Entwicklung einer Web Anwendung für die Erstellung von Metriken komplexer Netzwerke

Michael Seitlinger

Entwicklung einer Web Anwendung für die Erstellung von Metriken komplexer Netzwerke

Masterarbeit

an der

Technischen Universität Graz

vorgelegt von

Michael Seitlinger

Institut für Wissensmanagement (IWM),
Technische Universität Graz
A-8010 Graz

30. November 2011

Diese Arbeit ist in deutscher Sprache verfasst.

Begutachter: Ass.Prof. Dipl.-Ing. Dr.techn. Univ.-Doz. Denis Helic

Development of a Web Application for Metric-Based Analysis of Complex Networks

Master's Thesis

at

Graz University of Technology

submitted by

Michael Seitlinger

Institute of Knowledge Management (KMI),
Graz University of Technology
A-8010 Graz, Austria

30th November 2011

Advisor: Ass.Prof. Dipl.-Ing. Dr.techn. Univ.-Doz. Denis Helic

TU Graz

Kurzfassung

Vernetzte Strukturen sind in unserer modernen Gesellschaft, wie unmittelbar am Beispiel des World Wide Web (WWW) zu sehen ist, allgegenwärtig. Ein Netzwerk kann definiert werden als ein einfaches Muster, welches auf abstrakte Weise beschreibt, wie Dinge in einem System miteinander in Verbindung stehen. Ein wichtiger Aspekt ist der Konnex, dass ein Netzwerk immer mit Phänomenen der wirklichen Welt in Beziehung steht. Die Netzwerktheorie liefert die Basis für die wissenschaftliche Untersuchung von Netzwerken und ist ein interdisziplinäres Forschungsgebiet, da Netzwerke in den unterschiedlichsten Domänen auftreten. Die Herausforderung dieser Masterarbeit stellte sich mit der Aufgabe, einen einfachen Zugang zu den Analyse-Methoden der Theorie der Netzwerkanalyse zu bilden. Das Ziel dieser Arbeit lag daher in der Konzeption einer Web-Anwendung für die Analyse von Netzwerkdaten, die einfach bedienbar ist und als eine Lernumgebung im Rahmen einer Lehrveranstaltung oder auch im wissenschaftlichen Betrieb verwendbar sein soll. Aspekte des Web Engineerings und der Usability bilden neben der Netzwerkanalyse die theoretische Grundlage dieser Arbeit. Im Rahmen der praktischen Arbeit wurden Anforderungen erarbeitet, die bei der Erstellung der Architektur der Web Anwendung als Einflussfaktoren berücksichtigt wurden. Die Web-Anwendung wurde mit dem komponenten-basierten Web Framework Apache Wicket und mit der Programmiersprache Java umgesetzt. Auch wurden gegenwärtige Web Standards und Techniken, wie XML-Schema Definitionen, Ajax oder Polling-Methoden eingesetzt. Ein Hauptaugenmerk dieser Arbeit lag zudem auf dem Qualitätsfaktor Usability sowie der didaktischen Plausibiltät der Web-Anwendung. Anhand einer informalen Evaluierung der Web-Anwendung durch Studierende, Pädagogen und Experten erfolgte die Validierung dieser beiden Aspekte. Daraus abgeleitete Maßnahmen bilden als Hilfen zur Verbesserung des Systems Network Metric Explorer den Schlussteil dieser Masterarbeit.

Abstract

Networks - are omnipresent in our modern society, as is shown by the world wide web. A network can be defined as a simple pattern which describes in an abstract manner how things are interconnected within a system. One important aspect is, that a network always interrelates to a real world phenomenon. For the scientific examination of networks, network theory is used, which is an interdisciplinary research field, as networks appear in a variety of domains. The objective of this master thesis is to develop an easy access to the analytical methods of the theory of network analysis. The goal of the thesis was the application of a web tool for the analysis of data, which is easy to employ and can be used in a learning environment for example in a university lecture or in an enterprise dedicated to research. Aspects of web engineering and usability are the foundation of the theoretical framework of this thesis. The practical part of the thesis is influenced by several influential factors. The web tool was made with the web framework Apache Wicket and the programming language Java. Web standards and techniques such as XML schema definitions, Ajax and polling methods were also used. A main aspect was the quality factor usability, which was transferred with an informal evaluation through students, experts in the field of education and network theory. Some solutions regarding the results of the informal evaluation have been suggested for the further development of the web application Network Metric Explorer and they are the final part of this master thesis.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Place	Date	Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Ort	Datum	Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Danksagung

Ich möchte an dieser Stelle einen herzlichen Dank an Prof. Denis Helic aussprechen, den ich als einen konstruktiven Mentor und Betreuer kennenlernen durfte. Dr. Markus Strohmaier, Fabian Rampetsreiter, Franjo Bratic danke ich für die konstruktive Kritik und Zeit, die sie für die informale Evaluierung der Web-Anwendung aufgebracht haben. Ein Dank gebührt auch Dipl.-Ing. Georg Sauseng für die Bereitstellung der Server-Infrastruktur. Prof. Keith Andrews danke ich für die LaTeX Vorlage dieser Masterarbeit.

Es ist mir auch ein besonderes Anliegen mich für jegliche Unterstützung, die ich nicht nur im Laufe meines Studiums erfahren habe, bei meiner wundervollen Freundin Tamara, meinen Eltern Elisabeth und Horst, meinem Bruder Kevin, bei Dagmar und Tom als auch bei jeden meiner Freunde herzlich zu bedanken.

Michael Seitlinger
Graz, November 2011

Kapitel 1

Einleitung

1.1 Motivation

Vernetzte Strukturen sind in unserer modernen Gesellschaft allgegenwärtig. Ein Netzwerk kann definiert werden als ein einfaches Muster, welches auf abstrakte Weise beschreibt, wie Dinge in einem System miteinander in Verbindung stehen. Ein wichtiger Aspekt ist der Konnex, dass ein Netzwerk immer mit Phänomenen der wirklichen Welt in Beziehung steht. Die Netzwerktheorie liefert die Basis für die wissenschaftliche Untersuchung von Netzwerken und ist ein interdisziplinäres Forschungsgebiet, da Netzwerke in den unterschiedlichsten Domänen auftreten. Diese wissenschaftliche Untersuchung von Netzwerken ist eine bis dato sehr junge Disziplin und befasst sich mit Netzwerken der realen Welt und deren Phänomenen [Watts, 2004, Seite 13]:

> *"In the quiet corridors of academia, meanwhile, a new science has been emerging –one that speaks directly to the momentous events going around it. For want of a better term, we call this new science the* science of networks. *And unlike the physics of subatomic particles or the large-scale structure of the universe, the science of networks is the science of the real world –the world of people, friendships, rumors, disease, fads, firms, and financial crisis."*

Soziale Netzwerke, das World Wide Web (WWW), Straßennetze aber auch Stoffwechselnetzwerke sind nur einige Beispiele für Netzwerke, die im Fokus einer wissenschaftlichen Untersuchung stehen. In vielen Netzwerk treten Phänomene auf, die es erst zu erforschen gilt [Berners-Lee et al., 2006, Seite 12]:

> *"Various aspects of the Web are relatively well-understood, and as an engineered artefact its building blocks are crafted, not natural phenomena. Nevertheless, as the Web has grown in complexity and the number and types of interactions that take place have ballooned, it remains the case that we know more about some complex natural phenomena (the obvious example is the human genome) than we do about this particular engineered one."*

Mit steigender Komplexität kann ein Netzwerk nicht mehr rein aus seinen einzelnen Bausteinen beschrieben werden. Dadurch ist es auch nicht möglich resultierende Phänomene ohne empirische Untersuchungen zu erklären. Im Gegensatz zu einem Graphen ist ein Netzwerk kein statisches Artefakt, sondern ein dynamisches System, dass *etwas macht* und das über die Zeit Eigenschaften hervorbringen oder ändern kann. Das macht die Analyse von Netzwerken –im Sinne von *komplexen Systemen*– zu einem nicht trivialen Unterfangen.

Sehr viele Netzwerke gehorchen *überraschenderweise* gemeinsamen Organisationsprinzipien, weisen bestimmte Eigenschaften auf, oder haben bestimmte typische Merkmale (vgl. [Newman, 2010], [Barabási, 2003]). Dieser Umstand verbindet die einzelnen Wissenschaftsdisziplinen und begründet auch

die Notwendigkeit einer gemeinsamen Theorie über Netzwerke. Gleichermaßen bietet die Netzwerketheorie etablierte Techniken und Werkzeuge, mit denen die einfache Analyse von realen Netzwerken –welche in geeigneter Form repräsentiert sind– sofort Anwendung finden können. Die Herausforderung stellt sich mit der Aufgabe, einen einfache Zugang zu diesen Analyse-Methoden zu bilden.

1.2 Zielsetzung

Das Ziel dieser Arbeit liegt in der Konzeption einer Web-Anwendung für die Analyse von Netzwerkdaten sowie einer *den zeitlichen Ressourcen* entsprechenden Umsetzung. Die Web-Anwendung soll ein Plattform darstellen, auf dieser man durch eine möglichst einfache Art und Weise einen Zugang zu etablierten Netzwerk Analyse-Methoden bekommt und diese *interaktiv* anwenden kann. Im Vordergrund steht die Erstellung von Netzwerk-Metriken aus Netzwerkdatensätzen. Durch eine geeignete und generische Architektur soll das System sehr einfach erweiterbar sein, damit man zu einem späteren Zeitpunkt möglichst beliebige Analyse-Module in das System einbetten kann.

Die resultierende Web-Anwendung soll schlussendlich unterstützend für eine Lehrveranstaltung eingesetzt werden können, wodurch sich die Anforderung ableitet, dass die Zielgruppe des Systems keinen Expertenstatus haben muss. Es soll eine erweiterbare Lernumgebung geschaffen werden, mit deren Hilfe ein leichter Zugang zu der komplexen Thematik der Netwerkanalyse ermöglicht wird.

Neben der Einhaltung und der Verwendung von Web Standards (*technische Umsetzung*), stellt sich die Herausforderung eine Lernumgebung zu schaffen. Diese Arbeit verfolgt nicht das Ziel E-Learning Standards zu erfüllen, sondern folgt der DIY (*Do It Yourself*) Attitüde der *EduPunk* Bewegung, mit der Einsicht, dass Technologie Bildung alleine noch nicht wertvoll macht – sondern es vielmehr darum geht, entsprechende Umgebungen zu schaffen, in der die Interessen der Lernenden geweckt werden sollen. Dieses Ziel kann durch den vernünftigen Einsatz von Web-Technologie unterstützend erreicht werden. Eine Voraussetzung ist unter anderem, dass das resultierende System *useable* gestaltet wird, daher werden Aspekte der Web-Usability in dieser Arbeit ebenfalls berücksichtigt.

Die Web-Anwendung wird in einer abschließenden Evaluierung durch eine Zielgruppe bestehend aus Experten, Studenten und Pädagogen bewertet. Durch diese Feedback-Schleife soll eine Bewertung der resultierenden Qualität der Web-Anwendung erhalten und geeignete Maßnahmen abgeleitet werden können, die helfen sollen, das System in weiteren Iterationszyklen zu verbessern.

1.3 Struktur der Arbeit

Der erste Teil dieser Arbeit (Kapitel 2 bis 4) gibt einen Überblick über relevante theoretischen Aspekte, die für diese Arbeit von Bedeutung sind.

Kapitel 2 illustriert die wissenschaftlichen Untersuchung von Netzwerken. Es wird eine Übersicht über die grundlegende Theorie der Netzwerkanalyse geboten. Dieser Teil umfasst einerseits relevante Definitionen aus der Netzwerktheorie und Graphentheorie, bis hin zu einzelnen Merkmalen und Eigenschaften von Netzwerken. Zur besseren Veranschaulichung werden populäre Beispiele von Netzwerken aus der Literatur angeführt. Abschließend wird das Analysewerkzeug *Stanford Network Analysis Pack (SNAP)* vorgestellt, dass eine Software Bibliothek ist, die für die Analyse von Netzwerk Eigenschaften verwendet werden kann.

Kapitel 3 befasst sich mit der Disziplin Web Engineering. Es werden Aspekte der systematischen Entwicklung einer Web-Anwendung behandelt. Neben einer einführenden Begriffsdefinitionen werden Un-

terschiede zur klassischen Disziplin Software Engineering behandelt. Die Herausforderungen die sich bei der Erstellung und Umsetzung von modernen Web-Anwendungen ergeben, werden ebenso thematisiert, wie die Software Architektur von Web-Anwendungen. Dieses Kapitel umfasst schließlich auch die Beschreibung relevanter Entwurfsmuster, wie beispielsweise das Model-View-Controller Pattern.

Kapitel 4 gibt eine Überblick über die Usability von Web Anwendungen. Neben Gestaltungsrichtlinien werden Methoden des Usability Engineering beschrieben. Usability wird als ein Qualitätsmerkmal von Web Anwendungen diskutiert, dass sich durch geeignete Maßnahmen messen lässt. In einem Abschnitt werden die wichtigsten Methoden zur Evaluation der Usability erwähnt. Ein weiterer Abschnitt befasst sich mit der *EduPunk* Bewegung, die gegen vorhandene E-Learning Standards und deren Kommerzialisierung rebelliert und die Schaffung von geeigneten Lernumgebungen sowie den Lernenden ins Zentrum rückt.

Der zweite Teil dieser Arbeit (Kapitel 5 bis 8) beschreibt alle für die technische Umsetzung der Web-Anwendung relevanten Aspekte und die Ergebnisse der Evaluierung der Web-Anwendung hinsichtlich Usability Aspekten.

Alle Anforderungen die für die Erstellung der Web-Anwendung erarbeitet wurden, werden in Kapitel 5 beschrieben. Es wird auch argumentiert, nach welchen Aspekten ein geeignetes Framework für die Erstellung einer Web-Anwendung ausgewählt wurde.

Das erstellte System wird letztendlich in Kapitel 6 präsentiert. Dazu werden relevante Aspekte bezüglich Entwurf und Implementierung der Web-Anwendung aufgezeigt. Dieses Kapitel beinhaltet neben der erarbeiteten System-Architektur auch relevante und ausgesuchte Implementierungs-Details. Zusätzlich werden verwendete Werkzeuge und Frameworks skizziert, die für die Systemrealisierung verwendet wurden.

In Kapitel 7 werden schließlich die Ergebnisse der informalen Usability Untersuchung dargelegt, wo eine Gruppe aus Experten, Pädagogen und Studenten das System hinsichtlich Usability Kriterien untersucht hat. Anhand der Ergebnissen dieser Evaluierung ist es abschließend möglich Maßnahmen abzuleiten, die helfen das System in weiteren Projekten zu erweitern und zu verbessern.

Kapitel 8 diskutiert Trends –wie beispielsweise der Einsatz von HTML5–, beschreibt den Stand der Arbeit und sammelt Ideen und Maßnahmen, die für die Verbesserung des umgesetzten Systems verwendet werden können.

Abschließend liefert Kapitel 9 eine kurze Zusammenfassung über Erkenntnisse, die im Rahmen dieser Masterarbeit gewonnen werden konnten.

Kapitel 2

Netzwerkanalyse

“ Das Ganze ist mehr als die Summe seiner Teile. ”

[Aristoteles (384 - 322 v. Chr.)]

“ How does individual behavior aggregate to collective behaviour?”

[Watts [2004]]

Dieses Kapitel präsentiert einen Überblick über relevante Literatur für die Domäne der Netzwerkanalyse. Die einzelnen Abschnitte geben eine kurze Einführung in die Phänomenologie der komplexen Netzwerke und der Netzwerkanalyse und heben wichtige Merkmale und Metriken von Netzwerken hervor.

Das Kaptitel beinhaltet abschließend einen zusätzlichen Abschnitt über das *Stanford Network Analysis Package (SNAP)*. Das *Stanford Network Analysis Package (SNAP)* ist ein Tool mit dem man Netzwerke analysieren kann. Dieses Tool hat zudem sehr viele theoretische Aspekte der Netzwerkanalyse implementiert, die auch in den ersten Abschnitten dieses Kapitels beschrieben werden.

2.1 Netzwerke sind allgegenwärtig

Die wissenschaftliche Untersuchung von Netzwerken (vgl. [Newman, 2010], [Newman, 2003], [Watts, 2004]) ist ein *interdisziplinäres* Forschungsgebiet, dass viele Errungenschaften aus einzelnen Disziplinen wie Physik, Mathematik, Informatik, Biologie, Soziologie, Epidemiologie, Klimaforschung und vielen weiteren mit einander verbindet. Das junge Fachgebiet hat durch die mannigfaltigen Einflüsse, Ideen und durch die *empirischen* Untersuchungen von Real-Life-Netzwerken [Easley und Kleinberg, 2010] der unterschiedlichsten Forschungsdisziplinen enorm profitiert.

Bestimmte Techniken und Modelle konnten dadurch entwickelt werden, die für die Analyse oder Prognose über das Verhalten vernetzter Systeme [Newman, 2003] sehr hilfreich sind. Newman [2010] konstatiert neben diesem Vorteil auch einen Nachteil, der die Streuung des generierten Wissens über Netzwerke in die einzelnen Fachdisziplinen betrifft, wodurch der Zugang zu Entdeckungen und Errungenschaften aus anderen Fachgebieten oftmals schwierig ist.

Ein Grund warum Netzwerke in so vielen Disziplinen als Untersuchungsobjekte zu finden sind, ist die Tatsache, dass vernetzte Strukturen allgegenwärtig sind [Barabási und Bonabeau, 2004].

Analyse mittels Visualisierung Vernetzte Strukturen sind nicht immer intuitiv wahrnehmbar oder sichtbar. Oftmals ist es notwendig, mittels einer geeigneten *Visualisierung*, Netzwerke in eine graphische und visuell erfassbare Form zu bringen, damit ein Verständnis darüber entwickelt werden kann, wie die Elemente eines Systems miteinander in Verbindung stehen und interagieren.

Die Nervenzellen des Gehirns –die Neuronen– sind durch spezielle Verbindungen –die Synapsen– miteinander verbunden. Aus heutiger Sicht ist dieser Fakt nichts neues, jedoch war die Existenz einer solchen *vernetzten* Struktur lange Zeit nicht bekannt. Der spanischer Mediziner Santiago Ramón y Cajal[1] (1852–1943) konnte Ende des 18. Jahrhunderts durch eine spezielle Färbetechnik von Gehirngewebe diese Struktur unter einem Mikroskop sichtbar machen und durch weitere Untersuchungen feststellen, dass das Nervensystem aus Millionen von Nervenzellen besteht und diese über spezielle Verbindungen –die Synapsen– miteinander kommunizieren. Cajal benutzte schematische Darstellungen (*Visualisierungen*) als Arbeitshypothesen (vgl. ([Llinás, 2003]) Seite 77). Abbildung 2.1 zeigt eine solche verwendete Darstellung. Cajal bekam 1906 für eine Vielzahl von Entdeckungen den Nobelpreis für Medizin verliehen.

Visualisierung kann wie dieses Beispiel aus der Medizin zeigt, daher oft der erste Schritt sein wenn es darum geht, strukturelle Merkmale eines Netzwerkes zu analysieren [Newman, 2010]. Es ist für die Visualisierung von Netzwerk Daten sehr viel geeignete Software verfügbar. Für einfache Netzwerke kann man durch eine Visualisierung unmittelbar strukturelle Eigenschaften erkennen, die man aus den reinen Netzwerkdaten nur sehr schwer ablesen kann [Newman, 2010]. Newman [2010] konstatiert, dass die Analyse mittels Visualisierung nur für Netzwerke mit ein paar hundert Elementen und sehr wenigen Verbindungen zwischen diesen Elementen sinnvoll ist. Sobald eine massive Anzahl von Elementen und Verbindungen vorhanden sind, werden Visualisierungen schlichtweg einfach zu kompliziert für das menschliche Auge.

Interdisziplinäre Übereinkunft und Wissenschaftlichkeit Netzwerkanalyse befasst sich mit sehr großen Netzwerken, die aus Millionen von Elementen und Verbindungen bestehen. Die Analyse von komplexen Netzwerken erfordert andere Techniken als die der Visualisierung. Bestimmte Eigenschaften die durch Visualisierung eines einfachen Netzwerks mit *einfachem Auge* ersichtlich sind, könnte man auch mathematisch berechnen. Bei zunehmender Komplexität eines Netzwerkes ist eine mathematische Berechnung unabdingbar, da das reine ablesen von Eigenschaften durch eine Visualisierung nicht mehr möglich ist.

Die Netzwerkanalyse bietet eine Fülle von Werkzeugen und Techniken, die auch die Analyse von sehr großen Netzwerken ermöglichen, um herauszufinden was man aus Netzwerkdaten folgern kann [Newman, 2010].

Eine Herausforderung dieser noch jungen Wissenschaft war es, eine gemeinsame Theorie aufzubauen, damit die einzelnen Wissenschaftsbereiche eine gemeinsame Sprache haben. Demzufolge sollte ein gemeinsames Verständnis der Struktur und *Eigenschaften* von Netzwerken ermöglicht werden. Die folgenden Abschnitte liefern zusammenfassend die wichtigsten Definitionen aus der Literatur.

Struktur und Dynamik (Verhalten) von Netzwerken spielen eine wesentliche Rolle [Easley und Kleinberg, 2010]. Easley und Kleinberg [2010] nennen zwei große Theorien, die eng mit der Netzwerktheorie verbunden sind und in den weiterführenden Betrachtungen eine große Rolle spielen:

Graphentheorie: Ermöglicht die Analyse der Struktur von Netzwerken.

[1] Eine Biografie ist unter `http://www.nobelprize.org/nobel_prizes/medicine/laureates/1906/cajal-bio.html` und `http://de.wikipedia.org/wiki/Santiago_Ramon_y_Cajal` abrufbar.

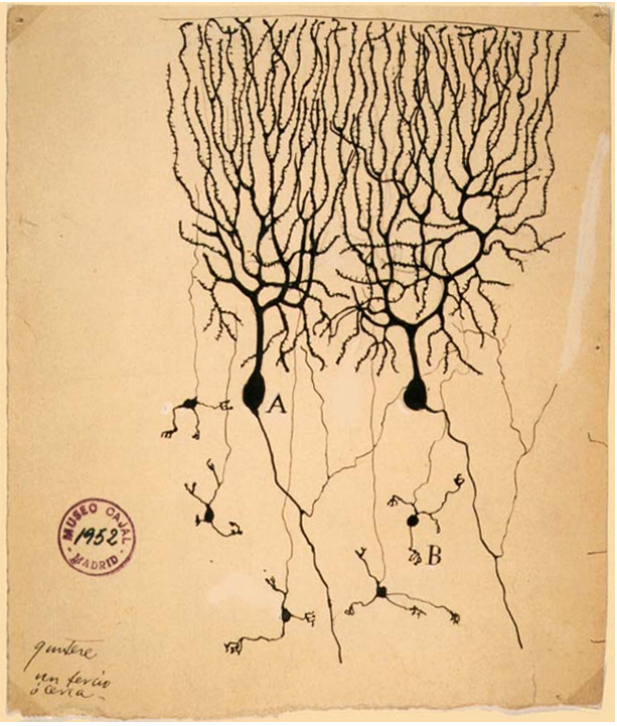

Abbildung 2.1: Darstellung zweier Neuronen gezeichnet von Santiago Ramón y Cajal, 1899; *Instituto Santiago Ramón y Cajal*, Madrid

Spieltheorie: Die Spieltheorie liefert Modelle, die sich mit dem Erklären und Vorhersagen von Verhalten und Handeln in Strukturen auseinandersetzt.

2.2 Was ist ein Netzwerk?

Ein Netzwerk wird in der mathematischen Literatur oft als Graph bezeichnet [Newman, 2010]. Es ergibt sich offensichtlich ein starker Konnex zu der Graphentheorie. Häufig wird das Königsberger Brückenproblem (1736), eines der klassischen Probleme der Graphentheorie, als die Geburtsstunde der Netzwerktheorie genannt [Newman, 2003]. Leonhard Euler (1707–1783) hat für seine Lösung des Königsberger Brückenproblem, ein Brücken-Netzwerk aus der realen Welt als Graphen-Modell abstrahiert und konnte mit diesem Modell die mathematische Fragestellung beantworten.

Die Graphentheorie bezieht sich hauptsächlich auf die theoretischen Aspekte von Graphen. Wie in Abschnitt 2.1 dargelegt, ist das Ziel der Netzwerkanalyse die empirsche Untersuchung von Real-Life-Netzwerken. Die Graphentheorie spielt bei der formalen Beschreibung von Netzwerken eine wichtige Rolle.

[Easley und Kleinberg, 2010] sehen auf einer abstrakten Ebene, Graphen als eine Struktur die es erlaubt Beziehungen zwischen Elementen zu definieren. Ein Netzwerk (Graph) kann definiert werden als eine Sammlung von Objekten (*Knoten*), die durch eine Menge von Verbindungen (*Kanten*) paarweise miteinander in Beziehung gebracht werden (vgl. [Newman, 2010], [Easley und Kleinberg, 2010]).

Ein *Knoten* ist die wesentlichste Einheit in einem Netzwerk. Zwei Knoten können durch eine *Kante* verbunden werden [Newman, 2003]. Knoten (*nodes*) (oder Ecken (*vertices*)) und Kanten (*edges*) werden in den verschiedene Fachgebieten auch folgendermaßen benannt [Newman, 2003]:

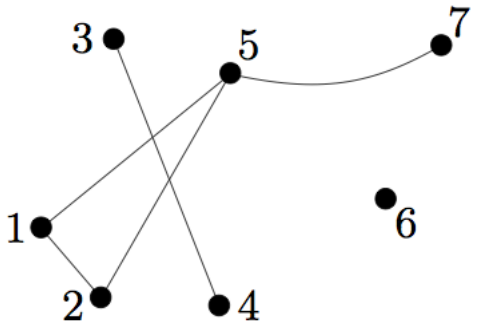

Abbildung 2.2: Bildliche Darstellung eines Graphen [Diestel, 2010]

Computerwissenschaft: Knoten (*nodes*) und Links (*links*)

Physik: Orte (*sites*) und Bindungen(*bonds*)

Soziologie: Akteure (*actors*) und Beziehungen (*ties*)

In der Netzwerktheorie sind Knoten und Kanten immer abstrakte Stellvertreter von spezifischen Objekten und Verbindungen eines realen Systems, welche sich mit einem Graphen beschreiben lassen. Abschnitt 2.5 zeigt Beispiele von realen Netzwerken und den zugehörigen Knotentypen und Kantentypen auf.

2.2.1 Grundlagen aus der Graphentheorie

Ein Graph ist ein mathematisches Modell. Formal kann ein Graph als ein Paar $G = (V, E)$ disjunkter Mengen mit $E \subseteq [V]^2$ definiert werden [Diestel, 2010]. Die Elemente von E sind 2-elementige Teilmengen von V. Knoten (Elemente) des Graphen G sind die Elemente von V; $V(G)$ ist die Knotenmenge. Kanten (Verbindungen) des Graphen G sind die Elemente von E; $E(G)$ ist die Kantenmenge.

Die einfachste Form eines Netzwerkes kann als eine Sammlung von Punkten die durch Linien paarweise miteinander in Verbindung sind, gedacht und auch bildlich dargestellt werden(vgl. [Newman, 2010], [Diestel, 2010]). Nach [Diestel, 2010] ist $G = (V, E)$ ein Graph *auf* V. Abbildung 2.2 zeigt eine Darstellung von G *auf* $V = \{1, 2, 3, 4, 5, 6, 7\}$ mit der Kantenmenge $E = \{\{1, 2\}, \{1, 5\}, \{2, 5\}, \{3, 4\}, \{5, 7\}\}$. Die Punkte sind die Knoten und die Linien sind die Kanten. Gemäß [Diestel, 2010] ist die formale Definition des Graphen von seiner bildlichen Darstellung unabhängig.

$|G|$ ist die Ordnung von G und wird als kurzschreibweise für $|V(G)|$ verwendet. $|G|$ ist die Anzahl der Knoten eines Graphen. $\|G\|$ ist die Anzahl der Kanten in einem Graphen und wird in der Literatur anstatt von $|E(G)|$ geschrieben [Diestel, 2010].

Die Begriffe *inzident* und *adjazenz* werden beispielsweise von [Diestel, 2010] wie folgt definiert:

- Eine Knoten v ist mit einer Kante e genau dann *inzident*, wenn $v \in e$ gilt. Zwei Knoten sind mit einer Kante e inzident, wenn die beiden Knoten die Endknoten von der Kante e sind, und e diese beiden Knoten verbindet.
- Zwei Knoten (x, y von G) sind *adjazent* (benachbart), wenn $xy \in E(G)$ gilt. Zwei Kanten $e \neq f$ sind benachbart, falls sie einen gemeinsamen Endknoten haben.

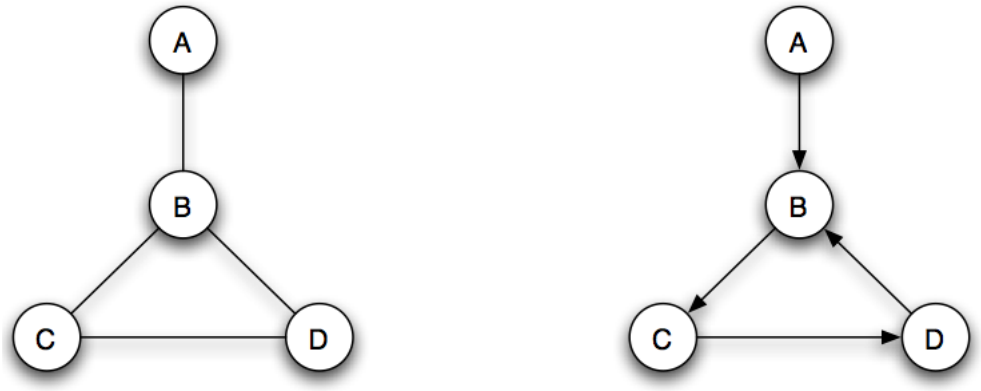

Abbildung 2.3: Graph mit ungerichteten Kanten (links) im Gegensatz ein Graph mit gerichteten Kanten (rechts) [Easley und Kleinberg, 2010]

2.2.2 Typen von Netzwerken

Die bisherigen Betrachtungen (vgl. Abschnitt 2.2 und 2.2.1) sind von einem Netzwerk in seiner einfachsten Form ausgegangen. D.h. ein Netzwerk mit jeweils genau einem Typ von Knoten sowie Kanten und mit der Einschränkung, dass eine Kante nur eine symmetrische Verbindung zwischen zwei Knoten beschreibt.

Ein Netzwerk kann an Komplexität gewinnen, wenn die Elemente Knoten und Kanten um Eigenschaften erweitert werden, oder wenn in einem Netzwerk verschiedene Typen von Knoten und Kanten benötigt werden, damit das real betrachtete System adäquat als Netzwerk abgebildet werden kann.

Kanten können eine symmetrische oder asymmetrische Verbindung zwischen zwei Knoten bedeuten. Man spricht hier von gerichteten und ungerichteten Kanten. Daraus resultiert folgende Unterscheidung:

Ungerichteter Graph: Ein Graph, der nur aus ungerichteten Kanten besteht.

Gerichteter Graph (*digraph*): Ein Graph, der aus gerichteten Kanten besteht. Dadurch können asymmetrische Relationen beschrieben werden (notwendig für Hierarchien). Kanten bekommen eine Richtung (Orientierung) aufgeprägt.

Abbildung 2.3 zeigt in der Gegenüberstellung einen Graphen mit ungerichteten und mit gerichteten Kanten. Kanten können auch weitere Eigenschaften haben:

Typ: Damit bestimmte Beziehungen abgebildet werden können, sind manchmal Kanten typisiert. Somit ist es möglich, Beziehungen wie beispielsweise Freundschaft, Bekanntschaft, Arbeitsbeziehung usw. abzubilden.

Gewicht: Die Verbindung zwischen zwei Knoten kann gewichtet sein, d.h. der Kante wird eine Zahl zugeordnet. Damit könnte man beschreiben wie stark eine Bindung zwischen zwei Geschäftspartnern ist, oder wie oft zwei Menschen mit einander kommunizieren. Ein *gewichteter Graph* besteht aus gewichteten Kanten.

Rangfolge: In einem Sozialen Netzwerk könnte man diese Eigenschaft benutzen, um zu beschreiben wer mein bester Freund, mein zweitbester Freund usw. ist.

Zentralitätsmasse (vgl. Abschnitt 2.3.9) und eine Vielzahl weitere Eigenschaften können als weitere Eigenschaften verwendet werden, sofern es in Abhängigkeit der Struktur des Netzwerkes als zweckmäßig

erscheint.

Es gibt eine Menge weiterer Konzepte die für die mathematische Modellierung von Netzwerken von Interessen sein können, wie beispielsweise *Hypergraphen* die eine Abbildung von *Schlingen* und *Mehrfachkanten* zulassen.

In einem Netzwerk können auch verschiedene Typen von Knoten vorkommen. In der bisherigen Betrachtung sind wir davon ausgegangen, dass ein Netzwerk nur aus einem Typ von Knoten besteht. Graphen können auch in verschiedensten Ausprägungen partitioniert sein [Newman, 2003]. Für solche Fälle werden hauptsächlich bipartite Graphen (*two mode network*) verwendet. Newman [2010] konstatiert, dass man *Hypergraphen* oftmals adäquater als bipartite Graphen abbilden kann.

Man sagt ein Graph $G = (V, E)$ heißt $r - partit$, wenn eine Partition von V in r Teile existiert und jede Kante von G in verschiedenen Partitionsklassen liegt [Diestel, 2010]. Es gilt die Einschränkung, dass Ecken aus der gleichen Klasse nicht benachbart sein dürfen. Bipartite Graphen (bzw. ein 2-partiter Graph) bestehen aus Knoten aus zwei verschiendenen Typen, mit Kanten die nur ungleiche Typen mit einander verbinden [Newman, 2003].

2.2.3 Repräsentation von Netzwerken

Netzwerke können mathematisch durch verschiedene Methoden komplett beschrieben werden. Im Wesentlichen geht es darum, dass man auf eine geeignete Repräsentation eines Netzwerkes Algorithmen anwenden kann und damit Probleme aus der Graphentheorie oder Netzwerkanalyse berechnen kann [Newman, 2010].

Es gibt eine Menge von verschiedenen Möglichkeiten, die es erlauben ein Netzwerk mathematisch abzubilden und die in der Literatur genau beschrieben sind:

- Adjazenzliste
- Adjazenzmatrix
- Inzidenzmatrix
- usw.

Alle Repräsentationen können sowohl mit einfachen Netzwerke (*one mode network*), die durch einen einfachen unipartiten Graphen beschrieben werden können, als auch mit komplexeren Netzwerken (*two mode network*), die als bipartite Graphen beschrieben werden, abgebildet werden.

Die Adjazenzliste und die Adjazenzmatrix sind in der Literatur die bedeutendsten Repräsentationen. Alternativ werden die Adjazenzliste und die Adjazenzmatrix oft auch als Nachbarschaftsliste oder Nachbarschaftsmatrix benannt.

In folgenden Abschnitt wird die Adjazenzmatrix eines einfachen Netzwerkes (*one mode network*) genauer beschrieben. Es werden Vorteile und Nachteile im Vergleich zu anderen Abbildungsmethoden erläutert. Abschließend wird anhand eines Beispiels die Erstellung der Adjazenzmatrix illustriert.

2.2.3.1 Adjazenzmatrix

Die Adjazenzmatrix $\mathbf{A}$ eines ungerichteten Graphen ist als Matrix mit den Elementen A_{ij} definiert durch [Newman, 2010]:

$$A_{ij} = \begin{cases} 1 & \text{falls eine Kante zwischen den Knoten } i \text{ und } j \text{ existiert,} \\ 0 & \text{sonst.} \end{cases} \tag{2.1}$$

Die Adjazenzmatrix ist eine $n \times n$ Matrix und beinhaltet eine kompakte Beschreibung eines Netzwerkes, dass aus n Knoten besteht. Einfache Metriken wie beispielsweise der Grad eine Knotens (vgl. Abschnitt 2.3.5) können daraus abgeleitet werden [Newman, 2010].

Die Adjazenzmatrix als Datenstruktur benötigt $\mathcal{O}(n^2)$ Speicher, unabhängig von der Anzahl der Kanten [Cormen et al., 2007, Seite 533f]. Die Adjazenzmatrix eines ungerichteten Graphen ist symmetrisch, d.h. die obere Dreiecksmatrix ist für die Abbildung ausreichend und es ist deshalb auch ausreichend für Berechnungen nur die obere Dreiecksmatrix zu verwenden. Somit ist es möglich für ungerichtete Graphen den erforderlichen Speicherplatz zu halbieren [Cormen et al., 2007, Seite 533f]. Die Adjazenzmatrix wird oft wegen ihrer Einfachheit der Adjazenzliste vorgezogen, sofern die Graphen einigermaßen klein sind. Die Adjazenzmatrix hat gegenüber der Adjazenzliste den Vorteil, dass für einen ungewichteten Graphen nur ein Bit pro Eintrag verwendet werden muss.

Die Adjazenzmatrix-Darstellung für das einfache Netzwerk aus Abbildung 2.4 kann durch Anwendung der Formel 2.1 wie folgt bestimmt werden:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

2.3 Netzwerk Metriken

In diesem Abschnitt werden verschiedene Netzwerk Metriken vorgestellt. Mit Netzwerk Metriken kann man Netzwerke charakterisieren. Die Wurzeln von sehr vielen Metriken liegen in der Graphentheorie. Dieser Abschnitt baut auf den Grundlagen aus Abschnitt 2.2.1 auf. Sofern es möglich und sinnvoll ist, wird das Netzwerk aus Abbildung 2.4 als Beispiel herangezogen.

2.3.1 Netzwerkgröße (*network size*)

Die größe eines ungerichteten Netzwerkes ist durch die Anzahl der Knoten definiert. In der Graphentheorie spricht man von der Ordnung eines Graphen (vgl. Abschnitt 2.2.1). Dies wird formal durch $|V(G)|$ dargestellt [Diestel, 2010]. Das Netzwerk aus Abbildung 2.4 hat die Ordnung $|V(G)| = 9$.

2.3.2 Maximale Anzahl von Kanten

[Diestel, 2010] konstatiert, das ein ungerichteter Graph mit n Knoten maximal $\frac{n(n-1)}{2}$ Kanten haben kann. Ist das der Fall, dann ist der Graph *vollständig*. Das Netzwerk aus Abbildung 2.4 ist mit $\frac{9(9-1)}{2} = \frac{72}{2} = 36$ Kanten vollständig.

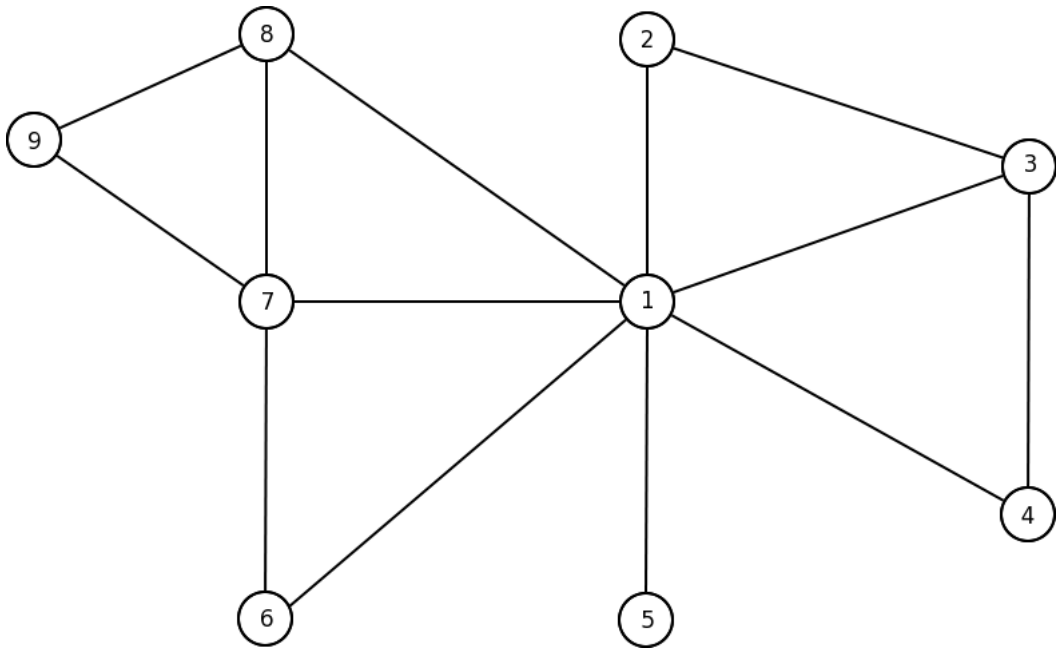

Abbildung 2.4: Ein einfaches Netzwerk mit 9 Knoten und 13 Kanten

2.3.3 Pfade und Pfadlängen

Ein *Pfad* gibt Aufschluss über die Navigationsfähigkeit in einem Netzwerk. Ein Pfad ist eine Verbindung zwischen zwei Knoten in einem Netzwerk. Die *Pfadlänge* entspricht der Anzahl der Kanten, die in einem Pfad vorkommen können [Newman, 2010].

Ein Netzwerk (*one-mode*) ist verbunden, sofern für jedes Knotenpaar ein Pfad existiert. In diesem Fall ist das Netzwerk eine *Komponente*. Sind nicht alle Knotenpaare über einen Pfad erreichbar, dann besteht das Netzwerk aus mehreren Komponenten. Komponenten beschreiben die Struktur von Netzwerken [Easley und Kleinberg, 2010]. Abbildung 2.5 zeigt ein Netzwerk, das aus drei Komponenten besteht. Zwischen Knoten aus verschiedenen Komponenten besteht kein Pfad.

Es gibt zwei *besondere* Pfade, die nachfolgend kurz beschrieben werden [Newman, 2010]:

Geodätische Distanz (*geodesic path*): Ist die kürzeste Distanz zwischen zwei Knoten, der in einem Netzwerk möglich ist.

Durchmesser (*diameter*): Der Durchmesser eines Netzwerkes ist jene Geodätischen Distanz mit der maximalen Pfadlänge.

2.3.4 Dichte

Die Dichte [Adamic, 2008a] eines Netzwerkes ist das Verhältnis der Anzahl der vorhandenen Kanten zu der maximalen Anzahl von Kanten, die in einem Netzwerk möglich sind. [Newman, 2010] definiert formal für ein ungerichtetes Netzwerk:

$$\rho = \frac{2m}{n(n-1)} \tag{2.2}$$

Die Variable m in der Formel 2.2 ist die Anzahl der Kanten in einem Netzwerk, wohingegen n die Anzahl der Knoten des Netzwerkes symbolisiert. Die Dichte eines Netzwerkes liegt immer im Bereich

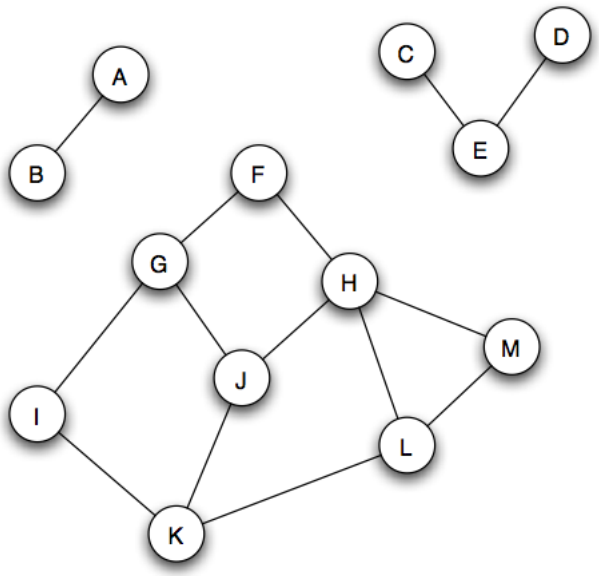

Abbildung 2.5: Beispiel für ein Netzwerk das aus drei Komponenten besteht [Easley und Kleinberg, 2010]

$0 <= \rho <= 1$. Ein Netzwerk wird als *dicht* bezeichnet, wenn der Wert ρ für $n \rightarrow \infty$ konstant bleibt [Newman, 2010].

Das Netzwerk aus Abbildung 2.4 hat eine Dichte $\rho = \frac{2*13}{9*(9-1)} = \frac{26}{72} = \frac{13}{36} \approx 0.36$.

2.3.5 Grad eines Knotens (*node-degree*)

In einem Graphen ist der Grad eines Knotens [Newman, 2010] definiert durch die Anzahl der Kanten mit denen der Knoten verbunden ist. Den Grad eines Knotens k_i für einen ungerichteten Graphen kann man unmittelbar über die Adjazenzmatrix (vgl. Abschnitt 2.1) ableiten:

$$k_i = \sum_{j=1}^{n} A_{ij} \tag{2.3}$$

D.h. der Grad des Knotens i, ist die Summe der Spalte i der Adjazenzmatrix. Der Knoten mit der Beschriftung *1* aus Abbildung 2.4 hat den Grad $k_i = 7$.

2.3.5.1 Eingangsgrad (*in-degree*)

Der Eingangsgrad eines Knoten k_i^{in} ist die Anzahl der gerichteten Verbindungen die zu dem Knoten zeigen. Für einen gerichteten Graphen, kann der Eingangsgrad wiederum über die Adjazenzmatrix bestimmt werden [Newman, 2010]:

$$k_i^{in} = \sum_{j=1}^{n} A_{ij} \tag{2.4}$$

Der Eingangsgrad eines Knoten i ist die Summe der Spalte i der Adjazenzmatrix.

2.3.5.2 Ausgangsgrad (*out-degree*)

Der Ausgangsgrad eines Knoten k_j^{out} ist die Anzahl der gerichteten Verbindungen, die bei einem Knoten starten. Die Adjazenzmatrix eines gerichteten Graphen kann wiederum für die Bestimmung des Ausgangsgrades herangezogen werden:

$$k_j^{out} = \sum_{i=1}^{n} A_{ij} \tag{2.5}$$

Die Summe der Zeile j der Adjazenzmatrix entspricht dem Ausgangsgrad k_j^{out} eines Knotens j.

2.3.6 Gradverteilung (*degree distribution*)

Die Häufigkeitsverteilung der Knotengrade ist einer der wichtigsten Eigenschaften eines Netzwerkes. Mit dieser Metrik lässt sich die Struktur des Netzwerkes charakterisieren. In Abschnitt 2.3.5 wurde der Grad eines Knotens, durch die Anzahl der Kanten die mit diesen Knoten verbunden sind, definiert. Die Anteile der Knoten mit dem Grad k wird durch p_k symbolisiert und kann als die Wahrscheinlichkeit, dass ein zufälliger Knoten eines Netzes den Grad k hat, interpretiert werden. Die graphische Darstellung von p_k für ein beliebiges Netzwerk, kann durch die Erstellung eines Histogrammes der einzelnen Knotengrade durchgeführt werden [Newman, 2003]. Ein solches Histogramm ist die Gradverteilung eines Netzwerkes.

Abbildung 2.6 zeigt die Darstellung der Gradverteilung für zwei unterschiedlich Netzwerke. Wie in Abbildung 2.6 rechts unten zu sehen, ist es oftmals üblich die Gradverteilung in doppelt logarithmischer Darstellung (*log-log plot*) abzubilden. Die Gradverteilung zeigt, wie verschieden die Kanten in einem Netzwerk auf die Knoten verteilt sind. Somit können Aussagen über die Struktur des Netzwerkes hinsichtlich Verschiedenheit der Knoten und der Verbundenheit des Netzes getroffen werden.

2.3.7 Naben (*hubs*)

Es ist der Fall, dass sehr viele reale Netzwerke durch eine kleine Zahl von großen Knoten *beherrscht* werden. Diese Knoten sind durch einen unverhältnismäßig hohen *Grad* ausgezeichnet und werden als Naben (*hubs*) bezeichnet. Ein *hub* kann in Anspielung einer Fahrradnabe, aus der sehr viele Speichen gehen, gedacht werden [Barabási und Bonabeau, 2004].

Das World Wide Web, soziale Netzwerke aber auch das Stoffwechselsystem einer Zelle [Barabási und Bonabeau, 2004] sind typische Beispiel von Netzen, die *hubs* beinhalten. In einem Sozialen Netzwerk wäre ein *hub* eine zentrale Person mit sehr vielen Bekannten [Adamic, 2008b]. Im World Wide Web gibt es nur wenige Web-Seiten (wie beispielsweise Google) die eine sehr hohe Anzahl von Links aufweisen und das Internet dominieren (vgl. [Barabási und Bonabeau, 2004], [Newman, 2010]).

Hubs sind in einem Netz sehr einflussreich und können das Verhalten und die Performance eines Netzes mit bestimmen [Newman, 2010]:

- Ein Netzwerk kann durch das Entfernen einer Nabe oder einen gezielten Angriff auf eine Nabe in Stücke gerissen werden (*Aussfallsicherheit*) [Barabási und Bonabeau, 2004].
- Ein *hub* hat sehr viel Einfluss in einem Netzwerk. Das kann in einem Sozialen Netzwerk auch negative Auswirkungen haben, sofern es sich um die Verbreitung von Krankheiten handelt (*Epidemien*) [Adamic, 2008b].

2.3.8 Cluster-Koeffizient

Der *Cluster-Koeffizient* spielt eine Rolle für die Analyse der Netzwerkstruktur. Es gilt zu analysieren, ob stark vernetzte Cluster oder Gruppen in einem Netzwerk existieren. Der Cluster-Koeffizient ist ein Maß für die Kompaktheit und die Dichte in einem Graphen. Es gibt zwei Ansätze, die bei der Analyse angewandt werden können: Die *lokale* Betrachtung eines einzelnen Knotens oder die *globale* Betrachtung des gesamten Graphen (vgl. [Scholz, 2008], [Newman, 2003]).

Der *lokale* Cluster-Koeffizient C_i bezieht sich auf einen einzelnen Knoten und beschreibt die Vernetzung von Nachbarknoten untereinander. C_i entspricht der Wahrscheinlichkeit eines Knotens i, dass zwei Nachbarknoten auch miteinander verbunden sind [Newman, 2003]. Es handelt sich hier um den Quotienten aus der Anzahl der Kanten, die zwischen den Nachbarknoten eines Knotens i verlaufen und der maximalen Anzahl der Kanten, die zwischen den Nachbarknoten möglich sind. Der Cluster-Koeffizient liegt in einem Wertebereich zwischen 0 und 1. Ein vollständig vernetzter Bereich hat den Cluster-Koeffizienten $C_i = 1$ und wird als *Clique* bezeichnet. Dies ist ein vollständiger Teilgraph und impliziert perfekte Transitivität [Scholz, 2008].

Im Netzwerk aus Abbildung 2.4 ist der Knoten 9 mit den Knoten 7 und 8 verbunden. Gleichermaßen ist zwischen den Knoten 7 und 8 eine Kante möglich und vorhanden. Für den Knoten 1 ist der Cluster-Koeffizient zu $C_1 = 1$. Der Knoten 8 hat die Knoten 1, 2 und 7 als Nachbarn. Es sind nur die Knoten 7 und 8 und die Knoten 1 und 7 miteinander verbunden. Dadurch ergibt sich der Cluster-Koeffizent zu $C_8 = \frac{2}{3}$.

Der *globale* oder der *durchschnittliche* Cluster-Koeffizient C für den gesamten Graphen wird aus den einzelnen lokalen Cluster-Koeffizienten C_i die gemittelt werden, ermittelt [Newman, 2003].

$$C = \frac{1}{n} \sum_{i=1}^{n} C_i \tag{2.6}$$

Der ermittelte Wert gibt Aufschluss darüber, wie stark Cliquenbildung im gesamten Graph ausgeprägt ist.

Ausgehend von dem lokalen Cluster-Koeffizienten C_i kann auch der durchschnittliche Cluster-Koeffizient aller Knoten vom Grad k berechnet werden und wird mit $C(k)$ symbolisiert. Mit diesem Wert kann man eine Aussage treffen wie die Ausprägung der Cliquenbildung mit der Anzahl von Nachbarn variiert.

2.3.9 Zentralitätsmaße (*centrality*)

Zentralitätsmaße werden für die Bestimmung der zentralsten und wichtigsten Knoten herangezogen. Beispielsweise können Knoten, die wichtig für den Zusammenhalt oder den Informationsfluss sind, anhand Zentralitätsmaße bestimmt werden. Zentralität wird oft auch als Prestige verstanden, doch hier ist Vorsicht geboten. Prestige oder Prominenz kann nur in einem gerichteten Graphen untersucht werden. Prestige ist nur gegeben, wenn viele Verbindungen auf einen Knoten gerichtet sind. Zentralität ist in einem ungerichteten Graphen dann gegeben, wenn er an vielen Verbindungen beteiligt ist. Dieser Knoten ist dann sichtbar und mächtig.

Es werden Maßzahlen berechnet. Es gilt, je größer der berechnete Wert für einen Knoten umso zentraler ist ein Knoten. Nachfolgend werden einige Metriken kurz beschrieben.

Grad-basierte Zentralität (*degree centrality*): Die Grad-basierte Zentralität ist das einfachste Zentralitätmaß und wird über den Grad eines Knotens bestimmt [Newman, 2010]. Beispielsweise hat

ein Knoten mit einer hohen Anzahl an direkten Beziehungen mehr Prestige, besseren Zugang zu Informationen oder mehr Einfluss [Newman, 2010]. Ein Knoten mit einer hohen grad-basierten Zentralität ist sehr oft ein Hub.

Eigenvektor Zentralität: Das Konzept der Eigenvektor Zentralität meint, dass ein Knoten der Verbindungen zu wichtigen Knoten hat, selbst wiederum wichtig ist [Newman, 2010]. Es gibt sehr viele Umstände wo diese Betrachtung in einem Netzwerk zutrifft, beispielsweise bewirkt die direkter Nachbarschaft eines Knotens zu zentralen Akteuren wiederum eine Erhöhung der Sichtbarkeit und Reputation. Die Eigenvektoren der Adjazenzmatrix (vgl. Abschnitt 2.2.3.1) beinhalten Informationen bezüglich der Relevanz der Knoten und werden für die Berechnung herangezogen [Newman, 2010]. Die Zentralität $\mathbf{x}$ erfüllt die Gleichung $\mathbf{Ax} = \lambda\mathbf{x}$ [Newman, 2010]. Es gibt verschiedene Eigenwerte λ für die eine Eigenvektor Lösung existiert. Zusätzliche Bedingung ist, dass nur der größte Eigenwert für die Bestimmung der Zentralität herangezogen werden soll.

Nähe-basierte Zentralität (*closeness centrality*): Dieses Maß wird sehr oft in sozialen und anderen Netzwerkstudien verwendet [Newman, 2010]. Die Nähe-basierte Zentralität ist ein Maß dafür, wie schnell ein Knoten auf andere Knoten zugreifen kann. Knoten mit hoher nähe-basierter Zentralität haben kurze Pfade zu anderen Knoten. Zentrale Knoten haben eine hohe Sichtbarkeit darauf, was in einem Netzwerk passiert. Bei der Berechnung wird die durchschnittliche Distanz von einem Knoten zu anderen Knoten berechnet [Newman, 2010].

Zwischenzentralität (*betweeness centrality*): Knoten, die auf Pfaden zwischen anderen Knoten liegen, sind in dieser Betrachtung zentral [Newman, 2010]. Es wird bestimmt, wie viele kürzeste Pfade durch einen Knoten führen. Die Zwischenzentralität ist ein Mass für die Fähigkeit eines Knotens anhand seiner Position Verbindungen zu anderen Gruppen herzustellen. Knoten mit hoher Zwischenzentralität können voneinander unabhängige Komponenten verbinden. Das gezielte Entfernen solcher Knoten kann den Zerfall des Netzes in Komponenten bedeuten. Ein Knoten mit hoher Zwischenzentralität hat beispielsweise eine beliebte bzw. starke Position inne und hat großen Einfluss darauf, was in einem Netzwerk passiert.

2.4 Netzwerkmodelle

Mit der einfachen Metrik der Gradverteilung lassen sich Netzwerke in verschiedene Modelle klassifizieren. Wie sich zeigte, haben zahlreiche komplexe Netzwerke bestimmte Eigenschaften gemeinsam. In der Literatur werden oft die beiden folgenden Modelle herangezogen:

- Zufallsnetzwerke
- Skalenfreie Netzwerke

Das Modell der Skalenfreie Netzwerke spielt vor allem für reale Netzwerke eine wichtige Rolle.

2.4.1 Zufallsnetzwerke

Das Modell der Zufallsnetzwerke ist in der Literatur auch als Erdös-Rényi Modell bekannt [Barabási und Oltvai, 2004]. Dieses Modell wurde in der Literatur anfangs verwendet, um die Entstehung großer Netzwerke zu beschreiben. In einem Zufallsnetzwerk sind die einzelnen Knoten nach einem reinen Zufallsprinzip miteinander verbunden. Wenn aus einem Zufallsnetzwerk ein Knoten entfernt wird, dann ist es noch immer ein Zufallsnetzwerk. Bemerkenswert ist, dass die Gradverteilung einer Poisson-Verteilung gleicht [Barabási und Oltvai, 2004]. In der Literatur wird auch die Binomialverteilung öfters erwähnt. D.h. die Grade der Knoten schwanken um einen Durchschnittswert, somit haben alle Knoten im Netzwerk einen ähnlichen Knotengrad. Hubs sind in solchen Netzwerken nicht zu finden. In Abbildung 2.6

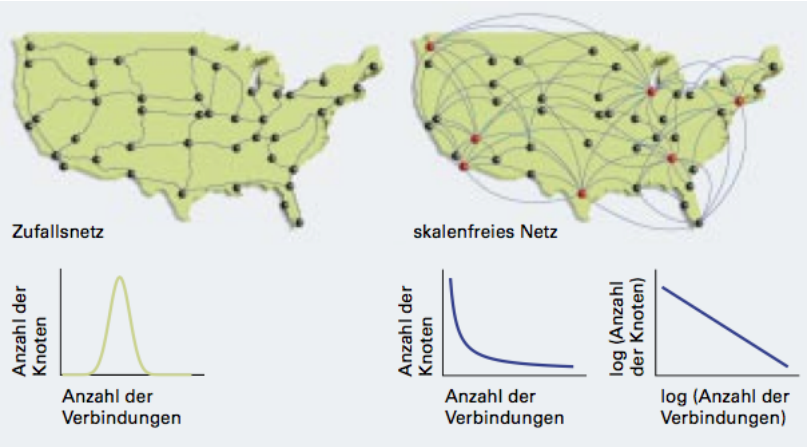

Abbildung 2.6: Vergleich eines Zufallnetzes mit einem Skalenfreien Netzwerk anhand der charakteristischen Gradverteilungen [Barabási und Bonabeau, 2004]

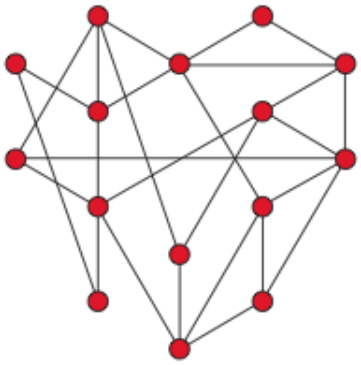

Abbildung 2.7: Darstellung eines Zufallnetzwerk [Barabási und Oltvai, 2004]

ist eine typische Gradverteilung für ein Zufallsnetzwerk ersichtlich Barabási und Bonabeau [2004]. Abbildung 2.7 zeigt eine typische Struktur eines Zufallsnetzwerkes. In einem Zufallsnetzwerk ist die mittlere Distanz zwischen zwei Knoten gering (geringer Durchmesser). Der lokale Clustering-Koeffizient ist unabhängig vom Knotengrad [Barabási und Oltvai, 2004] und ist in einem Zufallsnetzwerk wiederum gering. Im Gegensatz zum Modell der Skalenfreien Netzwerke sind sie kein adäquates Modell, um reale Netzwerke hinsichtlich Entstehung und Struktur zu beschreiben.

2.4.2 Skalenfreie Netzwerke (*scale-free networks*)

Die moderne Netzwerktheorie befasst sich heutzutage fast ausschließlich mit Skalenfreien Netzwerken (vgl. Abschnitt 2.5). Albert-László Barabási hat den Begriff *skalenfrei* eingeführt. Ein Netwerk deren Gradverteilung dem Potenzgesetz folgt, wird in der Literatur als skalenfreies Netzwerk bezeichnet (vlg. [Barabási und Bonabeau, 2004], [Newman, 2010]). Das Potenzgesetz bezeichnet eine ploynomiellen Zusammenhang der Form $f(k) = a * k^c$, wobei a und c beliebige Konstanten sind [Easley und Kleinberg, 2010]. Eine charakteristische Gradverteilung die dem Potenzgesetz folgt, ist in Abbildung 2.6 ersichtlich. Interessant ist die Tatsache, dass eine Verteilung die dem Potenzgesetz folgt in einem *log-log plot* als Gerade erscheint. Wenn man für die Formel $f(k) = a * k^c$ auf beiden Seiten den Logarithmus nimmt, dann bekommt man folgende Formel:

$$\log f(k) = \log a - c \log k \tag{2.7}$$

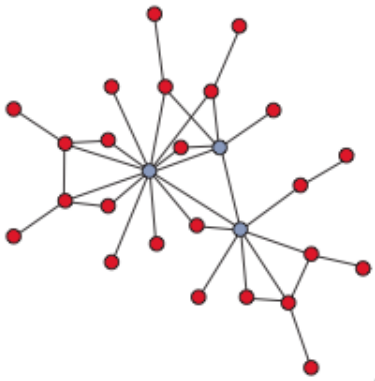

Abbildung 2.8: Darstellung eines Skalenfreien Netzwerk [Barabási und Oltvai, 2004]

Wird diese Beziehung dargestellt, dann erhält man eine gerade Linie. Die Steigung dieser Linie entspricht c, währenddessen $\log k$ dem y-Achsenabschnitt bestimmt [Easley und Kleinberg, 2010].

In einem solchen Netzwerk gibt es nur sehr wenige Knoten, die einen unverhältnismässig großen Grad aufweisen. Diese Knoten werden als *hubs* (vgl. Abschnitt 2.3.7) bezeichnet. Abbildung 2.8 zeigt eine typische Struktur eines kleinen Skalenfreien Netzwerkes. Alle anderen Knoten weisen dahingegen eine kleinen Grad auf. In einem Skalenfreien Netz gibt es keine typische Anzahl von Knotengraden. Barabási und Bonabeau [2004] konstatieren, dass einem solchen Netzwerk eine Skala im Sinne eines inneren Maßstabes fehlt.

Skalenfreie Netzwerke haben typische Merkmale: Einerseits sind diese Netze unempfindlich gegen zufällige Störungen und andererseits sind die Netze anfällig bei gezielte Angriffen auf wichtige Knoten. *Hubs* halten das Netzwerk zusammen und sorgen für eine Verbindung zwischen den einzelnen Bereichen. Werden in einem Skalenfreien Netz gezielt *hubs* entfernt, kann es passieren, dass das Netz in voneinander unabhängige Teile (*Komponenten*) zerfällt [Barabási und Bonabeau, 2004]. Durch die Existenz von *hubs* in einem solchen Netzwerk, ergibt sich, dass Skalenfreie Netzwerke einen geringen Durchmesser besitzen. Der Cluster-Koeffizient variiert in Skalenfreien Netzen, somit kann bzgl. des Cluster-Koeffizienten keine Aussage gemacht werden, die für eine eindeutige Charakterisierung herangezogen werden kann. Skalenfreie Netzwerke unterliegen einem Wachstumsprozess, d.h. hier handelt es sich nicht um geplante Netzwerke. Folgende zwei Mechanismen spielen bei der Entstehung solcher Netzwerke eine Rolle und erklären auch die Existenz von *hubs* [Barabási und Bonabeau, 2004] :

- Wachstum
- Verknüpfungsvorliebe (*preferential attachment*)

Die Verknüpfungsvorliebe ist in vielen Netzwerken zu beobachten. Neue Knoten verbinden sich bevorzugt mit *großen* Knoten. Beliebte Knoten sammeln über die Zeit immer mehr Verbindungen als die Knoten in der Nachbarschaft und können sich mit der Zeit zu einem *hub* entwickeln. Abbildung 2.9 zeigt den Wachstumsprozess in einem skalenfreien Netzwerk; in jedem Schritt wird ein neuer Knoten hinzugefügt, wobei dieser Knoten mit einer kleinen Anzahl (in Beispiel von Abbildung 2.9 wird ein neuer Knoten immer mit zwei vorhandenen Knoten verbunden) bereits vorhandenen Knoten verbunden wird. Die Wahrscheinlichkeit der Verlinkung ist proportional zu dem Grad des Knotens. Daraus folgt, dass ein neuer Knoten mit einer höheren Wahrscheinlichkeit mit einem Knoten, der einen hohen Grad besitzt, verbunden wird. [Barabási und Bonabeau, 2004] liefert unter anderem folgendes Beispiel für die Verknüpfungsvorliebe. In der wissenschaftlichen Literatur werden viel zitierte Arbeiten auch viel gelesen und falls sie wiederum Grundlage für neue Arbeiten (*Wachstum*) werden, dann werden diese

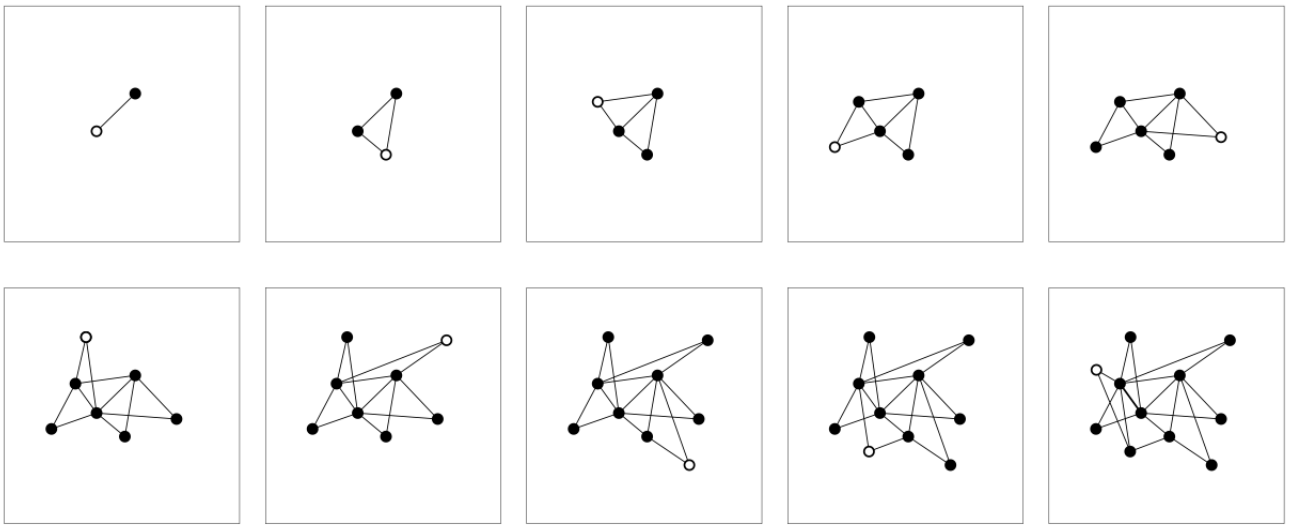

Abbildung 2.9: Wachstum in einem Skalenfreien Netzwerk (*preferential attachment*) [Sofer und Weinkamp, 2005]

Arbeiten wiederum zitiert (*Verknüpfungsliebe*). Dieses Phänomen wird auch oft als der *Matthäus Effekt* bezeichnet:

> *"Denn wer da hat, dem wird gegeben werden, dass er Fülle habe; wer aber nicht hat, von dem wird auch genommen, was er hat."(Matthäus 13, Vers 12)*

Eine weitere Bezeichnung für dieses Phänomen ist das *richer-get-richer* Prinzip.

2.5 Beispiele für reale Netzwerke

In diesem Abschnitt werden reale Netzwerke behandelt, die empirisch untersucht worden sind und in der Literatur häufig genannt werden. Grundsätzlich ist es sinnvoll Netzwerke in verschiedene Klassen einzuteilen. Newman [2010] ordnet reale Netzwerke nach dem folgenden Schema:

- Technologische Netzwerke
- Soziale Netzwerke
- Informationsnetzwerke
- Biologische Netzwerke

In den folgenden Abschnitten werden relevante charakteristische Beispiele angeführt.

2.5.1 Technologische Netzwerke

Diese Klasse beinhaltet künstlich geschaffene Netzwerke. Solche Netzwerke sind typischerweise aufgebaut worden, um Ressourcen zu verteilen. Typische Beispiele für Technologische Netzwerke sind:

- Internet
- Elektrische Stromnetze

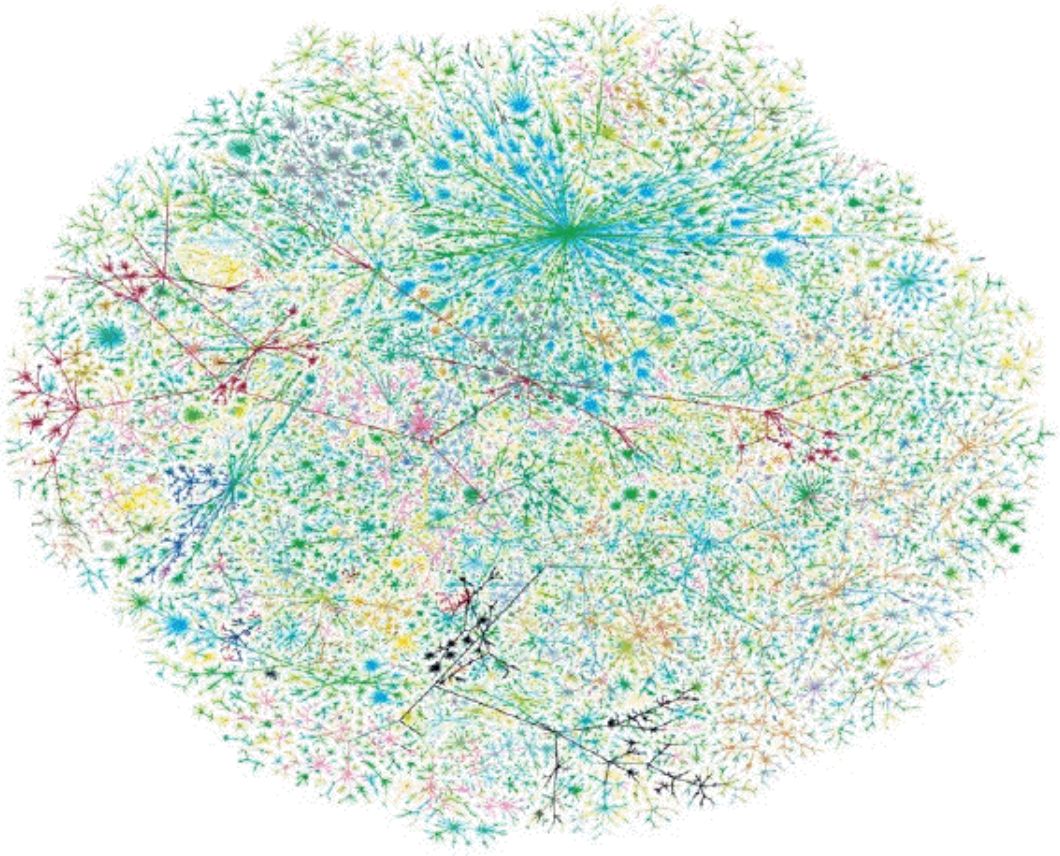

Abbildung 2.10: Visualisierung der Struktur des Internets auf der Ebene der autonomen Systeme (AS) [Newman, 2010]

- Telefonnetze
- Transport Netzwerke

Das Internet ist ein Paradebeispiel für ein Technologisches Netzwerk. Das Internet ist ein globales Netzwerk von Datenverbindungen, das einzelne Computer, Router oder andere Informationssysteme miteinander verbindet. Einzelne Computer oder Router repräsentieren die *Knoten* dieses Netzwerkes. Die *Kanten* wiederum sind phyiskalische Datenverbindungen (elektrisch, optisch oder wireless) die zwei Knoten miteinander verbinden [Newman, 2010].

Newman [2010] konstatiert, dass wir obwohl es sich bei dem Internet um ein von Menschen erzeugtes und sorgfältig aufgebautes Netzwerk handelt, kaum über die Struktur des Internets Bescheid wissen. Es gibt sehr viele Versuche auf empirischen Weg die Struktur des Internets abzuleiten. Abbildung 2.10 zeigt eine Darstellung des Internets auf der autonomen System (AS) Ebene. Hierbei handelt es sich um eine Sammlung von Computern und Routern, die als Einheit verwaltet werden und über ein gemeinsames internes Routing Protokoll verbunden sind. Das Internet besteht aus mehreren autonomen Systemen, die miteinander verbunden sind. Die Knoten in Abbildung 2.10 sind Computer und Router. Die Kanten in Abbildung 2.10 zeigen Routen die Daten zwischen Computern und Routern wandern [Newman, 2010].

In einem Transportnetzwerk sind *Knoten* Destinationen oder Ziele. *Kanten* sind direkte Verbindungen. Abbildung 2.11 zeigt ein Flugnetz und ein U-Bahn-Netz mit der zuvor beschrieben Knoten- und Kantenbeziehung. Auch Straßennetze und Zugnetze sind typische Beispiele für Transportnetzwerke.

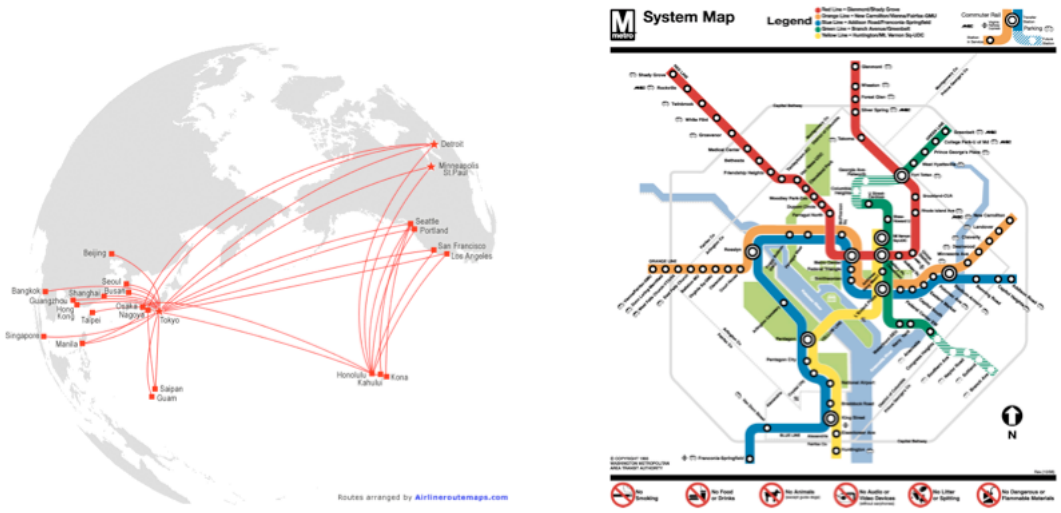

Abbildung 2.11: Beispiele für Transportnetzwerke. Die Abbildung von bestimmten Flugrouten (links) und ein Beispiel für ein typisches U-Bahn-Netz (rechts) [Easley und Kleinberg, 2010]

2.5.2 Soziale Netzwerke

Soziale Netzwerke repräsentieren Kommunikation aber auch Kooperation in einer Gesellschaft. *Knoten* sind typischerweise Personen, die als Akteure in einem sozialen Netzwerk handeln. Auch Gruppen von Personen können durch einen Knoten repräsentiert werden. Die *Kanten* repräsentieren die Beziehung zwei Personen miteinander verbindet oder die Art wie zwei Personen miteinander interagieren. Freundschaft wäre ein Beispiel einer solchen Beziehung. Auch das Muster von Geschäftsbeziehungen zwischen verschiedenen Unternehmen lässt sich mit Sozialen Netzwerken ausdrücken [Newman, 2010].

Abbildung 2.12 zeigt ein berühmtes Soziales Netzwerk aus der Soziologie. Bei dieser Abbildung handelt es sich um das Karate Klub Netzwerk von Wayne Zachary [Newman, 2010]. Die Knoten in diesem Netzwerk sind einzelne Personen und sind die Mitglieder des Karate Klubs. Einzelne Personen sind mit Kanten miteinander verbunden. Eine solche Verbindung bedeutet Freundschaft. Dieses Netzwerk drückt das Muster Freundschaft von verschiedenen Personen in einem Netzwerk aus. Die Knoten mit der Aufschrift 1 und 34 sind offensichtlich sehr beliebte Spieler. Das Karate Klub Netzwerk ist offensichtlich sehr klein. Auch in dieser Domäne existieren sehr große Netzwerke und auch sehr interessante Datenquellen für die empirische Erforschung von Sozialen Netzwerken, hier sei auf diverse Online Dienste wie beispielsweise Facebook oder Google+ verwiesen.

2.5.3 Informationsnetzwerke

Informationsnetzwerke sind von Menschenhand geschaffene Artefakte. Ein solches Netzwerk beinhaltet Elemente aus Daten, die in einer Form miteinander verbunden sind. Das World Wide Web (WWW) ist ein typisches Informationsnetzwerk mit Web-Seiten als Knoten und mit Hyperlinks als Kanten. Es gibt sehr viele Informationsnetzwerke, die auch soziale Aspekte beinhalten. Beispiele hierfür sind das Zitierungsnetzwerk von wissenschaftlichen Publikationen, Netzwerke von der E-Mail-Kommunikation oder Netzwerke auf Online-Kontaktnetzwerk Diensten wie Facebook[2], LinkedIn[3] oder Google+[4]. [Newman, 2010] konstatiert, dass die Klassifikation in die vorgestellten Klassen unscharf (*fuzzy*) ist und es immer

[2]Facebook: `http://www.facebook.com/`
[3] LinkedIn: `http://www.linkedin.com/`
[4]Google+: `https://plus.google.com/`

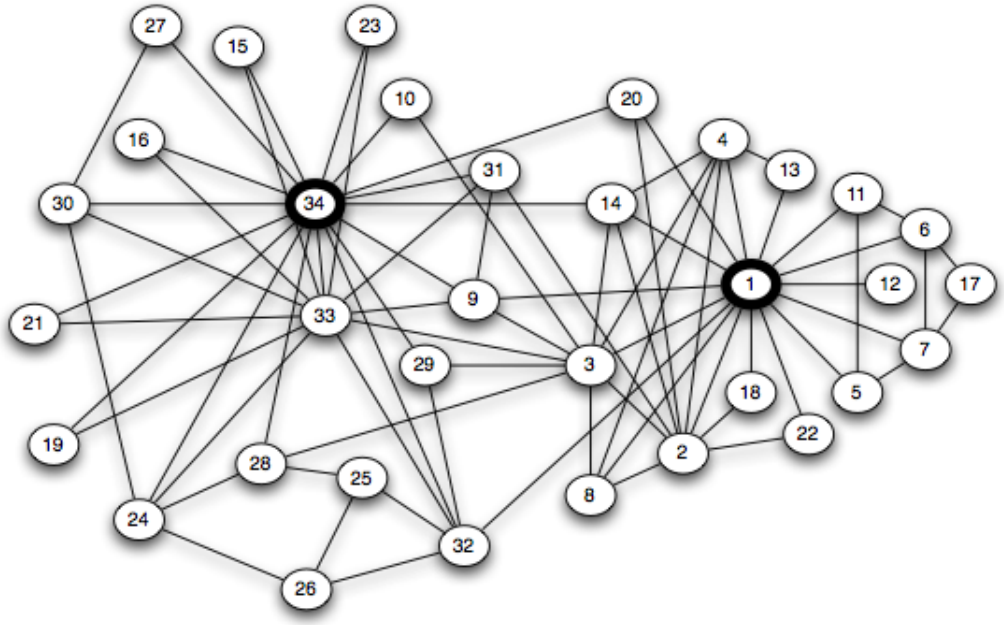

Abbildung 2.12: Ein soziales Netzwerk (Freundschaftsnetzwerk) eines Karate Clubs mit 34 Personen [Easley und Kleinberg, 2010]

Beispiele von Netzwerken gibt, die in mehreren Klassen berücksichtigt werden können.

Die folgenden beiden Informationsnetzwerke, beide sind sehr unterschiedlich, sind für die weitere Betrachtung von Interesse:

- World Wide Web (WWW)
- Zitierungsnetzwerk von wissenschaftlichen Publikationen (*citation network*)

2.5.3.1 World Wide Web (*WWW*)

Die Knoten im WWW sind Web-Seiten. Web-Seiten bestehen aus Informationen und können Text, Bilder usw. beinhalten. Die Kanten im WWW sind sogenannte Hyperlinks, die verschiedene Seiten miteinander verbinden. Mit einem Hyperlink kann man von einer Seite zu einer anderen Seite *navigieren*. Abbildung 2.13 zeigt eine beispielhafte Linkstruktur im WWW. In dieser Abbildung ist ersichtlich, dass ein Hyperlink immer nur in eine Richtung verläuft, d.h. man kann von einer Seite zu einer anderen Seite nur in eine Richtung navigieren, sofern kein Hyperlink zurück existiert. Das World Wide Web ist ein *gerichtetes Netzwerk*. Das World Wide Web ist zyklisch, weder gibt es Einschränkungen bzgl. geschlossenen Schleifen, noch ist eine Ordnung von Web-Seiten vorhanden [Newman, 2003].

Ein interessanter Aspekt ist, dass man das World Wide Web, wie viele gerichtete Netzwerke, mit dem Bow-Tie (*bow tie*) Diagramm beschreiben kann (vgl. [Dorogovtsev und Mendes, 2004], [Broder, 2000]). Dieses Diagramm hat eine typischer Form einer Fliege und ist in Abbildung 2.14 ersichtlich. Typischerweise bestehen gerichtete Netzwerke aus einer großen, stark verbundenen Komponente (*strongly connected component*) und vielen kleinen Komponenten, wobei jedes dieser Komponenten eine Eingangs-Komponente (*in-components*) und Ausgangs-Komponente (*out-components*) besitzt. Jede Eingangs-Komponente

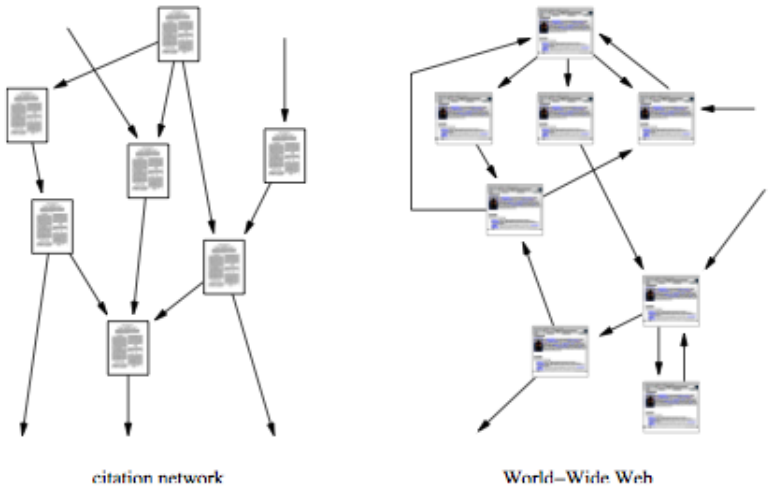

Abbildung 2.13: Zwei verschiedene Informationsnetzwerke. Links sieht man die Struktur des Zitierungsnetzwerk von akademischen Publikationen. Rechts sieht man eine Linkstruktur des World Wide Web [Newman, 2003].

und Ausgangs-Komponente hat eine stark verbundene Komponente als Teilmenge. Die größte stark verbundene Komponente mit zugehöriger Eingangs- und Ausgangs-Komponente belegt einen signifikanten Anteil des ganzen Netzwerkes [Newman, 2010]. In Abbildung 2.14 ist eine typisches Bow-Tie Diagram des Word Wide Web zu sehen. Man kann von jeden Knoten der Eingangs-Komponente über den Kern der SCC-Komponente zu jeden Knoten der Ausgangs-Komponente gelangen. Die sogenannte Tendrils (vgl. Abbildung 2.14) beinhalten eine Menge von Knoten, die von Teilen der Eingangs-Komponente aus erreichbar sind. Die Knoten der Tendrils können aber auch Teile der Ausgangs-Komponenten erreichen.

2.5.3.2 Zitierungsnetzwerk von wissenschaftlichen Publikationen

Die Knoten dieses Netzwerkes können Artikel, Patente oder andere Publikationstypen sein. Die gerichteten Kanten in diesem Netzwerk beschreiben das Konzept der Zitierung. Eine Publikation verweist durch eine Zitierung auf eine schon vorhandene Publikation. In Abbildung 2.13 ist eine typische Struktur eines Zitierungsnetzwerkes abgebildet. Ein Zitierungsnetzwerk ist ein gerichtetes Netzwerk. Im Gegensatz zum World Wide Web ist ein Zitierungsnetzwerk jedoch azyklisch. Dieser Umstand ist leicht nachvollziehbar, da eine Publikation nur eine Publikation zitieren kann, die schon geschrieben wurde [Newman, 2003].

2.5.4 Biologische Netzwerke

In der Biologie werden Netzwerke verwendet, um Muster der Interaktion zwischen bestimmten biologischen Elementen zu repräsentieren. Gleichzeitig können verschiedene biologische Systeme als Netzwerk abgebildet werden. Knoten können in biologischen Netzwerken Stoffwechselprodukte, Neuronen oder eine Spezies darstellen. Kanten können Metabolische Reaktionen, Synapsen oder eine Raubtier-Beute-Beziehung darstellen [Newman, 2010].

Newman [2010] führt unter anderem die folgenden Beispiele als typische Biologische Netzwerke an:

- Biochemische Netzwerke
  - Stoffwechsel Netzwerke

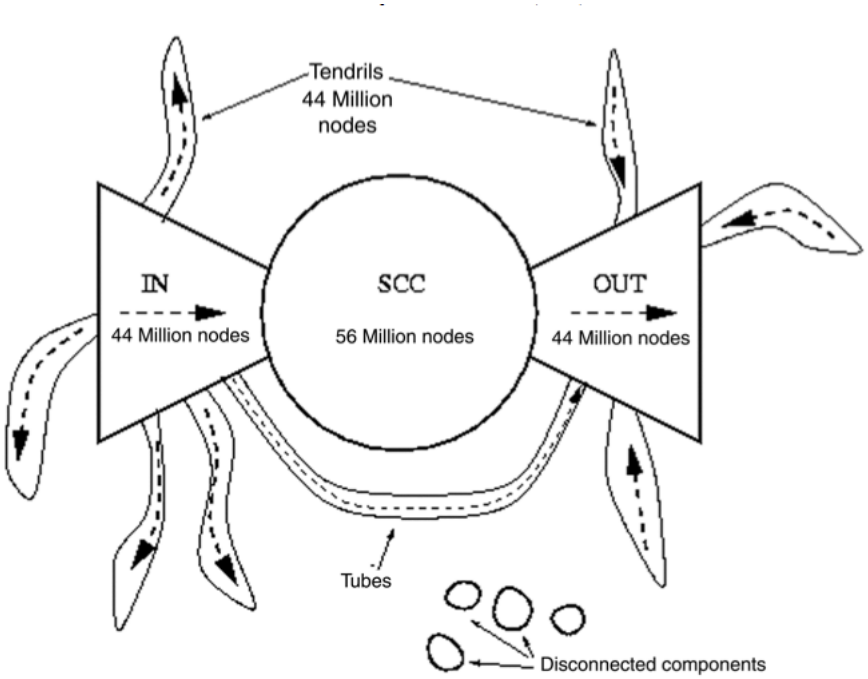

Abbildung 2.14: Bow-Tie Diagramm der Komponenten in einem gerichteten Netzwerk. Darstellung zeigt die Vernetzung im World Wide Web [Broder, 2000]

  - Protein-Protein Interaktionsnetzwerke

- Neuronale Netzwerke
- Ökologische Netzwerke
  - Ernährungsnetze (*food web*)

Abbildung 2.15 zeigt die Visualisierung eines Protein-Protein Interaktionsnetzwerk aus der Klasse der biochemischen Netzwerke. Die Knoten in diesem Netzwerk repräsentieren Proteine. Proteine können mit anderen Proteinen interagieren. Genau dann wenn zwei Proteine mit einander interagieren, werden sie mit einer ungerichteten Kante miteinander verbunden. Die Knoten in diesem Netzwerk haben verschiedene Farben, damit wird der phänotypische Effekt repräsentiert [Barabási und Oltvai, 2004]. Die Farbe Rot bedeutet tödlich, Gelb bedeutet ungefährlich, Orange steht für langsames Wachstum und die Farbe Gelb steht für unbekannt.

2.6 Stanford Network Analysis Package (SNAP)

Das Stanford Network Analysis Package (SNAP) ist eine Netzwerk Analyse Library und wurde von Jure Leskovec im Rahmen seiner PhD Studien entwickelt [SNAP, 2011b] . SNAP wurde für die *effiziente* Analyse von sehr großen Netzwerken ausgelegt. Große Netzwerke können auf Millionen von Knoten und Milliarden von Kanten skalieren. Diese Größe stellt bei Berechnungen und bei der Analyse eine große Herausforderung dar. Neben der Effizienz wurde auch auf die einfache Erweiterbarkeit der Library Wert gelegt. SNAP wurde mit der Programmiersprache `C++` entwickelt. SNAP baut auf der GLib Library auf. GLib bietet eine Menge von Basis Datenstrukturen wie Smart-Pointers, Vektoren, Hashtabellen und String. GLib ist ähnlich zu der Standard Template Library (STL) (vgl. [SNAP, 2011b]).

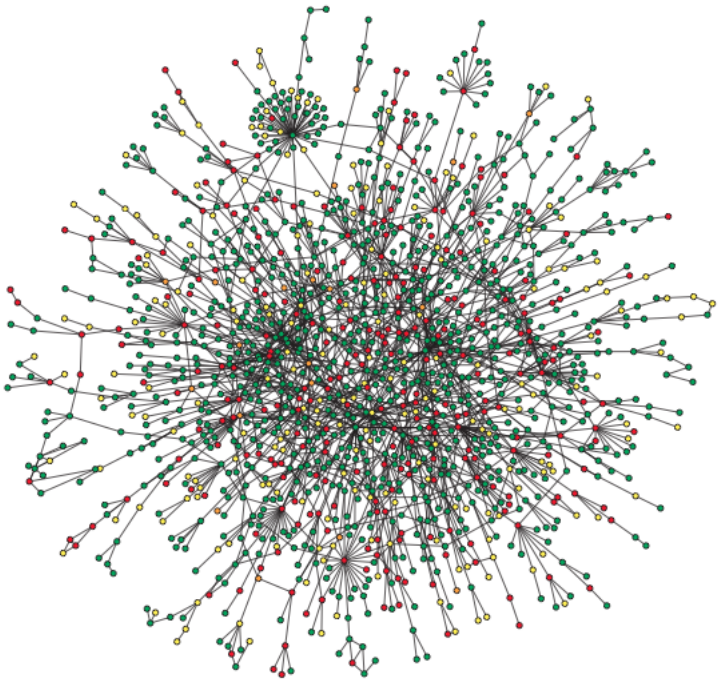

Abbildung 2.15: Darstellung eines skalenfreien Netzwerk [Barabási und Oltvai, 2004]

2.6.1 Graph und Netzwerk Typen

Die SNAP-Library unterscheidet zwischen Graphen und Netzwerken folgendermaßen [SNAP, 2011a]:

Graph: Ein Graph beschreibt eine reine Topologie und speichert nur Informationen, wie Elemente in einem Netzwerk miteinander in Verbindung stehen.

Netzwerk: Ein Netzwerk ist ein Graph mit der Erweiterung, dass Knoten und Kanten mit Daten versehen werden können. Knoten oder Kanten können mit Datentypen versorgt werden. Durch die Verwendung von Temaplates bietet SNAP für Netzwerke generische Container, mit der Möglichkeit per Template-Parametern Knoten und Kanten mit Datentypen zu versorgen. Somit ist es mit dieser Library möglich, verschiedenste Netzwerke zu implementieren ohne der Einschränkungen, welche Daten Knoten und Kanten beinhalten.

Die Unterscheidung von Graphen und Netzwerken hat zur Folge, dass Graphen als generische Container abbildet werden, mit deren Hilfe man nur Elemente und deren Vernetzung untereinander beschreiben kann. Wie in Abschnitt 2.2.2 beschrieben, gibt es verschiedene Arten von Graphen. Die SNAP-Library unterstützt die folgenden drei Typen [SNAP, 2011a]:

`TUNGraph`: Ein einfacher ungerichteter Graph kann mit diesem Typ abgebildet werden. Multikanten können mit diesem Typ nicht abgebildet werden.

`TNGraph`: Dieser Typ repräsentiert einen einfachen gerichteten Graphen. Multikanten werden durch diesen Typ nicht unterstützt.

`TNEGraph`: Mit diesem Typ kann man einen gerichteten Multigraphen abbilden. Ein Multigraph erlaubt, dass zwei Knoten durch mehrere Kanten miteinander verbunden werden.

```
1  PUNGraph Graph = TUNGraph::New();
2  Graph->AddNode(1); Graph->AddNode(2);
3  Graph->AddNode(3); Graph->AddNode(4);
4  Graph->AddNode(5); Graph->AddNode(6);
5  Graph->AddNode(7); Graph->AddNode(8);
6  Graph->AddNode(9);
7
8  Graph->AddEdge(1,2); Graph->AddEdge(1,3);
9  Graph->AddEdge(1,4); Graph->AddEdge(1,5);
10 Graph->AddEdge(1,6); Graph->AddEdge(1,7);
11 Graph->AddEdge(1,8); Graph->AddEdge(2,3);
12 Graph->AddEdge(3,4); Graph->AddEdge(6,7);
13 Graph->AddEdge(7,8); Graph->AddEdge(7,9);
14 Graph->AddEdge(8,9);
```

Listing 2.1: Das einfache ungerichtete Netzwerk aus Abbildung 2.4 kann mit der SNAP-Library folgendermaßen implementiert werden. Zuerst werden die 9 Knoten hinzugefügt, anschließend werden entsprechend der Abbildung die Knoten durch die 13 Kanten miteinander verbunden.

Durch die Verwendung von Netzwerk-Typen kann man Knoten und Kanten mit Daten versehen. Die SNAP-Library unterstützt die folgenden Netzwerk-Typen, mit entsprechenden Template-Parametern für Knoten und Kanten [SNAP, 2011a]:

`TNodeNet<TNodeData>`: Ein Netzwerk, das durch einen gerichteten Graphen modelliert wird, wobei jeder Knoten vom Typ `TNodeData` ist.

`TNodeEDatNet<TNodeData,TEdgeData>`: Konnektivität wird durch einen gerichteten Graphen beschrieben. Knoten sind vom Typ `TNodeData` und Kanten sind vom Typ `TEdgeData`.

`TNodeEdgeNet<TNodeData,TEdgeData>`: Im Kern wird bei dieser Ausprägung ein Mutligraph verwendet. Knoten und Kanten können mit den beiden Typen `TNodeData` und `TEdgeData` parametrisiert werden.

`TBigNet<TNodeData>`: Eine spezielle Implementierung des Netzwerk-Types `TNodeNet`. Dieser Typ ist durch eine Speicher-Effiziente Implementierung für Netzwerke mit Milliarden von Kanten gedacht.

Auflistung 2.1 zeigt, wie man das einfache Netzwerk aus Abbildung 2.4 mit der SNAP-Library abbilden kann. Die SNAP-Library ist gekennzeichnet durch die Verwendung von sogenannten Smart-Pointer. Smart-Pointer bieten den großen Vorteil, dass man die Ressourcen nicht selbst freigeben muss. Ein Smart-Pointer wird gelöscht, wenn dieser nicht mehr verwendet wird. In der ersten Zeile bei Auflistung 2.1 sieht man die Namenskonvention die SNAP für Typen und Objekte verwendet. Der Prefix `T` wird immer für Typen verwendet. Der Prefix `P` kennzeichnet, dass es sich um einen Pointer handelt [SNAP, 2011a].

2.6.2 Knoten und Kanten Iteratoren

Die SNAP-Library bietet Iteratoren für Knoten und Kanten [SNAP, 2011a]. Mit einem Knoten-Iterator kann man sehr einfach durch die einzelnen Knoten eines Netzwerkes iterieren. Durch die Verwendung dieser Iteratoren hat man die sehr mächtige Möglichkeit, Algorithmen zu implementieren, die unabhängig von dem Typ und der tatsächlichen Implementierung des Netzwerkes sind. Alle SNAP-Library

```
1  // create a directed random graph on 100 nodes and 1k edges
2  PNGraph Graph = TSnap::GenRndGnm<PNGraph>(100, 1000);
3
4  // traverse the nodes
5  for (TNGraph::TNodeI NI = Graph->BegNI(); NI < Graph->EndNI(); NI++) {
6    printf("node id %d with out-degree %d and in-degree %d\n", NI.GetId(), NI.
         GetOutDeg(), NI.GetInDeg());
7  }
8
9  // traverse the edges
10 for (TNGraph::TEdgeI EI = Graph->BegEI(); EI < Graph->EndEI(); EI++) {
11   printf("edge (%d, %d)\n", EI.GetSrcNId(); EI.GetDstNId());
12 }
13
14  // we can traverse the edges also like this
15  for (TNGraph::TNodeI NI = Graph->BegNI(); NI < Graph->EndNI(); NI++) {
16   for (int e = 0; e < NI.GetOutDeg(); e++) {
17     printf("edge (%d %d)\n", NI.GetId(), NI.GetOutNId(e));
18   }
19  }
```

Listing 2.2: Anwendung von Knoten und Kanten Iteratoren. Beispiel wurde entnommen aus [SNAP, 2011a].

Graph-Typen und Netzwerk-Typen haben für Kanten und Knoten geeignete Iteratoren definiert. Auflistung 2.2 zeigt die Anwendung von den unterschiedlichen Iteratoren in einem gerichteten Graphen. Der erste Teil bei Auflistung 2.2 beschreibt ein Beispiel, indem durch die einzelnen Knoten eines Netzwerkes iteriert wird, wobei für jeden Knoten die korrespondierende *ID* (`NI.GetId()`) sowie der Eingangsgrad `NI.GetInDeg()` und der Ausgangsgrad (`NI.GetOutDeg()`) bestimmt wird. Der zweite Teil der Auflistung 2.2 ab Zeile 25 zeigt die Anwendung eines Kanten-Iterators. Hier wird gezeigt, wie man aufgrund einer Kante auf die Knoten zurückschließen kann. Nachdem es sich um eine gerichtete Kante handelt, kann man mit `EI.GetSrcNId()` und `EI.GetDstNId()` implizit die Richtung der Kante ermitteln. Der dritte Teil der Auflistung 2.2 ab Zeile 30 illustriert, wie man nur mit einem Knoten-Iterator durch die einzelnen Kanten eines Graphen iterieren kann.

Die sehr mächtigen Iteratoren für Graphen und Netzwerke, die in der SNAP-Library zur Verfügung gestellt werden, bieten zusätzlich beispielsweise folgende Möglichkeiten [SNAP, 2011a]:

- Knoten haben immer eine eindeutige *ID*. Auf diese Kennung kann man mit einem Knoten-Iterator zugreifen.
- Mit einem Knoten-Iterator kann man den Eingangsgrad und den Ausgangsgrad eines Knotens bestimmen.
- Mit einem Knoten-Iterator kann man bestimmen, ob ein Knoten n ein Nachbar ist.
- Handelt es sich um einen Netzwerktyp, dann hat man auf die Datentypen, die mit einem Knoten oder mit einer Kante assoziert sind, Zugriff.

2.6.3 Berechnung struktureller Eigenschaften

Ein weiterer Aspekt dieser Library liegt in der Analyse von strukturellen Eigenschaften von vernetzten Strukturen (Graphen und Netzwerke). Mit der Library kann man sehr komplexe Algorithmen abbilden.

Im Funktionsumfang der Library sind unter anderem folgende Basisfunktionen bereitgestellt, die für jeden Graph und Netzwerktyp anwendbar sind [SNAP, 2011a]:

- Verteilung verbundener Komponenten
- Gradverteilung
- Eigenvektoren der Adjazenzmatrix
- Durchmesser des Graphen
- Anzahl der Triaden
- Cluster-Koeffizient

2.6.4 Datensätze

Netzwerke werden durch Datensätze beschrieben. Datensätze können unterschiedlichste Formate haben. Die Bandbreite reicht von einer einfachen Textdatei bis hin zu speziellen XML-Formaten. Die SNAP-Library unterstützt das Einlesen von einer Vielzahl von Datensätzen. Gleichzeitig bietet die SNAP-Library auch Schnittstellen für die Erstellung von Datensätzen. Datensätzem die Netzwerke der folgenden Form beschreiben, werden unterstützt [SNAP, 2011c]:

- gerichtete Netzwerke
- ungerichetete Netzwerke
- Bipartite Netzwerke
- Multigraphen

Es werden auch Netzwerke unterstützt, die Knoten oder Kanten mit entsprechender Kennzeichnung besitzen. Ein Knoten oder eine Kante kann beispielsweise mit einem Gewicht versehen worden sein.

2.6.5 Visualisierung

Die SNAP-Library grenzt sich stark von Aspekten der Visualisierung von Netzwerkdaten oder Netzwerk-Metriken ab und liefert hier keine eigenständigen Implementierung, die im Funktionsumfang der Library zu finden wäre. Sehr wohl bietet die SNAP-Library aber diesbezüglich Schnittstellen zu den folgenden zwei Tools [SNAP, 2011b]:

GnuPlot: GnuPlot wird für die Erstellung von Plots (graphische Darstellungen) und Diagrammen verwendet. Dadurch ist es möglich charakteristische Darstellungen von strukturellen Eigenschaften von Netzwerken zu erhalten. Beispiele hierfür wären die graphische Dartstellung von:

- Gradverteilung
- Cluster-Koeffizient
- etc.

GraphViz: GraphViz wird von SNAP dazu verwendet, um einfache Graphen darzustellen.

Kapitel 3

Web Engineering

> *“ There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult. ”*
>
> [Hoare [1981]]

Dieses Kapitel befasst sich mit der systematischen Entwicklung von Web-Anwendungen. Die zunehmende Verbreitung und der steigende Komplexitätsgrad stellt bei der Entwicklung von Web-Anwendungen eine nicht-triviale Herausforderung dar. Es zeigt sich, dass bei reinen Ad-Hoc-Vorgehensweisen notwendige Qualitätsmerkmale nicht erreicht werden können [Kappel et al., 2004]. Eine Web-Anwendung stellt ein komplexes Softwaresystem dar, deren Entwicklung eine methodisch fundierte Herangehensweise erfordert. Web Engineering hat eine ingenieursmässige Entwicklung von Web-Anwendungen zum Ziel, befasst sich jedoch auch mit der wissenschaftlichen Erforschung von systematischen und quantifizierbaren Ansätzen, die bei der Erstellung von Web-Anwendung angewandt werden sollen [Kappel et al., 2004].

Im Rahmen dieses Kapitels wird ein Überblick über die Disziplin Web Engineering geliefert. Abschnitt 3.1 startet mit einem kurzen Überblick über die Evolution des World Wide Web (WWW) und zeigt, in welchem Kontext Web-Anwendungen zu sehen sind. Die Erstellung von Web-Anwendungen stellt eine neue Herausforderung an die Disziplin Software Engineering dar. In Abschnitt 3.2.1 werden relevante Aspekte fokussiert und wichtige Differenzen heraus gearbeitet. Die Motivation von Web Engineering, sowie grundlegende Definitionen und der Konnex zu anderen Disziplinen werden in Abschnitt 3.2 beschrieben. Abschnitt 3.2.2 zeigt ein Schema, mit deren Hilfe Web-Anwendungen klassifiziert werden können. Abschnitt 3.2.3 behandelt Charakteristika von Web-Anwendungen.

Anforderungen an Software sind im Kontext von Web-Anwendungen immer auch Anforderungen an die zugrunde liegende Architektur. Es zeigt sich demnach, dass die Qualität einer Web Anwendung sehr stark von der zugrunde liegenden Architektur bestimmt wird. Abschnitt 3.3 behandelt alle relevanten Aspekte die mit der Architektur von Web-Anwendungen einhergehen. Dieses Kapitel wird mit Abschnitt 3.5, einem Überblick über Web Applikations Frameworks (WAF), beendet. In diesem Abschnitt werden Vorteile und Aspekte von Frameworks, die zur Erstellung von Web-Anwendungen verwendet werden sollen, zusammengefasst.

3.1 Evolution des Web und die Auswirkungen auf Web-Anwendungen

In Kapitel 2 wurde im Abschnitt 2.5.3 das World Wide Web (WWW) aus netzwerktheoretischer Perspektive als ein Informationsnetzwerk beschrieben. Sogenannte Web-Seiten sind die Knoten dieses Netzwer-

kes und repräsentieren elektronische Dokumente. Hyperlinks verbinden einzelne Web-Seiten miteinander und sind die Kanten dieses Netzwerkes. Das WWW wird auch einfach als Web bezeichnet. Das Web ist aber nicht Synonym mit dem Internet. Das Internet ist ein technologisches Netzwerk (vgl. Kapitel 2, Abschnitt 2.5.1), dass das Web als Dienst ermöglicht. Beide Netzwerke, dass Web sowie das Internet, sind eine der einflussreichsten Entwicklungen der Menschheit.

Tim Berners-Lee gilt als der Erfinder des WWW. Seine Vision kann durch das folgende Zitat wiedergegeben werden [Berners-Lee, 1998]:

> *"The dream behind the Web is of a common information space in which we communicate by sharing information."*

Diese Vision beruht unter anderem auf den folgenden Grundsätzen [Berners-Lee et al., 1992]:

- Das Web als System soll unabhängig davon sein, wo und wie Informationen gespeichert sind. Ferner soll das Web unabhängig von dem eingesetzten System sein, dass Informationen organisiert.
- Das Web ist aufgebaut durch Dokumente, die aufeinander per Links verweisen können. Ein Link ist eine unidirektionale Verbindung.
- Das Web unterliegt also dem Hypertext-Paradigma. Dokumente können Links beinhalten. Folgt man einen Link, dann kommt man zu einem anderen Dokument.
- Das Hypertext-Modell muss um eine Suche erweitert werden, damit das Web praktikabel ist.

Heutzutage wird das Web *inter alia* als ein Netzwerk aus verlinkten Dokumenten charakterisiert. Die Mächtigkeit des Web liegt einerseits in seiner Einfachheit, andererseits auch in dem Faktum, dass keine zentrale Kontrolle im Web vorgesehen ist, begründet. Damit die Vision auch tatsächlich implementiert werden konnte, war es notwendig, die folgenden drei Kerntechnologien aufzubauen und einzusetzen [Berners-Lee und Fischetti, 1999]:

- Uniform Resource Identifier (URI): URI ist ein Standard für die Identifikation von realen und virtuellen Ressourcen. Beliebige Ressourcen sollen durch eine URIs eindeutig identifiziert werden können. Eine Ressource kann jedes beliebige Objekt sein, dass man verlinken bzw. eindeutig referenzieren kann. Eine Ressource darf wiederum Referenzen zu anderen Ressourcen enthalten [Berners-Lee et al., 2005]. Mit diesem Konzept soll die Funktionsweise von Hyperlinks ermöglicht werden. Ein Uniform Resource Locator (URL) ist eine Unterart von URIs.
- Hypertext Transfer Protocol (HTTP): HTTP ist ein Standard Transport Protokoll. Es ist ein zustandsloses Protokoll, das eine Interaktion zwischen verschiedenen Agenten ermöglicht [Fielding et al., 1999]. Dadurch soll beispielsweise erreicht werden, dass ein Web-Browser (Client) Informationen von einem Web-Server (Server) anfordern kann.
- Hypertext Markup Language (HTML): HTML ist ein Darstellungsformat für die Beschreibung von Ressourcen. Es ermöglicht eine Strukturierung von Inhalten wie Text, Bildern und Hyperlinks. Durch die Verwendung dieser Auszeichnungssprache soll die Darstellung von dem Inhalt eines Dokumentes entkoppelt werden.

Diese Protokolle legten die Basis dafür, dass der Austausch von Informationen zwischen heterogenen Computern und Systemen möglich wurde [Berners-Lee und Fischetti, 1999]. Im Laufe der Zeit sind weitere *Standards* hinzugekommen, um vorhandenen Anforderungen zu genügen und gehen gleichzeitig mit

einem technischen Fortschritt einher. Cascading Style Sheets[1] (CSS) oder das Document Object Model[2] (DOM) sind nur einige hier zunennende Entwicklungen. Eine wichtige Institution ist das World Wide Web Consortium[3] (W3C). Diese Institution definiert und spezifiziert Standards für das Web, mit dem Ziel das gesamte Potenzial des WWW auszuloten.

Das Web hielt im Jahr 1991 Einzug in unsere Gesellschaft. Seit diesem Zeitpunkt ist das Web öffentlich und weltweit verfügbar. Ab diesem Zeitpunkt hat sich das WWW auch rasant weiterentwickelt. Heutzutage gibt es kaum einen Bereich des täglichen Lebens, indem das Web nicht Einzug gehalten hat [Kappel et al., 2004]. Das Web ist ein System bestehend aus Systemen. Das Web ist offen und dynamisch. Das Web ist eine technische Infrastruktur und unterstützt soziale Prozesse (Linking, Tagging). Das WWW ist kein statisches Netzwerk, sondern bringt immer neue Phänomene hervor. Sogenannte Web Seiten werden zu Web-Anwendungen. Web-Anwendungen entwickeln sich auch stetig weiter und stellen immer weitere Anforderungen, wodurch *prima facie* ein kontinuierlich steigender Komplexitätsgrad begründet ist [Rossi et al., 2007]. Das Web kann als ein großer Beweis für die Fähigkeit von Netzwerk Effekten angesehen werden. Dieses Netzwerk bringt nicht nur immer neue Phänomene hervor, sondern entwickelt sich im Sinne einer Evolution auch selbst immer weiter [Rossi et al., 2007]. Unter anderem konstatiert Rossi et al. [2007] die folgende Klassifikation des Web:

- Web 1.0
- Web 2.0
- Mobile Web
- Semantische Web

Hierbei handelt es sich um keine abgeschlossene Einteilung. Trends wie Rich Internet Applications (RIAs) oder die des Real-Time Web –beides Beispiele für den Einsatz bestimmter Praktiken und Technologien die wiederum in Web-Anwendungen zum Einsatz kommen– oder die Vision des Web of Things, spannen hier einen weiten Spielraum für zukünftige Entwicklungen und einer kontinuierlichen Evolution auf.

Rossi et al. [2007] konstatiert, dass bei der Klassifikation folgende Dimensionen und Perspektiven eine Rolle spielen:

- Anzahl und Wachstum von Web Seiten
- Anzahl der Web User
- Verhalten und Rolle von Benutzern beispielsweise bzgl. Seitenbesuche
- Funktionalität und Möglichkeiten der Interaktion von Web-Anwendungen
- Technologien, die zur Erstellung von Web-Anwendungen benötigt werden
- Sozialer, gesellschaftlicher und wirtschaftlicher Einfluss

[1] CSS: `http://www.w3.org/Style/CSS/`
[2] DOM: `http://www.w3.org/DOM/`
[3] W3C: `http://www.w3.org/`

3.1.1 Web 1.0

Das Web 1.0 ist ein künstlich eingeführter *terminus technicus* und bezeichnet das Web in seiner anfänglichen Entwicklung als ein großes Informationssystem von vernetzten Web-Seiten, mit der in Folge beschrieben Eigenschaften und Einschränkungen [Rossi et al., 2007]: Die Interaktionsmöglichkeiten von Benutzern sind nicht sehr stark ausgeprägt. In dieser Phase konnte man das Web in das statische Web (*shallow web*) und in das dynamische Web (*deep web*) einteilen. Daraus folgt, dass sehr viele statische Web Seiten vorhanden sind, die Informationen über Services usw. anbieten. Web Seiten sind in diesem Kontext einfache HTML Seiten. Durch den Einsatz von Datenbanken und durch verbesserter Technologie entstand das Bild des dynamischen Web. Web Seiten werden dynamisch generiert. Hierbei handelt es sich um virtuelle Ressourcen. Die Generierung von Inhalt ist einseitig und der Benutzer ist in diesem Prozess nicht inkludiert.

Ein weiteres Charakteristika des Web 1.0 ist die Ausschließlichkeit von Informationen. Daraus folgt, dass ein Benutzer eine bestimmte Web-Seite besuchen muss, um Zugriff auf bestimmte Informationen zu erhalten. Ferner kann das Web 1.0 als eine Einteilung bzw. Katalogisierung in brauchbare Verzeichnisse –im Sinne einer Taxonomie– verstanden werden [Strickland, 2007]. Die Rolle der Benutzer ist beschränkt auf das Konsumieren von Informationen, bzw. darauf, dass eine kleine Anzahl von Benutzern irgendwo im Netz ihre eigene Homepage haben. Zusammenfassend ist das Web 1.0 in seiner Entwicklung gekennzeichnet durch einen Mangel an Kontext, Interaktionsmöglichkeiten mit Benutzern und Skalierbarkeit.

3.1.2 Web 2.0

Das Web 2.0 ist ein Schlagwort und steht für eine Sammlung von Technologien, Geschäfts-modellen und sozialen Trends (vgl. [O'Reilly, 2005], [Rossi et al., 2007]). Web 2.0 steht für eine hoch interaktive und dynamische Plattform – und ist weit mehr als einfache statische oder dynamisch generierte Web Seiten. Es erweitert das Web 1.0 um eine Reihe von interaktiver und kollaborativer Elemente. Das Web hat sich mittlerweile zu einer *Applikations Platform* entwickelt. Web Seiten haben sich auf dieser Plattform zu Web Anwendungen entwickelt. Fokus liegt auf der Macht der Gemeinschaft (*power of the community*) bzgl. dem Erstellen und Validieren von Inhalten. Der Fokus liegt auf Gruppen und Nutzer und nicht mehr so stark auf Unternehmen, wie im Web 1.0 [Strickland, 2007]. Nutzer können nicht nur Inhalt konsumieren, sondern auch aktiv Inhalt generieren und zur Verfügung stellen. Im Gegensatz zu der Taxonomie des Web 1.0, bieten viele Web 2.0 Applikationen eine freie Form der Organisation und Klassifikation, die es erlaubt Benutzern mittels Tagging nach ihren Kategorien, Inhalte zu organisieren. Der Nutzen der kollektiven Intelligenz und die Datenvielfalt sind nur einige Schlagwörter, die diese Entwicklungen mit sich bringen [O'Reilly, 2005].

Der Einsatz von neuen Technologien wie Asynchronous JavaScript and XML (AJAX), oder Konzepten wie Tagging, Wikis und Blogs, ermöglichten es, dass das Web hoch interaktiv wurde. Gleichermaßen haben sich Web Anwendungen in Richtung Rich-User-Interfaces entwickelt, sodass Web-Anwendungen immer ähnlicher zu Desktop-Anwendungen werden [Rossi et al., 2007].

Ein weiterer Paradigmenwechsel ist dadurch gegeben, dass viele Dienste sogenannte *Hooks* als Service anbieten, um auf vorhandene Inhalte zugreifen zu können [Strickland, 2007]. Ein Mashup ist eine Verzahnung von Angeboten, durch eine Kombination von bereits bestehenden Inhalten wodurch neue Inhalte generiert werden können. Der Zugriff auf Inhalte erfolgt über offene Schnittstellen, sogenannte APIs[4] oder per RSS[5] als Feeds. Ein API wird per *Web Service* zur Verfügung gestellt. Ein Web Service

[4] Advanced Programming Interface
[5] Realy Simple Syndication

ermöglicht eine Maschine-Maschine-Kommunikation und kann innerhalb einer Programmiersprache als API verwendet werden. Web Services können als SOAP[6] Web Services, jedoch mit einen komplizierten WS-* Protokollstack [Wilde und Gaedke, 2008], oder als REST[7] Service (RESTful) basiert sein [O'Reilly, 2005].

Durch die Möglichkeit der Wiederverwendung von Inhalten und den offenen Zugriff per Schnittstellen hat sich das Web nicht nur in eine Applikations Plattform verwandelt, sondern hat auch die Rolle einer großen verteilten Datenbank eingenommen [O'Reilly, 2005].

Strickland [2007] sieht in dieser Entwicklung einen Mangel hinsichtlich der Möglichkeiten der Personalisierung sowie Problemen bei der Interoperabilität zwischen verschiedenen Systemen und Portabilität.

3.1.3 Mobile Web

Das Web ist *ubiquitär*. Mit steigender Konvergenz ist das Web überall vorhanden und dringt in immer mehr Bereiche ein. Die technische Entwicklung und der rasante Fortschritt in den Bereichen des Mobile Computing und in der kabellosen Kommunikation brachte mit sich, dass auf das WWW auch von mobilen Geräten aus (Smart Phones usw.) und von einer immer größere Anzahl von Nutzern zugegriffen werden kann. Das World Wide Consortium hat für das Web und Mobile Geräte die Mobile Web Initiative[8] eingerichtet. Das Ziel dieser Initiative ist, dass das Web auf so vielen Geräten wie möglich verfügbar ist. Web Anwendungen sind auch im Mobile Web verfügbar. Einerseits werden manche Web-Anwendungen für das Mobile Web adaptiert, andererseits kann eine Anwendung vom Funktionsumfang soweit reduziert werden, dass die Web-Anwendung auch auf dem mobilen Endgerät lauffähig ist (*graceful degradation*).

Die Entwicklung des Mobile Web ist interessant, aufgrund der Anforderungen und Herausforderungen, die an Web-Anwendungen in diesem Kontext gestellt werden und die sich vor allem von klassischen Web Anwendungen unterscheiden. Neue Aspekte wie die Integration von Standortbezogene Diensten (*location aware services*) sowie Kontextsensitivität (*context awareness*) und Aspekte der Personalisierung gewinnen an Bedeutung Rossi et al. [2007].

3.1.4 Semantische Web

Das Semantische Web ist eine Erweiterung des bestehenden Web. Informationen soll eine definierte Bedeutung zugeschrieben werden. Dadurch sollen Mensch und Computer in Kooperation arbeiten können. Ein Überfluss von Informationen im Web macht es den Menschen immer schwerer, relevante Informationen zu finden. Der Inhalt heutiger Web-Anwendungen ist durch natürliche Sprache repräsentiert. Das Web ist heutzutage so konstruiert, dass Informationen für den Menschen dargestellt werden können [Rossi et al., 2007]. Daraus folgt, dass die Inhalte des Web derzeit nur von Menschen verstanden und interpretiert werden können. Die Idee des Semantischen Web ist, dass intelligente Maschinen Informationen im Web interpretieren und automatisch weiterverarbeiten können, um beispielsweise relevante Informationen für Nutzer ordnen zu können [Behrendt, 2004]. Dazu ist es aber notwendig das bereitgestellte Informationen im Web von Maschinen verarbeitet werden können. Dieses Problem soll das Semantische Web lösen. Sofern Maschinen Informationen verarbeiten können, ist es möglich Verbindungen zwischen anderen Informationen herzustellen oder logische Fragen zu beantworten [Rossi et al., 2007].

[6] SOAP in der Version 1.1 war ein Akronym für *Simple Object Access Protocol* und wurde als W3C Standard implementiert. Seit der Version SOAP 1.2 kann das Akronym von SOAP auch als *Service Oriented Architecture Protocol* gelesen werden.

[7] Representational State Transfer

[8] Mobile Web Initiative: `http://www.w3.org/Mobile/`

Das Web bietet eine Möglichkeit, Daten miteinander zu verknüpfen. Das Semantische Web verfolgt das Ziel, Informationen auf der Ebene der Bedeutung miteinander zu verknüpfen. Die Daten im Semantischen Web sind strukturiert und in solcher Form repräsentiert, sodass auch Computer die Möglichkeit haben, den Inhalt zu verstehen und zu interpretieren, und dadurch hinsichtlich ihrer inhaltlichen Bedeutung zu verarbeiten [Behrendt, 2004].

Behrendt [2004] und Rossi et al. [2007] beschreiben drei wesentlich Aspekte, die im Semantischen Web ein große Rolle spielen:

- Informationslieferanten liefern semantisch gekennzeichnete Webseiten.
- Intelligente Softwareagenten werden für die Suche in semantisch gekennzeichneten Webseiten entwickelt.
- Informationslieferanten und Softwareagenten müssen sich zu einer gemeinsamen Begriffswelt (*Ontologie*) bekennen, damit die Inhalte für Maschinen erfassbar werden.

Ferner konstatiert Behrendt [2004], dass sich aus dieser Aufgabenteilung folgende Kerntechnologien ableiten lassen:

1. Die Extensible Markup Language (XML) wird als Trägerformat für die semantische Auszeichnung verwendet.
2. Das Resource Description Framework (RDF) wird zum Auffinden und Beschreiben von Web-Inhalten verwendet.
3. Web Ontology Language (OWL) wird für die Semantik der Begriffswelt (Ontologie) verwendet. Ontologien können innerhalb von RDF mittels OWL festgelegt werden.

Softwareagenten bilden die aktive Komponente im Semantischen Web. Ein Softwareagent kann Ontologien verstehen. Das Konzept kann bei einer Suche derart angewandt werden, dass ein Softwareagent wiederum nach Web-Inhalten suchen oder fragen kann, die entsprechend der Agentenontologie und dem Suchbegriff des Nutzers, wahrscheinlich relevant sind.

3.1.5 Rich Internet Applications

Der Begriff Rich Internet Application (RIA) wurde 2002 durch ein Konzeptpapier von Macromedia geprägt. Eine Rich Internet Application ist Web-basiert, d.h. die Applikation läuft wie eine normale Web-Anwendung in einem Web-Browser, kann jedoch mit Features und einen Funktionsumfang aufwarten, der sonst nur auf klassischen Desktop-Applikationen zu finden ist. Interessant ist eine Rich Internet Application aufgrund der Evolution der Rolle des Web-Browsers, der sich von einem statischen Request-Response Schnittstelle hin zu einer dynamischen und asynchronen Schnittstelle [Rossi et al., 2007] entwickelt hat.

Die technische Weiterentwicklung aber auch Aspekte des Web 2.0 hatten zur Folge, dass RIAs überhaupt erst möglich wurden. AJAX und Web Services sind einige der Technologien, die bei der Erstellung von RIAs zum Einsatz kommen. Es existieren auch RIA basierte Frameworks, die für die Erstellung entsprechender Anwendungen für das Web verwendet werden können. Ein Beispiel für ein solches RIA basiertes Framework ist Flex[9] von Adobe [Rossi et al., 2007].

RIAs können mit Aspekten aufwarten, die in vielen vorhandenen Web-Anwendungen nur sehr schwer

[9] Adobe Flex: `http://www.adobe.com/de/products/flex/`

erreicht werden können und deshalb bei klassischen Web-Anwendungen als Mangel angesehen werden. Die Aspekte die RIAs auszeichnen sind *Rich User Experience*, sehr hohe Interaktivitäts-Möglichkeiten und sehr gute Usability-Eigenschaften. Es sei dahingestellt, dass alleine durch die Verwendung geeigneter Technologie unmittelbar noch kein Mehrwert für die Nutzer erreicht wird. Damit User Experience erreicht werden kann, ist es notwendig vorher auch die Bedürfnisse der Nutzer wahrzunehmen und bei der Implementierung adäquat zu berücksichtigen [Rossi et al., 2007].

3.2 Web Engineering

In Abschnitt 3.1 wurde gezeigt, dass moderne Web-Anwendungen einen immer höher werdenden Komplexitätsgrad aufweisen, die Rolle der Nutzer und Interaktion immer wichtiger wird, ubiquitäre Merkmale beispielsweise für Mobile Geräte sowie auch klassische dokumentenzentrierte und transaktionale Merkmale zu berücksichtigen sind. Sogenannte Ad-Hoc-Vorgehensweisen sind in Anbetracht des hohen Komplexitätsgrades, welcher der Erstellung einer Web-Anwendung zugrunde liegt, nicht zielführend und es zeigte sich, dass sehr viele Projekte deshalb scheiterten bzw. notwendige Qualitätskriterien dadurch nicht erfüllt werden konnten ([Kappel et al., 2004], [Rossi et al., 2007]).

Die Erstellung von Web-Anwendungen ist eine Herausforderung. Ein Web-Anwendung ist ein *komplexes Softwaresystem*. Der Einsatzdomäne Web ist ein Alleinstellungsmerkmal einer Web-Anwendung im Vergleich zu einer klassischen Software-Anwendung. Eine Web-Anwendung erfordert den Einsatz und die Berücksichtigung von Standards und Technologien, die mit dem WWW einhergehen. [Kappel et al., 2004] definiert eine Web-Anwendung als ein Softwaresystem, *"das auf Spezifikationen des World Wide Web Consortium (W3C) beruht und Web-spezifische Ressourcen wie Inhalte und Dienste bereitstellt, die über eine Benutzerschnittstelle, den Web-Browser, verwendet werden"*.

Kappel et al. [2004] definiert Web Engineering als eine Disziplin die eine *systematische* Entwicklung von Web-Anwendungen zum Ziel hat. Durch die Anwendung von *systematischer* und *quantifizierbarer Ansätze* soll die Umsetzung qualitativ hochwertiger Web-Anwendungen durchgeführt werden. Es gilt Konzepte, Methode, Techniken sowie Werkzeuge in einer methodisch fundierten Herangehensweise mit einander zu verbinden, damit neben den zu erfüllenden Qualitätskriterien auch eine kosteneffektive Umsetzung einer Web-Anwendung möglich ist. In der Literatur wird sehr oft auch die folgende Definition verwendet, die mit der Definition von Kappel et al. [2004] einhergeht, aber zusätzlich auch Management-Methoden einbezieht [Murugesan et al., 2001]:

> *"Web Engineering is the establishment and use of sound scientific, engineering and management principles and disciplined and systematic approaches to the successful development, deployment and maintenance of high quality Web-based systems and applications. "*

Die Umsetzung einer Web-Anwendung ist in mehrere Phasen gegliedert und kann sich beispielsweise beginnend mit einer Anforderungsbeschreibung über die Phasen des Entwurfs und der Implementierung, hin zu einer Test-Phase bewegen. Der Betrieb als auch die Wartung einer Web-Anwendung sind ebenso Phasen der Entwicklung einer Web-Anwendung. Web Engineering berücksichtigt die einzelnen Phasen und erstreckt sich somit über den gesamten Lebenszyklus einer Web-Anwendung [Kappel et al., 2004].Charakteristika von Web-Anwendungen stellen besondere Anforderungen an Web-Engineering. Diese Anforderungen können aus mehreren Gesichtspunkten abgeleitet werden [Kappel et al., 2004]:

- Aspekte der Web-Anwendung als komplexes Softwaresystem
- Aspekte der Entwicklung von Web-Anwendungen
- Aspekte der Nutzung von Web-Anwendungen

Neben der ingenieursmäßigen Herangehensweise bzgl. der Umsetzung einer Web Anwendung ist Web Engineering auch eine *wissenschaftliche Disziplin*, die sich mit der Erforschung *systematischer* und *quantifizierbarer Ansätze* befasst [Kappel et al., 2004]. Web Engineering als wissenschaftliche Disziplin grenzt sich von der Disziplin Web Science dadurch ab, dass Web Engineering eine spezialisierte Variante des Software-Engineering ist und sich mit den speziellen Faktoren des Web als eine Entwicklungsumgebung für Web-Anwendungen befasst. Web Science dahingehend ist ein interdisziplinärer Ansatz der sich mit der Erforschung jeglicher Phänomenen des WWW, mit dem Ziel diese besser zu verstehen zu können, befasst [Wilde und Gaedke, 2008].

3.2.1 Web Engineering versus Software Engineering

Web Engineering ist ein eigenständiger Zweig des Software Engineerings. Zusammengefasst verfolgt Web Engineering die folgende Ziele [Kappel et al., 2004]:

- Festlegung von klar definierten Zielem und Anforderungen an die Web-Anwendung
- Systematische Entwicklung einer Web-Anwendung in Phasen
- Genaue Definition und Design der einzelnen Phasen
- Kontinuierliche Überwachung des Entwicklungsprozess

Dadurch soll ein Entwicklungsprozess erreicht werden, der einer ingenieursmässigen Herangehensweise mit dem Ziel von Planbarkeit und Wiederholbarkeit gleicht, und um damit auch eine kontinuierliche Weiterentwicklung von Web-Anwendungen sicher zu stellen. Diese Ziele sind sehr ähnlich zu den Grundprinzipien des Software-Engineering. Suh [2005] argumentiert, dass Web Engineering und Software Engineering weitgehend verschieden zu einander sind. Die Erstellung von Web-Anwendungen stellt für Software Engineering eine neue Arbeitsdomäne dar. Web-Anwendungen haben besondere Charkteristika. Daraus folgt, dass es notwendig ist, sehr viele Ansätze aus dem Software Engineering anzupassen oder neu zu entwickeln.

Web Engineering unterschiedet sich zum klassischen Software Engineering beispielsweise in den folgenden Punkten ([Suh, 2005], [Kappel et al., 2004], Rossi et al. [2007]):

- Aus der Definition des Begriffs Web-Anwendung geht hervor, dass eine starke Abhängigkeit zu Standards und Protokollen gegeben ist, die notwendig sind um eine Anwendung für das Einsatzgebiet Web zu ermöglichen.
- Web-Anwendungen müssen sehr viele verschiedene Inhalte wie Text, Graphiken, Bilder oder Videos präsentieren können.
- Ein Web-Anwendung muss für sehr viele unterschiedliche Nutzer ausgelegt sein. Nutzer können das System gleichzeitig benutzen.Die Interaktion mit Benutzern in Verbindung mit dem Web spielt eine große Rolle. Herauszuheben ist die Tatsache, dass man bei vielen Web-Anwendungen im Vorhinein nicht die Nutzer kennt. Dadurch, dass man im Web größtenteils unbekannte Nutzer hat, ist es sehr schwierig entsprechende User-Requirements abzuleiten.
- User-Requirements ändern sich sehr schnell und sehr oft. Web-Anwendungen müssen in sehr kurzer Zeit entwickelt werden. Web Systeme wachsen und ändern sich sehr schnell. Das stellt eine Herausforderung an den Entwicklungsprozess dar. Änderungen sind oft notwendig damit eine Web-Anwendung populär wird. Änderungen einer Web-Anwendungen sind oftmals auch erforderlich, da man im Web um Nutzer kämpft.

- Das Phänomen Web 2.0 hat eine neue Herangehensweise bei der Erstellung von Web-Anwendungen mit sich gebracht. *Perpetual Beta* bedeutet, dass man mit einer sehr einfachen Version einer Web-Anwendung beginnt. Die bereitgestellte Beta-Version der Web-Anwendung wird genutzt, um eine Nutzer-Basis aufzubauen. Feedback wird gesammelt und dient als Grundlage für die nächste Beta-Version, die bereitgestellt wird, die Nutzern unmittelbar eine Verbesserung zu bringen. Mit diesem Ansatz lassen sich User Requirements im Netz sammeln [O'Reilly, 2005].
- Web-Anwendungen entwickeln sich ständig weiter. Es ist in den meisten Fällen nicht möglich die Web-Anwendung vollständig zu spezifizieren. Es muss mit Änderungen gerechnet werden.
- Web-Anwendungen unterliegen hohen ästhetischen Anforderungen. Look-And-Feel Aspekte sind sehr wichtig. Die Navigation innerhalb einer Web-Anwendung muss sehr leicht und intuitiv gestaltet sein. Die Diversität der Nutzer, kulturelle Unterschiede usw. sind hier eine Herausforderung. Zusätzlich muss für eine Vielzahl von möglichen Endgeräten sichergestellt werden, dass eine Web-Anwendung ästhetischen Anforderungen genügt. Die Usability einer Web-Anwendung stellt ein Qualitätsmerkmal dar.
- Sicherheitsaspekte sind sehr wichtig. Security-Aspekte in einer Web-Anwendung sind auch Qualitätsmerkmale.
- Web Engineering ist ein interdisziplinäres Gebiet. Das stellt auch eine Herausforderung an die Entwickler von Web-Anwendungen dar. Abbildung 3.1 gibt eine Überblick, welche Gebiete –Software Engineering, Multimedie, Human Computer Interaction und viele andere– für die Entwicklung von Web-Anwendungen eine Rolle spielen.

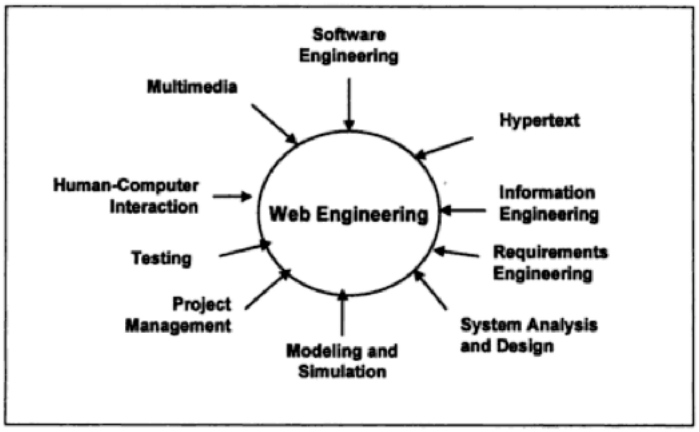

Abbildung 3.1: Abhängigkeit und Zusammenspiel von Web Engineering zu anderen Disziplinen [Suh, 2005]

3.2.2 Kategorien von Web-Anwendungen

Es existiert eine Vielfalt verschiedener Web-Anwendungen. Web-Anwendungen unterscheiden sich massiv durch die Vielzahl von Anwendungsbereichen und der damit verbundenen Komplexität. Web-Anwendungen bieten einen gewaltigen Unterschied hinsichtlich Funktionsumfang, Charakteristika sowie Anforderungen. Web-Anwendungen können in verschiedenste Weisen kategorisiert werden, beispielsweise nach Funktionalität [Suh, 2005]. Diese Einteilung hat den Vorteil, dass Anforderungen von Web-Anwendungen besser verstanden werden können. Eine andere Möglichkeit ist die Einteilung in verschiedene Kategorien anhand der Entwicklungshistorie und des Komplexitätsgrades [Kappel et al., 2004]. Abhängig von der Entwicklungshistorie und des Komplexitätsgrad weisen Web-Anwendungen dokumentenzentrierte,

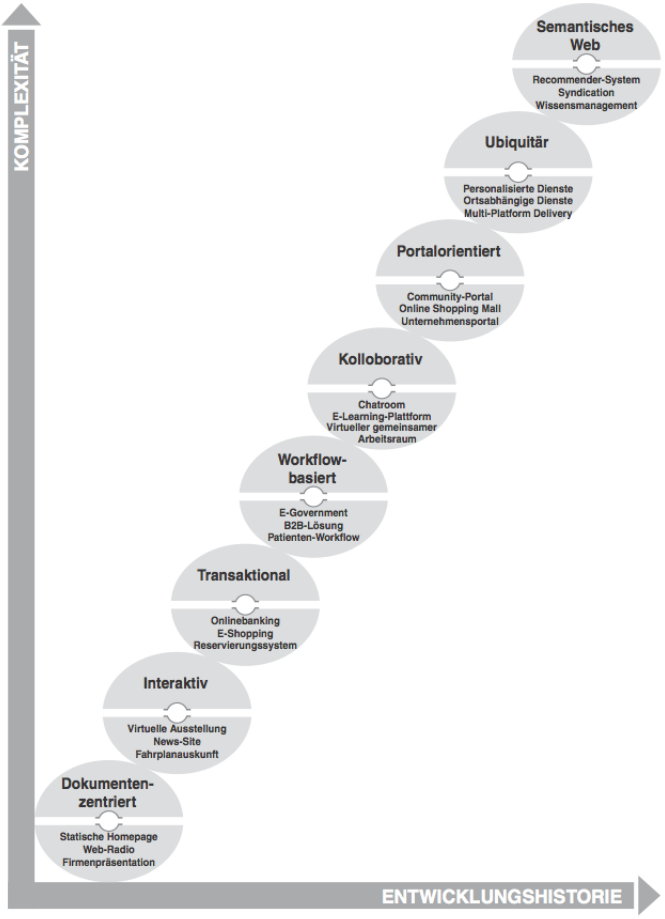

Abbildung 3.2: Kategorien von Web-Anwendungen nach Kappel et al. [2004]

interaktive, transaktionale, ubiquitäre und Merkmale des Semantischen Webs auf. Abbildung 3.2 zeigt diese Einteilung, wobei ersichtlich ist, dass einfache dokumentzentrierte Web-Seiten des Web 1.0, einen sehr niedrigen Komplexitätsgrad aufweisen. Der Komplexitätsgrad steigt kontinuierlich, seitdem Web-Seiten durch Web-Anwendungen abgelöst wurden.

3.2.3 Charakteristika von Web-Anwendungen

In Abschnitt 3.2.1 wurden einige Charakteristika von Web-Anwendungen angeführt, die für eine bessere Abgrenzung von Software Engineering zu Web Engineering notwendig waren. In Abschnitt 3.2.2 wurde erwähnt, dass eine weite Reihe von unterschiedlichen Kategorien von Web-Anwendungen existiert. Eine Web-Anwendung hat abhängig von der Kategorie naturgemäß verschiedene Charakteristika bzw. sind verschiedene Charakteristika unterschiedlich stark ausgeprägt. [Kappel et al., 2004] gibt einen Überblick über verschiedene Charakteristika von Web-Anwendungen und zeigt in Anlehnung an den ISO/IEC-9126-1-Standard, der zur Bewertung der Qualität von Softwareanwendungen herangezogen wird, eine Gruppierung von Charakteristika nach den folgenden drei Dimensionen:

Produkt: Produktbezogene Charakteristika finden sich im Inhalt (*Content*), Navigationsstruktur (*Hypertext*) und der Präsentation einer Web-Anwendung [Kappel et al., 2004].

Nutzung: Nutzungsbezogene Charakteristika werden nach Aspekten des sozialen, technischen und natürlichen Kontext unterteilt. Die Nutzung von Web-Anwendungen wird in sogenannte Nutzungssituationen

bzw. Kontexte eingeteilt, um verbundene Heterogenitäten hinsichtlich Nutzer, Kultur, Software, Zeit und Ort besser fassen zu können [Kappel et al., 2004].

Entwicklung: Entwicklungsbezogene Charakteristika umfassen verschiedene Ressourcen, die die Entwicklung einer Web-Anwendung prägen. Hierbei handelt es sich um den Projektmitarbeiter, die technische Infrastruktur und den Entwicklungsprozess [Kappel et al., 2004].

Kappel et al. [2004] konstatiert, dass durch die Zuordnung von Charakteristiken von Web-Anwendungen zu diesen Dimensionen, sich der Einfluss auf die Qualität der Web-Anwendung zeigen lässt. Die Idee ist nun, dass durch diese Zuordnung einzelne Charakteristika wiederum herangezogen werden können, um neue Anforderungen an Web Engineering abzuleiten.

Die *Evolution* von Web-Anwendungen kann in diesem Zusammenhang als ein Querschnittcharakteristikum gesehen werden. Der Evolution ist oben genannten Dimensionen unterworfen. In Abschnitt 3.2.1 wurde die Unbegrenztheit von Rahmenbedingungen und Anforderungen sowie die Schnelllebigkeit erwähnt. Neben dieser Herausforderungen ist es auch der Konkurrenzdruck, der die Evolution von Web-Anwendung als Notwendigkeit erscheinen lässt. In diesem Zusammenhang muss auch das Phänomen Perpetual Beta erwähnt werden, dass in Abschnitt 3.2.1 erklärt wurde.

3.3 Architektur

Performanz, Wartbarkeit, Erweiterbarkeit und Verfügbarkeit sind nur einige Qualitätsanforderungen an Web-Anwendungen, die maßgeblich durch die eingesetzte Architektur bestimmt werden. Es gilt, dass die zugrunde liegende Architektur einer Web-Anwendung stark dafür verantwortlich ist, ob Qualitätsanforderungen an das Softwaresystem erreicht werden können. Bei der Entwicklung einer Web-Anwendung ist es enorm wichtig sehr sorgfältig alle Aspekte bzgl. der zugrunde liegenden Architektur zu berücksichtigen, damit eine geeignete Auswahl und Festlegung erfolgen kann. Zusätzlich ist es auch notwendig und sinnvoll auf vorhandenes Architekturwissen zurückzugreifen. Der Einsatz von Entwurfsmustern und Frameworks kann einen positiven Einfluss auf die Qualität eine Web-Anwendung haben [Eichinger, 2004] .

In der Literatur findet sich keine eindeutige Definition für den Begriff Architektur bzw. Software-Architektur. Eichinger [2004] fasst die folgenden Eigenschaften einer Software-Architektur wie folgt zusammen:

- Eine Architektur beschreibt Struktur sowie statische und dynamische Aspekte eines Softwaresystems. Ein Softwaresystem besteht aus Strukturen. Eine Struktur kann in einzelne Komponenten zerlegt werden. Komponenten haben eine Schnittstelle und es kann die Beziehung zwischen Komponenten beschrieben werden.

- Funktionale Anforderungen und Qualitätsanforderungen werden bei der Erstellung einer Architektur durch Komponenten und deren Schnittstellen und Beziehungen abgebildet. Das ist beispielsweise durch ein iteratives Vorgehen möglich. Eichinger [2004] konstatiert, dass aus diesem Grund eine Architektur den Übergang zwischen Analyse und Realisierung bildet.

- Eine Architektur kann aus den folgenden Blickwinkeln betrachtet werden:

  - Die *konzeptuelle* Sicht betrachtet Details der Anwendungsdomäne.
  - Die *Laufzeit-Sicht* beschreibt Details zur Laufzeit des Systems.
  - Die *Prozess-Sicht* beschreibt die Abbildung von Abläufen.
  - Alle Softwareartefakte werden in der *Implementierungssicht* betrachtet.

Diese Betrachtung macht es möglich, dass verschiedene Architekturaspekte bearbeitet oder detailliert werden können. Die Beschreibung der Architektur kann anhand dieser Sichten erfolgen. Eine fünfte Sicht können Use Cases und Szenarien sein, mit deren Hilfe die Architektur besser illustriert werden kann [Kruchten, 1995]. [Booch, 2001] konstatiert, dass durch verschiedene Blickwinkel komplexe Systeme besser verstanden werden können.

- Eine Architektur macht ein Softwaresysteme durch die Strukturierung und Gliederung in mehrere Schichten verständlich als auch in seiner Komplexität beherrschbar. Die Einteilung in mehrere Schichten beruht auf dem Prinzip der Abstraktion. Mit dem Prinzip der Abstraktion soll erreicht werden, dass Systemdetails durch geeignete Kapsellung versteckt werden. Eigenschaften des Systems können dadurch besser identifiziert und aufgenommen werden. Komplexe Systeme können mehrere Abstraktionsebenen beinhalten. Jede Abstraktionsebene kann für sich eine eigene Architektur haben [Fielding, 2000].

- Tom DeMarco hat die Bezeichnung *framework of change* für den Begriff Architektur geprägt. Ein Softwaresystem wird im Rahmen einer Architektur entwickelt. Wenn eine Architektur Erweiterungen des Softwaresystems vorgesehen hat, dann ist eine *Architektur der Rahmen für ein flexibles System*.

3.3.1 Entwurf von Architekturen

Eine gute Architektur ist abhängig von Anforderungen und Zielen. Anforderungen an die Software sind in der Regel auch Anforderungen an die Architektur. Die Entwicklung einer Web-Anwendung ist stark dadurch gekennzeichnet, dass sich Randbedingungen und Anforderungen während der Entwicklung ändern. Eine Architektur unterliegt bei der Definition der Architektur den folgenden Einflussfaktoren und Randbedingungen: *Funktionale Anforderungen* und *Nich-Funktionale Anforderungen* beeinflussen die Architektur einer Anwendung am stärksten. Neben dem Funktionsumfang und Qualitätskriterien sind *Technische Randbedingungen*, wie beispielsweise verwendete Standards, aber auch die *Erfahrung* der Softwarearchitekten zu berücksichtigen [Eichinger, 2004], [Kruchten, 1995], [Booch, 2001].

Eine Architektur wird durch einen Softwareentwicklungsprozess entworfen, wobei die oben genannten Rahmenbedingungen und Anforderungen berücksichtigt werden müssen. Ein iteratives Vorgehen reduziert Risiken, die aus unzureichenden oder ändernden Anforderungen entstehen. Eine wichtige Rolle spielt der *Architekt*, der mit seinen Erfahrungen im Anwendungsbereich zu einer guten Architektur beiträgt. Ein elementarer Schritt ist die Wiederverwendung von Architektur-Wissen. Für konkrete Entwurfsprobleme gibt es *Muster* und *Frameworks*, die eingesetzt und angewandt werden können. Der Einsatz von Muster und Frameworks muss auch zum Problem passen und erfordert Erfahrung für den richtigen Einsatz [Eichinger, 2004].

3.3.1.1 Muster

Ein Muster ist ein bewährter Lösungsansatz für ein bekanntes und wiederkehrendes Entwurfsproblem, dass in einem spezifischen Kontext auftritt. Ein Muster beinhaltet erprobtes Entwurfswissen und ist gewissermassen die Wiederverwendung einer guten und erprobten Problemlösung. Die Entwicklung von *qualitativ hochwertigen* Softwaresystemen ist unter Berücksichtigung und Wiederverwendung von erprobten Entwurfswissen möglich [Eichinger, 2004]. Ein Muster beschreibt im Kontext der Problemstellung:

- alle beteiligen Komponenten

- Verantwortlichkeiten von Komponenten

- Beziehungen zwischen Komponenten
- Zusammenwirken von Komponenten

Es gibt eine Unterscheidung in *Architekturmuster* und *Entwurfsmuster*. Die grundlegende Organisation und Interaktion zwischen Komponenten einer Anwendung wird durch Architekturmuster beschrieben. Das Model-View-Controller Pattern ist eine typisches Beispiel für ein Architekturmuster und wird in Abschnitt 3.4 näher erklärt. Ein Entwurfsmuster beschreibt im Gegensatz keine Strukturierungmechanismen, sondern die Lösung für konkrete Teilprobleme auf einer niederen Abstraktionsebene.

3.3.1.2 Frameworks

Eichinger [2004] definiert ein Framework als ein wiederverwendbares Softwaresystem mit bereits implementierter, genereller Funktionalität, welches durch Spezialisierung in eine Anwendung überführt werden kann. Die Verwendung eines geeigneten Frameworks für einen bestimmten Anwendungsbereich ist eine Möglichkeit, bestehendes Architekturwissen, grundlegende Architekturen und einhergehenden Funktionsumfang weiterzuverwenden. Durch sogenannte *Hot-Spots* ist es auch möglich, Frameworks zu erweitern [Shan und Hua, 2006]. Hot-Spots können beispielsweise durch einen geeigneten Einsatz von Entwurfsmuster (beispielsweise dem Factory Design Pattern) erfolgen. Im Gegensatz definieren *Frozen-Spots* die grundlegende Architektur des Frameworks. Hierbei handelt es sich um Komponenten und deren Beziehungen zueinander, die nicht verändert werden können.

Abschnitt 3.5 gibt einen kurzen Überblick über vorhandene Frameworks für den Bereich Web-Anwendungen wieder.

3.3.2 Kategorisierung von Architekturen

Eine Architektur von Web-Anwendungen muss eine Reihe von Spezifika berücksichtigen. Einerseits spielen hohe Qualitätsanforderungen, die Integration und die Inhomogenität der notwendigen Technischen Infrastruktur (Web Server, Applikationsserver) als auch Globalität von Web-Anwendungen, im Sinne von Mehrsprachigkeit, eine große Rolle. Alle diese Aspekte müssen in einer Architektur berücksichtigt werden.

Eine Architektur für Web-Anwendungen besteht aus Komponenten. Komponenten sind miteinander in Beziehung und können in der Regel basierend auf dem Request-Response-Prinzip miteinander kommunizieren. Abbildung 3.4 bietet einen Überblick von Basiskomponenten einer generischen Architektur von Web-Anwendungen.

Eine Kategorisierung von Architekturen kann nach *Schichtenaspekten* und nach *Datenaspekten* erfolgen. Der Schichtenaspekt von Architekturen beruht auf dem Konzept der *separation of concerns*. Ein Softwaresystem kann in verschiedene Schichten gegliedert werden. Schichten werden in diesem Kontext auch als *tiers* bezeichnet. Sehr viele Web Application Frameworks (WAF) werden nach dem Schichtenaspekt strukturiert. In der Literatur findet man die folgenden Architekturen, die nach dem Schichtenaspekt strukturiert sind [Eichinger, 2004]:

2-Schichten-Architektur: Ein typische Beispiel hierfür wäre die Client/Server Architektur des Web 1.0. Statische oder Dynamische Web-Seiten werden über einen Web Server als Dienst zur Verfügung gestellt. Typisches Problem von 2-Schicht-Architekturen ist eine geringe Modularität, wodurch Probleme durch die Vermischung von Applikationslogik und der Darstellungslogik auftreten.

N-Schichten-Architektur: Mit dieser Architektur kann eine Web-Anwendung in beliebig viele Schichten gegliedert werden. Das Problem der 2-Schichten-Architektur lässt sich sehr einfach lösen,

indem man für die Darstellung/Präsentation und für die Anwendung einen eigenen Layer definiert. Die 3-Schichtige-Architektur hat dieses Problem gelöst und besteht typischerweise aus einer Präsentations-, Logik- und Datenhaltungsschicht. Sehr viele Web-Anwendungen beruhen auf diesem Prinzip und können auch in dieser Architektur gedacht werden. Eine sehr bekannte Ausprägung ist die JSP-Model-2 Architektur. Diese Ausprägung ist dadurch sehr interessant für Web-Anwendungen, da eine Implementierung des Model-View-Controller Patterns berücksichtigt wurde. Abschnitt 3.4 beschreibt dieses Architekturmuster im Detail.

Mit dem Schichten-Prinzip geht einher, dass jede Schicht eine Abstraktion eines konkreten Problembereiches ist [Jablonski et al., 2002]. Die Unterscheidung von Schichten in *Layers* und in *Tiers* ist ein fundamentaler Aspekt. Layers sind logische Schichten und erlauben die Strukturierung eines Systems in mehrere Schichten, wobei jede Schicht eine höhere Abstraktionseben darstellt als die darunterliegende. In diesem Kontext spricht man auch von einer vertikalen Strukturierung. Ein jeder Layer kann wiederum in verschiedene Komponenten horizontal organisiert werden, wodurch einzelne Aufgaben und Dienste lokalisiert werden können. Durch die vertikale Strukturierung soll die Abhängigkeit zwischen den Schichten verringert werden. Layer erlauben auf abstrakte Weise die Trennung von Aufgaben und Verantwortung. Layer sollen austauschbar sein und gegebenenfalls auch wiederverwendet werden können. Sogenannte *Tiers* sind physikalische Schichten. Tiers ist ein Muster, wie ein System in physikalische Schichten eingeteilt werden kann. Eine typische Trennung in verschiedene Tiers ist in Abbildung 3.3 zu erkennen. Es gilt, dass Layers auf Tiers verteilt werden können.

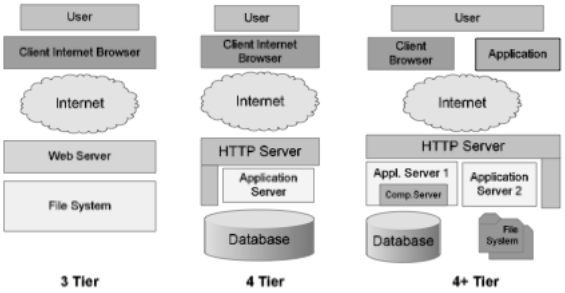

Abbildung 3.3: Darstellung einer 3-, 4- und 4-Tier-Architektur [Jablonski et al., 2002]

Datenaspekte von Architekturen betreffen die Unterstützung von unterschiedlichen Daten. Die Integration von relationalen Datenbanken in Web-Anwendungen ist in diesen Kontext eine Herausforderung für *Datenbankzentrierte Architekturen*. Die Integration von multimedialen Inhalten in Web-Anwendungen soll mit *Architekturen für multimediale Daten* erreicht werden. Daneben gibt es auch noch spezielle *Dokumentenzentrierte Architekturen*.

3.4 Model-View-Controller (MVC)

In Abschnitt 3.3.2 wurde erklärt, dass durch die Separation von verschiedenen Aspekten eines Systems Komplexität reduziert werden kann. Das Architekturmuster Model-View-Controller basiert auf dem Ansatz *Seperation of Concerns*. Im Rahmen der Weiterentwicklung von Smalltalk wurde 1979 von Trygve Reenskaug das Model-View-Controller Konzept für Benutzeroberflächen beschrieben. Das Architekturmuster, das besonders für die Strukturierung von interaktive Anwendungen geeignet ist, wurde auf den Bereich von Web-Anwendungen übertragen [Förther et al., 2009]. Mit diesem Muster ist ein Grobentwurf eines komplexen Systems in drei Einheiten möglich. Das Architekturmuster Model-View-Kontroller strukturiert ein Softwaresystem in die folgenden Einheiten (vgl. [Sun, 2002], [Förther et al., 2009], [Eichinger, 2004], [Hunt und Thomas, 1999]):

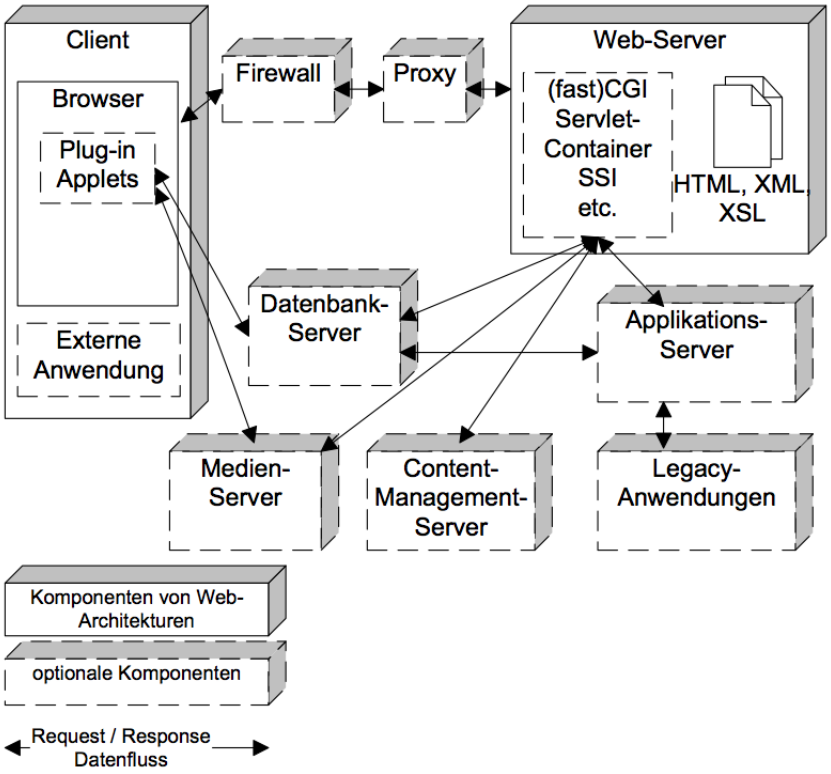

Abbildung 3.4: Komponenten eine generischen Architektur nach Eichinger [2004]

Model: Diese Schicht beinhaltet Daten, das Datenmodell und die Anwendungslogik des Systems.

View: Diese Einheit ist verantwortlich für die graphische Repräsentation des Model-Schicht. Ein Model kann auf verschiedene Arten dargestellt werden. HTML, XML, PDF-Reports sind nur einige Beispiele von Darstellungsmöglichkeiten aus dem Bereich der Web-Anwendungen.

Controller: Diese Einheit behandelt den Aspekt der Programsteuerung und beinhaltet Ablauf- und Steuerungslogik. Der Controller ist für die Verarbeitung von Benutzereingaben verantwortlich. Ausserdem muss der Controller den Zustand des Model konsistent halten und kann auf die Anwendungslogik des Model zugreifen. Der Controller ist auch für die Auswahl und Aktualisierung der grafischen Präsentation der Einheit View verantwortlich [Eichinger, 2004].

Abbildung 3.5 zeigt eine grafische Darstellung des Musters und die Beziehungen der einzelnen Einheiten zueinander. Im Kontext von Web-Anwendungen kann durch den Einsatz dieses Musters erreicht werden, dass komplexe Anwendungslogik von den Aspekten des User-Interface entkoppelt werden können. Daraus folgt, dass beide Schichten weitgehend unabhängig von einander entwickelt werden können. View und Controller bilden die Benutzerschnittstelle. Konsistenz des Models wird über über einen Benachrichtigungsmechanismus bzgl. Änderungen gesichert [Sun, 2002]. Das Observer-Pattern kann als ein solcher Mechanismus eingesetzt werden.

Dieses Muster hat den Vorteil, dass mehrere Sichten auf das selbe Model möglich sind. Damit eine neue Sicht aufgebaut werden kann, muss die View und ggfs. der Controller ausgetauscht werden. Ein Einsatzgebiet dieses Muster sind Frameworks. Ein Nachteil dieses Muster kann sein, dass View und Controller sehr eng miteinander verbunden sind. Auch der Datenzugriff auf das Model kann ineffizient sein.

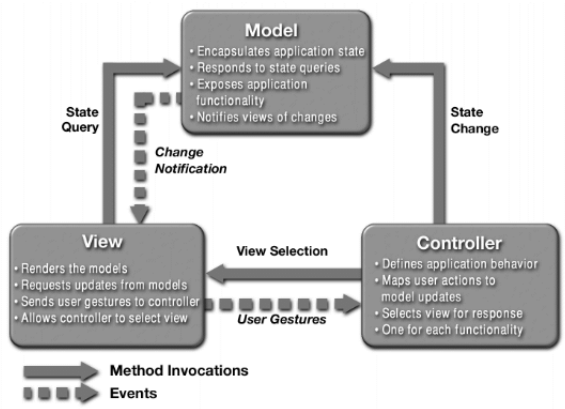

Abbildung 3.5: Darstellung des Model-View-Controller Muster [Sun, 2002]

3.5 Web Application Frameworks (WAF)

In Abschnitt 3.3.1.2 wurde der Begriff Framework definiert und es wurde argumentiert, dass durch den sinnvollen Einsatz eines Frameworks die Möglichkeit, bestehendes Architekturwissen für konkrete Entwurfsprobleme wiederzuverwenden, besteht. Frameworks, die für die Erstellung und Umsetzung von Web-Anwendungen verwendet werden können, werden als Web Application Frameworks (WAF) bzw. einfach als Web Framework bezeichnet. Ein Web Application Framework ist eine wiederverwendbare und modulare Plattform, die ein Grundgerüst für die spezifische Umsetzung einer Web-Anwendung bietet. Das in Abschnitt 3.4 beschriebene Model-View-Controller Muster wird typischerweise von Web Application Frameworks implementiert. Durch die Berücksichtigung von Standard Technologien und die Organisation in eine N-Tier-Architektur können Web Application Frameworks sehr einfach in die vorhandene bzw. etablierte Infrastruktur integriert werden [Shan und Hua, 2006].

Der Einsatz eines Web Frameworks kann die Produktivität der Erstellung einer Web-Anwendung und gleichzeitig die Qualität erhöhen [Shan und Hua, 2006]. Damit diese Werte im Endeffekt erreicht werden, ist es notwendig, dass Web Frameworks sinnvoll nach Aspekten und Anforderungen der Anwendungsdomäne ausgewählt werden. Es ist zusätzlich erforderlich, dass bei der Umsetzung der Web-Anwendung die Konzepte und Paradigmen des Web Framework berücksichtigt und sinnvoll angewandt werden.

Es gibt eine schier unübersichtliche Menge unterschiedlicher Frameworks am Markt. Damit man sich besser orientieren kann, ist es sinnvoll Web Frameworks nach bestimmten Kritierien einzuteilen. Eine vielzitierte Einteilung nach Shan und Hua [2006] lässt die folgenden Typisierung von Web Application Frameworks zu:

Request-basierte Frameworks: Diese Frameworks bieten die Möglichkeit eingehende Requests . durch Controler und Actions direkt abzuhandeln. Diese Framework sind zustandslos. Logik wird direkt auf URLs abgebildet. Eigene Module bestimmen die Strukturierung von Daten. Ein typische Beispiel ist das Framwork Struts[10].

Komponenten-basierte Frameworks: Diese Frameworks sind sehr ähnlich zu den Frameworks die zur GUI Erstellung von Desktopanwendungen eingesetzt werden. Aspekte der Request-Abwicklung werden in eigene Komponenten gekapselt, die unabhängig von Aspekten des WWW sind. Logik wird in Komponenten abgebildet, die wiederverwendbar sind. Zustände können in konkreten Instanzen von Komponenten gehalten werden. Typische Komponenten-basiserte Framworks sind

[10] Struts: `http://struts.apache.org/`

Java Server Faces[11] (JSF) oder Wicket[12].

Hypride Frameworks: Diese Frameworks kombinieren Request-basierte und Komponenten-basierte Frameworks und behalten aber über viele Aspekte die Kontrolle.

Meta Framework: Dieser Typ von Framework kann als ein Framework für Frameworks gedacht werden. Typischerweise stellen diese Frameworks eine Menge von Schnittstellen für Services zur Verfügung. Es können aber auch vorhandene Services und Komponenten integriert werden. Durch den Einsatz des Inversion of Control Muster wird es möglich, dass diese Frameworks mit anderen Frameworks und Komponenten zusammenarbeiten können. Das bekannteste Beispiele für ein Meta-Framwork ist Spring[13].

RIA-basiertes Framework: In Abschnitt 3.1.5 wurde der Begriff Rich Internet Application definiert. Diese Frameworks sind für die Erstellung solcher Applikationen geeignet und stellen diesbezüglich einen eigenen Container für die Client-Schicht bereit, wodurch unterstützt wird, dass der Client einen Zustand und zusätzlich der Client ein eigenes Interaktionsmodell besitzen kann. Echo2[14] ist ein bekanntes RIA-basiertes Framework.

Eine alternative und pragmatischere Kategorisierung von Web Frameworks liefert Wähner [2011a]. Diese Einteilung unterscheidet hinsichtlich der Typisierung nach Shan und Hua [2006] in den folgenden Punkten: Es wird berücksichtigt, dass je nach Einsatzgebiet einer Web Anwendungen die Größe variiert und gleichzeitig der Zeitaufwand für die Umsetzung steigt. Es gilt auch der Zusammenhang, dass für jeden Typ von Web-Anwendungen auch spezifische Web Frameworks existieren. In Abbildung 3.6 ist diese Kategorisierung ersichtlich. Die Unterscheidung zwischen Rich-Client und Rich Internet Application ist erwähnenswert, da diese auch Auswirkung auf das zugrunde liegende Framework hat. Ein Richt Client benötigt im Gegensatz zu einer RIA kein zusätzliches Plug-In, wodurch die Komplexität um einiges reduziert werden kann. Die Komplexität von Web Frameworks in Abhängigkeit von Frameworktypen ist ein sehr wertvolles Auswahlkriterium und ist in Abbildung 3.7 ersichtlich [Wähner, 2011b].

In Kapitel 5 Abschnitt 5.3 wird im Rahmen eines Umsetzungsprojektes gezeigt, nach welchen Überlegungen und Aspekten ein typischer Entscheidungs- bzw. Auswahlprozess für den Einsatz eines spezifischen Web Framework für ein Projekt durchgeführt werden kann. In Kapitel 6 wird die Umsetzung einer Web-Anwendung mit dem Komponenten-basierten Web Framework Wicket beschrieben.

[11] Java Server Faces: `http://java.sun.com/j2ee/javaserverfaces/`
[12] Wicket: `http://wicket.apache.org/`
[13] Spring: `http://www.springsource.org/`
[14] Echo: `http://echo.nextapp.com/site/echo2`

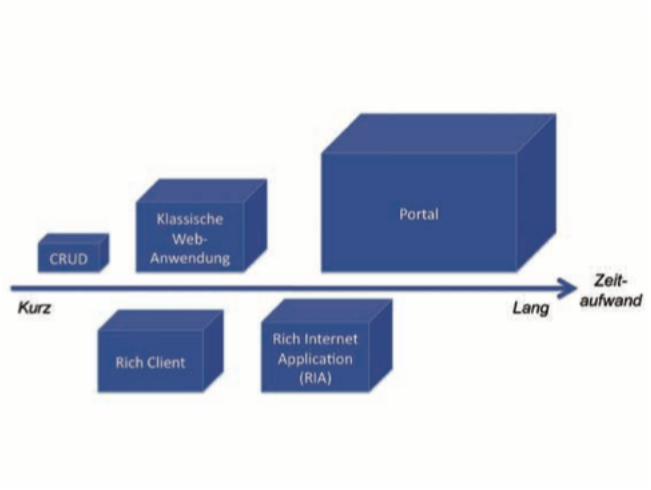

Abbildung 3.6: Kategorisierung von Web Frameworks nach Wähner [2011a]

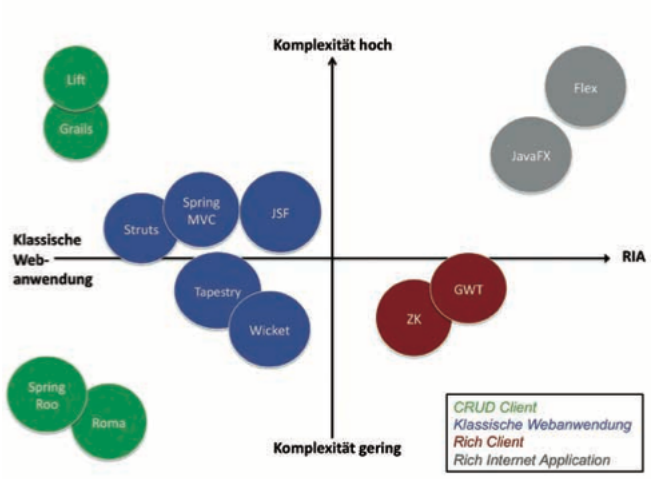

Abbildung 3.7: Darstellung von Komplexität von Web Frameworks in Abhängigkeit des Frameworktyps [Wähner, 2011b]

Kapitel 4

Usability und EduPunk

“ Easy is Hard. ”

[Peter Lewis, NY Times]

Dieses Kapitel ist in zwei Teile gegliedert. Beide Teile behandeln Qualitätsaspekte von Web-Anwendungen die im Kontext dieser Masterarbeit wichtig sind. Der erste Teil dieses Kapitels behandelt im Abschnitt 4.1 den Qualitätsfaktor *Usability*. Dieser Abschnitt zeigt den systemischen Charakter von Usability auf und bietet einen kurzen Überblick über relevante Aspekte aus der Literatur. Es wird festgehalten, dass bei komplexen Software-Projekten, die hinsichtlich Usability hohe Quailitäts-Anforderungen haben, es unabdingbar ist, Usability-Experten im Projektverlauf einzuplanen. Dieses Kapitel schärft und zeigt, dass Usability und Usability-Methoden in jeder Projektphase berücksichtigt werden können. Der zweite Teil dieses Kapitel gibt einen Überblick über die *EduPunk*-Bewegung und der damit verbundenen Attitüde, sich bewusst mit der Schaffung von modernen und flexiblen Lernumgebungen auseinanderzusetzen anstatt auf komplexe und technische E-Learning-Standards zurückzugreifen. Die thematische Auseinandersetzung mit der EduPunk-Bewegung geht mit einem Ziel dieser Masterarbeit (vgl. Kapitel 5, Abschnitt 5.1), eine Web-Anwendung die auch als Lernumgebung sinnvoll genutzt werden kann, einher.

4.1 Usability

Die Qualitätsaspekte Usability, Performance und Sicherheit sind für Web-Anwendungen von besonderer Relevanz. Die Gestaltung einer Web-Anwendung ist immer mit dem Ziel verknüpft, dass Benutzer Ziele effektiv, effizient und zufriedenstellend erreichen können. Andernfalls ist eine Nutzungsverweigerung von Benutzern, aufgrund einer nicht gebrauchstauglichen Anwendung zu erwarten. Es ist dadurch notwendig bei der Gestaltung einer Web-Anwendung auf Nutzungskontexte und auf die Bedürfnisse der Benutzer Rücksicht zu nehmen [Hitz und Leitner, 2003].

4.1.1 Definition

Usability ist ein wichtiges *Aktzeptanzkriterium* im Kontext von interaktiven Anwendungen. Usability zeichnet sich dadurch aus, dass dieser Aspekt oft erst wahrgenommen wird, wenn der Qualitätsfaktor nicht oder schlecht erfüllt ist. Usability kann mit dem Begriff Benutzerfreundlichkeit oder Gebrauchstauglichkeit umschrieben werden. Oftmals wird auch der Begriff Einfachheit verwendet. In der Tat umfasst die Bedeutung von Usability alle drei Begriffe und ist zu dem weitschichtiger. Eine genaue Definition von Usability ist notwendig. In der Literatur wird bei der Definition von Usability durchgängig auf die Internationale Norm ISO/IEC 9214-11 verwiesen [Hitz und Leitner, 2003]:

> *"Usability ist das Ausmaß, in dem ein Produkt durch bestimmte Benutzer in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen."*

Diese Definition beschreibt Anforderungen an die Gebrauchstauglichkeit und leitet sich aus Richtlinien der Interaktion zwischen Mensch und Maschine ab. Diese Definition ist deshalb so interessant, da die wesentlichen Aspekte, die bei der Erstellung einer gebrauchstauglichen Web-Anwendung berücksichtigt werden müssen, berücksichtigt sind. Einerseits handelt es sich um die Benutzer, den Nutzungskontext und um die *Ziele* der Web-Anwendung, die berücksichtigt werden müssen und andererseits um die folgenden drei Aspekte:

Effektivität: Dieser Aspekt ist erfüllt, wenn ein Benutzer die Ziele vollständig und genau erreichen kann.

Effizienz: Dieser Aspekt ist erfüllt, wenn ein Benutzer die Ziele in vertretbaren Aufwand bzw. mit sowenig Aufwand wie möglich erreichen kann.

Zufriedenheit: Dieser Aspekt ist erfüllt, wenn ein Benutzer die Nutzung als zufriedenstellend empfindet, d.h. die Nutzung frei von Beeinträchtigungen ist und im Endeffekt ein positiver Eindruck resultiert. Hitz und Leitner [2003] konstatieren, dass dieser Aspekt mitunter sehr stark von der Ästhetik der Web-Anwendung beeinflusst wird.

Diese Eigenschaften können durchaus in einem hierarchischen Verhältnis zueinander gesehen werden. D.h. das die Eigenschaft Effektivität einen größeren Stellenwert als Effizienz hat und die Eigenschaft Effizienz schlussendlich wichtiger als die Zufriedenheit ist. Fluckiger und Richter [2007] konstatieren, dass aus der Definition nach ISO/IEC 9214-11 hervorgeht, dass Usability nicht ausschließlich eine Eigenschaft der Web-Anwendung ist. Gleichermaßen gibt es kein Regelwerk, dass man einfach so anwenden kann. Usability hat einen systemischen Charakter. Es ist notwendig *in jedem einzelnen Fall* die Ziele des zu erstellenden Artefakts (beispielsweise eine Web-Anwendung), die Benutzer und den Nutzungskontext zu berücksichtigen. Eine Analyse der Benutzer, der Aufgaben und des Anwendungskontextes sowie die daraus folgende Definition des Funktionsumfangs und die Erarbeitung von Abläufen und Prozessen sind essenzielle Aufgaben [Fluckiger und Richter, 2007]. Im Kontext der Definition nach der ISO/IEC 9214-11 Norm wird festgehalten, dass man die Erreichung von Usability prozesshaft betrachten und in den Entwicklungsprozess integrieren soll [Hitz und Leitner, 2003].

Jakob Nielsen sieht neben den Aspekten Effizienz, Effektivität und Zufriedenheit noch die Aspekte *Erlernbarkeit*, *Einprägsamkeit* und die *Fehler* für das Erreichen einer guten Usability als notwendig [Nielsen, 1993]. In der Literatur wird weitgehend auf diese fünf Aspekte bei der Definition und der Beschreibung von Usability verwiesen Holzinger [2005]. Der Aspekt der Erlernbarkeit, bezieht sich auf den Lernprozess und auf den Zeitaufwand, welcher notwendig ist, um ein Produkt bedienen zu können. Dieser Aspekt ist eine Voraussetzung dafür, dass ein Benutzer schlussendlich Ziele erreichen kann. Der Aspekt Einprägsamkeit bedeutet die Forderung, dass nach längerer Aussetzung der Benutzung, kein erneuter Lernprozess notwendig ist. Eine Web-Anwendung mit hoher Usability geht mit einer leichten Einprägsamkeit der Funktionen einher. Fehler können bei der Benutzung von Produkten vorkommen. Es gilt die Unterscheidung in Fehler der Anwendung und Fehler, die der Benutzer verursacht. Dieser Aspekt bezieht sich stark auf die subjektive Zufriedenheit. Zufriedenheit ist neben den oben erwähnten Kriterien erfüllt, wenn die Anwendung funktioniert und keine Fehler aufweist [Nielsen, 1993].

4.1.2 Usability Eingineering

Usability Engineering ist ein Entwicklungsprozess der zum Ziel hat, die angestrebte Usability von Produkten bei der Entwicklung zu erreichen bzw. zu optimieren. Usability ist dabei ein Qualitätsfaktor der

definiert, gemessen und verbessert werden kann [Nielsen, 1993]. Usability Engineering ist zudem ein Prozess, der bei der Erstellung eines Produktes die Reduktion auf das Wesentliche zum Ziel hat. Unnötige Komplexität soll vermieden werden, indem auf die Benutzergruppen als auch Arbeitsabläufe Rücksicht genommen wird, um dadurch die Funktionalität des Produktes auf ein *ideales Minimum* zu reduzieren. Das System soll so konzipiert werden, dass bestimmte Benutzer bei der Anwendung die Ziele optimal erreichen können. Usability Engineering ist kein triviales Unterfangen, sondern erfordert die Anwendung von Usability-Methoden in einem systematischen Prozess. Der Anwendungsbereich von Usability Engineering ist überall dort, wo interaktive technische Systeme für Benutzer konzipiert werden. Heutzutage ist Usability von Produkten zu einem zunehmenden Differenzierungsfaktor geworden und geht oft mit Wettbewerbsvorteilen einher [Fluckiger und Richter, 2007], [Khazaeli, 2005].

Im Kontext von Software-Produkten ist die Entwicklung von Benutzerschnittstellen ein immer wichtiger werdender Faktor. Usability-Methoden können prinzipiell in beliebige Entwicklungsprozessmodelle (beispielsweiese im Software Engineering für die Entwicklung von benutzerorientierter Software) integriert werden. Vorraussetzung ist, dass der zugrundeliegende Entwicklungsprozess ein iteratives Vorgehen ermöglicht. Dadurch können Usability-Ziele als Steuerung verwendet werden. Vereinfacht lassen sich Usability-Methoden in die folgende Aufgabenbereiche integrieren [Fluckiger und Richter, 2007]:

Analyse: Der Benutzer und der Kontext muss analysiert und verstanden werden. Es sollen Techniken des Requirement Engineering ergänzt werden, um zusätzlich Anforderungen aus Benutzersicht zu erhalten. Entsprechende Methoden sind Beobachtungen oder Interviews. *Contextual Inquiry* ist beispielsweise eine Usability-Methode, die eingesetzt werden kann. Dabei handelt es sich um eine Analysetechnik bei der untersucht wird, welche Bedürfnisse Anwender haben, indem diese in einem bestimmten Einsatzumfeld beobachtet und befragt werden.

Modellieren: Beispielsweise durch *Personas* und *Szenarien*, *Storyboards*, *Mock-Ups* oder *UI-Prototyping* soll eine passende Lösung entworfen und optimiert werden, die den Anforderungen genügt. Damit die Anforderungen berücksichtigt werden, muss das System oft aus unterschiedlichen Sichten modelliert werden. Dazu ist es notwendig, zyklisch Entwürfe zu erarbeiten und dementsprechend Feedback einzuholen und wieder in die Modellierung einzuarbeiten.

Spezifizieren: Die resultierende Lösung muss für die Umsetzung spezifiziert werden. Es gilt alle funktionale und nicht-funktionale Anforderungen festzuhalten. Szenarien, Storyboards, UI-Prototypen aber auch Styleguides sind durch entsprechende Usability-Methoden abgeleitete Spezifikationen.

Realisierung: In der Phase der Umsetzung der Lösung sind vorher erarbeitete und aus den Anforderungen abgeleitete UI-Prototypen, Usability Guidelines und Stylguides geeignete Hilfsmittel, die sicherstellen, eine konsistente und regelkonforme Benutzerschnittstelle zu erhalten.

Evaluation: Die umgesetzten Resultate gilt es mit Benutzern zu prüfen und zu optimieren. Dementsprechend gilt es, erstellte Prototypen als auch fertige Produkte mit dem Ziel, Probleme zu erkennen, zu dokumentieren und Verbesserungsmaßnahmen abzuleiten, zu evaluieren. Usability wurde als eine Größe bestimmt, die definiert, gemessen und und durch geeignete Maßnahmen verbessert werden kann. Es gibt eine Menge von Usability-Methoden, die zur Evaluation eines Systems in bestimmten Kontexten herangezogen werden können. Der Einsatz bestimmter Evaluations-Methoden kann auch abhängig vom Entwicklungsgrad des Systemes sein. Holzinger [2005] unterscheidet prinzipiell zwischen zwei Klassen von Techniken, die für Evaluationen herangezogen werden können:

- Inspektionsmethoden
- Testmethoden

Das Ziel beider Methoden ist es, zu gewährleisten, dass ein System geforderte Usability-Charakteristiken (vgl. Abschnitt 4.1.1) aufweist. *Inspektionsmethoden* können verwendet werden, indem die Usability einer Benutzerschnittstelle gegen vorhandene Standards validiert wird [Holzinger, 2005]. In

der Literatur werden die Heuristische Evaluierung (*heuristic evaluation*), kognitive Durchgänge (*cognitive walkthrough*) und die Aktions-Analyse (*action analysis*) als Inspektionsmethoden bezeichnet. Inspektionsmethoden benötigen keine Benutzer, sondern werden von Experten durchgeführt. Usability Probleme sollen erkannt und eine Verbesserungen der Usability erreicht werden. Inspektionsmethoden werden meist bei Prototypen angewendet, können aber auch für fertige Produkte herangezogen werden. Durch Checklisten und Heuristiken kann überprüft werden, ob die Benutzerschnittstelle und seine interaktiven Elemente bewährten Standards genügen. Ein Walkthrough ist eine Inspektionsmethode, in der ein Experte die Funktionalität des Systems untersucht. Es soll das Benutzerverhalten im Kontext der Usability Charakteristik der Erlernbarkeit simuliert werden. *Testmethoden* umfassen die Thinking-Aloud-Evaluation, Feldbeobachtungen und Befragungen bzw. Fragebögen [Holzinger, 2005]. Diese Usability-Tests finden immer direkt mit Benutzern statt. Das Ziel ist es von den Benutzern direkte Informationen zu erhalten, wie das System verwenden wird und welche Probleme bei der Interaktion mit der Benutzerschnittstellen auftreten. Dazu werden Benutzer dabei beobachtet, wie sie mit dem System arbeiten. Die daraus erhaltene Information und das Feedback dient als Verbesserungspotential.

Eine sehr ähnliche Beschreibung und Zuordnung von Usability-Methoden ist in Hitz und Leitner [2003] zu finden. Hier erfolgt eine Zuordnung von Usability-Methoden zu einzelnen Phasen eines typischen Entwicklungsprozess. In diesem Abschnitt erwähnte Usability-Methoden werden in den Phasen Anforderungsanalyse, Entwurf, Implementierung und Einsatz berücksichtigt. Ferner wird auch konstatiert, dass der Entwicklungsprozess prinzipiell ein Iterationen unterstützen muss, da Usability-Ziele nicht in einem Durchgang erreicht werden können.

4.2 EduPunk

In der Literatur wird bei der Beschreibung des Begriffs *EduPunk* immer auf einen Blogeintrag von Jim Groom verwiesen, der diesen *terminus technicus* eingeführt und geprägt hat. Groom kritisierte die zunehmende Kommerzialisierung von Lernplattformen, vorallem mit dem Hintergrund, dass sehr viele pädagogische Aspekte herangezogen und in technische Systeme gepresst werden, die später kommerziell vermarktet werden [Ebner, 2008]. Ein wichtige Rolle spielt auch Stephen Downes, der dem Begriff EduPunk die folgenden Aspkete zugeschrieben hat ([Ebner et al., 2011], [Young, 2008], [Rowell, 2008]): Die Haltung von EduPunk geht damit einher, aus bestehenden Systemen als Reaktion gegen die Kommerzialisierung des Lernens auszubrechen. EduPunk steht auch für eine *do it yourself* (DIY) Attitüde die meint, dass nicht einfach nur auf vorgefertigte Systeme oder E-Learning-Standards zurückzugreifen sei, sondern das Lernen und Lehren aber auch damit verbundene pädagogische Aspekte in den Mittelpunkt gestellt werden müssen und gleichzeitig vorhandene Technologien und einfache Tools sinnvoll eingesetzt werden sollen. EduPunk steht auch dem Einsatz von Technologien kritisch gegenüber. Der Einsatz von Technologie alleine, schafft noch lange keinen Wert für das Lernen. E-Learning Standards werden gleichermaßen kritisiert, da solche Standards immer komplexer werdende Strukturen sind, meist von Technikern und nicht von Pädagogen mit konzipiert wurden und sich in der Regel nicht effizient und effektiv implementieren lassen. Ein weiterer Kritikpunkt gegen E-Learning-Standards ist neben der Kommerzialisierung, dass man nicht jede mögliche Lernsituation in einem Katalog abbilden und standardisieren kann.

Neben der aufgezählten Kritik von EduPunk an bestehende Systeme, stellt sich die Frage, welches Ziel EduPunk verfolgt. EduPunk kann als eine effektive Strategie für Technolgiegestütztes Lernen angesehen werden [Ebner, 2008]. Die Möglichkeiten des WWW, von Web-Anwendungen und modernen Web 2.0 Technologien können, sofern man sie *sinnvoll* einsetzt, dazu verwendet werden, um geeignete Lernräume und Lernumgebungen im WWW schaffen. EduPunk ist ein *mind-set*, welches neben den technologischen Aspekten, die folgende Punkte in den Mittelpunkt rückt:

- Es geht darum verstärkt über das (technologiegestützte) Lernen und das Lehren nachzudenken.
- Man soll sich Gedanken darüber machen, wie eine moderne und flexible Lernumgebung zu gestalten ist.
- Es geht auch darum, pädagogische Aspekte von Lernen zu berücksichtigen und zu verstehen, dass Lernen ein sozialer Prozess ist.
- Zudem soll ein Überblick geschaffen werden, welche Hilfsmittel bereit stehen und wie diese in einer Lernumgebung sinnvoll berücksichtigt werden können.
- Außerdem soll dadurch über die Zukunft von Lernen und das Lehren nachgedacht werden.

EduPunk ist daher zusammengefasst ein Weg technolgiegestütztes Lernen einen neuen Wert zu geben.

Kapitel 5

Anforderung

Dieses Kapitel beschreibt die Anforderungen für den praktischen Teil dieser Masterarbeit und spannt gleichzeitig einen Rahmen für ein konkretes Software- bzw. Umsetzungsprojekt. In Abschnitt 5.1 wird eine Beschreibung der erforderlichen Web Anwendung geliefert und die damit verbundene Motivation, Ziele und Aufgaben dargelegt. Abschnitt 5.2 definiert alle erarbeiteten und abgeleiteten Anforderungen die im Rahmen des Umsetzungsprojektes zu berücksichtigen sind. Abschließend wird in Abschnitt 5.3 argumentiert, nach welchen Kriterien das im Umsetzungsprojekt verwendete Web Framework Apache Wicket [Wicket, 2011] ausgewählt wurde und wie diese Entscheidung zu rechtfertigen ist.

5.1 Motivation, Ziele und Aufgaben

Das Projekt umfasst sehr viele, teils komplexe Aspekte. Dieser Abschnitt liefert eine kurze Systembeschreibung im Kontext der zu erfüllenden Aspekte. Diese Beschreibung ist die Grundlage, um Anforderungen abzuleiten.

5.1.1 Erstellung einer Web-Anwendung für Netzwerkanalyse

Das Ziel der praktischen Arbeit liegt in der Konzeption und Implementierung einer Web-Anwendung für die Analyse von Netzwerkdaten. Bzgl. der Definition des Begriffs Netzwerk und der Theorie der Netzwerkanalyse wird auf das Kapitel 2 verwiesen. Die Web-Anwendung soll ein System darstellen, mit dessen Hilfe man auf möglichst einfache Art und Weise Zugang zu etablierten Netzwerk-Analyse-Methoden bekommt und diese *interaktiv* anwenden kann. Im Vordergrund steht die Erstellung von Netzwerk-Metriken aus Netzwerkdatensätzen, wie beispielsweise die folgenden Berechnungen:

- Gradverteilung (*degree distribution*)
- Cluster-Koeffizient
- Zentralitätsmaße (*centrality*)
- Durchmesser (*diameter*)

5.1.2 Netzwerkdatensätze

Ein Netzwerkdatensatz ist eine geeignete Repräsentation eines Netzwerkes, und eine kompakte Beschreibung eines Netzwerkes, dass aus Knoten und Kanten besteht. Es sollen *einfache* gerichtete und ungerichetete Netzwerke (*one-mode-network*) unterstützt werden. Ein Netzwerkdatensatz kann auch Metadaten beinhalten, beispielsweise eine Beschreibung, Typ, Anzahl Knoten und Kanten. Ein geeignetes Format

soll für Netzwerkdatensätze unterstützt werden. Netzwerkdatensätze können je Benutzer mit der Web-Anwendung hochgeladen werden.

5.1.3 Berechnung von Netzwerk-Metriken

Für die Berechnung von Netzwerk-Metriken sollen die ausgewählte Beispielanwendungen der SNAP-Library (vgl. Kapitel 2, Abschnitt 2.6) verwendet werden. Diese Module stellen für die Web-Anwendungen die Kernfunktionalität für den Aspekt der Netzwerkanalyse bereit. Diese Netzwerkanalyse-Module müssen durch eine geeignete Strategie und Architektur in das Gesamtsystem der Web-Anwendung integriert werden. Die Netzwerkanalyse-Module müssen implizit über eine intuitive Benutzerschnittstelle bedient werden können. Netzwerkanalyse-Module sind fertige Programme, die in einer Konsole ausgeführt werden können. Ein Netzwerkanalyse-Modul kann durch Parameter gesteuert werden. Details der Module, Parameter usw. sollen durch eine stringente Benutzerschnittstelle und ein einfaches und durchdachtes Bedienkonzept versteckt werden. Durch eine geeignete und generische Architektur soll das System sehr einfach erweiterbar sein, damit man zu einem späteren Zeitpunkt möglichst beliebige Netzwerkanalyse-Module in das System einbetten kann.

Die Web-Anwendung soll einen Zugang je Benutzer unterstützen. Die Web-Anwendung soll Benutzern die Möglichkeit bieten sich anzumelden. Jeder Benutzer soll mit der Web-Anwendung seine eigene Netzwerkdatensatz-Sammlung verwalten können. Netzwerkdatensätze können zu einer Sammlung hinzugefügt und gelöscht werden. Beinhaltet ein Netzwerkdatensatz spezifische Metadaten, dann sollen diese im System angezeigt werden. Netzwerkdatensätze können ausgewählt werden, und durch die im System vorhandenen Netzwerkanalyse-Module ausgewertet werden. Es soll möglich sein mehrere Analysen durchzuführen. Eine laufende Analyse soll das Gesamtsystem nicht blockieren.

Das Ergebnis einer spezifischen Netzwerkanalyse sind Netzwerk-Metriken. Netzwerk-Metriken beinhalten Diagramme (PNG-Format), erstellte Datensätze, aber auch GNUPlot-Skripte. Alle berechneten Netzwerk-Metriken bleiben im System für einen Benutzer gespeichert. Eine Netzwerkanalyse soll nur initial für einen Netzwerkdatensatz durchgeführt werden. Ist eine Berechnung durch den Benutzer bereits durchgeführt worden, dann soll das Ergebnis der Berechnung wiederverwendet werden. Für einen Benutzer muss es ersichtlich sein, ob eine Netzwerkanalyse durchgeführt oder ob das Ergebnis einer Berechnung nur aufgerufen werden muss. Auf die Netzwerk-Metriken sollen einfach zugegriffen werden. Einzelne Datein sollen downloadbar sein. Diagramme sollen im System visualisiert werden.

Das System muss dafür ausgelegt sein, dass einzelne Berechnungen sehr rechenintensiv sind. Es ist davon auszugehen, dass Berechnungen einige Stunden, bis Tage in Anspruch nehmen können.

5.1.4 Einsatz als *usable* E-Learning Instrument: Usability und EduPunk

Die resultierende Web-Anwendung soll schlussendlich unterstützend für eine Lehrveranstaltung eingesetzt werden können, wodurch sich die Anforderung ableitet, dass die Zielgruppe des Systems keinen Expertenstatus haben muss und das besonders Usability als Qualitätsfaktor für die Web-Anwendung erfüllt sein muss. Das System soll einfach zu bedienen, benutzerfreundlich und gebrauchstauglich sein. Es soll eine erweiterbare Lernumgebung geschaffen werden, mit deren Hilfe ein leichter Zugang zu der komplexen Thematik der Netzwerkanalyse ermöglicht wird.

Neben der Einhaltung und der Verwendung von Web Standards (*technische Umsetzung*) stellt sich die Herausforderung eine Lernumgebung zu schaffen. Diese Arbeit verfolgt nicht das Ziel E-Learning Standards zu erfüllen, sondern folgt der DIY (*Do It Yourself*) Attitüde der *EduPunk* Bewegung (vgl. Kaptitel 4 Abschnitt 4.2), mit der Einsicht, dass Technologie Bildung alleine noch nicht wertvoll macht – son-

dern es vielmehr darum geht, entsprechende Umgebungen zu schaffen, in der die Interessen der Lernenden geweckt werden sollen. Dieses Ziel kann durch den vernünftigen Einsatz von Web-Technologie unterstützend erreicht werden. Eine Voraussetzung ist unter anderem, dass das resultierende System *useable* gestaltet wird, daher werden Aspekte der Web-Usability (vgl. Kaptitel 4 Abschnitt 4.1) in dieser Arbeit ebenfalls berücksichtigt.

5.1.5 Evaluierung der Usability

Die Web-Anwendung soll in einer abschließenden informalen Evaluierung (vgl. Kapitel 7) durch eine Zielgruppe bestehend aus

- Experten
- Studenten
- Pädagogen

bewertet werden. In Kaptitel 4 Abschnitt 4.1 wurden Inspektions- und Testmethoden kurz vorgestellt. Solche Evaluierungen sind aufgrund der zeitlichen Ressourcen, die mit dem Rahmen der Masterarbeit einhergehen nicht möglich *lege artis* und wissenschaftlich anzuwenden und durchzuführen. Das Ziel der Evaluierung ist es, zukünftige Akteure in den Entwicklungsprozess einzubeziehen. Durch diese Feedback-Maßnahme soll man eine Bewertung der resultierenden Qualität der Web-Anwendung erhalten, damit geeignete Maßnahmen abgeleitet werden können, die helfen das System in weiteren Iterationszyklen und Folgeprojekten zu verbessern.

5.1.6 Berücksichtigung von Standards, Frameworks und vorhandenem Architekturwissen im Umsetzungsprozess

Die Erstellung der Web-Anwendung soll die folgenden Rahmenbedingungen berücksichtigen:

- Die Webapplikation soll mit Java umgesetzt werden. Alle Tools und Libraries die mit der Java Enterprise Edition geliefert werden, sollen soweit deren Einsatz sinnvoll ist, auch verwendet werden.
- Ein geeignetes Web Applikation Framework soll evaluiert werden und für die Umsetzung der Web Applikation verwendet werden. Das ausgewählte Framework soll eine MVC-Architektur unterstützen.
- Die Web-Anwendung soll Web-Standards in vernünftiger Weise einhalten. Es soll XHTML verwendet werden. Die Formatierung soll per CSS erfolgen.
- Die Sprache der Benutzerschnittstelle soll English sein. Das System soll aber so ausgelegt werden, dass Mehrsprachigkeit prinzipiell möglich ist.
- Die Web-Anwendung soll ausreichende Konfigurationsmöglichkeiten zur Verfügung stellen. Eine Konfiguration soll soweit sinnvoll per XML oder Property Dateien möglich sein.

5.2 Anforderungen

Dieser Abschnitt gibt einen Überblick über alle abgeleiteten Anforderungen an die geforderte Web-Anwendung. Die Anforderungen wurden anhand der Beschreibung und den Aspekten aus Abschnitt 5.1 abgeleitet und folgend unterteilt:

- Funktionale Anforderungen
- Nicht-funktionale Anforderungen
- Technische Randbedingungen

In Kapitel 3 Abschnitt 3.3.1 wurde beschrieben, dass diese Einflussfaktoren (vgl. Abbildung 5.1) bei der Erstellung der Architektur einer Anwendung zu berücksichtigen sind:

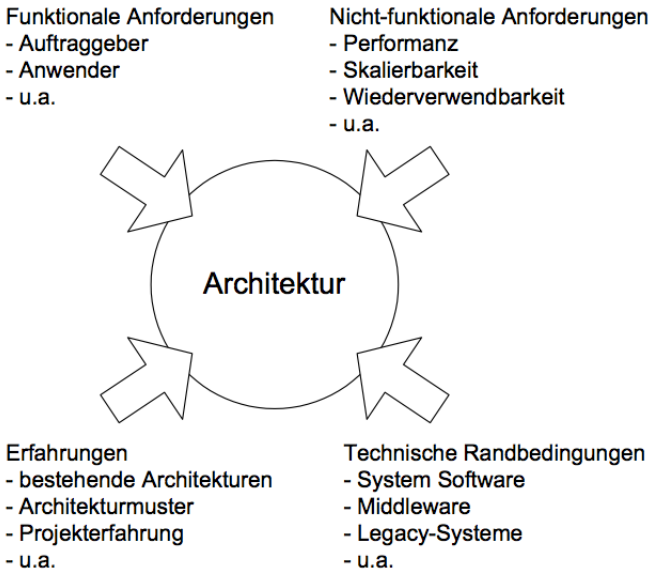

Abbildung 5.1: Einflussfaktoren auf die Entwicklung einer Architektur [Eichinger, 2004] (nach [Jacobson et al., 1999])

5.2.1 Funktionale Anforderungen

Die funktionalen Anforderungen legen fest, was die Web-Anwendung tun und können soll. Aus Gründen der Einfachheit werden die erarbeiteten Anforderungen in den folgende Kontexten beschrieben:

- Benutzer
- Netzwerkdatensatz
- Netzwerkanalyse-Module
- Netzwerk Metriken
- Zeit

5.2.1.1 Benutzer

Ein Benutzer kann mit der Web-Anwendung interagieren und verschiedene Aktionen ausführen. In diesem Kontext ergeben sich die folgenden Anforderungen:

Anforderung 1: Die Web-Anwendung kann nur durch authentifizierte Benutzer verwendet werden.

Anforderung 2: Benutzer können sich im System an- und abmelden.

Anforderung 3: Neue Benutzer können hinzugefügt werden.

Anforderung 4: Benutzer können Netzwerkdatensätze hinzufügen, löschen und downloaden.

Anforderung 5: Benutzer können Netzwerkdatensätze auswählen und danach Netzwerkanalyse-Berechnungen durchführen.

Anforderung 6: Benutzer können Netzwerk Metriken von Netzwerkanalyse-Berechnungen einsehen und einzelne Ergebnisse downloaden.

Anforderung 7: Benutzer können erstellte Netzwerk-Metriken löschen oder die Netzwerk-Metriken neu berechnen.

5.2.1.2 Netzwerkdatensatz

Die Web-Anwendung soll die Analyse von Netzwerken ermöglichen. Ein Netzwerk wird durch einen Netzwerkdatensatz repräsentiert. Ein Benutzer soll einen Netzwerkdatensatz mit der Web-Anwendung analysieren können. Dabei werden bestimmte Eigenschaften von Netzwerken berechnet und stehen als Netzwerk-Metriken bereit. In diesem Kontext ergeben sich die folgenden Anforderungen:

Anforderung 8: Die Web-Anwendung berücksichtigt einfache, gerichtete sowie ungerichtete Netzwerke (*one-mode-networks*).

Anforderung 9: Einfache Netzwerke werden durch einen Netzwerkdatensatz repräsentiert.

Anforderung 10: Ein Netzwerkdatensatz beschreibt anhand einer Liste von Knotenpaaren, welche Knoten in einem Netzwerk miteinander verbunden sind. In jeder Zeile dieser Liste ist ein Eintrag, der die Form $FromNodeI \; \backslash t \; ToNodeId$ besitzt (vgl. die Zeilen 4 bis 10 in Listing 5.1).

Anforderung 11: Ein Netzwerkdatensatz kann Metadaten (Typ des Netzwerkes, Beschreibung, Anzahl Knoten und Kanten) beinhalten (vgl. die Zeilen 1 bis 3 in Listing 5.1).

Anforderung 12: Metadaten eines Netzwerkdatensatz können in der Web-Anwendung eingesehen werden.

```
1  # Directed graph (each unordered pair of nodes is saved once): soc-Epinions1.txt
2  # Directed Epinions social network
3  # Nodes: 75879 Edges: 508837
4  # FromNodeId  ToNodeId
5  0 4
6  0 5
7  4 25
8  4 27
9  4 28
10 4 30
```

Listing 5.1: Format eines Netzwerkdatensatz. Datensatz wurde entnommen aus [SNAP, 2011c].

5.2.1.3 Netzwerkanalyse-Module

Die Web-Anwendung muss spezifische Netzwerkanalyse-Methoden als Dienst bereitstellen. Dadurch wird ermöglicht, dass ein Benutzer bestimmte Netzwerk-Metriken von Netzwerkdatensätzen erstellen kann. In diesem Kontext ergeben sich die folgenden Anforderungen:

Anforderung 13: Netzwerkanalyse-Module sind in die Web-Anwendung integriert und bilden die Basis für die Erstellung von Netzwerk-Metriken.

Anforderung 14: Netzwerkanalyse-Module können von Benutzern verwendet werden.

Anforderung 15: Netzwerkanalyse-Module berechnen bestimmte Eigenschaften von Netzwerken (Netzwerk-Metriken).

Anforderung 16: Netzwerkanalyse-Module haben eine aussagekräftige Bezeichnung und eine kurze Beschreibung.

Anforderung 17: Netzwerkanalyse-Module sollen durch eine XML-Konfiguration in das System integriert werden.

Anforderung 18: Es werden initial die Netzwerkanalyse-Module $netstat$ und $centrality$ der SNAP-Library ((vgl. Kapitel 2, Abschnitt 2.6)) verwendet. Dadurch soll die Web-Anwendung die folgenden Netzwerkanalyse-Methoden anbieten:

- cumulative degree distribution
- degree distribution
- hop plot (diameter)
- distribution of weakly connected components
- distribution of strongly connected components
- clustering coefficient
- singular values
- left and right singular vector
- centrality

5.2.1.4 Netzwerk Metriken

Eine Netzwerk Metrik ist das Resultat einer Netzwerkanalyse, die ein Benutzer mit einem Netzwerkdatensatz durchgeführt hat. In diesem Kontext ergeben sich die folgenden Anforderungen:

Anforderung 19: Netzwerk Metriken haben ein Erstellungsdatum.

Anforderung 20: Erstellte Netzwerk Metriken bleiben in der Web-Anwendung für einen Benutzer und ein Netzwerkanalyse Modul gespeichert und können eingesehen werden.

Anforderung 21: Netzwerk Metriken können gelöscht werden.

Anforderung 22: Netzwerk Metriken können neu berechnet werden.

Anforderung 23: Netzwerk Metriken können aus Diagrammen (PNG-Format), GNUplot Dateien und Text-Dateien bestehen.

5.2.1.5 Zeit

In diesem Kontext ergeben sich die folgenden Anforderungen:

Anforderung 24: Die Web-Anwendung muss so konzipiert werden, dass eine laufende Berechnung das System und die Benutzerschnittstelle nicht blockiert.

Anforderung 25: Es können mehrere Berechnungen parallel gestartet und ausgeführt werden.

Anforderung 26: Berechnungen können sehr rechenintensiv sein und Stunden bis einige Tage in Anspruch nehmen.

Anforderung 27: Laufende Berechnungen müssen in der Benutzerschnittstelle ersichtlich sein.

5.2.2 Nicht-funktionale Anforderungen

Nicht-funktionale Anforderungen definieren Qualitätsaspekte und Rahmenbedingungen, die bei der Umsetzung der Funktionalität der Web-Anwendung berücksichtigt werden müssen. In den folgenden beiden Abschnitten werden wichtige Qualitätsaspekte und Rahmenbedingungen identifiziert.

5.2.2.1 Qualitätsaspekte

Qualitätsaspekte der Web-Anwendung werden nach den Faktoren Usability, Security, Performance und Erweiterbarkeit unterteilt.

Usability

Anforderung 28: Die Web-Anwendung soll einfach und intuitiv zu bedienen sein.

Anforderung 29: Die Web-Anwendung soll als Lernumgebung eingesetzt werden.

Security

Anforderung 30: Eine Benutzername und Password Authentifizierung von Benutzern muss unterstützt werden.

Performance

Anforderung 31: Die Berechnung von Netzwerk-Metriken kann in Abhängigkeit der Größe des Netzwerks sehr rechnenintensiv sein.

Anforderung 32: Laufende Berechnungen sollen das Benutzerinterface nicht blockieren.

Erweiterbarkeit

Anforderung 33: Die Integration neuer Netzwerkanalyse-Module muss einfach möglich sein.

5.2.3 Technische Randbedingungen

In diesem Abschnitt werden technologische Randbedingungen beschrieben.

Anforderung 34: Als Entwicklungsplattform soll die Java Enterprise Edition[1] (Java EE) eingesetzt werden.

Anforderung 35: Als Web Framework soll Apache Wicket [Wicket, 2011] eingesetzt werden (vgl. Abschnitt 5.3).

Anforderung 36: Die Web-Anwendung soll auf einem Apache Tomcat[2] Web Server betrieben werden.

[1] Java Enterprise Edition: `http://www.oracle.com/technetwork/java/javaee/overview/index.html`

[2] Apache Tomcat: `http://tomcat.apache.org/`

5.3 Auswahl eines geeigneten Web Framework

5.3.1 Motivation

Im Rahmen des Umsetzungsprojektes ist es notwendig für die Erstellung der Web-Anwendung ein geeignetes Web Framework einzusetzen. In Kapitel 3 Abschnitt 3.5 wurden Aspekte von Web Application Frameworks bzw. Web Frameworks im Detail betrachtet. In diesem Abschnitt wurden ausgewählte vorhandene Web Frameworks anhand der Beschreibung aus Abschnitt 5.1 und den Anforderungen aus Abschnitt 5.2 miteinander verglichen. Zusätzliche Kriterien wie Komplexität, Lernkurve, notwendige Skills, Werkzeuge (Tools, Plugins) usw. werden in den Betrachtungen mit einbezogen. Es galt die Einschränkung, dass nur Web Frameworks betrachtet werden, die auf einer Model-View-Controller Architektur basieren. Das Model-View-Controller Muster wurde in Kapitel 3 Abschnitt 3.4 im Detail beschrieben. Ein weitere Einschränkung war, dass das Web Framework Java basiert ist. Mit dem ausgewählten Web Framework soll die Präsentation-Schicht der Web Anwendung implementiert werden.

5.3.2 Auswahl

Es gibt eine schier unüberschaubare Menge an Web Frameworks. Für die erfolgreiche Umsetzung dieses Projektes ist eine geeignete Auswahl ein wichtiger Faktor, deshalb wurde im Rahmen der Masterarbeit eine ausführliche Recherche durchgeführt. Im Web wird über dieses Thema sehr viel diskutiert und es existieren in Fachzeitschriften sehr viele hilfreiche Artikel (bespielsweise [Wähner, 2011a], [Wähner, 2011b]). Damit man eine rationale Entscheidung treffen kann, sind die folgenden Bereiche mit einzubeziehen:

- Anforderungen an das Projekt
- Erfahrung
- Domänenwissen über Frameworks, Web-Engineering, Web-Standards und Trends

Die Einschränkung ein Web Framework das Java und MVC basiert ist zu verwenden ist das initiale Ausschlusskriterium. Das zweite Ausschlusskriterium wurde aus den Anforderungen abgeleitet, aus denen hervorgeht, dass es nicht notwendig ist eine Rich Internet Application (RIA) zu erstellen. Aus diesen Grund werden RIA-basiserte Web Frameworks in der weiteren Betrachtung nicht mehr berücksichtigt. Ein weiteres Ausschlusskriterium war, das kein Plugin im Browser für die Web-Anwendung installiert werden soll. In der Regel trifft das auch auf RIA-basierte Web Frameworks wie JavaFX[3] oder Adobe Flex[4] zu, die beispielsweise als Browser-Plugin das Java Runtime Environment (JRE) oder den Adobe Flash Player benötigen.

Nach einer eingehenden Recherche wurden die folgenden vier Web Frameworks als geeignete Kandidaten identifiziert:

- Java Server Faces (JSF)[5]
- Apache Wicket[6]
- Google Web Toolkit (GWT)[7]

[3] JavaFx: http://javafx.com/
[4] Adobe Flex: http://www.adobe.com/de/products/flex/
[5] Java Server Faces (JSF): http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html
[6] Apache Wicket: http://wicket.apache.org/
[7] Google Web Toolkit (GWT): http://code.google.com/intl/de/webtoolkit/

	Apache Wicket	Apache Tapestry	GWT
Reifegrad	gut	gut	sehr gut
Komplexität	gering	gering	mittel
Produktivität	sehr hoch	sehr hoch	hoch
Erlernbarkeit/Lernkurze	sehr gut	gut	gut
Notwendige Skills	Java, XHTML	JAVA, TML[9]	Java, XML[10]
Unterstützung (Tools, Plugins)	sehr gut	gut	sehr gut
Community	sehr gut	gut	sehr gut
Dokumentation	befriedigend	gut	gut
Beispiele/Tutorials	gut	gut	sehr gut

Tabelle 5.1: Vergleich von geeigneten Web Frameworks

- ApacheTapestry[8]

Alle vier Frameworks sind komponenten-basiert [Wähner, 2011a]. In Kapitel 3 Abschnitt 3.5 wurden Komponenten-basierte Web Frameworks näher beschrieben. JSF, Apache Wicket und Apache Tapestry sind Frameworks für klassische Web-Anwendungen und können durch die durchgängige Ajax-Unterstützung auch Rich Client Aspekte berücksichtigen. GWT ist ein Framework für die Erstellung von Rich Client Anwendungen. Ein weiteres Alleinstellungsmerkmal von GWT im Vergleich zu den anderen Frameworks ist sein clientzentrischer Ansatz [Wähner, 2011a]. Die Abbildung 3.7 zeigt die Einordnung dieser Web Frameworks nach den Kriterien Komplexität und Frameworktyp.

Die Frameworks JSF, Apache Wicket und Apache Tapestry sind sehr ähnlich zu einander. Die Einsatzmöglichkeiten dieser Frameworks hängen mitunter sehr von der *Projektgröße* ab. Im Rahmen dieser Masterarbeit ist eine in Anbetracht kleine Web-Anwendung zu entwickeln, die eine hohe Benutzerfreundlichkeit, Gebrauchstauglichkeit und Einfachheit aufweisen soll. Wähner [2011a] konstatiert, dass Wicket und Tapestry sehr geeignet für kleine und mittelgroße Web-Anwendungen sind und eine hohe *Produktivität* aufweisen. Hohe Produktivität wird erreicht,,, da die Entwicklung hauptsächlich mit Java erfolgt. Wicket bietet sehr gute objektorientierte Unterstützung und Tapestry arbeitet stark annotationsbasiert. JSF hingegen –ein Web Framework Standard– entwickelt sein Potential erst bei sehr großen Web-Anwendungen, bietet aber Vorteile wie Komponentenbibliotheken und sehr gute Dokumentation und Fachliteratur [Wähner, 2011a]. Aufgrund der Argumentation bzgl. der Produktivität in Anbetracht der zu erwarteten Größe der Web-Anwendung wurde entschieden, dass Wicket und Tapestry mögliche Kandidaten sind und bei der weiteren Betrachtung der JSR-Standard Java Server Faces nicht mehr berücksichtigt wird.

Schlussendlich musste zwischen den Web Frameworks Apache Wicket, Apache Tapestry und GWT entschieden werden. Alle funktionalen Anforderungen und daraus abgeleiteten Lösungsmöglichkeiten (Fileupload, Ajax, Comet-Prinzip bzw. Long-Polling) können mit jeden dieser Frameworks abgebildet werden. Die Entscheidung musste anhand einer Reihe von Kriterien getroffen werden. Aus diesem Grund wurden die Kriterien Reifegrad, Komplexität (vgl. Abbildung 3.7), Produktivität, Erlernbarkeit/Lernkurve, Notwendige Skills, Unterstützung (Tools, Plugins), Community, Dokumentation und Beispiele definiert. Die Tabelle 5.1 zeigt den Vergleich der Web Frameworks anhand der Bewertung der spezifischen Kriterien im Kontext der Anforderungen.

[8] Tapestry: `http://tapestry.apache.org/`

5.3.3 Entscheidung

Im Rahmen dieser Masterarbeit wurde schlussendlich entschieden, für die Erstellung der Web-Applikation das komponenten-basierte Web Framework Apache Wicket zu verwenden. Der Einsatz jedes Web Frameworks ist immer mit einer Reihe von Vor- und Nachteilen verbunden. Ausschlaggebend war die geringe Komplexität bei hoher Produktivität, die leichte Erlernbarkeit und das die Skills Java und XHTML (und CSS) ausreichend sind. Der Nachteil der schlechten Dokumentation wird aufgrund der zahlreichen Beispiele inkl. Codedokumentation und vorhandener Fachliteratur (wie beispielsweise [Förther et al., 2009] und [Mosmann, 2009]) in Kauf genommen. Ajax wird von diesem Framework unterstützt und ist durch eigene Komponenten gekapselt. Nachteile wie fehlende Komponenten-Bibliotheken usw. gehen mit dieser Entscheidung einher und müssen bei der Umsetzung getragen werden. Tapestry ist wiederum sehr ähnlich zu Wicket – schlussendlich scheint Wicket in Anbetracht dieses Projektes die besser Alternative zu sein. GWT würde Vorraussetzungen mitbringen einen Rich Client-Lösung bestmöglich zu unterstützen, bietet zudem auch sehr gute Komponentenbibliotheken und wäre technologisch auch sehr interessant. Die höhere Lernkurve und Komplexität spricht aber im Rahmen dieses Projektes gegen dieses Web Framework.

Kapitel 6

Umsetzung

In Kapitel 5 wurde die Motivation, Zielsetzung und die Anforderungen für den praktischen Teil dieser Masterarbeit definiert und im Detail beschrieben. In den Kapiteln 2, 3 und 4 wurden alle notwendigen theoretische Aspekte für die praktische Umsetzung dieser Arbeit behandelt. In diesem Kapitel wird die praktische Umsetzung der geforderten Web-Anwendung beschrieben, wodurch der Konnex zu den vorhergehenden Kapiteln geschlossen wird.

Dieses Kapitel beschreibt den Lösungsweg und einige ausgewählte Details, die für die Erstellung der Web-Anwendung maßgeblich sind. Im Laufe der Umsetzungsphase hat sich für die Web-Anwendung der Name *Network Metric Explorer* etabliert. Dieses Kaptitel startet in Abschnitt 6.1 mit der Vorstellung der erarbeiteten Systemarchitektur, dass durch eine Einteilung in logische Schichten grob alle notwendigen Komponenten und deren Beziehungen zueinander zeigt. Anschließend werden verwendete Technologien und Werkzeuge in Abschnitt 6.2 vorgestellt, mit deren Hilfe die Web-Anwendung schlussendlich umgesetzt worden ist. Das realisierte System wird in Abschnitt 6.3 vorgestellt und anhand ausgewählter Konzepte und Implementierungsdetails genauer beschrieben.

6.1 Architektur – Network Metric Explorer

In Kapitel 3 wurden wichtige Aspekte angeführt, die bei der Erarbeitung der Architektur berücksichtigt wurden. Die Beschreibung der Architektur wird in zwei Teile gegliedert. In Abschnitt 6.1.1 wird das Konzept und die beteiligten Komponenten beschrieben. In Abschnitt 6.1.2 wird die resultierende Systemarchitektur dargelegt. Ferner werden kritische Aspekte und Herausforderungen festgehalten, die während der Umsetzung gelöst werden mussten.

6.1.1 Konzept Network Metric Explorer

In Kapitel 5 Abschnitt 5.2 wurden die folgenden zentralen Akteure identifiziert:

- Benutzer (vgl. Abschnitt 5.2.1.1)
- Netzwerkdatensatz (vgl. Abschnitt 5.2.1.2)
- Netzwerkanalyse-Module (vgl. Abschnitt 5.2.1.3)
- Netzwerk Metriken (vgl. Abschnitt 5.2.1.4)

Diese Akteure werden in der Konzeption der Architektur im Kontext der damit verbundenen Anforderungen und als Komponenten berücksichtigt.

Der wesentliche Nutzen dieses Systems ist, dass ein Benutzer einen Netzwerkdatensatz in das System hochladen und mit vorhandenen Netzwerkanalyse-Modulen spezifische Netzwerk-Metriken erstellen sowie einsehen kann. Prinzipiell ist es möglich, dass das System unmittelbar nach dem Hochladen eines Netzwerkdatensatzes alle im System registrierten Netzwerkanalyse-Methoden auf diesen Datensatz anwendet und auswertet, sodass alle Netzwerk-Metriken automatisch berechnet werden. Dieses Szenario hat den Vorteil, dass ein Benutzer im System keine Berechnungen starten müsste, sondern sich nur auf das Einsehen der einzelnen Netzwerk-Metriken konzentrieren kann. Dieses Szenario hätte allerdings auch einige Nachteile, denn es werden potentiell unnötige Netzwerkanalysen durchgeführt und ferner können damit verbunden Performanz-Probleme entstehen. Ein weiterer Nachteil ist, dass wenn zukünftig neue Netzwerkanalyse-Module in das System integriert werden und das System wiederum autonom alle ausstehenden Berechnungen durchführen müsste.

Schlussendlich wurde die Idee der autonomen Berechnung nicht weiterverfolgt. Jedoch wurde der Aspekt, dass das System ab den Zeitpunkt des Hochladens eines Netzwerkdatensatzes alle korrespondierenden Netzwerk-Metriken kennt (auch zukünftige) in die Konzeption mit aufgenommen. Ferner wurde auch der Aspekt berücksichtigt, einen Benutzer nicht mit der Komplexität des Startens einer Berechnung zu konfrontieren. Aus diesen Gesichtspunkten entstand die Idee des *Network Metric Explorer*, einem System, dass es erlaubt von Netzwerkdatensätzen durch *browsing* –Navigation in einer Hypermedia-Struktur– alle möglichen Netzwerk-Metriken zu erkunden. Ein Benutzer kann in einem Kontext navigieren und kann dabei implizit *on demand* eine neue Netzwerk-Metrik generieren. Netzwerk Metriken werden im Navigations-Kontext von Benutzer, Netzwerkdatensatz und Netzwerkanalyse-Module nur dann erstellt, sofern diese noch nicht vorhanden sind. In Abbildung 6.1 wurde das Prinzip und die Beziehungen der einzelnen Komponenten in einer Konzeptionellen Architektur manifestiert. Vorhandene Netzwerk-Metriken werden im Network Metric Explorer im Kontext eines Benutzers, Netzwerkdatensatzes und einer Netzwerkanalyse-Methode angezeigt. Nicht-Vorhandene Netzwerk-Metriken werden im Network Metric Explorer im Kontext eines Benutzers, Netzwerkdatensatz und einer Netzwerkanalyse-Methode *on demand* folgendermaßen erstellt und eingesehen:

- Der Network Metric Explorer verwendet ein konkretes Netzwerkanalyse-Modul, um die Berechnung einer Netzwerk-Metrik durchzuführen.
- Ein Netzwerkanalyse-Modul führt die Berechnung mit dem Netzwerkdatensatz durch und generiert eine neue Netzwerk-Metrik.
- Der Network Metric Explorer kann nach der Berechnung die Netzwerk-Metrik anzeigen.

6.1.2 Systemarchitektur

Die Architektur der resultierenden Web-Anwendung muss eine Reihe komplexer Aspekte und Funktionen berücksichtigen, um den Anforderungen (vgl. Kapitel 5) zu genügen. Beispielsweise ergibt sich hohe Komplexität unmittelbar aus der Anforderung, bestehender Netzwerkanalyse-Module als Basis für die Berechnung von Netzwerk-Metriken in das System, geeignet zu integrieren. Folgende Aspekte sind hier zu berücksichtigen:

- Einzelne Netzwerkanalyse-Module sind für die praktische Umsetzung bereitgestellt und liegen in Form von eigenen Modulen vor. In der ersten Phase sollen ausgewählte Module der SNAP-Library verwendet werden.
- Durch Kommandozeilenparameter ist es möglich einzelne Parameter wie Datensatz und Berechnungsart an die Berechnungsmodule zu übergeben.

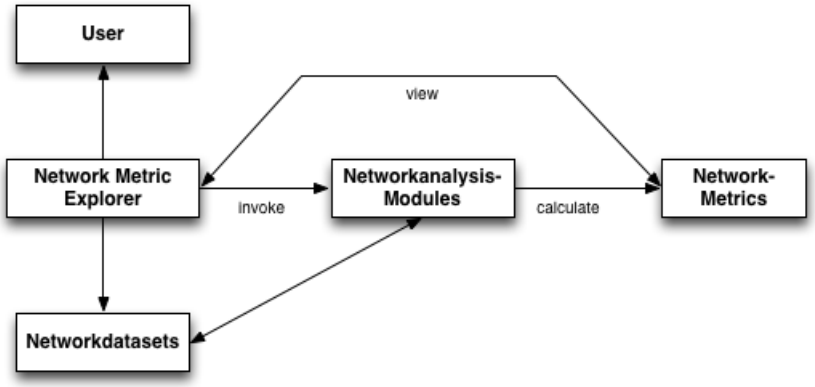

Abbildung 6.1: Darstellung Konzept und einzelne Komponenten

- Berechnungsmodule laufen auf Betriebssystem-Ebene in einem eigenen Prozess.
- Der Berechnungsvorgang kann abhängig vom Netzwerkdatensatz und der Berechnungsmethode sehr rechenintensiv sein. Es ist nicht ungewöhnlich, dass ein Berechnungsvorgang einige Stunden bis Tage in Anspruch nehmen kann.
- Der Berechnungsvorgang soll das Gesamtsystem nicht blockieren.
- Neue Berechnungsmodule sollen per Konfiguration in das System integriert werden können.

Die initiale Herausforderung war es ein Konzept zu finden, welches erlaubt diese Netzwerkanalyse-Module in das System zu integrieren und funktional zu nutzen. Folgende Ansätze wurden mit dem Ziel standardisiert und einfach Netzwerkanalyse-Module im System zu verwenden, in Betracht gezogen:

- Die Netzwerkanalyse-Module werden durch eine geeignete Schnittstelle als ein externes Web Service (SOAP/REST) zur Verfügung gestellt. Die Web-Anwendung kann die Schnittstelle einfach benutzen. Dieser Lösungsansatz erscheint sauber, hat aber zur Folge, dass die oben beschreibenden Probleme in einem anderen System verfrachtet werden und dort gelöst werden müssen.
- Es wird in Form einer API auf die externen Netzwerkanalyse-Module zugegriffen.

Der Aspekt der Integration bestehender Berechnungsmodule wurde durch das Entwurfsmuster Façade (vgl. Gamma et al. [1995]) gelöst. Dieses Strukturmuster wird eingesetzt, um eine einfache Schnittstelle und eine möglichst lose Kopplung zu Subsystemen zu ermöglichen. Durch diese Maßnahme soll auch erreicht werden, dass die Komplexität der Subsysteme versteckt wird. Netzwerkanalyse-Moduke sind in der Architektur durch den abstrakteren und generischeren Begriff Service-Module bezeichnet. Service-Module sind in einer geeigneten Struktur am Applikationsserver abgelegt und können später durch eine klar definierte Schnittstelle der Service-Module-Façade aufgerufen werden.

Damit die Komplexität des Gesamtsystems beherrschbar wird, wurde das System in einzelne logische Schichten getrennt, wobei jede Schicht einen klar definierten Aufgaben- und Funktionsbereich hat (vgl. [Buschmann et al., 1996]). Es gilt die Regel, dass jeder Layer Dienste für die obere Schicht zur Verfügung stellt. Ein Layer kann Aufgaben an die untere Schicht weitergeben. Durch diese Einteilung soll es später möglich sein, einzelne Layer auszutauschen oder wiederzuverwenden. Dieser Vorgang hat auch für die konkrete Umsetzung Vorteile. Durch den *Seperation of Concerns*-Ansatz können in jeder Schicht einzelne Probleme lokalisiert werden, um dort effektive Lösungen zu erarbeiten.

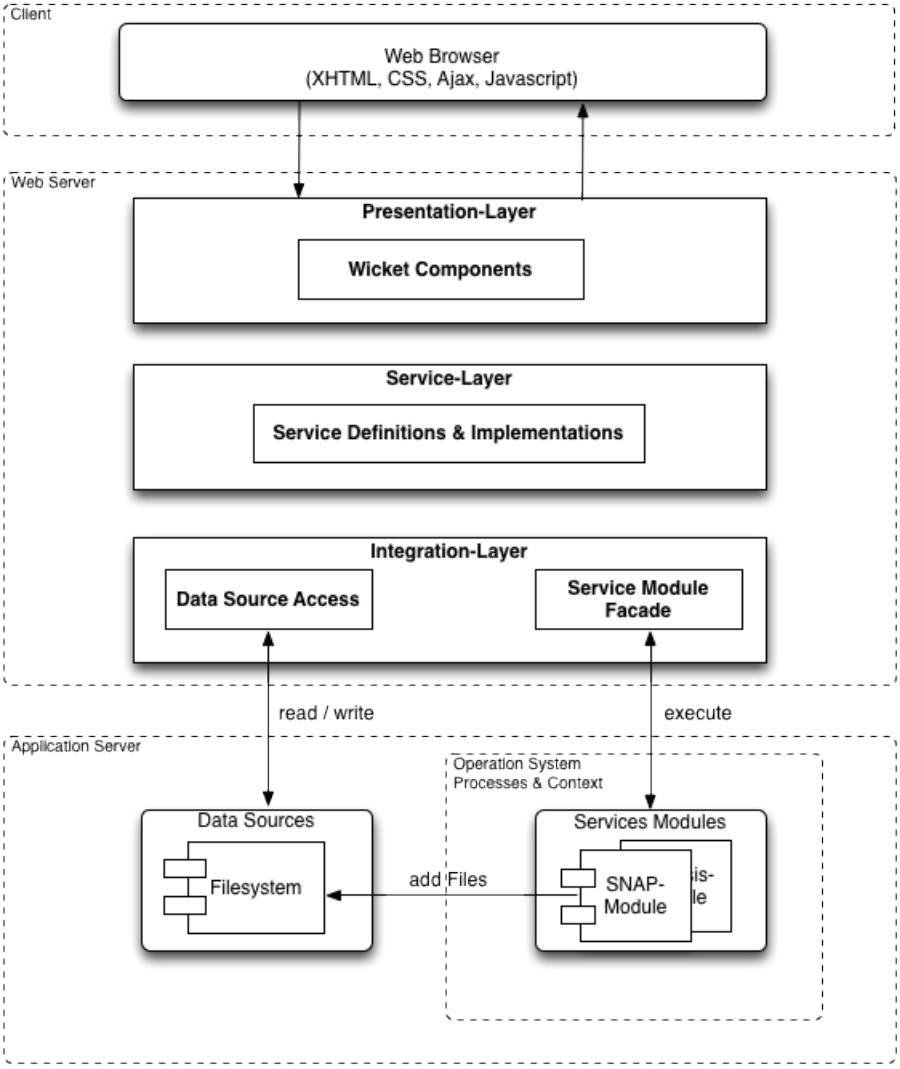

Abbildung 6.2: Darstellung der Architektur

Die Realisierung der Web-Anwendung erfolgt auf Basis einer dreischichtigen Architektur. Abbildung 6.2 zeigt diese Architektur und die Einteilung in die Layer, die nachfolgend im Detail beschrieben werden:

- Presentation-Layer
- Service-Layer
- Integration-Layer

6.1.2.1 Presentation-Layer

Dieser Layer wird durch den Einsatz des Web Framework realisiert. Der Einsatz des Web-Framework Wicket beschränkt sich auf diesen Layer. Der Einsatz von Wicket wurde in Kapitel 5 Abschnitt 5.2.3 als Anforderung festgelegt. Der Presentation-Layer besteht im wesentlichen aus Wicket-Komponenten, mit deren Hilfe die Web-Anwendung umgesetzt wird. In diesem Layer wird keine Geschäfts- und Applikationslogik implementiert, sondern die Wicket-Komponenten greifen auf die einzelnen Dienste des Service-Layer zu.

Folgende Herausforderungen müssen in diesem Layer berücksichtigt werden:

- Der Client muss mit der Web-Anwendung in unterschiedlichen Kommunikationsmodellen (synchron und asynchron) kommunizieren. Neben dem klassischen synchronen Request-Response Muster ist auch der Einsatz von Ajax notwendig.
- Techniken wie Comet-Konzept bzw. Long Polling müssen eingesetzt werden (vgl. Abschnitt 6.3.7). Durch den Einsatz dieser Techniken soll erreicht werden, dass der Server neue Daten an den Client übermitteln kann, ohne das hierfür eine explizite Benutzerinteraktion oder Anfrage durchgeführt werden muss [Wähner, 2011b]. Dadurch soll beispielsweise erreicht werden, dass bei rechenintensiven Berechnungen von Netzwerk Metriken in der Web-Anwendung immer die aktuellen Daten angezeigt werden.
- Benutzern muss eine Session zugeordnet werden.

6.1.2.2 Service-Layer

Der Service-Layer implementiert die notwendige Geschäfts- und Applikationslogik. Für den Service-Layer wurde kein eigenes Framework verwendet. Beispielsweise könnte man das Framework Spring[1] für diesen Layer einsetzen. Die Funktionalität wird in diesen Layer in einzelnen Service-Klassen implementiert, die auf Dienste und Funktionalität des Integration-Layer zurückgreifen und somit eine saubere Schnittstelle darstellen.

6.1.2.3 Integration-Layer

Der Integration-Layer kapselt den Zugriff auf Daten und einzelne Subsysteme, die in die Web-Anwendung integriert werden müssen. Durch geeignete Module (Komponente Data-Source-Access) kann beispielsweise der Zugriff auf das Filesystem durch eine definierte Schnittstelle sichergestellt werden. Die Module ermöglichen den Zugriff auf systemrelevante Daten wie beispielsweise Konfigurationsdateien und benutzerrelevante Daten wie beispielsweise hochgeladene Datensätze und Berechnungsergebnisse. System- und benutzerrelevante Daten sind in einer geeigneten Struktur abgelegt. In diesem Layer sind die Komponenten Benutzer, Netzwerksdatensatz und Netzwerk Metriken berücksichtigt.

Ein weitere Aufgabe des Integration-Layer ist die saubere Einbindung von einzelnen Service-Module in die Web-Anwendung, die für die Berechnung von Netzwerkmetriken verwendet werden. Dieser Layer stellt diebezüglich eine Service-Module-Façade bereit, das den Zugriff auf die einzelnen Service-Module ermöglicht. Die Service Module Façade können durch eine XML-Konfigurationsdatei konfiguriert werden, sodass neue Netzwerkanalyse-Module im System sehr leicht hinzugefügt werden können. Gleichzeitig wird mit dieser Komponente auch ein Mechanismus ermöglicht, rechenintensive Service Module zu starten, ohne dass das System blockiert wird. D.h. die Service Module Façade ermöglicht es, dass mehrere Berechnungen gleichzeitig gestartet werden können, ohne dass sich diese gegenseitig blockieren. In diesem Layer erfolgt der Zugriff auf die Netzwerkanalyse-Modul Komponenten.

6.2 Technologien und verwendete Tools

In Kapitel 5 Abschnitt 5.3 wurde aus den Anforderungen an das Software-Projekt abgeleitet, dass als Web Framework Apache Wicket verwendet werden soll. In Abschnitt 6.2.1 wird das eingesetzte Web Framework im Detail beschrieben. Zusätzlich wurden bei der Umsetzung der Web-Anwendung eine

[1] http://www.springsource.org/

Reihe weiterer Technologien und Werkzeuge verwendet. Die Wichtigsten eingesetzten Technologien und Werkzeuge werden in den folgenden Abschnitten kurz beschrieben.

6.2.1 Apache Wicket

> *“ Wicket bridges the impedance mismatch between the stateless Hypertext Transfer Protocol (HTTP) and stateful Java programming. ”*
>
> [[Dashorst und Hillenius, 2008]]

In diesem Abschnitt wird das Web Framework Apache Wicket, dass für die Umsetzung der Web Anwendung *Network Metric Explorer* eingesetzt wird, im Detail beschrieben. Dieser Abschnitt knüpft an Kapitel 3 Abschnitt 3.5 an, in dem eine Definition von Web Frameworks geliefert wurde. Dieser Abschnitt zeigt, welche Features und Konzepte ein Web Framework im Detail hat.

In Kapitel 5 Abschnitt 5.3 wurde festgehalten, dass die Dokumentation für das Wicket Framework nicht gut ist, es aber entsprechende Fachliteratur gibt, die das Framework exzellent beschreiben. Für die Recherche und den Know-How Aufbau wurden die Bücher von Förther et al. [2009], Mosmann [2009] und Dashorst und Hillenius [2008] verwendet.

6.2.1.1 Einführung in Apache Wicket

Apache Wicket ist ein komponenten-basiertes Web Application Framework, welches ein zustandsbehaftetes Programmiermodell mit Java und HTML ermöglicht und das MVC-Muster implementiert. Das Projekt Apache Wicket wurde 2004 von Jonathan als Open-Source-Web Framework initiiert. Wicket ist heute ein Top-Level-Projekt der Apache Software Foundation und hat zusätzlich eine sehr aktive Community. Apache verfolgt das Leitmotiv, ein Rahmenwerk bereitzustellen, durch das intuitives Vorgehen und hohe Effizienz bei der Erstellung einer Web Anwendung gefördert wird [Förther et al., 2009], [Dashorst und Hillenius, 2008].

Mit Wicket können sowohl klassische Webanwendungen (*Multi-Page-Modell*) als auch moderne Web 2.0 Anwendungen und Rich-Clients (*Single-Page-Modell*) entwickelt werden. Wicket liefert hierzu eine breite Unterstützung von Javascript und Ajax. Wicket hat den Vorteil, dass beide Modelle über das gleiche Programmiermodell abgebildet werden können. Beispielsweise wird die Komplexität von Rich-Clients und der dafür notwendigen Technologien (Ajax, asynchrone Requests usw.) durch Wicket –bzw. in Komponenten– gekapselt. Wicket stellt ein Framework-API bereit, das somit technische Details versteckt und demzufolge bei der Entwicklung nur Java-Kenntnisse benötigt werden [Förther et al., 2009]. Ein Vorteil von Wicket ist, dass dieses Framework einfach, konsistent und intuitiv [Mosmann, 2009] ist. Die Wicket-Architektur verfolgt einen leichtgewichtigen Komponentenansatz und orientiert sich vom Grundprinzip sehr an Swing. In Wicket wird alles in Java entwickelt, was den Vorteil hat, dass ein Fehler auch nur im Code auftreten kann. Der Komponentenansatz und die Analogie zu Swing bringt mit sich, dass das Konzept sehr leicht zu verstehen ist. Java steht im Fokus. Dadurch benötigt man neben der Kenntnis von Java sonst keine speziellen Vorbereitungen. Nachdem sich das Komponentenmodell von Wicket sehr stark an Swing orientiert, ergeben sich die folgenden Vorteile [Förther et al., 2009]:

- Es wird ein praxiserprobtes Komponentenmodell wiederverwendet.
- Entwickler mit Swing Erfahrung wird der Einstieg erleichtert.

Die Wicket Architektur ermöglicht ein zustandsbehaftetes Komponentenmodell mit Standard Webtechnologien, vor allem dem zustandslosen Htttp-Protokoll, abzubilden. Wicket stellt und verwaltet zur

Laufzeit für jeden Benutzer einer Anwendung einen virtuellen Anwendungsspeicher. Der Zustand wird nicht in einer Websession verwaltet, sondern wird durch das Framework in Form von Attributen mit Komponenten- und Seiten-Klassen assoziiert [Mosmann, 2009].

In Wicket werden Seiten und Komponenten durch Java-Klassen abgebildet und entsprechend der Möglichkeiten der Programmiersprache Java, können diese auch miteinander in Beziehung gesetzt werden. Der Einsatz von Java ermöglicht, dass objektorientierte Techniken wie Vererbung und Komposition sinnvoll genutzt werden können. Im Gegensatz zu Java Server Faces ist eine Wicket Anwendung selbst für die Instanziierung von Komponenten verantwortlich. Förther et al. [2009] konstatiert, dass dieses Vorgehen Vorteile aufweist, da man genau bestimmen kann in welcher Phase Komponenten erzeugt werden. Die Komponenten werden durch Konstruktoren der Komponenten-Klassen erzeugt, wodurch zur Compile-Zeit erkannt wird, ob die Komponentenschnittstellen korrekt bedient werden.

Wicket erlaubt, dass bei Komponenten sehr einfach auf Vererbung zugegriffen werden kann. Komponenten haben den Zweck Funktionalität vollständig in sich zu kapseln. Wicket selbst stellt technische Komponenten bereit. Zusätzlich ist es sinnvoll in einem Projekt fachliche Komponenten zu definieren. Komponenten können unabhängig von der Web-Anwendung implementiert werden. Eine Komponente kann man in einer Web-Anwendung uneingeschränkt oft benutzen. Aus einzelnen Komponenten kann man komplexe Anwendungen realisieren. Eigene Komponenten kann man auch in eigene Bibliotheken auslagern. Diese Argumente zeigen, wie Wiederverwendbarkeit in Wicket erreicht werden kann [Förther et al., 2009].

Wicket setzt XHTML-Templates ein, um das Layout und den statischen Inhalt einer Web-Anwendung zu definieren. In einem XHTML-Template befindet sich kein Code oder spezielle Logik. Wie anfangs erwähnt, wird in Wicket jegliche Funktion der Anwendung mit Java implementiert. Durch dieses saubere Trennung entsteht Programmcode nur an einer Stelle und nur in einer Sprache. In einem XHTML-Template können spezielle Markierungspunkte in Form von spezifischen HTML-Attributen (Attribute `wicket:id`) berücksichtigt werden, wodurch Wicket zur Laufzeit Komponenten einfügen kann. Diese XHTML-Templates haben keine UI-Logik. UI-Logik wird im Wicket Framework nur in Java abgebildet. Die XTML-Templates sind trotz der Wicket-Markierungen XHTML, das in jeder Phase der Umsetzung angepasst werden kann. Dadurch kann die Entwicklung von UI-Logik und Design parallelisiert werden [Förther et al., 2009].

Apache Wicket verfolgt das Ziel, neben XHTML-Templates soweit als möglich auf Nicht-Java-Artefakte zu verzichten. Zusammengefasst spielen folgenden Artefakte in Wicket eine Rolle [Förther et al., 2009]:

- web.xml[2]
- XHML-Templates
- CSS-Dateien
- Javascript-Dateien
- Resource-Bundles

In Wicket wird auf Konfigurationsdateien für Anwendungs- und Framework Parametrierung verzichtet. Die Konfiguration von Geschäftsregeln usw. muss natürlich von Wicket unterstützt werden. Wicket verfolgt hier das Convention over Configuration Paradigma. Durch eine Namenskonvention erfolgt eine Verknüpfung von zusammengehörigen Ressourcen, wie folgendermaßen am Beispiel für die Wicket-Seite `WicketPage`:

[2] Die Konfigurationsdatei web.xml ist ein Artefakt der Servlet-Spezifikation.

`WicketPage.java`: Ist eine Java-Klasse, die notwendige Funktionalität implementiert.

`WicketPage.html`: Ist eine Seitenvorlage bzw. ein XHTML-Template und definiert das Layout und statischen Content.

`WicketPage.properties`: In dieser Datei sind Texte und Fehlermeldungen in der *lingua franca* des Anwendungskontext enthalten.

`WicketPage_de.properties`: In dieser Datei sind Texte und Fehlermeldungen in deutscher Sprache enthalten. Durch den Suffix `_de` wird erkannt, dass diese Datei zu verwenden ist, wenn die Sprache der Benutzeroberfläche Deutsch sein soll.

6.2.1.2 Kernelemente einer Wicket-Anwendung

Wicket stellt für die Realisierung einer Web-Anwendung klar definierte Bausteine mit je nach Kontext entsprechender Abstraktion zur Verfügung. Abbildung 6.3 gibt einen Überblick über alle notwendige Elemente und illustriert die Abhängigkeit zu der Servlet-API. Abbildung 6.3 gibt auch einen Überblick über die folgenden Kernelemente einer Wicketanwendung Förther et al. [2009]:

- WebPages
- Components
- XHTML-Template
- Wicket-Models

Die einzelnen Elemente und deren Zusammenspiel werden in den folgenden Abschnitten beschrieben.

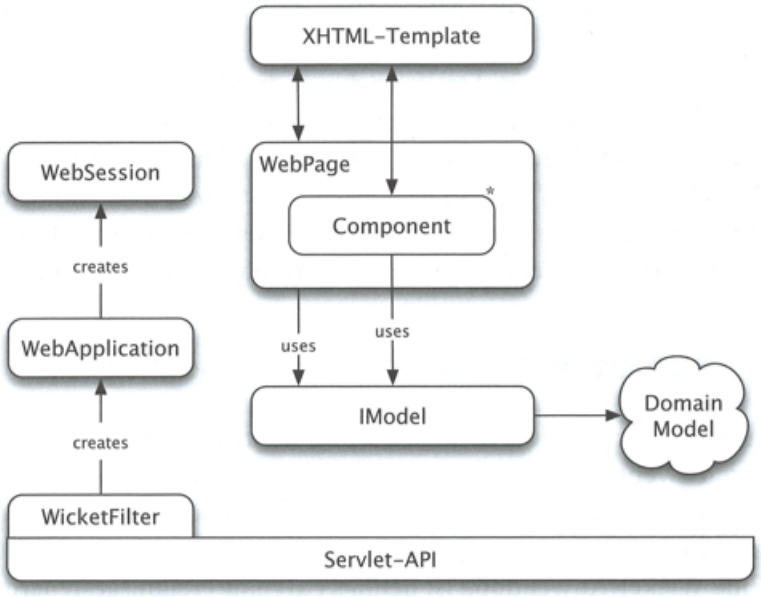

Abbildung 6.3: Kernelemente einer Wicket-Anwendung nach Förther et al. [2009]

6.2.1.3 Struktur einer Wicket-Anwendung

Im folgenden werden die wichtigsten Elemente die in einer Wicket Web-Anwendung zusammenspielen beschrieben [Mosmann, 2009], [Dashorst und Hillenius, 2008]:

WebApplication: Damit man eine eigene Wicket-Anwendung erstellen kann, muss man eine von der Klasse $WebApplication$ abgeleitete Klasse erstellen. Am Server existiert zur Laufzeit nur eine Instanz pro Anwendung. Ein Application Objekt ist der zentrale Akteur der Ablaufsteuerung. Eine Application ist ein Container, der alle Komponenten, Markupdateien etc. bündelt. Durch das Application Objekt kann konfiguriert werden, wie sich die Applikation zur Laufzeit verhält.

Session: In Wicket bekommt ein jeder Benutzer einer Anwendung eine eigene Session. Es ist zu jeden Benutzer typischerweise genau eine Session assoziert. In Wicket wird nicht versucht, den Zustand der Anwendung über eine URL abzubilden, sondern es werden alle relevanten Informationen bzgl. der Nutzerinteraktion in einer Session gespeichert. Mit Wicket ist es möglich eine für die Anwendung spezifische Session zu implementieren.

SessionStore: In einem SessionStore werden die Daten einer Session gespeichert. Typischerweise werden die Daten in einem HttpSession Objekt gespeichert. Der SessionStore hat die Aufgabe, die Browser-Historie in einer Anwendung für einen Benutzer aufzuzeichnen.

PageMap: Eine Session hat mindestens eine PageMap. Eine PageMap hält alle Seiten, die ein Benutzer aufgerufen hat. Dadurch wird in Wicket das Back-Button-Problem gelöst. Ein Benutzer kann im Browser auf ältere Seiten zurückspringen. Wicket kann durch die PageMap den Zustand der Anwendung wieder herstellen.

Page: Eine Page ist eine besondere Komponente. Eine Page kann mehrere Komponenten halten. Ausgezeichnet ist die Page aufgrund der Tatsache, dass es sich um die oberste Komponente im Komponentenbaum handelt. Die Darstellung einer Page liefert als Ergebnis (X)HTML-Repräsentation.

PageStore: Der PageStore ist ein Speicher für Seiten die nicht mehr in der PageMap gehalten werden. In der PageMap werden nur aktuelle Seiten im Speicher gehalten,ältere Seiten werden in der Regel in den PageStore abgelegt und dort serialisiert und gespeichert.

Component: Komponenten sind die Basiseinheiten einer Web-Anwendung.

6.2.1.4 Request-Behandlung

In Abschnitt 6.2.1.3 wurde dargelegt, dass die Elemente Application und Session wichtige Elemente in der Verarbeitung von Requests sind. Eine Application hat mehrere Sessions zu managen. Jede Session ist mit einem Benutzer assoziiert und jeder Benutzer kann mehrere Requests absetzen. Wicket kapselt bei der Reques-Behandlung $HttpRequest$ und $HttpResponse$ Klassen der Servlet-Spezifikation und stellt entsprechende Klassen bereit, die technische Aspekte verstecken. Im folgenden werden die wichtigsten Elemente die in einer Wicket für die Request-Behandlung verantwortlich sind [Mosmann, 2009], [Dashorst und Hillenius, 2008], beschrieben:

Request: Kapselt eine Benutzeranfrage. Beinhaltet eine URL und Parameter.

Response: Kapselt das Ergebnis der Benutzeranfrage.

RequestCycle: Requests und Responses werden durch einen RequestCylce abgearbeitet. Die Bearbeitung von einzelnen Phasen wird an den RequestCycleProcessor delegiert. Ein Request wird in einem einzigen RequestCycle abgearbeitet. Eine weiter Aufgabe des RequestCycle ist, eine Referenz zu einem RequestTarget zu halten.

RequestCycleProcessor: Dieses Klasse implementiert alle Verarbeitungsschritte eines Requests. Eine Instanz dieser Klassen wird für alle Requests gemeinsam genutzt.

RequestTarget: Der RequestTarget entscheidet, in welcher Form ein Request erzeugt wird.

6.2.1.5 Komponenten, Modelle, Markup und das MVC-Pattern

Apache Wicket ist ein Model-View-Controller (MVC) basiertes Framework. In Wicket wurde das MVC-Muster mit einer *pull-basierten* Architektur umgesetzt. Man spricht in diesem Kontext auch von einer komponenten-basierten MVC-Architektur. In Abbildung 6.4 ist das umgesetzte MVC-Muster anhand einer Benutzeranfrage dargestellt und zeigt alle beteiligten Komponenten. Die einzelnen Elemente und ihre Aufgaben sowie das resultierende Zusammenspiel, werden im folgenden beschrieben [Förther et al., 2009], [Mosmann, 2009]:

Komponenten: Eine Komponente ist die Basiseinheit einer Web-Anwendung. Eine Komponente übernimmt im Kontext des MVC-Patterns die Aufgabe des *Controllers* und übernimmt auch Teile des *View*.

Wicket trägt die Verantwortung, dass eine Aktion eines Benutzers zu der richtigen Komponente zugeordnet wird. Wicket kapselt diese Verarbeitung, sodass man sich als Anwender um diesen Aspekt nicht kümmern muss. In einer Komponente übernehmen sogenannte Event-Handler-Methoden (beispielsweise $onSubmit()$) die Verantwortung über die folgenden Bereiche:

- Verarbeitung von Benutzereingaben
- Aktualisierung des Modells
- View über Änderungen Informieren

Modelle: Modelle dienen dazu, um Komponenten mit Anwendungsdaten in Verbindung zu bringen. Für Modelle stehen ein $IModel$-Delegate-Interface zur Verfügung, wodurch ein standardisierter Zugriff auf Klassen der Anwendungsdomäne erfolgt.

Markup: Das Markup für eine Browser wird aus dem XHTML-Template und den zugehörigen Seite und ihren Komponenten erzeugt. Eine jede Komponente kann ihr Markup durch eine standardisierte Funktion ($renderComponent()$) selber erstellen. Eine Komponente kann statische Daten als auch Daten des Modells durch einen lesenden Zugriff berücksichtigen.

6.2.1.6 Strikte Trennung von Design und Logik

Wicket verfolgt das Paradigma Logik in Java und Design über XHTML-Templates. Dadurch wird eine strikte Trennung von Design und Logik ermöglicht. In den Abschnitten zuvor wurde erwähnt, dass Seiten und Komponenten Logik kapseln. In diesen Komponentklassen kann Anwendungslogik und User-Infterface (UI) Logik abgebildet werden [Förther et al., 2009].

Durch die Verwendung von XHTML-Templates wird das Design von Seiten und Komponenten ermöglicht. XHTML-Templates definieren das Layout und statischen Inhalt. XHTML-Templates basieren auf dem XHTML[3]-Standard.

Die Verbindung zwischen Template und Java-Klasse erfolgt wie in Abschnitt 6.2.1.1 einerseits über einen Convention over Configuration Ansatz aufgrund einer Namenskonvention und andererseits durch

[3]XHTML: `http://www.w3.org/TR/xhtml1/`

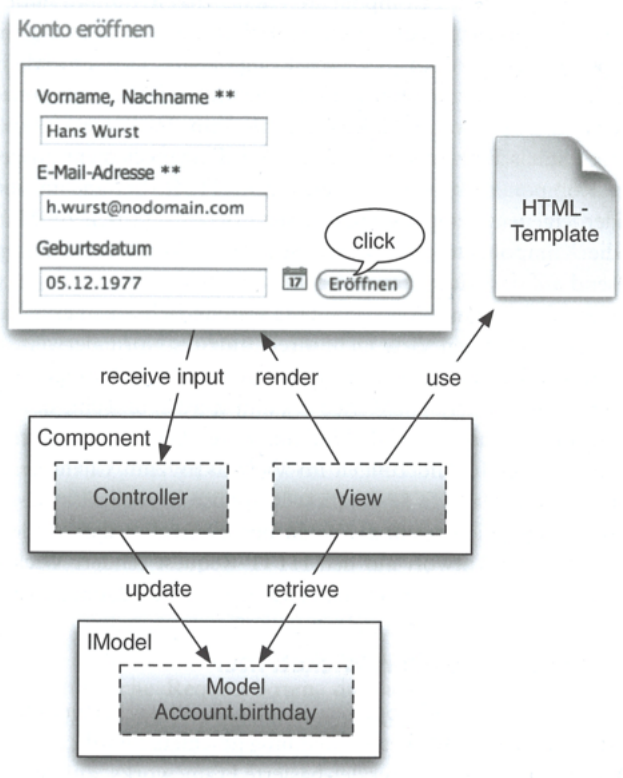

Abbildung 6.4: Erklärung des MVC-Patterns in Apache Wicket anhand einer Benutzeranfrage nach Förther et al. [2009]

die Verwendung und Vergabe von Wicket-IDs. Wicket-IDs werden in einem XHTML-Template als Attribute `wicket:id` zu Elementen hinzugefügt. In Java werden die Wicket-IDs bei der Erzeugung von Komponenten im Konstruktor übergeben. Dadurch ist es möglich eine Verbindung herzustellen. Abbildung 6.5 illustriert diesen Zusammenanhang anhand eines Beispiels.

6.2.2 Apache Tomcat

In diesem Projekt wird Apache Tomcat[4] als Servlet-Container eingesetzt. Die Web-Anwendung *Network Metric Explorer* wird mit Apache Wicket entwickelt. Eine Web-Anwendung die mit dem Apache Wicket Web Framework entwickelt wurde, stellt eine zur Servlet-API 2.3 konforme Web-Anwendung dar [Förther et al., 2009]. Das Web Framework Apache Wicket stellt sonst keine expliziten Anforderungen an einen Web-Server. Es ist daher jeder Standard-konformer Servlet-Container, der diese Servlet-Spezifikation unterstützt, ausreichend. Alternativ zu Apache Tomcat –ab der Version 4.0.0– könnten auch die folgenden Servlet-Container verwendet werden:

- Glassfish[5]
- Jetty[6]

Die erstellte Web-Anwendung *Network Metric Explorer* kann auf einen Apache Tomcat Server sehr leicht deployed bzw. installiert werden. Die Web-Anwendung muss diesbezüglich nur als ein Web App-

[4] Apache Tomcat: `http://tomcat.apache.org/`
[5] Glassfish: `http://glassfish.java.net/`
[6] Jetty: `http://jetty.codehaus.org/jetty/`

Abbildung 6.5: Zusammenhang zwischen XHTML-Template und Java Komponenten [Dashorst und Hillenius, 2008]

lication Archive (WAR)[7] bereitgestellt werden.

6.2.3 Java

Die Web-Anwendung *Network Metric Explorer* sowie alle zusätzlichen Komponenten und Module wurden mit Java entwickelt. Eine Ausnahme bilden die Netzwerkanalyse-Module der SNAP-Library. Diese Module werden als externe Services in die Web-Anwendung integriert und über eine geeignete Abstraktion durch das Façade-Pattern (vgl. Gamma et al. [1995]) und der Verwendung der ProcessBuilder[8]-API in das System integriert und können dadurch mit Java-Objekten gesteuert werden. Die Klasse ProcessBuilder gibt es erst seit Java 5. Die Anwendung wurde mit dem Java Development Kit (JDK) ab Version 1.5 umgesetzt.

6.2.4 Java Architecture for XML Binding (JAXB)

Java Architecture for XML Binding (JAXB)[9] ist ein Java-API, dass in Anwendungen verwendet werden kann, um sehr einfach XML-Dokumente zu verabeiten und darauf zugreifen zu können, ohne das explizites Wissen über XML notwendig ist [Ort und Mehta, 2003]. JAXP ermöglicht durch einen sogenannten Binding Compiler ein XML-Schema (XSD) an Java-Klassen zu binden. Ein Binding Compiler führt eine Abbildung eines XML-Schema auf einzelne Java-Klassen (inkl. Interfaces und Enums) durch. Dabei werden auch die im XSD definierten Datentypen auf Java-Datentypen abgebildet [Ort und Mehta, 2003]. In Abbildung 6.6 ist das Prinzip von JAXB und der Abbildung eines XML-Shema durch einen Binding-Compiler auf Java-Klassen dargestellt.

In einer Anwendung kann nun mit den gebundenen Java-Klassen in Verbindung mit der JAXB-API gearbeitet werden. Es wird eine Objekt-zu-XML-Transformation unterstützt, dieses wird in der Literatur oft als Marshalling bezeichnet. In diesem Fall kann ein XML-Dokument anhand von Java-Objekten (Instanzen der zuvor erstellten Java Klassen) generiert werden. Durch Unmarshalling, eine XML-zu-Objekt-Transformation, wird in diesem Fall ein XML-Dokument in eine Menge von Java-Objekte umgewandelt [Ort und Mehta, 2003]. In Abbildung 6.6 ist der Vorgang von Marshalling und Unmarshalling darge-

[7] Web Application Archive (WAR): `http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/WCC3.html`

[8] ProcessBuilder: `http://download.oracle.com/javase/1.5.0/docs/api/java/lang/ProcessBuilder.html`

[9] Java Architecture for XML Binding (JAXB): `http://jaxb.java.net/`

stellt. In Abschnitt 6.3.4 und 6.3.5 wird gezeigt, dass für die Komponenten Benutzer, Netzerkanalyse-

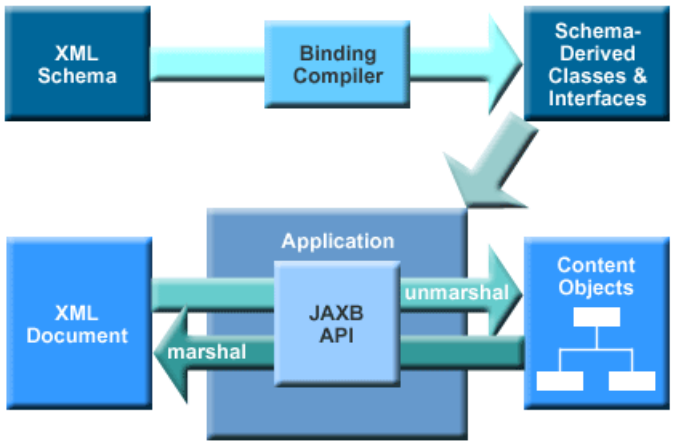

Abbildung 6.6: Darstellung der Konzepte Binding, Marschal und Unmarschal von Java Architecture for XML Binding (JAXB) [Ort und Mehta, 2003]

Module und Netzwerk-Metriken eigene XML-Schemas (XSD) entwickelt und definiert werden. Diese XSD-Dateien definieren die Struktur von XML-Dokumenten. Die einzelnen XML-Dokumente werden wiederum in der Web-Anwendung verwendet um,

- einzelne Netzwerwerkanalyse-Module zu beschreiben. Diese aussagekräftige Beschreibung wird zur Laufzeit eingelesen und dient als Interface-Beschreibung, damit man mit einer generischen Architektur ein Netzwerkanalyse-Modul aufrufen und mit den richtigen Parametern versorgen kann. Relevante Daten wie Bezeichnung und Beschreibung werden in der Benutzerschnittstelle angezeigt.
- erstellte Netzwerk-Metriken an einem spezifischen Ort im Filesystem ablegen zu können. Diesbezüglich wurden ein eigener XML-Dialekt entwickelt, um zusätzliche Metadaten wie Netzwerkanalyse-Modul, Netzwerkdatensatz und Status der Berechnung zu hinterlegen.
- Benutzer in einer eigenen XML-Struktur ablegen zu können.

Daraus lässt sich ableiten, das XML-Dateien im System gelesen, erstellt und bearbeitet werden müssen. Aus diesem Grund wurde JAXB, vor allem aufgrund seiner einfachen Anwendung, für die Verarbeitung der XML-Dateien verwendet.

6.2.5 Apache Maven (MVN)

Apache Maven[10] ist ein umfangreiches Projekt-Management-Tool bzw. Build-Management-Tool und basiert auf Java. Apache Maven wird oft als Alternative zu dem Build-Werkzeug Apache Ant[11] gesehen. Im Vergleich zu Ant bietet Maven jedoch nicht nur das Ziel einer automatisierten Erzeugung einer Anwendungen aus den Quelltexten, sondern unterstützt auch konsequent ein flächendeckendes Software-Lebenszyklus-Management [Massol und van Zyl, 2006]. Maven bildet einen Standard-Lebenszyklus eines Sotwareprojektes nach dem Software Paradigma *Convention over Configuration* ab. Dieser Lebenszyklus unterteilt ein Projekt in mehrere Phasen. Maven unterstützt jede Phase so, dass möglichst viele Schritte automatisiert werden können. Einzelne Phasen sind beispielsweise Scaffolding, Validieren, Kompilieren, Testen usw. Maven schreibt nicht vor, dass alle Phasen in einem Software-Projekt

[10] Apache Maven: `http://maven.apache.org/`
[11] Apache Ant: `http://ant.apache.org/`

durchlaufen werden müssen [Massol und van Zyl, 2006]. In einer XML Konfigurationsdatei ($pom.xml$) können einzelne Phasen konfiguriert und koordiniert werden.

Maven bringt in diesem Projekt eine gute Unterstützung, vor allem in Verbindung mit Apache Wicket mit. Das Apache Wicket Framework wird mit Maven gebaut, zusätzlich existiert ein Maven-Archetyp (für die Phase Scaffolding) für Wicket, mit dem ein Grundgerüst für ein Wicket-Projekt erzeugt werden kann Förther et al. [2009].

6.2.6 Stanford Network Analysis Package (SNAP)

Die SNAP-Library ist eine Netzwerk Analyse Library und wurde in C++ geschrieben. Die SNAP-Library stellt neben einer C++-API auch Beispielanwendungen bereit, die kompiliert und als Konsolen-Programme verwendet werden können. Die SNAP-Library wurde näher in Kapitel 2 Abschnitt 2.6 beschrieben. Die SNAP-Library ist Grundlage für die Netzwerkanalyse-Module. Netzwerkanalyse-Module sind im Rahmen dieser Masterarbeit fertige Komponenten die wiederverwendet werden, und mit deren Hilfe komplexe Netzwerkanalysen durchgeführt werden können. In Kapitel 5 wurde festgelegt, welche Beispielanwendungen aus der SNAP-Library als Netzwerkanalyse-Module in der Web-Anwendung Berücksichtigung finden sollen.

6.3 Implementierung

Dieser Abschnitt liefert einen Überblick über ausgewählte Details der Umsetzung der Web-Anwendung.

6.3.1 Konfiguration der Entwicklungsumgebung

In Kapitel 5 Abschnitt 5.3 wurde festgelegt, dass die Web-Anwendung mit dem Web Framework Apache Wicket umgesetzt werden soll. Apache Wicket ist Java basiert und setzt den Java Developement Kit (JDK) in der Version 1.5 voraus (vgl. Abschnit 6.2.3). Aus diesem Grund wurde der JDK auf dem Entwicklungssystem installiert. Eine Web-Anwendung die mit Wicket implementiert wurde, stellt eine zur Servlet-API 2.3 konforme Web-Anwendung dar und benötigt einen Servlet-Container. Deshalb wurde der Apache Tomcat-Servlet Container mit der Version 6 installiert. Ab der Version 4 ist Tomcat mit der Servlet-API 2.3 konform. Der Servlet-Container Apache Tomcat wurde in Abschnitt 6.2.2 beschrieben.

Ein weiterer Schritt war die Installation der SNAP-Library, die in der Version 2011-03-06 verwendet und in Abschnitt 6.2.6 beschrieben wurde. Das Ziel war es die Beispielanwendungen $NetStat$ und $Centrality$, die in Form von C++ Sourcecode bereitgestellt sind, zu kompilieren. Als Entwicklungssystem wurde ein Unix-basiertes System eingesetzt. Damit man auf diesem System C++ Sourcen kompilieren kann, wird die GNU Compiler Collection (GCC) [12] vorausgesetzt und musste installiert werden. GCC umfasst das Komando g++ und ist ein C++ Compiler. Durch Makefiles der SNAP-Library konnten alle Beispielanwendungen kompiliert werden und standen im Rahmen der Umsetzung als Konsolenprogramme zur Verfügung. Die SNAP-Library setzt voraus, dass GNUPlot am System verfügbar ist. GNUPlot[13] wird für die Erstellung von Diagrammen die mit dem PNG Format generiert werden, von der SNAP-Library benötigt. GNUPlot musste zusätzlich auf der Entwicklungsmaschine installiert werden.

Ein weiterer Schritt war entsprechende *Testdaten* aufzubauen bzw. zu beschaffen. Damit der Funktionsumfang der SNAP-Library genutzt werden kann, wurden Netzwerkdatensätze benötigt. In weiterer Folge werden auch Netzwerkdatensätze für die Web-Anwendung benötigt, damit die Funktionalität der

[12] GCC: http://gcc.gnu.org/
[13] GNUPlot: http://www.gnuplot.info/

Netzwerkanalyse-Module in Anspruch genommen werden kann. Geeignete Netzwerkdatensätze wurden nach einer Recherche direkt von SNAP [2011c] bezogen.

Für die Umsetzung der Web Anwendung wurde Eclipse in diesem Projekt als integrierte Entwicklungsumgebung (IDE) ausgewählt und installiert. Alternativ wäre es auch möglich, Netbeans oder JDeveloper als IDE zu verwenden. Zusätzlich wurde Maven (vgl. Abschnitt 6.2.5) installiert und es wurde für die Eclipse IDE das Eclipse-Plugin *m2e*[14] installiert. Das Eclipse-Plugin Wicket Bench[15] wurde unterstützend für das Web Framework Wicket berücksichtigt. Ebenso wurden ein Eclipse-Plugin für JAXB[16] –JAXB wurde in Abschnitt 6.2.4 beschrieben– installiert. In Eclipse wurde der Apache Tomcat Server eingebunden. Dadurch war es möglich, dass während der Entwicklung die Web Anwendung auf einem Tomcat Server automatisch deployed und der Server-Output usw. direkt in Eclipse eingesehen werden konnte.

Damit mit der Umsetzung der Web-Anwendung begonnen werden konnte, wurde ganz einfach per Maven ein Wicket Archetyp-Projekt generiert. Listing 6.1 zeigt wie Maven dafür verwendet werden kann. Dadurch wurde ein Template für ein Wicket-Projekt samt geeigneter Projektstruktur und mit allen notwendigen Dateien und einer kleinen Testanwendung ($web.xml$, Applikationsklasse, etc) generiert, die sofort auf dem Apache Tomcat Server gestartet werden konnte. Darauf aufbauend konnte mit der Entwicklung begonnen werden.Das Programm Altova XMLSpy[17] wurde für die Entwicklung der XML-

```
1 mvn archetype:create
2   -DarchetypeGroupId=org.apache.wicket
3   -DarchetypeArtifactId=wicket-archetype-quickstart
4   -DarchetypeVersion=1.4.19
5   -DgroupId=edu.masterthesis.networkanalysis
6   -DartifactId=NetworkMetricExplorer
7   -DarchetypeRepository=https://repository.apache.org/
8   -DinteractiveMode=false
```

Listing 6.1: Erstellung eines Template für ein Wicket Projekt mit Maven [SNAP, 2011c].

Schema Dateien, die in Abschnitt 6.3.5 beschrieben sind, verwendet.

6.3.2 Komponenten der Web Anwendung

In Abbildung 6.7 sind die Wicket-Komponenten der Web-Anwendung dargestellt. Diese Komponenten sind dem Presentation-Layer zugeordnet. In diesem Abschnitt werden die wichtigsten Komponenten erklärt und kurz beschreiben. Die WebPage- und Panel-Komponenten sind interessant, da diese der Benutzerschnittstelle (vgl. Abschnitt 6.3.3) zugeordnet sind und alle technischen und logisches Details kapseln. Die folgende Beschreibung gruppiert alle wichtige Komponenten um die folgenden Basisklassen (vgl. Abbildung 6.7):

WebApplication: Die Klasse $NetworkAnalysisWebApplication$ ist die Anwendungsklasse der Web-Anwendung. In dieser Klasse wird die verwendete Session ($NetworkAnalysisSession$) zugeordnet und eine Security-Strategie konfiguriert, die mit der WebPage $AuthenticatedWebPage$ assoziiert ist. Dadurch wird von der Anwendung sichergestellt, dass alle von dieser abgeleiteten Web-Seiten nur authentifiziert verwendet werden dürfen. Wicket unterstützt auch einen Developement-Modus. Dieser kann auch in dieser Anwendungsklasse konfiguriert werden. Diese Klasse muss im

[14] mse: http://eclipse.org/m2e/
[15] Wicket Bench: www.laughingpanda.org/mediawiki/index.php/Wicket_Bench
[16] JAXB Plugin: http://sourceforge.net/projects/jaxb-builder/
[17] Altova XMLSpy: http://www.altova.com/de/xmlspy.html

Deployment Descriptor $web.xml$ eingebunden werden. Zur Laufzeit wird sichergestellt, dass es nur eine Instanz von dieser Klasse gibt.

Session: Für die Anwendung wurde eine eigene Session implementiert. Wicket kapselt und verwaltet die $HttpSession$ in eigenen Komponenten. Für dieses Projekt wurde eine $AuthenticatedWebSession$ verwendet, die von der Wicket-API bereitgestellt wird und durch eine spezifische Ableitung an die Anforderungen des Projektes angepasst wurde. Die Klasse $NetworkAnalysisSession$ hat die Methode $authenticate(...)$. In dieser Methode wird die Benutzername-Password Authentifizierung implementiert. Das Session-Objekt hält zur Laufzeit spezifische Benutzerinformationen.

WebPage und Panel: WebPage und Panel sind gruppierende Komponenten einer Web-Anwendung. Diese Komponenten ermöglichen, vor allem die Komplexität die mit der Benutzerschnittstelle verbunden ist, einfach durch diese Komponenten zu verstecken. Zusätzlich lässt sich damit die Web-Anwendung in einzelne Bereiche gruppieren. Die $AbstractBasePage$ ist die Basisseite für dieses Projekt. Diese Klasse repräsentiert das Grundgerüst einer XHTML Seite und enthält im Body-Element lediglich das Element `<wicket:child/>`, wodurch gekennzeichnet ist, dass jede von dieser Klasse abgeleitete Klasse diesen Bereich erweitert. Die $AuthenticatedWebPage$ definiert für angemeldete Benutzer den Header-Bereich der Anwendung.

Eine konkrete Ableitung der Klasse $AuthenticatedWebPage$ ist $NetworkMetricExplorerPage$. Diese Klasse definiert die Web-Anwendung in einem Single-Page-Model und bindet Komponenten für die Navigation (Breadcrump), Auswahl von Netzwerkanalyse-Modulen und Darstellung von Netzwerk-Metriken in Form von Panel-Komponenten ein.

Eine jede WebPage die von $AuthenticatedWebPage$ abgeleitet ist, muss im XHTML-Template im Markup das Element `<wicket:extend>` an oberster Stelle haben. Dieser Teil wird zur Laufzeit von Wicket an die Stelle des Element `<wicket:child>` des XHMTL-Templates der Basisklasse eingefügt. Durch die Elemente `<wicket:child>` und `<wicket:extend>` kann abgebildet werden, dass ein Bereich eines XHTML-Templates durch ein anderes Template erweitert werden kann und umgekehrt das bestehende Bereiche einfach wiederverwendet und nur an einer Stelle definiert werden können. Somit ist es möglich, dass nicht für jede Seite das gesamte Layout einer Seite immer neu definiert werden muss, sondern durch das Konzept der Ableitung kann in Java definiert werden, dass bestimmte Bereiche, wie beispielsweise der Header einer Anwendung, wiederverwendet werden können und somit wiederkehrende Strukturen nur an einer Stelle definiert und gewartet werden müssen.

Die $NetworkMetricExplorerPage$ und das zugehörige XHTML-Template definiert die Struktur und das Layout der Anwendung und hat auch die Aufgabe, die einzelnen zugeordneten Komponenten zu verwalten. Eine weiter Kernaufgabe dieser Page ist die Darstellung und die Auswahl von Netzwerkdatensätzen.

Die $NetworkMetricExplorerPage$-Komponente bindet neben Panel-Komponenten auch komplette Seiten als *modale Fenster* ein. Die Page $NetworkDatasetMultiUpload$ wird als modales Fenster eingebunden und ermöglicht den Upload von Netzwerkdatensätzen im Kontext der Web-Anwendung. Die Folgenden eingebundenen Panel-Komponenten definieren selbst ihr Layout und kapseln jeweils UI-Logik:

- $NavigationPanel$ hat die Aufgabe in Form von *Breadcrump-Navigation* die aktuelle Navigation in der Web-Anwendung darzustellen.
- $NetworkAnalysisModulesPanel$ beinhaltet die Darstellung aller im System eingebundenen Netzwerkanalyse-Module. Einzelne Module können, sofern ein Netzwerdatensatz ausgewählt wurde, angewandt werden.

- $NetworkMetricsPanel$ beinhaltet die Darstellung von berechneten Netzwerk-Metriken. Erstellte Diagramme werden entsprechend skaliert. Die Page $NetworkMetricDetailView$ wird als modales Fenster in diesem Panel eingebunden und für die Darstellung von PNG-Diagrammen von Netzwerk-Metriken in originaler Auflösung verwendet.

ListView: Diese Komponenten werden dafür verwendet, um Daten die in Form einer Liste vorliegen, abzubilden. Die einzelnen Komponenten haben die folgenden Aufgaben und greifen dabei auch auf Dienste des Service-Layer zu:

- Die Komponente $NetworkDatasetListView$ hat die Aufgabe alle Netzwerkdatensätze mit den zugehörigen Metadaten im Kontext eines Benutzers zu ermitteln und darzustellen. Diese Komponente wird von der $NetworkMetricExplorerPage$ verwendet. Die UI-Logik bzgl. der Auswahlmöglichkeit löschen und download von einem einzelnen Datensatz wird von dieser Komponente implementiert.
- Die Komponente $NetworkMetricPlotListView$ hat die Aufgabe PNG-Diagramme von Netzwerk-Metriken darzustellen und wird von der Komponente $NetworkMetricsPanel$ verwendet.
- Die Komponente $NetworkMetricFileListView$ hat die Aufgabe alle Dateien von erstellten Netzwek-Metriken aufzulisten und als Download anzubieten. Diese Komponente wird von der Komponente $NetworkMetricsPanel$ verwendet.

Im folgenden Abschnitt wird die erarbeitete Web-Anwendung anhand der Benutzerschnittstelle vorgestellt. Es wird auch versucht, zuvor beschrieben Komponenten mit den zugehörigen Bereichen in der Benutzerschnittstelle zu verbinden.

6.3.3 Benutzerschnittstelle

Ein wichtiger Aspekt dieser Masterarbeit war die Konzeption der Benutzerschnittstelle der Web-Anwendung. In diesem Abschnitt werden alle relevanten Aspekte die bei der Umsetzung berücksichtigt wurden, beschrieben. Ein weiteres Ziel dieses Abschnitt ist die Illustration der umgesetzte Web-Anwendung durch Screenshots und aussagekräftigen Erklärungen.

6.3.3.1 Design der Benutzerschnittstelle

Das Design der Benutzerschnittstelle wurde in Gesprächen zuerst anhand von einfachen Handskizzen und später durch Mock-ups konzipiert. Parallel wurde ein erster Prototyp mit dem Wicket-Framework entwickelt, mit dem Ziel erarbeitetes Know-how über das Framework Wicket praktisch umzusetzen und zu validieren (Learning by Doing). Es sollte durch den Prototypen gleichzeitig ein Gefühl dafür entwickeln werden, wie man technisch die Benutzerschnittstelle umsetzen kann. Wicket bietet die sehr mächtige Möglichkeit von XHTML-Templates. Diese Templates können direkt in einem Web Browser angezeigt werden. Zu einem späteren Zeitpunkt wurde nur mehr mit den XHTML-Templates gearbeitet, wodurch das Layout unabhängig von der UI-Logik entwickelt werden konnte und das Resultat in Form von einer statischen Web Seite mit einem Web Browser betrachtet werden konnte. Durch den Einsatz und die Verwendung von Cascading Style Sheets (CSS) und durch die Anwendung eines DIV-basiertes Layouts wurde die resultierenden Benutzerschnittstelle erfolgreich designed.

6.3.3.2 Trennung Darstellung und Logik

Für jede der in Abschnitt 6.3.2 vorgestellte Klassen mit den Basistyp $WebPage$ und $Panel$ existieren assoziierte XHTML-Templates. In den Java-Klassen wurde die notwendige UI-Logik implementiert und in

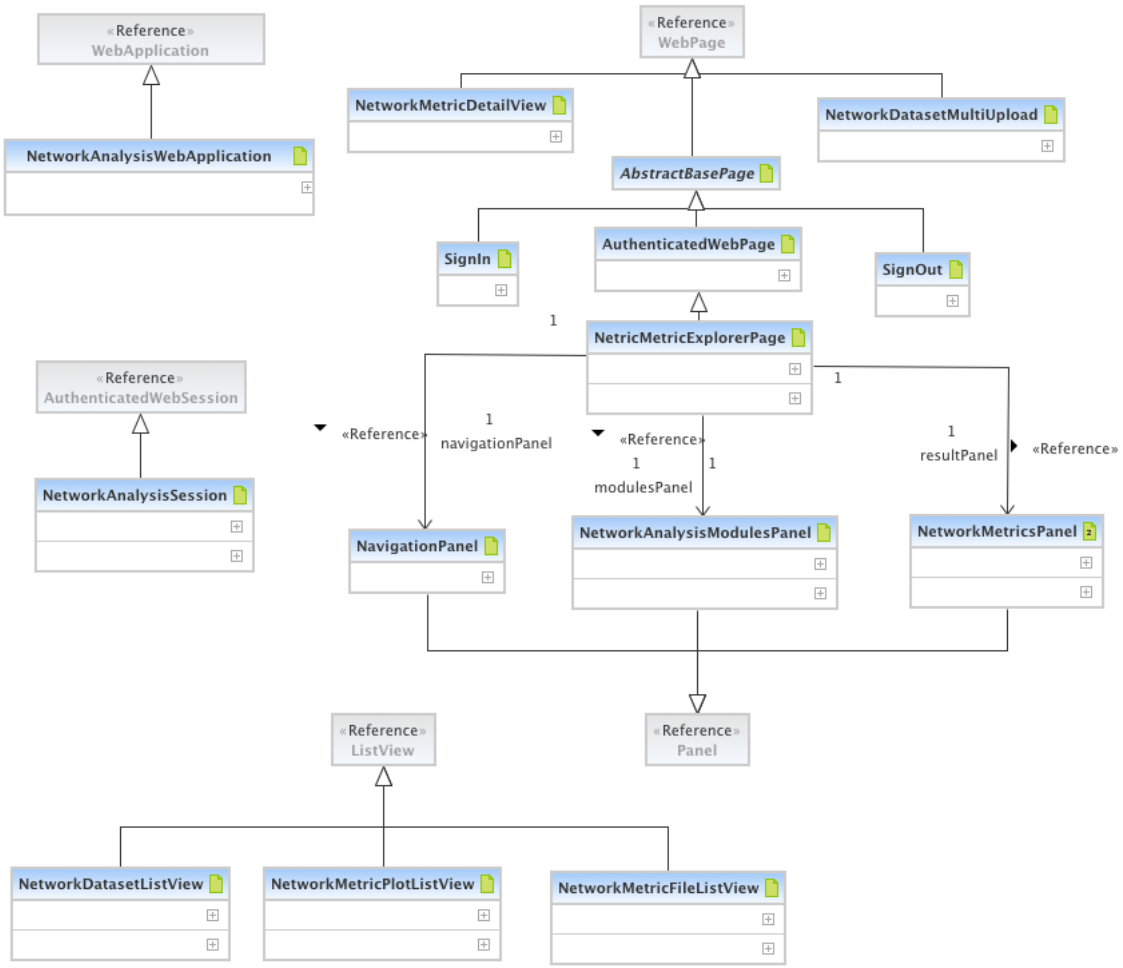

Abbildung 6.7: UML Klassendiagramm der Komponenten der Web Anwendung

den XHTML-Templates –und per CSS– Struktur, Layout und der statische Inhalt der Web-Anwendung definiert. Es wurde das Ziel, Trennung von Darstellung und Logik durch den Einsatz des Web Frameworks Wicket (wie in Abschnitt 6.2.1 beschrieben) unter der Berücksichtigung der bereitgestellten Wicket-Paradigma sehr konsequent umgesetzt. Für eine jede Klasse, die UI-Logik und ein XHTML-Template definiert hat, wurden auch alle Interface-Texte in eigene Properties-Dateien ausgelagert. Durch das Element `<wicket:message>` und mit dem Attribut `key` können Interfacetexte, Fehlermeldungen etc. aus einer Proberty-Datei referenziert und sprachunabhängig in das Layout eingebunden werden. In Abschnitt 6.2.1 wurde erklärt, wie man mit einer einfache Namenskonvention von Properties-Dateien und den Einsatz von dem Element `<wicket:message>` internationalisierte Web-Anwendungen entwickeln kann. Die Umsetzung der Web-Anwendung *Network Metric Explorer* unterstützt dieses Konzept und ist damit für Mehrsprachigkeit (Lokalisierung) in der Tat vorbereitet.

6.3.3.3 Implementierung der UI-Logik durch Komponenten

In Abschnitt 6.3.2 wurden die wesentlich Wicket-Komponenten beschrieben. In Abbildung 6.7 sind die einzelnen WebPage- und Panel-Komponenten durch ein UML-Klassendiagramm dargestellt. Dieser Abschnitt liefert ein Mapping zwischen den zuvor beschrieben Wicket-Komponenten und den einzelnen Bereichen der resultierenden Benutzerschnittstelle der Web-Anwendung.

In Abbildung 6.20 wird das Basisfenster der Web-Anwendung *Network Metric Explorer* dargestellt. Die-

ses ist in fünf Bereiche gegliedert. Diese Ansicht ist nur für einen authentifizierten Benutzern einsehbar. Diese Webseite wurde durch die Komponente $NetworkMetricExplorerPage$, die von $AuthenticatedWebPage$ abgeleitet wurde, implementiert.

Bereich 1: Dieser Bereich entspricht dem Header der Anwendung. Dieser Bereich wurde durch die Ableitung von der Komponente $AuthenticatedWebPage$ geerbt. In diesem Bereich wird neben dem Titel der Anwendung (*Network Metric Explorer*) auch der Benutzernamen angezeigt und es wird die Möglichkeit geboten, dass man sich vom System, durch den Link *"Sign Out"*, abmeldet.

Bereich 2: Dieser Bereich wird durch die Panel-Komponenten $NavigationPanel$, die der WebPage $NetworkMetricExplorerPage$ zugeordnet ist, implementiert. Die primäre Aufgabe dieses Bereiches ist, dass Benutzer Informationen über die Auswahl von Netzwerkdatensätzen und ggfs. des Netzwerkanalyse-Moduls geboten wird. Dieser Bereich wird durch eine *Breadcrump-Navigation* repräsentiert und stellt den aktuellen Navigationskontext dar.

Bereich 3: Dieser Bereich ist der Kernbereich der $NetworkMetricExplorerPage$. In diesem Bereich werden alle Netzwerkdatensätze eines Benutzers angezeigt. Zu jedem Netzwerkdatensatz werden verfügbare Metadaten, wie Name, Bezeichnung, Typ, Anzahl Knoten und Kanten angezeigt. Durch den Link *"Add a new networkdataset"* kann ein modales Fenster geöffnet werden, mit diesen man mehrere Netzwerkdatensätze in das System hochladen kann. Hochgeladene Netzwerkdatensätze können gelöscht werden und stehen auch als Download bereit. Das Ziel dieses Bereiches ist neben der Darstellung und dem Upload von Netzwerkdatensätzen vor allem, dass man für eine weitere Verarbeitung *einfach* einen Netzwerkdatensatz auswählen kann. Ein ausgewählter Datensatz wird in Bereich 2, in der Breadcrump-Navigation, berücksichtigt.

Bereich 4: Dieser Bereich wird durch die Komponente $NetworkAnalysisModulesPanel$ gekapselt und ist der Komponente $NetworkMetricExplorerPage$ zugeordnet. In diesem Bereich kann man, sofern in Bereich 3 ein Netzwerkdatensatz ausgewählt wurde, eine Netzwerkanalyse-Methode auswählen, wodurch Netzwerk-Metriken berechnet (Link *"Generate Metric"*) oder bereits berechnete aufgerufen (Link *"View Metric"*) werden können. Diese Komponente erkennt, ob ein Benutzer einen Netzwerkdatensatz bereits ausgewertet hat oder nicht. In diesem Bereich werden alle im System registrierten Netzwerkanalyse-Module dargestellt. Ein jedes Netzwerkanalyse-Modul besitzt einen Bezeichnung und eine kurze Beschreibung.

Bereich 5: Dieser Bereich stellt berechnete Netzwerk-Metriken dar. Voraussetzung ist, dass zuvor ein Netzwerkdatensatz und eine Netzwerkanalyse-Methode ausgewählt worden ist. Dieser Bereich ist durch die Komponente $NetworkMetricsPanel$ implementiert und in die Seite eingebunden. Hauptaufgabe dieser Komponente ist, die Darstellung von berechneten Netzwerk-Metriken. Einerseits werden erstellte Diagramme im PNG-Format als Thumbnails mit einer Bezeichnung gelistet und durch einen einfachen Link (Lupen-Symbol) ist es möglich, dass Diagramm in einem modalen Fenster in Originalgröße betrachten zu können. Zusätzlich werden alle generierten Dateien in einen eigenen Sub-Bereich (*Download Files*) als Download angeboten. Diese Komponente bietet auch die Möglichkeit, die Berechnung von der Netzwerk-Metrik zu wiederholen oder die gesamte Netzwerk-Metrik zu löschen.

6.3.3.4 Vorstellung der Web-Anwendung

In diesem Abschnitt wird ein typischer Ablauf –vom Anmelden bis zur Erstellung einer Nezwerk-Metrik– anhand von Screenshots beschrieben. Es soll gezeigt werden, dass die Anwendung von komplexen Netzwerkanalyse-Methoden durch die Abbildung einer einfachen Navigationsstruktur sehr simpel erfolgen kann. Das Ziel dieser Web-Anwendung ist es, so einfach wie möglich Netzwerk-Metriken von Netzwerkdatensätzen durch Netzwerkanalyse-Module zu erstellen. Dieser Prozess muss intuitiv,

verständlich, einprägsam und einfach zu erlernen sein. Für authentifizierte Benutzer beginnt der Prozess mit dem Upload oder der Auswahl eines Netzwerkdatensatz und endet mit dem Betrachten oder dem Download von Netzwerk-Metriken. Der Prozess kann für jeden Netzwerkdatensatz und jedes Netzwerkanalyse-Modul beliebig oft durchgeführt werden.

Die Web-Anwendung *Network Metric Explorer* ist nur authentifiziert benutzbar. Abbildung 6.8 zeigt die umgesetzte Login-Seite der Web-Anwendung. Nachdem man sich erfolgreich im System angemeldet hat, folgt die Darstellung des *Network Metric Explorer*, der die eigentliche Web-Anwendung in einem Single-Page-Modell repräsentiert. In Abbildung 6.9 ist die Web-Anwendung dargestellt. In dieser Darstellung wurde noch keine Benutzer-Aktion durchgeführt, d.h. es ist noch kein Netzwerkdatensatz ausgewählt worden. Der Bereich für die Netzwerk-Metriken (*Network Metrics*) ist initial leer, und stellt den Output-Bereich für das Ende des Netzwerkanalyse-Prozesses dar.

Es können im Bereich *Network Dataset Collection* durch den Link *"Add a new network dataset"* neue Netzwerkdatensätze hochgeladen werden. In Abbildung 6.10 ist das modale Fenster für den Upload von Netzwerkdatensätzen dargestellt. Bemerkenswert ist, dass mit diesen Upload mehrere Netzerkdatensätze gleichzeitig zu einer Netzwerkdatensatz-Sammlung eines Benutzers hinzugefügt werden können.

Eine Auswertung eines Netzwerkdatensatzes kann sehr einfach durchgeführt werden. Im Bereich *"Network Dataset Collection"* muss dafür ein Netzwerkdatensatz ausgewählt werden. Danach muss im Bereich *"Network Analysis Modules"* ein gewünschtes Netzwerkanalyse-Modul ausgewählt werden, wodurch mit diesem Schritt die Berechnung der Netzwerk-Metrik startet. Sollte die Netzwerk-Metrik zuvor schon einmal berechnet worden sein, wird lediglich die vorhandene Netzwerk-Metrik angezeigt. In Abbildung 6.11 ist die Auswahl von einem Netzwerkdatensatz und einem Netzwerkanalyse-Modul dargestellt und im Bereich *"Network-Metric"* wird eine laufende Auswertung (vgl. Abbildung 6.17) visualisiert. Laufende Auswertungen werden entsprechend dieser Abbildung dargestellt. Somit erhält der Benutzer Feedback darüber, dass der Vorgang im Hintergrund ggfs. noch einige Zeit benötigt. In Abbildung 6.12 wird das Resultat nach einer abgeschlossenen Berechnung angezeigt. Eine erstellte Netzwerk-Metrik wird in der Web-Anwendung automatisch ohne eine weitere Benutzerinteraktion eingeblendet. Abbildung 6.18 zeigt das Resultat einer abgeschlossenen Netzwerk-Metrik Berechnung.

Weitere Berechnungen können jederzeit sehr einfach durchgeführt werden, indem man einfach einen neuen Netzwerkdatensatz oder eine neues Netzwerkanalyse-Modul auswählt. Es können Auswertungen auch dann gestartet werden, wenn noch laufende Auswertungen durchgeführt werden.

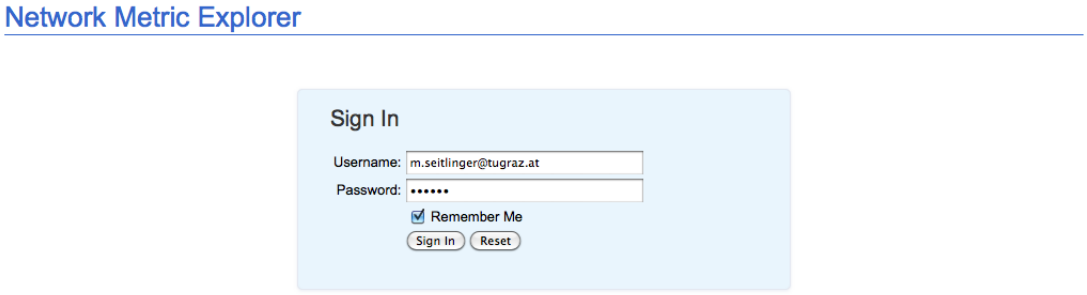

Abbildung 6.8: Login-Seite der Web-Anwendung

6.3.4 Konfigurationsmöglichkeiten

Die Web-Anwendung verfügt über die folgenden zwei Konfigurationsdateien:

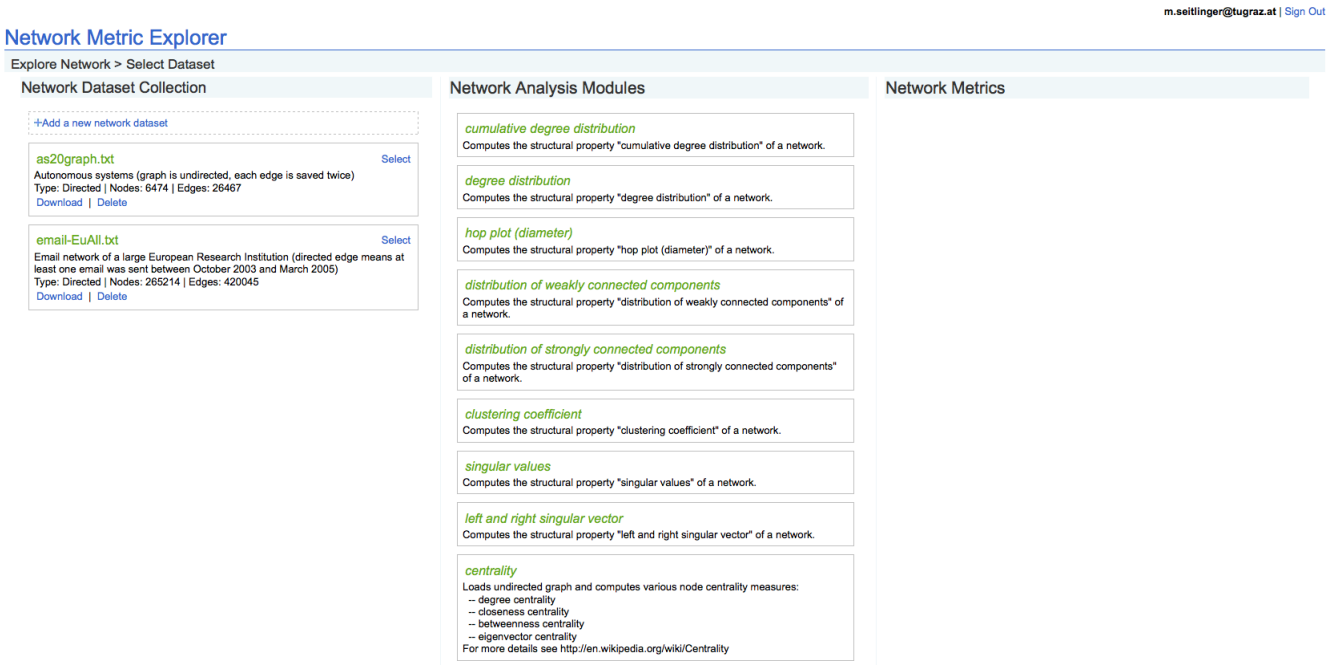

Abbildung 6.9: Startansicht der Web-Anwendung

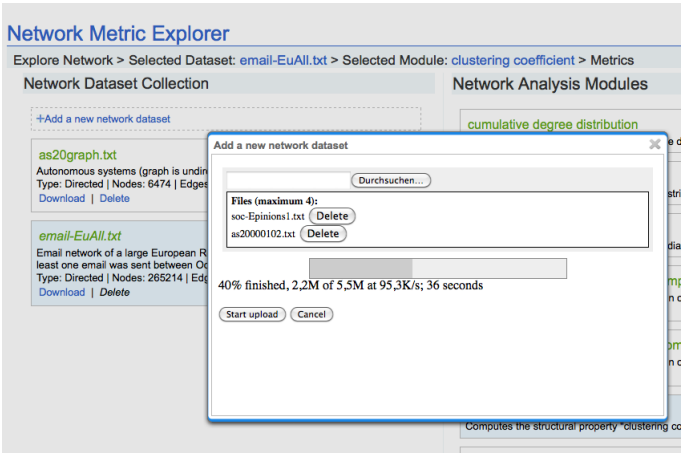

Abbildung 6.10: Upload von Netzwerkdatensätzen

EnvironmentSettings.properties: In dieser Datei wird die Verbindung zu den Komponenten Benutzer und Netzwerkanalyse-Module definiert. Die Benutzer werden im Filesystem in einem beliebigen Ordner abgelegt und verwaltet. Der Parameter $userPath$ muss konfiguriert werden und in dem angegebenen Verzeichnis muss sich die Datei $users.xml$ (vgl. Abschnitt 6.3.5) befinden, in der alle Benutzeraccounts verwaltet sind. Durch den Parameter $serviceModulesPath$ und $serviceModulesConfigFile$ wird der Pfad zu den Netzwerkanalyse-Modulen und der XML-Schnittstellen-Beschreibung der einzelnen Netzwerkanalyse-Module (vgl. Abschnitt 6.3.5) eingestellt. Der Parameter $gnuplotPath$ wird für den Pfad der GNUPlot Anwendung verwendet.

NetworkAnalysisWebApplication.properties: In dieser Datei kann der Name der Web-Anwendung (Parameter $application.title$), die Anzahl der Netzwerkdatensätze die gleichzeitig hochgeladen werden können (Parameter $numberOfMultipleUploads$) und die erlaubte Grösse in Megabytes (Parameter $maxUploadFileSize$) konfiguriert werden.

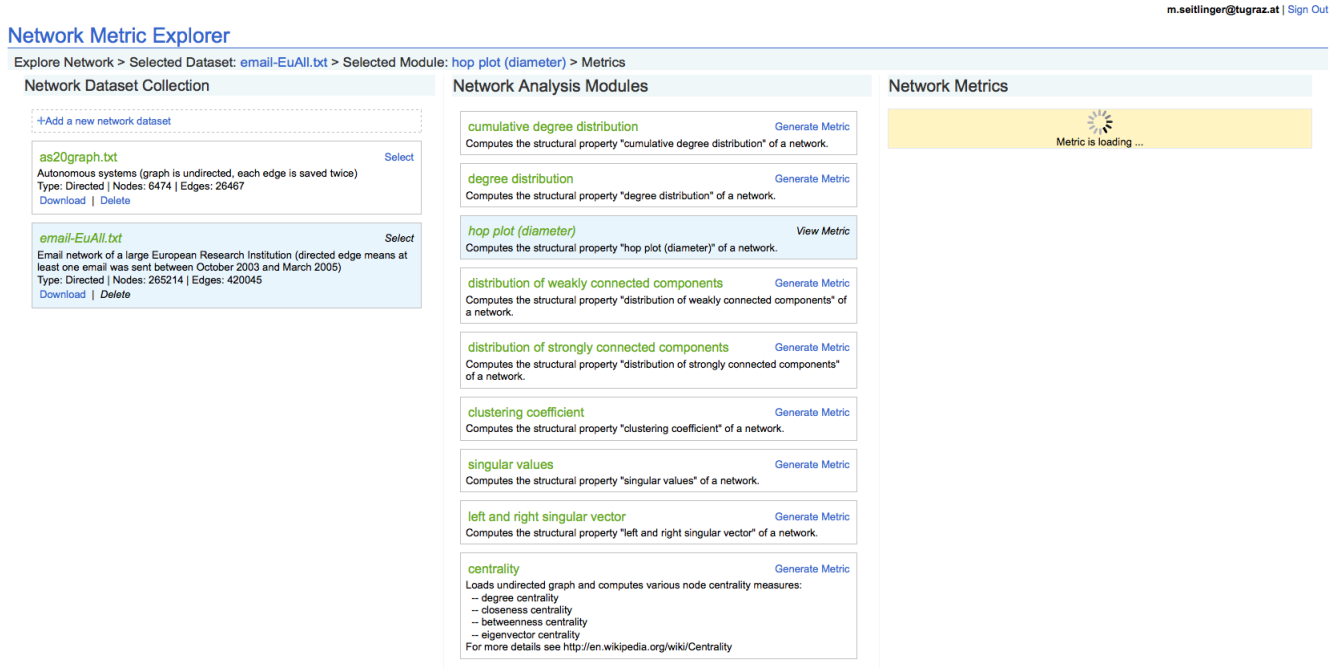

Abbildung 6.11: Durchführung einer Netzwerkanalyse

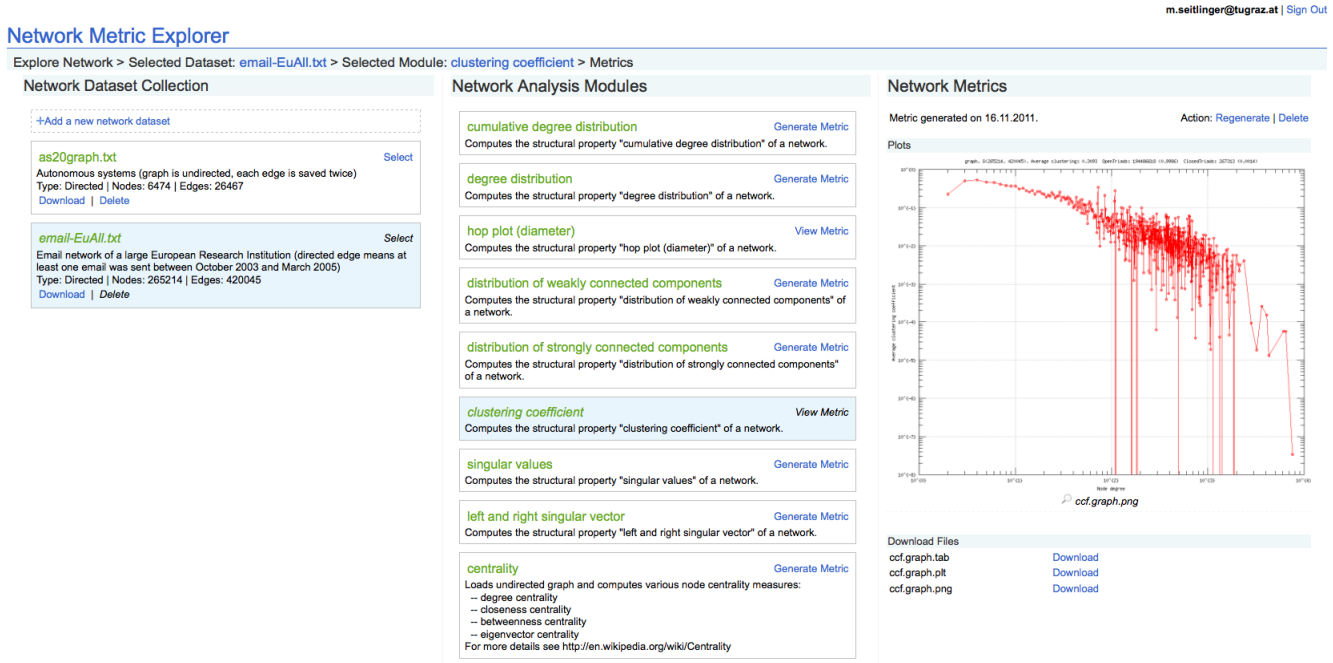

Abbildung 6.12: Ergebnis einer Netzwerkanalyse

Ferner wurden für alle WebPage- und Panel-Komponenten aus Abschnitt 6.3.2 jeweils eigene korrespondierende Properties-Dateien, indem Interfacetexte und ggfs. Fehlermeldungen abgelegt sind, angelegt. In Abschnitt 6.3.3.2 wurde beschrieben wie diese Texte von der Applikation eingebunden werden.

6.3.5 Verwaltung von Benutzer, Netzwerkanalyse-Modulen und Netzwerk-Metriken

Die Implementierung der Web-Anwendung hat im Rahmen dieser Masterarbeit auf eine relationale Datenbank, für die Verwaltung von anwendungsbezogenen Daten verzichtet. Als Persistenz-Schicht wird in diesem Projekt einfach das Filesystem verwendet. Aus diesem Grund müssen, wie in Abschnitt 6.3.4 beschrieben, bestimmte Verzeichnisse für die Web-Anwendung erstellt werden. Es müssen im Kontext dieser Web-Anwendung sehr viele Daten bzgl. der Komponenten Benutzer, Netzwerkanalyse-Module und Netzwerk-Metriken gespeichert und verwaltet werden. Diese Daten können strukturiert abgebildet werden. Ferner kann daraus ein Datenmodell abgeleitet werden. Im Rahmen dieser Masterarbeit wurden für die saubere Strukturierung dieser Daten eigene XML-Schemas entwickelt. Die Erarbeitung von XML-Schemas hat den Vorteil, dass in der Anwendung durch den Einsatz der JAXB-API (vgl. Abschnitt 6.2.4) sehr einfach und standardisiert mit entsprechenden XML-Instanzen gearbeitet werden kann. Die einzelnen Schemas werden im Folgenden beschrieben:

- NetworkAnalysisModule.xsd
- ResultNode.xsd
- Users.xsd

6.3.5.1 NetworkAnalysisModule.xsd

Dieses Schema lässt die Beschreibung von einer Liste von Netzwerkanalyse-Module zu. Ein einzelnes Netwerkanalyse-Modul, beispielsweise die einzelnen Ausprägungen des Funktionsumfangs der Anwendung $NetStat$ aus der SNAP-Library, lassen sich dadurch beschreiben. Ein jedes Modul muss eine eineindeutige ID, einen Namen und eine Beschreibung besitzen. Diese Attribute werden für die Zuordnung zu Netzwerk-Metriken und die Darstellung in der Web-Anwendung verwendet. Alle anderen Elemente sind für die Beschreibung der Schnittstelle sowie zur Referenzierung im Filesystem gedacht. In der Beschreibung wird das Konsolenprogramm und die einzelnen Parameter abgebildet und es wird der Parameter, der das Modul mit dem Datensatz versorgt, definiert. In Abbildung 6.13 ist das XML-Schema dargestellt. Eine konkrete Ausprägung einer XML-Beschreibung eines SNAP-Moduls ist in Listing 6.2 beschrieben.

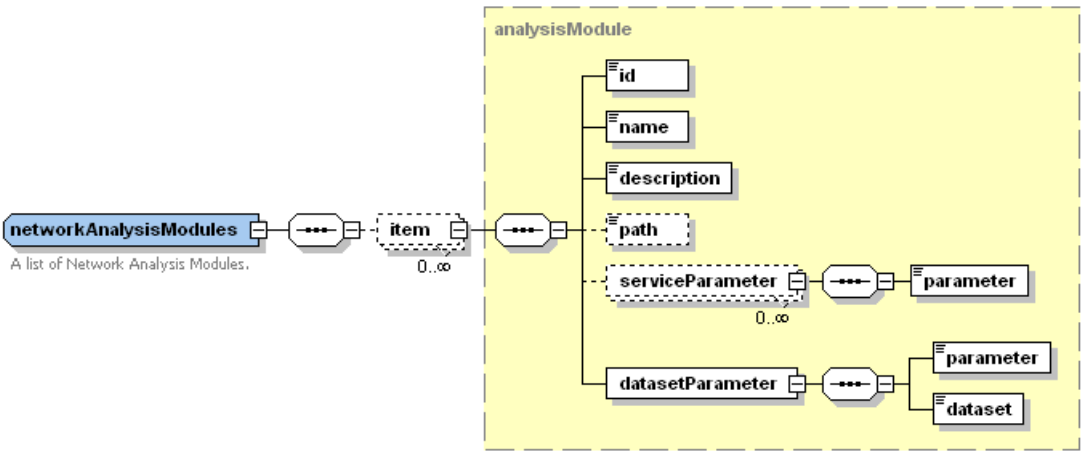

Abbildung 6.13: XML-Schema für die Beschreibung von Netzwerkanalyse-Module

6.3.5.2 ResultNode.xsd – Metadaten über Netzwerk-Metriken

Eine Netzwerk-Metrik wird im System unter der folgenden Ordnerstruktur gespeichert:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<networkAnalysisModules>
  <item>
    <id>1</id>
    <name>cumulative degree distribution</name>
    <description><![CDATA[Computes the structural property "cumulative degree
        distribution" of a network.]]></description>
    <path>/servicemodules/Snap-11-03-06/examples/netstat/netstat</path>
    <serviceParameter>
      <parameter>-p:c</parameter>
    </serviceParameter>
    <datasetParameter>
      <parameter>-i:</parameter>
    </datasetParameter>
  </item>
</networkAnalysisModules>
```

Listing 6.2: Beschreibung der Schnittstelle eines SNAP-Moduls mit XML

- $$userPath$/$$username$/results/$$UUID$/

Der Pfad $userPath$ wurde in Abschnitt 6.3.4 beschrieben. Für jeden Benutzer wird im Pfad $userPath$ der Ordner $username$ mit dem Unterordner $results$ angelegt. Der Ordner $UUID$ hat die folgende Bedeutung:

- Jeder Berechnungsvorgang einer Netzwerk-Metrik hat eine *unique* Identifikationsnummer (ID).
- Diese ID wird als Ordner für die erstellten Netzwerk-Metrik Dateien verwendet.
- Es wird vom System sichergestellt, dass nur eine einzige ID für einen Berechnungsvorgang im Kontext von Benutzer, Netzwerkdatensatz und Netzwerkanalyse-Module existiert.
- Durch diese ID kann das System jederzeit auf den Benutzer, Netzwerkdatensatz und Netzwerkanalyse-Modul und den Status der Berechnung schließen.

Damit obige Punkte tatsächlich realisiert werden, wurde ein eigenes XML-Schema entwickelt, dass zusätzliche Metadaten eines Berechnungsvorganges abbilden kann. Im wesentlichen werden Daten wie die ID des Netzwerkanalyse-Moduls und der Pfad zu dem Netzwerkdatensatz abgebildet. Wesentlich ist, dass auch der Status der Berechnung berücksichtigt wird, wodurch man darauf schliessen kann ob eine Berechnung läuft oder schon abgeschlossen ist. In Abbildung 6.14 ist das XML-Schema dargestellt. Eine konkrete Ausprägung einer XML-Instanz ist in Listing 6.2 beschrieben.

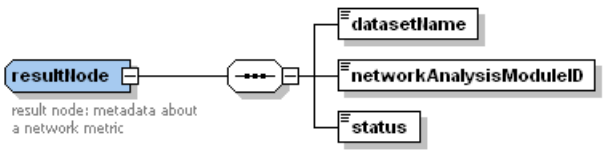

Abbildung 6.14: XML-Schema für die Beschreibung von Metadaten eines Berechnungsvorgangs einer Netzwerkanalyse

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <resultNode>
3     <datasetName>/home/mseitlinger/nawa_v1.0/user/m.seitlinger@tugraz.at/datasets/
          email-EuAll.txt</datasetName>
4     <networkAnalysisModuleID>6</networkAnalysisModuleID>
5     <status>released</status>
6 </resultNode>
```

Listing 6.3: Beschreibung eines Berechnungsvorgangs durch XML.

6.3.5.3 Users.xsd

Für die Verwaltung von Benutzern wurde ebenso ein sehr einfaches XML-Schema definiert (vgl. Abbildung 6.15). Im Wesentlichen können damit in einer Liste die folgenden Attribute eines Benutzers abgelegt werden:

- Benutzername ($username$)
- Password ($password$)

Der Pfad $userPath$ (siehe Abschnitt 6.3.4) muss eine XML-Datei mit dem Namen $users.xml$ beinhalten. Dieses Datei muss gegen das Schema $Users.xsd$ validiert werden können. Für jeden Benutzer dieser Datei ist es möglich, sich in der Web-Anwendung anzumelden. Es wird automatisch für jeden dieser Benutzer *on demand* ein eigenes Environment eingerichtet. Es handelt sich um einen eigenen Ordner, der vom Benutzernamen abgeleitet wird und umfasst zwei Unterordner. Im Unterordner $datasets$ werden alle Netzwerkdatensätze für einen Benutzer abgelegt. Der Unterordner $results$ wird für die Speicherung aller Netzwerk-Metriken verwendet. Das System erkennt, ob für einen Benutzer sein Benutzer-Environment eingerichtet ist, und kann ggfs. direkt nach der Authentifizierung das Environment bereitstellen.

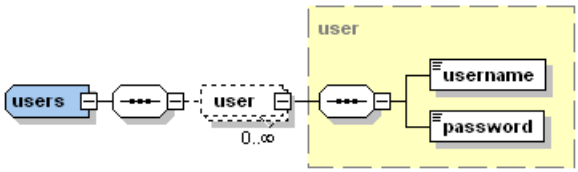

Abbildung 6.15: XML-Schema Benutzer-Daten

6.3.6 Integration von Netzwerkanalyse-Modulen

Dieser Abschnitt beschreibt das erarbeitete und umgesetzte Konzept für die Verwendung von externen Netzwerkanalyse-Modulen. In Abschnitt 6.3.4 wurde beschrieben, dass Netzwerkanalyse-Module in einem ausgezeichneten Ordner abgelegt werden können. Ferner wurde in Abschnitt 6.3.5.1 gezeigt, dass die Schnittstelle von einzelnen Netzwerkanalyse-Modulen durch ein XML-Schema beschrieben werden kann. Die in dieser Form durch XML beschriebenen Netzwerkanalyse-Module sind in das System integriert und werden in der Web-Anwendung gelistet und können auch funktional genutzt werden.

Im Service-Layer und im Integrations-Layer wurden einzelne Klassen implementiert, durch die Netzwerkanalyse-Module in die Web-Anwendung integriert und aufgerufen werden können. Alle relevanten Klassen und deren Beziehung werden in Abbildung 6.16 durch ein UML-Klassendiagramm dargestellt und im Folgenden beschrieben.

6.3.6.1 Implementierung der Dienste für den Integrations-Layer

Im Integrations-Layer wurde durch die Klasse $Command$ eine Abstraktion in Form einer Façade für die $ProcessBuilder$-Klasse, die zum Funktionsumfang von Java gehört, implementiert. Die Klasse $ProcessBuilder$ kann verwendet werden, um auf Betriebssystem-Ebene einen Prozess (Klasse $Process$) zu starten. Dies ermöglicht, dass externe Programme von Java aus aufgerufen werden können.

Die Klasse $Command$ kann verwendet werden, um ein externes Programm asynchron, durch den Aufruf der Methode $executeAsync$, aufzurufen und auszuführen. Der Methode $executeAsync$ wird eine Instanz der Template-Klasse $Future$ übergeben. Diese Methode erzeugt mit dem ProcessBuilder einen neuen Prozess, der explizit in einen eigenen Thread gestartet wird. Durch den Aufruf von $process.waitFor()$; in diesen Thread wird erzwungen, dass dieser Prozess solange blockiert, bis das externe Programm terminiert ist.

Das $Future$-Objekt ist dem Client bekannt und dieser hält eine Referenz darauf. Mit der Methode $isDone()$ kann der Client überprüfen, ob das Resultat schon erstellt ist, d.h. ferner ob das externe Programm schon terminiert ist. Die Methode $setDone()$ des $Future$-Objekts wird nachdem der Prozess des externen Programms terminiert ist, in dem Thread der durch die $Command$-Klasse erzeugt wurde, aufgerufen. Dadurch wird dem Client bei der nächsten Prüfung von $isDone()$ signalisiert, dass der Prozess terminiert ist, und beispielsweise ein Ergebnis bereitliegt.

Die $Future$-Klasse dient zusätzlich als Wrapper für eine beliebige Klasse, die das Resultat einer Berechnung darstellt. Im Kontext der Web-Anwendung wird die Klasse $Result$ verwendet, die als Membervariable die ID des Berechnungsvorgang hält.

6.3.6.2 Konzept im Service-Layer

Für den Service-Layer wurde die Klasse $NetworkAnalysisModuleService$ implementiert. Diese Service-Klasse hat die folgenden Aufgaben:

- Bereitstellung von Informationen über alle Netzwerkanalyse-Module für einzelne Wicket-Komponenten des Presentation-Layer.
- Durchführung von Netzwerkanalyse-Berechnungen im Kontext eines Benutzers, Netzwerkdatensatz und Netzwerkanalyse-Moduls. Für die Berechnung wird eine unique ID generiert und mit dem Berechnungsvorganges verbunden. Es wird daraus eine ResultNode-Beschreibung erstellt (vgl. Abschnitt6.3.5.2) und gespeichert.

Diese Klasse kann durch die Factory $NetworkAnalysisModuleObjectFactory$ Java-Objekte integrierten Netzwerkanalyse-Module erzeugen. Aufgrund einer besseren Performance cached das Service-Objekt alle Netzwerkanalyse-Module und wurde als *Singleton* implementiert. In Abschnitt 6.3.5.1 wurden Details über die XML-Beschreibung von Netzwerkanalyse-Module beschrieben.

Diese Service-Klasse kennt alle im System integrierten Netzwerkanalyse-Methoden. Durch die Verwendung der zuvor beschriebenen $Command$ Klasse kann die Service-Implementierung einzelne Netzwerkanalyse-Module aufrufen. Diesbezüglich dient die Methode $applyModuleAsync$, die im Kontext für einen

Benutzer, Netzwerkdatensatz und ein Netzwerkanalyse-Modul aufgerufen wird. Aufgrund der Implementierung der Klasse $Command$ wird ein *asynchroner* Aufruf von Netzwerkanalyse-Modulen durchgeführt. Die Methode $applyModuleAsync$ liefert eine Instanz der $Future$ Klasse zurück, sodass im Service-Layer der Status des Berechnungsvorgang überprüft werden kann.

Die Methoden $initEnvironment()$ und $afterExecuteProcess()$ der $Command$-Klasse werden in der Service-Klasse in einer *anonymen* Klasse überschrieben. Durch die Implementierung der Methode $initEnvironment()$ wird GNUPlot in den PATH eingebunden, sodass sichergestellt wird, dass in den Netzwerkanalyse-Modulen GNUPlot verwendet werden kann. Die Implementierung der Methode $afterExecuteProcess()$ berücksichtigt, dass nach dem abgeschlossenen Berechnungsvorgang, der im Abschnitt 6.3.5.2 beschriebenen Status der Berechnung in einer vorhandenen XML-Datei auf den Wert "Berechnung abgeschlossen" gesetzt wird. Durch diese Praxis können Details berücksichtigt werden, ohne dass die Generik von der Klasse $Command$ eingeschränkt wird. In Listing 6.4 ist dieser Vorgang in den Zeilen 1 bis 21 zusehen. Die Implementierung der Service-Methode $applyModuleAsync$ ist in den Zeilen 23 bis 49 zu sehen. Die $ResultNode$-Klasse ist die Java-Repräsentation des in Abschnitt 6.3.5.2 beschrieben XML-Schema ResultNode.xsd und entspricht der konkreten Anwendung sowie der Abstraktion von JAXP (vgl. Abschnitt 6.2.4).

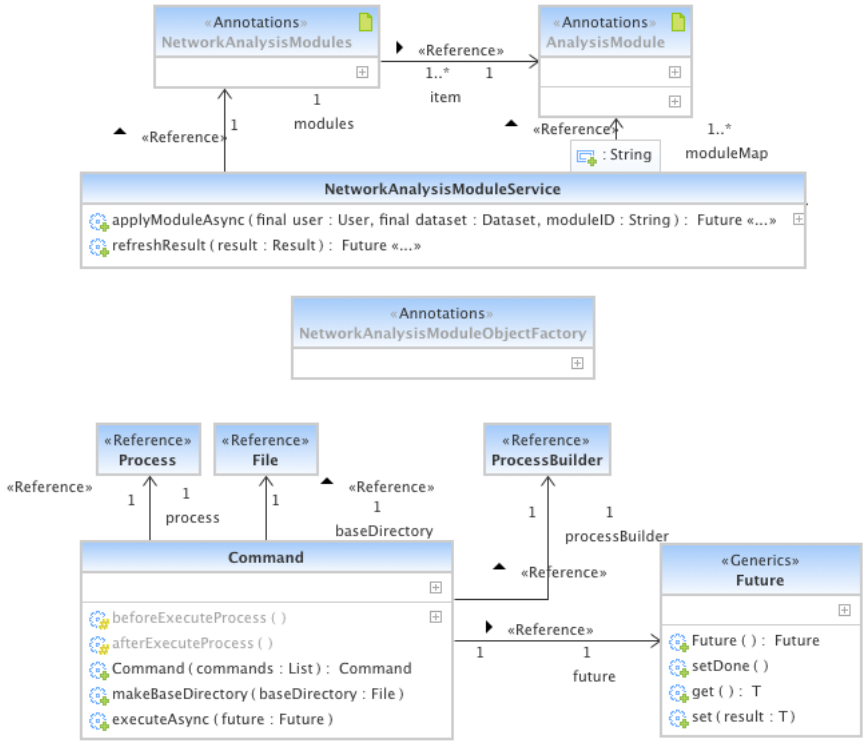

Abbildung 6.16: UML Diagramm aller relevanten Klassen für die Integration von Netzwerkanalyse-Modulen

6.3.7 Umsetzung der (Long-)Polling Strategie

In Abschnitt 6.3.6 wurde beschrieben, wie Netzwerkanalyse-Module asynchron aufgerufen werden können ohne dass das System blockiert. Ferner wurde gezeigt, dass durch eine geeignete Service-Implementierung und durch das Objekt $Future$ im Service-Layer geprüft werden kann, ob die Berechnung schon abgeschlossen und die Netzwerk-Metriken erstellt wurden. Ferner existiert die Möglichkeit im Service-Layer über eine eindeutige ID eines Berechnungsvorganges für den Kontext Benutzer, Netzerkdatensatz und Netzweranalyse-Modul zu prüfen, ob der Berechnungsvorgang noch läuft oder bereits abgeschlossen ist,

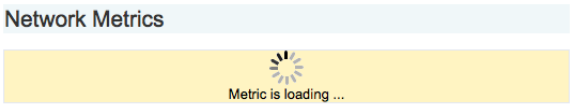

Abbildung 6.17: Hinweis für eine laufende Auswertung

indem in der XML-Datei *resultNode.xml* (vgl. Abschnitt 6.3.5.2) das Element $status$ überprüft wird.

Eine nicht-triviale Herausforderung ist es, den Aspekt der asynchronen Berechnung und die damit ver-

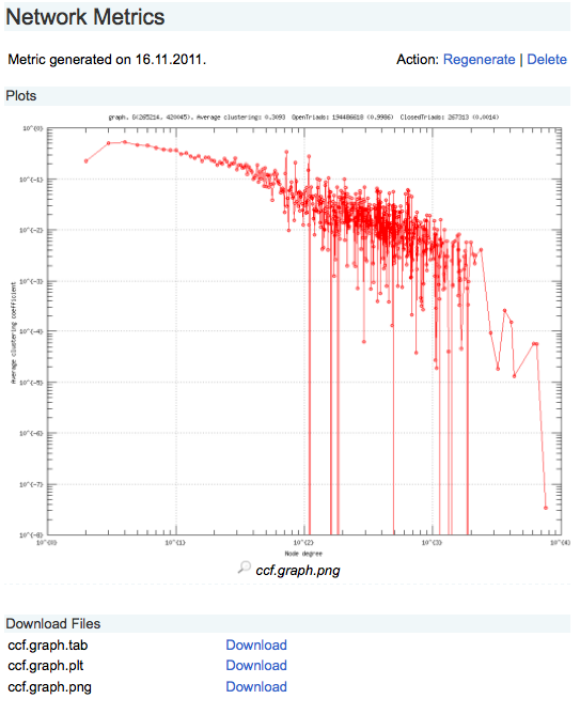

Abbildung 6.18: Darstellung von einer berechneten Netzwerk-Metrik

bundenen transienten Daten in der Benutzerschnittstelle adäquat zu berücksichtigen. Aus den Anforderungen (vgl. Kapitel 5) geht hervor, dass bei einer laufenden Berechnung, die Web-Anwendung nicht blockieren darf und dass mehrere Berechnungen parallel durchgeführt werden können sollen. Ferner soll in der Benutzerschnittstelle ein Benutzer Feedback über laufende Berechnungen erhalten und sofern Ergebnisse einer Auswertung vorliegen, müssen diese im Kontext eines Benutzers, Netzwerkdatensatz und Netzwerkanalyse-Methode angezeigt werden.

Es stellt sich ein typisches Problem der *Real-Time Revolution* im WWW. Der Benutzer der eine Web-Anwendung mit aktuellen Daten geladen hat, sieht womöglich im nächsten Moment schon veraltete Daten [Weßendorf, 2010]. Es gibt mehrere Möglichkeiten wie man dieses Problem lösen kann. Einerseits kann man die Seite mit Ajax-Polling aktuell halten. In diesem Fall wird durch einen Ajax-Request alle N-Sekunden überprüft, ob am Server für einem konkreten Kontext neue Daten vorhanden sind. D.h. im praktischen Anwendungsfall dieser Web-Anwendung, ob der Berechnungsprozess schon abgeschlossen ist und Netzwerk-Metriken vorhanden sind und geladen werden können. Eine Alternative ist der Ansatz von Long-Polling. In diesen Fall wird per Ajax ein Request an den Server gestellt. Dieser Request wird vom Sever so lange offen gehalten, bis neue Daten verfügbar sind [Weßendorf, 2010] oder ein Time-Out-Intervall erreicht wurde. Eine weitere Möglichkeit ist Http-Streaming, wobei ein Client einen

Request abschickt, der offen gelassen wird, der Server jedoch dem Client mehrere Requests zukommen lässt [Weßendorf, 2010]. In Abbildung 6.19 ist das Prinzip von Long-Polling und Http-Streaming dargestellt. Ajax-Polling, Long-Polling und Http-Streaming sind keine idealen Lösungen und sind *per se* mit Nachteilen verbunden. Die Palette der Nachteile reicht von unnötigen Traffic, Latenzprobleme usw. Durch diese Alternativen wird versucht ein Bidirektionales Kommunikationsmodel zwischen Client und Server zu emullieren. In Zukunft wird es mit der Durchdringung von HTML5[18] sogenannte WebSockets [19] geben, wodurch ein Bidirektionaler Kommunikationsstandard für das Web verfügbar wäre [Weßendorf, 2010].

Schlussendlich wurde in diesem Projekt Ajax-Pooling verwendet. Das Web-Framework Wicket stellt die Klasse $AjaxSelfUpdatingTimerBehavior$ bereit. Diese Klasse kann jeder Komponente zugeordnet werden, wodurch definiert wird, wie sich diese Komponente verhält. Im konkreten Anwendungsfall würde die Zuordnung bewirken, dass sich eine Komponente nach einem Intervall neu rendert. Ein großer Vorteil dieser Komponenten ist, dass alle technischen Details hinter dieser Komponente versteckt und gekapselt sind. Die Komponente wird mit einem Intervall in Sekunden versorgt, der definiert in welchen Zeitabständen diese Komponente sich selbst updaten soll. Es kann die Methode $onPostProcessTarget$ implementiert werden. Die Methode macht so lange nichts bis erkannt wurde, dass die Berechnung der Netzwerk-Metrik abgeschlossen ist. In diesem Fall werden in der Web-Seite per Ajax und DOM-Operationen einzelne Container ausgetauscht, wodurch ein Übergang der Darstellung einer laufenden Berechnung (vgl. Abbildung 6.17) zu der konkreten Ansicht der Netzwerk-Metriken inklusive der Darstellung von Diagrammen (Abbildung 6.18) ermöglicht wird. Listing 6.5 zeigt die konkrete Definition der eingesetzten $AjaxSelfUpdatingTimerBehavior$ Komponente.

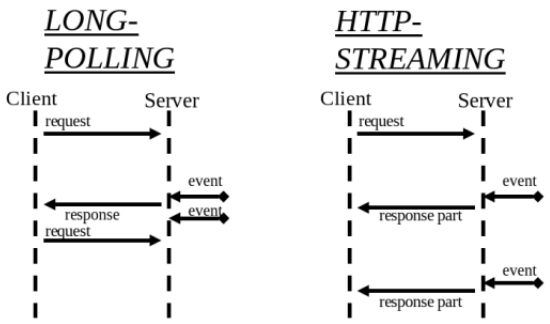

Abbildung 6.19: Prinzip und Ablauf von *Long-Polling* und *Http-Streaming* [Weßendorf, 2010]

[18] HTML5: `http://www.w3.org/TR/html5/`
[19] WebSockets: `http://dev.w3.org/html5/websockets/`

```
1    private Command getAsyncCommand (List<String> commands) {
2      return new Command( commands ) {
3
4        @Override
5        protected void afterExecuteProcess () {
6          ResultNodeService resultNodeService = new ResultNodeService();
7            ResultNode resultNode = resultNodeService.getResultNodeByPathName(
                 baseDirectory.getAbsolutePath()  );
8          resultNode.setStatus( StatusType.RELEASED );
9          resultNodeService.addResultNode(baseDirectory.getAbsolutePath(),
               resultNode);
10       }
11
12       @Override
13       protected void initEnvironment () {
14         /** Path for GNUPlot */
15         Map<String, String> env = super.processBuilder.environment();
16         env.clear();
17         String path =  env.get("PATH");
18         env.put("PATH", path + File.pathSeparator + EnvironmentSettings.
               getInstance().getGnuplotPath() );
19       }
20     };
21   }
22
23     public Future <Result> applyModuleAsync (final User user, final Dataset
           dataset, String moduleID ) {
24     ResultNodeService resultNodeService = new ResultNodeService();
25     Command command = getAsyncCommand ( getModuleCommand(moduleID, dataset) );
26
27     String resultNodeID = UUID.randomUUID().toString();
28
29     ResultNode resultNode = resultNodeService.createResultNode();
30     resultNode.setNetworkAnalysisModuleID(moduleID);
31     resultNode.setDatasetName(dataset.getPathName());
32     resultNode.setStatus(StatusType.UNRELEASED);
33     Result result = new Result ( resultNode );
34     result.setResultID( resultNodeID );
35
36     /** new resultID directory    */
37     command.makeBaseDirectory( new File(user.getResultDirectory()+"/"+resultNodeID
           ) );
38
39     Future <Result> future = new Future <Result> ();
40     future.set(result);
41     try {
42       command.executeAsync( future );
43     } catch (IOException e) {
44       resultNode.setStatus(StatusType.RELEASED);
45     }
46     resultNodeService.addResultNode(user, resultNodeID, resultNode);
47     return future;
48   }
```

Listing 6.4: Details der Implementierung der Klasse NetworkAnalysisModuleService

```
1   final AjaxSelfUpdatingTimerBehavior timerBehavior = new
        AjaxSelfUpdatingTimerBehavior(Duration.seconds(1)) {
2     protected void onPostProcessTarget(AjaxRequestTarget target) {
3       if ( resultID != null ) {
4         if ( !isResultPending() ) {
5           longPollingFeedbackContainer.setVisible(false);
6           ResultNodeService resultNodeService = new ResultNodeService();
7           fileListView.setList( resultNodeService.getResultFiles( ((
                NetworkAnalysisSession) getSession()).getUser(), resultID ) );
8           downloadContainer.setVisible(true);
9           target.addComponent( downloadContainer );
10
11          imageListView.setList( ImageService.getInstance().getImages(
                getResultDirectory(resultID).getAbsolutePath() ) );
12          plotListContainer.setVisible(true);
13          target.addComponent( plotListContainer );
14
15          metricInformationContainer.setVisible(true);
16          target.addComponent( metricInformationContainer );
17
18        }
19      }
20    }
21  };
```

Listing 6.5: Details der Implementierung der Polling-Strategie

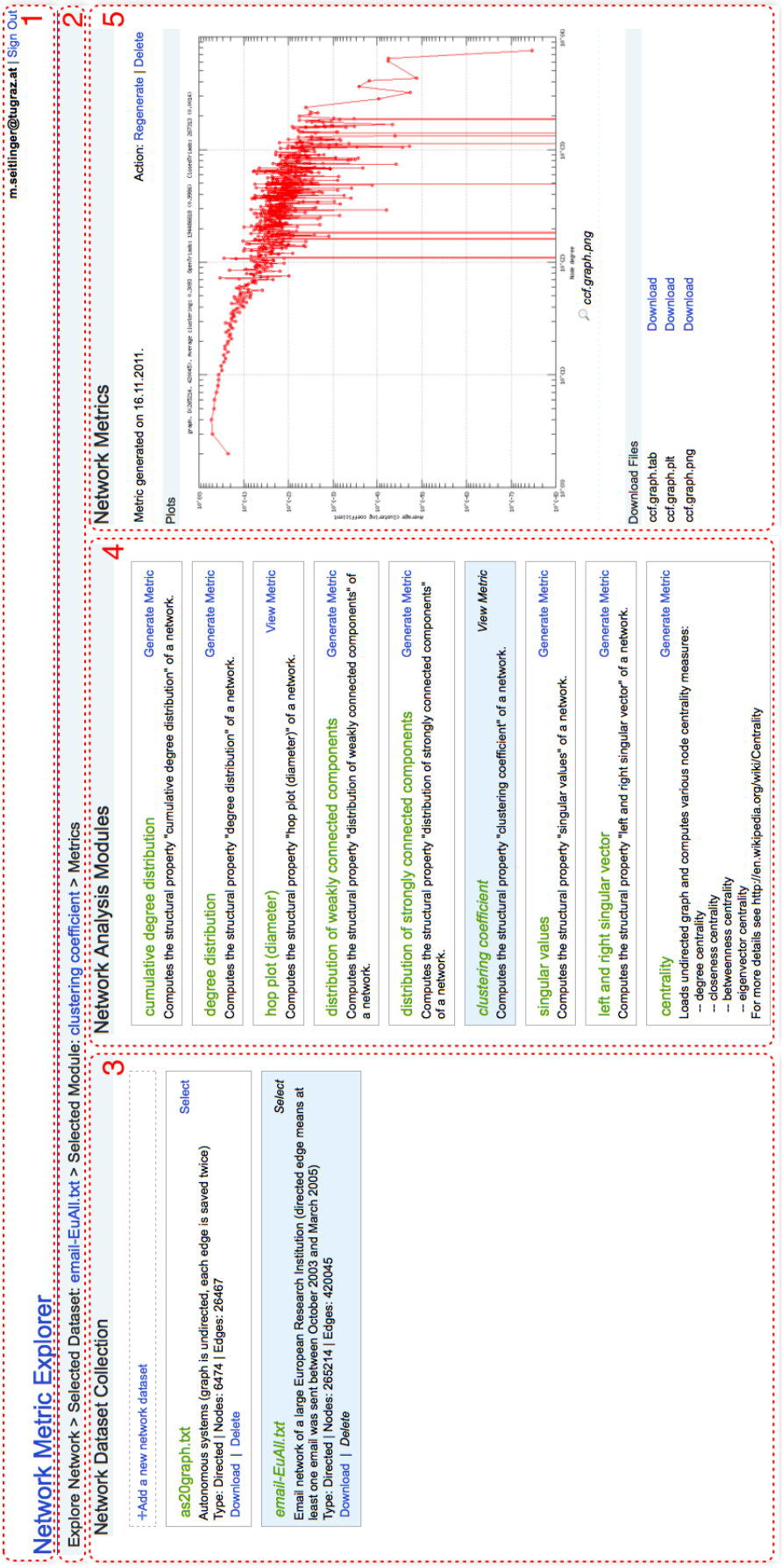

Abbildung 6.20: Layout und einzelne Bereiche der Web Anwendung *Network Metric Explorer*

Kapitel 7

Evaluierung

Dieses Kapitel liefert einen Überblick über die im Rahmen dieser Masterarbeit durchgeführte Evaluierung der erstellten Web Anwendung *Network Metric Explorer*.

7.1 Motivation

Die im Rahmen dieser Masterarbeit konzeptuell erarbeitete und umgesetzte Web Anwendung, soll im Rahmen der Lehrveranstalltung *Web Science and Web Technology* Studierenden als Lernumgebung für spezifische Aufgaben im Bereich der Netzwerkanalyse zur Verfügung gestellt werden. In Kapitel 5 wurde als ein Ziel dieser Masterarbeit definiert, dass die erarbeitete Web-Anwendung schlussendlich als *usable* E-Learning Instrument eingesetzt werden kann und das der Qualitätsfaktor Usability (vgl. Kapitel 4) dafür notwendigerweise bestmöglich erfüllt werden muss. Auch Aspekte der EduPunk-Bewegung (vgl. Kapitel 4) mussten berücksichtigt werden, mit dem Ziel eine moderne Lernumgebung für das Web zu schaffen.

Aus diesem Grund ist es ungemein wichtig zu prüfen, ob die Anforderungen erreicht wurden und ob der Qualitätsfaktor Usability ausreichend ist. Somit ergab sich als abschließende Aufgabe der praktischen Arbeit dieser Masterarbeit die Durchführung einer informalen Evaluierung der Web Anwendung. Dabei wurden folgende Ziele verfolgt:

- Identifikation und Behebung von Fehler und Show-Stopper
- Erlangung von Feedback, ob das System *per se* praktikabel ist
- Ableitung von Maßnahmen, die helfen das System in weiteren Projekten zu erweitern und zu verbessern

In Kapitel 5 wurde argumentiert, warum keine der in Kaptitel 4 Abschnitt 4.1 vorgestellten Inspektions- und Testmethoden eingesetzt wurden. Solche Evaluierungen sind aufgrund der zeitlichen Ressourcen, die mit dem Rahmen der Masterarbeit einhergehen nicht möglich *lege artis* und wissenschaftlich anzuwenden und durchzuführen. Das Ziel dieser Evaluierung ist es zukünftige Akteure in den Entwicklungsprozess einzubeziehen. Durch diese Feedback-Maßnahme soll man eine Bewertung der resultierenden Qualität der Web-Anwendung erhalten, damit geeignete Maßnahmen abgeleitet werden können, die helfen das System in weiteren Iterationszyklen und Folgeprojekten zu verbessern.

In Kapitel 8 wird in einem abschließenden Ausblick alle erhaltenen und abgeleiteten Ideen und Maßnahmen, die für eine Verbesserung des umgesetzten Systems verwendet werden können, einbezogen und in den entsprechenden Kontext gesetzt.

7.2 Zielgruppe

Es ist sinnvoll, dass die Web-Anwendung aus möglichst vielen Blickwinkeln evaluiert wird. Es wurde zudem entschieden, dass die folgenden Zielgruppen bei der Evaluierung berücksichtigt werden:

Experten: Experten der Domäne Web Science können den Prozess von Netzwerkanalysen nachvollziehen und ggf. Fehler erkennen.

Pädagogen: Der Blickwinkel auf didaktische Aspekte und Fehler ist durch die Einbeziehung von Pädagogen möglich. Zusätzlich können durch diese pädagogische Sicht auch Aspekte berücksichtigt werden, wie diese Web-Anwendung als Lernumgebung genutzt werden kann.

Studierende: Studierende sind die zukünftigen Anwender dieser Web-Anwender. Studierende können sofort Feedback geben, ob die Ziele effektiv und effizient erreicht werden können und ob die Benutzerschnittstelle intuitiv, einfach zu bedienen und gebraustauglich ist.

7.3 Durchführung

Nachdem die Web-Anwendung einen stabilen Entwicklungsgrad erreicht hatte, wurde die Anwendung auf einem Server (`https://ext201.know-center.tu-graz.ac.at:8443/NAWA_v1.0/`) installiert. Für Testpersonen wurden Accounts eingerichtet.

Anschließend wurden Testpersonen nach den Kriterien aus Abschnitt 7.2 ausgewählt. Alle Testpersonen wurden instruiert mit einem eigenen Account in die Web-Anwendung einzusteigen und typische Schritte durchzuführen:

- Login in das System
- Upload von Netzwerkdatensätzen
- Durchführung von Netzwerkanalysen
- Betrachtung und Download von Netzwerk Metriken
- etc.

Alle Testpersonen waren angehalten ihr Feedback verschriftlichen und per Mail abzugeben. Aus den gesammelten Informationen wurden zu potentielle Stärken und potentielle Verbesserungen gebündelt. Abschnitt 7.4 dokumentiert die Ergebnisse dieser Auswertung.

Show-Stopper und aufgetretene Fehler sind in der folgenden Betrachtung nicht enthalten, sondern wurden im Testsystem bereits ausgebessert.

7.4 Auswertung

Dieser Abschnitt liefert einen Überblick über alle abgeleiteten potentiellen Stärken und Verbesserungen.

7.4.1 Potentielle Stärken

Aus den Rückmeldungen kann geschlossen werden, dass das System einfach zu bedienen ist, die Web-Anwendung übersichtlich gestaltet wurde und von den Benutzern akzeptiert wird. Die derzeitige Version sieht bereits sehr gut aus und könnte in dieser Version und mit diesem Funktionsumfang den Studierenden der Lehrveranstaltung *Web Science and Web Technology* im Sommersemester 2012 zur Erprobung zur Verfügung gestellt werden. Daraus ist zu schließen, dass die Anforderungen aus Kapitel 5 bereits erfüllt wurden.

7.4.2 Potentielle Verbesserungen

Die Rückmeldungen der Evaluierung haben auch eine Liste von potentiellen Verbesserungsvorschlägen gebracht. Dieses Feedback ist sehr wertvoll, denn daraus können Maßnahmen abgeleitet werden, die helfen das System zu verbessern. Einerseits stellen die folgenden Punkte erweiterte Anforderungen im Kontext von Kapitel 5 dar und zusätzlich können damit auch vorhandene Anforderungen geschärft werden.

7.4.2.1 Erläuterungen, Hilfeseite und Kontaktmöglichkeit

Aus *pädagogisch-didaktischer* Sicht könnte die Web-Anwendung am meisten profitieren, wenn die Umgebung Erläuterungen liefert. In der vorhandenen Umsetzung ist auch keine Hilfeseite vorgesehen. In einer Hilfeseite könnte der Zusammenhang und der Aufbau der drei Spalten der Hauptseite erläutert werden. Einzelne Netzwerkanalyse-Methoden könnten auch mit relevanten Ressourcen (Wikipedia, etc.) verlinkt werden. Durch eine kurze Erläuterung oder durch ein Beispiel könnte die Upload-Funktion von Netzwerkdatensätzen intuitiver gestaltet werden. Das modale Fenster für den Upload von Netzwerkdatensätzen könnte diesbezüglich um einen Info-Button erweitert werden.

In einer Hilfeseite könnten neben einer Kontaktmöglichkeit auch zusätzlich die folgenden Fragen beantwortet werden:

- Was ist zu tun, falls man sein Password verloren hat.
- Was ist zu tun, falls ein spezifisches Problem auftritt.

7.4.2.2 Einführende Erklärung

Nachdem man sich das erste mal in der Web Anwendung angemeldet hat, muss man Netzwerkdatensätze hochladen, damit man folgend die Netzwerkanalyse-Funktionalität anwenden kann. Eine potentielle Verbesserung um das System intuitiver zu gestalten wäre, wenn man die Benutzer in der Benutzerschnittstelle darauf hinweist, dass zuerst Netzwerkdatensätze hochzuladen sind.

Ideal wäre es, wenn in der Breadcrump-Section der Web-Anwendung – solange noch kein Netzwerkdatensatz hochgeladen wurde – die Information *"Explore Network >Upload a Dataset first"* stehen würde.

7.4.2.3 Download von Plot-Dateien

Der Download von einzelnen Plot-Dateien ist sofern ein Netzwerkanalyse-Modul mehrere Diagramme erstellt unübersichtlich. Eine Verbesserung könnte erzielt werden, wenn der Download direkt unter einem Diagramm angeboten wird. Die Spalte *"Network Metrics"* könnte ggfs. auch mehr Platz bekommen.

7.4.2.4 Download von Netzwerk Metriken als Zip-Archiv

In der Web-Anwendung werden einzelnen Dateien einer Netzwerk Metrik als Download angeboten. Es kam der Wunsch auf, dass in der Web-Anwendung alle Dateien einer Netzwerk Metrik als ein Zip-Archiv als Download angeboten werden.

7.4.2.5 Netzwerk Metriken und Erstellungsdatum

Eine Netzwerk Metrik hat ein Erstellungsdatum, durch das man schließen kann, wann eine Berechnung durchgeführt wurde. Die Evaluierung brachte das Ergebnis, dass zusätzlich zum Datum auch eine Uhrzeit wünschenswert wäre. Führt man an einem Tag mehrere Berechnungen durch oder man probiert mehrere Sachen durch, dann würde die Uhrzeit helfen sich besser zu orientieren.

7.4.2.6 View Metric versus Generate Metric

Der Unterschied zwischen *Generate Metric* und *View Metric* ist nicht immer klar, obwohl aus *pädagogisch-didaktischer* dieses Feature hervorgehoben wurde. Die Intention des Begriffspaares war, dass dadurch informiert werden soll ob eine Berechnung durchgeführt werden muss (*Generate Metric*) oder ob eine vorhandene Netzwerk-Metrik einfach eingesehen werden kann (*View Metric*).

Ggfs. muss hier eine intuitivere Bezeichnung gefunden werden oder es wird in der Web-Anwendung eine Erläuterung (vgl. 7.4.2.1)angeführt. Tatsächlich werden durch das Begriffspaar konkrete Prozess-Details nach außen getragen.

7.4.2.7 Auswahlmöglichkeit Netzwerktyp

Die Beispielanwendungen der SNAP-Library können in der Regel parametrisiert werden, ob der zu analysierende Datensatz als gerichtetes oder ungerichtetes Netzwerk interpretiert und ausgewertet werden soll.

Gerichtete und ungerichtete Netzwerke sind in der Web Applikation erlaubt. Für gerichtete Netzwerke ergeben sich ggfs. andere Netzwerk Metriken als für das gleiche Netzwerk, das nur ungerichtete Kanten besitzt. Pädagogisch relevant wäre, wenn man gerichtete Netzwerke auch als ungerichtete Netzwerke auswerten kann. Dadurch werden Unterschiede sofort sichtbar.

Eine entsprechende Auswahl-Möglichkeit wäre in der Web-Anwendung ein sehr wünschenswertes Feature.

7.4.2.8 Berechnungen wiederverwenden

Das derzeitige Konzept der Web-Anwendung ist, dass für einen Benutzer und ein Netzwerkanalyse-Modul ein Netzwerkdatensatz genau einmal ausgewertet wird. Das Ergebnis der Auswertung ist eine Netzwerk Metrik und wird in diesem Kontext gespeichert, sodass ein Benutzer keine Berechnung mehrmals durchführen muss.

Führt ein weiterer Benutzer mit dem selben Netzwerkdatensatz als ein anderer Benutzer die selbe Berechnung durch, dann werden im System nicht auf vorhandenen Ergebnisse zurückgegriffen, sondern es wird für diesen Benutzer die Berechnung durchgeführt und das Ergebnis in diesem Kontext gespeichert.

Das System sollte erkennen, ob ein Netzwerkdatensatz schon von einem anderen Benutzer ausgewertet wurde, da die Berechnung von Netzwerk Metriken für größere Netzwerke mitunter sehr zeitintensiv ist.

Die Web-Anwendung könnte man diesbezüglich verbessern, indem man Berechnungen, die im System schon einmal durchgeführt wurden, wiederverwendet. Somit können Benutzer ggfs. auf die Auswertungsergebnisse von anderen Benutzern zugreifen. Dieser Aspekt würde helfen sogenannte *computational cycles* sparen. Dadurch könnte die Performance des Systems verbessert werden.

Kapitel 8

Ausblick

Diese Masterarbeit hat die Möglichkeit der technischen Erstellung einer im Kern sehr komplexen Web-Anwendung für Netzwerkanalyse-Methoden mit hoher Usability bewiesen. Es wurde zusätzlich erreicht, dass eine Lernumgebung im Sinne der EduPunk-DIY Attitüde erstellt wurde. Dieses Kapitel gibt in Abschnitt 8.1 einen kurzen Überblick über die erzielten Resultate, die in dieser Masterarbeit bezüglich der Umsetzung (vgl. Kaptitel 6) und Evaluierung (vgl. Kapitel 7) der Web-Anwendung *Network Metric Explorer* erreicht wurden. In diesem Kontext werden in Abschnitt 8.2 gegenwärtige Trends aufgegriffen und mit den bestehenden Ergebnissen vernetzt. Abschließend werden in Abschnitt 8.3 Ideen für die zukünftige Weiterentwicklung des erstellten Systems geliefert.

8.1 Network Metric Explorer

Diese Masterarbeit war mit einer eingehenden Literaturrecherche über die Bereiche Netzwerkanalyse (vgl. Kapitel 2), Web Engineering (vgl. Kapitel 3), Usability und EduPunk (vgl. Kapitel 4) verbunden. Parallel wurde an der Konzeption und der Erstellung der Web-Anwendung *Network Metric Explorer* gearbeitet. Im Rahmen dieser Arbeit wurden Anforderungen abgeleitet (vgl. Abschnitt 5), technische Hilfsmittel ausgewählt, vorhandenes Architekturwissen wiederverwendet und systematisch die Web-Anwendung *Network Metric Explorer* (vgl. Abschnitt 6) entwickelt.

In der Umsetzungsphase der Web-Anwendung war das Web Framework Apache Wicket eine geeignete Auswahl, jedoch muss der Nachteil von fehlenden Komponenten-Bibliotheken festgehalten werden. In einer abschließenden Untersuchung wurde das System einer Evaluierung (vgl. Kapitel 7) unterzogen, welche zu dem positiven Ergebnis kam, dass die Anforderungen weitgehend erfüllt sind und dass das System von den Benutzern akzeptiert wird.

Schließlich konnten auch potentielle Verbesserungen abgeleitet werden, die helfen würden, dass System in Zukunft zu verbessern. Durch relativ wenig Aufwand könnte erreicht werden, dass das System intuitiver gemacht werden kann. Im System müsste man nur eine Hilfeseite und bedarfsgerecht bessere Beschreibungen in der Benutzerschnittstelle integrieren. Im Sinne des EduPunks liegt sehr viel Potential in der sinnvollen Berücksichtigung von verbesserten Interaktionsmöglichkeiten. Beispielsweise wäre es didaktisch sehr angebracht und wertvoll, wenn ein lernender Benutzer bei einem Netzwerkdatensatz selbst entscheiden kann, ob ein Netzwerkdatensatz gerichtet oder ungerichtet durch eine Netzwerkanalyse ausgewertet werden soll. Dieses Feature würde sofort die Neugier der Lernenden wecken, wodurch Unterschiede beispielsweise in Form von unterschiedlichen Diagrammen bemerkbar werden.

Das System kann natürlich auch technisch verbessert werden. Die Auswertung von Netzwerkdatensätzen ist mitunter sehr zeitintensiv. Es ist absehbar, dass in Zukunft mit dieser Applikation mehrerer Benutzer

arbeiten, die mitunter die gleichen Netzwerkdatensätze auswerten müssen. Die Performance des Systems könnte verbessert werden, indem Auswertungen besser wiederverwendet werden. Derzeit werden nur bestehende Auswertungen für einen Benutzer wiederverwendet.

8.2 Aktuelle Trends

HTML5 ist ein aktueller Trend und ein kommender Standard. Web-Anwendungen können von HTML5 Features enorm profitieren und es steht mitunter eine neue (technische) Revolution im Web bevor. HTML5 bringt durch sogenannte WebSockets unmittelbar eine Lösung für die *Real-Time Revolution* des WWW [Weßendorf, 2010]. Mit WebSockets wird ein bidirektionaler Kommunikationsstandard für das Web verfügbar. Dadurch könnten Probleme, die durch die Emulation eines bidirektionalen Kommunikationsmodells und durch Techniken wie Ajax-Polling, Long-Polling oder Http-Streaming entstehen, ausgeschlossen werden. Weitere Entfaltungsmöglichkeiten von Web-Anwendungen sind mit HTML5 durch das Canvas Element und durch sogenannte WebWorkers gegeben. Es ist zu erwarten, dass bestehende Web-Frameworks auf die Features von HTML5 reagieren und einzelne Konzepte in den Funktionsumfang aufnehmen. HTML5 wird zur Zeit noch nicht vollständig von allen Browsern unterstützt. Es existieren jedoch schon die ersten HTML5-Web-Anwendungen. Apache Wicket unterstützt teilweise schon einige Features.

8.3 Ideen für zukünftige Entwicklungen

Ein großer Mehrwert der erstellten Web-Anwendung ist die generische Architektur, die es ermöglicht beliebige Analysemodule in das System zu integrieren. Diesbezüglich könnten eine Reihe von spezifischen und didaktisch relevanten Netzwerkanalyse-Modulen entwickelt und in die Web-Anwendung integriert werden.

Die Web-Anwendung *Network Metric Explorer* könnte auch dahingehend erweitert werden, gemeinsam genutzte Netzwerkdatensatz Sammlungen zu unterstützen. Denkbar wäre, dass die Web-Anwendung eine Netzwerkdatensatz Sammlung eines Instituts in einer eigenen Rubrik bereitstellt. Durch die Navigationsmöglichkeiten der bestehenden Web-Anwendung können diese Netzwerkdatensätze und die Netzwerk Metriken sehr einfach eingesehen werden.

Die zur Zeit implementierte Ajax-Polling Technik könnte durch WebSockets abgelöst werden. Diese Entwicklung würde das System technisch verbessern, jedoch würde ein Benutzer nicht unmittelbar davon profitieren.

Sehr viel Potential liegt in der Darstellung von Diagrammen von Netzwerk-Metriken. Die derzeit integrierten Netzwerkanalyse-Module generieren die Plots mit GNUPlot. GNUPlot bietet in der Version 4.5 einen sogenannten HTML5 Canvas Terminal[1], wodurch sich ungeahnte Möglichkeiten in der Darstellung von Diagrammen im Web ergeben.

[1] GNUPlot HTML5 Canvas Terminal`http://www.gnuplot.info/demo_canvas/`

Kapitel 9

Zusammenfassung

In dieser Masterarbeit wurde meine Forschungstätigkeit für die Bereiche Netzwerkanalyse, Web-Engineering, Usability und EduPunk präsentiert, die schlussendlich als Vorraussetzung für die praktische Arbeit und einer erfolgreichen Umsetzung der Web-Anwendung *Network Metric Explorer* zu sehen ist. In Kapitel 2 bis 4 wurden theoretische Aspekte, die für diese Arbeit von Bedeutung sind vorgestellt. Kapitel 2 lieferte einen Überblick über die grundlegende Theorie der Netzwerkanalyse und bot zudem eine Beschreibung der SNAP-Library, die für den praktischen Teil dieser Arbeit eine zentrale Rolle für die Bereitstellung von benötigter Funktionalität in einer Web-Anwendung gespielt hat. Kapitel 3 gab einen Überblick über die Disziplin Web Engineering und behandelte Aspekte, die mit einer systematische Entwicklung von Web-Anwendungen einhergehen. Kapitel 4 lieferte einen Überblick über die Usability von Web Anwendungen. Zudem wurde Usability als ein Qualitätsmerkmal einer Web-Anwendung definiert. In Kapitel 4 wurde auch die EduPunk Bewegung behandelt, die eine DIY-Attitüde für die Erstellung von modernen Lernräumen durch den sinnvollen Einsatz von Web 2.0 Technologien lebt.

Kapitel 5 bis 8 repräsentiert den praktische Teil dieser Arbeit und behandelt alle Aspekte der Erstellung der Web-Anwendung *Network Metric Explorer*. Kapitel 5 lieferte eine Beschreibung und alle Anforderungen der Web Anwendung, die im Rahmen dieser Masterarbeit mit dem Web Framework Apache Wicket umgesetzt wurde. Das Web Framework Apache Wicket wurde anhand den Anforderungen ausgewählt. Das erstellte System wird in Kapitel 6 präsentiert und beschrieben. Die erarbeitete Architektur wird im ersten Teil in Abschnitt 6.1 dargelegt. Der zweite Teil beschreibt in Abschnitt 6.2 alle verwendeten Tools und Technologien, die für die Umsetzung der Web-Anwendung verwendet wurden. Schlussendlich wird die resultierende Lösung in Abschnitt 6.3 beschrieben. Kapitel 7 gab schließlich einen Überblick über die Ergebnisse einer informalen Usability-Evaluierung die durch eine Gruppe bestehend aus Experten, Pädagogen und Studenten anhand der erstellten Web-Anwendung durchgeführt wurde. Abschliessend bot Kapitel 8 einen Überblick von Ideen und Maßnahmen, die für eine Verbesserung des umgesetzten Systems verwendet werden können. Zusätzlich wurden zukünftige Trends wie beispielsweise der Einsatz von HTML5 oder den Einsatz von WebSockets diskutiert.

Literaturverzeichnis

Adamic, Lada [2008a]. *Network basics & some tools*. Vorlesungsskript. http://open.umich.edu/sites/default/files/1446/SI508-F08-Week2.pdf. Zugriffsdatum: 19.09.2011. (Zitiert auf der Seite 12.)

Adamic, Lada [2008b]. *Why networks are interesting to study*. Vorlesungsskript. http://open.umich.edu/sites/default/files/1446/SI508-F08-Week1.pdf. Zugriffsdatum: 19.09.2011. (Zitiert auf der Seite 14.)

Barabási, Albert-László [2003]. *Linked - how everything is connected to everything else and what it means for business, science, and everyday life.* Plume. ISBN 978-0-452-28439-5, 1-294 Seiten. (Zitiert auf der Seite 1.)

Barabási, Albert-László und Eric Bonabeau [2004]. *Skalenfreie Netze*. *Spektum der Wissenschaft*, Seiten 62–69. (Zitiert auf den Seiten v, 5, 14, 17 and 18.)

Barabási, Albert-László und Zoltan N. Oltvai [2004]. *Network biology: understanding the cell's functional organization*. *Nature Reviews Genetics*, 5(2), Seiten 101–113. ISSN 1471-0056. (Zitiert auf den Seiten v, 16, 17, 18, 24 and 25.)

Behrendt, Wernher [2004]. *Semantisches Web - Das Netz der Bedeutungen im Netz der Dokumente*. In *Web Engineering: Systematische Entwicklung von Web-Anwendungen*, Seiten 345–368. (Zitiert auf den Seiten 33 and 34.)

Berners-Lee, T., R. Fielding, und L. Masinter [2005]. *RFC 3986, Uniform Resource Identifier (URI): Generic Syntax*. Request For Comments (RFC). http://www.ietf.org/rfc/rfc3986.txt. (Zitiert auf der Seite 30.)

Berners-Lee, T., W. Hall, J. A. Hendler, K. O'Hara, N. Shadbolt, und D. J. Weitzner [2006]. *A Framework for Web Science*. *Foundations and Trends¨ in Web Science*, 1(1). (Zitiert auf der Seite 1.)

Berners-Lee, Tim [1998]. *The World Wide Web: A very short personal history*. http://www.w3.org/People/Berners-Lee/ShortHistory. Zugriffsdatum: 19.09.2011. (Zitiert auf der Seite 30.)

Berners-Lee, Tim, Robert Cailliau, und Bernd Pollermann [1992]. *World-Wide Web: The Information Universe*. *Communications of the ACM*, 37, Seiten 76–82. (Zitiert auf der Seite 30.)

Berners-Lee, Tim und Mark Fischetti [1999]. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*. 1st Auflage. Harper San Francisco. ISBN 0062515861. (Zitiert auf der Seite 30.)

Booch, Grady [2001]. *The architecture of Web applications*. http://www.ibm.com/developerworks/ibm/library/it-booch_web/. Zugriffsdatum: 16.09.2011. (Zitiert auf der Seite 40.)

Broder, A. [2000]. *Graph structure in the Web*. *Computer Networks*, 33(1-6), Seiten 309–320. ISSN 13891286. doi:10.1016/S1389-1286(00)00083-9. http://dx.doi.org/10.1016/S1389-1286(00)00083-9. (Zitiert auf den Seiten v, 22 and 24.)

Buschmann, Frank, Regine Meunier, Hans Rohnert, Peter Sommerlad, und Michael Stal [1996]. *Pattern-Oriented Software Architecture Volume 1: A System of Patterns*. 1 Auflage. Wiley. ISBN 0471958697. (Zitiert auf der Seite 65.)

Cormen, T.H., C.E. Leiserson, R.L. Rivest, und C. Stein [2007]. *Algorithmen- eine Einführung*. Oldenbourg. ISBN 9783486582628. (Zitiert auf der Seite 11.)

Dashorst, Martijn und Eelco Hillenius [2008]. *Wicket in Action*. Manning Publications Co., Greenwich, CT, USA. ISBN 1932394982, 9781932394986. (Zitiert auf den Seiten vi, 68, 71 and 74.)

Diestel, Reinhard [2010]. *Graphentheorie*. Springer Verlag. (Zitiert auf den Seiten v, 8, 10 and 11.)

Dorogovtsev, Sergey N. und J. F. F. Mendes [2004]. *The shortest path to complex networks*. *CoRR*, cond-mat/0404593. Informal publication. (Zitiert auf der Seite 22.)

Easley, David A. und Jon M. Kleinberg [2010]. *Networks, Crowds, and Markets - Reasoning About a Highly Connected World.* Cambridge University Press. ISBN 978-0-521-19533-1, I-XV, 1-727 Seiten. (Zitiert auf den Seiten v, 5, 6, 7, 9, 12, 13, 17, 18, 21 and 22.)

Ebner, M., N. Scerbakov, P. Tsang, und A. Holzinger [2011]. *EduPunks and Learning Management Systemsâ Conflict or Chance?* In *Pro-ceedings of International Conference on Hybrid Learning IHCL 2011*, Seiten 224–238. http://www.scribd.com/doc/61914608/EduPunks-and-Learning-Management-Systems-%E2%80%93-Conflict-or-Chance. (Zitiert auf der Seite 50.)

Ebner, Martin [2008]. *Why we need EduPunk*. *JOURNAL OF SOCIAL INFORMATICS*, 9, Seiten 1–9. (Zitiert auf der Seite 50.)

Eichinger, Christian [2004]. *Architektur von Web-Anwendungen*. In *Web Engineering: Systematische Entwicklung von Web-Anwendungen*, Seiten 77–100. (Zitiert auf den Seiten v, 39, 40, 41, 42, 43 and 56.)

Fielding, R., J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, und T. Berners-Lee [1999]. *RFC 2616: Hypertext transfer protocol – HTTP/1.1*. http://www.ietf.org/rfc/rfc2616.txt. (Zitiert auf der Seite 30.)

Fielding, Roy T. [2000]. *Architectural Styles and the Design of Network-based Software Architectures*. Dissertation, University of California, Irvine. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm. (Zitiert auf der Seite 40.)

Fluckiger, M.D. und M. Richter [2007]. *Usability Engineering Kompakt: Benutzbare Software Gezielt Entwickeln*. IT Kompakt Series, Spektrum Akademischer Verlag. ISBN 9783827418371. (Zitiert auf den Seiten 48 and 49.)

Förther, R., C.E. Menzel, und O. Siefart [2009]. *Wicket: Komponentenbasierte Webanwendungen in Java*. Dpunkt-Verl. ISBN 9783898645690. (Zitiert auf den Seiten vi, 42, 62, 68, 69, 70, 72, 73 and 76.)

Gamma, Erich, Richard Helm, Ralph E. Johnson, und John Vlissides [1995]. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA. ISBN 978-0-201-63361-0. (Zitiert auf den Seiten 65 and 74.)

Hitz, Martin und Gerhard Leitner [2003]. *Usability von Web-Anwendungen*. In *Web Engineering: Systematische Entwicklung von Web-Anwendungen*, Seiten 265–296. (Zitiert auf den Seiten 47, 48 and 50.)

Hoare, C. A. R. [1981]. *The Emperor's Old Clothes. Commun. ACM*, 24(2), Seiten 75–83. (Zitiert auf der Seite 29.)

Holzinger, Andreas [2005]. *Usability engineering methods for software developers. Communications of the ACM*, 48(1), Seiten 71–74. http://portal.acm.org/citation.cfm?id=1039539.1039541. (Zitiert auf den Seiten 48, 49 and 50.)

Hunt, Andrew und David Thomas [1999]. *The pragmatic programmer: from journeyman to master*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. ISBN 0-201-61622-X. (Zitiert auf der Seite 42.)

Jablonski, Stefan, Ilia Petrov, und Christian Meiler [2002]. *Ein konzeptionelles Architekturrahmenswerk für Web-Anwendungen*. In *Promise 02*, Seiten 175–187. (Zitiert auf den Seiten v and 42.)

Jacobson, Ivar, Grady Booch, und James Rumbaugh [1999]. *The unified software development process*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. ISBN 0-201-57169-2. (Zitiert auf den Seiten v and 56.)

Kappel, Gerti, Birgit Pröll, Siegfried Reich, und Werner Retschitzegger [2004]. *Web Engineering - Die Disziplin zur systematischen Entwicklung von Web-Anwendungen*. In *Web Engineering: Systematische Entwicklung von Web-Anwendungen*, Seiten 1–28. (Zitiert auf den Seiten v, 29, 31, 35, 36, 37, 38 and 39.)

Khazaeli, C.D. [2005]. *Systemisches Design: Intelligente Oberflächen für Information und Interaktion*. rororo Taschenbücher, Rowohlt Taschenbuch Verlag. ISBN 9783499600784. (Zitiert auf der Seite 49.)

Kruchten, Phillippe [1995]. *Architectural Blueprints – The "4+1" View Model of Software Architecture*. *IEEE Software*, 12(6). (Zitiert auf der Seite 40.)

Llinás, Rodolfo R [2003]. *The contribution of Santiago Ramón y Cajal to functional neuroscience*. *Nature Reviews Neuroscience*, 4(1), Seiten 77–80. http://www.ncbi.nlm.nih.gov/pubmed/12511864. (Zitiert auf der Seite 6.)

Massol, Vincent und Jason van Zyl [2006]. *Better Builds with Maven*. MaestroDev. http://www.maestrodev.com/better-build-mavenl. Zugriffsdatum: 14.11.2011. (Zitiert auf den Seiten 75 and 76.)

Mosmann, M. [2009]. *Praxisbuch Wicket: Professionelle Web-2.0-anwendungen entwickeln*. Hanser Fachbuchverlag. ISBN 9783446419094. (Zitiert auf den Seiten 62, 68, 69, 71 and 72.)

Murugesan, San, Yogesh Deshpande, Steve Hansen, und Athula Ginige [2001]. *Web Engineering : A New Discipline for Development of Web-based Systems*. *Web Engineering*, 2016(0), Seiten 3–13. http://www.springerlink.com/index/8RY1C125Q4UJT3B9.pdf. (Zitiert auf der Seite 35.)

Newman, Mark E. J. [2003]. *The structure and function of complex networks*. *SIAM REVIEW*, 45, Seiten 167–256. (Zitiert auf den Seiten v, 5, 7, 10, 14, 15, 22 and 23.)

Newman, Mark E. J. [2010]. *Networks: An Introduction*. Oxford University Press. ISBN 978-0-19-920665-0. (Zitiert auf den Seiten v, 1, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 19, 20, 21 and 23.)

Nielsen, Jakob [1993]. *Usability Engineering*. Morgan Kaufmann. (Zitiert auf den Seiten 48 and 49.)

O'Reilly, Tim [2005]. *What Is Web 2.0? Design Patterns and Business Models for the Next Generation of Software*. http://oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html. (Zitiert auf den Seiten 32, 33 and 37.)

Ort, Ed und Bhakti Mehta [2003]. *Java Architecture for XML Binding (JAXB)*. http://www.oracle.com/technetwork/articles/javase/index-140168.html. Zugriffsdatum: 14.11.2011. (Zitiert auf den Seiten vi, 74 and 75.)

Rossi, Gustavo, Oscar Pastor, Daniel Schwabe, und Luis Olsina [2007]. *Web Engineering: Modelling and Implementing Web Applications (Human-Computer Interaction Series)*. 1 Auflage. ISBN 184628922X, 9781846289224. (Zitiert auf den Seiten 31, 32, 33, 34, 35 and 36.)

Rowell, Laurie [2008]. *"Edupunk" rocks the (virtual) house*. http://elearnmag.acm.org/archive.cfm?aid=1454090. Zugriffsdatum: 02.11.2011. (Zitiert auf der Seite 50.)

Scholz, Matthias [2008]. *Komplexe Netzwerke: Netzwerkstrukturen (community structure)*. Vorlesungsskript. http://www.network-science.org/komplexeNetzwerke_VL06.pdf. Zugriffsdatum: 19.09.2011. (Zitiert auf der Seite 15.)

Shan, Tony C. und Winnie W. Hua [2006]. *Taxonomy of Java Web Application Frameworks*. In *Proceedings of the IEEE International Conference on e-Business Engineering*, Seiten 378–385. ICEBE '06, IEEE Computer Society, Washington, DC, USA. ISBN 0-7695-2645-4. doi:http://dx.doi.org/10.1109/ICEBE.2006.98. http://dx.doi.org/10.1109/ICEBE.2006.98. (Zitiert auf den Seiten 41, 44 and 45.)

SNAP [2011a]. *SNAP documentation*. http://snap.stanford.edu/snap/doc.html. Zugriffsdatum: 16.09.2011. (Zitiert auf den Seiten 25, 26, 27 and 28.)

SNAP [2011b]. *SNAP network analysis library*. http://snap.stanford.edu/snap/index.html. Zugriffsdatum: 16.09.2011. (Zitiert auf den Seiten 24 and 28.)

SNAP [2011c]. *Stanford Large Network Dataset Collection*. http://snap.stanford.edu/data/index.html. Zugriffsdatum: 16.09.2011. (Zitiert auf den Seiten 28, 57 and 77.)

Sofer, Susanne und Oliver Weinkamp [2005]. *Infografik für die Ausstellung Netzsuche in Bozen*. http://pro.unibz.it/projects/netzwerke/doku/infografik/infografik_web_070205.pdf. Zugriffsdatum: 26.09.2011. (Zitiert auf den Seiten v and 19.)

Strickland, Marta [2007]. *The Evolution of Web 3.0*. http://www.slideshare.net/mstrickland/the-evolution-of-web-30. Zugriffsdatum: 19.10.2011. (Zitiert auf den Seiten 32 and 33.)

Suh, W. [2005]. *Web engineering: principles and techniques*. ITPro collection, Idea Group Pub. ISBN 9781591404330. (Zitiert auf den Seiten v, 36 and 37.)

Sun, Microsystems [2002]. *Java BluePrints Model-View-Controller*. http://java.sun.com/blueprints/patterns/MVC-detailed.html. Zugriffsdatum: 16.09.2011. (Zitiert auf den Seiten v, 42, 43 and 44.)

Wähner, Kai [2011a]. *Webframeworks – Eine Kategorisierung*. *Java Magazin*, 1. (Zitiert auf den Seiten v, 45, 46, 60 and 61.)

Wähner, Kai [2011b]. *Welche Webframeworks für welche Projekte - Einsatzgebiete von Webframeworks*. *Java Magazin*, 2. (Zitiert auf den Seiten v, 45, 46, 60 and 67.)

Watts, Duncan J. [2004]. *Six Degrees: The Science of a Connected Age*. W. W. Norton & Company. ISBN 0393325423. (Zitiert auf den Seiten 1 and 5.)

Weßendorf, Matthias [2010]. *WebSockets im Überblick – Push me to the Limit*. *Java Magazin*, 12. (Zitiert auf den Seiten vi, 90, 91 and 102.)

Wicket [2011]. *Apache Wicket*. http://wicket.apache.org/. Zugriffsdatum: 16.09.2011. (Zitiert auf den Seiten 53 and 59.)

Wilde, Erik und Martin Gaedke [2008]. *Web Engineering Revisited*. In *Proceedings of the 2008 British Computer Society (BCS) Conference on Visions of Computer Science*. London, UK. (Zitiert auf den Seiten 33 and 36.)

Young, Jeffrey R. [2008]. *Frustrated With Corporate Course-Management Systems, Some Professors Go "Edupunk"*. http://chronicle.com/blogs/wiredcampus/frustrated-with-corporate-course-management-systems-some-professors-go-edupunk/3977. Zugriffsdatum: 02.11.2011. (Zitiert auf der Seite 50.)