



Masterarbeit

# **Ein System zur partiellen Synchronisation von Wissensbasen für dezentrale soziale Netzwerke**

von Jens Grundmann  
21. September 2015

Hochschule für Technik und Wirtschaft Berlin  
Fachbereich Wirtschaftswissenschaften II  
Studiengang Angewandte Informatik

Erstgutachter/in Prof. Vorname Name  
Zweitgutachter/in Vorname Name

# Inhaltsverzeichnis

|          |                                                           |          |
|----------|-----------------------------------------------------------|----------|
| <b>1</b> | <b>Einleitung</b>                                         | <b>2</b> |
| <b>2</b> | <b>Grundlagen</b>                                         | <b>3</b> |
| 2.1      | Funktionale und nicht funktionale Anforderungen . . . . . | 3        |
| 2.2      | Shark Framework . . . . .                                 | 3        |
| 2.2.1    | Context Space . . . . .                                   | 3        |
| 2.2.2    | Knowledge Base . . . . .                                  | 5        |
| 2.2.3    | SyncKB . . . . .                                          | 5        |
| <b>3</b> | <b>Konzeption</b>                                         | <b>5</b> |
| 3.1      | Ein erweiterter Kontext . . . . .                         | 5        |
| <b>4</b> | <b>Quellenverzeichnis</b>                                 | <b>6</b> |
| <b>5</b> | <b>Abbildungsverzeichnis</b>                              | <b>7</b> |

# 1 Einleitung

Soziale Netzwerke erfreuen sich immer mehr Beliebtheit in den letzten Jahren. Sei es Facebook oder schlicht das Forum zum Online Game, dass man gerade spielt. Der Mensch möchte sich austauschen. Allerdings spätestens seit den Enthüllungen Edward Snowdens gegen Ende Mai 2013 stellt sich hier die Frage der Sicherheit der persönlichen Daten. Die meisten sozialen Netzwerke basieren auf einer Client-Server Architektur. Dies bedeutet, dass alle Daten auf einem entfernten Server gespeichert sind. Der Nutzer hat somit keine Kontrolle bzw. nur die beschränkte Kontrolle, welche der Betreiber des sozialen Netzwerkes anbietet, wie mit seinen Daten umgegangen wird.

Dies ist Motivation für ein Umdenken. Anstatt die Daten an zentraler Stelle zu speichern verbleiben sie auf den lokalen Systemen der Benutzer. Das Client-Server Modell wird durch ein Peer to Peer Model ersetzt. Daten werden nur mit den Personen ausgetauscht, für die sie bestimmt sind. Hier ist eine Methode benötigt, welche die Daten der lokalen Systeme mit einander synchronisiert.

Ziel dieser Arbeit ist es eine Softwarekomponente zu entwickeln, die eine partiellen Synchronisation von Wissensbasen ermöglicht. Eine Wissensbasis ist dabei nichts anderes als eine Menge an Daten, die in einer bestimmten Struktur vorliegen bzw. durch eine abstrakte Darstellung beschreibbar sind. Mittels dieser abstrakte Darstellung soll die Implementierung von Chats, Foren bis hin zu Source Code Management Systemen auf einer Peer to Peer Basis vereinfacht werden. Als Grundlage dient hierzu das Shark Framework [4] von Prof. Dr. Thomas Schwotzer und die darin enthaltene SyncKB Klassensammlung. Diese ermöglicht bereits eine Synchronisation aller Daten in einer Wissensbasis. Diese soll nun dahingehend ausgebaut werden, dass Teile der Wissensbasis beschrieben werden können und nur diese beschriebenen Teile synchronisiert werden.

Zuerst muss mit einer Möglichkeit gefunden werden, wie ein Teilbereich der Wissensbasis beschrieben werden kann. Diese müssen dann zwischen den einzelnen Peers kommunizierbar und auf den lokalen Systemen der Peers persistierbar sein. Die durch diese Beschreibungen extrahierten Daten werden synchronisiert, wobei der Peer auf diese Aktion reagieren kann um sie beispielsweise in einer grafischen Oberfläche auszugeben. Hierbei ist darauf zu achten, dass die Beschreibung so abstrakt gewählt wird, dass sie auf möglichst viele Fälle, wie die erwähnte Möglichkeit der Implementierung von Chats oder Foren, anwendbar ist.

Als Beweis der Funktionalität der Softwarekomponente wird schließlich eine Chat mit grafischer Oberfläche geschrieben, in dem einzelne Peers sich miteinander unterhalten können.

Die Arbeit wird zuerst die Grundlagen, wie das Shark Framework [4], auf denen die Implementierung beruht, erläutern. Danach wird das Konzept der Softwarekomponente erstellt. Es werden verschiedene Möglichkeiten diskutiert, um die im letzten Absatz beschriebenen Anforderungen umzusetzen. Im Anschluss wird die eigentliche Implementierung vorgestellt und auf diese eingegangen. Nachfolgend werden die Tests und Methoden zur Qualitätssicherung gezeigt und erklärt. Zuletzt wird ein Fazit gezogen sowie ein Ausblick auf mögliche Verbesserungen oder Erweiterungen.

## 2 Grundlagen

Das folgende Kapitel widmet sich den Grundlagen, auf denen die Arbeit aufbaut. Zuerst werden die Funktionale und nicht funktionale Anforderungen festgelegt, da diese die Basis der zu entwickelnden Softwarekomponente darstellen. Anschließend wird auf das Shark Framework eingegangen, auf dem diese Arbeit aufbaut.

### 2.1 Funktionale und nicht funktionale Anforderungen

Im Folgendem werden die funktionalen und nicht funktionale Anforderungen besprochen. Diese beschreiben welche Features die Softwarekomponente bereitstellen soll, sowie wichtige Aspekte der Qualitätssicherung.

#### Funktionale Anforderungen

- **Beschreibbarkeit:** Es ist möglich einen Raum von Daten zu beschreiben und diesen von einem anderen Raum von Daten abzugrenzen.
- **Abhängigkeiten:** Es ist möglich Abhängigkeiten zwischen Räumen zu definieren. So soll beispielsweise der Raum Java-Chat ein Kind des Programmiersprachen-Chat Raumes sein können.
- **Persistenz:** Es soll möglich sein die Beschreibung der Räume von Daten persistent zu speichern. Die gespeicherten Räume bleiben somit erhalten und können zu späterem Zeitpunkt neu geladen werden.
- **Synchronisation:** Es ist möglich die Räume von Daten und ihre Abhängigkeiten mit Peers in einem Peer to Peer Netzwerk zu synchronisieren. Ziel ist es, dass die Räume nach der Synchronisation identisch von Aufbau und Inhalt sind.

#### Nicht funktionale Anforderungen

- **Build-Management:** Die Softwarekomponente ist mittels eines zu bestimmenden Build Tools so eingerichtet, dass das Aufsetzen der Entwicklungsumgebung für andere Entwickler schnell und einfach zu erledigen ist. Mögliche Synergien des gewählten Tools mit anderen Systemen zur Softwareentwicklung und Qualitätssicherung, zum Beispiel Jenkins [2], sind wünschenswert.
- **Testbarkeit:** Die zu Softwarekomponente ist modular so aufgebaut, dass sie durch Modultest testbar ist.
- **Modultest:** Es existieren bereit eine Reihe von Modultest, welche die grundlegende Funktionalität der Softwarekomponente sicherstellen.

### 2.2 Shark Framework

In diesem Unterkapitel wird auf die grundlegenden Features des Shark Framework [4] eingegangen, die für die Arbeit benötigt werden. Das gesamte Framework wird nicht erklärt. Weiterführende Informationen sind im Developer Guide [3] zu finden.

#### 2.2.1 Context Space

Die wichtigste Grundlage ist die Context Space, der hier vereinfacht Kontext genannt werden soll. Hierbei handelt es sich um eine Datenstruktur. Abbildung 1 zeigt ein vereinfachtes Modell dieser Struktur.

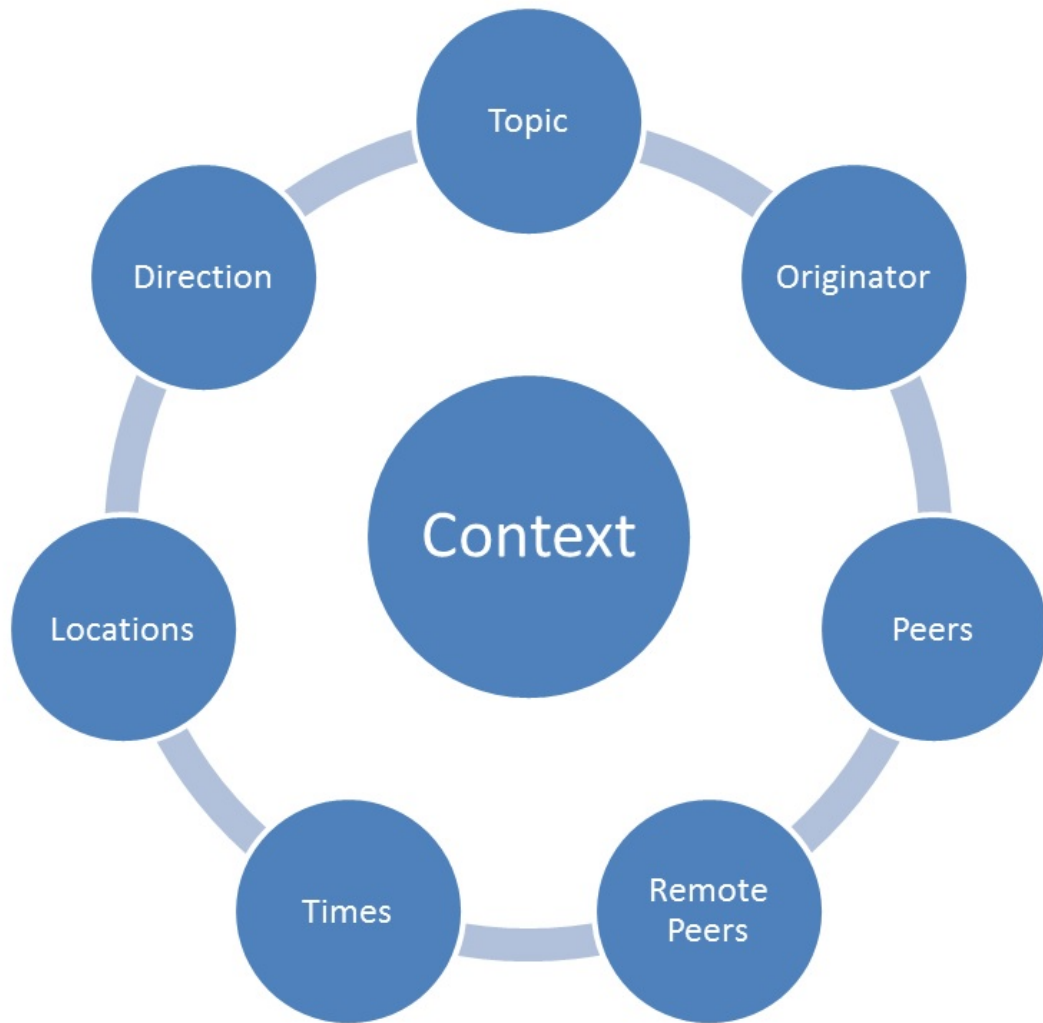


Abbildung 1: Shark Context Space Modell

Die einzelnen Elemente haben dabei folgende Bedeutung:

- **Topic:** Dies ist das Thema eines Kontextes. Es ist eine Beschreibung was der Kontext bedeutet.
- **Originator:** Der Autor der beschriebenen Kontextes. Die kann entweder der Ersteller der sein oder Autor der beiliegenden Informationen. Bei einem wissenschaftlichen Artikel wäre dies der Autor des Titels.
- **Peers:** Peers kann unterschiedlich interpretiert werden. Zum einen kann es der Ersteller des Kontextes sein, der sich vom Autor der beiliegenden Informationen unterscheiden kann. Zum anderen kann es der Besitzer des eines Kontextes sein, wobei Besitzer hier je nach beiliegendem Fall anders interpretiert werden kann. Im Allgemeinen kommt die Interpretation dieser auf den vorliegenden Fall an.
- **Remote Peers:** Dies sind die Peers an welche der Kontext gesendet werden soll.
- **Times:** Eine Zeitinformation die je nach vorliegendem Fall anders Interpretiert werden kann. Diese Dimension bietet die Möglichkeit ein Zeitintervall anzugeben.
- **Locations:** Gibt einen Ort an. Diese Information kann je nach vorliegendem Fall anders interpretiert werden.

- **Direction:** Die Richtung der Kommunikation des Kontextes. Es ist möglich nur Informationen über einen Kontext zu empfangen, sie nur zu senden, zu lesen und senden und keine der genannten Aktionen durchzuführen.

Zu beachten ist, dass wenn von einem Kontext gesprochen wird, dann handelt es sich hierbei um eine Beschreibung einer Menge von Kontextpunkten. Diese Punkte haben den gleichen Aufbau wie in Abbildung 1 gezeigt. Der wesentliche Unterschied ist, dass ein Kontext eine Vielzahl von Inhalten, ausgenommen der Direction, besitzen kann. Währenddessen besitzt der Kontextpunkt nur je genau einen Inhalt. Beispielhaft beschreibt der Kontextpunkt nur das Thema Java, während der Kontext Java und C++ beinhaltet. Die Dimensionen selbst werden als Kontextkoordinaten bezeichnet. Der Kontextpunkt vereint Koordinaten und Informationen. Der Kontext wiederum ist eine Möglichkeit eine Menge an Kontextpunkten aus einer zugrundeliegenden Wissensbasis zu extrahieren.

### 2.2.2 Knowledge Base

Die Knowledge Base, zu deutsch Wissensbasis, ist eine Sammlung von Wissen. Wissen ist dabei eine Menge an Informationen, die anhand von Kontextkoordinaten beschrieben sind. Die Vereinigung von Kontextkoordinaten und Informationen bildet einen Kontextpunkt.

Die Wissensbasis bietet die Möglichkeit Peers und Themen zu speichern, sowie die Themen in einer Taxonomie einzuordnen. Der für die zu entwickelnde Softwarekomponente wichtige Punkt ist aber die Möglichkeit den Kontextpunkt in ihr zu speichern und anhand eines Kontextes zu extrahieren. Die Wissensbasis selbst kann hierbei ähnlich einer Datenbank angesehen werden. Ziel ist es Teilbereiche, sprich eine bestimmte Menge an Daten dieser, mit anderen Wissensbasen zu synchronisieren. Zu diesem Zweck gibt es bereits eine Implementierung, SyncKB genannt, auf der aufgebaut wird.

Zusätzlich zu den genannten Eigenschaften gibt es die Properties an der Wissensbasis zu speichern. Dies sind Name-Wert-Paare, wobei sowohl Name als auch Wert eine Zeichenkette ist.

### 2.2.3 SyncKB

Leider gibt es, abgesehen der Javadoc Dokumentation, keine genaue Beschreibung der Funktionsweise der SyncKB. Daher soll hier auf die generelle Funktionsweise dieser eingegangen werden. Weitere Informationen können den Quellcode selbst entnommen werden, der im Github vom SharkFW zu finden ist. [1]

## 3 Konzeption

### 3.1 Ein erweiterter Kontext

Die Context Spache aus dem Kapitel 2.2.1 ist für unsere Anforderungen nicht ausreichend.

## 4 Quellenverzeichnis

### Webseiten

- [1] *Github SharkFW - SyncKB*. URL: <https://github.com/SharedKnowledge/SharkFW/tree/master/src/java/core/net/sharkfw/knowledgeBase/sync> (besucht am 21.09.2015).
- [2] *Jenkins*. URL: <http://jenkins-ci.org/> (besucht am 15.09.2015).
- [4] Prof. Dr. Thomas Schwotzer. *Shark framework*. URL: <http://www.sharksystem.net/> (besucht am 31.08.2015).

### Handbücher

- [3] Prof. Dr. Thomas Schwotzer. *Building Semantic P2P Applications with Shark*. URL: [http://www.sharksystem.net/sharkDeveloperGuide/Shark2.0\\_DevelopersGuide.pdf](http://www.sharksystem.net/sharkDeveloperGuide/Shark2.0_DevelopersGuide.pdf) (besucht am 18.09.2015).

## 5 Abbildungsverzeichnis

|   |                                      |   |
|---|--------------------------------------|---|
| 1 | Shark Context Space Modell . . . . . | 4 |
|---|--------------------------------------|---|