Gottfried Wilhelm
Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Praktische Informatik
Fachgebiet Software Engineering

Visualisierung der Ausbreitung von Informationen in einem Digitalen Sozialen Netzwerk

Bachelorarbeit

im Studiengang Informatik

von

André Schnabel

Prüfer: Prof. Dr. Kurt Schneider
Zweitprüfer: Prof. Dr. Jörg Hähner
Betreuer: Leif Singer

Hannover, 04.02.2011

Zusammenfassung

Moderne digitale soziale Netzwerke bieten Mechanismen zur Unterstützung der Ausbreitung von Informationen. Der bekannteste Vertreter ist hier der Empfehlungsmechanismus. Über das Weiterempfehlen von Informationen gelangt diese an bisher nicht adressierte Empfänger. Gerade in Digitalen Sozialen Netzwerken speziell für die Verwendung in Unternehmen ist ein Informationsfluss ohne jegliche Art von Reibungsverlusten essentiell.

Zur Evaluation der Effektivität dieser Mechanismen sowie einem besseren Verständnis der Informationsverbreitung in einem Digitalen Sozialen Netzwerk bieten sich Visualisierungen an. In dieser Arbeit werden relevante Phänomene mit Visualisierungen auf Basis einer Literaturrecherche graphisch aufgezeigt. Die implementierten Visualisierungen lassen sich über eine klar definierte Schnittstelle an beliebige Digitale Soziale Netzwerke anbinden. Die vorliegende Arbeit enthält bereits Anbindungen für das am Fachgebiet Software Engineering der Leibniz Universität Hannover entwickelte Digitale Soziale Netzwerk “Hallway” sowie den beliebten Mikroblogging-Dienst Twitter.

Abstract

Modern social networking sites provide mechanisms supporting information diffusion. The most well known is the ability to recommend a unit of information to one's contacts. This allows information to arrive at recipients that weren't directly addressed beforehand. Especially in social networking sites tailored for company usage the need for information distribution without loss due to some sort of friction is essential.

To evaluate the effectiveness of those mechanisms and to improve the comprehension of information distribution itself in a social networking site, the use of visualizations seem like a good fit. Phenomena relevant for this purpose are emphasized in visualizations, which are supported by literature research. The implemented visualizations can be tied to different digital social networks using interfaces. This work already provides bindings to Twitter and another social networking site called Hallway, which was developed at the Software Engineering Group of Leibniz University Hannover.

Danke

Allen Personen, die mir direkt oder indirekt bei der Erstellung dieser Arbeit und dem zugehörigen Stück Software geholfen haben, sei hiermit mein Dank ausgesprochen.

Ganz besonders möchte ich mich bei Herrn Leif Singer für die hervorragende Betreuung bedanken. Dass er sich die Zeit nahm auch kleine Fragen zwischen den Treffen zügig zu beantworten, erleichterte mir meine Arbeit enorm.

Abkürzungsverzeichnis

API Application Programming Interface (Programmierschnittstelle)
AWT Abstract Window Toolkit (GUI-Bibliothek)
CSV Comma Separated Values*
DSN Digitales Soziales Netzwerk*
FLOW Forschungsprojekt zu Informationsflüssen (keine Abkürzung)
gdw. genau dann, wenn
HTML Hypertext Markup Language (Auszeichnungsformat)
HTTP Hypertext Transport Protocol (Anwendungsschicht-Protokoll)
iframe Inlineframe*
IP Internet Protocol (Netzwerkschicht-Protokoll)
JFC Java Foundation Classes (API-Sammlung, Teil des JRE)
JPA Java Persistence API (API für objekt-relationale Abbildung)
JRE Java Runtime Environment (Laufzeitumgebung von Java)
JVM Java Virtual Machine (Teil des JRE)
MVC Model View Controller*
OpenGL Open Graphics Library (API für Computergrafik)
Play-App Play-Application*
UML Unified Modeling Language (graphische Modellierungssprache)
URI Uniform Resource Identifier (Ressourcen-Identifikator)
URL Uniform Resource Locator (URI mit zusätzlicher Lokalisierung)
SQL Structured Query Language (Anfragesprache)
SVG Scalable Vector Graphics (Auszeichnungsformat)

*mit Definition bzw. Erklärung im Einführungskapitel.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Das am Fachgebiet Software Engineering der Leibniz Universität Hannover entwickelte “Hallway” und viele weitere aktuelle Digitale Soziale Netzwerke (DSN) erlauben es, dass Nutzer Informationen, welche sie empfangen haben, an andere Nutzer weiterleiten. Dadurch können Informationen auch an nicht vorher adressierte Personen gelangen. Dieser Empfehlungsmechanismus wird oft über spezielle Kennzeichnung einer Information durch die Leser realisiert. Dies hilft bei der Entdeckung von vorher unbekannten Informationsquellen und der Verbreitung von Informationen in Gruppen, welche zuvor für den Sender nicht als Interessenten erkennbar waren.

Neben den genannten Vorteilen ergeben sich aber auch Nachteile durch die neuartigen Empfehlungsmechanismen: Vorher konnten die Ausbreitungswege einer Information lediglich anhand der Struktur des Digitalen Sozialen Netzwerks (Beziehungen von Personen untereinander und zu Artefakten) bestimmt werden. Durch Weiterempfehlung von Informationen können diese aber nun durch Kanten verlaufen, die sonst nicht zur Weiterleitung dieser Information verwendet werden würden. Dadurch ist auch die Menge der empfangenden Knoten schwieriger zu erkennen. Beides führt zu geringerem Verständnis und damit auch schlechterer Steuerbarkeit der Informationsausbreitung im DSN.

Um diesem Effekt entgegenzuwirken, müssen Maßnahmen ergriffen werden. Diese sollen helfen die neuen Verbreitungswege zu überblicken. Naheliegend ist es hierfür die Ausbreitung von Informationen im DSN zu visualisieren. Nach [Chen ’10] ist nämlich das Erlangen von Einsichten die oberste Maxime, der “Holy Grail” der Informationsvisualisierung. Das schließt ein vertieftes Verständnis mit ein und ist somit gut geeignet, um die genannten nachteiligen Effekte von Empfehlungsmechanismen abzuschwächen.

Dass die Erstellung von Visualisierungen eine zweckmäßige und nicht unübliche Herangehensweise bei der Analyse der Informationsausbreitung in DSN mit Empfehlungsmechanismus ist, wird ebenso in der Literatur bestätigt. Mindestens Kurvendiagramme lassen sich in vielen Artikeln zum Thema finden. Sie veranschaulichen dort die Zusammenhänge der unterschiedlichen Faktoren in der Ausbreitung der Informationen.

1.2 Problemstellung

Das Ziel dieser Arbeit ist die Entwicklung von Visualisierungen für die Informationsverbreitung in DSN. Zu diesem Zweck wird mit der Festlegung von Phänomenen mit Relevanz für die Nachrichtenausbreitung begonnen. Daraufhin werden Visualisierungen zur Hervorhebung dieser Phänomene festgelegt. Es werden für die Visualisierungen drei Beispielszenarien entwickelt. Die für diese Szenarien zu erwartenden Visualisierungsergebnisse werden als Mockups skizziert. Danach werden die Visualisierungen umgesetzt und nach Fertigstellung werden die Ergebnisse für die Beispielszenarien mit den Mockups verglichen.

1.3 Gliederung

Der Aufbau der Arbeit ergibt sich aus der Chronologie der Problemstellung. Daher beginnt das zweite Kapitel mit der Klärung der begrifflichen Grundlagen der Arbeit.

Sind die fachlichen Grundlagen geklärt, folgt im dritten Kapitel eine Beschreibung der Phänomene, welche in den Beispielszenarien vorhanden und in den Visualisierungen gut erkennbar sein sollen. Dieses Kapitel nimmt also eine Konkretisierung der Anforderungen an die im Anschluss zu erstellenden Visualisierungen vor.

Diese sind dann Thema des vierten Kapitel. Es werden abschnittsweise alle Visualisierungen beschrieben. Die Beschreibung orientiert sich an der Funktionsweise: Aus welchen Eingabedaten wird nach welchem Verfahren ein Ausgabebild erzeugt. All dies bleibt auf der konzeptionellen Ebene. Die Umsetzung bedarf aber eines Sprungs auf die Software-Ebene. Ganz im Sinne des Test Driven Development sollten jedoch vor der Implementierung Tests erarbeitet werden.

Deshalb geht es im fünften Kapitel um die Erzeugung der beiden Beispielszenarien. Für jedes dieser Beispielszenarien und jede der Visualisierungen wird ein Mockup skizziert, welches das erwartete Visualisierungsergebnis zeigt. Die Beispielszenarien sind also als Eingabedaten des Black-Box-Tests zu verstehen und die Mockups als grobe Soll-Werte.

Nachdem nun vollständig geklärt worden ist "*was* gemacht werden soll" wird es vor der Implementierung aber noch darum gehen, *wie* die Lösung grob aufgebaut sein soll. Daher beschäftigt sich das sechste Kapitel mit dem Entwurf einer Visualisierungsanwendung. Es plant bereits den groben Aufbau der Anwendung mit seinen einzelnen Komponenten.

Nun kann im siebten Kapitel mit der Umsetzung der Visualisierungen im Rahmen einer eigenständigen Visualisierungsanwendung begonnen werden. Notwendige Änderungen an den Controllern und Views von Hallway werden besprochen. Erweiterungen am Modell werden ebenfalls erläutert. Die konkrete client-seitige Umsetzung der Visualisierungen in JavaScript ist Inhalt der letzten Abschnitte dieses Kapitels. Dies schließt auch den letzten Schritt der konstruktiven Teile der Problemstellung ab.

Darauf folgt im achten Kapitel die notwendige Auswertung der im vorherigen Kapitel implementierten Software. Es werden dafür die Ergebnisvisualisierungen (Ist-Werte) der Visualisierungssoftware (Prüfling) für die Beispielszenarien (Eingabe-Werte) mit den Mockups (Soll-Werte) aus dem fünften Kapitel kurz und knapp verglichen.

Das nachfolgende neunte Kapitel enthält eine kurze erläuternde Auflistung einiger verwandter Arbeiten.

Das letzte und zehnte Kapitel beginnt mit einer kritischen Würdigung dieser Arbeit. Darauf wird ein Ausblick auf mögliche Verbesserungen der Visualisierungen und Erweiterungen gegeben. Dinge, die aus Platz- oder Zeitgründen ausgelassen werden mussten, werden genannt. Abschließend erfolgt eine Zusammenfassung.

2 Grundkonzepte

Es folgt eine Auflistung der in dieser Arbeit zentralen Fachbegriffe.

Definition 2.0.1 (Information)**.** Eine Information ist in dieser Arbeit eine Abfolge von Zeichen. Es wird nicht zwischen dem Inhalt einer Nachricht[1] und Informationen unterschieden, weshalb beide Begriffe nachfolgend synonym verwendet werden.

2.1 Digitale Soziale Netzwerke

Definition 2.1.1 (DSN)**.** Ein DSN ist nach [Boyd und Ellison '07] ein web-basierter Dienst, welcher Einzelpersonen folgendes erlaubt:

1. Das Anlegen eines öffentlichen oder semi-öffentlichen Profils in einem umgrenzten System.
2. Die Angabe einer Liste von anderen Nutzern, mit welchen eine Assoziation besteht.
3. Das Betrachten und Durchlaufen ihrer Verbindungsliste und der von anderen Personen innerhalb dieses Systems.
4. Das Veröffentlichen von Nachrichten, welche von den assoziierten Nutzern gelesen werden können.
5. Das Weiterempfehlen von Nachrichten an die assoziierten Nutzer.
6. Die Kommentierung von bestehenden Nachrichten.

Die letzten drei Funktionen, welche in der vorangehenden Definition von einem DSN gefordert werden, stammen dabei nicht von [Boyd und Ellison '07], sondern wurden für diese Arbeit zusätzlich gefordert. Dadurch soll das Vorhandensein von Informationsausbreitung in einem DSN sichergestellt werden. Definition 2.1.1 wird zum Nachschlagen an dieser Stelle vollständig angegeben, obwohl für das Verständnis der letzten drei Funktionen die erst später genannten Definitionen 2.1.4 und 2.1.5 erforderlich sind.

Definition 2.1.2 (Verbindungsmodelle)**.** Es gibt in DSN zwei Verbindungsmodelle:

- **Das asymmetrische Verbindungsmodell** (auch *nicht-reziprok* genannt). Dies lässt sich durch einen gerichteten Graphen (Digraph) modellieren (siehe auch 2.1.3). Zwischen zwei Nutzern A und B kann es keine Verbindung, eine von A nach B, eine von B nach A sowie eine in beide Richtungen geben.

[1] Wenn es vom Kontext her klar ist, wird zur Erhöhung der Lesbarkeit oft der "Inhalt einer Nachricht" im Text abgekürzt zu "Nachricht".

- **Das symmetrische Verbindungsmodell** (auch *reziprok* genannt). Hier ist ein ungerichteter Graph zur Modellierung nötig bzw. ein gerichteter Graph mit der Zusatzbedingung der Symmetrie. Zwei Nutzer sind entweder miteinander verbunden oder sie sind es nicht. Eine feinere Abstufung wie im asymmetrischen Modell gibt es nicht.

Stark abgewandelt nach [Peters '10].

In dieser Arbeit wird als Verbindungsmodell das asymmetrische verwendet, da die Verbindungszustände des symmetrischen Modells eine Teilmenge der Zustände des asymmetrischen sind. Dabei ist sich der Autor der Tatsache bewusst, dass dieses Verbindungsmodell in der Praxis wie bspw. beim DSN Twitter[2] dazu führen kann, dass das DSN eher die Rolle einer Art Nachrichtenmedium übernimmt und nicht mehr wie ein klassisches soziales Netzwerk arbeitet. Vgl. [Kwak et al. '10].

Definition 2.1.3 (Digraph eines DSN)**.** Ein DSN mit asymmetrischem Verbindungsmodell kann als gerichteter Graph $G = (V, E)$ modelliert werden. Die Knotenmenge V entspricht dabei der Nutzerliste aus 2.1.1. Die Kantenmenge E entspricht den gerichteten Assoziationen des asymmetrischen Verbindungsmodells.

Definition 2.1.4 (Folgerelation)**.** Die Kantenrelation E des Digraph $G = (V, E)$ eines DSN (2.1.3) wird nachfolgend auch als Folgerelation bezeichnet. Denn es gilt $(a, b) \in E$ für zwei Nutzer $a, b \in V$ gdw. a dem Nutzer b folgt.

Definition 2.1.5 (Empfehlungsrelation)**.** Die Empfehlungsrelation $R \subseteq V \times M$ sei definiert durch $(a, b) \in R$ gdw. Nutzer a die Nachricht b weiterempfohlen hat, wobei $M \subset \mathbb{N}$ die Menge der Identifikationsnummern von Nachrichten[3] ist und $a \in V$ sowie $b \in M$ gelten.

Definition 2.1.6 (Phänomen)**.** Ein Phänomen ist hier definiert als etwas bei voller Einsicht in die Daten des DSN Beobachtbares, das durch Weiterempfehlung von Nachrichten hervorgerufen wird.

Definition 2.1.7 (Skalenfreiheit)**.** Ein Netzwerk wird skalenfrei oder skaleninvariant genannt, falls der Anteil $P(k)$ an Knoten mit k Verbindungen zu anderen Knoten k proportional zu $k^{-\gamma}$ ist, kurz: $P(k) \propto k^{-\gamma}$.

Definition 2.1.8 (90-9-1 Theorie)**.** Es wird vermutet, dass sich in DSN die Beteiligung grob verteilt auf

- etwa 90% reinen Konsum (sogenannte Lurker),
- etwa 9% insignifikante Beiträge und
- etwa 1% signifikante Beiträge[4].

Definitionen 2.1.8 und 2.1.7 stammen aus [WP:9091] bzw. [WP:ScaleFree].

[2] https://twitter.com

[3] Wir gehen davon aus, dass es in der Praxis — also in der Implementierung — eine bijektive Abbildung zwischen den Identifikationsnummern und den zugehörigen Nachrichten gibt.

[4] Der überaus geringe Anteil an wichtigen Beiträgen in solchen Systemen prägte auch die alternative Bezeichnung "1% Gesetz", da nur eine sehr kleine Minderheit der Nutzer tatsächlich an der Erstellung von Inhalten im System aktiv beteiligt ist.

2.1.1 Informationsfluss durch Weiterempfehlung

In dieser Arbeit soll es insbesondere um den Informationsfluss in einem DSN gehen. Bildlich kann man sich zwar das “Fließen” einer Nachricht von einem Knoten zu einem anderen vorstellen. Es erfolgt entgegengesetzt zur Richtung des Pfeils, welcher die Folge-Beziehung visualisiert. Tatsächlich ist aber von der Umsetzung her jede Nachricht persistent bis zu ihrer Löschung. Sie wird in einer Tabellenzeile einer Datenbank durch die DSN-Software gespeichert. Daher findet das eigentliche “Fließen” der Information erst später statt. Sie fließt genau dann, wenn zu einem späteren Zeitpunkt der reale Nutzer diese Nachricht durch Verwendung des DSN bemerkt und liest. Die bijektive Zuordnung vom realen Nutzen zum Nutzer-Knoten im DSN wird durch Authentifizierung hergestellt. Es ist also für das Bestimmen der wirklichen Zeitdauer einer Nachricht von einem Knoten zu einem anderen Knoten nötig, dass das Ereignis des Lesens dieser Nachricht als Information aus den Protokollen der DSN-Software hervorgeht.

2.1.2 Empfehlungen und Kommentare

Wird eine Nachricht von einem Nutzer durch spezielle Markierung weiterempfohlen, so kann diese zu all seinen Verfolgern weiter fließen. D.h. zusätzlich zu den Verfolgern des Autors der ursprünglichen Nachricht werden ebenfalls die Verfolger eines weiterempfehlenden Empfängers erreicht. So wirkt dieser Empfänger, der weiterempfiehlt, für dessen Verfolger als Informationsquelle. Entsprechend lässt sich das Weiterempfehlen rekursiv fortführen bis theoretisch jeder Knoten des DSN-Graphen die Nachricht erhalten hat.

Das Weiterleiten (Forwarding) von E-Mails kann ähnliches erreichen. Jedoch gibt es den großen Unterschied, dass bei Empfehlungsmechanismen die Weiterleitung der Nachrichten mit geringem Aufwand (meist ein Klick) und automatisiert durch das System erfolgt, während beim Weiterleitung von E-Mails durch den Nutzer eine explizite Nennung aller Adressaten erforderlich ist.

Eine andere Möglichkeit auf eine bestehende Nachricht zu reagieren ist, diese zu kommentieren. Ein Kommentar ist eine Nachricht, welche eine Referenz auf eine andere Nachricht enthält. Diese Nachricht kann wiederum ein Kommentar sein oder eine eigenständige Nachricht. Da Kommentare prinzipiell Nachrichten sind, d.h. sie genau so an die Verfolger ihres Autors verteilt werden, können durch sie ebenfalls Nachrichten verbreitet werden. Denn die Nachricht, auf welche sich der Kommentar bezieht, muss nicht zwangsläufig von einem Autor stammen, dem die Verfolger des Kommentators folgen. Der Effekt ist also ähnlich zu einer Weiterempfehlung mit dem Unterschied, dass Weiterempfehlungen einer Nachricht lediglich eine gewisse Relevanz zusprechen, wohingegen ein Kommentar eine differenzierte Beurteilung einer Nachricht erlaubt. Zusätzlich können Kommentare zur Kommunikation verwendet werden, indem sie als Antwort auf die referenzierte Nachricht betrachtet werden.

2.2 Informationsvisualisierung

Zwar kann Visualisierung am Computer prinzipiell jede Form von digitalen Bildern und damit Pixelmengen erzeugen, in der Praxis jedoch haben sich bestimmte Layouts und Ansätze besonders bewährt. Typische Vertreter sind auszugsweise ein Kartenuntergrund bei ortsbezogenen Informationen, Diagramme für absolute und relative quantitative Daten und Graphen für Netzwerke. Die bestehenden Notationen, auf welchen die später zu entwickelnden Visualisierungen basieren, werden nun aufgelistet.

Definition 2.2.1 (Punktdiagramm)**.** Zwei aufeinander normal (senkrecht) stehende Achsen spannen eine Fläche auf, in welcher die Wertepaare als Punkte (Kreuze, Kreise) eingetragen werden. Allgemeiner und in der Statistik wird dies auch Streudiagramm genannt.

Definition 2.2.2 (Liniendiagramm)**.** Basiert auf dem Punktdiagramm. Im Unterschied zum Punktdiagramm werden im Liniendiagramm die Punkte miteinander durch Linien (Geraden, Kurven) verbunden. Ist die Fläche zwischen Achse und Linie ausgefüllt, spricht man von einem Flächendiagramm.

Die Definition 2.2.1 und 2.2.2 wurden [WP:Diagramm] entnommen.

Definition 2.2.3 (Force Directed Layout)**.** Ein verbreiteter Ansatz zur Anordnung der Knoten beim Zeichnen von gerichteten und ungerichteten Graphen. Die Knoten werden wie Punktladungen mit einer gleichnamigen nichtverschwindenden elektrischen Ladung $Q \neq 0$ belegt. Die Kanten zwischen diesen Knoten wirken als eine Art Gummi oder Federn, welche die Knoten zusammenhalten. Durch die elektrostatischen Kräfte nach dem coulombschen Gesetz $F = \frac{Q \cdot Q'}{4\pi\epsilon_0 \cdot r^2}$ stoßen sich die Knoten ab. Die Anordnung der Ladungen im Kräftegleichgewicht liefert dann ein verglichen mit dem Rechenaufwand übersichtliches Layout.

Definition 2.2.4 (Treemap)**.** In einer Treemap wird eine rechteckige Grundfläche, die quantitativ die Gesamtheit einer Größe darstellt, rekursiv in Unterrechtecke unterteilt. Diese Unterrechtecke erhalten eine Fläche entsprechend der Anteile der von ihr repräsentierten Werte zur gesamten Größe. So wird sukzessive weiter vorgegangen. Besonders gut lassen sich mit diesen verschachtelten Rechtecken Hierarchien darstellen.[5].

2.3 Technisches

Es folgen einige technische Definitionen insbesondere für Kapitel 7.

Definition 2.3.1 (Comma-Separated Values)**.** CSV bezeichnet ein einfaches Dateiformat zur tabellenartigen Strukturierung von Daten. Spalten werden durch ein Trennzeichen (oft Komma, daher der Name) getrennt und Zeilen durch einen Zeilenumbruch (newline). Häufig gibt die erste Zeile die Spaltenbezeichnungen an.

[5] Ein interessantes Beispiel für eine Treemap ist eine im Web erhältliche interaktive Visualisierung des Bundeshaushalts der BRD. Sie ist erreichbar auf http://bund.offenerhaushalt.de

Definition 2.3.2 (Ereignis)**.** Ein Ereignis in einem DSN bezeichnet hier ein 4-Tupel (u, a, p, t) mit folgenden Bestandteilen:

- Der Nutzer u, der das Ereignis verursacht.
- Aktion a, welche beschreibt, was vom Subjekt u in diesem Ereignis getan worden ist.
- Parameter p, welcher zusätzliche Informationen zur Aktion a gibt.
- Zeitpunkt t, an welchem das Ereignis geschehen ist.

Definition 2.3.3 (Nachricht)**.** Eine Nachricht bezeichnet in dieser Arbeit ein 4-Tupel (i, u, c, r) mit folgenden Bestandteilen:

- Eineindeutige Identifikationsnummer $i \in \mathbb{N}$ der Nachricht.
- Name u des Autoren, welcher die Nachricht verfasst hat.
- Nachrichteninhalt c. Vgl. 2.0.1.
- Referenz $r \in \mathbb{N}$ auf eine andere Nachricht. Sie ist 0, außer bei Kommentaren.

Bei Ereignissen und Nachrichten sind alle Komponenten k mit $k \notin \mathbb{N}$ vom Datentyp her Zeichenketten (String). Die restlichen Komponenten sind Ganzzahlen (Integer).

Definition 2.3.4 (Model View Controller)**.** Das MVC-Muster besteht aus drei Teilen:

1. Das *Model* enthält die Daten.
2. Der *Controller* fängt Benutzerabfragen ab und ändert das *Model* entsprechend. Darauf teilt er dem *View* mit, dass sich etwas verändert hat.
3. Das *View* stellt die Daten dem Benutzer dar.

Definition 2.3.5 (Inlineframe)**.** Ein Inlineframe ist ein Frame (d.h. ein Rahmen für HTML-Dokumente), welches an belieber Stelle in ein Hauptdokument gesetzt werden kann. Es ermöglicht die Einbettung einer fremden Seite.

Definition 2.3.6 (Play-Application)**.** Eine Play-Application ist eine Webanwendung, welche zur Kommunikation mit dem Nutzer das Play-Framework[6] verwendet. Play-Applications haben eine vorgegebene Architektur.

[6] http://www.playframework.org/

3 Phänomene

Als ersten Schritt auf dem Weg zu fertigen Visualisierungen werden nun die Phänomene festgelegt. Man soll sie anhand der Visualisierungen besonders schnell erkennen können. Diese Phänomene verstärken, sofern wahrgenommen, das Verständnis des Informationsflusses in einem DSN.

In der Literaturrecherche ergab sich, dass die meiste Literatur zur Visualisierung von Zusammenhängen in DSN sich mit der Struktur des DSN selbst befasst: Es werden dort nämlich Methoden zur Analyse von Beziehungen in sozialen Netzwerken entwickelt und angewandt[1]. Da hier aber der Fokus auf der Informationsausbreitung mithilfe eines Empfehlungsmechanismus liegt, waren diese Artikel für diese Arbeit nicht weiter von Nutzen. Es wurden sieben wissenschaftliche Artikel direkt zum Thema gefunden: [Suh et al. '10], [Romero et al. '10], [Ratkiewicz et al. '10], [Magnani et al. '10], [Lee et al. '10], [Kwak et al. '10] sowie noch [Ye und Wu '10]. Zusätzlich beinhaltet der Foliensatz [Zarrella '09] einige Phänomene und Visualisierungen. Weitere Visualisierungsideen wurden im Web in [Sysomos '10] und [Kim '10] gefunden.

Die jetzt folgenden Phänomene stellen eine Auswahl aus einer längeren Liste von gefundenen Phänomenen dar. Die Auswahl traf der Autor in Absprache mit seinem Betreuer. Der Betreuer unterstützte durch Hinweise auf Artikel ebenfalls die Literaturrecherche. Ansonsten wurde in der Recherche die Literatur zum Thema mithilfe der Suchmaschine "Google Scholar"[2] gefunden.

3.1 Einfluss von Autor und Nachricht auf die Weiterempfehlung

Das Ausmaß an zusätzlicher Verbreitung einer Nachricht durch Weiterempfehlung lässt sich an drei Größen messen:

1. der Gesamtzahl an Weiterempfehlungen,
2. der größten Weglänge der Weiterempfehlung im Graph des DSN (Tiefe),
3. dem Verhältnis der Anzahl der Weiterempfehlenden zur Anzahl der Leser der Nachricht.

[1] Solche Arbeiten fallen in das Feld der *Social Network Analysis* (SNA). Die Ergebnisse der SNA finden vor allem in der Soziologie Verwendung. Das Fundament für SNA bietet jedoch die Netzwerkforschung, welche ein Teil der mathematischen Graphentheorie ist.

[2] http://scholar.google.de/

Diese sind neben dem Inhalt der Nachricht abhängig von einer Reihe von Eigenschaften der Nachricht und ihres Autors. Konkret sind die Autoreneigenschaften die Anzahl der gefolgten Nutzer, folgenden Nutzer sowie das Alters des Nutzerkontos. Die Nachrichteneigenschaften haben mit der Form (Länge) und dem Inhalt zu tun (Nennungen von Personen (Mentions), beliebte Schlüsselwörter, Neuheit)[3]. Diese Abhängigkeit ist zentrales Thema von [Zarrella '09] und [Suh et al. '10]. Es gibt noch einen weiteren Faktor, der aber wohl sehr aufwendig zu analysieren und visualisieren wäre: Der Einfluss der Beziehung des Lesers zum Nachrichtenautor bei der Weiterempfehlung. Denn es ist davon auszugehen, dass gegenseitiges Folgen, örtliche Nähe und Stärke der bisherigen Kommunikation zweier Nutzer im System eine große Rolle bei der Weiterempfehlung ihrer Nachrichten untereinander spielt.

3.2 Verbreitung einer Nachricht durch Weiterempfehlung anderer Nutzer

Bei diesem Phänomen wird jeweils eine einzelne Nachricht betrachtet: Welchen Weg nimmt sie im Graph des DSN und welche Rolle spielen dabei die Eigenschaften der Weiterempfehlenden? Im Gegensatz zu 3.1 ist also nicht der Einfluss von Nachrichten oder Autoreneigenschaften hier im Fokus, sondern die Eigenschaften der Nutzer, welche die Nachricht weiterempfehlen. Eine gewisse Nähe zu 3.1 besteht, da neben dem Autor und Nachrichteninhalt für den Leser, der eine Weiterempfehlung erwägt, auch eine Rolle spielt, von wem er die Nachricht empfohlen bekommen hat (falls er sie nicht direkt vom Autor empfangen hat). Das Phänomen stammt aus [Kim '10].

3.3 Ausbreitungsstruktur der Nachrichten zu einem Thema

Durch die Weiterempfehlung von Nachrichten zu einem Thema in einem DSN ensteht implizit ein zweiter Graph, der nicht notwendigerweise viele Gemeinsamkeiten mit dem Graph des DSN selbst besitzen muss. Jede Zusammenhangskomponente dieses Graphen hat im Zentrum den Autor einer Nachricht zum Thema, von welchem Kanten zu Knoten ausgehen, welche die Nachricht weiterempfohlen haben. Von diesen wiederum gehen rekursiv weiter Kanten zu weiteren Knoten aus, die weiterempfohlen haben. Die Struktur solcher Graphen sagt viel über die Informationsdiffusion durch Weiterempfehlung im DSN aus. Das Phänomen wurde aus Kapitel 6.2. in [Kwak et al. '10] entnommen.

[3] Nennungen und häufig benutzte Schlüsselwörter werden in Twitter explizit mit einem Präfix belegt. Für Nennungen ist dies "@" und für Schlüsselwörter "#". Solche mit dem Präfix "#" gekennzeichnete Schlüsselwörter werden in Twitter Hashtags genannt und helfen dort besonders bei der Analyse der Nachrichtenausbreitung. Das liegt daran, dass sich in Nachrichten mit Bezug auf populäre Themen häufig die Verwendung eines gemeinsamen Schlüsselworts durchsetzt.

3.4 Sackgassen

Der Empfehlungsmechanismus in expliziter und impliziter Ausprägung (siehe Grundlagenkapitel) wird nicht von allen Nutzern eines DSN verwendet. Die Nutzer, welche nie weiterempfehlen, spielen bei der Informationsausbreitung eine besondere Rolle, weil sie unter Umständen die Nachricht daran hindern eine große Anzahl weiterer Nutzer, die selbst weiterempfehlen würden, zu erreichen. Sie werden entsprechend dem Einfluss, den sie auf Informationsflüsse haben, als Sackgassen bezeichnet.

3.5 Sprünge

Nutzer erhalten Weiterempfehlungen von anderen Nutzern, denen sie folgen. Die Autoren der weiterempfohlenen Nachrichten sind aber meist nicht mit dem Empfänger der Weiterempfehlung assoziiert. Häufig ist aber zumindest eine transitive Verbindung über eine Vielzahl an Kanten zwischen Autor und Empfänger vorhanden. Interessant ist aber, dass dies nicht notwendigerweise der Fall ist: Über öffentliche Nachrichtenströme (in Twitter nennt man dies Public Timeline) und Suchfunktion kann ein Nutzer Nachrichten von Autoren außerhalb seiner Graphenkomponente lesen und weiterempfehlen. Das wird nachfolgend Sprung genannt.

3.6 Bremsen und Beschleuniger

Auf alle Nutzer bezogen gibt es in einem DSN eine mittlere Rate des Weiterempfehlens. Diese Rate ist das Verhältnis von weiterempfohlenen Nachrichten zur Gesamtzahl der gelesenen Nachrichten. Für bestimmte Knoten und über bestimmte Kanten weicht diese Rate aber stark vom Mittelwert ab. Bei geringer Varianz sagen solche Abweichung viel über die Rolle des Knoten bzw. der Kante im Graphen bei der Informationsausbreitung über Weiterempfehlung aus.

3.7 Verhältnis der Reaktionen bei Nachrichten

Nachrichten ohne Reaktionen, mit Weiterempfehlungen, mit Kommentaren und verschiedenen Kombinationen lassen sich in ihrer Anzahl vergleichen und lassen so Rückschlüsse auf die Akzeptanz der verschiedenen Reaktionsmöglichkeiten im DSN zu. Dieses Phänomen basiert auf Überlegungen in [Sysomos ’10].

4 Visualisierungen

Nachdem in Kapitel 3 die Phänomene festgelegt und beschrieben worden sind, können jetzt die passenden Visualisierungen entwickelt werden. Dabei soll es die Aufgabe einer Visualisierung sein, jeweils mindestens eines der Phänomene besonders intuitiv und schnell erkennbar graphisch aufzuzeigen. Die Verfahren werden dabei hier noch etwas abstrakt in Form von Handlungsvorschriften beschrieben. Mehr ins Detail geht es erst im nächsten Kapitel 7.

Die in diesem Kapitel beschriebenen Visualisierungen stellen informelle Anforderungen an die in Kapitel 7 umgesetzen Visualisierungen dar.

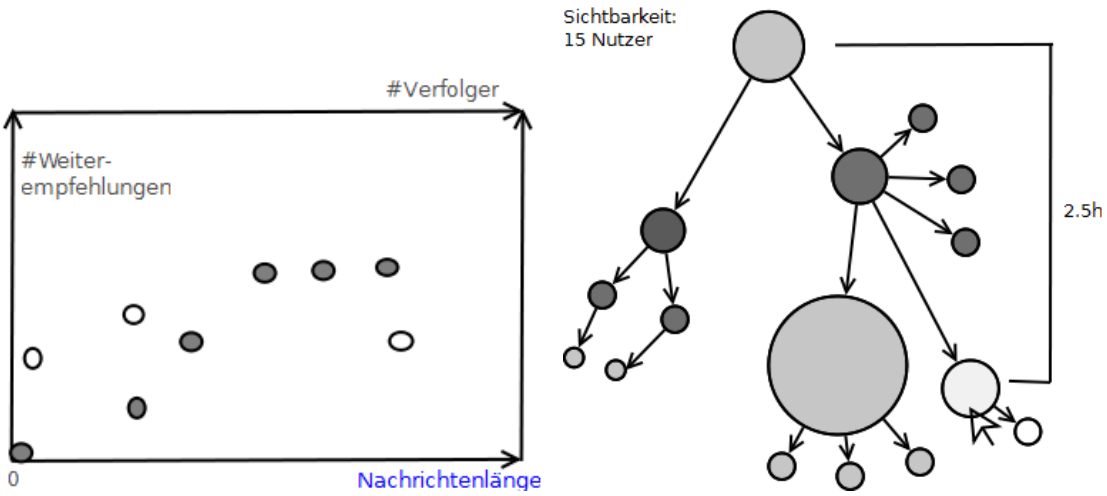
Eine Evaluation der verschiedenen Diagrammarten auf ihre Eignung bei der Visualisierung der genannten Phänomene hat ergeben, dass speziell drei Arten von Diagrammen besonders passend sind:

1. Gerichtete Graphen (Digraphen) zur Darstellung der Topologie des DSN,
2. Punkt- und Liniendiagramme für funktionale Abhängigkeiten,
3. Treemaps (vgl. 2.2.4) bei hierarchisch verfeinerten Strukturen.

Neben einer allgemeinen Recherche nach Diagrammarten und Visualisierungsansätzen aus Artikeln zum Thema wurde noch die Notation des Projekts FLOW[1] für die Verwendung in den Visualisierungen evaluiert. Dies war naheliegend, da FLOW am Fachgebiet Software Engineering entwickelt worden ist und Informationsflüsse sehr nah am Thema dieser Arbeit liegen. Sicherlich könnte man die Verbreitung von Textnachrichten in DSN als feste Informationsflüsse in FLOW modellieren. Dennoch wurde die Entscheidung getroffen diese Notation in keiner mitzuliefernden Visualisierung zu verwenden aus folgenden Gründen:

- eine Erweiterung der Visualisierungssoftware um zusätzliche Visualisierungen, die FLOW-Notation beinhalten, soll ohnehin leicht umsetzbar sein (Dazu folgt später mehr in 7.4.3),
- der Autor schätzt den Implementierungsaufwand für ein automatisches Layout von Diagrammen in FLOW-Notation als sehr hoch ein. Speziell weil in DSN meist eine sehr hohe Anzahl an Nutzern und somit auch Informationsflüssen ist,
- die dem Autor favorisierte JavaScript-Visualisierungsbibliothek “Protovis” bietet keine einfache Möglichkeit FLOW-Modelle zu zeichnen.

[1] Die Notation wurde [Schneider et al. ’08] entnommen. Weitere Informationen gibt es im Internet auf http://se.uni-hannover.de/forschung/flow/.



(a) Skizze zu Visualisierung 4.1 (b) Skizze zu Visualisierung 4.2

Abbildung 4.1: Visualisierungsskizzen

4.1 Weiterempfehlungsanzahl abhängig von Länge und Verfolgern

Punktdiagramm mit zwei Punktgruppen (jeweils mit eigener Farbe). Die erste Punktmenge (blau gefärbt) zeigt die Anzahl der Weiterempfehlungen in Abhängigkeit von der Länge der Nachricht. Die zweite Kurve (rot gefärbt) zeigt die Anzahl der Weiterempfehlungen in Abhängigkeit von der Anzahl der Verfolger des Autors. Dabei gibt es zwei Abszissen jeweils mit Skalen für Nachrichtenlänge und Anzahl der Verfolger sowie eine Ordinate mit der Anzahl der Weiterempfehlungen (Siehe Abbildung 4.1 (a)). Diese Visualisierung ist für Phänomen 3.1 (“Einfluss von Autor und Nachricht auf die Weiterempfehlung) entwickelt worden.

Wenn in der Visualisierung z.B. die roten Punkte sich auf einer Linie mit positiver Steigung anordnen, würde dies auf eine proportionale Abhängigkeit der Anzahl der Weiterempfehlungen einer Nachricht von der Anzahl der Verfolger ihres Autors hindeuten. Zusätzlich kann man an dieser Visualisierung erkennen, in welchem Ausmaß Nachrichten überhaupt weiterempfohlen werden. Ebenso zeigt sich an den blauen Punkten die Verteilung der Längen der Nachrichten.

Verfahren *Eingabedaten:* Nutzerliste, Folgerelation, Nachrichten

Für die erste Punktmenge wird für jede Nachricht die Anzahl der Zeichen (x-Koordinate des Punkts) und die Anzahl der Weiterempfehlungen (y-Koordinate des Punkts) gezählt. Für die zweite Punktmenge wird für jede Nachricht die Anzahl der Verfolger des Autors gezählt (x-Koordinate) und analog zur ersten Punktmenge die Anzahl der Weiterempfehlungen (y-Koordinate).

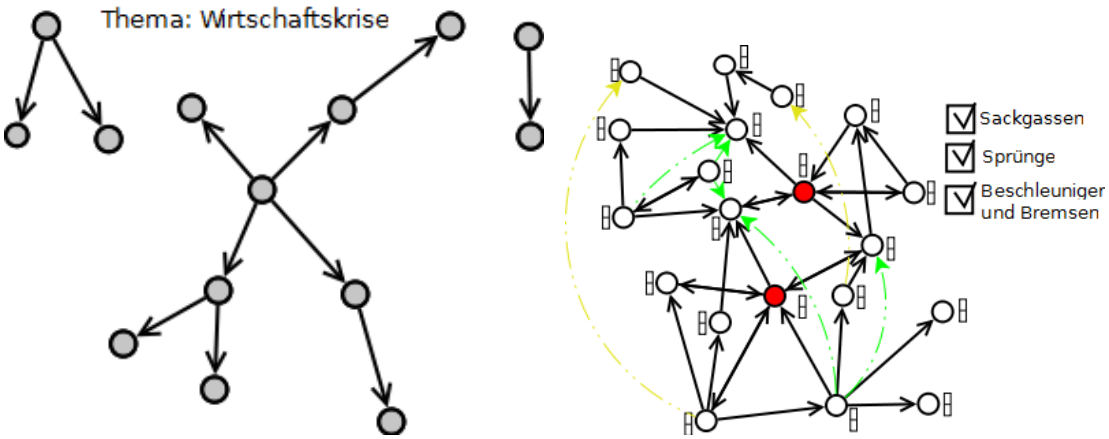
4.2 Baum der Weiterempfehlung einer Nachricht

Ein Baum für eine bestimmte Nachricht im System. Wurzel ist der Autor der Nachricht. Interne Knoten sind Nutzer, welche die Nachricht weiterempfehlen. Nutzer, welche sie lediglich lesen, werden als Blätter abgebildet. Die Symbolik für das Zeichnen von Knoten wird variiert: Die Knotengröße ist proportional zur Anzahl der Verfolger des Nutzers und die Knotenhelligkeit ist antiproportional zum Alter des Nutzeraccounts. Neben den Knoten werden die Namen der Benutzer, welche durch die Knoten repräsentiert werden, angezeigt. Als interaktives Element wird bei dieser Visualisierung beim Überfliegen eines internen Knotens mit dem Mauszeiger die gesamte vergangene Zeit vom Erstellen der Nachricht durch den Autor bis zur Weiterempfehlung der Nachricht durch den selektierten Nutzer als Textlabel im Vordergrund angezeigt. Die Gesamtzahl der Nutzer, die diese Nachricht empfangen, wird ebenfalls als Textlabel angezeigt. Skizziert wurde dies in Abbildung 4.1 (b).

Diese Visualisierung ist für Phänomen 3.2 und basiert stark auf [Kim ’10]. Ob nur eine kleine Teilmenge der Verfolger eines Autors dessen Nachrichten weiterempfiehlt, könnte man mit dieser Visualisierung herausfinden. Dazu müsste man nur die Bäume der Nachrichten dieses Autors parallel betrachten. Alternativ kann man in der Visualisierung auch schauen, ob die längsten Wege von der Wurzel zu einem Blatt Gemeinsamkeiten haben. Bspw. könnten sie nur durch große Knoten, d.h. Nutzer mit vielen Verfolgern, verlaufen. Das Nutzeralter in die Darstellung zu übernehmen könnte eine Korrelation des Alters des Benutzerkontos und der Verwendung der Weiterempfehlungsfunktion zeigen.

Verfahren *Eingabedaten:* Folgerelation, Nachrichten, Empfehlungen

Aus praktischen Gründen werden zuerst alle Nachrichten ausgefiltert, welche nicht mindestens einen gewissen Schwellenwert an Weiterempfehlungen haben. Danach werden diese in einer Auflistung dem Nutzer zur Auswahl angezeigt (möglicherweise als Combo-Box). Nachdem der Betrachter der Visualisierung sich eine Nachricht ausgesucht hat, werden die Informationen Name, Verfolgerzahl und Zugangsalter jeweils zum Autor der Nachricht und allen Nutzern, welche diese weiterempfohlen haben, beschafft. Die noch nicht ermittelten Namen von Nutzern, welche zwar einem der im letzten Schritt ermittelten Nutzer folgen aber die Nachricht nicht weiterempfehlen und nicht der Autor sind, müssen ebenfalls geklärt werden. Danach kann über die Metadaten der Weiterempfehlungen (von welchem Nutzer weiterempfohlen) die Struktur des Baums konstruiert werden. Die Latenz von Nachrichtenerstellung bis zu einer bestimmten Weiterempfehlung lässt sich anhand des Protokolls ermitteln. Zuletzt werden alle Knoten im Baum aufgezählt und dekrementiert (Wurzel, d.h. Autor, ist kein Empfänger) und dadurch die Zahl der Empfänger der Nachricht errechnet. Das Ergebnis wird dann angezeigt.



(a) Skizze zu Visualisierung 4.3 (b) Skizze zu Visualisierung 4.4

Abbildung 4.2: Weitere Visualisierungsskizzen

4.3 Weiterempfehlungsbäume zu einem Thema

Es wird der in 3.3 genannte implizite Graph durch Weiterempfehlung von Nachrichten für jede Nachricht zu einem Thema gezeichnet. Eine Nachricht gehört zu einem Thema, falls es im Inhalt eines der Schlüsselwörter des Themas enthält. Die Anzahl der angezeigten Nachrichtenbäume wird daher monoton mit der Anzahl der Schlüsselwörter eines Themas wachsen. Vgl. Abbildung 4.2 (a).

Diese Visualisierung zeigt Phänomen 3.3 und stammt aus [Kwak et al. '10]. Auf der einen Seite ist hier sofort die Beliebtheit der angegebenen Schlüsselwörter bei den Nutzern des DSN zu erkennen. Falls die Schlüsselwörter alle zu diesem Thema verwendeten Wörter enthalten, kann man an der Visualisierung auch die Beliebtheit eines Themas sehen. Sind es sehr wenige Ausbreitungsbäume ist ein Schlüsselwört bzw. Thema unbeliebt. Viele Bäume würden entsprechend hohe Beliebtheit zeigen. Ansonsten ist der größte Vorteil dieser Visualisierung, dass sie die Ausbreitungsstruktur der Informationen auf einen Blick zeigt. Wird "breit" weitergeleitet oder findet die Weiterempfehlung eher in die Tiefe statt mit Ausbreitungsbäumen, die mehr einer Liste ähneln.

Verfahren *Eingabedaten:* Folgerelation, Nachrichten, Empfehlungen

Die Nachrichten, welche zu einem Thema passen, werden über Schlüsselwörter gefunden. Diese Schlüsselwörter sollen am Anfang als Liste vom Nutzer in einer Textbox (einzelne Schlüsselwörter getrennt mit Komma) eingegeben werden. Darauf werden Nachrichten, die keines der Schlüsselwörter enthalten, weggefiltert. Sie passen nicht zum Thema. Nun wird für jede Nachricht ähnlich zu 4.2 ein Baum aufgebaut. Jedoch werden Nutzer, die die Nachricht lediglich lesen, nicht eingezeichnet und die Knotensymbolik nicht variiert. Die Bäume in dieser Visualisierung stellen also eine Generalisierung der Bäume aus 4.2 dar.

4.4 Ergänzter Graph

Der Digraph des DSN wird als Grundebene gezeichnet. Sackgassen 3.4, Sprünge 3.5 sowie Bremsen und Beschleuniger 3.6 kann der Nutzer über eine Radiobox für jedes dieser Phänomene als zusätzliche Ebene hinzu zeichnen lassen. Folglich visualisiert jede Ebene das Phänomen, dessen Namen es trägt. Genauer werden Sackgassen als rot eingefärbte Knoten, Sprünge als zusätzliche Kanten und Bremsen/Beschleuniger mit einer Art Thermometer (mit Ursprung in der vertikalen Mitte) neben den Knoten eingezeichnet. Siehe dazu auch Abbildung 4.2 (b).

Diese Visualisierung ist besonders gut dazu geeignet die Akzeptanz der Weiterempfehlung in einem DSN zu zeigen. Falls diese nämlich gering ist, gibt es sehr viele Sackgassen, d.h. rote Knoten. Zusätzlich kann man in der Visualisierung bei vielen Sprüngen einen Mehrwert der Weiterempfehlung für die einzelnen Nutzer sehen. Denn Sprünge bedeuten, dass durch die Weiterempfehlung Nutzer an Informationen gelangen, welche sie ansonsten nicht erhalten würden. Die rechteckigen Anzeigen neben jedem Knoten ermöglichen außerdem das Auffinden von Gruppen, welche außerordentlich selten Nachrichten weiterempfehlen. Diese Personen könnten dann gezielt befragt werden, weshalb sie von dieser vom DSN bereitgestellten Funktionalität keinen Gebrauch machen.

Verfahren *Eingabedaten:* Nutzerliste, Folgerelation, Nachrichten, Empfehlungen

Der Graph als Hintergrund wird über die Nutzerliste und die Folgerelation vollständig bestimmt. Er wird mit einem Force Directed Layout (siehe 2.2.3) gezeichnet.

Für die Sackgassen muss die Anzahl der Weiterempfehlungen, welche ein Nutzer getätigt hat, für jeden Nutzer ermittelt werden. Dies funktioniert, indem die Liste der Nutzer dupliziert und beim Durchlaufen der Empfehlungen jeden Nutzer aus dieser neuen Nutzerliste entfernt, welcher diese Empfehlung gemacht hat. Ist er bereits entfernt, braucht und kann dies natürlich kein zweites Mal geschehen. Die übrig bleibenden Nutzer werden mit der Farbe Rot eingefärbt.

Für die Sprünge wird für jede Weiterempfehlung geschaut, ob der Empfehler dem Autor der Nachricht direkt folgt. Ist dies nicht der Fall, wird eine zusätzliche grüne Kante eingezeichnet, wenn er dem Autor zumindest transitiv folgt. Folgt er ihm nicht einmal transitiv, wird eine rote Kante vom Empfehlenden zum Autor eingezeichnet.

Für die Bremsen und Beschleuniger wird die Weiterempfehlungsrate jedes Nutzers bestimmt. Neben dem Knoten eines Nutzers wird ein hohes Rechteck mit einer horizontalen Linie in der vertikalen Mitte gezeichnet. Sie stellt die mittlere Weiterempfehlungsrate dar. Nun wird, je nachdem ob der Nutzer eine höhere oder niedrigere Weiterempfehlungsrate hat, ein grüner oder roter Balken nach oben bzw. unten eingezeichnet. Die Sättigung des Grün bzw. Rot ist proportional zur Stärke der Abweichung.

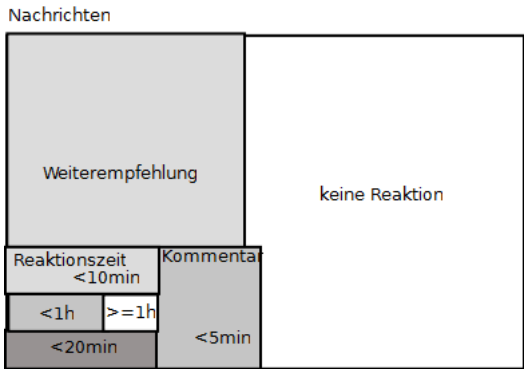


Abbildung 4.3: Skizze zu Visualisierung 4.5.

4.5 Treemap des Reaktionsverhaltens

Es wird aggregiert über alle Nachrichten der Anteil der Nachrichten, welche eine Reaktion erhalten und nach Reaktion unterteilt (Weiterempfehlung, Kommentar), als Treemap (siehe 2.2.4) gezeichnet. Die einzelnen Teilflächen werden dabei entsprechend der Baumstruktur der Knoten der Treemap unterteilt, was mehr Details liefert. Bspw. beim Block “Weiterempfehlung” wird unterteilt in Gruppen von Zeiträumen, in welchem weiterempfohlen wurde: 5min, 10min, 30min, 1h, 2h, 5h und größer 5h. Siehe Abbildung 4.3.

Diese Visualisierung zeigt das Phänomen 3.7 und ist angelehnt an eine Visualisierung mit Kreisdiagrammen aus [Sysomos ’10]. Zeigt diese Visualisierung nur eine große Fläche mit der Aufschrift “keine Reaktion”, sind sich die Nutzer u.U. der Möglichkeit der Reaktion auf Nachrichten gar nicht bewusst. Befinden sich alle Reaktionen in Intervallen von nur wenigen Minuten, zeigt dies, dass auf Nachrichten entweder direkt reagiert wird oder diese nach bereits kurzer Zeit aus dem Bewusstsein der Leser verschwinden.

Verfahren *Eingabedaten:* Nachrichten, Kommentare, Empfehlungen

Es wird für jede Nachricht überprüft, ob es zu ihr eine Reaktion gibt. Falls nicht wird der Zähler für Nachrichten ohne Reaktion inkrementiert. Gibt es eine Reaktion wird je nach Art der Zähler für Nachrichten mit Weiterempfehlung bzw. der für Nachrichten mit Kommentar inkrementiert. Die Treemap enthält danach als Grundmenge die Anzahl der Nachrichten gesamt und die übrigen Teilmengen werden aus den Zählern ermittelt. Für die Teilflächen werden die Größen analog ermittelt. Das geht solange, bis der Algorithmus bei den Blättern der Hierarchie angelangt ist.

5 Beispielszenarien

Es wird nachfolgend die Erzeugung der Beispielszenarien beschrieben.

Für eine besonders kompakte Beschreibung der Szenarien werden diese über eine Simulation erzeugt, bei welcher die Aktionen jeweils bestimmte Wahrscheinlichkeiten besitzen, welche als konstant festgelegt werden.

Der umgesetzte Simulationsalgorithmus, welcher in folgendem Abschnitt konzeptionell entworfen wird, ist ein zusätzlicher Bestandteil der entwickelten Visualisierungsanwendung (vgl. Kapitel 7). Dies erlaubt neben der Verwendung der drei gespeicherten Beispielszenarien das Erstellen beliebig vieler weiterer Szenarien mit den zwei nachfolgend beschriebenen Parametersätzen.

Als Zeiteinheit werden in der Simulation sogenannte “Ticks” verwendet. Da dies positive Ganzzahlen sind, handelt es sich also um eine zeitdiskrete Simulation.

5.1 Generierung von Szenarien

Als Ausgangspunkt für die hier entwickelte Simulation sollen die Resultate der “90-9-1 Theorie” 2.1.8 gelten. Diese Theorie beschreibt das Verhältnis der Aktivitätshäufigkeit in den verschiedenen Teilnahmequalitäten vom reinen Konsum hin zum Erstellen von signifikanten Ressourcen. Es wird folgendermaßen zugeordnet:

- Insignifikante Beiträge sind Weiterempfehlungen und Kommentare.
- Zu den signifikanten Beiträgen gehört das Erstellen von Nachrichten sowie das Folgen und Beenden einer Folge-Beziehung.

Konsum wäre das Lesen einer Nachricht. Dieses Ereignis wurde aber später bei der Umsetzung wieder entfernt (siehe Abschnitt 7.1).

Es gibt noch eine Menge weiterer Gesetzmäßigkeiten, die in echten DSN auftreten. So besitzen DSN im Allgemeinen die Eigenschaft skalenfrei (Siehe 2.1.7) zu sein. Der Fokus dieser Arbeit liegt jedoch auf der Visualisierung und nicht auf der Simulation. Daher werden zusätzliche Bedingungen neben 2.1.8 hier nicht weiter beachtet.

Allgemein soll es in der Simulation zu Beginn 20 Nutzer geben. Diese wären dann in jedem erzeugten Szenario enthalten. Damit nicht alle den gleichen Zeitpunkt der Erstellung ihres Zugangs haben, wird bei der Simulation eine Startzeit anzugeben sein. Der Zeitpunkt des Anlegens eines Nutzerzugangs soll sich dann gleichmäßig über das Intervall vom Zeitpunkt 0 bis zur Startzeit verteilen.

Zusätzlich sollen am Startzeitpunkt 20 Nachrichten mit zufälligem Inhalt erzeugt werden. Die Autoren der Nachrichten sollen rein zufällig aus der Menge der bereits angelegten Nutzer ausgewählt werden.

5.2 Parametersätze

5.2.1 Erster Parametersatz

Für jeden Nutzer liegt pro Tick die Wahrscheinlichkeit eine neue Aktion zu erzeugen bei 5%. Jeweils pro Nutzer ist dann die Wahrscheinlichkeit pro Tick ein Artefakt zu konsumieren 90%. Das Erzeugen eines Artefakts unter diesen Vorbedingungen hat eine Wahrscheinlichkeit von 10%. Davon fallen 9% auf insignifikante Artefakte wie Empfehlungen und Kommentare. Die restlichen 1% Verfallen auf signifikante Beiträge wie Nachrichten erstellen sowie Folgen und “Entfolgen”. Das Löschen von Nachrichten oder Nutzerkonten tritt in der Simulation nie als Ereignis auf. Dies sollte die Umsetzung vereinfachen ohne die Qualität der Simulation deutlich zu verringern, da diese beiden Aktionen üblicherweise sehr selten sind.

5.2.2 Zweiter Parametersatz

Der zweite Parametersatz unterscheidet sich vom ersten nur in der Wahrscheinlichkeit für das Weiterempfehlen von Nachrichten. So war diese im ersten Satz 4.5%. Im zweiten Satz hingegen liegt sie zufällig zwischen 5% und 15%.

5.3 Zwei Beispielszenarien

Aus dem ersten und zweiten Parametersatz werden jeweils zwei Szenarien durch die Simulation generiert. Diese werden abgespeichert und als Beispielszenarien für die Mockups selbst und den späteren Vergleich der Visualisierungsergebnisse mit diesen verwendet. Die Serialisierung der Szenarien erfolgt in CSV (siehe 2.3.1), wobei als Trennzeichen hier ein Semikolon anstelle des meist in CSV üblichen Komma verwendet worden ist. Im Deskriptor einer Nachricht[1] werden die einzelnen Eigenschaften jeweils über einen Doppelpunkt getrennt.

Beide Simulationen liefen 1000 Ticks mit einer Startzeit von 500 Ticks.

Zeilen, welche mit zwei aufeinanderfolgenden Schrägstrichen (//) beginnen, werden übersprungen, was für Kommentare verwendet werden kann.

Die erste Zeile einer solchen Datei ist immer eine Auflistung der Capabilities. Das Format für die Serialisierung von Szenarien wird später ebenfalls zur Serialisierung der

[1] Serialisierte Textrepräsentation einer Nachricht für Verwendung in der Spalte “action” (s.u.).

eingelesenen Daten aus einem DSN verwendet. Da durch die Programmierschnittstellen der unterschiedlichen DSN auch verschiedene Arten an Daten preisgegeben werden, können u.U. nicht genügend Daten für alle Visualisierungen verfügbar sein. Deshalb sollen die sogenannten Capabilities zu Beginn eines Datensatz zeigen, welche Art Daten aus der Quelle des Datensatz ausgelesen worden konnten. Die Capabilities sind dafür nötig, da das reine Fehlen einer Klasse von Daten auch einfach darin begründet sein kann, dass es tatsächlich bspw. keine Nachrichten im DSN gibt. Es gibt folgende Capabilities: userlist (Nutzerliste), followsrel (Folgerelation), recommendsrel (Empfehlungsrelation), messages (Nachrichtenliste), events (Ereignisliste). Danach wird für jedes Ereignis (vgl. 2.3.2) der Simulation eine Zeile mit folgender Struktur zugefügt:

```
username; action; parameter(s); timestamp
```

Der Nutzername muss natürlich eindeutig sein.

Die möglichen Aktionen mit Parameter sind:

- `newUser;name` einen neuen Nutzer mit Namen `name` anlegen.
- `createMsg;msgDescriptor` eine neue Nachricht anlegen, welche durch den Nachrichtendeskriptor `msgDescriptor` beschrieben wird. Der Aufbau der Deskriptoren für Nachrichten ist: `msgId:author:text:ref`.
- `follow;name` einem Nutzer mit Namen `name` folgen.
- `unfollow;name` einem Nutzer mit Namen `name` nicht mehr folgen.
- `recommend;msgId` eine Nachricht mit Identifikationsnummer `msgId` empfehlen.

Die Extraktion der Daten aus Hallway erfolgt später durch eine Hallway-Erweiterung, welche in dasselbe Datenformat exportiert, wie es hier für die Beispielszenarien beschrieben ist. Mehr dazu folgt in Kapitel 6.

Die Textdateien (`.csv`) der gespeicherten Beispielszenarien, welche hier verwendet werden, liegen der im Anhang mitgelieferten CD bei. Sie befinden sich im Ordner von DSNViz.

5.3.1 Zusätzliches drittes Beispielszenario

Da sich beim Erstellen der Mockups für die Beispielszenarien für Visualisierung 4.1 nichts außer statistisches Rauschen erkennen ließ[2], wurde als Basis für ein Mockup zu dieser Visualisierung ein dritter Datensatz per Hand erzeugt und gespeichert (ebenfalls mitgeliefert). Er enthält nur halb so viele Nutzer (10 statt 20) und die Anzahl der Empfehlungen ist proportional zur Anzahl der Verfolger des Autors und antiproportional zur Länge der Nachricht.

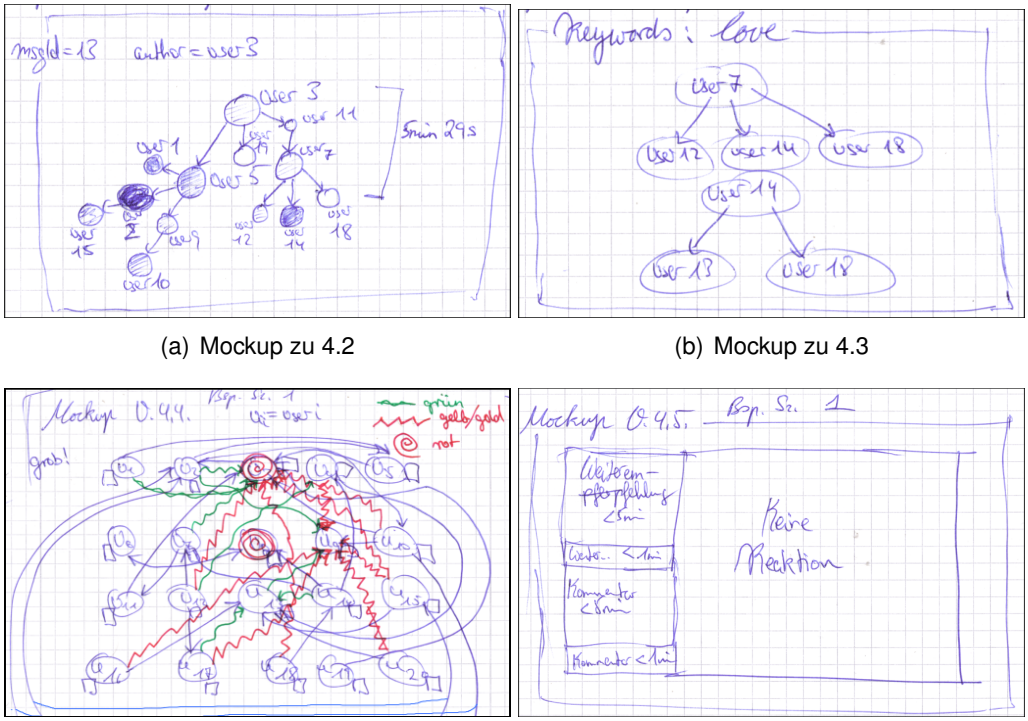
[2] Dies weist auf Mängel der Simulation hin. Die Schwächen der Simulation werden auch im Ausblick thematisiert.

5.3.2 Mockups

Die Mockups dienen als grobe Soll-Werte für einen Test der zu erstellenden Visualisierungssoftware. Es handelt sich um einen Black-Box-Test, da bei der Erstellung der Mockups lediglich die Beschreibung der Visualisierungsergebnisse selbst in Kapitel 4 beachtet werden soll. Das schon skizzierte zugehörige Verfahren soll jeweils ignoriert werden. Die Mockups können erst nach Festlegung der Beispielszenarien gezeichnet werden, denn selbst für die Skizze eines Visualisierungsergebnis werden Eingabedaten benötigt.

Allgemein sei zu den nachfolgenden Mockups noch anzumerken, dass teilweise nur Ausschnitte des zu erwartenden Visualisierungsergebnis skizziert worden sind. Das liegt daran, dass bspw. für Visualisierung 4.2 bei beiden Szenarien Mockups von über 100 Graphen notwendig wären, da die Nachrichtenbäume für jede Nachricht gezeichnet werden können und die beiden Szenarien jeweils mehr als 100 Nachrichten enthalten. Deshalb wurde in den Mockups zu Visualisierung 4.2 jeweils die Identifikationsnummer der Nachricht, dessen Ausbreitungsbaum eingezeichnet ist, mit eingefügt. Alle Mockups werden in den nächsten beiden Seiten abgebildet.

Für die Mockups zu Visualisierung 4.4 ist festzustellen, dass Beispielszenarien mit weniger als 20 Nutzern in diesem Fall zweckmäßiger gewesen wären.



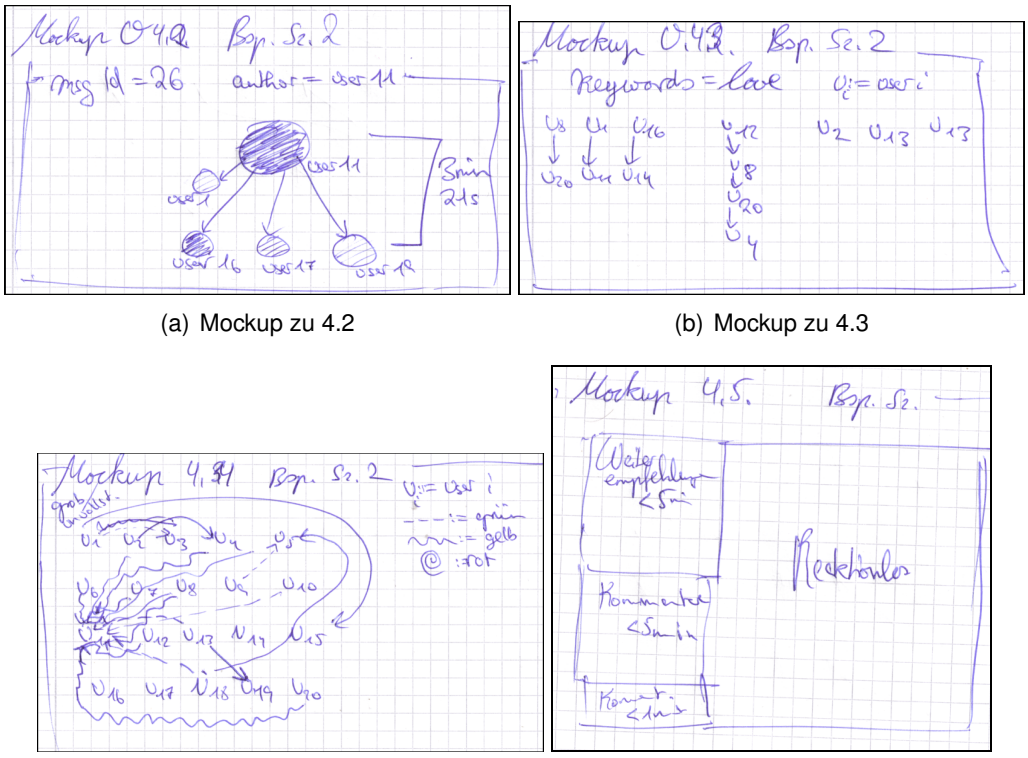
(a) Mockup zu 4.2

(b) Mockup zu 4.3

(c) Mockup zu 4.4

(d) Mockup zu 4.5

Abbildung 5.1: Mockups zu Beispielszenario 1



(a) Mockup zu 4.2

(b) Mockup zu 4.3

(c) Mockup zu 4.4

(d) Mockup zu 4.5

Abbildung 5.2: Mockups zu Beispielszenario 2

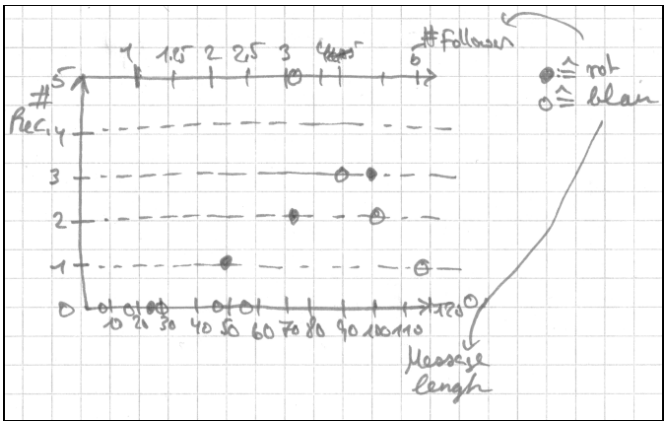


Abbildung 5.3: Mockup zu 4.1 für Beispielszenario 3

6 Entwurf

Die auf konzeptioneller Ebene beschriebenen Verfahren sollen unter Verwendung des Play-Frameworks auf Seite des Servers bzw. von Bibliotheken für die Skriptsprache JavaScript auf Seite des Clients umgesetzt werden.

Die Vorarbeiten für die Umsetzung der einzelnen Visualisierungen wurde bereits in den vorangegangenen beiden Kapiteln abgeschlossen. Zur Vermeidung von Redundanz bei der Implementierung der Beschaffung und Aufbereitung der Daten, soll zusätzlich eine Infrastruktur für Visualisierungen von Daten in DSN geschaffen werden.

Die dafür entwickelte Play-App mit Namen “DSNViz” soll auf einem Datenmodell arbeiten, welches nur die für die Visualisierungen nötigen Daten in einer Form enthält, dass es eine besonders hohe Kompatibilität zu vielen DSNs und insbesondere auch Hallway hat. Dies hat direkten Einfluss auf den Implementierungsaufwand der Anbindungen für weitere DSN. Diese Anbindungen werden nachfolgend Transformatoren[1] genannt.

Das Play-Framework wurde aufgrund folgender Argumente gewählt:

- Es wird ebenfalls in Hallway verwendet, was eine unter Umständen nötige Integration des Codes von DSNViz in Hallway hinein deutlich vereinfachen würde.
- Es vereinfacht die Verwendung der JPA[2] und damit das Zwischenspeichern von für die Visualisierung benötigten Daten in einer Datenbank. Dieses Zwischenspeichern ist nötig, um nicht bei jeder Anfrage nach einer Visualisierung die Daten neu laden zu müssen.

Durch das Play-Framework wird die Verwendung des MVC-Patterns erzwungen. Die Interaktion des Benutzers mit dem Controller findet dabei durch das HTTP statt. Das funktioniert, indem HTTP-Anfragen von Nutzern über sogenannte Routen an die Controller weitergeleitet werden.

Routen müssen Play in der `routes`-Datei im Ordner `conf` der Play-App angegeben werden. Eine Route wird dabei durch ein 3-Tupel aus HTTP-Anfragemethode (hier nur GET oder SET), URI-Muster und Name des Controllers mit angehängtem Methodennamen gebildet. Bspw. bedeutet `GET /listUsers/Application.listUsers`, dass bei Empfangen der HTTP-Get-Anfrage “/listUsers/” die Methode `listUsers` des Controllers `Application` aufgerufen wird.

[1] Tatsächlich entspricht ein Transformator einem Adapter aus dem Entwurfsmuster Objektadapter (Adapter mit Delegation). Der Klient ist dabei der Anwendungscontroller, dessen Anfragen an die entsprechenden Anfragen für den Dienst der Datenquelle angepasst werden und so über die Schnittstelle, die der Adapter implementiert, eine Kommunikation zwischen Klienten und Zieldienst ermöglichen.

[2] http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html

Die Übertragung der Daten aus dem Modell in das View wird in Play durch Verwendung von sogenannten Templates im HTML der Views realisiert. Play-Template Elemente sind Textblöcke, welche mit geschweiften Klammern vom restlichen Text abgegrenzt werden. Ein voran gestelltes Zeichen gibt ihren Typ an (Ausdruck, Dekorator, Tag, Aktion, Kommentar, Skript). Der Inhalt dieser Einschübe wird in Groovy-Syntax ausgedrückt. Bei Groovy handelt es sich um eine Java ähnliche Skriptsprache, welche in der JVM läuft.

6.1 Grundlegende Architektur

Um die in dieser Arbeit entwickelten Visualisierungen auch für andere DSN als Hallway verwenden zu können, wurde die Entscheidung getroffen diese nicht komplett in Hallway einzuarbeiten. Stattdessen sind die Visualisierungen Teil einer eigenständigen neuen Web-Anwendung, welche Daten aus einem DSN sammelt, aufbereitet und per Anfrage zusammen mit der passenden Visualisierung zurück liefert.

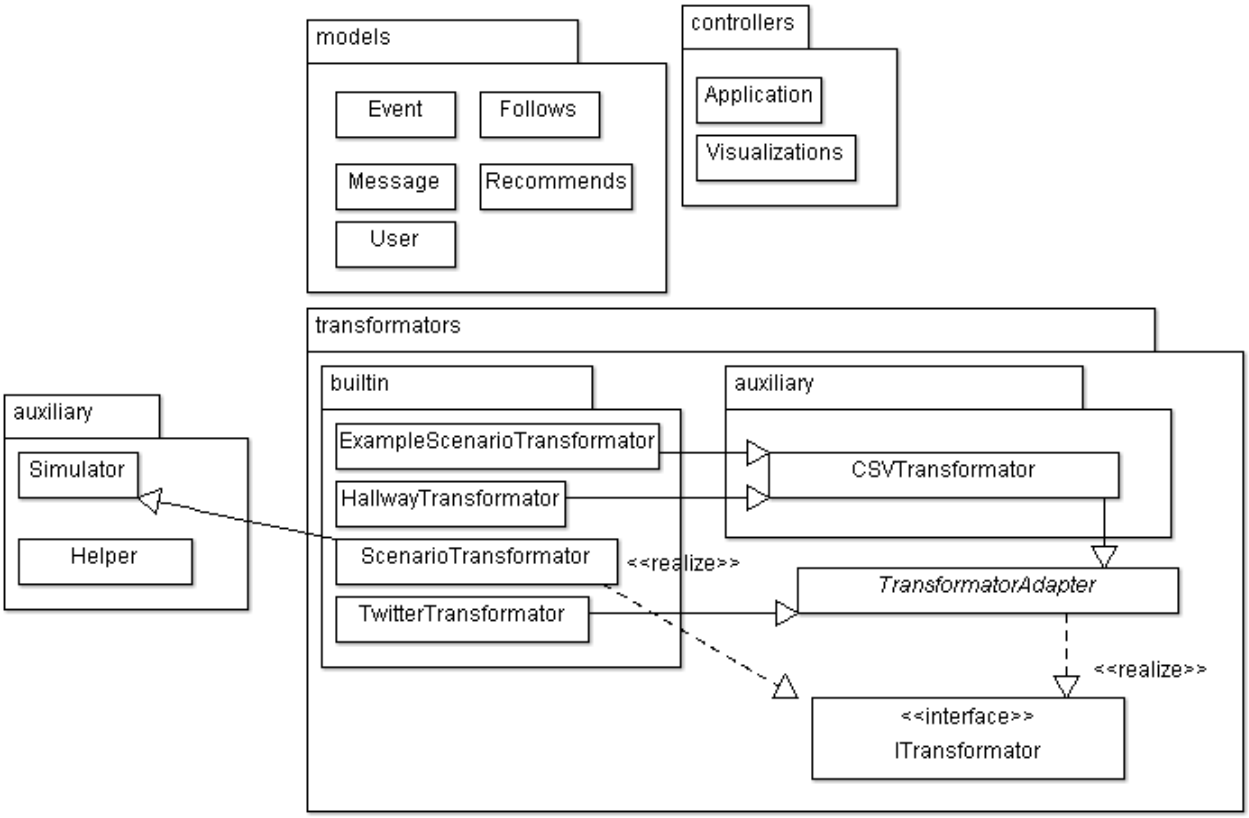


Abbildung 6.1: Grobe Struktur als Klassendiagramm

Diese Entkopplung und die damit erreichte erhöhte Modularität hilft dabei eine Reihe von Risiken zu vermeiden:

- Funktionsfehler bei Weiterentwicklung von Hallway bzw. aufwändige Reintegration des in dieser Arbeit entwickelten Codes in neuere Versionen von Hallway.
- Hohen Einarbeitungsaufwand in den Code von Hallway, welcher eine recht hohe Komplexität vorweist.
- Keine Weiterverwendbarkeit bei Wechsel von Hallway auf andere DSN-Software.

Ein Überblick über geplante Struktur der Visualisierungsanwendung gibt das UML-Diagramm in Abbildung 6.1. Auf den Aufbau und die Rolle der einzelnen Pakete und Klassen wird nachfolgend weiter eingegangen.

6.2 Gemeinsames Datenmodell

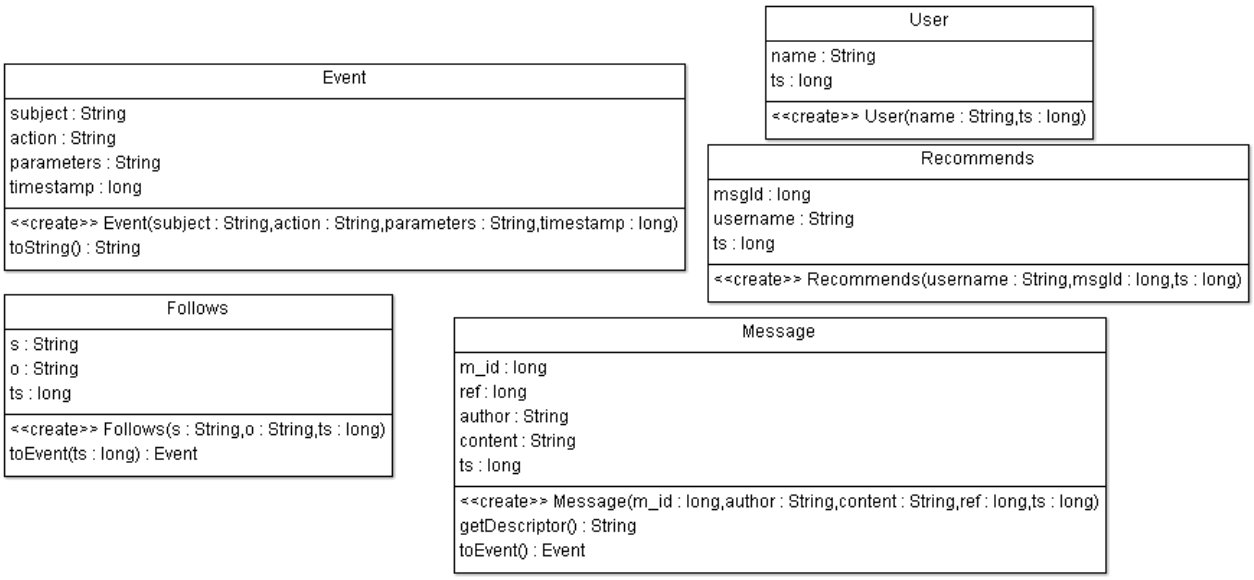


Abbildung 6.2: Datenmodell von DSNViz

Die Daten für die Visualisierung werden in Objektlisten gespeichert. Die Klassen dieser Objekte repräsentieren einen Nutzer, ein Folge-Tupel, ein Empfehlungs-Tupel, eine Nachricht sowie ein Ereignis.

- Ein Nutzer (User) hat einen Namen `name` sowie einen Zeitpunkt `ts`, an dem sein Zugang erstellt worden ist.
- Eine Nachricht (Message) wird gekennzeichnet durch ihre Identifikationsnummer `m_id`, den Namen ihres Autoren `author`, ihren Inhalt `content`, den Zeitpunkt `ts` ihrer Erstellung sowie eine mögliche Referenz `ref` auf eine weitere Nachricht.

Ist diese nicht gesetzt, enthält sie den Wert 0. Ein Wert ungleich 0 bedeutet, dass es sich um einen Kommentar zur Nachricht handelt, dessen Identifikationsnummer in `ref` gespeichert ist.

- Ein Folge-Tupel (Follows) besteht aus dem Namen `s` des Subjekts (dem, der folgt), dem Namen `o` des Objekts (dem gefolgt wird) sowie dem Zeitpunkt `ts` an dem die Folgebeziehung hergestellt worden ist.
- Ein Empfehlungs-Tupel (Recommends) wird vollständig beschrieben durch den Namen `username` des Erstellers der Empfehlung, der Identifikationsnummer `msgId` der empfohlenen Nachricht sowie dem Zeitpunkt `ts` der Empfehlung.

Zusammen werden die zugehörigen Klassen in Abbildung 6.2 als Klassendiagramm dargestellt.

Die Struktur des Graphen eines DSN wird dabei bereits durch die Nutzermenge (User) und die Folgerelation (Following) gespeichert. Nachrichten (Message) sind häufig von Interesse und werden deshalb separat gesichert. Die sogenannten Log-Ereignisse (Event) stellen atomare Aktionen eines Nutzers im DSN dar. Sie dienen lediglich einer vereinfachten Serialisierung von Ereignissen, welche für den CSV-Export gebraucht werden.

6.3 Transformatoren

Die Transformatoren werden über eine Schnittstelle (siehe Abbildung 6.3) in die Play-App eingesteckt. Sie ermöglichen es, verschiedene Digitale Soziale Netzwerke als Datenquellen für die Visualisierungen zu verwenden.

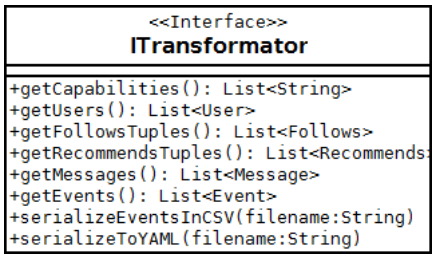


Abbildung 6.3: Schnittstelle, welche jeder Transformator implementieren muss.

6.3.1 Schnittstelle für Transformatoren

Gemeinsame Schnittstelle aller Transformatoren ist das Interface `ITransformator` (vgl. Abbildung 6.3). Die Transformatoren implementieren Funktionen wie zum Beispiel `getUsers()`, welche der Nutzerliste aus Punkt 2 in Definition 2.1.1 entspricht. Die Funktion `getFollowsTuples()` gibt entsprechend die Assoziationen zwischen den

Benutzern aus Punkt 3 in 2.1.1 wieder. Analog werden die restlichen Daten von den verbleibenden Getter-Methoden zurückgeliefert.

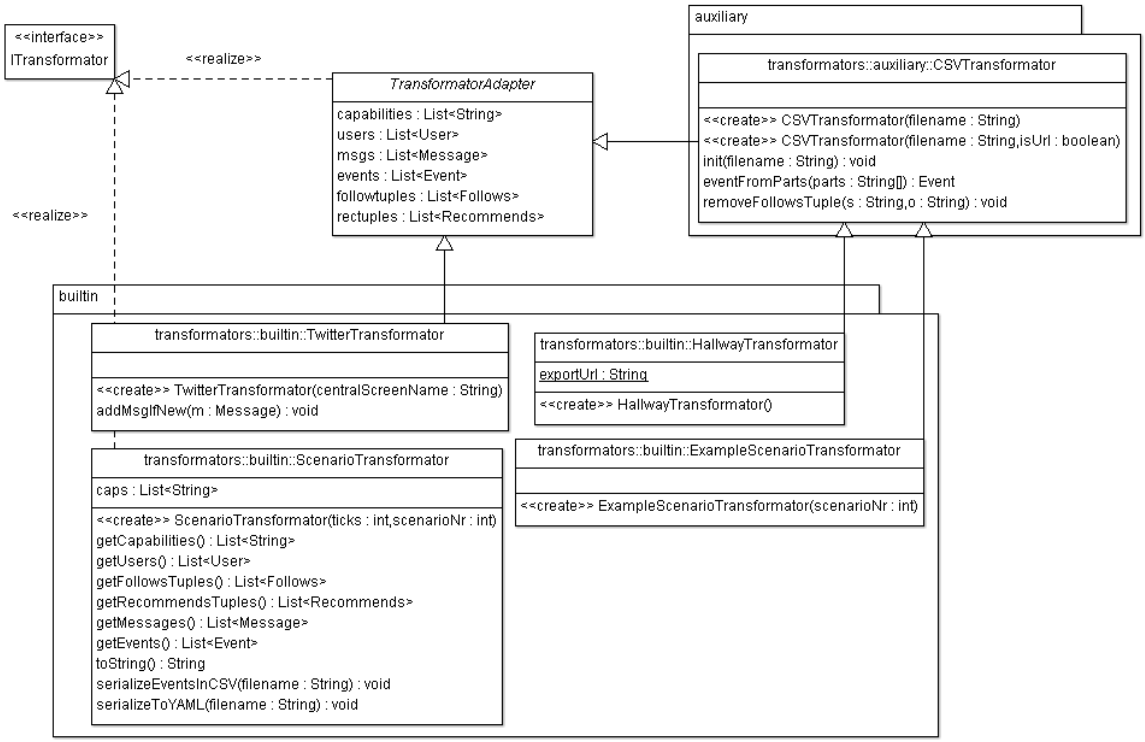


Abbildung 6.4: Paket `transformators`.

Abbildung 6.4 zeigt das `transformators` Paket sowie die Beziehung der verschiedenen Transformatoren untereinander und zu ihrer Schnittstelle.

6.3.2 TransformatorAdapter

Eine Gemeinsamkeit aller Transformatoren ist, dass sie die in den Abfragemethoden (Getter) zurückgegebenen Listen in der Regel alle gemeinsam laden werden und bis zu einer Anfrage zwischenspeichern werden. Entsprechend werden sie alle diese Listen als Attribute enthalten. Zur Vermeidung von Redundanz werden diese gemeinsamen Attribute und zugehörigen zu erstellenden Implementierungen der Getter in eine abstrakte Klasse mit Namen `TransformatorAdapter` geschoben.

Das Suffix “Adapter” im Namen dieser abstrakten Klasse soll in diesem Kontext nicht auf das Design-Pattern Adapter hinweisen sondern ist angelehnt an die Bezeichnung der abstrakten Klassen[3] in der AWT-API, welche Teil der Java Foundation Classes

[3] Ein Beispiel ist hier die abstrakte Klasse `java.awt.event.EventAdapter`, welche die Methoden der Schnittstelle `java.awt.event.EventListener` mit Methoden ohne Inhalt implementiert.

(JFC) ist. In Swing dienen diese abstrakten Klassen dazu, einem die Implementation von Schnittstellenmethoden mit leeren Methoden zu ersparen.

Bei der Erweiterung von DSNViz um zusätzliche Transformatoren wird empfohlen von dieser Klasse zu erben, anstatt die Schnittstelle `ITransformator` direkt zu implementieren. Erstens wird dadurch redundanter Code vermieden, der nur zusätzliche Wartungsarbeiten erzeugt. Zweitens sinkt dadurch die Wahrscheinlichkeit, dass die Schnittstelle semantisch fehlerhaft implementiert wird.

6.3.3 CSVTransformator und Unterklassen

Da sowohl der Exporter der Daten von Hallway seine Ergebnisse in CSV speichern soll und auch die Beispielszenarien in diesem Format gespeichert sind, ist es sinnvoll, diese Funktionalität in eine Oberklasse auszulagern. Dieser CSVTransformator soll seine Daten aus einer CSV-Datei sowohl auf dem lokalen Datenträger (für die Beispielszenarien) als auch von einer URL (für den Hallway-Export) beziehen können. Der Hallway-Transformator braucht dann nur noch die Adresse der lokalen Hallway-Instanz aus der Konfiguration von Play auslesen und bekommt die restliche nötige Funktionalität von seiner Oberklasse vererbt. Für den Beispielszenario-Transformator reicht die Angabe des Dateipfads zum Beispielszenario.

6.3.4 Weitere Transformatoren

Der Twitter-Transformator soll als eine Anbindung an ein großes populäres DSN mitgeliefert werden. Der Transformator für die Szenarien dient dazu neben den fest gespeicherten Beispielszenarien noch beliebig viele neue Durchläufe des Simulators machen zu können und deren Ergebnis für die Visualisierungen einzulesen.

7 Implementierung

Nach Erstellung der Mockups und eines groben Entwurfs kann jetzt mit der Programmierung begonnen werden. Um den Aufwand zu senken wurden ein paar Annahmen gemacht.

7.1 Vereinfachende Annahmen

Es wird angenommen, dass ein Nutzer eine Nachricht zu dem Zeitpunkt liest, an dem er sie zum ersten Mal erhalten kann. D.h. er liest sie direkt am Zeitpunkt der Erstellung, falls der Nutzer dem Autor folgt. Er liest sie zum Zeitpunkt ihrer Weiterempfehlung, falls er dem Autor nicht folgt, aber dem Weiterempfehler. Diese Vereinfachung wurde vorgenommen, da es technisch sehr schwierig ist festzustellen, wann und ob ein Nutzer eine für ihn sichtbare Nachricht liest. So gibt es zwar in neueren Versionen von Hallway ein Protokoll der Seitenaufrufe mit Parametern, jedoch bedeutet das Aufrufen einer Seite mit einer Nachricht nicht zwangsläufig, dass der Nutzer diese Nachricht liest. Zusätzlich gibt es bei allen anderen öffentlichen DSN kein detailliertes Protokoll der Seitenaufrufe.

Das Löschen eines Nutzers wird in der Visualisierungssoftware weder bei der Simulation für die Beispielszenarien, noch in den Transformatoren beachtet. Begründung dafür ist, dass Hallway selbst seinen Nutzern keine Möglichkeit gibt ihr Nutzerkonto zu löschen. Es gibt lediglich für Administratoren die Möglichkeit dies zu tun.

Es war nötig bei Folge-Tupeln (Followship) und Nutzern (User) im Datenmodell von Hallway die Zeitpunkte ihrer Erstellung hinzuzufügen. Denn die Visualisierung 4.2 benötigt den genauen Zeitpunkt der Accounterstellung für die Berechnung des Alters des Zugangs. Zusätzlich werden die Folgezeitpunkte benötigt, um herauszufinden, wann ein Nutzer eine Nachricht liest. Ansonsten waren bereits alle benötigten Daten im Modell von Hallway explizit oder implizit enthalten. Die impliziten Daten wurden in der Anwendung häufig durch statische Funktionen der Klasse `Helper` in explizite umgewandelt. Zum Beispiel befinden sich alle Empfehlungs-Tupel und Nachrichten im Datenmodell von Hallway. Die Anzahl der Weiterempfehlungen einer Nachricht wird aber nicht direkt gespeichert. Um die Anzahl zu bestimmen, müssen die Empfehlungs-Tupel, welche die Nachricht enthalten, gesammelt und gezählt werden. Das wäre dann Aufgabe einer Funktion von `Helper`.

7.2 Zum Testen

Neben den im vorletzten Kapitel gezeigten Mockups werden in DSNViz bis auf ein paar kleine Unit-Tests für die wichtigsten Klassen keine weiteren Tests erstellt. Oberflächentests wären im Idealfall ebenfalls zu erstellen. Sie wurden nach einer Kosten-Nutzen-Abwägung ausgelassen und wären bei Weiterentwicklung der Anwendung hinzuzufügen.

Der umgesetzte Simulationsalgorithmus befindet sich in der Klasse `Simulator` im Hilfspaket `auxiliary`.

7.3 Transformatoren

7.3.1 Anbindung für Hallway

Für die Verwendung von Daten aus dem DSN Hallway ist es erforderlich, dieses in den Standardmodus “Follow” zu schalten, um das asymmetrische Verbindungsmodell zu verwenden. Weiterhin wird eine Hallway-Erweiterung benötigt, welche die für den Hallway-Transformator in DSNViz erforderlichen Daten erst aus dem Modell extrahiert. Dieser Pfad ist dem Transformator anzugeben. Dort liegt die Datei `export.csv`, welche das Protokoll der Nutzeraktionen als Liste aus durch Komma getrennte Werte enthält. Das Format dieser Datei wurde im Kapitel 5 beschrieben.

Nachdem durch eine CSV-Exporter-Erweiterung für Hallway die Ereignisse aus Hallway als CSV-Datei exportiert worden sind, können diese vom CSV-Transformator in DSNViz eingelesen und verwendet werden.

An dieser Stelle sei darauf hingewiesen, dass weiterempfehlbare[1] Nachrichten nicht Nachrichten, sondern Aktivitäten genannt werden. Nachrichten sind in Hallway persönliche Nachrichten von einem Nutzer zu einem anderen. Aktivitäten können jedoch nicht nur durch das Textfeld “Woran arbeiten sie?” (UserActivity) in Hallway erzeugt werden, sondern bspw. auch wenn ein Nutzer ein Dokument anlegt (UserActivityAtDocument). Daher sind die weiterempfehlbaren Nachrichten also nur eine echte Teilmenge der Aktivitäten. Zur Vereinfachung des Exporters wird hier aber jede Aktivität mit einer Textlänge von mindestens einem Zeichen als eine Nachricht interpretiert.

7.3.2 Anbindung für Szenarien

Um die Architektur von DSNViz nicht zu verletzen, werden die Daten des Simulators ebenfalls über einen Transformator beschafft. Im Gegensatz zu allen anderen Transformatoren (mit Ausnahme des experimentellen Transformators auf Basis eines Zufallsgenerators) liefert dieser keine Daten aus einem echten DSN oder einem Archiv. Stattdessen stammen diese aus einer Simulation nach Vorgaben, welche in 5 beschrieben

[1] Weiterempfehlung heißt in Hallway “like”.

worden sind. Es gibt bei der Verwendung von Szenarien in DSNViz dabei folgende zwei Wahlmöglichkeiten:

- Der `ScenarioTransformator` kann die Daten aus einer neuen Simulation nach einem der beiden Regelsätze gewinnen.
- Der `ExampleScenarioTransformator` kann die drei im CSV-Format vorliegenden Beispielszenarien auslesen.

Dabei erbt der `ExampleScenarioTransformator` seine Fähigkeit des Datenimports aus CSV-Dateien vom `CSVTransformator`. Dies wurde bereits im letzten Kapitel motiviert.

7.3.3 Anbindung für Twitter

Bei der Erstellung eines Transformators für Twitter gab es ein besonders anspruchsvolles Problem: Aufgrund einer technischen Beschränkung der maximalen Anzahl an API-Aufrufen in einer Stunde auf 150, bzw. 350 für authentifizierte Nutzer, ist dieses DSN als Datenquelle für diese Arbeit nicht sonderlich gut geeignet. Eine deutliche Erhöhung dieser Anzahl lässt sich über eine Anfrage an Twitter zum sogenannten “Whitelisting” erreichen. Der Autor der Arbeit erzielte hiermit jedoch leider keinen Erfolg. Dennoch gab es Gründe zur Entwicklung eines Transformators für dieses DSN. So ist es sehr populär und es gibt einige weitere DSN (wie bspw. Identi.ca), welche eine mit der Twitter-API kompatible API anbieten. Diese können durch den Twitter-Transformator also ebenfalls angesprochen werden.

Aufgrund der genannten Beschränkung in den API-Aufrufen ist der Transformator darauf optimiert, möglichst wenige dieser Aufrufe zu benötigen. Daher liest er nur die Folgerelation von einem angegebenen zentralen Nutzerknoten zu den Nutzern hin, denen er folgt. Zusätzlich betrachtet er, wie diese Nutzer sich untereinander folgen. Es werden nur die neusten Nachrichten der so im Graphen enthaltenen Nutzer ausgelesen. Für Nutzer mit “whitegelisteter” IP-Adresse sollte eine Option eingefügt werden, größere Datenmengen zu beziehen.

7.3.4 Zufalls-Transformator

Der letzte Transformator wird zwar mitgeliefert, ist aber nicht eigentlicher Bestandteil der Arbeit. Er ist der einfachste Transformator der Anwendung und daher als Referenzimplementierung der Schnittstelle anzusehen. Der Zufalls-Transformator erzeugt rein zufällige Daten. In den Visualisierungen wird sich daher bei Verwendung dieses Transformators kein Phänomen, sondern lediglich ein statistisches Rauschen zeigen.

7.3.5 Eigene Transformatoren

Bei der Erstellung eines neuen Transformators kann der Zufalls-Transformator als Vorlage verwendet werden. In jedem Fall muss eine neue Klasse im bereits vorhandenen Paket `transformators.custom` erstellt werden und die `ITransformator`-Schnittstelle implementieren. Alternativ kann sie von `TransformatorAdapter` erben. Der neue Transformator ist nach Fertigstellung dieser Klasse über die Adresse `http://localhost:9001/updateModel?src=transName` zu erreichen, wobei `localhost` durch den korrekten Hostnamen und `transName` durch den Namen der Klasse des neuen Transformators ersetzt werden müssen.

7.4 Implementierung der Visualisierung

Im Folgetext wird die Umsetzung der im vorherigen Kapitel genannten Visualisierungen in Java (Berechnungen) und JavaScript (Zeichnen und Layout) vorgenommen. Die Semantik folgt direkt aus den Erläuterungen in Kapitel 4. Lediglich geringfügige Abwandlungen aufgrund von technischen Beschränkungen werden hier beschrieben. Der komplette Quelltext befindet sich auf dem mitgeliefertem Datenträger (vgl. Kapitel “Compact Disc” im Anhang) in `./app/controllers/Visualizations.java` sowie dem Ordner `./app/views/`.

Nachdem die Daten von einem Transformator beschafft worden sind, werden diese vom Controller `Visualizations` ausgelesen und für die Visualisierungen aufbereitet. Es wird dabei die Hilfsklasse `Helper.java` vielfach benutzt, welche statische Funktionen für diverse Berechnungen bereit stellt. Genau betrachtet gibt es in `Visualizations` für jede Visualisierung eine Funktion, welche von der entsprechenden Route aufgerufen wird. Sind die Daten genug aufbereitet, werden diese schließlich an den passenden View übergeben. Für jede Visualisierung gibt es einen speziellen View, welcher aus HTML, JavaScript für Protovis und Play-Template Elementen besteht. Die Play-Template Elemente dienen dem Auslesen von durch den Controller übergebenen Daten und erzeugen weiterhin dynamisch HTML und JavaScript-Code, um hier entsprechend den Daten angepassten Code zu liefern.

7.4.1 Protovis

Das eigentliche Zeichnen wird dann von der JavaScript Bibliothek Protovis[2] übernommen, welches die deskriptiv beschriebenen Diagramme in ein HTML-Canvas schreibt. Da HTML-Canvas selbst SVG im Element enthalten, lässt sich dies auch einfach mittels einer JavaScript-Operation auslesen und zur Weiterverwendung extern speichern.

[2] Die Visualisierungsbibliothek Protovis ist erhältlich auf http://vis.stanford.edu/protovis/ und wird ausführlich in [Bostock und Heer ’09] motiviert, beschrieben und mit Alternativen verglichen.

Für die Verwendung von Protovis sprachen die folgenden Argumente:

1. Es ist aufgrund des deklarativen Programmierstils, vieler Beispiele und einer guten Dokumentation einfach erlernbar.
2. Man ist nicht auf graphische Primitive beschränkt, sondern erhält bereits fertige Layouts geliefert.
3. Die Ergebnisse lassen sich einfach in SVG exportieren.

7.4.2 Hauptvisualisierungen

Visualisierung 4.1 Eine besondere Schwierigkeit bei dieser Visualisierung war die Existenz zweier Abszissen. Entsprechend mussten die Koordinaten der Punkte der zweiten Datenreihe mit entsprechenden Faktoren bearbeitet werden, damit sie zu ihrer Skala passten.

Visualisierung 4.2 Hier wurde nach einer Aufwandsabschätzung die Entscheidung getroffen, das interaktive Element dieser Visualisierung nicht umzusetzen. Der Nutzer kann also nicht über einen Mausklick auf einen Nutzer sich die Zeit anzeigen lassen, welche die Nachricht benötigte, bis sie von ihm gelesen worden ist. Stattdessen wird für jeden Knoten diese Zeit durchgehend in Klammern nach dem Knotennamen dargestellt. Die Zeit wird in Millisekunden (10^{-3}s) angezeigt.

Visualisierung 4.3 Um die einzelnen Verbreitungsbäume der Nachrichten in dieser Visualisierung korrekt von Protovis darstellen zu lassen, war es notwendig für Nutzer, welche mehrfach vorkommen, neue Namen zu erzeugen. Diese werden durch sukzessives Anhängen eines Unterstrichs gebildet. Dadurch wird sichergestellt, dass kein Knotenname mehrfach vorkommt.

Visualisierung 4.4 Während sich Sprünge und Sackgassen einfach umsetzen ließen, war die Anzeige für Bremsen und Beschleuniger am Rande eines jeden Knotens deutlich umständlicher zu implementieren. In bestimmten Vergrößerungsstufen sind die dafür gezeichneten Rechtecke so groß, dass sie den Knotennamen verdecken. Ansonsten wurde aufgrund des recht hohen Aufwands bei geringem Nutzen die Einfärbung der Anzeige für die Weiterempfehlungsrate (Bremsen/Beschleuniger) neben jedem Knoten weggelassen.

Visualisierung 4.5 Diese Visualisierung ließ sich am einfachsten schreiben. Der größte Aufwand bestand darin die Zeitabstände von Millisekunden in die bestimmten Klassen hinein zu sortieren.

7.4.3 Zusätzliche Visualisierungen

DSNViz enthält noch eine weitere Visualisierung, welche nicht im Rahmen der Arbeit entworfen ist, mit dem Namen “simpleGraph”. Sie stellt die Topologie des DSN in einem Graphen dar.

Aufgrund der durch das MVC-Muster erzwungenen Trennung von Daten (Model) und Darstellung (View) ist eine Erweiterung der Anwendung um zusätzliche Visualisierungen besonders einfach möglich. Angenommen wir wollten eine neue Visualisierung mit Namen “flowVisualization” erstellen. Dann bräuchten wir nur eine neue Funktion mit diesem Namen in den Controller `Visualizations` sowie ein View mit Namen `flowVisualization.html` zu den Visualisierungs-Views[3] hinzufügen. Automatisch wird dann eine Verlinkung auf die neue Visualisierung auf der Hauptseite von DSNViz vorgenommen. Die passende Route wird ebenfalls von selbst generiert. Der Controller sollte dann entsprechend mit Code gefüllt werden, welcher Daten aus dem Modell ausliest und mit Hilfe ggf. ebenfalls zu schreibender Hilfsfunktionen diese für die Visualisierung aufbereitet und letztlich als Parameter in einem `render()`-Aufruf dem View übergibt.

7.5 Integration

Die Visualisierungen erhalten über den Transformator für Hallway nun ihre Eingabedaten und sind damit im Kontext der eigenständigen Play-App bereits verwendbar. Besonders benutzerfreundlich ist dies jedoch nicht. Eine leichtere Anwendung durch die Hallway-Nutzer erfordert einen Zugang zu diesen Visualisierungen aus der Hallway-Play-App selbst heraus. Dies umfasst ebenfalls einen Aufruf der Exportfunktion, welche die für DSNViz bereitgestellte CSV-Datei mit den Daten aktualisiert.

Daher wurden Erweiterungen vorgenommen: Das View `home_hallway.html` wurde um einen zusätzlichen Navigationseintrag “Visualisierungen” erweitert, welcher eine Verlinkung enthält, die einen neu hinzugefügten Controller “Visualizations” aufruft. Dies geschieht wie in Play üblich über eine Route von der URL auf den Controller, welche dafür in die routes.conf hinzugefügt worden ist. Der neue Controller lässt dann ein neues View `Visualizations/index.html` anzeigen, welches DSNViz in einem iframe anzeigt. Durch die Einbettung in einem Inlineframe kann darauf DSNViz ohne weitere Veränderungen an Hallway uneingeschränkt verwendet werden.

[3] Diese befinden sich alle im Ordner `app/views/Visualizations`

7.6 Benutzung von DSNViz

7.6.1 Einrichtung und Voraussetzungen

Zur Verwendung von DSNViz ist es erforderlich ein JRE[4] installiert zu haben sowie das Play-Framework[5]. Dann lässt sich DSNViz starten, indem der Befehl `play run` im Verzeichnis von DSNViz aufgerufen wird. Unter Windows genügt auch ein Klick auf die Stapelverarbeitungsdatei `start.bat` im DSNViz-Verzeichnis, welche genau diesen Befehl aufruft. In jedem Fall ist es natürlich nötig das Verzeichnis der entpackten Play-Distribution in der PATH-Umgebungsvariable des Systems zu speichern[6], weil ansonsten der Befehl `play` nicht korrekt aufgelöst werden kann.

Zur Visualisierung von Daten aus Hallway ist eine mit den hier vorgestellten Erweiterungen angepasste Hallway-Instanz vonnöten, deren URL in der Konfigurationsdatei `application.conf`[7] von DSNViz unter `url.hallway=...` angegeben ist.

Eine modifizierte Variante von Hallway mit Namen “HallwayViz” wird mitgeliefert. Die dort veränderten Dateien und hinzugefügten Dateien sind in der ebenfalls mitgelieferten Datei `changedFiles.txt` dokumentiert. In der `application.conf` von HallwayViz ist analog die URL von DSNViz in der Konstante `url.dsnviz=...` anzugeben. Für die Verwendung von HallwayViz ist zu beachten, dass vorher ein MySQL-Datenbankserver zu starten ist, dessen Informationen unter `db.url, db.user` sowie `db.pass` in die `application.conf` von HallwayViz einzutragen sind. Zusätzlich muss auf diesem MySQL-Server eine Datenbank mit Namen “Hallway” erstellt werden. Dies kann bspw. durch die SQL-Anfrage `create database Hallway;` erfolgen. Danach kann HallwayViz analog zu DSNViz durch den Aufruf `play run` gestartet werden.

Zum Testen bietet es sich dann noch an unter `/setup` den von Hallway zur Verfügung gestellten Fakedatensatz (fake Only) zu erstellen. Dieser wurde um Nachrichten, Kommentare und Empfehlungen erweitert. Der für die Autorisierung nötige Geheimschlüssel befindet sich in der Datei `application.conf` unter `application.secret`.

7.6.2 Anwendungsfunktionen

Ist die Anwendung nun gestartet (und je nach Bedarf auch Hallway), lässt sie sich über die Adresse des Servers mit angehängter Portnummer 9001 aufrufen. Bspw. wäre die Adresse bei lokaler Verwendung `http://localhost:9001`. Nun muss man im linken Bereich erst eine Datenquelle auswählen.

Sind die Daten von einem ausgewählten Transformator eingelesen, lassen sie sich in den Visualisierungen verwenden. Entsprechend wird auf der rechten Seite ein Aus-

[4] http://www.oracle.com/technetwork/java/javase/downloads/index.html
[5] http://www.playframework.org/download
[6] Dies funktioniert bspw. in Unixoiden-Systemen häufig durch zusätzliches Hinzufügen von “`export PATH=pfad/zu/play:$PATH`”.
[7] enthalten in `DSNViz/conf`

wahlbereich für diesen Zweck sichtbar. Der Nutzer kann nun die gewünschte Visualisierung anklicken, woraufhin die Daten durch Berechnungen aufbereitet und danach in den Visualisierungen veranschaulicht werden.

Eine Verlinkung in Form eines blauen Hauses in jedem View gibt dem Nutzer zu jeder Zeit die Gelegenheit wieder auf die Startseite von DSNViz zurück geleitet zu werden.

Beenden lässt sich DSNViz durch versenden des `SIGINT`-Signals. Dies lässt sich am einfachsten durch Drücken der Tastenkombination `CTRL+C` im Terminal, in welchem der Play-Prozess im Vordergrund läuft, bewerkstelligen.

8 Auswertung

Es werden nun die Ergebnisse der Visualisierungen mit den Mockups aus Kapitel 5 verglichen. Die Abbildungen der Visualisierungsergebnisse sind auf den nachfolgenden beiden Seiten zu finden.

Visualisierung 4.1 Visualisierungsergebnis und Mockup stimmen weitestgehend bis auf eine Kleinigkeit überein: Die Ordinate ist bei der umgesetzten Visualisierung nicht explizit gezeichnet worden. Der entsprechende Wert ist jedoch aufgrund der an der linken Seite am grauen Gitter angebrachten Skala abzulesen.

Visualisierung 4.2 Hier gleicht das Ergebnis in beiden Szenarien den Mockups im wesentlichen Punkt: Der Topologie. Das Layout hingegen ist unterschiedlich. Im Mockup wurde die Wurzel wie bei Bäumen üblich ganz oben eingezeichnet, wohingegen in der Visualisierung die Wurzel zentrale Knoten eines Graphen ist. Dieser Unterschied ist durch die Verwendung des Force Directed Layout in der Umsetzung zu erklären. Leider gibt es in Protovis für Bäume kein geeignetes Layout. Die unterschiedliche Anordnung rechtfertigt aber meiner Meinung nach auch nicht den Lernaufwand durch den Wechsel auf eine andere Visualisierungsbibliothek an dieser Stelle.

Visualisierung 4.3 Bei dieser Visualisierung entspricht das Ergebnis bis auf geringfügige Abweichungen im Layout der Komponenten des Graphen dem Mockup. Dass der Knotenname im Mockup innerhalb des Knotens und im Visualisierungsergebnis neben dem Knoten angezeigt wird, verändert nichts an der Aussagekraft der Visualisierung und ist ein akzeptabler Unterschied.

Visualisierung 4.4 Da das Mockup bereits recht grob war, ist der Vergleich mit dem Ergebnis hier noch unter Vorbehalten auszuwerten. Die Sprünge und Sackgassen sind zumindest identisch. An dieser Stelle würde ein weiteres Beispielszenario mit weniger Nutzern ($\tilde{4}$-5), in dem alle durch die Visualisierung aufzeigbare Phänomene vorkommen, eine genauere Auswertung der Visualisierung ermöglichen.

Visualisierung 4.5 Hier entspricht das Visualisierungsergebnis wieder dem Mockup. Die Anordnung der Rechtecke für die einzelnen Klassen ist unterschiedlich. Das trifft ebenso auf das Layout der Textbeschriftungen der Klassen zu. Da aber die Verhältnisse der Flächen gleich sind, sind diese Unterschiede nicht relevant.

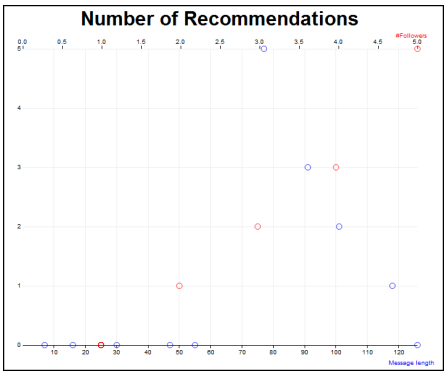
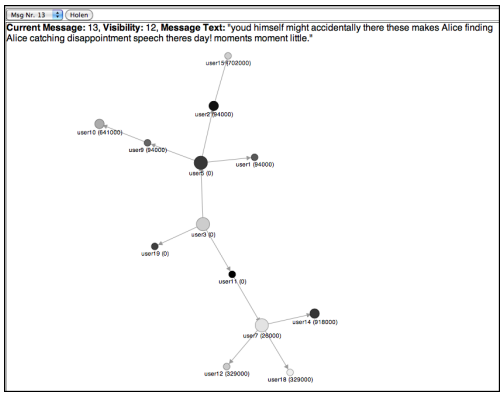
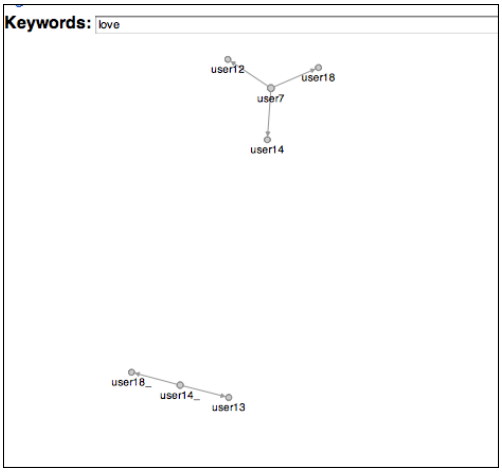


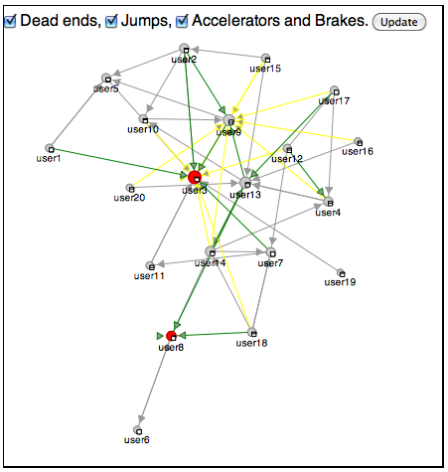
Abbildung 8.1: Ergebnis zu 4.1 für Beispielszenario 3



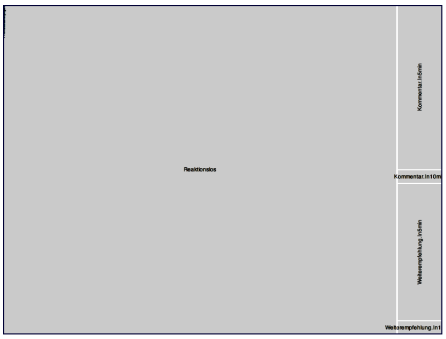
(a) Ergebnis zu 4.2



(b) Ergebnis zu 4.3

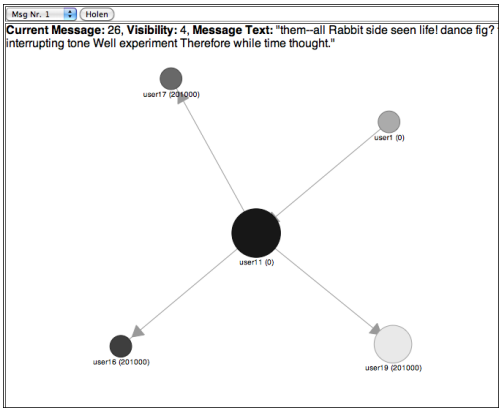


(c) Ergebnis zu 4.4

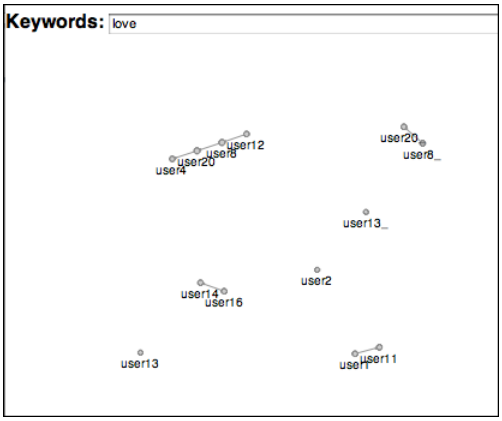


(d) Ergebnis zu 4.5

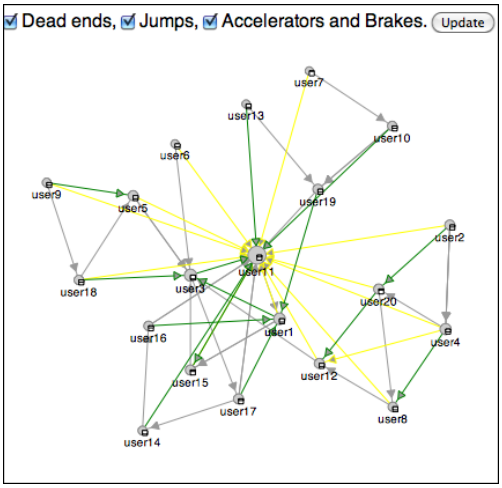
Abbildung 8.2: Visualisierungsergebnisse zu Beispielszenario 1



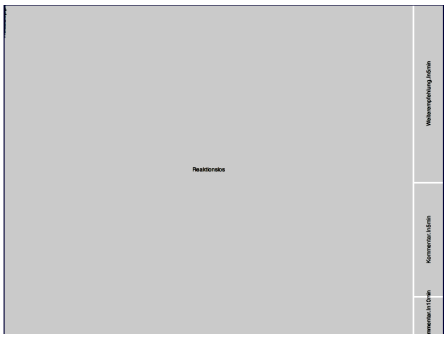
(a) Ergebnis zu 4.2



(b) Ergebnis zu 4.3



(c) Ergebnis zu 4.4



(d) Ergebnis zu 4.5

Abbildung 8.3: Visualisierungsergebnisse zu Beispielszenario 2

9 Verwandte Arbeiten

Die meisten Arbeiten zur Visualisierung von sozialen Netzwerken betrachten nicht die Nachrichten, welche in diesen fließen. Häufig wird nur der Graph selbst dargestellt und analysiert. Typische Eigenschaften, die dann betrachtet werden, ergeben sich meist aus der Struktur des Graphen und die dazu getätigten Betrachtungen liegen entsprechend auf der Schnittstelle zur Graphentheorie.

Zur Nachrichtenverbreitung durch Weiterempfehlung gibt es insbesondere Aufgrund der Relevanz von Twitter einige Dokumente.

Die Visualisierungen stellen in den nachfolgenden Artikeln lediglich ein Werkzeug zur besseren Auswertung der in den DSN gesammelten Daten dar. Das ist in dieser Arbeit ganz anders. Hier sind die Visualisierungen zur Informationsausbreitung das Ergebnis und nicht die durch ihre Verwendung bestätigte Hypothesen. Zusätzlich werden in den Artikeln im Ggs. zu dieser Arbeit nur bestimmte Aspekt der Nachrichtenausbreitung betrachtet.

Die betrachteten Aspekte der Nachrichtenausbreitung sowie die ggf. bereits daraus gewonnenen Ergebnisse der verwandten Arbeiten werden nun kurz aufgelistet:

- In [Ratkiewicz et al. '10] geht es im Rahmen des Forschungsprojekt "Truthy"[1] darum, einen Webdienst zu entwickeln, welcher politische Schmutzkampagnen in Twitter aufdeckt. Es geht also speziell darum Nachrichten zu verfolgen, welche Teil solcher Kampagnen sind. Als Ergebnis werden beispielhaft korrekt vom Dienst ermittelte Kampagnen vorgestellt.
- In [Lee et al. '10] spielt die Nachrichtenausbreitung nur insofern eine Rolle, als dass sie Rückschlüsse auf den Einfluss, den Nutzer auf andere Nutzer ausüben, zulässt. Hierfür wurden große Datenmengen aus Twitter entnommen und die Verbreitung von Nachrichten zu einem aktuellen beliebten Thema betrachtet. Als Resultat wird eine Methode zum Auffinden der einflussreichsten Twitter-Nutzer vorgestellt. Eine Auflistung von zehn mit dieser Methode ermittelten Nutzern wird zusätzlich angegeben.
- [Romero et al. '10] sammeln ebenso Daten aus Twitter. Hier liegt der Schwerpunkt der Analyse auf der Entwicklung eines Algorithmus, welcher den Einfluss und die Passivität von Nutzern in Abhängigkeit von ihrer Weiterempfehlungsaktivität ermittelt. Es stellt sich heraus, dass die Korrelation zwischen Beliebtheit und Einfluss eines Nutzers geringer ist als erwartet.
- [Ye und Wu '10] betrachten speziell die Ausbreitung von Eilmeldungen.

[1] http://truthy.indiana.edu/

- [Magnani et al. '10] haben einen Fokus auf Friendfeed, einem Dienst, welcher Informationen aus verschiedenen DSN aggregiert.
- In [Suh et al. '10] und [Zarrella '09] wird analysiert, welchen Einfluss die Eigenschaften von Nachrichten und deren Autoren auf das Weiterempfehlungsverhalten anderer Nutzer haben.
- Der ausführliche Artikel von [Kwak et al. '10] nutzt gesammelte Daten aus Twitter zur Beantwortung der Frage, ob es sich bei diesem Dienst um ein klassisches DSN handelt oder eher um ein Mittel zur Verbreitung von Nachrichten.

Wegen der Popularität dieses DSN stammen auch einige Dokumente zu dem Thema aus dem Bereich des Marketings (vgl. [Zarrella '09]). Die Verbesserung der Verteilung von Werbeinhalten über Nachrichten in DSN ist kein Thema dieser Arbeit. DSNViz könnte jedoch die Umsetzung von Visualisierungen zu diesem Vorhaben deutlich erleichtern (siehe 7.4.3).

10 Fazit und Ausblick

In der vorliegenden Arbeit wurden Visualisierungen für Phänomene der Informationsausbreitung in DSN konzeptionell entworfen und im Rahmen einer neu entwickelten Visualisierungsanwendung umgesetzt. Es folgt nun eine kritische Würdigung der Arbeit sowie ein Ausblick, welcher mögliche Folgearbeiten und in dieser Arbeit aus Platzgründen nicht weiter verfolgte Vertiefungen skizziert. Abschließend folgt eine rückblickende Zusammenfassung.

10.1 Kritische Würdigung

Bei der Planung der Arbeit und bei der Literaturrecherche wurde der Rahmen anfänglich zu weit gefasst, was auch bei einem Zwischenvortrag der Arbeit deutlich wurde.

Nach Verfeinerung der Problemstellung und nach Erstellung eines detaillierten Aufgabenplanes sowie einer neuen selektiven Literaturrecherche konnte die Arbeit enger am Thema orientiert umgesetzt werden.

Es wurde in dieser Arbeit eine flexible Visualisierungssoftware erstellt, welche durch die mitgelieferten Anbindungen für Hallway und Twitter eine Verwendung mit dieser DSN erlaubt.

Diese Visualisierungssoftware kann auch für weitere DSN verwendet werden, wobei lediglich eine Anbindung programmiert werden muss.

10.2 Ausblick

Das Ereignis ”Lesen einer Nachricht/Kommentar“ wurde anfänglich für DSNViz beachtet, da sich dies theoretisch aus den Protokollen von Hallway extrahieren lassen müsste. Jedoch stellte sich der Implementierungsaufwand hierfür sowie für die Simulation dieses Ereignisses in realistischer Weise als unerwartet hoch heraus. Daher wird dieses Ereignis nicht mehr explizit von den Transformatoren geliefert sondern für die Visualisierungen implizit erschlossen. Gibt es von einem Nutzer eine Folge-Beziehung vom Autor oder Weiterempfehler einer Nachricht, so wird angenommen, dass dieser Nutzer die Nachricht sofort nach der Erstellung durch den Autor liest. Dies ist zwar in der Realität nur möglich, falls alle Nutzer ihre Schlafzyklen synchronisiert haben und über neue Nachrichten sofort per Push-Notifikationen informiert werden. Nichtmenschliche Informationsquellen (bspw. Systemnachrichten) dürften zu den Schlafzeiten der Nutzer ebenfalls keine Nachrichten senden. Zugegebenermaßen ist dies

ein sehr künstliches Szenario. Daher wäre es hilfreich ein realistischeres Modell zu entwickeln.

Der Hallway-Exporter betrachtet das Protokoll nur für spätere Ereignisse nach seiner Initialisierung. Ansonsten entnimmt er die Nutzerlisten, Folgerelation, Empfehlungsrelation sowie Nachrichten direkt aus dem Modell von Hallway. Der Nachteil daran ist, dass alle Daten als statisch angesehen werden. Bspw. ist die implizite Lesezeit falsch, wenn eine Nachricht geschrieben wurde und ein Nutzer dem Autor zu diesem Zeitpunkt noch nicht gefolgt ist, es aber später tut. Denn es wird als implizite Lesezeit für die Nachricht deren Erstellungsdatum genommen und nicht der Zeitpunkt, als der Leser dem Autor zuerst gefolgt ist.

In Hallway gibt es vom System gesteuerte Informationsquellen. Diese nichtmenschlichen Informationsquellen werden beim Hallway-Export als Nutzer interpretiert. Das ist nicht so gut. Idealerweise wäre in Visualisierungen darauf hinzuweisen, was menschlicher Nutzer und was Maschine ist.

In der Simulation werden bestimmte Ereignisse wie bspw. das Löschen einer Nachricht nicht erzeugt. In realen DSN aber gibt es solche Ereignisse und sie haben Auswirkungen auf die Ausbreitung von Nachrichten. Zusätzlich könnten weitere Ereignisse für neue Visualisierungen in DSNViz von Bedeutung sein. Bspw. speichert Hallway den Ort eines Nutzers explizit ab. Das machen auch viele andere DSN. Daher sollte das zugehörige Ereignis vom CSV-Exporter für Hallway mit exportiert werden und durch die Simulation erzeugt werden. Entsprechend wären die folgenden Ereignisse zu integrieren:

- Löschen einer Nachricht.
- Löschen des Nutzerkontos.
- Ändern des Ortes eines Nutzers.

Der mitgelieferte Transformator für Twitter ist aufgrund von Zeitgründen unfertig geblieben. Andere Teile der Arbeit hatten aufgrund der Aufgabenstellung eine höhere Priorität.

Obwohl durch den Einsatz einer Schnittstelle für das Auslesen der Daten aus einem DSN die entwickelte Visualisierungsanwendung einfacher mit anderen DSN als Hallway verwendbar sein soll, wurden zur Verringerung des Programmieraufwands einige Annäherungen an konkrete Details von Hallway unternommen. So gibt es bspw. in Twitter keine Kommentare. Stattdessen kann man andere Nutzer ansprechen und antworten, indem der Nutzername des Adressaten mit voran gestelltem @-Zeichen referenziert wird. Da im Falle einer Antwort Twitter in den Metadaten ebenso eine Referenz auf die Nachricht, auf welche sich die Antwort bezieht, gespeichert wird, könnte man dies als Kommentar interpretieren. Entsprechende Modifikationen im Transformator wären vonnöten.

Die Visualisierungsbibliothek Protovis stellte sich als zu langsam für Graphen mit großer Anzahl an Knoten und Kanten (ca. gegen 1000) heraus. Hier könnte man noch optimieren, indem man beim Force Directed Layout die interaktiven Zieh-Operationen herausnimmt. Ansonsten wäre eine Behebung der mangelnden Skalierbarkeit durch

einen Wechsel auf eine Visualisierungsbibliothek auf tieferer Software-Ebene zu erwägen. So könnte man mit Raphaël[1] eine Generalisierung des Graphen bei ausreichender Verkleinerung implementieren. Ansonsten wäre sicherlich die effizienteste Implementierung eine eigenständige C-Anwendung unter Verwendung der hardwarebeschleunigten Grafikbibliothek OpenGL. Für OpenGL gibt es auch eine Vielzahl an Programmierschnittstellen wie z.B. OpenGL Performer. Im Controller könnte diese Anwendung dann aufrufen und die Visualisierungsdaten über eine Datei übergeben. Das Zeichenprogramm könnte dann das Ergebnis in eine Bilddatei schreiben, welche wiederum in das View eingebunden wird.

10.3 Zusammenfassung

Aufgabe dieser Arbeit war es, Visualisierungen zur Informationsausbreitung in DSNen allgemein und in Hallway speziell zu entwerfen und danach umzusetzen. Der Entwurf sollte dabei besonders auf Basis von bestehenden Notationen aus der Literatur geschehen.

Dafür mussten ganz am Anfang die notwendigen Begrifflichkeiten und Sachverhalte aufgelistet werden. Dies beinhaltete eine Erklärung der für die Informationsverbreitung relevanten Konzepte der Weiterempfehlung und Kommentare. Darauf wurde in der Literatur nach Phänomenen und Visualisierungsansätzen zum Thema dieser Arbeit gesucht. Die Ergebnisse dieser Suche wurden zusammen mit eigenen Ideen in eine Auflistung der Phänomene und Visualisierung überführt. Drei beispielhafte Szenarien wurden entwickelt und die Visualisierungen auf Basis dieser Szenarien als Mockups skizziert. Darauf ist der geplante grobe Aufbau der Visualisierungsanwendung, welche den Unterbau für die Umsetzung der Visualisierungen liefert, vorgestellt worden. Es folgten danach einige knappe Erläuterungen und Bemerkungen zur Umsetzung. Die Visualisierungen im fertigen Programm wurden dann im Vergleich mit den Mockups ausgewertet. Es gab leichte Abweichungen in kleinen Details aber nichts, was eine Fehlinterpretation hervorgerufen hätte. Auf den Bezug der in der Literaturrecherche gefundenen Artikel zu dieser Arbeit wurde danach noch etwas näher eingegangen.

Es stellte sich heraus, dass zur Visualisierung 4.2 eine Erweiterung des Datenmodells von Hallway nötig war. Nämlich mussten die Klassen `User` und `Followship` um ein Attribut erweitert werden, welches den Zeitpunkt speichert, an welchem der Nutzerzugang angelegt worden ist bzw. das Folge-Tupel erstellt worden ist. Dafür war jedoch jede Visualisierung in der Umsetzung darauf angewiesen, benötigte Daten aus den Daten im Modell von Hallway zu errechnen. Bspw. ist der Ausbreitungsbaum einer Nachricht zu den Verfolgern des Autors und den Verfolgern der Weiterempfehlenden einer Nachricht nur implizit in der Liste der Folge-Tupel (Followship) und Empfehlungs-Tupel (LikesActivity) im Hallway-Modell enthalten.

Neben der vorliegenden Ausarbeitung wurde im Verlauf dieser Arbeit eine sehr flexible Visualisierungssoftware erstellt. Sie verwendet zur Interaktion mit dem Nutzer

[1] http://raphaeljs.com/

das Play-Framework und für das Zeichnen in den bereits vorhandenen Visualisierungen die Protovis-Visualisierungsbibliothek. Die beiden mitgelieferten Anbindungen für Hallway und Twitter erlauben eine Verwendung der Software mit diesen beiden DSN. Für die Benutzung mit einem anderen DSN ist es lediglich nötig eine entsprechende Anbindung zu programmieren. Die Schnittstelle dafür und was in dieser bereit gestellt werden soll, wurde in 7.3.5 beschrieben. Die im Kapitel 4 gelisteten Visualisierungen gehören zum mitgelieferten Repertoire der Software. Zusätzliche Visualisierungen könnte man bei Beachtung von 7.4.3 ebenso einfach hinzufügen. Allein mit bereits explizit in Hallway gespeicherten Daten kommt nur die zusätzlich hinzugefügte Visualisierung `simpleGraph` aus.

Literaturverzeichnis

[Bostock und Heer '09] Michael Bostock and Jeffrey Heer,
Protovis: A Graphical Toolkit for Visualization,
Washington, IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis), 2009.

[Boyd und Ellison '07] Danah M. Boyd and Nicole B. Ellison,
Social Network Sites: Definition, History, and Scholarship,
Online, Journal of Computer-Mediated Communication, 2007.

[Chen '10] Chaomei Chen,
Information visualization,
New York, Wiley Interdisciplinary Reviews: Computational Statistics, 2010.

[Kim '10] Bob Wan Qi Kim,
Twitter Data Visualization: The Real Future of Twitter (that only Investors and Businessmen will like),
http://journik.posterous.com/somebody-should-build-a-twitter-rt-visualizat, 2009,
Letzter Abruf: 19.01.2011.

[Kwak et al. '10] Haewoon Kwak, Changhyun Lee, Hosung Park, Sue Moon,
What is Twitter, a social network or a news media?,
New York, ACM Press (WWW10), 2010.

[Lee et al. '10] Changhyun Lee, Haewoon Kwak, Hosung Park, Sue Moon,
Finding influentials based on the temporal order of information adoption in twitter,
New York, ACM Press (WWW10), 2010.

[Magnani et al. '10] Matteo Magnani, Danilo Montesi, Luca Rossi,
Information propagation analysis in a social network site,
Washington, IEEE (Conference on Advances in SNA and Mining), 2010.

[Peters '10] Maximilian Peters,
Konzeption und Implementierung eines erweiterbaren Digitalen Sozialen Netzwerks (Bachelorarbeit),
Hannover, Fachgebiet Software Engineering, Leibniz Universität Hannover, 2010.

[Ratkiewicz et al. '10] Jacob Ratkiewicz, Michael Conover, Mark Meiss, Bruno Goncalves, Snehal Patil, Alessandro Flammini, Filippo Menczer,
Detecting and Tracking the Spread of Astroturf Memes in Microblog Streams,
Ithaca, arXiv (Cornell University Library), 2010.

[Romero et al. ’10] Daniel M. Romero, Wojciech Galuba, Sitaram Asur, Bernardo A. Huberman,
Influence and Passivity in Social Media,
Ithaca, arXiv (Cornell University Library), 2010.

[Schneider et al. ’08] Kurt Schneider, Kai Stapel, Eric Knauss,
Beyond Documents: Visualizing Informal Communication,
Hannover, Fachgebiet Software Engineering, Leibniz Universität Hannover, 2008.

[Suh et al. ’10] Bongwon Suh, Lichan Hong, Peter Pirolli, Ed H. Chi,
Want to be Retweeted? Large Scale Analytics on Factors Impacting Retweet in Twitter Network,
Washington, IEEE (SocialCom Second International Conference), 2010.

[Sysomos ’10] sysomos a Marketwire Company,
Replies and Retweets on Twitter,
http://www.sysomos.com/insidetwitter/engagement/, 2010,
Letzter Abruf: 19.01.2011.

[WP:9091] Diverse Autoren,
90-9-1 Theory,
http://www.wikipatterns.com/display/wikipatterns/90-9-1+Theory,
Letzter Abruf: 19.01.2011.

[WP:Diagramm] Diverse Autoren,
Diagramm,
http://de.wikipedia.org/wiki/Diagramm,
Letzter Abruf: 19.01.2011.

[WP:ScaleFree] Diverse Autoren,
Scale-free network,
http://en.wikipedia.org/wiki/Scale-free_network,
Letzter Abruf: 19.01.2011.

[Xiang et al. ’10] Rongjing Xiang, Jennifer Neville, Monica Rogati,
Modeling Relationship Strength in Online Social Networks,
New York, ACM Press, 2010.

[Ye und Wu ’10] Shaozhi Ye, Felix Wu,
Measuring Message Propagation and Social Influence on Twitter.com,
Berlin/Heidelberg, Springer (LNCS, Volume 6430), 2010.

[Zarrella ’09] Dan Zarrella,
Science of Retweets,
http://danzarrella.com/science-of-retweets.pdf, 2009,
Letzter Abruf: 19.01.2011.

Abbildungsverzeichnis

Compact Disc

Jedem Exemplar dieser Arbeit liegt eine CD mit den folgenden Daten bei:

- die LaTeX-Quellen dieses Dokuments sowie eine daraus erzeugte Datei im PDF-Format in `./TexSrc`,
- die im Verlauf dieser Arbeit entwickelte Software *DSNViz* sowie die für sie entworfenen Beispielszenarien in `./DSNViz`,
- ein modifiziertes Hallway mit Namen *HallwayViz*, welches DSNViz in seine Oberfläche integriert hat. Es liegt in `./HallwayViz`,
- die Textdatei `changedFiles.txt`, welche die an Hallway vorgenommenen Veränderung zur Erstellung von DSNViz dokumentiert,
- Kopien der im Literaturverzeichnis genannten Dokumente (HTML mit referenzierten Dateien bzw. PDF), welche als Quellenverweis nur einen Hyperlink enthalten. Sie befinden sich in `./WebArchive`.

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Hannover, den 04.02.2011

_____

André Schnabel