Bachelorarbeit

Eine Architektur für dezentral organisiertes Social Networking

Benjamin Heisig
B. Sc. Wirtschaftsinformatik

<benjamin@heisig.name>
<www.benjaminheisig.de/smocean>

Version 1.0
vom 2. Februar 2011

Eine Architektur für dezentral organisiertes Social Networking

Projekt-Webseite mit weiteren Informationen unter:
<www.benjaminheisig.de/smocean>
Kontakt zum Autor:
<benjamin@heisig.name>

Dieses Dokument basiert auf einer Bachelorarbeit am Institut für Informatik und Wirtschaftsinformatik (ICB) der Universität Duisburg-Essen. Sie wurde am 2. Februar 2011 der Fakultät für Wirtschaftswissenschaften vorgelegt.

Diese Arbeit steht unter einer Creative-Commons-Lizenz (Namensnennung, Weitergabe unter gleichen Bedingungen, Version 3.0 Deutschland, kurz CC-BY-SA 3.0 DE), abzurufen unter:
<http://creativecommons.org/licenses/by-sa/3.0/de/>.

zuletzt kompiliert am 2. März 2011 mit LaTeX unter Kubuntu GNU/Linux;
enthält 20.082 Wörter auf 90 Seiten

Inhalt

1 Einleitung **1**
1.1 Motivation . . . 1
1.2 Ziele . . . 2
1.3 Ähnliche Arbeiten . . . 3
1.4 Vorgehensweise . . . 3

2 Grundlagen dieser Arbeit **6**
2.1 Semantic Social Networking . . . 6
2.1.1 Social Web . . . 6
2.1.2 Semantic Web . . . 10
2.1.3 Social Semantic Web . . . 13
2.2 Decentralized, Distributed . . . 15
2.2.1 Decentralized Computing . . . 15
2.2.2 Distributed Computing . . . 18
2.2.3 Web Services . . . 19
2.2.4 Internet Agents . . . 22
2.3 Free/Libre and Open Source Software . . . 23
2.3.1 Open Source, Free Software und artverwandte Begriffe . . . 23
2.3.2 Abgrenzung von anderen Paradigmen . . . 24
2.3.3 Rechtliche Rahmenbedingungen . . . 25
2.3.4 Projekte, Entwicklung und Communities . . . 26

3 Herausforderungen im Kontext von Social Networking **31**
3.1 Überblick . . . 31
3.1.1 Statistiken . . . 31
3.1.2 Zentral organisiertes Social Networking . . . 32
3.2 Auswirkungen von Social Networking . . . 34
3.3 Anforderungen an Online Social Networks und deren Architekturen . . . 35
3.3.1 Technische Anforderungen . . . 35
3.3.2 Soziale Anforderungen . . . 36
3.3.3 Rechtliche Anforderungen . . . 37
3.3.4 Ökonomische Anforderungen . . . 37
3.3.5 Ökologische Anforderungen . . . 38
3.4 Nachteile durch zentral organisiertes Social Networking . . . 39
3.5 Dezentral organisiertes Social Networking . . . 40

4 Entwicklung eines dezentral organisierten Online Social Networks 42
4.1 Idee und Konzeption . . . 42
4.1.1 Konzepte . . . 42
4.1.1.1 Dezentralisierung . . . 42
4.1.1.2 Offenheit . . . 43
4.1.1.3 Transparenz . . . 44
4.1.1.4 Sicherheit . . . 45
4.1.1.5 User-Zentrierung . . . 47
4.1.2 Ziel- und Interessengruppen . . . 47
4.1.3 Features . . . 48
4.2 Architektur . . . 49
4.2.1 Paradigmen . . . 49
4.2.1.1 Ressourcen-orientierte Architektur mit REST . . . 50
4.2.1.2 Modularisierung, Integration und Wiederverwendung . . . 51
4.2.1.3 Selbstorganisation . . . 58
4.2.1.4 Access Controll und Rights Management . . . 60
4.2.2 Kernkomponenten . . . 62
4.2.2.1 Portal . . . 62
4.2.2.2 Community . . . 64
4.2.2.3 Service . . . 65
4.2.2.4 Client . . . 66
4.3 Organisation . . . 68
4.3.1 Private und (nicht-)kommerzielle Provider . . . 69
4.3.2 Foundation und Entwickler-Community . . . 70

5 Evaluation 72
5.1 Chancen und Risiken . . . 72
5.2 Limitation . . . 73

6 Schlussbetrachtung 75

Quellen 76

Abbildungen 84

Tabellen 85

Quelltexte 86

Akronyme 87

1 Einleitung

Das Social Web bietet einige interessante Ansatzpunkte für innovative Ideen und Weiterentwicklungen des Webs. In dieser Arbeit wird eine dieser möglichen Innovationen diskutiert: dezentral organisiertes Social Networking. Die dazugehörige und in dieser Arbeit zu entwickelnde Architektur vereint Konzepte und Technologien aus den Bereichen Semantic Web, Decentralized und Distributed Computing sowie Free/Libre and Open Source Software (FLOSS) zu einem Online Social Network (OSN).

1.1 Motivation

An dieser Stelle wird auf das Thema der Arbeit eingegangen. Dessen Hintergründe dienen als Motivation zu dieser Arbeit und damit zur Entwicklung einer möglichen Lösung der im folgenden beschriebenen Problemfelder.

Social Networking im Internet ist im Laufe der letzten Jahre zu einem integralen Bestandteil unserer Gesellschaft geworden. Privat wie auch beruflich interagieren Menschen im Web, vernetzen sich, generieren neue Inhalte und greifen auf bestehende Daten zu, die aus verschiedenen Quellen stammen und neu miteinander arrangiert werden, um etwas Neuwertiges zu erschaffen. Mit dem globalen Ausbau der Internet-Infrastruktur wird das Social Web, auch Web 2.0 genannt, immer wichtiger für die weltweiten User.

Während dieser Trend weiter anhält, wächst der Unmut vieler User, die den Providern im Social Web – beispielsweise denen eines OSNs – ihre Daten anvertrauen: Problematische Auswirkungen der Marktmacht großer, zentral organisierter OSNs verschärfen weltweit zunehmend die Diskussion um die Privatsphäre der User sowie um die Verwendung und Verknüpfung ihrer Daten. Viele Geschäftsmodelle beruhen auf der Verarbeitung von Daten der User; die wiederum verlieren den Überblick darüber, welche Daten von ihnen in welchem Umfang verarbeitet, weitergegeben und gespeichert werden. Hier stellt sich die Frage nach dem Grund für die kommerzielle Handhabung privater Daten und ob es nicht Alternativen gibt, die solche Geschäftsmodelle obsolet machen können.

Die eben geschilderte, problematische Entwicklung spitzt sich durch die Always-On-Mentalität der User weiter zu: Die steigende Nutzung von OSNs führt zu einem Echtzeit-Internet, das durch mobile Applikationen und den Ausbau von Netzkapazitäten ständig

verfügbar wird. Es werden zunehmend netzpolitische Debatten im Kontext von Social Networking thematisiert. Ein politisches, ökonomisches wie auch gesellschaftliches Thema ist die Netzneutralität: Techniken wie Quality of Service (QoS) zur Priorisierung von Netzwerkdaten, Netzwerkanalysen durch Deep Packet Inspection, zusätzliche Geschäftsmodelle auf Basis einer Bepreisung von Netzwerkdaten und vieles mehr gefährden die Neutralität des Internets und somit auch die Nutzung von OSNs, für die eine neutrale, breitbandige und möglichst latenzfreie Nutzung von Netzwerkkapazitäten existenziell ist.

Einen weiteren Punkt stellen die technischen Herausforderungen an große OSNs dar. Hohe User-Zahlen verursachen einen enormen Ressourcenverbrauch, dem, aus technischer Sicht betrachtet, von der Hardware- und Software-Seite her begegnet werden muss. Eine nachhaltige Entwicklung im Bereich der Informationstechnologie (IT), auch Green IT genannt, stellt einen für das Social Networking gewichtigen, ökonomischen Faktor dar, welcher von zentral organisierten Architekturen technisch bedingt nur erschwert begegnet werden kann.

Aufgrund dieser kritischen Punkte ändern sich die Anforderungen an OSNs und deren Architekturen stetig. Es gilt, aus den identifizierten Problemen Anforderungen an das Social Networking allgemein, an die OSNs und die dahinter stehenden Architekturen sowie an die Handlungen der beteiligten Stakeholder (beispielsweise Provider, User und Regierungen) zu stellen. Aus diesen Anforderungen lassen sich Kriterien entwickeln, mit denen sich OSNs evaluieren lassen. Diese Evaluierung wird für dezentral organisiertes Social Networking angestrebt, insbesondere für die Architektur des in dieser Arbeit entwickelten Konzepts.

Das konstituierende Merkmal dieser Arbeit liegt darin, ein OSN mit dessen dahinter liegenden Architektur in der Theorie zu entwickeln. Diese Architektur integriert verschiedene, bereits bestehende Komponenten wie Technologien und Standards aus den Bereichen Semantic Web, Decentralized und Distributed Computing sowie FLOSS. Vordergründig betrachtet stellt dieses OSN für Internet-Nutzer ein freies, offenes Werkzeug zur Informationsgenerierung, zum -konsum sowie zur Vernetzung dar; im Hintergrund geht es darum, den technischen, sozialen, ökonomischen, rechtlichen und auch ökologischen Herausforderungen möglichst umfassend zu begegnen, denen sich heutige OSNs zukünftig stellen müssen.

1.2 Ziele

Die zentrale Fragestellung für diese Arbeit wird wie folgt formuliert: Wie kann ein nachhaltiges, den Anforderungen der Stakeholder genügendes Konzept zum Social Networking gestaltet werden? Aufgabe ist es, ein dazu passendes OSN mit der dahinter liegenden Architektur zu entwickeln.

Das Hauptanliegen dieser Arbeit besteht darin, Erkenntnisse über den Gegenstand dezentral organisiertes Social Networking zu gewinnen, um daraus die theoretische Entwicklung der OSN-Architektur abzuleiten. Darüber hinaus werden die Ideen und Konzepte des OSNs vermittelt, auf die sich die Architektur stützt.

Ein darauf aufbauendes Anwendungsziel ist, das Software-Entwickler, Gründungsinteressierte und/oder Internet Service Provider (ISP) den diskutierten Ansatz in die Praxis umsetzen. Weiterhin bieten sich innerhalb der Forschungsschwerpunkte Informatik, Wirtschafts- und Sozialwissenschaften weitere Untersuchungsmöglichkeiten an, welche an denjenigen Stellen ansetzen, an denen diese Arbeit limitiert ist (vergleiche später Kapitel 5.2).

1.3 Ähnliche Arbeiten

E existiert eine große Anzahl an wissenschaftlichen Arbeiten rund um die Themen Social Networking im Internet, Semantic Web, Decentralized/Distributed Computing oder FLOSS. Ebenso existieren Arbeiten, die zwei bis drei dieser Themen in Kombination diskutieren. Ansätze, alle relevanten Themen ganzheitlich und wissenschaftlich aufbereitet abzudecken, entziehen sich den Kenntnissen des Autors.

Daher werden die erwähnten Arbeiten innerhalb der folgenden Kapitel diskutiert, so wie sie inhaltlich passend erscheinen. Arbeiten, die keinen wissenschaftlichen Anspruch erheben, werden ebenfalls behandelt, um mehr interessante Ideen und Konzepte einzubringen oder zumindest in Betracht zu ziehen.

1.4 Vorgehensweise

Wie diese Arbeit aufgebaut ist, ist Inhalt dieses Kapitels. In diesem Rahmen werden die Inhalte der einzelnen Kapitel, die aus verschiedenen Perspektiven beleuchtet werden, kurz angerissen. Zuletzt wird das Thema der Arbeit weiter eingegrenzt.

Auf Basis des in Kapitel 1.1 definierten Themas befasst sich die Arbeit mit Maßnahmen zur Erreichung der in Kapitel 1.2 definierten Ziele. Die Vorgehensweise sieht folgende Punkte vor: In Kapitel 2 werden die Grundlagen für diese Arbeit gelegt. Anschließend werden die Kenntnisse über den Gegenstand dezentral organisiertes Social Networking in den Kontext der gegenwärtigen Praxis gestellt (Kapitel 3), um Anforderungen für die eigens zu entwickelnde Architektur zu bestimmen. Diese Architektur mit zugehöriger Konzeption sind als zentraler Bestandteil dieser Arbeit in Kapitel 4 beschrieben. Aufgrund dieses Vorgehens

ist die Arbeit der gestaltungsorientierten Wirtschaftsinformatik zugeordnet[1]. Eine aus den Anforderungen abgeleiteten und auf den eigenen Ansatz angewandten Evaluierung wird anschließend in Kapitel 5 behandelt und limierende Faktoren dieser Arbeit diskutiert, bevor in Kapitel 6 ein Fazit mit Ausblick folgt.

Mit den in Kapitel 1.1 identifizierten Problemfeldern beginnend zieht sich ein roter Faden durch die gesamte Arbeit. Um eine Weitsicht auf das Subjekt dieser Arbeit (sprich: eine Architektur für dezentral organisiertes Social Networking) zu erlangen und somit den Kontext (sprich: Social Networking im Internet, Semantic Web, Decentralized/Distributed Computing und FLOSS) möglichst umfassend zu erfassen, werden die Themen aus fünf Perspektiven beleuchtet:

- technische Perspektive,
- soziale Perspektive,
- ökonomische Perspektive,
- rechtliche Perspektive,
- ökologische Perspektive.

Die Wahl der Perspektiven beruht auf der Annahme, mit ihnen die vorgestellten Themen möglichst umfassend zu diskutieren. Dies gilt auch für die Architektur: Sie als Selbstzweck zu entwickeln, ergäbe wenig Sinn. Daher dienen die Perspektiven für eine Diskussionsgrundlage, in die sich die Konzepte hinter der Architektur bettet. Die Perspektiven können für sich betrachtet stehen. Darüber hinaus bedingen sie sich gegenseitig, denn eine wohldefinierte Abgrenzung ist, wie sich beispielsweise in Kapitel 3.3 zeigen wird, nicht immer möglich und gegebenenfalls auch ungewollt, um einseitige Perspektiven zu vermeiden.

Der Begriff der Entwicklung wird weit gefasst: Scheinbar indirekt für die technische Entwicklung einer Architektur, aber für diese Arbeit existenziell, gilt es, nicht nur die Architektur zu entwickeln und zu diskutieren, sondern auch die Bereiche, die darüber hinaus gehen, zumindest thematisch anzuschneiden. Kurz gefasst: Das *Wie* kann nicht ohne das *Warum* stehen. Auf diese Weise ergibt sich ein Gesamtbild.

Es gibt einige Fragen und Punkte, die nicht in dieser Arbeit behandelt werden können (vergleiche später Kapitel 5.2), weil sie zu weit gefasst sind. Sie beziehen sich auf die Zukunft des Internets und dienen als Denkanregungen bei der Entwicklung der Architektur:

- Wie werden sich das Internet, das Web und insbesondere Social Networking weiterentwickeln?

[1] Vergleiche dazu das „Memorandum zur gestaltungsorientierten Wirtschaftsinformatik“ unter `http://www.wip.wiwi.uni-due.de/fileadmin/fileupload/WI-WIP/Dokumente/Memorandum-GWI-2010-03-08.pdf`, abgerufen am 22. Oktober 2010.

  - Welche Technologien werden eine Rolle spielen?
  - Welche Provider und Geschäftsmodelle werden sich etablieren?
  - Welchen (rechtlichen, nationalen/internationalen) Rahmenbedingungen wird sich das Web stellen müssen?
- Was wird *The Next Big Thing* im Internet sein und welche Auswirkungen wird es haben?

2 Grundlagen dieser Arbeit

Dezentral organisiertes Social Networking ist in eine komplexe Thematik eingebettet, die zudem im späteren Verlauf aus verschiedenen Perspektiven beleuchtet wird. Als Grundlage für alle weiteren Betrachtungen dienen deshalb die in den folgenden Unterkapiteln erarbeiteten Begriffsdefinitionen und -abgrenzungen, die nach dem Terminus „Free/Libre and Open Source Decentralized, Distributed and Semantic Social Networking“ gegliedert sind (aus didaktischen Gründen in umgekehrter Folge). Der Terminus umreißt kompakt, welche zentralen Aspekte bei der Entwicklung der Architektur zu berücksichtigen sind[1]. Zum besseren Verständnis werden Grundkenntnisse zu den Gegenständen Standards in der IT (Melzer 2008, Seite 24) und Services vorausgesetzt.

2.1 Semantic Social Networking

Das Social Web, gepaart mit dem Semantic Web – davon handelt dieses Kapitel. Auf den ersten Blick scheinen beide Begriffe nicht zueinander zu passen, was im Folgenden revidiert wird. Zunächst folgt eine Einführung in die wichtigsten Begriffe zu diesem Themenkomplex, in der sich die zu entwicklende Architektur einbettet.

2.1.1 Social Web

Das Social Web ist Gegenstand dieses Kapitels. Neben einer kurzen Definition mit Erläuterung kommen weitere wichtige Begriffe aus diesem Themenfeld zur Sprache, die für diese Arbeit eingeführt und voneinander abgegrenzt werden, um den Themenkomplex Social Networking zu umfassen.

Das World Wide Web (WWW), hier verkürzt Web genannt, ist ein wesentliches Element des Internets, wie wir es heute kennen. Ursprünglich von Tim Berners-Lee 1989 erstmals der Öffentlichkeit vorgestellt (Friedman 2009, Seite 33–38), wird es vom World Wide Web

[1] Für die verwendeten Begriffe werden die englischen Fachausdrücke bevorzugt, weil der Großteil an herangezogener Literatur englischsprachig ist, eine nicht immer exakte Übersetzung ins Deutsche existiert oder schlicht verschiedensprachige Ausdrücke vermieden werden sollen.

Consortium (W3C) weiterentwickelt und standardisiert. Mit dem Hypertext Transfer Protocol (HTTP) als Basis, das mittlerweile in Version 1.1 vorliegt (Fielding et al. 1999), entstanden unter anderem die weit verbreitete Hypertext Markup Langage (HTML) und das in Kombination mit eXtensible Markup Language (XML) wohlgeformtere eXtensible HyperText Markup Language (XHTML), welches in Version 1.0 seit mehreren Jahren Bestand hat (Pemberton et al. 2000). Zur Zeit wird innerhalb des W3Cs HTML5 standardisiert (Hickson 2010), das im Zuge des Social Webs neu hinzu gekommenen Anforderungen gerecht werden soll.

Bild 2.1 Logo der World Wide Web Consortiums

Das Web ist dazu gedacht, Ressourcen in einem Netzwerk (vorzugsweise im Internet) zur Verfügung zu stellen. Dazu wird jede Ressource über eine Uniform Resource Locator (URL)[2] identifiziert und in dem spezifischen Netzwerk lokalisiert (Berners-Lee et al. 1994). Eine URL ist vom Aufbau her von einer Uniform Resource Identifier (URI) abgeleitet, die eine Ressource eindeutig referenziert (Berners-Lee et al. 2005). Die auf dem Unicode-Zeichensatz aufbauende Internationalized Resource Identifier (IRI) gilt als internationales Äquivalent zur URI (Dürst und Suignard 2005). Den schematischen Aufbau einer HTTP URL beschreibt Code 2.1.

```
http://example.net:80/path/file?foo=bar&bar=baz#foobar
|      |           |  |         |               |
|      |           |  |         |               fragment/
|      |           |  |         |               anchor
|      |           |  |         |               identifier
|      |           |  |         searchpart
|      |           |  url-path
|      |           port
|      host
protocol
```

Code 2.1 Schematischer Aufbau einer HTTP URL nach Berners-Lee et al. (1994)

Das Web hat sich seit der Veröffentlichung von Tim Berners-Lee sehr stark weiterentwickelt: Wurden die Inhalte anfangs nur von wenigen Menschen erstellt und von vielen konsumiert (One-to-Many-Verbindung), was als Web 1.0 bezeichnet wird, wandelte sich das Web zu einem Mitmach-Web, dem Social Web, auch Web 2.0 genannt (O'Reilly 2005). Der „Consumer" wurde zum „Prosumer", weil er in die Lage versetzt wurde, Inhalte selbst zu erstellen (Schaffert 2006), welche von anderen Usern konsumiert oder bearbeitet werden (Many-to-Many-Verbindung). Diese Bearbeitung können beispielsweise Annotationen in Form von Tags sein, nach denen sich Inhalte und Ressourcen verschlagworten lassen. Tags sind ein gängiges Mittel, um Ressourcen und Inhalte im Web zu suchen, zu sortieren,

[2] Im Deutschen müsste es „der URL" (maskulin) heißen, doch da URL im sprachlichen Gebrauch mit der „Internetadresse" (feminin) gleichgesetzt wird, wird für diese Arbeit der feminine Genus gewählt. Diese Behandlung gilt im Folgenden auch für weitere, ähnliche Akronyme.

in Relation mit anderen Ressourcen und Inhalten zu setzen und so weiter. Tags lassen sich clustern oder in einer Tag Cloud visualisieren (Bild 2.2).

Bild 2.2 Eine Tag Cloud über das Web 2.0 nach Cremonini (2006).

Das Social Web besteht aus Web-Applikationen und -Services, die daher auch Social Software genannt werden. Diese Software umfasst „diejenigen onlinebasierten Anwendungen [...], die das Informations-, Identitäts und Beziehungsmanagement in den (Teil-)Öffentlichkeiten hypertextueller [...] und sozialer Netzwerke unterstützen.“ (Schmidt 2006, Seite 37). Die drei genannten Handlungskomponenten grenzen verschiedene Arten von Social Software ein, wozu unter anderem Wikis, Weblogs (auch Blogs genannt) und Microblogs, Instant Messenger sowie Software für ein Social Networking Service (SNS) zählen und die in Bild 2.3 visualisiert werden (OSN heißt dort SNS). Schmidt (2006, Seite 38) ergänzt: „Diese Beispiele machen deutlich, dass der Begriff 'social software' in gewisser Weise irreführend ist: Nicht die Software an sich ist sozial, sondern diese Qualität entsteht erst im gemeinsamen, sinnhaft auf andere bezogenen Gebrauch einer spezifischen Anwendung.“[3]

[3]Rechtschreibung und Grammatik der Zitate werden so, wie sie sind, übernommen.

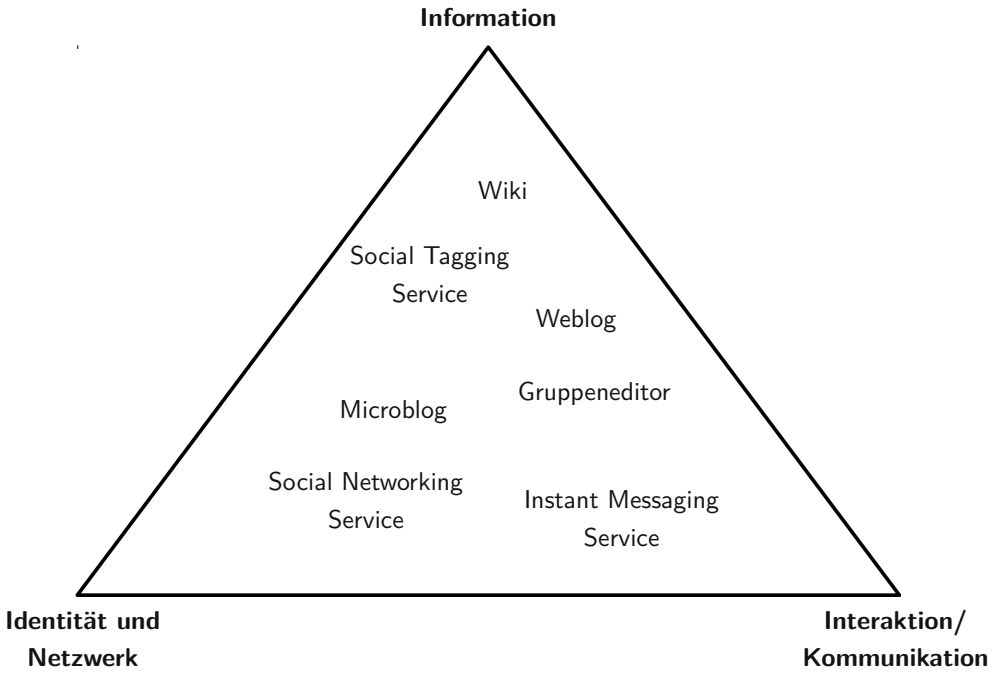


Bild 2.3 Das Social-Software-Dreieck nach Koch und Richter (2009, Seite 11–14), basierend auf Schmidt (2006).

SNS enthält den Begriff Networking, unter dem nach Koch und Richter (2009, Seite 53) „der Aufbau und das Pflegen von Beziehungen“ verstanden wird. Weiter definieren Koch und Richter (2009, Seite 54) Dienste als SNS, die ihren Usern Funktionen zum Identitätsmanagement zur Verfügung stellen; darüber hinaus ermöglichen diese Funktionen die Vernetzung mit anderen Usern. Insgesamt werden nach Koch und Richter (2009, Seite 54–57) sechs Grundfunktionen unterschieden, die ein SNS bietet:

- Das **Identitätsmanagement** umfasst Profile und Gruppenzugehörigkeiten.
- Die **Expertensuche** meint das (erweiterte) Suchen und Empfehlen.
- **Kontextawareness** beschreibt das Vertrauen zwischen Usern, welches zum Beispiel durch Anzeigen gemeinsamer Bekannte geschaffen werden soll.
- **Netzwerkawareness** setzt das Abrufen von Status-Nachrichten in den Vordergrund.
- Zum **Kontaktmanagement** gehören die Pflege des persönlichen Netzwerks und die Verwaltung von Kontaktdaten.
- Ein **gemeinsamer Austausch** gestaltet sich durch das Senden und Empfangen von Nachrichten.

Für diese Arbeit werden SNS und Begriffe wie die Onlinecommunity als OSN bezeichnet. Ein OSN ist somit ein Service im Social Web, der die Grundfunkionalität zum Social Networking bietet.

Die Versionierung des Webs in 1.0 und 2.0 impliziert die Frage, was Web 3.0 sein könnte. Hierbei ist sich die Literatur uneins beziehungsweise kritisiert die Wahl der Versionierung, weshalb an dieser Stelle von den Versionierungen Abstand genommen wird.

Für das bessere Verständnis werden einige der bereits verwendeten Begriffe an dieser Stelle konkretisiert: Als Provider wird in dieser Arbeit ein Betreiber eines Systems oder Services bezeichnet. Dieser Provider kann eine natürliche oder juristische Person darstellen, wird allerdings eher eine juristische Person beziehungsweise eine Organisation darstellen. Organisationen können kommerzielle oder nicht-kommerzielle Interessen verfolgen. Daher fallen unter den Begriff der Organisation Unternehmen, Vereine, öffentliche Verwaltungen et cetera. Beispiel für einen Provider ist ein ISP, der Services im Internet betreibt und Dritten zur Nutzung anbietet. Diese Dritten werden definitorisch für diese Arbeit als User, das heißt (Be-)Nutzer/Anwender eines Systems oder Services eingeordnet. Ein System oder Service kann an Endnutzer gerichtet sein. Jene Endnutzer werden in dieser Arbeit jedoch ebenfalls zu den Usern gezählt. Eingebettet werden Provider und User in die Gruppe der Stakeholder, zu denen alle natürlichen und juristischen Personen zählen, die innerhalb einer Domäne – zum Beispiel Social Web – (inter-)agieren. Werden Personen und Maschinen (vergleiche später Kapitel 2.2.4) betrachtet, wird in dieser Arbeit von Akteuren gesprochen.

2.1.2 Semantic Web

An dieser Stelle folgt eine Einführung in das Semantic Web, eine weitreichende „Erweiterung“ des Webs, das hierbei mit Semantiken angereichert wird. Kurz wird auf die dahinter stehende Ideologie, die Vorteile, die sich aus der Anwendung ergeben, und die verwendeten Standards und Technologien eingegangen.

Die Inhalte und Strukturen im Web sind darauf ausgelegt, von Menschen gelesen und verstanden zu werden (Berners-Lee et al. 2001). Die Bedeutung eines Satzes auf einer Webseite oder die Beschreibung eines Bildes oder eines Videos im Web können sich lediglich dem menschlichen User erschließen. Selbst Menschen mit Einschränkungen (beispielsweise in Form einer Sehbehinderung) können teilweise nur mit großem technischen Aufwand am Web partizipieren. Was fehlt, sind Inhalte, die nicht nur von einem durchschnittlich intelligenten Menschen, sondern auch von einer Maschine erfasst, richtig interpretiert und weiterverarbeitet werden können. Damit eine Maschine bei einer Suche im Web die User zielgerichtet zum gewünschten Ergebnis führen kann, muss sie die Bedeutung, sprich die Semantik der Inhalte, die sie für diese Suche durchforstet, verstehen können. Im Jahr

2001 wurde von Berners-Lee et al.[4] das Semantic Web erstmals einer breiten Öffentlichkeit vorgestellt. Das Semantic Web gilt ihnen zufolge nicht als ein neuer, vom bisherigen Web separierter Ansatz, sondern als Erweiterung des bestehenden Webs. Daten und daraus generierte Informationen sollen eine wohldefinierte Bedeutung erhalten und einer Logik folgen. Letztlich ist das Semantic Web gemäß den Autoren dazu bestimmt, das Wissen der Menschheit im Ganzen zu evolutionieren, was Goddard und Byrne (2010, Seite 3)[5] folgendermaßen auf den Punkt bringen: „One of the most compelling aspects of the Semantic Web vision is the idea that computers will be able to create new knowledge from existing information."

Grundsätzlich stehen hinter dem Semantic Web zwei simple Ideen (Goddard und Byrne 2010, Seite 2): Daten werden zum einen mit Semantik angereichert und zum anderen zueinander in Relation gesetzt. Durch die Anreicherung entsteht ein Web of Data, ein häufig verwendetes Synonym für das Semantic Web (Berners-Lee 2006). Ein ebenso häufig auftauchender Begriff ist Linked Data (Berners-Lee 2006), bei dem die Relationen von strukturierten Daten im Vordergrund stehen. Hierbei gelten die Regeln von Berners-Lee (2006) für eine sinnvolle Anwendung und Verbreitung des Semantic Webs:

1. „Use URIs as names for things"
2. „Use HTTP URIs so that people can look up thoses names. " (Vergleiche zu beiden Punkten die Grundlagen aus Kapitel 2.1.1.)
3. „When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)" (dazu später mehr)
4. „Include links to other URIs. so that they can discover more things."

Aus diesen Regeln wird deutlich, dass Daten (im Zitat „things" genannt) im Web eine eindeutige Referenzierung in Form von URIs benötigen, damit ein Datum eine eindeutige Bedeutung erhält und von verschiedenen Usern und Maschinen auf dieselbe Art interpretiert werden kann.

Die Umsetzung der Ideologie des Semantic Webs steht seither vor großen Herausforderungen: Wie das Web soll auch das Semantic Web dezentral strukturiert sein, denn genau das macht die Universalität des Webs aus (Berners-Lee et al. 2001). Die Dezentralisierung erfordert jedoch Kompromisse: ein in sich konsistentes Modell, das alle Inhalte im Web organisieren kann, wird es nicht geben können, prognostizierten die Autoren damals. Sie forderten eine Sprache für Regeln, die so ausdrucksstark wie nötig ist, um weitgehende Schlussfolgerungen den Maschinen zu ermöglichen.

[4] Statt der Originalquelle lag lediglich eine Online-Version vor (vergleiche Literaturverzeichnis), die zwar den Orginaltext, aber ein anderes Layout verwendet. Daher wird auf eine Seitenangabe verzichtet, um Konflikte mit den Seitenangaben des Orignals zu vermeiden.

[5] Die für diese Quelle genannten Seiten beruhen auf eigener Seitenzählung.

Um eine geeignete Sprache mit den passenden Regeln zu entwickeln und auf eine technisch solide Basis zu stellen, wurden innerhalb des W3Cs zahlreiche Standards veröffentlicht (W3C 2010), welche ansatzweise im Semantic Web Stack (Bild 2.4) in Zusammenhang gebracht werden. Nach diesem Stack – auch Layer Cake genannt (unter anderem von Hendler (2001)) – fußt das Semantic Web auf URIs, IRIs und Unicode auf der untersten Ebene, um Daten im Web zu identifizieren. Eine Ebene darüber werden XML-basierte Standards zum Strukturieren von Daten definiert (Berners-Lee et al. 2001), wozu auch Standards zum Signieren und Verschlüsseln von Daten zählen. Auf diesem Fundament fußen weitere Standards, die vor allem im Semantic Web eingesetzt werden: Dazu zählt das Resource Description Framework (RDF) (Manola und Miller 2004), das aus Tripeln (Subjekt, Prädikat, Objekt) besteht und Daten somit eine Bedeutung zuweisen kann. RDF kennt verschiedene Schreibweisen: Neben der XML-basierenden Schreibweise (Manola und Miller 2004) bietet sich das vereinfachte Notation3 (N3) an (Berners-Lee 1998). Um Daten direkt in XHTML/HTML semantisch zu annotieren, kann RDF in Attributes (RDFa) verwendet werden (Adida et al. 2008). Das nötige Vokabular zuzüglich einer Art Grammatik wird in Ontologien organisiert. Hendler (2001, Seite 30) definiert den Begriff Ontologie folgendermaßen: „[. . .] I employ the term as it is currently being used in Semantic Web circles. I define ontology as a set of knowledge terms, including the vocabulary, the semantic interconnections, and some simple rules of inference and logic for some particular topic.“ Zu einer ähnlichen Definition gelangen Berners-Lee et al. (2001): „The most typical kind of ontology for the Web has a taxonomy and a set of inference rules.“ Eine Taxonomie besteht den Autoren zufolge aus Klassen von Objekten und deren Relationen zueinander. Bekannte Ontologien sind RDF Schema (RDFS) und Web Ontology Language (OWL) in den Ausprägungen OWL Lite, OWL Description Logic (OWL-DL) und OWL Full (McBride (2004) und Hitzler et al. (2009)). Für die Abfrage wurden ebenfalls Sprachen entwickelt, von denen vor allem die SPARQL Protocol and RDF Query Language (SPARQL) hervorzuheben ist (Prud'hommeaux und Seaborne 2008).

Wenn es um die semantische Auszeichnung von Daten geht, muss nicht zwingend eine Ontologie mit dazugehörigen Abfragesprachen eingesetzt werden: Im Web hat sich der Einsatz einer Folksonomy als Taxonomie bewährt, besser bekannt unter dem Begriff Tagging (Lassila und Hendler 2007, Seite 91), was auch Gegenstand von Kapitel 2.1.1 ist. Als simple, semantische Erweiterungen von HTML gelten Microformats (Lassila und Hendler 2007, Seite 91) und der Einsatz von Meta-Tags im `<head>`-Bereich des HTML-Codes, beispielsweise Dublin Core (Dublin Core Metadata Initiative 2010).

Auch wenn eine Versionierung des Webs bereits in Kapitel 2.1.1 abgelehnt wird, sei hier kurz angemerkt, dass das Semantic Web oftmals als Web 3.0 bezeichnet wird (vergleiche zum Beispiel den Titel „Embracing 'Web 3.0'“ eines Artikels von Lassila und Hendler (2007)). Nach Blumauer und Pellegrini (2009, Seite 12) ist das Social Web (dort Web 2.0 genannt) das „Frontend“ für die Konnektivität und das Semantic Web das „Backend“ für die Kontextualisierung.

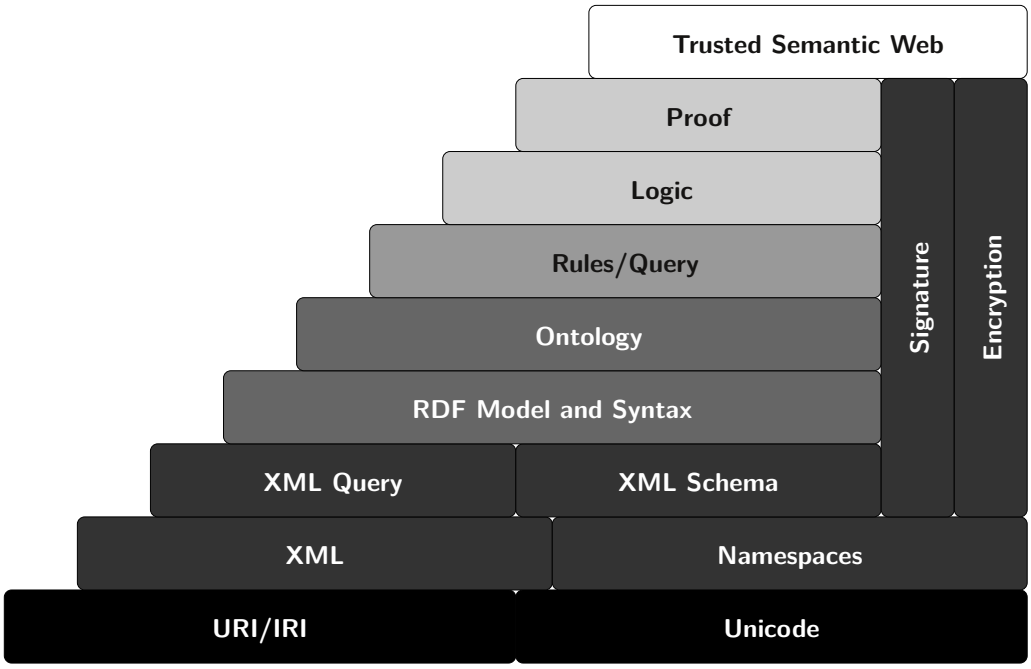


Bild 2.4 Der Semantic Web Stack nach Signore (2005), basierend auf Berners-Lee (2000).

2.1.3 Social Semantic Web

Die Kombination aus den vorher diskutierten Themen Social Web (Kapitel 2.1.1) und Semantic Web (Kapitel 2.1.2) ist Schwerpunkt dieses Kapitels. Es wird gezeigt, inwiefern sich beide ergänzen und welche Chancen sich dadurch auf beiden Seiten ergeben.

Während hinter dem Social Web eine Applikationsebene, bestehend aus Social Software, liegt (Kapitel 2.1.1), kommt beim Social Semantic Web auf Applikationsebene folglich die Semantic Social Software zum Einsatz. Sie ist die Kombination von Social Software und Technologien des Semantic Webs (Schaffert 2006, Seite 2). Dabei werden zwei Richtungen nach Schaffert (2006, Seite 4) unterschieden: Die Semantically Enabled Social Software wendet Technologien des Semantic Webs in Social Software an, zum Beispiel in Form von Metadaten (Schaffert 2006, Seite 4–5). Das Socially Enabled Semantic Web hingegen unterstützt das Semantic Web durch Social-Web-Applikationen, die zum Beispiel die Erstellung von Metadaten vereinfachen und verbesseren (Schaffert 2006, Seite 5). Pellegrini und Blumauer (2009, Seite 273) sehen im Social Semantic Web die Community-basierte Wissensorganisation, kombiniert mit der maschinellen Inhaltserschließung (Bild 2.5).

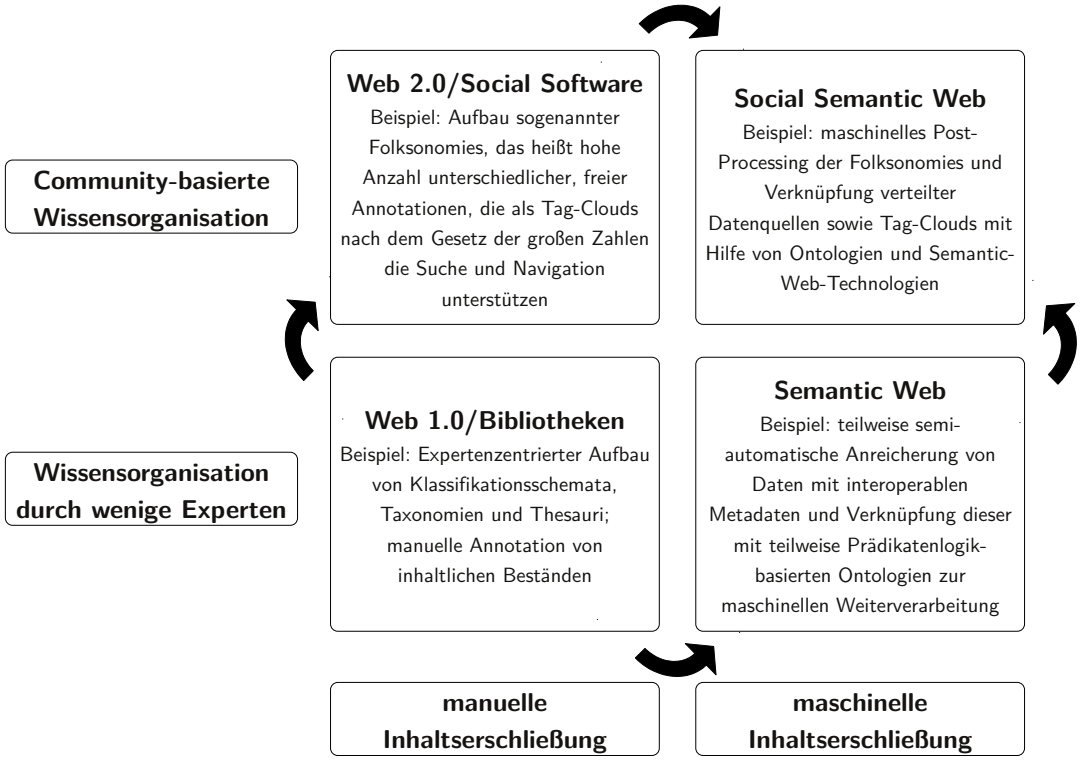


Bild 2.5 Die Entwicklungspfade zum Social Semantic Web entlang der Dimensionen Wissensorganisation und Inhaltserschließung nach Pellegrini und Blumauer (2009, Seite 273).

Nach Blumauer und Pellegrini (2009, Seite 5) verfolgt das Semantic Web einen Top-Down-Ansatz: Die Vertreter des Semantic Webs suchen die Akzeptanz im Web in der Standardisierung; die Skeptiker bevorzugen eher den Bottom-Up-Ansatz des Social Web, das neue Technologien ohne lange Standardisierungsprozesse adaptiert. Wie Schaffert gezeigt hat (siehe oben), treffen sich beide Ansätze an einem Punkt. Nach Blumauer und Pellegrini (2009, Seite 12) ist dieser Punkt erreicht, wenn Social Web als „Frontend“ zur Konnektivität und Semantic Web als „Backend“ zur Kontextualisierung verbunden werden können.

Je enger Informationen miteinander verknüpft werden und intensiver soziale Aktivitäten im Web geschehen, desto näher gelangt das Web zu einer Art Metaweb (Spivack 2004). Bild 2.6 verdeutlicht diese Betrachtung. Es zeigt, dass dezentral organisierte OSNs (dort als Decentralized Communities bezeichnet) als eine Art der Weiterentwicklung des Webs gelten.

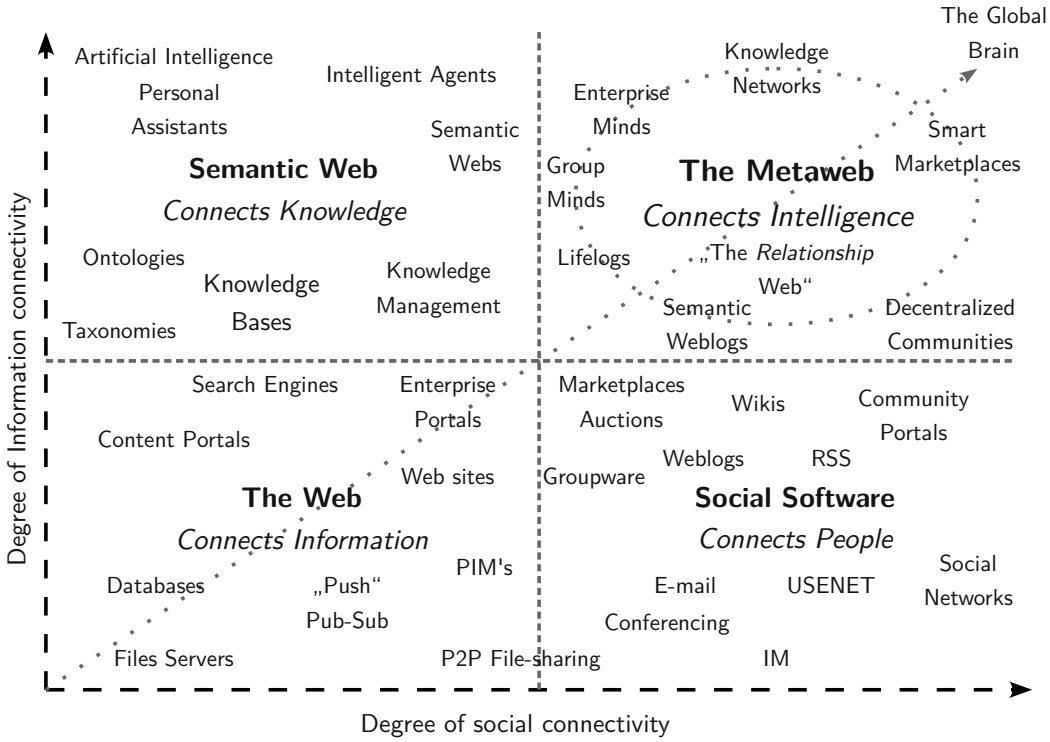


Bild 2.6 Das Metaweb nach Spivack (2004).

2.2 Decentralized, Distributed

Decentralized und Distributed Computing sind zwei weitreichende Begriffe aus der Informatik, die vor allem im Internet relevant sind. Viele Basistechnologien aus dem Protokollstapel des Internets, aber auch das Internet und das Web selbst sind dezentralisierte und verteilte Systeme wie beispielsweise das Domain Name System (DNS) oder E-Mail. In den folgenden Unterkapiteln werden die zentralen Begriffe eingeführt, welche essentiell für die zu entwickelnde Architektur sind, um eine dezentrale Infrastruktur aufzubauen.

2.2.1 Decentralized Computing

Dezentrales Rechnen, hier Decentralized Computing genannt, ist Gegenstand dieses Kapitels. Hier kommt vor allem der Themenkomplex Peer-to-Peer (P2P) zum Tragen. Die Anforderungen an P2P, damit die dahinter liegende Architektur nützlich erscheint, sind in Ansätzen auch in den Bereichen Internet-, Web-Technologien und im (nicht auf das Web bezogenen) Social Networking wiederzuerkennen.

Laut Steinmetz und Wehrle (2005, Seite 12) läutet Decentralized Computing ein sich änderndes Paradigma in der Internet-Kommunikation ein:

- von der Koordination zur Kooperation von Entitäten;
- von zentralisierten Strukturen hin zu dezentralen;
- von der Abkehr von Kontrollinstanzen zum Leistungsanreiz.

Unter anderem bei Steinmetz und Wehrle (2005, Seite 10) findet sich eine Definition von P2P, der federführenden Technologie hinter Decentralized Computing: Demnach ist ein P2P-System ein sich selbst organisierendes System aus gleichwertigen, autonomen Entitäten (Peers), die sich in einem Netzwerk zusammenschließen, um verteilte Ressourcen miteinander zu teilen. P2P dient dazu, verschiedene Typen von Ressourcen zu verwalten (Schoder et al. 2005): Die Ressourcen können Informationen, Dateien, Bandbreite, Datenspeicher oder auch Prozessorzyklen sein. Diese Ressourcen werden möglichst gleichmäßig auf alle Peers verteilt. Dabei sind die Peers innerhalb eines Netzwerks, vorzugsweise im globalen Internet, miteinander verbunden und können auf die Ressourcen anderer Peers zugreifen (Steinmetz und Wehrle 2005, Seite 10-11). Die Autonomie der Peers ermöglicht eine direkte Interaktion, was eine zentrale Organisation überflüssig macht (Bild 2.7). Das heißt, Peers sind in diesem Netzwerk gleichzeitig Dienstanbieter (Server) und -nutzer (Client) (Steinmetz und Wehrle 2005, Seite 11-12). Das bedeutet einen wesentlichen Vorteil gegenüber dem Client-Server-Ansatz, denn dieser Ansatz wird von den steigenden Anforderungen des Internets nicht länger gerecht, argumentieren Steinmetz und Wehrle (2005, Seite 9–10). Client-Server-Modelle sind tendenziell teuer und schwerfällig in der Modifizierung. P2P-Systeme helfen, Flaschenhälse und Attacken, sogenanntes Distributed Denial of Service (DDOS), zu vermeiden. Ergo: P2P verspricht einfachere Lösungen in einer immer komplexeren (Internet-)Welt (Steinmetz und Wehrle 2005, Seite 9–10).

Bei der Verwaltung von Ressourcen innerhalb eines P2P-Systems werden wesentliche Unterschiede deutlich: Die erste Generation von P2P-Systemen verwaltet Ressourcen wenig effizient. Meist werden neben den Peers weitere Server benötigt, um Ressourcen aufzufinden, was zu einer Hybrid-Lösung führt (Bild 2.7). Dieses läuft allerdings dem P2P-Gedanken von völlig autonomen Peers zuwider und kann zu ähnlichen Problemen wie beim Client-Server-Ansatz führen (vor allem Skalierungsprobleme). Diese Systeme gelten daher als unstrukturiert (Steinmetz und Wehrle 2005, Seite 15). Die zweite Generation legt folglich den Fokus darauf, nicht nur den Austausch von Ressourcen, sondern auch die Verwaltung dessen effizient auf alle Peers zu verteilen. Ressourcen werden in einer verteilten Datenstruktur strukturiert und adressiert (Steinmetz und Wehrle 2005, Seite 15–16). Diese Struktur wird allgemein als Distributed Hash Table (DHT) bezeichnet (Wehrle et al. 2005).

Lehmann und Kaufmann (2005, Seite 58) fassen den Gedanken von P2P zusammen, indem sie vier Voraussetzungen aufstellen, damit P2P-Systeme nützlich sind:

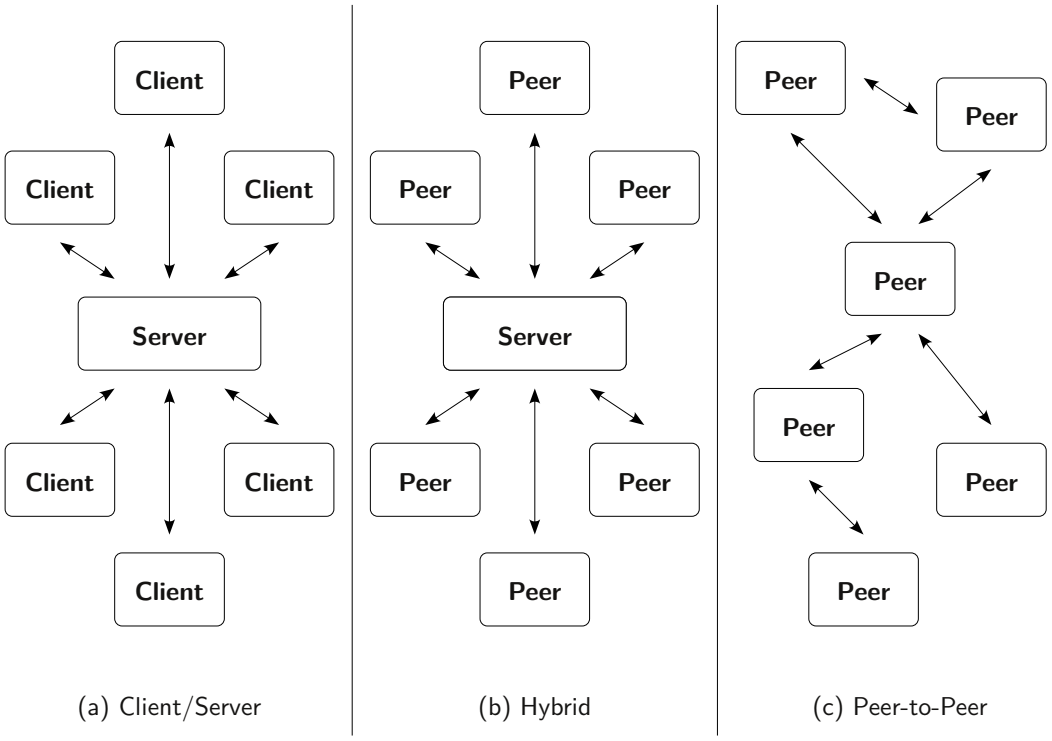


Bild 2.7 Klassifikation von P2P-Systemen nach Steinmetz und Wehrle (2005, Seite 11).

Dezentralisierung: Es gibt kein zentrales Element zur Organisation. Stattdessen existieren Regeln zum Hinzufügen weiterer Elemente (sprich Peers), Regeln für das Routen von Informationen untereinander und Regeln zum Abfragen von Informationen.

Struktur: Strukturen sind nützlich, um Information finden und routen zu können.

Verlässlichkeit trotz dynamischer Änderungen: Wichtige Systemeigenschaften müssen „stabilisiert“ werden.

Skalierung: Die Bedürfnisse der Peers müssen auf effiziente und schnelle Art und Weise befriedigt werden.

Diese Voraussetzungen können bei genauerer Betrachtung auch auf die Eigenschaften von (realen wie virtuellen) Social Networks, des Internet und des Web übertragen werden. Daher lohnt es sich, die Technologien aus dem Bereich des Decentralized Computing für diese Arbeit in Betracht zu ziehen. Konkrete Beispiele für P2P-Protokolle, aus denen sich ein System erstellen lässt, bieten unter anderem die Frameworks Chaupal, ehemals JXTA (Verstrynge 2010), GNUnet (GNUnet 2011) und Freenet (The Freenet Project 2011). Die beiden letzteren sind insbesondere auf Anonymität und Privatsphäre der User ausgerichtet (vergleiche später Kapitel 3, in dem diese Punkte eine größere Rolle spielen). Populär ist auch das Protokoll von BitTorrent, welches ursprünglich als hybrides P2P-System entwi-

ckelt wurde, den Peers aber unter anderem durch die Einführung von DHTs Autonomie verleiht (BitTorrent.org 2011). Durch seine weite Verbreitung bietet sich BitTorrent für die Verteilung von Software an, was sich einige Linux-Distributoren zu Eigen machen (vergleiche später Kapitel 2.3.4).

2.2.2 Distributed Computing

Distributed Computing, sprich das verteilte Rechnen gehört wie das Decentralized Computing zu einem wichtigen Ansatz im Internet wie auch im Web. Darauf basierende Architekturen – wie die in dieser Arbeit zu entwickelnde – profitieren in mehrfacher Hinsicht von dieser Technologie, was im Folgenden näher erläutert wird.

Eine Abgrenzung zwischen Decentralized und Distributed Computing ist in der Literatur nicht klar zu erkennen. Dennoch ergibt eine für diese Arbeit gewählte Trennung Sinn: Aufgrund der engen Verwandtschaft ergänzen sich beide Architekturparadigmen zu einem umfassenderen Konstrukt.

Aus dem wissenschaftlichen Bereich stammt der Begriff Grid Computing, wozu in der Literatur keine allgemeingültige Definition existiert. Mauthe und Heckmann (2005, Seite 193–194) zufolge, bietet Grid Computing Zugang zu einem gemeinschaftlichen, sehr großen Pool verschiedener Ressourcen, um mit ihnen innovative Applikationen umzusetzen. Hierbei geht es um „die koordinierte, transparente und gesicherte Nutzung von IT-Ressourcen, welche über geografische und institutionelle Grenzen hinweg gemeinschaftlich genutzt werden“ (Melzer 2008, Seite 339). Ein Grid bietet „seine Ressourcen bedarfsabhängig in einer standardisierten Art und Weise“ (Melzer 2008, Seite 339) an. Melzer (2008, Seite 339–340) beschreibt weiterhin verschiedene Arten von Grids: Computing-Grids („Verbund von Rechenleistung“), Data-Grids („gemeinsame Nutzung von Speicherkapazitäten“) und Resource-Grids („gemeinsame Nutzung von [...] wissenschaftlichen Geräten“). Ziele des Grid Computings sind „Zuverlässigkeit, hohe Verfügbarkeit, geringe Zusatzkosten und eine koordinierte Verteilung der Ressourcen an die jeweiligen Konsumenten“ (Melzer 2008, Seite 339). Dadurch ist eine „Kopplung von existierenden heterogenen Systemen in unterschiedlichen administrativen Domänen“ möglich, „ohne dass eine zentrale Kopplung oder Steuerung vorhanden ist“ (Melzer 2008, Seite 340). Trotzdem bedarf es einiger Koordnierungsmechanismen, um die Sicherheit, Skalierbarkeit und die Dienstgüte zu gewährleisten, ergänzt Melzer (2008, Seite 340). Auch ist Grid Computing nur durch eine Middleware als Infrastruktur möglich, die die Funktionen „Nutzer-Authentifizierung, Ressourcenauffindung, Ressourcenreservierung und -nutzung sowie das Management von Daten“ übernimmt (Melzer 2008, Seite 340).

Grid Computing hat einige Gemeinsamkeiten zu P2P (Kapitel 2.2.1), sodass es keine klare Abgrenzung geben kann (Mauthe und Heckmann 2005, Seite 202–205). Trozdem zeigen sich unterschiedliche Merkmale: Grids bieten eine Plattform zur Integration verschiedener

Applikationen, während P2P-Applikationen vertikal integriert sind (Mauthe und Heckmann 2005, Seite 193–194). Oder anders ausgedrückt (Mauthe und Heckmann 2005, Seite 203): Eine Grid-Infrastruktur bietet Zugang zu einem Pool von Ressourcen (vergleiche die Arten von Grids), P2P-Applikationen hingegen „nur“ zu einfachen Ressourcen (vergleiche die beschriebenen Ressourcen aus Kapitel 2.2.1).

Eine Form des Grid Computing, welches im Web Verbreitung findet, ist das Content Distribution Network (CDN). Provider von CDNs bauen eine weltweite Infrastruktur auf, um Inhalte effizienter, sprich mit geringerer Latenz und höherer Bandbreite für Akteure bereitzustellen. Ein Beispiel für ein nicht-kommerzielles, frei zugängliches CDN ist Coral, das ein weltweites Netzwerk zum Caching von Webseiten aufbaut. Die Basis bilden P2P-Technologien mit DHT, HTTP-Proxy-Server und DNS-Nameserver. Eine Einführung in CDNs im allgemeinen und Coral im speziellen bietet Freedman (2010).

Distributed Computing wird seit einiger Zeit mit Cloud Computing verbunden, einem sehr abstrakten Begriff, der viele Bedeutungen hat. Daher versuchen Lenk et al. (2009, Seite 25) den Begriff zu kategorisieren (Bild 2.8). Zu den Kategorien gehören Human as a Service (HuaaS), Software as a Service (SaaS), Platform as a Service (PaaS) sowie Infrastructure as a Service (IaaS). Diese Services werden in der Literatur auch mit dem Begriff Everything as a Service (XaaS), auch kurz EaaS, zusammengefasst. Die jeweiligen Services setzen auf einer Hardware auf und werden – ähnlich dem Grid Computing – zentral koordiniert.

Durch die Nutzung all dieser Technologien wird es möglich, Ressourcen und darauf aufbauende Services zu dezentralisieren. Es muss sich nicht mehr ein einzelner Akteur um Ressourcen und Services, die er nutzen will, selbst kümmern, sondern kann sich die Arbeit mit anderen Akteuren teilen oder auf fremde Ressourcen und Services zugreifen. Dazu existieren Standards und Kontrollmechanismen, die die Handhabung erleichtern.

2.2.3 Web Services

Web Services sind zu zentralen Technologien im Web geworden. An dieser Stelle wird der Begriff, dessen Wurzeln sich in Service-orientierten Architekturen wiederfinden, um die Nützlichkeit dessen für ein OSN zu demonstrieren.

Eine Service Oriented Architecture (SOA) dient der Kommunikation zwischen Maschinen, die vollständig automatisiert abläuft (Melzer 2008, Seite 9–27). Da keine einheitliche Definition existiert, wird für diese Arbeit auf die Definition von Melzer (2008, Seite 13) verwiesen: „Unter einer SOA versteht man eine Systemarchitektur, die vielfältige, verschiedene und eventuell inkompatible Methoden oder Applikationen als wiederverwendbare und offen zugreifbare Dienste repräsentiert und dadurch eine plattform- und sprachenunabhängige Nutzung und Wiederverwendung ermöglicht.“ Veranschaulicht wird dies in Bild 2.9: Als notwendige Voraussetzung für eine SOA – hier als Fundament dargestellt – gelten

Einfachheit, Sicherheit und Akzeptanz (durch Standards) (Melzer 2008, Seite 12). Darauf aufbauend stehen die Säulen Verteiltheit, lose Kopplung, Verzeichnisdienst und Prozessorientierung, welche eine SOA stützen. Einen groben Ablauf, wie auf eine SOA zugegriffen wird, wird in Bild 2.10[6] beschrieben: Ein Provider veröffentlicht einen Service im Service Directory. Ein User benötigt diesen Service und sucht ihn im Directory. Der wiederum verweist auf den Provider, damit der User dort auf den Service zugreifen kann. Dieser Vorgang ist für eine maschinelle Kommunikation existenziell.

Web Services lassen sich als eine SOA, die auf das Web beschränkt ist, interpretieren (de Bruijn et al. 2008, Seite 14). Das Web kann nicht nur für die Nutzung von Menschen aufbereitet werden (was Gegenstand in Kapitel 2.1.1 ist), sondern wird durch Web Services auch für Maschinen nutzbar (Richardson und Ruby 2007, Seite 1–21). Daher wird es von Richardson und Ruby auch als „machine-readable Web“ oder auch „programmable Web“ bezeichnet, das das „human-readable Web“ ergänzt und mit ihm verschmilzt.

Es existieren zahlreiche Web-Service-Standards, zu denen unter anderem das Protokoll Remote Procedure Call (RPC) mit XML als Austauschformat, kurz XML-RPC, gehört (Winer 1999). Das W3C hat verschiedene Spezifikationen, um Web Services zu gestalten, zu Standards erklärt (W3C 2010). Dazu gehört das Protokoll Simple Object Access Protocol (SOAP), welches als Weiterentwicklung von XML-RPC gilt. SOAP wird durch Web Services Description Language (WSDL) ergänzt, das als Beschreibungssprache von Web Services verwendet wird (Chinnici et al. 2007). Ein passender Service Directory wird auf Basis von Universal Description Discovery and Integration (UDDI) realisiert (Clement et al. 2004).

Der typische Ablauf zum Abruf eines Web Services ähnelt dem einer SOA (Bild 2.11), wobei es drei Arten der Beschreibung von Services gibt (de Bruijn et al. 2008, Seite 20–21):

Funktionale Beschreibung: Welche Funktionen werden angeboten?

Verhaltens-orientierte und Schnittstellenbeschreibung: Welche Interaktion findet statt? Welche Daten/Informationen werden ausgetauscht?

Nicht-funktionale Beschreibung: weitere Parameter wie Trust (Vertrauen) und Security (Sicherheit; vergleiche beide Parameter später in Kapitel 4.1.1.4), QoS und Preise (ein Thema der Netzneutralität; Kapitel 1.1 und 3.3)

Der Vorteil der Protokolle XML-RPC und SOAP liegt in der wohlstrukturierten Serialisierung von Methoden, die ein Server nach einer Deserialisierung direkt verarbeiten kann. Nachteilig erscheint in der Praxis der hohe Protokoll-Overhead: Die Protokolle weisen eine hohe Komplexität auf und sind daher nicht leicht von Programmierern umzusetzen, was vor allem XML geschuldet ist.

[6] Im Gegensatz zu den deutschen Bezeichnungen im Bild, wird die englische Terminologie präferiert.

Die Nachteile führen zu den Vorteilen von Representational State Transfer (REST), mit dem sich ebenfalls Web Services definieren lassen (Fielding 2000). REST ist kein Protokoll, sondern „ein Architekturstil, der definiert, wie existierende Web-Protokolle verwendet werden können, um beispielsweise möglichst einfach Web Services zu realisieren.“ (Melzer 2008, Seite 103). Die Einfachheit besteht unter anderem darin, sich auf Applikationen im Web zu beschränken und somit lediglich HTTP einzusetzen (Melzer 2008, Seite 103). Melzer (2008, Seite 103) ergänzt: „Ist [ein Web Service] konform zum REST-Architekturstil, wird [er] als RESTful bezeichnet.“ Nach Richardson und Ruby (2007, Seite 13) sind Web Services RESTful, wenn sie aussehen wie das Web. Das heißt, sie verwenden lediglich die gängigen Technologien, die Webseiten auch zur Verfügung stehen, wie HTTP und URLs. Die Methoden-Information wird in der HTTP-Methode umgesetzt (Richardson und Ruby 2007, Seite 13). Web Services mit REST sind Ressourcen-orientiert, weil sie „eine beliebige Menge von Ressourcen“ verwalten (Melzer 2008, Seite 103). „Eine Ressource kann nach Definition vom W3C prinzipiell alles sein, ihre Repräsentation muss jedoch mittels einer URL zugreifbar sein.“ (Melzer 2008, Seite 103) Daher ergänzen Richardson und Ruby (2007, Seite 13), dass eine Ressourcen-orientierte Architektur die Abfrage in der URL stellt. Code 2.2 gibt ein Beispiel für eine einfache Anfrage an einen Web Service, die a) RESTful und b) Ressourcen-orientiert gestaltet ist (Richardson und Ruby 2007, Seite 13). Eine passende Beschreibungssprache von RESTful Web Services stellt die Web Application Description Language (WADL) dar (Hadley 2009).

```
  GET /reports/open-bugs HTTP/1.1
2 |   |                  |
  |   |                  HTTP-Version
4 |   Ressourcen-orientiert
  RESTful
```

Code 2.2 Das Beispiel zeigt eine einfache Anfrage an einen Web Service, der RESTful und Ressourcen-orientiert ist.

Auch wenn REST ein Architekturstil und kein Protokoll ist, stellt es eine „interessante Alternative zu SOAP und XML-RPC“ (Melzer 2008, Seite 105) dar: Anfragen an Web Services sind „sehr einfach als HTTP-Operationen“ zu gestalten und es werden „weder Programmierkenntnisse noch spezielle Entwicklungswerkzeuge benötigt“ (Melzer 2008, Seite 105). Weitere Vorteile ergeben sich durch die sinnvolle Ausnutzung von HTTP: Performance, Skalierbarkeit, Verfügbarkeit, Erweiterbarkeit und eine mögliche Punkt-zu-Punkt-Verschlüsselung via SSL/TLS (Melzer 2008, Seite 105). Nachteile sieht Melzer (2008, Seite 105) darin, dass Methodenaufrufe nicht gekapselt werden können und somit ein erhöhter Server-seitiger Programmieraufwand nicht zu unterschätzen ist.

Web Services bieten sich als „machine-readable“ Schnittstelle für Webseiten aus dem Social Web (Kapitel 2.1.1) an. Diese Schnittstelle wird oftmals als Application Programming Interface (API) bezeichnet. Dadurch können einzelne Services miteinander verwoben werden – ein neuer Service, ein sogenanntes Mashup entsteht (Auer et al. 2009, Seite 259–260). Die Konzepte des Semantic Webs (Kapitel 2.1.2) ergänzen sich mit denen von Web Services: „Web services might be one of the most powerful uses of Web ontologies [...]“ (Hendler 2001, Seite 32)

An anderer Stelle können Web Services und P2P-Systeme (Kapitel 2.2.1) voneinander „lernen“: Auf der Seite der Web Services existieren zahlreiche Standards zur Kommunikation, Beschreibung, Registrierung, Verbindung, Interoperabilität und Sicherheit (Hillenbrand und Müller 2005, Seite 220–221). Auf der anderen Seite der P2P-Systeme stehen zum Beispiel die Dezentralisierung (Hillenbrand und Müller 2005, Seite 221–222). Miteinander kombiniert, ergeben sich nach Hillenbrand und Müller (2005, Seite 222–223) nützliche Seiteneffekte, was Bandbreite, Sicherheit und Wartung betrifft.

2.2.4 Internet Agents

Web Services auf der einen, Internet Agents auf der anderen Seite: In die Interaktion von Agenten-Systemen mit Web Services führt dieses Kapitel ein, wobei es die Konzepte aus dem Bereich des Semantic Social Networking (Kapitel 2.1) fortsetzt.

Auf Web Services greifen hauptsächlich nicht Menschen zu, sondern Maschinen (Kapitel 2.2.3). Die Maschinen können einfache Clients sein, die von Menschen bedient werden. Das kann zum Beispiel ein Webbrowser sein. Diese Maschinen agieren jedoch nicht automatisch ohne Zutun eines Menschen. Anders ist es bei Maschinen – hier auf Software-Basis betrachtet –, die als Agents bezeichnet werden. Sie sind in der Lage autonome Handlungen zu vollziehen: Software Agents im Web sind nach Berners-Lee et al. (2001) Computer-Programme, die Web-Inhalte aus verschiedenen Quellen sammeln, die erhaltenen Informationen verarbeiten und die Resultate mit anderen Programmen austauschen. Im Semantic Web (Kapitel 2.1.2) spielen Agents eine besondere Rolle, denn durch die Anreicherung von Web-Inhalten mit Semantik wird es selbst für Agents möglich, miteinander zu interagieren und Daten auszutauschen, obwohl sie ursprünglich nicht dafür entwickelt wurden (Berners-Lee et al. 2001). Dieselben Autoren gehen sogar so weit, zu behaupten, das Semantic Web werde das Fundament und den Rahmen für eine künstliche Intelligenz schaffen (Berners-Lee et al. 2001).

Web Services bieten sich geradezu an, Anfragen von Agents entgegen zu nehmen. Neben dieser Kommunikationsform beschreibt Hendler (2001, Seite 34–35) zwei weitere: Zum einen kann ein Agent mit einem User interagieren, indem zum Beispiel der Agent Informationen aus dem Web für einen User verarbeitet und diesen gegebenenfalls um eine Entscheidung bittet. Zum anderen können Agents auch untereinander agieren. Hierbei verständigen sie sich mittels einer gemeinsamen Ontologie.

Für ein OSN, das die Konzepte von Web Services und dem Semantic Web umsetzt, stehen somit eine Vielzahl von Möglichkeiten zur Verfügung: Agents können Daten, die sie von verschiedenen Quellen zusammentragen, für die User aufbereiten, damit diese schneller und bequemer an diese Daten gelangen und nur diejenigen Daten erhalten, die sie interessieren.

2.3 Free/Libre and Open Source Software

Open Source (OS) und Free Software, die zusammengefasst auch als FLOSS bezeichnet werden, sind zwei Paradigmen aus der Software-Entwicklung. Im Bereich von Internet- und Web-Technologien, vor allem aber im Bereich des Social Networkings nehmen sie eine bedeutene Rolle ein, weil sie neben offenen Standards zur Verbreitung einzelner Technologien beitragen. Dies lässt sich potenziell für die in dieser Arbeit zu entwickelnde Architektur nutzen, um Aspekte der Organisation, die die Entwicklung der Software betreut, abzudecken.

2.3.1 Open Source, Free Software und artverwandte Begriffe

Für ein grundlegendes Verständnis wird zunächst geklärt, wie sich OS und Free Software definieren und worin sie sich unterscheiden. Zudem werden weitere, artverwandte Begriffe im Kontext erläutert.

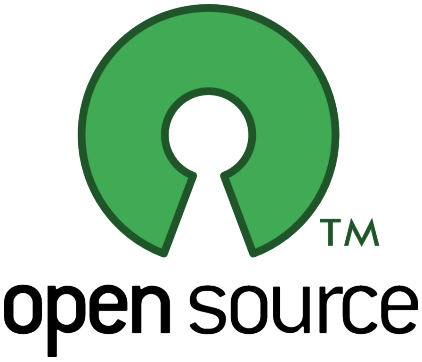


Bild 2.12 Logo der Open Source Initiative

OS, wörtlich übersetzt „offene Quelle“, bezeichnet Quelltexte, zum Beispiel von Software, die für jeden öffentlich zugänglich sind. Demnach ist diese Software in der Regel frei verfügbar, sprich kostenlos im Internet herunterzuladen. Die Open Source Initiative (OSI) hat sich zum Ziel gesetzt, den Begriff „Open Source“ einheitlich zu definieren, und hat dazu 10 Kriterien aufgestellt (Open Source Initiative 2010a), die in der Open-Source-Community allgemein anerkannt werden: So muss nicht nur der freie Zugang zum Quelltext sichergestellt sein, sondern auch die Nutzungsrechte daran weitläufige Freiheiten bieten. Das bedeutet beispielsweise, dass der Quelltext von Dritten bearbeitet, weiterverbreitet oder auch mit eigenen und anderen Quelltexten kombiniert werden darf.

Weitergehend als OS ist der Begriff Free Software, zu Deutsch „Freie Software“ (Free Software Foundation 2010a), geprägt von der Free Software Foundation, kurz FSF (Free Software Foundation 2007), und dem dazugehörigen Projekt namens GNU's Not Unix (GNU) (Free Software Foundation 2010b). Deren Definition zufolge stehen die der Software und deren Quelltexten zugesprochenen Freiheiten im Vordergrund, die immer gewahrt sein müssen: Eine der Freiheiten besagt, dass die Rechte von Free Software und ihrer Derivate nicht eingeschränkt werden dürfen. Diese Verhinderung der freiheitlichen Einschränkung

wird von der Free Software Foundation auch als „Copyleft“ (in Anlehnung an „Copyright“) bezeichnet.

Auch wenn OS und Free Software nicht dasselbe sind, werden sie oftmals synonym verwendet. Für diese Arbeit wird daher der zusammenfassende Begriff FLOSS verwendet. Abstrahiert wird dabei, dass in FLOSS das Wort „Software“ enthalten ist. Das würde eine Einschränkung auf Software bedeuten, wird aber hier vernachlässigt. Die in dieser Arbeit verwendete Definition berücksichtigt neben Software beispielsweise auch Programmbibliotheken (Libraries), eine Zusammenstellung verschiedener Software (Software Stack) oder – für diese Arbeit vor allem relevant – Web Services (Kapitel 2.2.3).

Bild 2.13 Logo des GNU-Projekts

2.3.2 Abgrenzung von anderen Paradigmen

Neben FLOSS existieren viele weitere, oft ähnliche Ausdrücke, von denen im weiteren Verlauf abgegrenzt wird. Weit verbreitet sind Closed-Source- und artverwandte Modelle, welche an dieser Stelle in Kürze behandelt werden, um die Abgrenzung von FLOSS zu komplettieren.

Bild 2.14 Logo der Creative Commons

FLOSS selbst bettet sich nach den Definitionen von Open Content (Creative Commons 2007) und Free Cultural Works (Definition of Free Cultural Works 2008) in eine Liste weiterer offener und/oder freier „Dinge“ ein. Neben den Bewegungen rund um OS und Free Software existieren weitere Bewegungen abseits der Software-Entwicklung, die sich der freien Verfügbarmachung von Inhalten verschreiben. Erwähnenswert sind beispielsweise Open Access für wissenschaftliche Inhalte (Hartmann und Jansen 2008) und Creative Commons für künstlerische Inhalte (Creative Commons 2010b).

Werden Quelltexte nicht offen gelegt, gehören sie zu Closed Source (CS). Synonym dazu gilt der Begriff „proprietär“. Auf CS basierende Software wird demnach als Closed Source Software (CSS) bezeichnet.

Kommerziell vermarktet werden FLOSS und CSS gleichermaßen. Nicht verwechselt werden darf FLOSS mit Freeware, womit die Software als kostenlos verfügbar bezeichnet wird, doch deren Quelltexte nicht veröffentlicht wird.

2.3.3 Rechtliche Rahmenbedingungen

FLOSS berührt viele rechtliche Rahmenbedingungen, welche im Folgenden kurz umrissen werden. Es wird deutlich, warum dieses Thema nicht nur im Hinblick auf die hier zu entwickelnde Architektur von Bedeutung ist, sondern auch für den gesamten Bereich der Internet- und Web-Technologien.

Grundsätzlich gilt: FLOSS bedeutet nicht, dass sie frei von Rechten ist. Ein Autor erlangt automatisch das Recht als Urheber des Werkes, das nach deutscher Rechtsprechung erst 70 Jahre nach dessen Tod erlischt. Diese Werke sind ab dann gemeinfrei. In anderen Ländern können Autoren ihr Werk der Öffentlichkeit übergeben, Public Domain genannt (Creative Commons 2007). Ebensowenig ist FLOSS frei von Gewährleistungs-, Haftungs-, Marken- und auch möglichen Patentrechten.

Die Nutzungsbedingungen von FLOSS werden in Lizenzen festgehalten. Die OSI verzeichnet mittlerweile über 70 Lizenzen, die sie als OS-Lizenzen anerkennt (Open Source Initiative 2010b). Sie unterscheiden sich meist im Detail: Eine liberale Lizenz wie die BSD License erlaubt die Nutzung auch in CS, ohne diese selbst unter eine OS-Lizenz stellen zu müssen (Open Source Initiative 2008). Wesentlich restriktiver ist die GNU General Public License (GPL). Quelltexte, die Teile aus GPL-lizensierten Quelltexten beinhalten, müssen zwingend ebenfalls unter der GPL oder einer zu ihr kompatiblen Lizenz lizensiert werden. So soll die „Freiheit“ von Free Software gewahrt werden, während Software-Entwickler bei liberalen Lizenzen frei entscheiden können, unter welche Lizenz sie ihre Quelltexte, die OS-lizensierte Quelltexte Dritter enthalten, stellen.

Lizenzen können untereinander kompatibel sein – beispielsweise die GPL und die Apache License (The Apache Software Foundation 2004) – oder unabhängig voneinander auf denselben Quelltext angewandt werden. Hierbei wird von einer Mehrfachlizensierung gesprochen. Der Autor kann den Nutzern die Wahl lassen, ob der Quellcode zum Beispiel unter der GPL, der BSD License oder einer Lizenz aus dem CS-Bereich stehen soll.

Die von der OSI anerkannten Lizenzen beschränken sich nicht auf Software: Für Web Services (Kapitel 2.2.3) und Libraries stehen dedizierte Lizenzen bereit, um Konflikte mit anderen Lizenzen zu minimieren.

2.3.4 Projekte, Entwicklung und Communities

Die Entwicklung von FLOSS in Projekten und Communities wird in diesem Kapitel erläutert. Es ist zu zeigen, inwiefern diese organisatorischen Ausprägungen der Software-Entwicklung nützlich für die zu entwickelnde Architektur sind.

Eine Einführung in die Software-Entwicklung von FLOSS bietet Raymond (1997). Dort wird beschrieben, dass FLOSS üblicherweise in Projekten entwickelt wird, um die sich eine Community bildet. Während viele Communities aus freiwilligen Entwicklern bestehen, die in ihrer Freizeit unentgeltlich an einem Projekt arbeiten, gibt es auch FLOSS-Projekte, die von Unternehmen geführt werden, welche einen Teil ihrer Mitarbeiter für die Entwicklung bereit stellen. Mischformen aus Privatleuten, (nicht-)kommerziellen Organisationen und öffentlichen Verwaltungen sind ebenso möglich. Um große Projekte dieser Art bilden sich oftmals Organisationsstrukturen mit einer Non-Profit-Organisation (NPO) an der Spitze. Beispiele für solche Strukturen und deren Projekte liefern Mozilla mit Firefox (mozilla.org 2010), die Apache Software Foundation (ASF) mit ihrem Webserver (The Apache Software Foundation 2010a), die Document Foundation (OpenOffice.org Deutschland e. V. 2010) mit LibreOffice (ein Fork, das heißt Abspaltung von OpenOffice.org) und das GNU Operating System mit betriebssystemnaher Standard-Software und vielen Libraries auf unixoiden Betriebssystemen. Ein weiteres großes Projekt, hinter dem keine antreibende kommerzielle Kraft steckt, sondern eine Community aus Individuen, ist Debian, eine im privaten wie kommerziellen Bereich etablierte freie GNU/Linux-Distribution (Debian 2010). Das besondere an der Community ist ihre demokratische Organisation mit eigener Verfassung und einem Gesellschaftsvertrag, dem sich alle Community-Mitglieder verpflichtet sehen.

Zusammengefasst existiert FLOSS in allen Software-Kategorien und auf so gut wie allen Hard- und Software-Plattformen. Darüber hinaus finden die dahinter stehenden Ideologien in vielen anderen Bereichen (Kultur, Wissenschaft und vieles mehr) Anhänger, die sich zu Communities zusammenschließen, um ihre Interessen und Ziele zu verfolgen. Diese Communities bilden eine Gemeinschaft, die mit oder ohne kommerzielle Absichten Projekte vorantreiben.

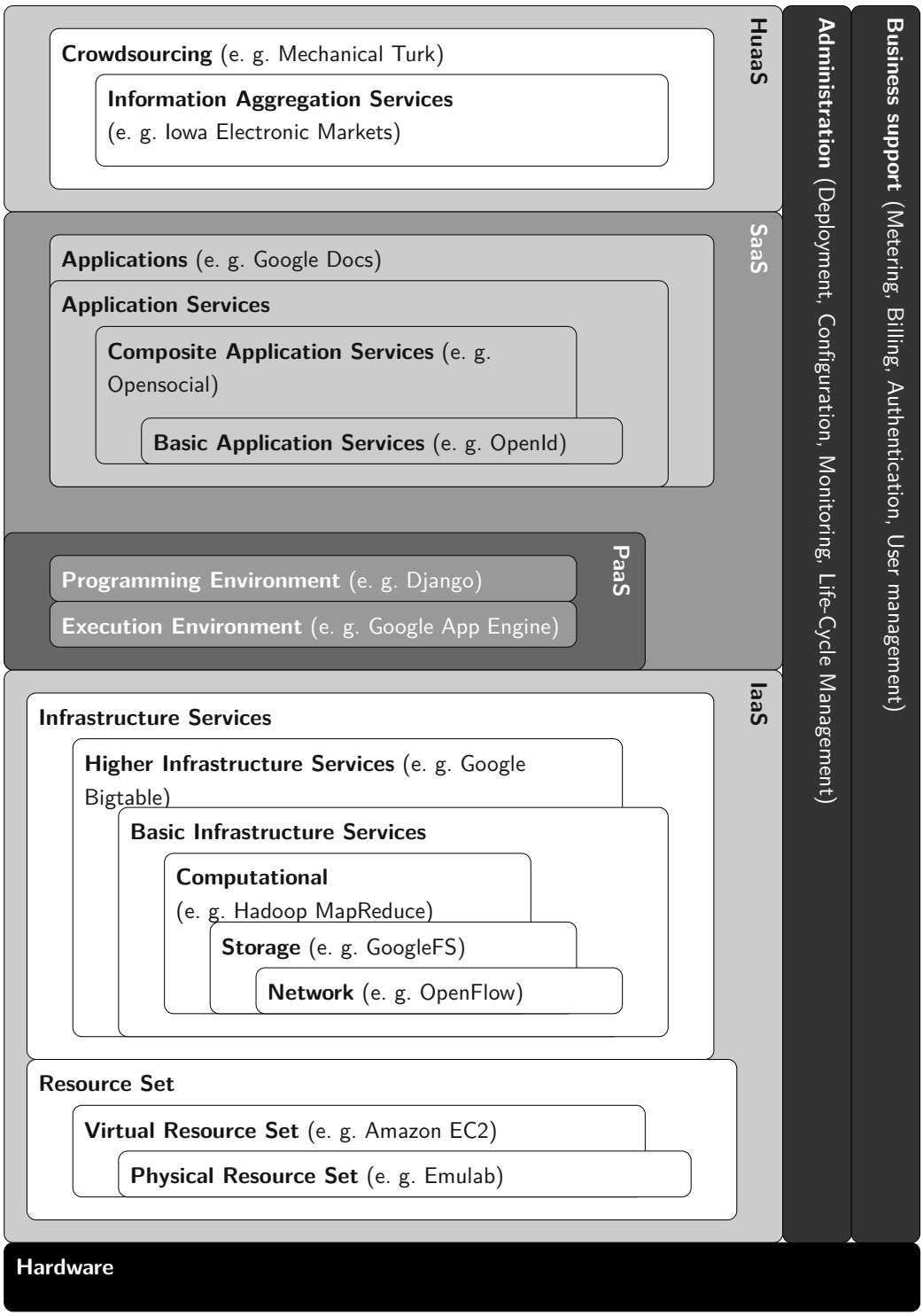


Bild 2.8 Cloud Stack nach Lenk et al. (2009, Seite 25).

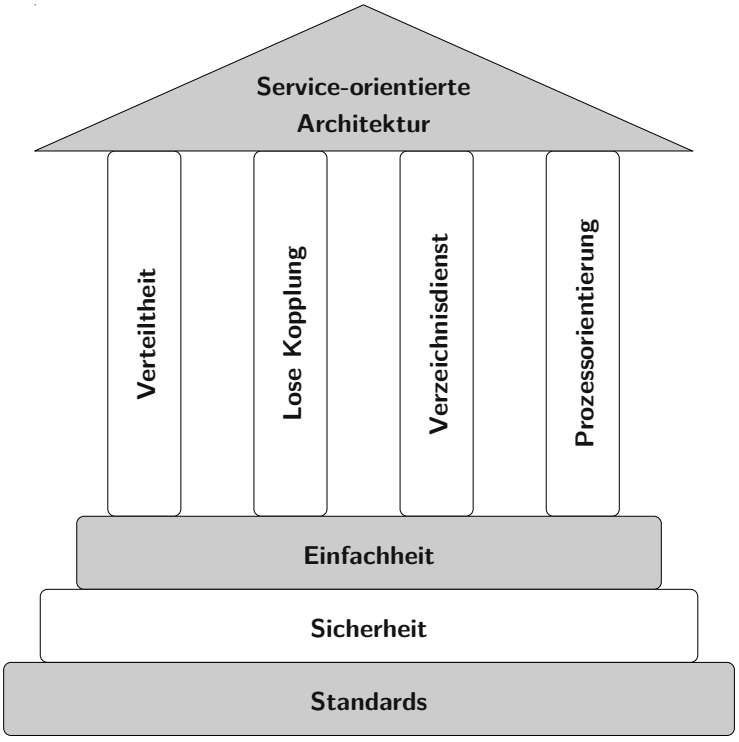


Bild 2.9 Der SOA-Tempel nach Melzer (2008, Seite 13).

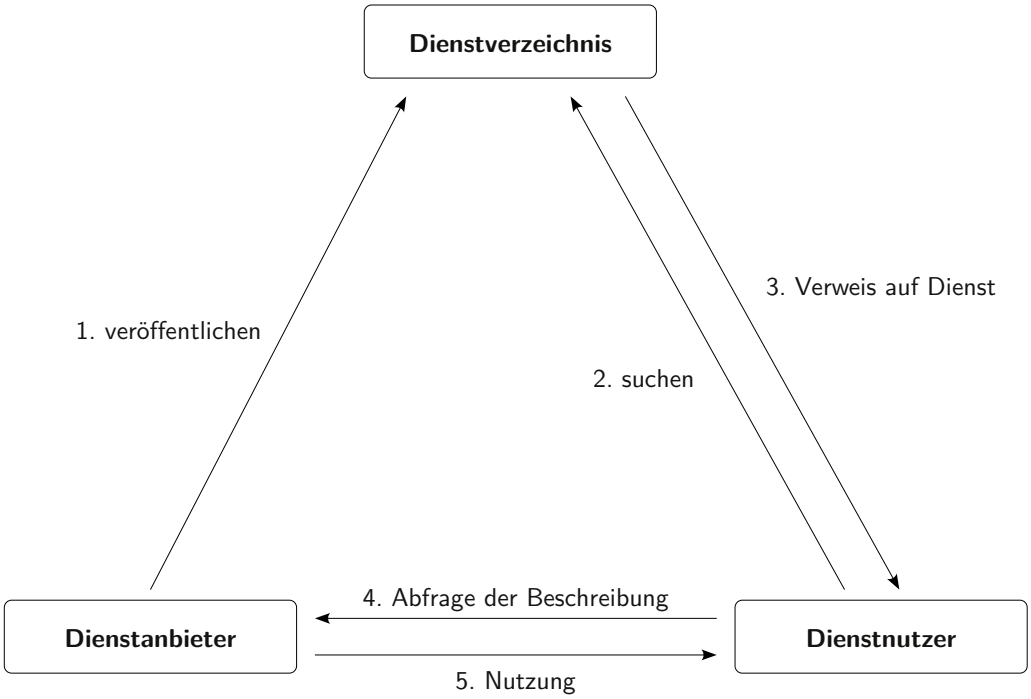


Bild 2.10 Das magische Dreieck einer SOA nach Melzer (2008, Seite 14).

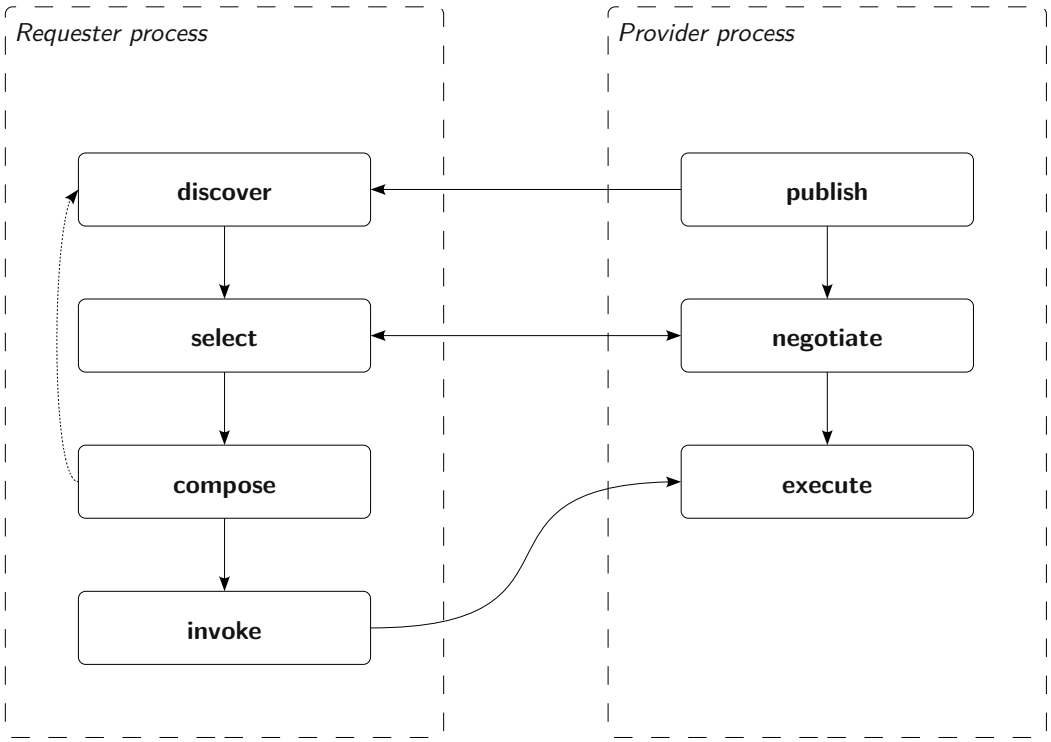


Bild 2.11 Verbindung zu einem Web Service mittels WSDL nach de Bruijn et al. (2008, Seite 19).

3 Herausforderungen im Kontext von Social Networking

Social Networking nimmt eine gewichtige Rolle im Internet ein und wird diese Rolle – zumindest kurzfristig – nicht abgeben. Dabei wurde das Internet selbst in den vergangenen Jahren zu einem Leitmedium, über das immer mehr Menschen Informationen konsumieren, aber auch selbst generieren. In diesem Kapitel wird gezeigt, welchen technischen, sozialen, ökonomischen, rechtlichen und auch ökologischen Herausforderungen sich Social Networking stellen muss. Aus ihnen werden Anforderungen abgeleitet, an denen sich auch die allgemeinen Konzepte hinter dezentral organisiertem Social Networking messen müssen. Beispiele zentral und dezentral organisierter OSNs verdeutlichen das Bild.

3.1 Überblick

Bevor die Herausforderungen identifiziert und daraus Anforderungen abgeleitet werden, gilt es zunächst, sich einen Überblick über die derzeitige Lage des Social Networkings zu verschaffen. Ebendieser Überblick ist wichtig, um zu verstehen, welche – beispielsweise ökonomische – Dimension dieses Thema umfasst. Dazu zählt nicht nur der Status Quo, sondern auch ein Blick in die bisherige Entwicklung des Social Networkings.

3.1.1 Statistiken

Eine Analyse, wie Menschen das Internet im Allgemeinen und Social Networking im Speziellen nutzen, ist Gegenstand dieses Kapitels. Die aufgezeigten Statistiken geben ein Bild über das Nutzungsverhalten verschiedener Gruppierungen wieder. Vorweg ist zu erwähnen, dass manche der folgenden Zahlen keinen wissenschaftlichen Anspruch erheben, da die Bezifferungen oftmals auf Schätzungen und Hochrechnungen beruhen. Dennoch zeigen die Zahlen eine Tendenz, die eine grobe Einordnung der zu entwickelnden Architektur erlaubt.

Die Anzahl der Menschen, die das Internet nutzen, also „online“ sind, steigt stetig an: Während am Ende des Jahres 2000, sprich nach der Popularisierung des Internets in den

1990er Jahren, nur circa 360 Millionen Menschen Zugang zum Internet hatten (Internet World Stats 2010b), überstieg diese Zahl Anfang 2009 bereits die Milliardengrenze (heise online 2009). In der Mitte des Jahres 2010 sind es bereits knapp 2 Milliarden Menschen (Internet World Stats 2010b). Bei einer geschätzten Weltbevölkerung von circa 6,8 Milliarden Menschen entspricht dies einem Anteil von knapp 29 Prozent, wobei die Anteile in Industrienationen deutlich höher liegen gegenüber Schwellen- und Entwicklungsländern (Internet World Stats 2010b).

Beispielsweise sind gemäß van Eimeren und Frees (2010, Seite 335) in Deutschland im Jahr 2010 knapp 70 Prozent der Bevölkerung online, in der Gruppe der 14- bis 19-Jährigen beträgt der Anteil sogar etwa 99 Prozent. Das Internet wird fast täglich genutzt (5,7 Tage pro Woche) mit einer durchschnittlichen Verweildauer von 136 Minuten pro Tag (van Eimeren und Frees 2010, Seite 337). Werden die Web-Applikationen (dort Online-Anwendungen genannt) betrachtet, ergibt sich folgendes Bild (van Eimeren und Frees 2010, Seite 338): die Kommunikationsmöglichkeiten wie E-Mail, Instant Messaging (IM) oder Foren werden von 84, 29 beziehungsweise 19 Prozent der Online-Nutzer in Deutschland mindestens einmal wöchentlich genutzt, OSNs (dort „Onlinecommunitys“ genannt) von insgesamt 32 Prozent. van Eimeren und Frees (2010, Seite 340). Mit steigender Tendenz fällt fast die Hälfte der online verbrachten Zeit auf die Kommunikation (van Eimeren und Frees 2010, Seite 342). 41 Prozent der Online-Nutzer sind Mitglieder eines OSNs (van Eimeren und Frees 2010, Seit 340), wovon durchschnittlich jeder zwei Profile in unterschiedlichen OSNs besitzt (van Eimeren und Frees 2010, 341–342). Eine differenziertere Betrachtung der Nutzung von typischen Social-Web-Applikationen bieten Busemann und Gscheidle (2010): 39 Prozent der Online-Nutzer in Deutschland sind in privaten OSNs aktiv, 7 Prozent in beruflichen (Busemann und Gscheidle 2010, Seite 362). Weblogs oder Microblogging (dort mit Beschränkung auf „Twitter“) werden von 7 beziehungsweise 3 Prozent genutzt.

Es ist anzunehmen, dass die Nutzung von OSNs weiter ansteigen wird und in einigen Bevölkerungsschichten Nutzungspotenziale vorhanden sind (Busemann und Gscheidle 2010, Seite 367). Werden die Angaben aus Deutschland auf die internationale Nutzung projiziert, ergibt sich ein grobes Bild, welche Stellung und welches Potenzial Social Networking inne hat – aus sozialer Perspektive mit weitreichenden Folgen (vergleiche später Kapitel 3.2).

3.1.2 Zentral organisiertes Social Networking

Der Überblick aus Kapitel 3.1, der vor allem die soziale Perspektive betrachtet, wird in diesem Kapitel um eine technische und eine ökonomische Perspektive erweitert. Es wird konkretisiert, welche Entwicklungen und welchen Einfluss einige der international oder national populären Dienste im Social Web derzeit haben. Es werden Herausforderungen für die Provider und deren Dienste ersichtlich, aus denen Hypothesen abgeleitet werden.

Die hier betrachteten Dienste sind zentral organisiert. Dies bedeutet: Im Hintergrund agiert jeweils ein einziger Provider, der sich für den Dienst verantwortlich zeigt. Er unterhält die nötige Infrastruktur, um einer breiten Basis an Usern typische Social-Networking-Funktionen anzubieten (vergleiche zu den Funktionen später Kapitel 4.1.3). Beschränkt ist diese Auswahl auf einige populäre Provider und deren Dienste, insbesondere OSNs, die derzeit am Markt agieren.

Das derzeit erfolgreichste OSN stellt Facebook dar (Facebook 2010). Es ist weltweit aktiv und hat bereits über 500 Millionen Mitglieder (Internet World Stats 2010a), davon über 14 Millionen in Deutschland (Facebookmarketing.de 2011). Twitter hingegen beschränkt sich auf wenige Social-Networking-Funktionalitäten und stellt das Microblogging (Kapitel 2.1.1) in den Vordergrund (Twitter 2010). Mit 175 Millionen Usern, die 25 Milliarden Statusnachrichten (dort „Tweets“ genannt) in den letzten 12 Monaten abgesetzt haben, erfreut sich Twitter einer hohen Popularität (Swisher 2010), auch wenn in Deutschland lediglich 3 Prozent diesen Dienst aktiv nutzen (Kapitel 3.1.1). Jedoch verfügt Twitter über kein tragfähiges Geschäftsmodell, sondern generiert relativ geringe Umsätze durch Werbung (heise online 2010). Das Unternehmen Google verfügt nicht nur über eine Suchmaschine im Web, sondern über viele weitere, unterschiedliche Dienste für das Social Web, die miteinander verknüpft werden (Google 2011). Google finanziert sich zum großen Teil über Werbung: im 3. Quartal 2010 betrug der Umsatz über 7 Milliarden US-Dollar (USD) bei über 2 Milliarden USD Netto-Gewinn (Google 2010). Beispiele für OSNs für spezielle Interessengruppen sind das US-amerikanische LinkedIn (LinkedIn Corporation 2011) und das deutsche XING (XING AG 2011) für berufliche Interessengruppen.

Die gewählte Auflistung der aufgezählten Provider und Dienste erhebt keinen Anspruch auf Vollständigkeit. Sie dient lediglich als Beispiel zur Erstellung der folgenden Hypothesen:

- Der Markt im „Social Web“ wird von wenigen großen, international agierenden Providern dominiert (Oligopole), wobei Facebook ein Quasi-Monopol im internationalen Markt der OSNs darstellt.
- Die Geschäftsmodelle beruhen zum großen Teil auf Werbung, was nur bei einigen Providern erfolgreich ist.
- In einige Provider wird in mehrstelliger Millionenhöhe investiert. Dies führt zu einer Bewertung in Milliardenhöhe, was ein Risikogeschäft mit ungewissem Ausgang bedeuten kann.
- National agierende Provider verlieren Mitglieder an die international agierenden; OSN mit spezieller Ausrichtung (zum Beispiel beruflich genutzte OSNs oder Kontaktbörsen mit OSN-Funktionalität) verlieren Mitglieder an weniger spezialisierte OSNs.
- Durch die Nutzung der Dienste von Millionen von Usern ist es nötig, in eine aufwendige Infrastruktur zu investieren.

Eine vollständige Überprüfung der Hypothesen kann an dieser Stelle nicht geleistet werden. Wie die Statistiken aus Kapitel 3.1.1 dienen die Hypothesen dazu, einen Überblick über die gegenwärtige Situation des Social Networkings zu erlangen, damit die zu entwickelnde Architektur thematisch eingeordnet werden kann.

3.2 Auswirkungen von Social Networking

Während sich für die beteiligten Stakeholder einerseits viele Vorteile durch Social Networking ergeben – ansonsten wären manche OSNs nicht so erfolgreich –, werden andererseits die negativen Aspekte im Social Web in der Öffentlichkeit diskutiert. Die Folgen der Nutzung dieser OSNs sind noch nicht abzusehen. Doch zunächst gilt es, in diesem Kapitel allgemeine Vorteile von Social Networking herauszustellen, bevor dessen Nachteile beleuchtet werden. Damit wird ein Gesamtbild der Auswirkungen des Social Networkings deutlich, wodurch die Herausforderungen aus Kapitel 3.1.2 ergänzt werden.

OSNs werden „für ihre Mitglieder eine zentrale Online-Anlaufstelle, quasi eine netzbasierte Kommunikationszentrale“, „die um verschiedenste [Services] erweiterbar ist und die zudem andere [Social-Web-Applikationen] [...] in sich vereint.“ (Busemann und Gscheidle 2010, Seite 368). Weiter heißt es, die Grenzen zwischen den unterschiedlichen Diensten verschwimmen. Diese Flexibilität und Zentralisierung bietet für die User demnach erhebliche Vorteile, denn für sie nützliche Funktionen werden sinnvoll in einen Dienst integriert. Durch die Offenheit mancher Dienste (vor allem durch Web-Service-Schnittstellen wie APIs) können weitere miteinander verknüpft werden (sogenannte Mashups; Kapitel 2.2.3). Die Interaktion, Kommunikation und soziale Vernetzung der User kann durch das Social Web gefördert werden, weil reale Grenzen und Distanzen in der virtuellen Welt des Internets potenziell keinen Bestand haben[1]. Die Zusammenarbeit, Nutzung und Generierung von Inhalten wird im Social Web gefördert.

Doch diese vorteilhafte Darstellung verbirgt durchaus negative Aspekte. Am Social Networking nehmen Millionen Menschen teil, wie in Kapitel 3.1 gezeigt. Für sie ist die Kernleistung, sprich die Teilnahme an einem Dienst meist kostenlos. Die Provider agieren jedoch nicht gemeinnützig oder kostendeckend, sondern gewinnorientiert. Daher sind sie auf hohe User-Zahlen und -Aktivitäten angewiesen, denn ihre Geschäftsmodelle beruhen darauf, Daten und Informationen von Usern zu sammeln, um sie gewinnbringend zu verwerten (Beispiel: personalisierte Werbung). Die Teilnahme an einem OSN ist daher keineswegs kostenlos, weil der User bereit ist, dem Provider persönliche Daten zu überlassen, damit dieser aus der Datennutzung Umsätze generieren kann. Anhand der Zahlen zum OSN Facebook (Kapitel 3.1.2) wird dies anschaulich: Bei einer Bewertung von gerundet 50 Milliarden USD und einer – ebenfalls gerundeten – User-Anzahl von 500 Millionen entspricht dies einem Wert von 100 USD pro User.

[1] Dieser Vorteil wird durch das Prinzip der Netzneutralität sichergestellt; Kapitel 1.1.

Das plakative Beispiel ist eher abstrakt und es ist durchaus legitim, dass Unternehmen marktwirtschaftlich agieren. Deutlich realer sind die Missbrauchs- und Gefahrenpotenziale: Es betrifft vor allem den Schutz der persönlichen Daten, die die User den Providern offerieren. In den Teilnahmebedingungen, oftmals Allgemeine Geschäftsbedingungen (AGB) genannt, werden vielen Providern weitreichende Nutzungs- und Verwertungsrechte eingeräumt (Schmieder 2010). Bei einigen Services hat der User keine Kontrolle darüber, wo, wie und wie lange seine Daten gespeichert und an wen sie weitergegeben werden. Die Komplexität mancher Dienste erhöht zudem den Kontrollverlust, das heißt, wird die Nutzung eines Dienstes oder die Verflechtung mit anderen Providern/Diensten für den User nicht ersichtlich, kann er nicht oder nur teilweise nachvollziehen, was mit seinen Daten geschieht. Ebenso kann der User nicht immer seine einmal im Internet freigegebenen Daten wieder löschen (Spiekermann und Cranor 2009, Seite 72).

3.3 Anforderungen an Online Social Networks und deren Architekturen

Die in Kapitel 3.1.2 vorgestellten Social-Networking-Dienste dienen als Musterbeispiele, welchen Herausforderungen sich (insbesondere) OSNs und deren zugrunde liegende Architekturen stellen müssen und voraussichtlich stellen werden. Darauf aufbauend und im Hinblick auf die in Kapitel 3.2 aufgezeigten (positiven wie negativen) Aspekte lassen sich Anforderungen ableiten, wie ein OSN und dessen Architektur gestaltet werden muss.

Die Anforderungsdomäne sind OSNs inklusive ihrer Provider. Als Stakeholder, sprich alle Akteure, die am OSN partizipieren, gelten wiederum die Provider, aber auch die User und weitere Akteure, die beispielsweise mit Bezug zum OSN geschäftlich mit den Providern kooperieren. Die Anforderungsanalyse betrachtet fünf Perspektiven, die zu anfangs (Kapitel 1) bereits erwähnt wurden: die technische, soziale, ökonomische, rechtliche und ökologische. Wie dort ebenfalls erwähnt, lassen sich die Perspektiven nicht klar voneinander abgrenzen. Um die Diskussion jedoch zu vereinfachen, werden Anforderungen denjenigen Perspektiven zugeteilt, zu denen sie am besten passen.

3.3.1 Technische Anforderungen

In Anlehnung an die Anforderungen an Software nach Sommerville (2001, Seite 97–115) würden in dieser Perspektive die funktionalen und nicht-funktionalen Anforderungen, die Domänen-, User- sowie Systemanforderungen beleuchtet werden. Doch diese Herangehensweise wäre auf Software zugeschnitten. In dieser Arbeit hingegen geht es um Architekturen innerhalb der Domäne Internet/Web. Dort stellen sich die Anforderungen, wie sie schon

in Kapitel 2 angerissen wurden und an dieser Stelle zusammengetragen und verfeinert werden.

Zu den Anforderungen zählen:

Skalierbarkeit: Die Architektur eines OSNs muss mit den Aktivitäten von Millionen von Usern umgehen können (Stichwort: hohe Bandbreiten).

Performanz: Die Architektur muss nicht nur auf viele User zugeschnitten sein (Skalierbarkeit), sondern auch deren Anfragen möglichst schnell beantworten können (Stichwort: niedrige Latenzen).

Verfügbarkeit, Robustheit, Korrektheit, Zuverlässigkeit: Die Funktionen innerhalb der Architektur müssen möglichst fehlerfrei, ohne Ausfälle und tolerant gegenüber Fehlbedienungen sein.

Sicherheit: Die Architektur muss den gängigen Anforderungen an IT-Sicherheit genügen, wozu verschiedene Formen von Sicherheiten und Schutzziele zählen (Eckert 2008, Seite 4–13), um Bedrohungen von Informationssystemen (Eckert 2008, Seite 41–82) und Bedrohungen im Internet (Eckert 2008, Seite 83–156) abzuwenden.

3.3.2 Soziale Anforderungen

Die soziale Perspektive betrifft die User und Stakeholder (Kapitel 2.1.1) der OSNs sowie deren Architekturen. Die Anforderungen beruhen auf den Betrachtungen aus Kapitel 3.2.

Es lassen sich folgende Anforderungen identifizieren:

Datenmanagement: Daten sind als die wichtigste Ressource im Web (Kapitel 2.1) anzusehen. Die User benötigen Werkzeuge, die das vollständige Management der Daten, die sie einem OSN zur Verfügung stellen, effektiv und effizient ermöglichen. Dazu gehört, dass Daten jederzeit und überall bereitgestellt und abgerufen, bearbeitet und gelöscht werden können. Datenflüsse sollen nachvollziehbar und beeinflussbar sein. Das Management umfasst Funktionen, damit keine Lücken in der Handhabung entstehen.

Zielgruppenerreichung: Allen Interessen- und Zielgruppen, die der Provider des OSNs potenziell erreichen möchte, soll er neutral und offen gegenüber stehen.

Social-Networking-Funktionen: Das OSN erfüllt die derzeit gültige Grundfunktionalität und ermöglicht es, diese Funktionalität bedarfsgerecht anzupassen und zu erweitern.

Usability und Accessability: Die Benutzung des OSNs, insbesondere das Datenmanagement muss so einfach wie möglich gestaltet werden, damit jeder User am OSN teilnehmen kann.

3.3.3 Rechtliche Anforderungen

Die rechtlichen Anforderungen umfassen zahlreiche nationale Gesetze, Richtlinien oder ähnlich Verbindliches. Werden auch ausländische User für die Teilnahme am OSN akzeptiert, kommen weitere internationale Gesetze und so weiter hinzu, die berücksichtigt werden müssen. Darüber hinaus sind politische, religiöse oder auch ethnische Faktoren einzubeziehen. Sie dienen einem Provider als Rahmenbedingungen für den Betrieb eines OSNs.

Die oben genannte Vielzahl an rechtsverbindlichen Dokumenten werden für diese Arbeit zur besseren Lesbarkeit unter den Begriff der rechtlichen Bestimmungen zusammengefasst. Da eine jeweilige Betrachtung den Rahmen dieser Arbeit überschreiten würde, werden lediglich allgemeingültige Anforderungen formuliert:

Einhaltung der rechtlichen Bestimmungen: Die Einhaltung der rechtlichen Bestimmungen ist eher eine Voraussetzung als eine Anforderung. Vielmehr gilt es, die Bestimmungen, die ein User dem OSN auferlegt, zu erfüllen. Darunter fällt zum Beispiel, dass ein europäischer User von einem OSN erwartet, das seine Daten auf Hardware mit europäischem Standort gespeichert werden, damit nicht andere Bestimmungen gültig werden, die aus Sicht des Users negative Folgen mit sich brächten. Vielmehr soll der User die Wahl haben, an welchem Standort seine Daten gespeichert werden.

Gefahren- und Missbrauchspotenziale minimieren: Neben technischen Aspekten der IT-Sicherheit (Kapitel 3.3.1) gilt es vor allem, die Privatsphäre und informationelle Selbstbestimmung eines jeden Users zu respektieren (vergleiche später Kapitel 4.1.1.4). Die Architektur des OSNs sollte so gestaltet sein, dass Gefahren und Missbrauchsfälle möglichst ausgeschlossen oder zumindest minimiert werden.

Einsatz für die Netzneutralität: Aus eigenem Interesse sollte ein Provider den neutralen Transport der eigenen Netzwerkpakete im Internet fordern und fördern.

Verständliches Kommunizieren eigener Bedingungen: Provider-eigene Richtlinien, AGB und (Nutzungs-)Bestimmungen (im Englischen oftmals mit „Terms of Condition“, „Privacy Policies“ oder ähnlichem betitelt) müssen auf der einen Seite die rechtlichen Bestimmungen berücksichtigen, auf der anderen Seite müssen sie für die User allgemein-verständlich konzipiert sein. Alle Akteure – so auch Maschinen – sollten die Bestimmungen „verstehen“ können.

3.3.4 Ökonomische Anforderungen

Durch die hohe Verbreitung des Social Networkings im Internet (Kapitel 3.1) entstehen auf Providerseite teils erhebliche Aufwendungen, die gedeckt werden müssen, da ansonsten

die Services mittel- oder langfristig nicht angeboten werden können[2]. Es ist daher legitim, dass Provider zumindest kostendeckend agieren und dafür Geschäftsmodelle entwickeln.

Doch diese Kommerzialisierung zieht einige Anforderungen nach sich, die aus ökonomischer Perspektive berücksichtigt werden müssen:

Tragfähige Geschäftsmodelle: Im Social Web gelingt es scheinbar nicht jedem Provider, ein tragfähiges Geschäftsmodell aufzubauen, sodass die Services durch Fremdkapital finanziert werden müssen. Es ist daher für einen Provider existenziell, ein Geschäftsmodell zu entwickeln, das tragfähig ist.

Neuartige Geschäftsmodelle: Viele der tragfähigen Geschäftsmodelle sind aus rechtlicher und sozialer Sicht kritisch zu hinterfragen, denn sie beruhen auf den Daten der User. Wiederum sollten diese geschützt werden (Kapitel 3.3.3) – ein Widerspruch, weshalb andere, neuartige Geschäftsmodelle zu entwickeln und zu etablieren sind.

Corporate Social Responsibility (CSR): Organisationen stehen in einer sozialen Verantwortung, der sie nachkommen müssen (Busch 2007). Kommerzielle Interessen müssen demnach sozial und ökologisch vertretbar sein.

3.3.5 Ökologische Anforderungen

Zu guter Letzt wird die ökologische Perspektive herangezogen, um geeignete Anforderungen zu formulieren. Im Rahmen des Themas Green IT wurde in den letzten Jahren die Bedeutung einer nachhaltigen, weil ökologisch, sozial und ökonomisch sinnvollen IT diskutiert. Durch die Verlagerung von Logik und Daten in Netzwerke wie das Internet (Kapitel 2.2) ist Green IT für Internet und Web Services, insbesondere aus dem Social-Networking-Bereich, ein Thema, das nicht mehr ignoriert werden kann.

Daher werden einige Anforderungen aufgezählt, die hier der ökologischen Perspektive zugeschrieben werden (auch wenn eine Beschränkung der Themen Green IT und Nachhaltigkeit auf ökologische Aspekte nicht gerecht wird):

Hardware: Bei der Auswahl der Hardware für den Betrieb von OSNs soll darauf geachtet werden, dass sie energieeffizient ist und sich in einen nachhaltigen „Lebenszyklus“ (knapp formuliert: von der Rohstoffgewinnung, über die Produktion, über den Betrieb bis zum Recycling) einbettet.

Software: Auf der Hardware soll Software zum Einsatz kommen, die schonend mit Ressourcen umgeht. Da dies sich nicht bei jeder Software beeinflussen lässt, weil deren Hersteller es nicht vorsieht, sollte der Einsatz von anpassbaren FLOSS-Komponenten bevorzugt werden.

[2] In der schnelllebigen IT-Branche ist die zeitliche Einordnung in kurz-, mittel- und langfristig eine komplexe Angelegenheit, die hier nicht weiter betrachtet wird.

3.4 Nachteile durch zentral organisiertes Social Networking

Aufbauend auf den in Kapitel 3.2 und 3.3 diskutierten Aspekten, die sich auf das Social Networking im Allgemeinen beziehen, stehen im Folgenden zentral organisierte OSNs im Fokus. Es werden einige Punkte offen gelegt, die aufzeigen, inwiefern diese Form des Social Networkings limitiert ist und dass die OSNs den Anforderungen derzeit nur unzureichend genügen können.

Bereits in Kapitel 3.1.2 wurde erläutert, wie ein zentral organisiertes OSN beschaffen ist. Es gibt lediglich einen Provider, der das OSN betreibt, die Entwicklung vorantreibt und letztlich von der Strategie bis zum operationalen Tagesgeschäft alle Schritte vorgibt und steuert. Dabei stehen kommerzielle Interessen über den Interessen Dritter, beispielsweise der User. Diese Faktoren führen zu wesentlichen Nachteilen zentral organisierten Social Networkings, die hier nach den fünf Perspektiven aus Kapitel 3.3 aufgeschlüsselt werden.

Die technische Perspektive (Kapitel 3.3.1): Der Provider eines populären Services benötigt eine hohe Anzahl an Ressourcen, um den Service anbieten zu können, ihn weiterzuentwickeln und an sich ändernde Faktoren (zum Beispiel steigende User-Zahlen) anzupassen. Der Provider ist in diesen Punkten auf sich allein gestellt und ist folglich für die gesamte technische Infrastruktur verantwortlich. Theoretisch mag die Architektur mit den wachsenden Anforderungen skalieren und weiterhin performant sein, doch die Umsetzung von der Theorie in die Praxis erfordert hohe Investitionen. Ist der Service von Sicherheitslücken oder sogar einem Ausfall betroffen, wird diese zentralisierte Architektur zu einem Flaschenhals.

Die soziale Perspektive (Kapitel 3.3.2): Die User haben bei einem zentral organisierten Service einen einzigen Ansprechpartner, doch sind sie von seinem „Wohlwollen“ abhängig. Das bedeutet, es bestimmt allein der Provider, welche Funktionen angeboten und wie diese genutzt werden, welches Datenmanagement zum Einsatz kommt, et cetera. Der User kann versuchen, durch Zusammenschluss mit anderen (auch durch Hinzugewinnen der Presse oder öffentlichen Verwaltungen, Verbraucherschützern und so weiter) auf den Provider Druck auszuüben. Der Erfolg dieses Vorgehens kann zum Beispiel durch die Marktmacht des Providers und/oder seine Sitze im Ausland, wo andere Bestimmungen gelten oder möglich sind, limitiert sein.

Die rechtliche Perspektive (Kapitel 3.3.3): Ein zentral organisiertes OSN speichert die Daten eventuell nur in einem Rechenzentrum, das seinen Standort in einem Land hat. Folglich gelten erst einmal nur die dortigen rechtlichen Bestimmungen. Das kann sich für ausländische User negativ auswirken, für die andere rechtliche Bestimmungen gelten. Will ein Provider dies in Einklang bringen, müssen ausländische Standorte für die eingesetzte Hardware eröffnet werden, was zusätzliche Aufwendungen nach sich zieht.

Die ökonomische Perspektive (Kapitel 3.3.4): Ein tragfähiges Geschäftsmodell gelingt nicht jedem Provider, da die hohen Ausgaben für die Infrastruktur durch entsprechende Einnahmen gedeckt werden müssen. Die Generierung von Umsätzen durch (personalisierte) Werbung schreckt diejenigen (potenziellen) User ab, die sich um den Datenschutz sorgen.

Die ökologische Perspektive (Kapitel 3.3.5): Populäre Services benötigen performante Hardware-Ressourcen, die aufgrund der hohen Nutzung stark beansprucht werden. Die Ressourcen müssen ein möglichst gutes Preis-/Leistungsverhältnis haben, um finanzierbar zu sein. Bei hohem Investitionsvolumen ist es für die Provider ökonomisch wenig sinnvoll, in relativ teure, aber ökologisch sinnvolle Ressourcen zu investieren.

Als Zwischenfazit für diese Arbeit wird wird aufgrund der zahlreich aufgezeigten Nachteile folgende Hypothese aufgestellt: Zentral organisiertes Social Networking genügt nicht den derzeitigen Anforderungen. Die in diesem Bereich agierenden Provider von OSNs haben offensichtlich die falschen Strategien (und Architekturen) gewählt, um die Bedürfnisse aller Stakeholder zu befriedigen – wenn sie denn *alle* Stakeholder befriedigen möchten. Auch wenn sich die Services durchaus erfolgreich am Markt behaupten können, so deutet sich über die letzten Jahre eine Fehlentwicklung an.

3.5 Dezentral organisiertes Social Networking

Die in Kapitel 3.3 erarbeiteten Anforderungen werden, wie in Kapitel 3.4 gezeigt, nur ungenügend durch zentral organisiertes Social Networking erfüllt. Seit wenigen Jahren gibt es Bestrebungen, dezentral organisierte OSNs zu entwickeln und gegen die etablierten „Größen“ aufzustellen. Nach einer Definition von dezentral organisiertem Social Networking werden einige Projekte dieser Art von OSNs im Folgenden vorgestellt, denn sie versprechen, die abgesteckten Anforderungen zu erfüllen.

Dezentral organisiertes Social Networking unterscheidet sich von dem zentral organisierten dahingehend, dass es eine Abkehr von einem Provider hin zu mehreren bis beliebig vielen gibt. Ist der angebotene Service offen im Internet zugänglich, kann jeder Akteur ein Teil des Services werden und auch als Provider agieren, um anderen Usern Zugang zum Service zu ermöglichen.

Seit mehreren Jahren existieren einige Projekte, die versuchen, dezentral organisierte OSNs zu etablieren. Diese Projekte entwickeln Software, die sie unter eine FLOSS-Lizenz stellen, sodass sich jeder Interessierte die Software beschaffen, selbst installieren und ausprobieren kann. Zu ihnen zählen unter anderem die Projekte Diaspora (Grippi et al. 2010), KDE Social Desktop (KDE Social Desktop 2009), NoseRub (NoseRub 2010), Friend2Friend (Friend2Friend 2010), DiSo (DiSo 2010), kopal (kopal 2010), mensch.coop (mensch.coop

2010), Appleseed (The Appleseed Collective 2008), HelloWorld (HelloWorld 2008) und OneSocialWeb (Vodafone Group 2010).

4 Entwicklung eines dezentral organisierten Online Social Networks

Der Fokus dieser Arbeit liegt auf der Entwicklung eines dezentral organisierten OSNs, das aus den in Kapitel 3.3 aufgeführten Anforderungen abgeleitet ist und sich von zentral organisierten wie anderen dezentral organisierten OSNs (Kapitel 3.1.2 und 3.5) unterscheidet. In diesem Kapitel werden die dahinter stehende Idee und Konzeption mit den Besonderheiten der Architektur sowie der Organisation beschrieben.

4.1 Idee und Konzeption

Zunächst wird der Rahmen des OSNs geschaffen. Er besteht aus verschiedenen elementaren Konzepten. Darauf aufbauend werden die Grundfunktionen des OSNs eingegrenzt, die nicht nur auf „klassisches“ Social Networking abzielen, sondern auch breit gestreute Ziel- und Interessengruppen ansprechen sollen.

4.1.1 Konzepte

Im Folgenden werden die Konzepte der Dezentralisierung, der Offenheit und Transparenz, der verschiedenen Sicherheitskonzepte und der User-Zentrierung erläutert. Sie beschreiben abseits der technischen Aspekte, dass sie, miteinander kombiniert, eine Möglichkeit darstellen, den Anforderungen an dezentral organisierte OSNs möglichst vollständig zu genügen und damit eine reale Alternative zu zentral organisierten OSNs darstellen.

4.1.1.1 Dezentralisierung

Das Konzept der Dezentralisierung führt die Ideen eines ebenfalls dezentral organisierten Netzwerks fort, nämlich die des Internets. Im Gegensatz zu bisherigen zentral organisierten OSNs werden die Rollen der Stakeholder und der hierarchischen Strukturen aufgeweicht. Es gilt, positive Eigenschaften zu bündeln, aber Aufwendungen und Risiken zu verteilen.

Die Architektur des Internets ist dezentral aufgebaut und besteht bekanntermaßen aus vielerlei kleineren Netzen, die jeweils als Autonomous System (AS) bezeichnet werden und an Austauschknoten miteinander verbunden werden können. Auf dem Internet aufbauende Systeme auf Applikationsebene wie DNS oder E-Mail führen die Dezentralisierung fort. Das dezentral organisierte OSN überträgt dieses Konzept auf das Web und unterteilt die Architektur in beliebig viele kleinere AS, an denen (wiederum) beliebig viele Peers gebunden sein können. In Kapitel 4.2 werden die technischen Komponenten der Architektur entwickelt, sodass an dieser Stelle das Konzept weiter erläutert werden kann.

Im dezentral organisierten OSN verschwimmt die Grenze zwischen User und Provider. Auf der einen Seite ist jeder User befähigt, auch Provider für eine beliebige Anzahl User zu sein. Auf der anderen Seite können Provider gleichzeitig als User agieren. Des Weiteren ist eine Verschachtelung möglich, sodass eine freiwillig vereinbarte Kette oder Hierarchie von Providern entsteht. Es entsteht eine n-zu-n-Beziehung zwischen der Eintität Stakeholder und der Entität Rolle: Ein Stakeholder kann eine oder beliebig viele Rollen einnehmen; eine Rolle kann von einem oder beliebig vielen Stakeholdern eingenommen werden (Bild 4.1).

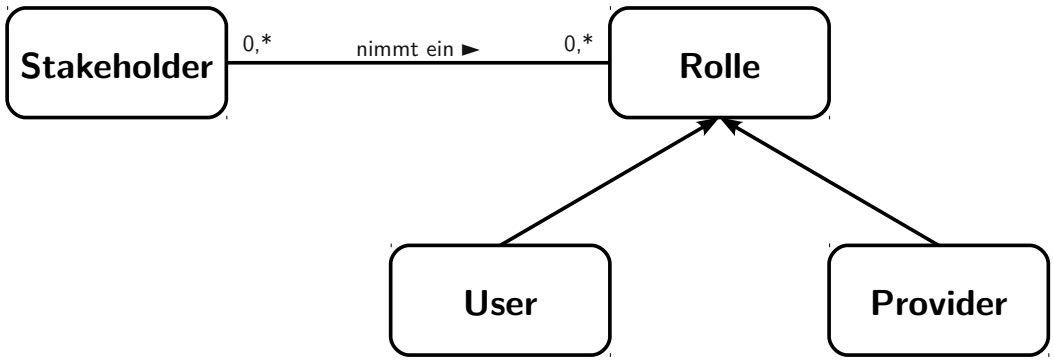


Bild 4.1 Die Relation zwischen den Entitäten Stakeholder und Rolle.

Zwingendes Ziel ist die Gleichberechtigung der Gruppen. Die Hierachien werden aufgeweicht, sodass nicht die Handlungen und Interessen einer Organisation die Handlungen und Interessen anderer signifikant beeinflusst (oder sogar einschränkt). Vielmehr werden gemeinsame Interessen und Ziele geschaffen, die von den Stakeholdern mitgetragen werden. Ziel ist es, mit diesen Vorgaben niemanden in der Art der Partizipation einzuschränken, sondern die Vorgaben so zu gestalten, dass die individuelle Freiheit der Stakeholder gefördert wird. In Kapitel 4.3 werden die organisatorischen Strukturen weiter verfeinert.

4.1.1.2 Offenheit

Das Konzept der Offenheit beschreibt, dass das OSN für alle Stakeholder einen Raum zur Entfaltung ihrer Interessen bietet. Erreicht wird dies durch den gleichzeitigen Abbau

von Konventionen und den Aufbau von Funktionen, die jene Entfaltungsmöglichkeiten fördern.

„One Size Fits All“ bedeutet bei einem Kleidungstück, dass alle Größenunterschiede berücksichtigt werden. Auf die Architektur eines dezentral organisierten OSNs bezogen bedeutet diese Aussage, dass sich die Architektur den Stakeholdern maximal öffnet: Ziel ist es, Stakeholdern die gleichen Rechte einzuräumen, ihre Interessen in die Entwicklung der Architektur einzubringen, damit ihre Bedürfnisse befriedigt werden und kein potenzieller Stakeholder ausgeschlossen wird. Dieser Anspruch klingt recht abstrakt, spiegelt aber letztlich die Freiheiten und Rechte in einer demokratischen Gesellschaft oder in einer Marktwirtschaft wieder.

Zu diesem Anspruch gehört die Abkehr von Konventionen. Faltin (2008, Seite 30) beschreibt aus ökonomischer Perspektive, dass ein erfolgversprechendes Konzept die Funktionalität in den Vordergrund stellt, statt den üblichen Konventionen zu folgen. Daher erscheint es sinnvoll, manche Konventionen, denen ein „klassisches“ OSN folgt, abzubauen und Grenzen aufzuweichen. Konkrete Ansatzpunkte sind:

- Die klare Trennung zwischen Usern und Providern wird aufgehoben (was sich in das Konzept der Dezentralisierung in Kapitel 4.1.1.1 einbettet).
- Die Prozesse innerhalb der Architektur sind transparent und von allen Stakeholdern nachvollziehbar zu gestalten (vergleiche später Kapitel 4.1.1.3).
- Die Architektur öffnet sich neuen Features und adaptiert Features verschiedener Online- und Offline-Applikationen (vergleiche später Kapitel 4.1.3).
- Die Entwicklung der Architektur ist offen zu gestalten (vergleiche später Kapitel 4.3).
- Die Beschreibungen der Ideen und Konzepte, der Architektur (vergleiche später Kapitel 4.2) und der Organisation sind bewusst nicht allzu detailiert formuliert, um den Stakeholdern Möglichkeiten zur gemeinsamen Konsensfindung und zur individuellen Umsetzung zu bieten.

Das Konzept der Offenheit hat allerdings seine Grenzen: Von einem Stakeholder, insbesondere von einem User kann nicht verlangt werden, alle ihn betreffenden Daten zu veröffentlichen. Dies würde die Sicherheitskonzepte verletzen (vergleiche später Kapitel 4.1.1.4).

4.1.1.3 Transparenz

Die Architektur und die Prozesse innerhalb der Architektur unterliegen sind transparent gestaltet. Damit ist nicht gemeint, dass die Daten der User offen für alle zugänglich sind.

Eine „durchsichtige“, sprich „transparente“ Architektur beschreibt das Konzept, sodass alle Prozesse innerhalb der Architektur derart gestaltet sind, dass sie für die Stakeholder jeder-

zeit nachvollziehbar sind. Dies beginnt bei der Strategie, die die Stakeholder gemeinsam in einem stetigen Prozess verfolgen, wird auf die Gestaltung (das Design) der Architektur sowie auf die Entwicklung der Komponenten übertragen und wirkt sich ebenfalls in der Nutzung des OSNs aus, was die Datenverbindungen, -verarbeitung und -speicherung betrifft. So sind die Bestimmungen der Provider (Policies) leicht verständlich, aber trotzdem umfassend und detailiert, rechtssicher und individuell verträglich. Als Beispiel für eine solche Transparenz ist die Lizensierung nach Creative Commons (Creative Commons 2010a): Die Lizenz existiert in einer für Menschen leicht verständlichen Form, in einer ausführlichen, rechtlichen Anforderungen genügenden Form und in maschinen-lesbarer Form.

Das Konzept der Transparenz räumt den Stakteholdern – ähnlich wie das Konzept der Offenheit (Kapitel 4.1.1.2) – weitreichende Rechte und Pflichten ein, was die Partizipation am OSN betrifft. Diese Transparenz ist allerdings nicht auf die Stakeholder, Akteure und ihre Daten anzuwenden. Das Recht auf Transparenz darf nicht die Privatsphäre oder andere Rechte beschneiden. So kann von einem User nicht verlangt werden, dass er einen Teil seiner Privatsphäre aufgibt, um im Social Web aktiv zu sein, wie es bei zentral organisierten OSNs, deren Geschäftsmodelle sich auf die Verarbeitung von persönlichen User-Daten stützen (Kapitel 3.4), der Fall ist. Vielmehr soll die Transparenz die Stakeholder unterstützen, statt ihnen potenziell zu schaden.

4.1.1.4 Sicherheit

Die Architektur vereint mehrere Sicherheitskonzepte, die Sicherheit schaffen und erhalten sollen. Dazu zählen die Sicherung und Einhaltung des Datenschutzes, das Schaffen von Vertrauen und die Einhaltung von Rechten und gesetzlichen Bestimmungen.

Den Konzepten der Offenheit und der Transparenz sind Grenzen gesetzt (Kapitel 4.1.1.2 und 4.1.1.3): Sicherheitskonzepte sollen verhindern, dass nicht alle Daten kompromisslos offen zugänglich und nicht alle Aktivitäten innerhalb – und wenn möglich auch außerhalb – des OSNs transparent für alle gemacht werden. Die Konzepte verschreiben sich der Umsetzung der rechtlichen Anforderungen aus Kapitel 3.3.3: Es gilt, alle rechtlichen Bedingungen zu erfüllen und zu bewahren, Gefahrenpotenziale zu minimieren und Missbrauch zu verhindern.

Eine rechtliche Bestimmung erfährt im Social Web eine besondere Bedeutung: der Datenschutz (im Sinne des englischen Begriffs Data Protection). Er dient dem Schutz der Privatsphäre (englisch: Privacy), der auf Informationssysteme übertragen nicht nur die Kontrolle über den Zugriff auf eine Person oder deren Daten umfasst, sondern auch die Risikominimierung, dass persönliche Daten der Person ungewollt verwendet werden (Spiekermann und Cranor 2009, Seite 68). Privatsphäre wird nach Spiekermann und Cranor (2009, Seite 68–69) in drei technisch orientierte Sphären aufgeteilt: die User-Sphäre umfasst das Gerät des Users (zum Beispiel ein Notebook), die Joint-Sphäre die Infrastruktur der Pro-

viderservices und die Recipient-Sphäre das Netzwerk zwischen User und Provider. In der deutschen Rechtsprechung wird zwischen Intim-, Privat- und Sozialsphäre unterschieden (Paschke 2009, Seite 336–339). Bei der Entwicklung der Architektur gilt es, alle Sphären zu berücksichtigen und zu respektieren. Darüber hinaus gelten (zumindest in Deutschland) weitere Grundrechte, die die persönlichen Daten der User betreffen: der Schutz der informationellen Selbstbestimmung, der Schutz gegen ungewollte Kommerzialisierung und das Grundrecht auf Gewährleistung der Vertraulichkeit und Integrität informationstechnischer Systeme, kurz IT-Grundrecht (Paschke 2009, Seite 339–344).

In der sogenannten Hackerethik wird zwischen privaten und öffentlichen Daten unterschieden (Chaos Computer Club 2011): „Öffentliche Daten nützen, private Daten schützen." Dieser Leitspruch findet sich auch in der Architektur wieder. Neben dem Schutz der privaten Daten gilt es, rechtlich unbedenkliche Veröffentlichungen zu nutzen, das heißt, Daten, die innerhalb der Architektur veröffentlicht werden, sollen möglichst vielen Akteuren schnell und unkompliziert zur Verfügung stehen.

Da die Privatsphäre sehr weitreichenden Einfluss auf die Gestaltung der Architektur hat, integriert sie zwei Konzepte: Privacy-by-Architecture und Privacy-by-Design (Spiekermann und Cranor 2009, Seite 73–74). Ersteres setzt auf Datensparsamkeit, das heißt, es werden – je nach Feature (vergleiche später Kapitel 4.1.3) – nur die nötigsten persönliche Daten abgefragt, verarbeitet und gespeichert. Zweiteres legt fest, wie der Umgang mit bereits vorhandenen persönlichen Daten erfolgt: Der User wird über die Nutzung informiert; er hat die Wahl, ob Daten erhoben werden, und kann auf sie zugreifen, sie bearbeiten und löschen; er hat die Gewissheit, dass die Daten geschützt werden (Spiekermann und Cranor 2009, Seite 73).

Zudem gilt es, Vertrauen (Trust) zwischen den Stakeholdern aufzubauen. Begünstigt durch die Konzepte der Offenheit und Transparenz kann eine Vertrauensbildung zum OSN, zur Architektur und zu den organisatorischen Strukturen geschaffen werden. Das Vertrauen zwischen Usern kann durch kryptografische Funktionen gefördert werden.

Neben dem Datenschutz gibt es nach Eckert (2008, Seite 4–5) drei weitere Typen von Sicherheit: Funktionssicherheit (Safety), Informationssicherheit (Security) und Datensicherheit (Protection). Unterstützt werden diese abermals durch geeignete kryptografische Funktionen und durch Authentifizierung sowie Autorisierung von Akteuren. Weiterhin gilt es, die vom User gewollte Anonymität oder Pseudonymität zu berücksichtigen und eine vom User ungewollte Identifikation, zum Beispiel durch Tracking (Verfolgen von User-„Wegen" im Netzwerk), zu vermeiden (Spiekermann und Cranor 2009, Seite 74).

Die Sicherheitskonzepte gelten ebenso zur Einhaltung und Durchsetzung von Nutzungs-, Urheber- und weiteren Medienrechten. Allerdings ist hier auf unerwünschte Nebeneffekte von der Beschneidung anderer Rechte bis hin zu Zensurmaßnahmen zu achten.

4.1.1.5 User-Zentrierung

User-Zentrierung bedeutet, dass die Bedürfnisse des User in den Mittelpunkt gestellt werden. Diese Bedürfnisse gilt es von den übrigen Stakeholdern zu befriedigen, um das OSN für die User „attraktiv“ zu gestalten. Denn ohne die Nutzung der User fehlt dem OSN die Existenzberechtigung.

Die Gestaltung einer Webseite, sprich eines human-readable Web Services (Kapitel 2.2.3) wird mittels einer Informationsarchitektur gestaltet, welche folgende Bedeutungen hat:

1. „The structural design of shared information environments.
2. The combination of organization, labeling, search, and navigation systems within web sites and intranets.
3. The art and science of shaping information products and experiences to support usability and findability.
4. An emerging discipline and community of practice focused on bringing principles of design and architecture to the digital landscape.“ (Morville und Rosenfeld 2007, Seite 4)

Eine nach den Konzepten der Informationsarchitektur gestaltete Webseite erlaubt es demnach, die Bedürfnisse der User adäquat zu befriedigen. Daten (aus denen Informationen geschaffen werden) werden derart aufbereitet, dass die User sie bestmöglichst finden und nutzen können.

Die Informationsarchitektur lässt sich in die zu entwickelnde Architektur integrieren und deren Konzepte wiederverwenden, um den User in den Mittelpunkt zu stellen. Dieses Konzept der User-Zentrierung ist, aus ökonomischer Perspektive betrachtet, existentiell für das OSN: Das Social Web „lebt“ vom Mitmachen der User (Kapitel 2.1.1). Die Nutzung muss daher möglichst einfach sein und der User soll bei seinen Aktivitäten unterstützt werden. Um auf die Features des OSN zugreifen zu können, muss er nicht erst die Architektur verstehen. Das Konzept der Transparenz (Kapitel 4.1.1.3) erlaubt dem User zwar einen tieferen Einblick in die Strukturen der Architektur, doch möchte sich ein durchschnittlicher User stattdessen mit Freunden und Kollegen vernetzen. Daher ist aus Sicht der User eine Abstraktion von der Architektur nötig.

4.1.2 Ziel- und Interessengruppen

Die in Kapitel 4.1.1 gezeigten Konzepte erheben den Anspruch, den Stakeholdern des OSNs von Nutzen zu sein. Wie bei einer Produktentwicklung üblich, wird die Definition von Ziel- und Interessengruppen, die von dem OSN erreicht werden sollen, angestrebt. Jedoch

ergibt sich die Tatsache, dass eine klare Abgrenzung der anvisierten Gruppen schwierig bis unmöglich erscheint, hier sogar ungewollt ist.

Bei näherer Betrachtung des momentan größten OSNs Facebook, aber auch der VZ-Gruppe (Kapitel 3.1.2) wird deutlich, dass es eine Abkehr von bestimmten Ziel- und Interessengruppen gab: Waren es früher lediglich Studenten, so wird heute quasi niemand mehr ausgeschlossen – außer ein potenzieller User möchte nicht teilnehmen. Aber nicht nur private User-Gruppen werden angesprochen, sondern auch Organisationen mit (teilweise) kommerziellem Interesse, die eine Art Profil oder Themenseite einrichten oder sich als Werbepartner/externer Service-Provider anschließen können. Doch die organisatorische Hierarchie mit dem zentralen Provider des OSNs bleibt stets gewahrt, sodass sich jeder Stakeholder den Interessen des einen Providers unterstellen muss.

Das dezentral organisierte OSN führt den eben geschilderten Gedanken, alle potenziellen User einzubeziehen und keinen User auszuschließen, weiter fort. Zudem bezieht es das Konzept der Dezentralisierung ein und rückt von einer klaren Rollentrennung zwischen User und Provider ab. Somit erweitern sich die Ziel- und Interessengruppen um diejenigen, die in einem zentral organisiertem OSN die Provider darstellen.

Die Offenheit des OSNs (Kapitel 4.1.1.2) weicht die Definition der Ziel- und Interessengruppen weiter auf: „Special-Interest“-Gruppen, die beispielsweise in einem spezialisierten OSN wie LinkedIn oder XING (Kapitel 3.1.2) lediglich berufliche Kontakte knüpfen möchten, werden hier ebenfalls als potenzielle Teilnehmer angesprochen.

Weiterhin ergibt sich die Chance, zusätzliche (Rand-)Gruppen zu erreichen: Es gibt Gruppen, die dem Social Web und Social Networking kritisch gegenüber stehen, weil sie ähnliche Defizite sehen, wie in dieser Arbeit aufgezeigt (Kapitel 3.4), und eine realistische Einschätzung haben, welche Möglichkeiten bestehen, wie ihre Daten im Internet genutzt werden (Spiekermann und Cranor 2009, Seite 71–72). Angenommen, manche Personen dieser Gruppen würden gerne teilnehmen, wenn sie das OSN mitgestalten könnten. Die Mitgestaltung ist durch das Konzept der Offenheit möglich, sogar erwünscht. Diese Personengruppen würden die Chance erhalten, die Möglichkeiten des Social Networkings zu nutzen.

Folglich ergibt es bei einem dezentral organisiertem OSN weniger Sinn, klar definierte Ziel- und Interessengruppen zu bestimmen, als zum Entschluss zu kommen: Jeder Stakeholder – egal welcher Gruppe er angehört – hat dieselben Chancen, am OSN teilzunehmen.

4.1.3 Features

Der aus der Software-Entwicklung bekannte Begriff „Feature“ beschreibt Funktionen, die eine Software abdeckt. Daher wird bei Social Software, die dieses OSN darstellt und durch

die Architektur in einen „Rahmen“ gefasst wird, ebenfalls von Features gesprochen. In den letzten Jahren haben sich im Bereich des Social Networkings verschiedene Features etabliert, neue kommen ständig hinzu. Einen Überblick, welche Features dezentral organisierte OSNs anbieten können (und müssen, um konkurrenzfähig mit zentral organisierten OSNs zu sein und um die anvisierten Ziel- und Interessengruppen zu erreichen), bietet dieses Kapitel.

Die Features leiten sich aus den Aussagen in Kapitel 3.3, insbesondere Kapitel 3.3.2 über die Anforderungen, Kapitel 4.1.1, insbesondere Kapitel 4.1.1.5 über die Konzepte ab. Sie basieren zudem auf den in Kapitel 2.1.1 zitierten Arten von Social Software und Grundfunktionen von OSNs.

Das hier zu entwickelnde OSN ist in der Gestaltung, Adaption und Erweiterung von Features offen konzipiert (Kapitel 4.1.1.2). Es erweitert die Grundfunktionalität eines OSN um beliebig viele zusätzliche Features, die sich in dieser Arbeit nicht alle abbilden lassen. Durch die Dynamik des Marktes werden immerzu neuartige Features im Social Web erschaffen, deren Konzepte unter Umständen jedoch wieder verworfen werden. Offline-Applikationen, die auf einem lokalen Computer ausgeführt werden, zum Beispiel Personal Information Manager (PIM) oder Office-Applikationen, werden ins Internet/Web verlagert und als SaaS (Kapitel 2.2.2) angeboten. Services im Web werden in Mashups zusammengefasst oder in andere Services integriert.

4.2 Architektur

Die Idee und die dazugehörigen Konzepte hinter dem dezentral organisierten OSN (Kapitel 4.1) bilden den Rahmen für dessen Architektur. Das Design der Architektur beruht dabei auf verschiedenen Paradigmen. Es wird beschrieben, wie Standards und Technologien aus den Bereichen Semantic Web, Decentralized/Distributed Computing unter Verwendung von FLOSS-Lizenzen miteinander harmonieren können. Das Resultat sind vier Kernkomponenten der Architektur.

4.2.1 Paradigmen

Die Paradigmen der Architektur werden nachfolgend beschrieben. Die Architektur besteht aus einem Netz von Komponenten: Services und Agents, die über Schnittstellen miteinander kommunizieren und auf die von außen zugegriffen werden kann. Die Komponenten sind modularisiert aufgebaut. Die innerhalb der Komponenten verarbeiteten und gespeicherten Daten sowie die Verbindungen zwischen den Komponenten werden mittels verschiedener Techniken verschlüsselt und signiert.

4.2.1.1 Ressourcen-orientierte Architektur mit REST

Die Architekturkomponenten erweitern ein Schnittstellenmodell. Da es sich um eine Architektur im Social Web handelt, sind die meisten Schnittstellen mittels Web Services umgesetzt. Die Web Services basieren auf den Prinzipien der REST-Architektur und sind Ressourcen-orientiert.

In einem dezentral organisierten OSN existieren zahlreiche Verbindungen zwischen den Komponenten über Schnittstellen, sodass ein Graph aus Datenflüssen entsteht, wenn zum Beispiel mehrere User Nachrichten miteinander austauschen. Um welche Daten es sich dabei handelt, ist in Tabelle 4.1 abzulesen. In der Tabelle werden verschiedene Kategorien an Daten aufgelistet, die sich an den Features aus Kapitel 4.1.3 orientieren. Ergänzt wird die Tabelle durch verschiedene Typen von Daten[1] und einer Auswahl an (überwiegend) offenen/freien Standards, Dateiformaten und Protokollen, die für die jeweiligen Typen geeignet scheinen.

Auf Basis der Daten werden Web Services (Kapitel 2.2.3) konstruiert. Die Web Services basieren auf der REST-Architektur und werden daher als RESTful bezeichnet. Durch REST lassen sich die Web Services derart gestalten, dass andere verwendete Technologien und Standards möglichst einfach integriert und wiederverwendet werden können, zum Beispiel aus dem Bereich des Semantic Webs (Kapitel 2.1.2). REST ermöglicht es außerdem, Webseiten wie Web Services zu nutzen und umgekehrt (Richardson und Ruby 2007, Seite 18). Auf diese Weise genügt *eine* Architektur für das Web, um Inhalte für Menschen (human-readable) und für Maschinen (machine-readable) aufzubereiten. Diese Vereinfachung des Architekturdesigns begünstigt, dass (externe) Provider und Entwickler mit geringem Aufwand auf die Web Services des OSNs zugreifen können.

Die folgenden Schritte lehnen sich an die von Richardson und Ruby (2007, Seite 216) an. Aus den Daten, die von den Komponenten der Architektur verarbeitet werden, wird ein Datenmodell entwickelt. Zur einfachen Veranschaulichung kann jede objektorientierte Programmiersprache herangezogen werden: Das Modell enthält Objekte, die die jeweiligen Daten repräsentieren. Woher diese Daten stammen, ist an dieser Stelle erst einmal unwichtig. Die Attribute und Methoden gleichartiger Objekte werden zu Klassen zusammengefasst. Für den Web Service werden aus den Klassen und Objekten schließlich Ressourcen abgeleitet. Jede Ressource erhält eine URI zur eindeutigen Identifizierung, woraus später bei einem produktiven System eine URL würde, die der Lokalisierung der jeweiligen Ressource im Internet dient. Somit sind die Web Services nicht nur RESTful, sondern auch Ressourcen-orientiert.

Zum Austausch von Ressourcen werden bei Anfragen (Request) und Antworten (Response) nicht die Ressourcen, sondern Repräsentationen von Ressourcen verwendet, die zum Serialisieren von Objekten (und somit von Daten) genutzt werden. Gängige (Datei-)Formate

[1] Der Ausdruck ist nicht zu verwechseln mit dem Begriff Datentyp aus der Programmierung.

Kategorie	**Typen**	**Datenformat** (Auswahl an Standards, Dateiformaten, Protokollen)
Dokumente	Texte, Tabellen, Präsentationen, Archive, Datenbanken	Binär- mit Metadaten oder Textdaten: txt, PDF, ODF, OpenXML, EPUB, ZIP, Tar, SQL
Multimedia (Dateien)	Audio-, Video-, Grafik-Dateien	Binärdaten mit Metadaten oder Textdaten: Ogg Vorbis, Flac, MP3, WebM, Ogg Theora, H.264, Matroska, JPG. PNG, TIFF, GIF, WebP, SVG
Multimedia (Meta)	Metadaten, Playlists	Textdaten: EXIF, XMP, IDv3, XSPF, M3U, PLS
Streams	Audio, Video, Grafik, Text	Binärdaten mit Metadaten oder Textdaten: RTP/RTSP, DCCP, WebM, Ogg Vorbis, MP3, H.264, SMIL
Nachrichten	E-Mails, IM, Blog Posts und Kommentare, Microblogging Posts, Feeds, SMS, Status	Textdaten: Maildir++, XMPP/Jabber, ATOM, RSS, OPML
PIM	Adressbücher, Kalenderdaten, Lesezeichen	Textdaten: vCard, iCalendar, XBEL
Sprache	Telefonie, Dikate	Binärdaten: VoIP/SIP, Speex, CELT, G.711, G.729
Location Based Services	Wegpunkte, Routen, Tracks	Textdaten: GPX, GML, KML NMEA
Konfigurationen	Einstellungen für genutzte Clients, Portals, Communities, Services	Textdaten
Erweiterungen	Apps, Widgets, Add-ons, Plug-ins	Binärdaten oder Textdaten
Profile	Identitäten: privat und öffentlich, privat und beruflich, Anonyme, Pseudonyme, Portal-, Community-, Service-spezifische	Textdaten
Account	Passwörter (Hashes mit Salts), Signaturen, Schlüssel, IDs, URIs	Textdaten: SHA1, OpenPGP, X.509, OpenID

Tabelle 4.1 Die Daten, die im OSN verwaltet werden, lassen sich kategorisieren.

für die Repräsentationen enthält Tabelle 4.2. Sie lassen sich nach ihrem Internet Media Type (den Content-Type im HTTP-Header) oder ihrer Dateierweiterung unterscheiden. Zudem eignen sie sich für unterschiedliche Zwecke, denn sie können human-readable oder machine-readable oder für Mensch *und* Maschine aufbereitet sein.

4.2.1.2 Modularisierung, Integration und Wiederverwendung

Die Komponenten der Architektur sind modular aufgebaut. Auf höchster Gestaltungsebene existieren vier Kernkomponenten mit jeweiligen Unterkomponenten. Neben Web Services werden weitere Services und Agents in die Architektur integriert, um dem Konzept der Offenheit (Kapitel 4.1.1.2) zu genügen.

Bei der Entwicklung der Architektur gilt, das Prinzip der Modularisierung zu verwenden (Raymond 2003, Kapitel 4). Für jedes Feature (Kapitel 4.1.3) werden spezialisierte Kom-

Format Name	Internet Media Type	Dateierweiterung	HTTP Request human-readable	HTTP Request machine-readable	HTTP Response human-readable	HTTP Response machine-readable
Klartext	text/plain	.txt	—	—	—	✓
URL Encoding	application/x-www-form-urlencoded	nicht verfügbar	—	✓	—	✓
XHTML, HTML5	application/xhtml+xml	.xhtml, .html	✓	✓	✓	✓
XML (zum Beispiel RSS)	diverse (application/rss+xml)	diverse (.rss, .xml)	—	✓	—	✓
JSON	application/json	.json	—	✓	—	✓
Spezialformat (zum Beispiel PNG)	diverse (image/png)	diverse (.png)	vom Format abhängig	✓	vom Format abhängig	✓

Tabelle 4.2 Formate bei HTTP-Anfragen (Request) und -Antworten (Response).

ponenten entwickelt. Komponenten erweitern die Architektur um Daten und eine Logik bestehend aus Klassen und Objekten mit Attributen sowie Methoden (Kapitel 4.2.1.1). Schnittstellen erlauben den Akteuren (Menschen und Maschinen), auf die Komponenten zuzugreifen, und können machine-readable (API) oder human-readable (User Interface (UI)) sein. Komponenten können eigene Schnittstellen besitzen, aber auch bestehende erweitern. Zwingend notwendig ist eine Anbindung als Web Service, damit zumindest *eine* einheitliche Schnittstelle entsteht.

Mehrere Komponenten lassen sich auf einer höheren Abstraktionsebene zu Kernkomponenten zusammenfassen, um gleichartige Komponenten zu vereinen. Die Anzahl der Instanzen an (Kern-)Komponenten ist beliebig. Jedoch muss eine Komponente – wie Ressourcen auch – eindeutig zu identifizieren sein.

Später in Kapitel 4.2.2 werden die jeweiligen Kernkomponenten näher definiert und erläutert. Zunächst werden vier Typen von Kernkomponenten unterschieden:

- Portale,
- Communities,
- weitere Services und
- Clients

Bild 4.2 zeigt jeweils eine Instanz der vier Kernkomponenten im Zusammenhang. Clients verbinden sich zu Portalen, die wiederum mit Communities und weiteren Services interagieren. Die Gründe für die Wahl der Kernkomponenten und deren Zusammenhang werden ebenfalls später erläutert. In diesem Kapitel wird lediglich die Modularisierung beschrie-

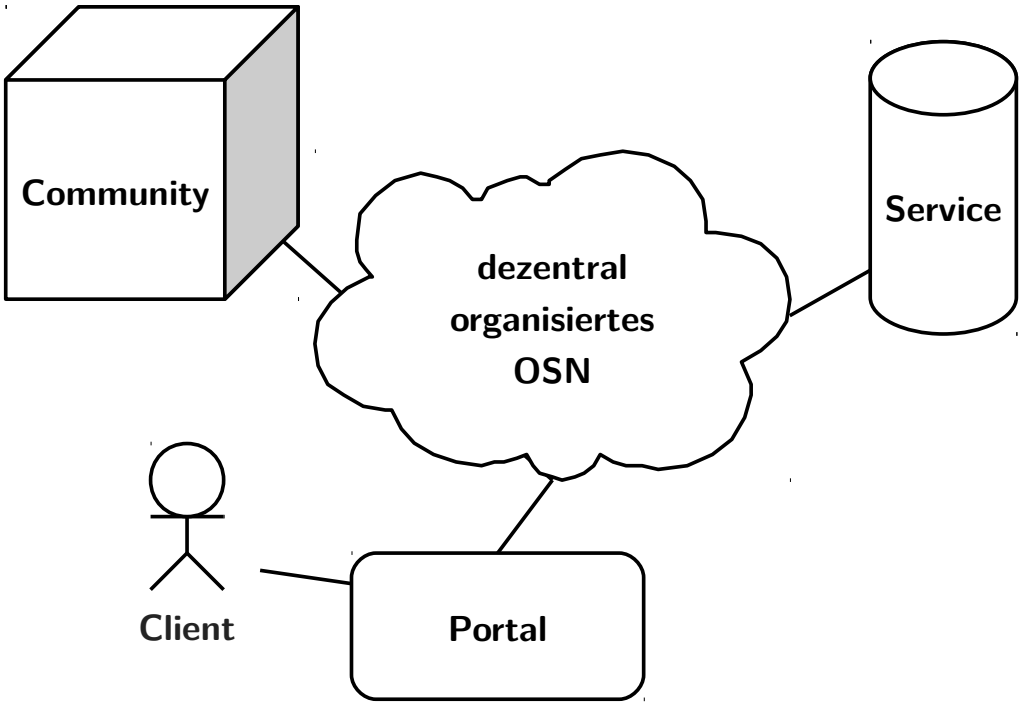


Bild 4.2 Das dezentral organisierte OSNs besteht aus vier Kernkomponenten.

ben, sprich wie Komponenten aufgebaut sind, wie sie sich integrieren und wie sich ihre Methoden und Daten wiederverwenden lassen.

Bild 4.3 fasst die zuvor beschriebe Modularisierung noch einmal zusammen. Zudem führt sie weitere Begriffe ein: Add-on, Service und Agent.

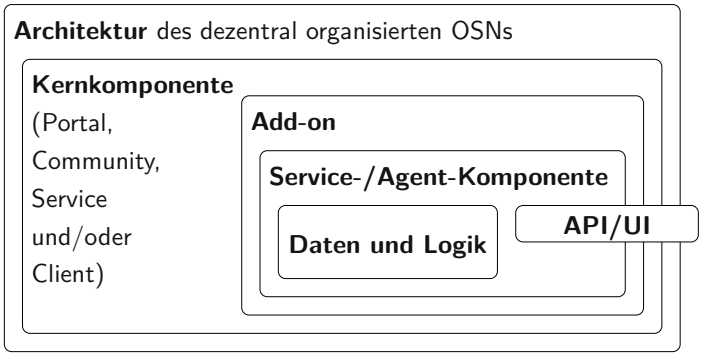


Bild 4.3 Die Architektur ist modular aufgebaut.

Die in eine Kernkomponente integrierte Komponente wird in dieser Arbeit als Add-on bezeichnet. Ein Add-on kann entweder vom Typ Service oder vom Typ Agent sein (Kapitel 2.2.4, vergleiche später Kapitel 4.2.1.3). Code 4.1 zeigt den generischen Aufbau der Schnittstelle für einen Web Service, nach dem sich alle Komponenten zu richten haben. Das Konzept lehnt sich an Richardson und Ruby (2007, Seite 227–228) an.

```
1 http://{example.net}/[{add-on}[/{object}[/{class}[/{object}]]]]
          |             | |          |          |         |
3         |             | |          |          |         untergeordnetes
          |             | |          |          |         Ressourcenobjekt
5         |             | |          |          untergeordnete
          |             | |          |          Ressourcenklasse
7         |             | |          Add-on-Ressourcenobjekt
          |             | Add-on-Ressourcenklasse
9         |             Web-Root-Verzeichnis
          Host
```

Code 4.1 Die Web-Service-Schnittstelle wird durch den Aufruf einer URL (hier generisch dargestellt) via HTTP angesprochen.

Die Formel veranschaulicht einen Web Service, der RESTful und Ressourcen-orientiert ist. Gezeigt wird eine vollständige URL mit HTTP als Protokoll, die vom Aufbau der Code 2.1 aus Kapitel 2.1.1 ähnelt. Platzhalter sind in geschweiften Klammern enthalten, optionale Angaben in eckigen Klammern. Hierarchien von Ressourcen werden durch den Pfad angegeben. Hinter dem Web-Root-Verzeichnis /, also dem Hauptverzeichnis des Web Services können optional die Ressourcen folgen: Zuerst wird ein bestimmtes Add-on referenziert, bestehend aus der jeweiligen Ressourcenklasse `{add-on}`, um die die Schnittstelle erweitert wird, und dem dazugehörigen Ressourcenobjekt `{object}`, das eine Instanz der Klasse ist. Abhängig davon, ob das Add-on weitere Ressourcen einbringt, folgt nun eine Hierarchie weiterer Ressourcen, die abermals in Klassen (`{class}`) und Objekten (`{object}`) untergliedert sind. Auf diese Weise werden Add-on-Hierarchien möglich, das heißt, Add-ons können andere Add-ons beinhalten. Die Konzepte Vererbung, Generalisierung und Spezialisierung aus der objektorientierten Programmierung werden dadurch ebenfalls unterstützt.

Add-on	**Resource's URI**	**HTTP Method** **GET**	**POST**	**PUT** to a new resource	**PUT** to an existing resource	**DELETE**
`{Add-on}`	`{add-on}`	List add-on's objects.	Create an add-on's object.	—	—	—
	`/{add-on}/{object}`	List object's sub classes.	—	Create this object.	Modify this object.	Delete this object.
	`/{add-on}/{object}/{class}`	List class's objects.	Create a new object.	—	—	—
	`/{add-on}/{object}/{class}/{object}`	Get a representation of this object.	—	Create this object.	Modify this object.	Delete this object.
	`/{add-on}/{object}/{class}/{object}/{sub class}/{object}`	Get a representation of this object.	—	Create this sub class's object.	Modify this sub class's object.	Delete this sub class's object.

Tabelle 4.3 Die allgemeine, für alle Kernkomponenten gültige Web-Service-Schnittstelle.

Die Kernkomponenten greifen oftmals auf dieselben Daten mit denselben Methoden zu. Daher erscheint es sinnvoll, allgemeine Add-ons zu entwickeln, die allen Kernkomponenten zur Verfügung stehen. Der generische Aufbau aus Code 4.1 wird daher in Tabelle 4.3

erweitert und dient als Vorlage für die Entwicklung weiterer Add-ons. Jede Zeile umfasst neben dem Namen des Add-ons die URIs der dazugehörigen Ressourcen und die für einen RESTful Web Service relevanten HTTP-Methoden zum Lesen/Auflisten, Erstellen, Modifizieren und Löschen von Daten. Aufbau und Syntax dieser und der darauf aufbauenden Tabellen orientieren sich an Richardson und Ruby (2007, Seite 53)[2].

Add-on	**Resource's URI**	**HTTP Method GET**	**POST**	**PUT** to a new resource	**PUT** to an existing resource	**DELETE**
Home Page	/	—	—	—	—	—
Tags	/tags	List tags.	Create a new tag.	—	—	—
	/tags/{tag}	List resources tagged with this tag.	—	Create this tag.	Modify this tag.	Delete this tag.
Tag Clusters	/tagclusters	List tag clusters.	Create a new tag cluster.	—	—	—
	/tagclusters/{tag cluster}	Get tag cluster with tags and tagged resources.	—	Create this tag cluster.	Modify this tag cluster.	Delete this tag cluster.
Categories	/categories	List categories.	Create a new category.	—	—	—
	/categories/{category}	List resources filed in this category.	—	Create this category.	Modify this category.	Delete this category.
Aggregation/ Syndication	/feeds	List feeds.	Create a new feed.	—	—	—
	/feeds/{feed}.{format}	Get this feed in that format.	—	Create this feed.	Modify this feed.	Delete this feed.
Sitemaps	/sitemaps	List sitemaps.	Create a new sitemap.	—	—	—
	/sitemaps/{sitemap}[.xml|.gz]	Get this sitemap in that format.	—	Create this sitemap.	Modify this sitemap.	Delete this sitemap.
Website's Themes	/themes	List themes.	Create a new theme.	—	—	—
	/themes/{theme}	Get some information about this theme.	—	Create this theme.	Modify this theme.	Delete this theme.
	/themes/{theme}/styles	List CSS files.	Create a new CSS file.	—	—	—
	/themes/{theme}/styles/{style}.css	Get this CSS file.	—	Create this CSS file.	Modify this CSS file.	Delete this CSS file.
	/themes/{theme}/images	List image files.	Create a new image file.	—	—	—
	/themes/{theme}/images/{image}.{format}	Get this image file.	—	Create this image file.	Modify this image file.	Delete this image file.

Tabelle 4.4 Die allgemeinen Add-ons bilden die Basis der Web-Service-Schnittstelle.

Die allgemeinen Add-ons bilden die Basis der Web-Service-Schnittstelle und werden in Tabelle 4.4 erläutert. Sie umfassen Ressourcen Tags, Tag Cluster, Kategorien, Feeds und Sitemaps, mit denen sich Inhalte strukturieren lassen. Für die visuelle Gestaltung der Web Services, die gleichzeitig auch als Webseiten dienen, kann ein Add-on für Themes eingesetzt werden.

Damit die Administration der Komponenten via Web möglich wird, sind in Tabelle 4.5 weitere Add-ons aufgelistet. Mit ihrer Hilfe lassen sich die Kern-, Service- und Agents-Komponenten verwalten. Wie in Kapitel 2.2.3 bereits erwähnt, existiert eine Beschreibungssprache für RESTful Web Services (WADL), welche als eigenständige Datei ebenfalls in die Web Services als Ressource eingebettet werden kann. Mittels dieser Ressource

[2] Als Fachsprache wird für die Tabellen Englisch gewählt.

können externe Akteure automatisiert ermitteln, welche Funktionalität der jeweilige Web Service umfasst. Ein Service Directory, das wie bei SOAP WSDL-Daten sammelt und zum Abruf bereitstellt, lässt sich für WADL-Daten ebenfalls realisieren – allerdings als selbst entwickeltes Add-on.

Add-on	**Resource's URI**	**HTTP Method GET**	**POST**	**PUT** to a new resource	**PUT** to an existing resource	**DELETE**
Core Components	`/admin`	List core components.	Create a new core component.	—	—	—
	`/admin/{core component}`	List core component's objects.	—	Create this core component.	Modify this core component.	Delete this core component.
Add-ons	`/admin/{core component}/add-ons`	List core component's add-ons.	Create a new add-on.	—	—	—
	`/admin/{core component}/add-ons/add-on`	Get information of this add-on.	—	Create this add-on.	Modify this add-on.	Delete this add-on.
About	`/admin/{core component}/about`	List some information about this core component.	Create a new information object.	—	—	—
	`/admin/{core component}/about/wadl.xml`	Get WADL file in XML format.	—	Create WADL file with a XML syntax.	Modify WADL file with a XML syntax.	Delete WADL file.
Packages	`/admin/add-ons/packages`	List installed packages.	Install a new package.	—	—	—
	`/admin/add-ons/packages/{core component|add-on}`	Get information about installed package.	—	Install this package.	Update/upgrade this package.	Remove this package.

Tabelle 4.5 Add-ons zur Administration der Komponenten erweitern die Web-Service-Schnittstelle.

Durch ein standardisiertes Integrationsmodell können neben Web Services weitere Technologien und Standards in die Architektur integriert und über eine Schnittstelle angeboten werden. Es erscheint vor allem dann sinnvoll, wenn Technologien und Standards mit den Konzepten und Paradigmen des OSNs harmonieren und eine weite Verbreitung im Internet haben. Die Datenkategorisierung in Tabelle 4.1 bildet die Grundlage für einige Beispiele: E-Mail und IM in der Kategorie Nachrichten (im Sinne von Messages) und Voice over IP (VoIP) in der Kategorie Sprache. Für alle drei Technologien existieren bereits funktionierende Infrastrukturen, die auf einer Vielzahl von offenen und/oder freien Standards, Protokollen, Formaten et cetera fußen und dezentral organisiert sind. E-Mail gilt im deutschsprachigen Raum immer noch als die populärste „Anwendung“ im Internet (van Eimeren und Frees 2010, Seite 338), das IM-Protokoll Extensible Messaging and Presence Protocol (XMPP) (auch bekannt als Jabber) verwendet XML als Format (XMPP Standards Foundation 2010) und VoIP steht in Konkurrenz zur klassischen Festnetztelefonie.

Eine beispielhafte Umsetzung wird in Bild 4.4 gezeigt. Die Komponenten bestehen in diesem Beispiel aus verschiedenen Agent Add-ons, die neben einer Web-Service-Schnittstelle weitere Schnittstellen auf Basis gängiger E-Mail-Protokolle integriert haben. Das Service Add-on Mail kann E-Mails senden, empfangen, lesen, löschen und dergleichen, indem auf den im Hintergrund agierenden Mail User Agent (MUA) zugegriffen wird. Der MUA koordiniert die Kommunikation zwischen dem Service Add-on und den anderen Agents. Eine zu sendende E-Mail wird an den Message Submission Agent (MSA) weitergeleitet, der sie an den Mail Transfer Agent (MTA) übergibt. Der MTA ist für die externe Kommunikation

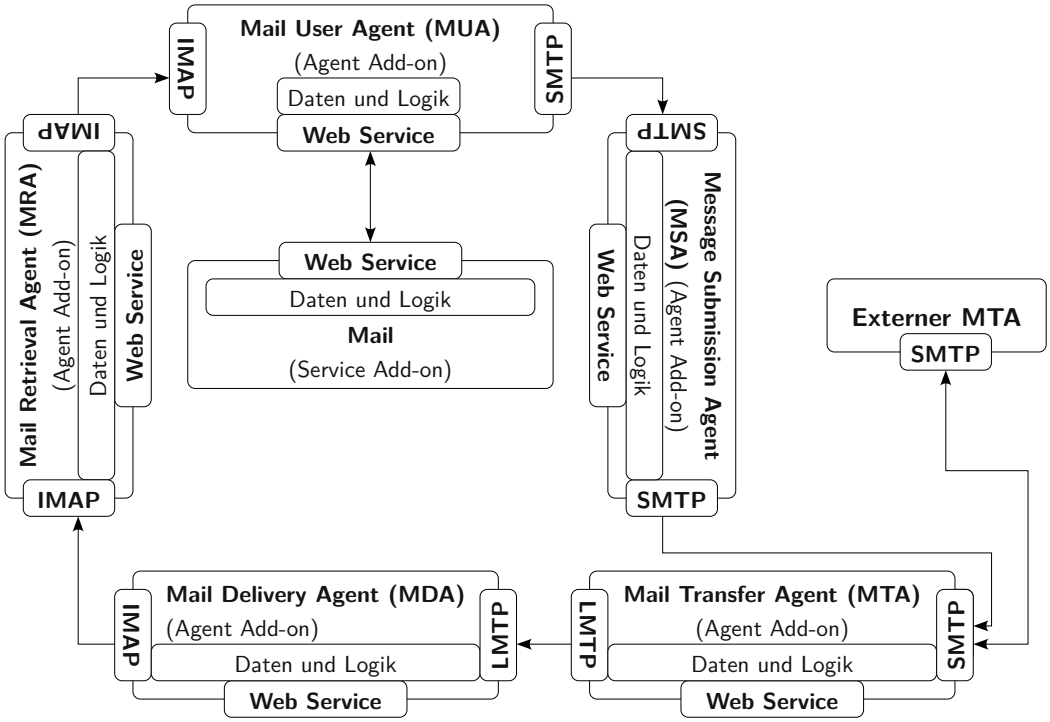


Bild 4.4 Verschiedene Service und Agent Add-ons beteiligen sich am Senden und Empfangen von E-Mail-Nachrichten.

zuständig und kann mit einem externen MTA, der nicht zur selben Kernkomponente oder gar nicht zum OSN gehört, E-Mails austauschen. Empfängt der MTA eine E-Mail, wird sie an den Mail Delivery Agent (MDA) übergeben. Der Mail Retrieval Agent (MRA) greift auf den MDA zu, um E-Mails für den MUA zwischenzuspeichern. So gelangt die empfangene E-Mail bei Bedarf zum Service Add-on Mail. Das Beispiel zeigt, wie Services und Agents miteinander kommunizieren können. Jedes Add-on hat eine spezielle Funktion und greift über Schnittstellen auf die Funktionen anderer Add-ons zu. Das Service Add-on Mail ist im human-readable Web nichts anderes als eine Webmail-Applikation.

Zusammengefasst betrachtet: Es bietet einige Vorteile, Technologien wie die eben vorgestellten wiederzuverwenden und in die Architektur als Services zu integrieren, anstatt konkurrierende Services zu entwickeln. Darüber hinaus steht es jedem Stakeholder frei, alternative Technologien zu entwickeln und diese neben den anderen innerhalb des OSNs zu etablieren (Konzept der Offenheit, Kapitel 4.1.1.2).

Auch im Bereich der Web Services finden sich viele Beispiele dafür, externe Ressourcen für die eigene Architektur zur Verfügung zu stellen. Beispielsweise können externe Mel-

dungen von Nachrichtenagenturen aggregiert werden und so dem User eine umfassende Kommunikationszentrale zur Verfügung stellen (Goddard und Byrne 2010, Seite 3).

4.2.1.3 Selbstorganisation

Die Architektur verfügt über eine selbstorganisierende Infrastruktur, bestehend aus Agents und P2P-Systemen. Sie agieren im Hintergrund, damit Daten permanent zwischen den Einzelkomponenten der Architektur ausgetauscht werden können. Auf diese Weise entsteht ein Netzwerk aus Peers, das Daten innerhalb und außerhalb des OSNs dort zur Verfügung stellt, wo sie abgefragt werden.

Ein großer Vorteil zentral organisierter OSN-Architekturen ist, dass Daten innerhalb eines OSNs „kürzere Wege“ durchlaufen oder die „Weglängen“ tendenziell vorhersehbar sind. Die Vernetzung zwischen zwei Usern ist – verkürzt formuliert – lediglich eine Transaktion in einem Datenbanksystem. Bei der hier zu entwickelnden Architektur hingegen sind die „Datenwege“ wesentlich länger beziehungsweise lässt sich keine exakte Aussage über die „Länge“ treffen. Ein User, der auf einer Kernkomponente aktiv ist, möchte sich mit einem anderen User, der auf einer anderen Kernkomponente aktiv ist, vernetzen. Die Vernetzung muss folglich in beiden Kernkomponenten vermerkt werden. Komplizierter wird es, wenn eine größere Anzahl an Usern viele Daten miteinander möglichst gleichzeitig austauschen möchte. Bevor ein Datenaustausch stattfindet, tauchen zunächst viele Fragen auf, unter anderem: Welche Daten sollen von welchem User an welchen User geschickt werden? Auf welcher Kernkomponente ist der jeweilige User aktiv? Nach einer erfolgreichen Recherche werden dann schließlich die Daten ausgetauscht, müssen aber noch verarbeitet werden, bevor ein User sie aufbereitet erhalten kann und so weiter. Die Beispiele zeigen, dass ein dezentral organisiertes OSN gegenüber zentral organisierten ein massives Problem aufweist: Der Datentransfer erfolgt zu langsam. Zwar gewinnt die Architektur wie bei einem P2P-System mit jeder hinzugefügten Kernkomponente, die über ausreichende Hardware-Ressourcen verfügt, an Verfügbarkeit und Performance, doch sind die Datenflüsse per se komplexer gestaltet und benötigen mehr Ressourcen. Die Ressource Zeit ist für User allerdings entscheidend: Verzögert sich die Übermittlung einer Nachricht signifikant gegenüber einer Übermittlung innerhalb eines zentral organisierten OSNs, werden die User eher Abstand von Dezentralisierung nehmen. Anhand von Bild 4.5 kann das Problem nachvollzogen werden.

Um dieses Problem zu minimieren, benötigt die Architektur Mechanismen zur Selbstorganisation. In Kapitel 4.2.1.2 wurde gezeigt, dass neben den Services die Agents eine wesentliche Rolle im dezentral organisierten OSN haben. Agents sind im Hintergrund agierende Komponenten, die aktiv und abgeschieden vom human-readable Web Daten verarbeiten und den Akteuren zuarbeiten. Als Beispiel dient Bild 4.4 aus Kapitel 4.2.1.2: Damit ein Akteur Nachrichten (dort in Form von E-Mails) schreiben und senden oder empfangen und lesen kann, interagieren im Hintergrund eine Reihe von spezialisierten Agents. Sie sind in

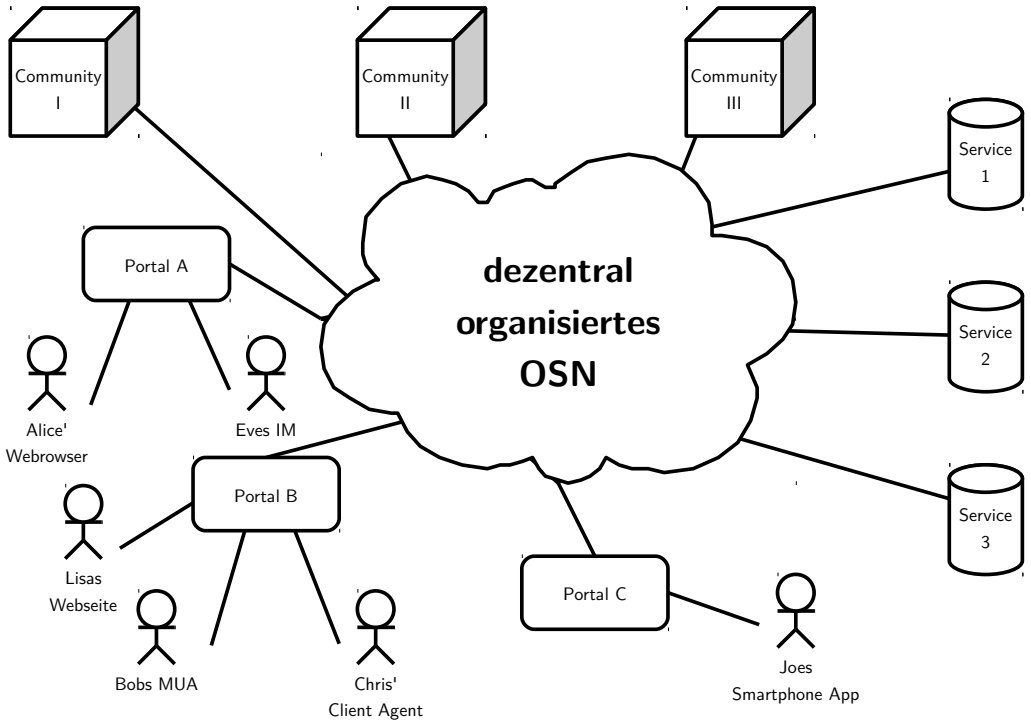


Bild 4.5 Das OSN wächst mit jeder hinzugefügten Kernkomponente und wird dadurch komplexer.

der Lage, permanent neue E-Mails zu empfangen oder bei hoher Auslastung Mails zeitversetzt zu versenden – auch wenn der Akteur zur selben Zeit nicht das Service-Add-on anspricht.

Über eigene Web-Service-Schnittstellen können Agents-Add-ons veranlasst werden, Daten anzunehmen, zu verarbeiten und/oder zu beschaffen. In Tabelle 4.6 wird die dafür benötigte Web-Service-Schnittstelle dargestellt. Unterstützt wird eine Typisierung von Agents (vergleiche später Kapitel 4.2.2) und eine Aufgabenliste, damit Aufgaben im Hintergrund von den Agents bearbeitet werden können (vergleiche ein ähnliches Konzept bei Richardson und Ruby (2007, Seite 228–230)).

Wie in Kapitel 2.2.4 angedeutet, können mit Hilfe des Semantic Webs Ontologien eingesetzt werden, um sich zu „verständigen“ und ihre Funktionen effizienter zu erledigen. Gümüs et al. (2007) haben auf Basis vorhandener Standards ein System aus mehreren Agents entwickelt, das ebendiese Möglichkeiten des Semantic Webs aufgreift. Ihre Konzepte lassen sich in die hier zu entwickelnde Architektur integrieren, indem aus den von ihnen beschriebenen Agents und Services neue Add-ons entwickelt werden, die andere Add-ons erweitern und unterstützen.

Add-on	Resource's URI	HTTP Method GET	POST	PUT to a new resource	PUT to an existing resource	DELETE
Types	/agents	List supported agents types.	Create a new agent type.	—	—	—
	/agents/{agent type}	Get some information about this agent type.	—	Create this agent type.	Modify this agent type.	Delete this agent type.
Jobs	/agents/{agent type}/jobs	List job queue.	Create a new job.	—	—	—
	/agents/{agent type}/jobs/{job}	Get this job from queue.	—	Create this job.	Modify this job.	Delete this job.
Agents	/agents/{agent type}/agents	List agents.	Create a new agent.	—	—	—
	/agents/{agent type}/agents/{agent}	Get some information about this agent.	—	Create this agent.	Modify this agent.	Delete this agent.

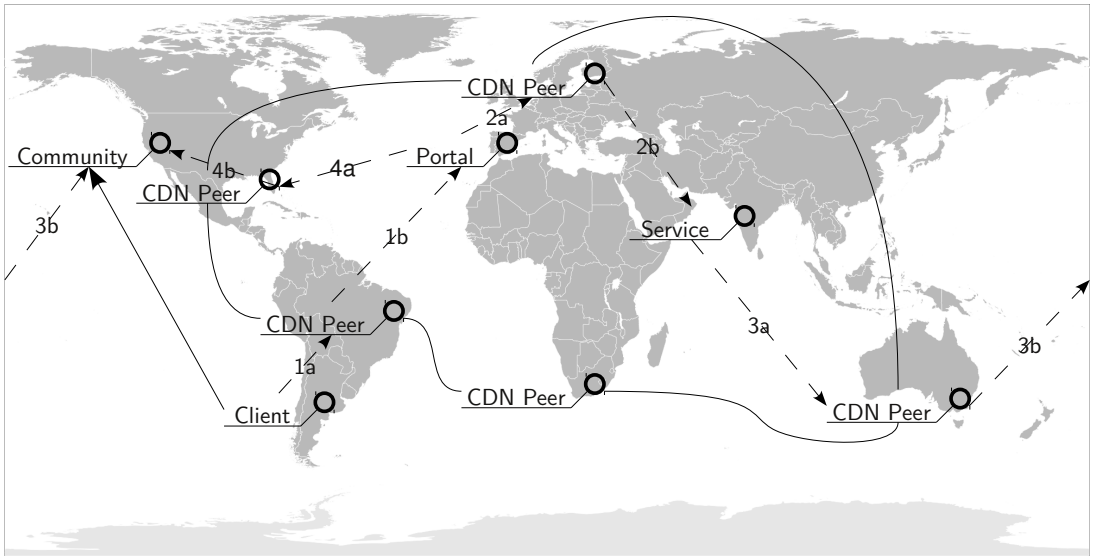
Tabelle 4.6 Die Agent-Add-ons erweitern die Web-Service-Schnittstelle.

Nicht nur Agents tragen zur Selbstorganisation bei, sondern auch weitere Systeme aus dem Bereich Decentralized/Distributed Computing (Kapitel 2.2). Sie helfen, große Datenmengen mit möglichst niedriger Latenz, aber hoher Bandbreite zu verarbeiten (womit unter anderem auch das Speichern von Daten gemeint ist). Die Anwendungsszenarien sind durch die Offenheit der Architektur vielfältig, daher soll für diese Arbeit ein einfaches Beispiel dienen, welches in Bild 4.6 dargestellt wird. Greifen zu viele Akteure auf dieselben Kernkomponenten zu, droht die Überlastung. Sind Akteure und Kernkomponenten geologisch weit entfernt, müssen die auszutauschenden Daten über weite Strecken transportiert werden, was zum Beispiel zu Verzögerungen führen kann. Ein CDN unterstützt die Kernkomponenten dabei, die Anfragen von Akteuren effizienter zu verarbeiten. Das Konzept stammt von Freedman (2010), das sich auf die hier zu entwickelnde Architektur übertragen lässt. Die Web Services verwenden das HTTP-Protokoll und lassen sich prinzipiell mit gängigen Techniken im CDN zwischenspeichern (Caching) und weiterverteilen. Somit werden die Kernkomponenten entlastet (Bandbreite steigt) und Daten werden durch verkürzte Wege schneller transportiert (Latenz sinkt). Das CDN kann proaktiv Daten zwischenspeichern und weiterverteilen, bevor Probleme durch zu viele oder zu weit entfernte Akteure auftreten. Jede neue Installation im CDN wird ein Peer und damit Teil eines sich selbst organisierenden P2P-Systems.

4.2.1.4 Access Controll und Rights Management

Die von Spiekermann und Cranor (2009, Seite 68) identifizierten Bereiche Access Control und Rights Management werden in die Architektur integriert. Sie haben im OSN die Aufgabe, die Sicherheitskonzepte aus Kapitel 4.1.1.4 umzusetzen.

Aufgrund des sparsamen Umgangs mit Daten und der konsequenten Vermeidung von Datenansammlungen, die keinen Beitrag zur Funktionalität eines Add-ons leisten, verlangen nur einige Add-ons die Authentifizierung eines Akteurs, um ihn für Zugriffe zu autorisieren (Privacy-by-Architecture, Kapitel 4.1.1.4). Zur Teilnahme am OSN benötigt jeder User



Ein Akteur (zum Beispiel ein User) benutzt die Client-Kernkomponente, um auf Daten zuzugreifen, von denen er nicht weiß, dass sie in der Community-Kernkomponente zu finden sind. Da alle Kernkomponenten (ausgenommen der Client) zur Zeit der Anfrage überlastet sind, wird ein CDN eingesetzt, das die Antworten der Kernkomponenten zwischenspeichert, um weitere Anfragen selbstständig beantworten zu können.
1a–1b: Der Client schickt eine Anfrage an das Portal, mit dem er verbunden ist.
2a–2b: Das Portal fragt beim Service an, wo es die gewünschten Daten finden kann.
3a–3b: Der Service leitet die Anfrage an die Community weiter – mit Erfolg – und antwortet dem Portal mit den Daten als Inhalt.
4a–4b: Das Portal kann nun dem Client eine Antwort mit den Daten als Inhalt senden.
Aus Sicht des Akteurs existiert eine direkte, wenn auch verzögerte Verbindung zwischen Client und Community.

Bild 4.6 Das P2P-CDN unterstützt die Kernkomponenten beim Datenaustausch.

einen Account, durch den er allerdings nicht automatisch identifiziert werden kann (vergleiche später Kapitel 4.2.2.1). Ein Account pro User ist ausreichend, um die Features innerhalb des OSNs nutzen zu können. Möglich wird dies durch eine Vertrauensbildung auf Basis einer Public-Key-Infrastruktur (PKI). Alle Komponenten und Akteure erhalten Zertifikate, mit denen sich Daten verschlüsseln und signieren lassen. Etabliert hat sich in diesem Bereich X.509 (Cooper et al. 2008), womit eine hierarchische PKI aufgebaut wird (Baumstruktur). Zertifkate können von einer Certification Authority (CA) ausgestellt werden (Eckert 2008, Seite 381–382). Eine Alternative zu X.509 stellt OpenPGP dar (Callas et al. 2007), das eine dezentrale PKI aufbaut (Graphenstruktur). Durch das gegenseitige Signieren von öffentlichen Schlüsseln wird ein Web of Trust gebildet. Eine freie Implementierung von OpenPGP ist GnuPG (Free Software Foundation 2011).

Um sich an anderen Services zu authentifizieren, kann ein User darüber hinaus ein Add-on verwenden, dass ihm eine OpenID zur Verfügung stellt (The OpenID Community 2007). Mit dieser ID kann er sich an externen Services authentifizieren, ohne dort einen seperaten Account einrichten zu müssen. Da die User oftmals auf die Unterstützung von Agents angewiesen sind (Kapitel 4.2.1.3), benötigen Agents die Möglichkeit, im Namen der User zu agieren. Erreicht wird dies durch die Implementierung von OAuth (Hammer-Lahav 2010). Zur Interaktion zwischen den Komponenten können weitere Standards auf Basis von

Zertifikaten eingesetzt werden, zum Beispiel XML Encryption (Imamura et al. 2002) und XML Signature (Bartel et al. 2008), das mit Hilfe der XML Key Management Specification (XKMS) verwaltet wird (Hallam-Baker und Mysore 2005)

Die Datenspeicherung und der -transport geschehen mittels verschlüsselter Verbindungen. Zur verschlüsselten Speicherung kann die PKI eingesetzt werden, die abgesicherte Punkt-zu-Punkt-Verbindungen ebenfalls ermöglicht. Bei HTTP- und anderen Verbindungen kann der Standard namens Transport Layer Security (TLS), ehemals Secure Sockets Layer (SSL), verwendet werden.

Machine-readable Privacy Policies können auf Basis von P3P umgesetzt werden (Cranor et al. 2002b), was das Konzept von Privacy-by-Design unterstützt (Kapitel 4.1.1.4). Es wird durch APPEL erweitert (Cranor et al. 2002a): User und Services legen vorher ihre Datenschutzeinstellungen fest, die mittels P3P und APPEL automatisiert abgeglichen werden.

Access Control und Rights Management vereinen somit zahlreiche Funktionen, um dem User in einer Entscheidungfindung, wem er seine Daten anvertraut, zu helfen. Was die Funktionen in diesem Entwicklungstadium nicht leisten, ist, wie Daten gelöscht werden können. Zertifkate, die Daten verschlüsseln, können zurückgezogen, aber nicht gelöscht werden, um die Daten unbrauchbar zu machen. Öffentliche Schlüssel können im Web of Trust auf sogenannten Schlüsselservern von Dritten veröffentlicht und eingesehen werden, was identifizierbare Details über die User preisgeben könnte. Rechtsverletzungen, insbesondere im Bereich der Nutzungs- und Urheberrechte, können ebenfalls nur bedingt verhindert werden. Diese Aspekte führen zur Schlussfolgerung, dass Technologie allein keine umfassende Sicherheit gewährleisten kann. Vielmehr ist die Kombination von Technologie mit rechtlichen und gesellschaftlichen Prinzipien nötig.

4.2.2 Kernkomponenten

Die Architektur enthält vier unterschiedliche Kernkomponenten, die bereits in Kapitel 4.2.1.2 entwickelt wurden. In diesem Kapitel werden sie näher erläutert und voneinander abgegrenzt.

4.2.2.1 Portal

Das Portal bietet den Zugang zum OSN und ist das Bindeglied zwischen einem Client und allen Kernkomponenten. Portale enthalten diverse Features, die einen User und dessen Daten betreffen.

Auch in einem dezentral organisierten OSN benötigen die User einen Einstiegspunkt, mit dem sie sich verbinden können, um wesentliche Funktionen zum Social Networking (Kapitel 2.1.1) zur Verfügung gestellt zu bekommen. In zentral organisierten OSNs existiert als Einstiegspunkt eine Webseite, mit deren Startseite (oder auch Homepage genannt) sich der User verbindet. Von dort aus kann der User einen Account einrichten und verwalten, sein Profil bearbeiten und so weiter. In dem hier zu entwickelnden OSN gilt die Kernkomponente Portal als ein solcher Einstiegspunkt.

Das Portal gilt als Bindeglied zwischen Client (vergleiche später Kapitel 4.2.2.4) und allen anderen (Kern-)Komponenten und externen Services/Agents. Portale sind im Idealfall permanent online, das heißt im Internet zu erreichen, damit die Verfügbarkeit der Datentransaktionen, -verarbeitung und -speicherung gewährleistet bleibt.

Ein Portal vereint diverse Features, die die Individualität des Users betreffen. Ein Portal ist die zentrale Instanz für mehrere User und deren Daten, das heißt, mehrere User können ein Portal als Einstiegpunkt zum OSN wählen und dort ihre Daten verwalten. Ein User benötigt lediglich einen einzigen Account im gesamten OSN, der auf einem der Portale verwaltet wird. Folgende Features sind daher einem Portal zuzuordnen:

Accounting: Administration von User-Konten (Accounts)

Profiling: Administration von User-eigenen Profilen

Networking: Vernetzung und Interaktion der User untereinander

Content: Administration von User-eigenen Inhalten, zum Beispiel Blogs, Dokumente, Audio- und Videodateien

Für das Accounting existieren mehrere Add-ons, damit eine Trennung zwischen dem Account eines Users und dessen Profilen geschaffen werden kann. Ein User benötigt lediglich einen Account, kann aber beliebig viele Profile erstellen. Dadurch kann ein User nach außen hin verschiedenen Gruppen unterschiedliche Profile zuweisen. Beispielsweise kann ein öffentliches Profil, welches uneingeschränkt im Web abrufbar ist, lediglich allgemeine Informationen enthalten; in einem Profil für Freunde und Familie werden Hobbys veröffentlicht; ein berufliches Profil enthält einen Lebenslauf.

Im Hintergrund sind User- und Portal-eigene Agents aktiv. Zum Beispiel können User beim Auffinden von Freunden unterstützt werden, indem ihnen andere User empfohlen werden, die ähnliche Interessen und einen ähnlichen Lebenslauf in der Jugend haben. Portale profitieren von ihren Agents, um das Netzwerk der anderen Kernkomponenten zu erschließen.

4.2.2.2 Community

Da es nicht *die eine* Interessengruppe bei einem dezentral organisierten OSN gibt, werden verschiedene Gruppen in Communities organisiert. Hierbei gilt es, „Communities of Interests“, sprich Gruppen mit gemeinsamen Interessen zu bilden und ihnen gruppenspezifische Features zur Nutzung anzubieten.

Während Portale Raum zur Individualisierung der User bieten (Kapitel 4.2.2.1), enthält eine Community Gestaltungsmöglichkeiten für eine Gruppe von Usern. User können sich dort nach Interessensgebieten geordnet in Gruppen zusammenschließen und somit „Communities of Interests“ bilden. Das Spektrum an Interessengebiete ist vielfältig: Mit dieser Kernkomponente lassen sich beispielsweise die Stammtischrunde eines Vereins, typische Werkzeuge für eine Entwickler-Community, die Groupware eines Unternehmens oder die E-Learning-Plattform einer Universität abbilden.

Ein User kann sich beliebig vielen Communities oder auch keiner Community als Teilnehmer anschließen (Membership). Umgekehrt kann eine Community aus beliebig vielen Teilnehmern bestehen, aus Relevanzgründen aber mindestens aus einem Teilnehmer. User greifen nicht direkt auf die Communities zu, sondern wählen den Weg über das jeweilige Portal, das ihren Account verwaltet. Da Communities – wie Portale auch – idealerweise ständig im Internet verfügbar sind, können beide Kernkomponenten die Kommunikation kontinuierlich und im Hintergrund aufrechterhalten.

Folgende Features sind einer Community zuzuordnen:

Grouping: Administration von Gruppen

Membership: Administration von Mitgliedschaften in Gruppen

Networking: Kommunikation zwischen Mitgliedern

Content: Administration von gruppeneigenen Inhalten, zum Beispiel Dokumente und andere Medien

Eine Community-Kernkomponente ist nicht auf eine einzige Interessengruppe beschränkt, sondern kann beliebige Gruppen (oder auch Projekte genannt) verwalten. Die Mitgliedschaft in einer Gruppe wird nicht mit dem Account eines User verknüpft, sondern mit einem seiner Profile. Auf diese Weise wird es einem User erlaubt, unter einem Pseudonym zu agieren, um seine Anonymität zu wahren. Anonymität wird erreicht, indem die Verknüpfung zwischen Profil und Mitgliedschaft aufgehoben wird und jeder User unabhängig von einer Mitgliedschaft die Features der Community nutzt. Die Add-ons Mailinglist und Multi User Chat ergänzen die Mitgliedschaft um zwei Kommunikationsformen: eine Art Forum auf Basis von E-Mail-Mailinglisten (vergleiche das E-Mail-Beispiel aus Kapitel 4.2.1.2) sowie ein Chat für mehrere Akteure gleichzeitig (ebenfalls mit bestehenden Technologien und Standards wie XMPP umsetzbar).

Ähnlich wie bei Portalen erweitern oder unterstützen Agents den Funktionsumfang einer Community. Sie sind in Community- und Group-Agents aufgeteilt.

4.2.2.3 Service

Ein Service stellt verschiedene Dienstleistungen innerhalb des OSNs bereit. Services stehen eher im Hintergrund und sorgen vor allem dafür, die Funktionalität des OSNs zu unterstützen und die Verfügbarkeit zu erhöhen.

In dem dezentral organisierten OSN werden Portale und Communities (Kapitel 4.2.2.1 und 4.2.2.2) durch eine weitere Kernkomponente ergänzt, die Add-ons bereitstellt, damit die Kommunikation zwischen den Kernkomponenten gefördert wird. Die Service-Kernkomponente kann bestehende Funktionen aus anderen Kernkomponenten übernehmen, damit diese nicht mehr unabhängig voneinander die selben Funktionen anbieten müssen und somit Ressourcen verschwenden, sondern die Funktionen an die Service-Kernkomponente auslagern. Diese Kernkomponente kann Funktionen anderer Kernkomponenten erweitern und ergänzen, aber auch neue anbieten.

Die Service-Kernkomponenten sind Teil der Selbstorganisation (Kapitel 4.2.1.3) und treten daher aus Sicht der User in den Hintergrund. Sie bilden das Rückgrat der Architektur, weil sie Funktionen übernehmen, die die Verfügbarkeit und Skalierbarkeit des OSNs erhöhen. Die dauerhafte Verfügbarkeit einer Service-Kernkomponente ist wie bei Portalen und Communities anzustreben. Während Portale für einzelne User und Communities für einzelne User-Gruppen als unverzichtbar gelten, können Services mit demselben Funktionsumfang beliebig oft „kopiert“ werden, um die Verfügbarkeit dieser Funktionen zu erhöhen. (Das setzt allerdings voraus, dass die Funktionen nicht allzu speziell sind wie beispielsweise die Generierung von Zufallswerten mittels eines Quantencomputers, um mit diesen Werten kryptografische Funktionen im OSN zu unterstützen.)

Folgende Features sind einem Service zuzuordnen:

Infratructure: Die IT-Infrastruktur unterhalb der Komponentenebene gilt es zu stabilisieren, das heißt, hochverfügbar und ausfallsicher zu gestalten. Erreicht werden diese Ziele zum Beispiel durch Firewalls, Load Balancer und Nameserver für das DNS.

Access: Der Zugang zu dem OSN könnte in manchen Regionen der Welt zum Beispiel aus politischen Gründen gesperrt werden. Virtual Private Network (VPN)-Gateways und Anonymisierungsdienste wie die Software The Onion Router (TOR) können Restriktionen entgegenwirken (Tor Project 2011). Durch eine verschlüsselte Verbindung erhöhen diese Services darüber hinaus die Sicherheit. Auch zwischen Kernkomponenten geschaltete Anonymisierungsdienste wären denkbar, um beispielsweise mit einem User-Account anonym Communities zu nutzen.

Distribution: Die Verteilung von Daten kann durch Caching (Zwischenspeichern), Mirrors (Server, die Daten spiegeln), Relays (Weiterleitung) und/oder ein CDN beschleunigt werden.

Filtering: Damit Schadsoftware (Malware) und andere unerwünschte Daten wie Spam die Integrität des OSNs nicht schwächen, können Filter-Services implementiert werden. Sie agieren auf Basis von Antiviren-Software oder Black-/White-/Greylisting und können Daten als unerwünscht markieren und anfragende Komponenten bei der Verarbeitung dieser Daten unterstützen.

Storage: Das Speichern von Daten können Service-Komponenten ebenfalls übernehmen. Hier lässt sich eine Technik aus dem Bereich des Grid Computing nutzen: ein verteiltes Dateisystem.

Processing: Die Services zur Verarbeitung von Daten lassen sich in mehrere Kategorien untergliedern, wovon einige an dieser Stelle erwähnt werden.

- **Crawler:** Sie durchsuchen das OSN nach verwertbaren Daten und sammeln diese.
- **Indexer:** Sie indexieren die von Crawlern gesammelten Daten.
- **Directory:** Indexierte Daten werden in Verzeichnissen katalogisiert.
- **Search:** Indexierte Daten werden bei Suchanfragen ausgewertet.
- **Recommender:** Indexierte Daten werden für Empfehlungen ausgewertet.
- **Annotation:** Daten werden automatisch mit passenden Tags oder anderen Daten annotiert.
- **Converter:** Unstrukturierte Daten werden mit zusätzlichen Daten für das Semantic Web aufbereitet und in das RDF-Format konvertiert; strukturierte Daten (Tabelle 4.1) werden direkt in RDF konvertiert (Goddard und Byrne 2010, Seite 3).

Monitoring: Wo es das Konzept der Sicherheit (Kapitel 4.1.1.4) erlaubt, können Daten zur Überwachung des OSNs erhoben, gesammelt und ausgewertet werden.

4.2.2.4 Client

Die als letztes genannte Kernkomponente des dezentral organisierten OSNs ist der Client, der dem User als persönliches Werkzeug zur Verfügung steht. Der Client stellt auf der einen Seite Funktionen zum UI bereit, auf der anderen Seite (zumindest) eine Schnittstelle zur Portal-Kernkomponente (Kapitel 4.2.2.1).

Die Kernkomponente Client erhält eine Sonderrolle, denn ein Client kann sich konzeptionell von den anderen Kernkomponenten unterscheiden. „Kann“ bedeutet in diesem Fall, dass es nicht so sein muss, aber darf (vergleiche das Konzept der Offenheit, Kapitel 4.1.1.2). Bevor auf diese Unterscheidung eingegangen wird, sind weitere Unterschiede zu identifizieren.

Die übrigen Kernkomponenten tragen zur Stabilität des OSNs bei, zum Beispiel durch eine höhere Verfügbarkeit, wenn a) möglichst viele Instanzen miteinander verbunden, b) permanent online sind und c) eine hohe Performanz aufweisen. Ein Client trägt nur bedingt zu dieser Stabilität bei, denn er muss nicht permanent online sein oder eine hohe Performanz aufweisen, weil er tendenziell *einem* User zur Verfügung steht und nicht vielen. Ein User kann seinen persönlichen Client vom OSN trennen, abschalten oder beenden, ohne dass es negative Auswirkungen auf das OSN hätte. Möglich wird dies, weil die Funktionen, die einen User individuell betreffen, nicht vom Client, sondern vom Portal angeboten werden. Ein Client greift folglich auf ein Portal zu, um sich mit dem OSN zu verbinden, Daten zu synchronisieren und dem User ein UI oder – da er ein Mensch ist – ein Grafical User Interface (GUI) bereitzustellen.

Einem Client sind daher folgende Features zuzuordnen:

Human Interface: Dem User wird eine Schnittstelle bereitgestellt (zum Beispiel in Form eines UIs oder GUIs), damit die human-readable Services für ihn dargestellt werden können.

API: Für den Zugriff auf ein Portal verfügt der Client mindestens über eine Web-Service-Schnittstelle.

Synchronisierung: Da Client und Portal über ein Netzwerk miteinander verbunden sein können, ist es sinnvoll, Daten zu synchronisieren (lokal auf dem Client zu speichern) und/oder zwischenzuspeichern (Caching).

Offline-Modus: Die Verbindung zwischen Client und Portal könnte als unterschiedlichen Gründen unterbrochen werden. Damit ein User trotzdem auf die Daten zugreifen kann, benötigt der Client einen Offline-Modus. Mit diesem Modus ist der User in der Lage, offline an Daten weiterzuarbeiten und später, wenn eine Verbindung wiederhergestellt wird, zu synchronisieren.

Um auf andere Kernkomponenten wie Communities oder Services zuzugreifen, kann der Client auf die Web-Service-Schnittstelle des Portals zugreifen. Das Portal nimmt die Anfrage über ein Service-Add-on entgegen. Sollten noch nicht genügend Daten innerhalb des Portals vorhanden sein, um die Anfrage zu beantworten, wird eine Agent-Komponente von der Service-Komponente beauftragt. Diese kommuniziert mit der entsprechenden Kernkomponente und speichert das Resultat im Portal, damit die Service-Komponente dem Client eine Antwort zurückschicken kann.

Neben diesen Features, kann ein Client weitere optional anbieten: Schnittstellen (Interfaces) zu externen Komponenten (wie einem Betriebssystem oder einem anderen OSN) oder zu internen Komponenten (wie ein direkter Zugriff auf E-Mails, dem Beispiel aus Kapitel 4.2.1.2), Erweiterungen in Form von Service oder Agent Add-ons und vieles mehr.

Es existieren verschiedene Typen von Clients, wie in Bild 4.7 dargestellt. Unvollständige Clients, die beispielsweise über kein Human Interface verfügen, werden daher auch als externe Komponenten bezeichnet. Der einfachste Typ eines Clients ist der Webbrowser.

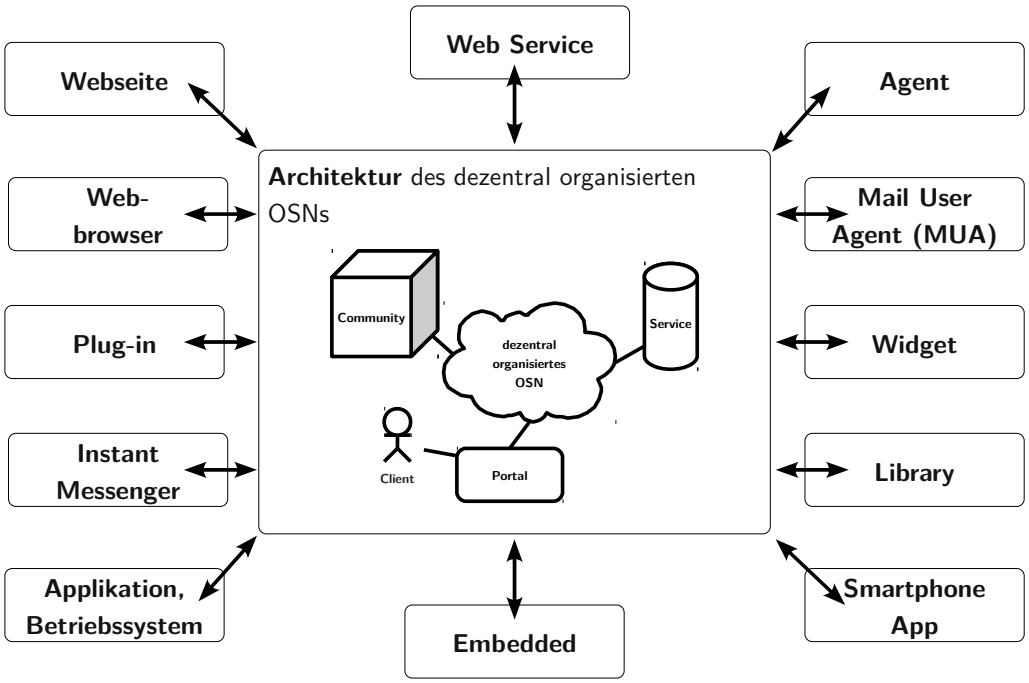


Bild 4.7 Externe Komponenten und Clients, die mit dem OSN interagieren.

4.3 Organisation

Hinter einem dezentral organisierten OSN muss nicht ein einziger Provider stehen. Das Idee sieht vor, dass möglichst viele Stakeholder auf beiden Seiten zusammenkommen: Auf der einen Seite sind es die User, auf der anderen Seite verschiedenartige Organisationen und Privatpersonen, sprich kommerzielle wie nicht-kommerzielle Provider. Neben diesen Providern des OSNs formiert sich eine weitere Organisation um die (Weiter-)Entwicklung des OSNs: die Entwickler-Community, hier Foundation genannt.

4.3.1 Private und (nicht-)kommerzielle Provider

Innerhalb des OSNs bestehen vielerlei Anreize zur Partizipation. Die Möglichkeit der privaten und/oder nicht-kommerziellen Provider existiert ebenso wie die, dass kommerzielle Provider Geschäftsmodelle aus dem bisherigen Social Networking auf den neuen Ansatz übertragen, aber auch neuartige Geschäftsmodelle entwickeln und anwenden können.

Es existieren verschiedene Typen von Providern. Daher ist folgende Abgrenzung angebracht:

- Organisationen
  - mit kommerziellen Interessen
  - mit nicht-kommerziellen Interessen
- Privatpersonen

Organisationen können kommerzielle und/oder nicht-kommerzielle Interessen verfolgen, Privatpersonen naturgemäß nicht-kommerzielle. Zu den Organisationen zählen Unternehmen, Freiberufliche, öffentliche Verwaltungen, Stiftungen, Vereine und so weiter. Folglich sind verschiedene Relationen möglich: Business-to-Consumer (B2C), Business-to-Business (B2B), Business-to-Administration (B2A) und vice versa, wobei Consumer die Privatpersonen und Administration die öffentliche Verwaltung bezeichnen.

Die wesentlichen Leistungen der Provider bestehen im Betreiben und Anbieten der Kernkomponenten (Kapitel 4.2.2) und zusätzlicher Services rund um das OSN. Die Kernleistung besteht in der Teilnahme am OSN und ist in der Regel kostenlos (ist für den Provider aber kein Zwang). Um die Kernleistung siedeln sich verschiedene Nebenleistungen an, die innerhalb oder außerhalb des OSNs kostenpflichtig oder kostenlos angeboten werden (Tabelle 4.7).

Angebot	**intern**	**extern**
kostenpflichtig	Hosting, Support	Anbindung an weitere Medien (Telefon, Mobilfunk, VoIP), Apps
kostenlos	Add-ons	Spiele, Plug-ins, Mashups

Tabelle 4.7 Beispiele für Nebenleistungen der Provider des OSNs.

Bisherige Geschäftsmodelle, beispielsweise auf Basis zusätzlicher kostenpflichtiger Services oder auf Basis personalisierter Werbung, können angewendet werden und sind nicht ausgeschlossen. Neuartige, mit den Ideen und Konzepten (Kapitel 4.1) vereinbare Modelle wären allerdings zu präferieren, um das Vertrauensmodell aufrechterhalten zu können[3].

[3] Von einer ausführlichen Beschreibung der Leistungen und Geschäftsmodelle wird in dieser Arbeit abgesehen. Die Erwähnung gibt daher lediglich einen Einblick, dass das OSN und dessen Architektur Möglichkeiten zur betriebswirtschafltichen Betrachtung bietet.

Durch Dezentralisierung sollen Strukturen vermieden werden, die Marktmacht bündeln und die Bildung von (Quasi-)Monopolen oder Oligopolen begünstigen. Vielmehr sollen Ressourcen (im ökonomischen Sinn, sprich materielle und immaterielle Güter) auf möglichst viele Provider verteilt werden. Es ist möglich, auf Ressourcen anderer Provider zuzugreifen und im Gegenzug anderen Providern eigene Ressourcen anzubieten. Die Aufwendungen eines jeden Providers sollen dadurch möglichst gering ausfallen.

4.3.2 Foundation und Entwickler-Community

Die in Kapitel 2.3.4 vorgestellten Modelle dienen als Vorbild für die Entwickler-Community des dezentral organisierten OSNs. Die Community formiert sich zu einer Organisation, hier Foundation genannt, die von von all denjenigen unterstützt wird, die den Betrieb und die Weiterentwicklung des OSNs aufrechterhalten möchten. Sie gilt somit als organisatorische Basis und „Antriebsmotor“ des OSNs.

Um das Projekt zur (Weiter-)Entwicklung der Architektur und deren Komponenten formiert sich idealerweise eine Entwickler-Community. Sie besteht aus denjenigen Stakeholdern, die die Entwicklung des OSNs und dessen Architektur aktiv oder passiv fördern wollen. Die Community gibt sich einen rechtlichen Rahmen und eine Organisationsstruktur, sprich eine NPO, in der ein Stakeholder Mitglied werden kann. Dieses Vorgehen hat sich bei FLOSS-Communities bewährt, wie zum Beispiel in der ASF (The Apache Software Foundation 2010a). Die Organisation kann sich global formieren und Ableger in verschiedenen Ländern bilden, wie es beispielsweise die Free Software Foundation (FSF) vormacht (Free Software Foundation Europe 2010). Demokratische Strukturen mit eigener Verfassung wie beim Debian-Projekt (Debian 2010) lassen sich innerhalb der NPO ebenso aufbauen wie das Konzept der Meritokratie (durch Mithilfe verdient sich ein Mitglied Mitspracherecht), was bei der ASF angewandt wird (The Apache Software Foundation 2010b). Als Rechtsform bietet sich für Deutschland ein gemeinnütziger, eingetragener Verein an, wie es unter anderem beim europäischen Ableger der FSF der Fall ist. Der Projektname enthält – wie bei anderen NPOs auch – den Zusatz Foundation.

Argumente für eine Foundation, die hinter dem Projekt steht, liefert Ingo (2010): Er beschreibt, dass die „großen“ FLOSS-Communities in einer NPO organisiert sind und tendenziell mehr Erfolg haben als andere Projekte, die beispielsweise rein Community-basiert sind oder hinter denen lediglich ein kommerziell orientiertes Unternehmen steht. Ingo (2010) führt weiter an, dass sich das Projekt gut modularisieren lässt, was auf die hier zu entwickelnde Architektur zutrifft.

Auf die Foundation lassen sich die in Kapitel 4.1.1 genannten Konzepte übertragen, denn sie steht wie die Architektur für dieselben Ziele. Hier entsteht auf den ersten Blick ein Widerspruch: Wie kann das Konzept der Dezentralisierung auf eine scheinbar zentral organisierte Vereinigung wie die der Foundation übertragen werden? Wie bereits weiter oben

angedeutet, ist die Foundation nicht zentral organisiert, sondern möglichst dezentral – mit verschiedenen Untergruppierungen, möglichst vielen Ablegern in anderen Ländern. Zentral sind nur die gemeinsamen Ziele.

Die Organisationsstrukturen beschränken sich nicht nur auf die Entwicklung: Auch der Betrieb kann organisatorisch verwaltet werden. Angelehnt an das Konzept der Virtual Organizations (Foster et al. 2001) bilden die Provider der Kernkomponenten einen virtuellen Zusammenhalt. Da die Provider nicht nur opportunistisch agieren, erweitern sie durch Hinzufügen von (ökonomischen) Ressourcen das OSN.

5 Evaluation

Das in dieser Arbeit diskutierte OSN und dessen Architektur werden in diesem Kapitel kurz evaluiert. Es wird unter anderem darauf eingegangen, inwiefern sie den Anforderungen aus Kapitel 3.3 genügen können. Eine vollständige Evaluation nach Heinrich (2001, Seite 268–273) ist zwar wünschenswert, aber letztlich zu umfangreich für diese Arbeit. Dazu wäre eine dedizierte Studie zum Thema dezentral organisiertes Social Networking angemessen.

5.1 Chancen und Risiken

An dieser Stelle werden die Chancen und Risiken von dezentral organisiertem Social Networking im Allgemeinen (Kapitel 3.5) und dem hier entwickelten OSN mit dessen Architektur im Speziellen (Kapitel 4) zusammengetragen. Die negativen Kritikpunkte verdeutlichen, an welchen Stellen die Ansätze an Grenzen stoßen.

Angesichts der Tatsache, dass an den Auswirkungen von Social Networking und an den Aktivitäten der Provider zentral organisierter OSNs massiv und öffentlich Kritik geübt wird, bietet sich die Chance für einen Paradigmenwechsel zum Vorteil von dezentral organisierten OSNs. Eine Vielzahl an Projekten und Unterstützern geben den Anlass, Dezentralisierung als eine realistische Alternative zu betrachten. Das Risiko für die Projekte, sich mit diesem Paradigma nicht durchzusetzen, erscheint allerdings hoch, wenn die Marktmacht bisheriger Provider betrachtet wird. Da noch keines dieser Projekte eine „kritische Masse“ an Usern für sich gewinnen konnte, bleibt abzuwarten, wie sich OSNs entwickeln werden. Als Nachteilig könnte sich die relativ große Anzahl an Projekten herausstellen, da Kompotenzen scheinbar nicht gebündelt werden, um eine ernsthafte Konkurrenz zu Facebook und anderen OSNs zu etablieren.

Bezogen auf die hier entwickelte Architektur ergeben sich weitere Problemfelder, die abzuwägen sind:

- Das Semantic Web hat sich bisher nicht durchsetzen können (Goddard und Byrne 2010, Seite 4). Standards des Semantic Webs werden als zu komplex erachtet, als dass sie von einer breiten Masse, insbesondere von Software-Entwicklern aufgegriffen und verwendet werden (Goddard und Byrne 2010, Seite 4).

- Neben dem Semantic Web werden P2P-Systeme, Grid Computing und ähnliches in die Architektur einbezogen, was die Komplexität erhöht. Das könnte potenzielle Stakeholder, insbesondere User und Software-Entwickler von einer Partizipation abhalten, weil sie die Zusammenhänge nicht verstehen.
- Ebenso könnten die Konzepte, die dem OSN zugrunde liegen, auf Ablehung stoßen, weil sie als zu ideologisch gewertet werden könnten.
- Die Einbeziehung von dezentralisierten Organisationsstrukturen, ökologischen Aspekten (Green IT) und freien Lizenzen (FLOSS) könnten kommerziellen Interessen entgegenwirken und somit Provider abschrecken.
- Dezentrale Strukturen verkomplizieren die Ermittlung, welche Stakeholder wofür die Verantwortung tragen, falls ein Missbrauch im OSN stattfindet. Die Möglichkeiten der Anonymisierung könnten absichtliche Rechtsverletzungen begünstigen.

Die Liste an Problemfeldern könnte noch weiter geführt werden. Während auf der einen Seite Kritik an der bisherigen Entwicklung der Architektur geübt werden könnte, ergeben sich auf der anderen Seite vielerlei Möglichkeiten der weiteren Entwicklung. Die jeweils an *einem* Standort gebundenen Kernkomponenten könnten in ein sich selbst organisierendes P2P-System überführt werden, um die Dynamik des OSNs weiter auszubauen. Jeder Client und jede andere Kernkomponente würden zu einem Peer umfunktioniert werden. Diese „Vision“ würde die Komplexität der Architektur allerdings weiter erhöhen.

Das Zwischenfazit lautet: Das OSN bietet einige Chancen, birgt aber auch Risiken, zum Beispiel durch die Komplexität. Zur Lösung der Probleme und zum Abmildern von Risiken ist die (permanente) Weiterentwicklung der Architektur von Nöten.

5.2 Limitation

Neben den in Kapitel 5.1 behandelten Risiken weist diese Arbeit aufgrund ihres Charakters eine Limitation in der Betrachtungsweise auf. Im Folgenden wird kritisch hinterfragt, inwieweit diese Arbeit dem Umfang des diskutierten OSNs und dessen entwickelter Architektur genügt.

Aufgrund des Charakters dieser Arbeit kann kein vollständiges, bis in hohe Detailstufen reichendes Design der Architektur gezeigt werden. Es wurden daher Kompromisse eingegangen, um mehr Beispiele für einzelne Aspekte zu liefern, als eine vollständige Spezifikation niederzuschreiben. Die Diskussion um die Entwicklung herum reißt Themen an, die nur unvollständig wiedergegeben wurden: der organisatorische Aufbau der Foundation und der Entwickler-Community sowie die Philosophien und Ideologien hinter den Konzepten; die „interne“ Gestaltung der Architektur-Komponenten mittels Daten- und Schichtenmodellen; die Informationsarchitektur zur Gestaltung der Webseiten; weitere Add-ons mit konkreten

Anwendungsszenarien und Ablaufdiagrammen und so weiter. Diese Aspekte hätten jedoch einen weitaus größeren Umfang der Arbeit erfordert oder hätten den weitgefassten Begriff der „Entwicklung“, der in der inhaltlichen Ausgestaltung als Grundlage für weitere Fragstellungen dient, in der erforderlichen Länge beschnitten.

Diese Arbeit genügt demnach nicht dem Anspruch einer umfassenden Architekturentwicklung, denn sie schließt die begonnenen Entwicklungen nicht ab. Sie gibt vielmehr die theoretische Basis für kommende Entwicklungen wieder.

6 Schlussbetrachtung

Als Abschluss werden die wichtigsten Aspekte der Arbeit zusammengefasst. Diesem Fazit folgt ein kurzer Ausblick.

Das in dieser Arbeit entwickelte OSN und dessen Architektur integrieren viele bereits existierende, in der Praxis erprobte Konzepte, Paradigmen, Technologien und Standards. Es wurde gezeigt, wie Synergieeffekte durch die Integration und Wiederverwendung ebendieser, die im Terminus „Free/Libre and Open Source Decentralized, Distributed and Semantic Social Networking“ angeschnitten werden, zu erzielen sind. Durch die gleichzeitige Betrachtung aus fünf verschiedenen Perspektiven ergab sich die Chance, ein möglichst umfassendes Bild darüber zu liefern, wie dezentral organisiertes Social Networking gestaltet werden kann und muss, um den heutigen Anforderungen an das Social Networking zu genügen.

Diese Arbeit hat sich auf die theoretische Entwicklung der Architektur konzentriert, um eine Basis für weitere Fragestellungen zu liefern. Wünschenswert wäre darüber hinaus die Fortführung, die Diskussion und der Ausbau der Konzepte und Ideen, der Architektur und der organisatorischen Strukturen – eine Umsetzung ebendieser nicht ausgeschlossen.

Quellen

Adida, Ben; Birbeck, Mark; McCarron, Shane; Pemberton, Steven (2008): RDFa in XHTML: Syntax and Processing – A collection of attributes and processing rules for extending XHTML to support RDF. http://www.w3.org/TR/rdfa-syntax/, abgerufen am 26. November 2010.

Auer, Sören; Lehmann, Jens; Bizer, Christian (2009): Semantische Mashups auf Basis Vernetzter Daten. In: Blumauer, Andreas; Pellegrini, Tassilo (Hrsg.): Social Semantic Web – Web 2.0 – Was nun?, Kap. 14, Springer, Hamburg, Heidelberg, S. 259–286.

Bartel, Mark; Boyer, John; Fox, Barb; LaMacchia, Brian; Simon, Ed (2008): XML Signature Syntax and Processing (Second Edition). http://www.w3.org/TR/xmldsig-core/, abgerufen am 12. November 2010.

Berners-Lee, Tim (1998): Notation 3. Version vom 9. März 2006, http://www.w3.org/DesignIssues/Notation3.html, abgerufen am 12. November 2010.

Berners-Lee, Tim (2000): Architecture. http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html, abgerufen am 12. November 2010.

Berners-Lee, Tim (2006): Linked Data. Version vom 18. Juni 2009, http://www.w3c.org/DesignIssues/LinkedData.html, abgerufen am 9. November 2010.

Berners-Lee, Tim; Fielding, Roy Thomas; Masinter, Larry (2005): RFC 3986: Uniform Resource Identifier (URI) – Generic Syntax. http://tools.ietf.org/rfc/rfc3986.txt, abgerufen am 3. Dezember 2010.

Berners-Lee, Tim; Hendler, James; Lassila, Ora (2001): The Semantic Web. In: Scientic American 284 (5), S. 34–43, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.115.9584&rep=rep1&type=pdf, abgerufen am 30. Juni 2010.

Berners-Lee, Tim; Masinter, Larry; McCahill, Mark P. (1994): RFC 1738: Uniform Resource Locators (URL). http://tools.ietf.org/rfc/rfc1738.txt, abgerufen am 3. Dezember 2010.

BitTorrent.org (2011): Index of Proposals. http://www.bittorrent.org/beps/bep_0000.html, abgerufen am 12. Januar 2011.

Blumauer, Andreas; Pellegrini, Tassilo (2009): Semantic Web Revisited – Eine kurze Einführung in das Social Semantic Web. In: Blumauer, Andreas; Pellegrini, Tassilo (Hrsg.): Social Semantic Web – Web 2.0 – Was nun?, Kap. 1, Springer, Hamburg, Heidelberg, S. 3–22.

Busch, Thorsten (2007): Freie Software als Entwicklungshelfer? – Corporate Citizenship und der Digital Divide. VDM-Verlag, Saarbrücken.

Busemann, Katrin; Gscheidle, Christoph (2010): Web 2.0: Nutzung steigt – Interesse an aktiver Teilhabe sinkt. In: Media Perspektiven 7-8/2010, S. 359–368, `http://www.media-perspektiven.de/uploads/tx_mppublications/07-08-2010_Busemann.pdf`, abgerufen am 24. September 2010.

Callas, Jon; Donnerhacke, Lutz; Finney, Hal; Shaw, David; Thayer, Rodney (2007): RFC 4880: OpenPGP Message Format. `http://www.ietf.org/rfc/rfc4880.txt`, abgerufen am 31. Januar 2011.

Chaos Computer Club (2011): hackerethics. `http://ccc.de/de/hackerethics`, abgerufen am 19. Januar 2011.

Chinnici, Roberto; Moreau, Jean-Jacques; Ryman, Arthur; Weerawarana, Sanjiva (2007): Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. `http://www.w3.org/TR/wsdl20/`, abgerufen am 26. November 2010.

Clement, Luc; Hately, Andrew; von Riegen, Claus; Rogers, Tony (2004): UDDI Version 3.0.2. `http://uddi.org/pubs/uddi_v3.htm`, abgerufen am 26. November 2010.

Cooper, David; Santesson, Stefan; Farrell, Stephen; Boeyen, Sharon; Housley, Russell; Polk, Tim (2008): RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. `http://tools.ietf.org/rfc/rfc5280.txt`, abgerufen am 31. Januar 2011.

Cranor, Lorrie; Langheinrich, Marc; Marchiori, Massimo (2002a): A P3P Preference Exchange Language 1.0 (APPEL1.0). `http://www.w3.org/TR/P3P-preferences/`, abgerufen am 12. November 2010.

Cranor, Lorrie; Langheinrich, Marc; Marchiori, Massimo; Presler-Marshall, Martin; Reagle, Joseph (2002b): The Platform for Privacy Preferences 1.0 (P3P1.0) Specification. `http://www.w3.org/TR/P3P/`, abgerufen am 12. November 2010.

Creative Commons (2007): Legal Concepts. `http://wiki.creativecommons.org/index.php?title=Legal_Concepts&oldid=8874`, abgerufen am 26. Oktober 2010.

Creative Commons (2010a): Licenses. `http://creativecommons.org/about/licenses`, abgerufen am 26. Oktober 2010.

Creative Commons (2010b): What is CC? `http://creativecommons.org/about/what-is-cc`, abgerufen am 26. Oktober 2010.

Cremonini, Luca (2006): Web 2.0 Map. `http://upload.wikimedia.org/wikipedia/commons/a/a7/Web_2.0_Map.svg`, abgerufen am 3. Dezember 2010.

de Bruijn, Jos; Fensel, Dieter; Kerrigan, Mick; Keller, Uwe; Lausen, Holger; Scicluna, James (2008): Modeling Semantic Web Services – The Web Service Modeling Language. Springer, Berlin, Heidelberg.

Debian (2010): `http://debian.org/`, abgerufen am 26. November 2010.

Definition of Free Cultural Works (2008): Definition. http://freedomdefined.org/index.php?title=Definition&oldid=5437, abgerufen am 26. Oktober 2010.

DiSo (2010): http://diso-project.org/, abgerufen am 16. April 2010.

Dublin Core Metadata Initiative (2010): DCMI Specifications. http://dublincore.org/specifications/, abgerufen am 26. November 2010.

Dürst, Martin; Suignard, Michel (2005): RFC 3987: Internationalized Resource Identifiers (IRIs). http://tools.ietf.org/rfc/rfc3987.txt, abgerufen am 7. Dezember 2010.

Eckert, Claudia (2008): IT-Sicherheit: Konzepte – Verfahren – Protokolle. 5. Aufl., Oldenbourg, München.

Facebook (2010): http://www.facebook.com/, abgerufen am 8. September 2010.

Facebookmarketing.de (2011): Facebook Nutzerzahlen. http://facebookmarketing.de/userdata/, abgerufen am 18. Januar 2011.

Faltin, Günter (2008): Kopf schlägt Kapital. 5. Aufl., Hanser, München.

Fielding, Roy Thomas (2000): Architectural Styles and the Design of Network-based Software Architectures. Dissertation, University of California, Irvine, http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf, abgerufen am 15. November 2010.

Fielding, Roy Thomas; Gettys, J.; Mogul, Jeffrey C.; Frystyk, Henrik; Masinter, Larry; Leach, James; Berners-Lee, Tim (1999): RFC 2616: Hypertext Transfer Protocol – HTTP/1.1. http://www.ietf.org/rfc/rfc2616.txt, abgerufen am 15. November 2010.

Foster, Ian; Kesselman, Carl; Tuecke, Steven (2001): The Anatonmy of the Grid – Enabling Scalable Virtual Organizations. In: International Journal of High Performance Computing Applications 15 (3), S. 200–222.

Free Software Foundation (2007): GNU General Public License. http://www.gnu.org/licenses/gpl-3.0.txt, abgerufen am 26. Oktober 2010.

Free Software Foundation (2010a): The Free Software Definition. http://www.gnu.org/philosophy/free-sw.html, abgerufen am 26. Oktober 2010.

Free Software Foundation (2010b): The GNU Operating System. http://www.gnu.org/, abgerufen am 26. Oktober 2010.

Free Software Foundation (2011): The GNU Privacy Guard. http://gnupg.org/, abgerufen am 31. Januar 2011.

Free Software Foundation Europe (2010): The FSF* network. http://fsfe.org/about/fsfnetwork.en.html, abgerufen am 26. Oktober 2010.

Freedman, Michael J. (2010): Experiences with CoralCDN: A Five-Year Operational View. In: 7th USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '10), San Jose, http://www.coralcdn.org/docs/coral-nsdi10.pdf, abgerufen am 19. November 2010.

Friedman, Vitaly (2009): Praxisbuch Web 2.0 – Moderne Webseiten programmieren und gestalten. 2. Aufl., Galileo Press, Bonn.

Friend2Friend (2010): http://wiki.friend2friend.net/wiki/Main_Page, abgerufen am 16. April 2010.

GNUnet (2011): About GNUnet. https://gnunet.org/, abgerufen am 12. Januar 2011.

Goddard, Lisa; Byrne, Gillian (2010): Linked Data tools: Semantic Web for the masses. In: First Monday 15 (11), S. –, http://www.uic.edu/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/3120/2633, abgerufen am 15. November 2010.

Google (2010): Google Announces Third Quarter 2010 Financial Results. http://investor.google.com/earnings/2010/Q3_google_earnings.html, abgerufen am 19. Januar 2011.

Google (2011): Weitere Google-Produkte. http://www.google.de/intl/de/options/, abgerufen am 19. Januar 2011.

Grippi, Daniel; Salzberg, Maxwell; Sofaer, Raphael; Zhitomirskiy, Ilya (2010): Diaspora. http://www.joindiaspora.com/, abgerufen am 16. Mai 2010.

Gümüs, Özgür; Gürcan, Önder; Kardas, Geylani; Ekinci, Erdem Eser; Dikenelli, Oguz (2007): Engineering an MAS Platform for Semantic Service Integration Based on the SWSA. In: Meersman, Robert; Tari, Zahir; Herrero, Pilar (Hrsg.): On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops, Lecture Notes in Computer Science, Bd. 4805, Springer, Berlin, Heidelberg, S. 85–94.

Hadley, Marc (2009): Web Application Description Language. http://www.w3.org/Submission/wadl/, abgerufen am 15. November 2010.

Hallam-Baker, Phillip; Mysore, Shivaram H. (2005): XML Key Management Specification (XKMS 2.0). http://www.w3.org/TR/xkms2/, abgerufen am 12. November 2010.

Hammer-Lahav, Eran (2010): RFC 5849: The OAuth 1.0 Protocol. http://tools.ietf.org/rfc/rfc5849.txt, abgerufen am 31. Januar 2011.

Hartmann, Bernd; Jansen, Felix (2008): Open Content – Open Access: Freie Inhalte als Herausforderung für Wirtschaft, Wissenschaft und Politik, Fazit-Schriftenreihe, Bd. 16. MFG Stiftung Baden-Württemberg, Stuttgart, http://www.fazit-forschung.de/fileadmin/_fazit-forschung/downloads/FAZIT-Schriftenreihe_Band_16.pdf, abgerufen am 26. Oktober 2010.

Heinrich, Lutz J. (2001): Wirtschaftsinformatik – Einführung und Grundlegung. 2. Aufl., Oldenbourg, München.

heise online (2009): Internet-Bevölkerung übersteigt Milliardengrenze. http://www.heise.de/newsticker/Internet-Bevoelkerung-uebersteigt-Milliardengrenze--/meldung/122291, abgerufen am 21. August 2009.

heise online (2010): Auf der Suche nach des Twitters Kern. http://www.heise.de/newsticker/meldung/Auf-der-Suche-nach-des-Twitters-Kern-1143452.html, abgerufen am 19. Januar 2011.

HelloWorld (2008): http://www.helloworld-network.org/, abgerufen am 14. Mai 2010.

Hendler, James (2001): Agents and the Semantic Web. In: IEEE Intelligent Systems 16 (2), S. 30–37, http://icc.mpei.ru/documents/00000828.pdf, abgerufen am 30. Juni 2010.

Hickson, Ian (2010): HTML5 – A vocabulary and associated APIs for HTML and XHTML. Entwurf vom 19. Oktober 2010, http://www.w3.org/TR/2010/WD-html5-20101019/, abgerufen am 26. November 2010.

Hillenbrand, Markus; Müller, Paul (2005): Web Services and Peer-to-Peer. In: Steinmetz, Ralf; Wehrle, Klaus (Hrsg.): Peer-to-Peer Systems and Applications, Kap. 14, Springer, Berlin, Heidelberg, S. 207–224.

Hitzler, Pascal; Krötzsch, Markus; Parsia, Bijan; Patel-Schneider, Peter F.; Rudolph, Sebastian (2009): OWL 2 Web Ontology Language Primer. http://www.w3.org/TR/owl-primer/, abgerufen am 12. November 2010.

Imamura, Takeshi; Dillaway, Blair; Simon, Ed (2002): XML Encryption Syntax and Processing. http://www.w3.org/TR/xmlenc-core/, abgerufen am 12. November 2010.

Ingo, Henrik (2010): How to grow your open source project 10x and revenues 5x. http://openlife.cc/blogs/2010/november/how-grow-your-open-source-project-10x-and-revenues-5x, abgerufen am 6. Dezember 2010.

Internet World Stats (2010a): Facebook Users in the World. http://www.internetworldstats.com/stats25.htm, abgerufen am 18. Januar 2011.

Internet World Stats (2010b): Internet Usage Statistics. http://www.internetworldstats.com/stats.htm, abgerufen am 18. Januar 2011.

KDE Social Desktop (2009): http://techbase.kde.org/Projects/Social-Desktop, abgerufen am 13. Oktober 2009.

Koch, Michael; Richter, Alexander (2009): Enterprise 2.0 – Planung, Einführung und erfolgreicher Einsatz von Social Software in Unternehmen. 2. Aufl., Oldenbourg, München.

kopal (2010): http://code.google.com/p/kopal/, abgerufen am 16. April 2010.

Lassila, Ora; Hendler, James (2007): Embracing "Web 3.0". In: IEEE Internet Computing 11 (3), S. 90–93, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.80.3801&rep=rep1&type=pdf, abgerufen am 25. August 2010.

Lehmann, Katharina Anna; Kaufmann, Michael (2005): Random Graphs, Small-Worlds and Scale-Free Networks. In: Steinmetz, Ralf; Wehrle, Klaus (Hrsg.): Peer-to-Peer Systems and Applications, Kap. 6, Springer, Berlin, Heidelberg, S. 57–76.

Lenk, Alexander; Klems, Markus; Nimis, Jens; Tai, Stefan; Sandholm, Thomas (2009): What's Inside the Cloud? An Architectural Map of the Cloud Landscape. In: Software Engineering Challenges of Cloud Computing, 2009. CLOUD '09. ICSE Workshop on, Vancouver, S. 23–31.

LinkedIn Corporation (2011): LinkedIn. http://www.linkedin.com/, abgerufen am 12. Januar 2011.

Manola, Frank; Miller, Eric (2004): RDF Primer. http://www.w3.org/TR/rdf-primer/, abgerufen am 12. November 2010.

Mauthe, Andreas; Heckmann, Oliver (2005): Distributed Computing – Grid Computing. In: Steinmetz, Ralf; Wehrle, Klaus (Hrsg.): Peer-to-Peer Systems and Applications, Kap. 13, Springer, Berlin, Heidelberg, S. 193–206.

McBride, Brian (2004): RDF Vocabulary Description Language 1.0: RDF Schema. http://www.w3.org/TR/rdf-schema/, abgerufen am 12. November 2010.

Melzer, Ingo (2008): Service-orientierte Architekturen mit Web Services – Konzepte – Standards – Praxis. 3. Aufl., Spektrum Akademischer Verlag, Heidelberg.

mensch.coop (2010): https://mensch.coop/, abgerufen am 22. April 2010.

Morville, Peter; Rosenfeld, Louis (2007): Information Architecture for the World Wide Web. 3. Aufl., O'Reilly, Sebastopol.

mozilla.org (2010): Mozilla Organizations. http://www.mozilla.org/about/organizations.html, abgerufen am 26. Oktober 2010.

NoseRub (2010): http://noserub.com/, abgerufen am 16. April 2010.

Open Source Initiative (2008): The BSD License. http://www.opensource.org/licenses/bsd-license.php, abgerufen am 26. Oktober 2010.

Open Source Initiative (2010a): The Open Source Definition. http://www.opensource.org/docs/osd, abgerufen am 26. Oktober 2010.

Open Source Initiative (2010b): Open Source Licenses by Category. http://www.opensource.org/licenses/category, abgerufen am 26. Oktober 2010.

OpenOffice.org Deutschland e. V. (2010): The Document Foundation. http://www.documentfoundation.org/, abgerufen am 26. Oktober 2010.

O'Reilly, Tim (2005): What Is Web 2.0 – Design Patterns and Business Models for the Next Generation of Software. http://oreilly.com/lpt/a/6228, abgerufen am 3. Dezember 2010.

Paschke, Marian (2009): Medienrecht. 3. Aufl., Springer, Berlin, Heidelberg.

Pellegrini, Tassilo; Blumauer, Andreas (2009): Social Semantic Web. In: Back, Andrea; Gronau, Norbert; Tochtermann, Klaus (Hrsg.): Web 2.0 in der Unternehmenspraxis – Grundlagen, Fallstudien und Trends zum Einsatz von Social Software, 2. Aufl., Kap. 5.4, Oldenbourg, München, S. 270–279.

Pemberton, Steven; Austin, Daniel; Axelsson, Jonny; Çelik, Tantek; Dominiak, Doug; Elenbaas, Herman; Epperson, Beth; Ishikawa, Masayasu; Matsui, Shin'ichi; McCarron, Shane; Navarro, Ann; Peruvemba, Subramanian; Relyea, Rob; Schnitzenbaumer, Sebastian; Stark, Peter (2000): XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition) – A Reformulation of HTML 4 in XML 1.0. Revision 1 vom August 2002, http://www.w3.org/TR/xhtml1/, abgerufen am 26. November 2010.

Prud'hommeaux, Eric; Seaborne, Andy (2008): SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/, abgerufen am 26. November 2010.

Raymond, Eric Steven (1997): The Cathedral and the Bazaar. Version 3.0 vom 2. August 2002, http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/, abgerufen am 11. Mai 2010.

Raymond, Eric Steven (2003): The Art of Unix Programming. http://www.catb.org/~esr/writings/taoup/html/, abgerufen am 28. Januar 2011.

Richardson, Leonard; Ruby, Sam (2007): RESTful Web Services – Web Services for the Real World. O'Reilly, Sebastopol.

Schaffert, Sebastian (2006): Semantic Social Software – Semantically Enabled Social Software or Socially Snabled Semantic Web? Techn. Ber., Salzburg Research Forschungsgesellschaft, http://www.schaffert.eu/download/Schaffert2006_SemanticSocialSoftware.pdf, abgerufen am 30. Juni 2010.

Schmidt, Jan (2006): Social Software – Onlinegestütztes Informations-, Identitäts- und Beziehungsmanagement. In: Forschungsjournal Neue Soziale Bewegungen 2, S. 37–47.

Schmieder, Fabian (2010): Im Meer der tausend Klauseln – Das Kleingedruckte bei Twitter, Facebook & Co. In: c't magazin für computertechnik 19.9.2010 (20), S. 148–151.

Schoder, Detlef; Fischbach, Kai; Schmitt, Christian (2005): Application Areas. In: Steinmetz, Ralf; Wehrle, Klaus (Hrsg.): Peer-to-Peer Systems and Applications, Kap. 4, Springer, Berlin, Heidelberg, S. 25–32.

Signore, Oreste (2005): The Semantic Web stack. http://www.w3c.it/talks/2005/openCulture/slide7-0.html, abgerufen am 12. November 2010.

Sommerville, Ian (2001): Software Engineering. 6. Aufl., Addison-Wesley, Harlow.

Spiekermann, Sarah; Cranor, Lorrie Faith (2009): Engineering Privacy. In: IEEE Transactions on Software Engineering 35, S. 67–82, http://amor.cms.hu-berlin.de/~spiekers/website/pdfs/Engineering%20Privcy_IEEE.pdf, abgerufen am 30. Januar 2011.

Spivack, Nova (2004): New Version of My „Metaweb" Graph – The Future of the Net. http://www.novaspivack.com/science/new-version-of-my-metaweb-graph-the-future-of-the-net, abgerufen am 2. Dezember 2010.

Steinmetz, Ralf; Wehrle, Klaus (2005): What Is This „Peer-to-Peer" About? In: Steinmetz, Ralf; Wehrle, Klaus (Hrsg.): Peer-to-Peer Systems and Applications, Kap. 2, Springer, Berlin, Heidelberg, S. 9–16.

Swisher, Kara (2010): Exclusive: Twitter Raises $200 Million at a $3.7 Billion Valuation; Adds McCue and Rosenblatt to Board. The Wall Street Journal: All Things Digital, http://kara.allthingsd.com/20101215/exclusive-twitter-raises-200-million-at-3-7-billion-valuation-adds-mccue-and-rosenblatt-to-board/, abgerufen am 19. Januar 2011.

The Apache Software Foundation (2004): Apache License. http://www.apache.org/licenses/LICENSE-2.0.txt, abgerufen am 26. Oktober 2010.

The Apache Software Foundation (2010a): The Foundation. http://apache.org/foundation/, abgerufen am 26. Oktober 2010.

The Apache Software Foundation (2010b): How the ASF works. http://apache.org/foundation/how-it-works.html, abgerufen am 26. Oktober 2010.

The Appleseed Collective (2008): The Appleseed Project. http://appleseed.sourceforge.net/, abgerufen am 10. Mai 2010.

The Freenet Project (2011): What is Freenet? http://freenetproject.org/whatis.html, abgerufen am 12. Januar 2011.

The OpenID Community (2007): OpenID Authentication 2.0 – Final. http://openid.net/specs/openid-authentication-2_0.html, abgerufen am 31. Januar 2011.

Tor Project (2011): Overview. https://www.torproject.org/about/overview.html.en, abgerufen am 30. Januar 2011.

Twitter (2010): http://twitter.com/, abgerufen am 8. September 2010.

van Eimeren, Birgit; Frees, Beate (2010): Fast 50 Millionen Deutsche online – Multimedia für alle? In: Media Perspektiven 7-8/2010, S. 334–349, http://www.media-perspektiven.de/uploads/tx_mppublications/07-08-2010_Eimeren.pdf, abgerufen am 24. September 2010.

Verstrynge, Jérôme (2010): Chaupal, a P2P Framework. http://wiki.apache.org/incubator/ChaupalProposal, abgerufen am 12. Januar 2011.

Vodafone Group (2010): onesocialweb. http://onesocialweb.org/, abgerufen am 17. Mai 2010.

W3C (2010): Semantic Web. http://www.w3.org/standards/semanticweb/, abgerufen am 30. November 2010.

Wehrle, Klaus; Götz, Stefan; Rieche, Simon (2005): Distributed Hash Tables. In: Steinmetz, Ralf; Wehrle, Klaus (Hrsg.): Peer-to-Peer Systems and Applications, Kap. 7, Springer, Berlin, Heidelberg, S. 79–93.

Winer, Dave (1999): XML-RPC Specification. Version vom 30. Juni 2003, http://www.xmlrpc.com/spec, abgerufen am 26. November 2010.

XING AG (2011): XING. http://www.xing.com/, abgerufen am 12. Januar 2011.

XMPP Standards Foundation (2010): XMPP Technologies Overview. http://xmpp.org/about-xmpp/technology-overview/, abgerufen am 29. Januar 2011.

Abbildungen

2.1 Logo der World Wide Web Consortiums 7
2.2 Eine Tag Cloud über das Web 2.0 nach Cremonini (2006). 8
2.3 Das Social-Software-Dreieck nach Koch und Richter (2009, Seite 11–14), basierend auf Schmidt (2006). 9
2.4 Der Semantic Web Stack nach Signore (2005), basierend auf Berners-Lee (2000). 13
2.5 Die Entwicklungspfade zum Social Semantic Web entlang der Dimensionen Wissensorganisation und Inhaltserschließung nach Pellegrini und Blumauer (2009, Seite 273). 14
2.6 Das Metaweb nach Spivack (2004). 15
2.7 Klassifikation von P2P-Systemen nach Steinmetz und Wehrle (2005, Seite 11). 17
2.12 Logo der Open Source Initiative 23
2.13 Logo des GNU-Projekts 24
2.14 Logo der Creative Commons 24
2.8 Cloud Stack nach Lenk et al. (2009, Seite 25). 27
2.9 Der SOA-Tempel nach Melzer (2008, Seite 13). 28
2.10 Das magische Dreieck einer SOA nach Melzer (2008, Seite 14). 29
2.11 Verbindung zu einem Web Service mittels WSDL nach de Bruijn et al. (2008, Seite 19). 30

4.1 Die Relation zwischen den Entitäten Stakeholder und Rolle. 43
4.2 Das dezentral organisierte OSNs besteht aus vier Kernkomponenten. . . . 53
4.3 Die Architektur ist modular aufgebaut. 53
4.4 Verschiedene Service und Agent Add-ons beteiligen sich am Senden und Empfangen von E-Mail-Nachrichten. 57
4.5 Das OSN wächst mit jeder hinzugefügten Kernkomponente und wird dadurch komplexer. 59
4.6 Das P2P-CDN unterstützt die Kernkomponenten beim Datenaustausch. . 61
4.7 Externe Komponenten und Clients, die mit dem OSN interagieren. 68

Tabellen

4.1 Die Daten, die im OSN verwaltet werden, lassen sich kategorisieren. . . . 51
4.2 Formate bei HTTP-Anfragen (Request) und -Antworten (Response). . . . 52
4.3 Die allgemeine, für alle Kernkomponenten gültige Web-Service-Schnittstelle. 54
4.4 Die allgemeinen Add-ons bilden die Basis der Web-Service-Schnittstelle. . 55
4.5 Add-ons zur Administration der Komponenten erweitern die Web-Service-Schnittstelle. 56
4.6 Die Agent-Add-ons erweitern die Web-Service-Schnittstelle. 60
4.7 Beispiele für Nebenleistungen der Provider des OSNs. 69

Quelltexte

2.1 Schematischer Aufbau einer HTTP URL nach Berners-Lee et al. (1994) . . 7
2.2 Das Beispiel zeigt eine einfache Anfrage an einen Web Service, der RESTful und Ressourcen-orientiert ist. 21

4.1 Die Web-Service-Schnittstelle wird durch den Aufruf einer URL (hier generisch dargestellt) via HTTP angesprochen. 54

Akronyme

AGB	Allgemeine Geschäftsbedingungen
API	Application Programming Interface
AS	Autonomous System
ASF	Apache Software Foundation
B2A	Business-to-Administration
B2B	Business-to-Business
B2C	Business-to-Consumer
CA	Certification Authority
CDN	Content Distribution Network
CS	Closed Source
CSR	Corporate Social Responsibility
CSS	Closed Source Software
DDOS	Distributed Denial of Service
DHT	Distributed Hash Table
DNS	Domain Name System
FLOSS	Free/Libre and Open Source Software
FSF	Free Software Foundation
GNU	GNU's Not Unix
GPL	GNU General Public License
GUI	Grafical User Interface
HTML	Hypertext Markup Langage
HTTP	Hypertext Transfer Protocol
HuaaS	Human as a Service
IaaS	Infrastructure as a Service
IM	Instant Messaging
IRI	Internationalized Resource Identifier
ISP	Internet Service Provider
IT	Informationstechnologie

JSON	JavaScript Object Notation
MDA	Mail Delivery Agent
MRA	Mail Retrieval Agent
MSA	Message Submission Agent
MTA	Mail Transfer Agent
MUA	Mail User Agent
N3	Notation3
NPO	Non-Profit-Organisation
OS	Open Source
OSI	Open Source Initiative
OSN	Online Social Network
OWL	Web Ontology Language
OWL-DL	OWL Description Logic
P2P	Peer-to-Peer
PaaS	Platform as a Service
PIM	Personal Information Manager
PKI	Public-Key-Infrastruktur
QoS	Quality of Service
RDF	Resource Description Framework
RDFa	RDF in Attributes
RDFS	RDF Schema
REST	Representational State Transfer
RPC	Remote Procedure Call
SaaS	Software as a Service
SNS	Social Networking Service
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Query Language
SSL	Secure Sockets Layer
TLS	Transport Layer Security
TOR	The Onion Router
UDDI	Universal Description Discovery and Integration
UI	User Interface
URI	Uniform Resource Identifier

URL	Uniform Resource Locator
USD	US-Dollar
VoIP	Voice over IP
VPN	Virtual Private Network
W3C	World Wide Web Consortium
WADL	Web Application Description Language
WSDL	Web Services Description Language
WWW	World Wide Web
XaaS	Everything as a Service
XHTML	eXtensible HyperText Markup Language
XKMS	XML Key Management Specification
XML	eXtensible Markup Language
XMPP	Extensible Messaging and Presence Protocol