mapping in fachdidaktischen Forschungsprojekten der Physik und Chemie Berlin: logos-Verlag 147-178.

Gellert W. [et al.] Kleine Enzyklopädie Mathematik [Buch]. - Leipzig : Bibliographisches Institut Leipzig, 1977. - 387 144 323 9.

Goldsmith T. E. und Davenport D. M. Assessing structiral similarity of graphs. - [s.l.] : In R.W. Schvaneveldt (Ed.), Pathfinder associatove networks: Studies in knowledge organization. (pp. 75-87). Norwood, NJ: Ablex, 1989.

Grande M. 100 Minuten für Anforderungsmanagement - Kompates Wissen nicht nur für Projektleiter und Entwickler [Buch]. - Wiesbaden : Springer, 2011. - 978-3-8348-1431-9.

Hachenberger D. Mathematik für Informatiker [Buch]. - München : Pearson Studium, 2008. - 987-3-8273-7320-5.

Hegarty-Hazel E. und Prosser M. Relationship between students conceptual knowledge and study strategies. Part 1: Student learning in physics. - [s.l.] : Interantional journal of Science Education, 13, 303-312, 1991.

Heinrich G. und Mairon K. Objektorientierte Systemanalyse [Buch]. - [s.l.] : Oldenbourg Wissenschafstverlag GmbH, 2008. - 978-3-486-58366-3.

Hood C. [et al.] Requirments Management - The Interface between Requirements Development and all other Systems Engineering Process [Buch]. - Heidelberg : Springer, 2008. - 987-3-540-47689-4.

Hubwieser P. und Mühling A. Investigating Cognitive Structures of Objective Oriented Programming Courses. - 2011.

Hubwieser P. und Mühling A. Kowpats: Patterns of declarative knowledge // Searching Frequent Knowledge Patterns about Object-Orientation. - 2010.

Ifenthaler D. Diagnose lernabhängiger Veränderung mentaler Modelle // Entwicklung der SMD-Technologie als methodologisches Verfahren zur relationalen, strukturellen und semantischen Analyse individueller Modellkonstruktionen. - Müllheim : [s.n.], 2006.

Klieme E. Was sind Kompetenzen und wie lassen sie sich messen. - [s.l.] : Auszug aus Pädagogik 6, S.10-13, 2004.

Leiserson C., Rivest R. L. und Stein C. Algorithmen - Eine Einführung [Buch]. - München : Oldenbourg Wissenschaftsverlag, 2007. - 987-3-486-59002-9.

Mandl H. und Fischer F. Mapping-Techniken und Begriffsnetze in Lern- und Kooperationsprozessen // Wissen sichtbar machen - Wissensmanagement mit Mapping-Techniken (pp.3-12). - Göttingen : [s.n.], 2000.

Matthes F. Einführung in die Softwaretechnik. - München : [s.n.], 2009.

McClure J. R. und Bell B. E. Effects of an environmental education related STS approach instruction on cognitive structures of pre-service science teachers. - [s.l.] : university Park, PA: Pennsylvania State University. (ERIC Document Reproduction Service No. ED 341 582), 1990.

McClure J. R., Sonak B. und Suen H. K. Concept Map Assessment of Classroom learning: Reliability, Validity, and Logistical Practicality. - [s.l.] : Journal of Research in science Teaching, 1998. - Bde. Vol 36, No. 4, PP.475-492 (1999).

Mühling A. What Students (should) know about Object Oriented Programming. - 2011.

Norman D. A. 'Some observations on mental models. - Hillsdale : in Mental models: eds. D. Gentner &A.L. Stevens, Lawrence Erlbaum Associates pp. 7-14, 1983.

Novak J. D. und Canas A. J. The Theory Underlying Concept Maps and How to Construct and Use Them. - 2008.

Novak J. D. und Gowin D. B. Learning how to lern. - New York : Cambridge University Press, 1984.

Pirsig R. M. Lila oder ein Versuch der Moral [Buch]. - 2006.

Quibeldey-Cirkel K. Entwurfsmuster - Design Patterns in der objektorientierten Softwaretechnik [Buch]. - Heidelberg : Springer, 1999. - 3-540-65825-4.

Riuz-Primo M. A. und Shavelson R. J. Problems and Issues in the Use of Concept Maps in Science Assessment. - [s.l.] : Journal of Research in Science Teaching Vol.33, No.6, S.569-600, 1996.

Saake G. und Sattler K.-U. Algorithmen & Datenstrukturen: Eine Einführung mit Java 2 [Buch]. - Heidelberg : dpunkt verlag, 2004.

Schatten A. [et al.] Best Practice Software-Engineering - Eine praciserprobte Zusammenstellung von komponentenorientireten Konzepten, Methoden und Werkzeugen [Buch]. - Heidelberg : Spektrum, 2010. - 978-3-8274-2486-0.

Schienmann B. Kontinuierliches Anforderungsmanagement - Prozesse - Techniken - Werkzeuge [Buch]. - München : Addison-Wesley, 2002. - 3-8273-1787-8.

Sommerville I. und Sawyer P. Requirements Engineering - A good practice guide. - [s.l.] : Wiley, 1997.

Steiner C. M. und Albert D. Investigating application validity of Concept Maps. - 2008.

Stracke I. Einsatz computerbasierter Concept Maps zur Wissensdiagnose in der Chemie - Empirische Untersuchungen am beispiel des Chemischen Gleichgewichts [Buch]. - Kiel : Waxmann, 2003. - 3-8309-1362-1.

Surber J. R. und Smith P. L. Testing for misunderstanding. - [s.l.] : Educational Psychologist, 16, 163-174, 1981.

Trowbridge J. E. und Wandersee J. H. The concept map as a research tool: exploring conceptual change in biology. - [s.l.] : Journal of Research in science Teaching, 27, 1033-1052, 1990.

Vanides J. [et al.] Using Concept Maps in the Science Classroom. - [s.l.] : National Science Teachers Association (NSTA), 2005.

Wackersreuther B. Dynamische Mustersuche in Biologischen Netzwerken. - 2007.

Weinert F. E. Leistungsmessungen in Schulen [Buch]. - [s.l.] : Beltz, 2002. - 3-407-25256-0.

Weyde T. und Wissmann J. Dynamic Concepts Maps for Music. - 2004.

Wieczorrek H. W. und Mertens P. Management von IT-Projekten [Buch]. - Heidelberg : Springer, 2011. - 987-3-642-16126-1.

yWorks Files for Java Developer's Guide - Chapter 9. Input and Output TGF [http://docs.yworks.com/yfiles/doc/developers-guide/tgf.html]. - Betrachtungsdatem: 14.01.2012.

7. Abbilungs- / Formel- / Tabellenverzeichnis

7.1. Abbildungsverzeichnis

7.2. Tabellenverzeichnis

7.3. Formelverzeichnis

Design und Implementierung eines Backends für ein Online-Werkzeug zur Erhebung von Concept Maps

eine Arbeit von Manuel Schmidt