Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Florian Burka

Ein verteiltes Agentensystem zur Koordination sozialer Interaktion

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Florian Burka

Ein verteiltes Agentensystem zur Koordination sozialer Interaktion

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang Informatik (Master)

am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. rer. nat. Gunter Klemke
Zweitgutachter : Prof. Dr. rer. nat. Kai von Luck

Abgegeben am 14. Oktober 2010

Florian Burka

Thema der Bachelorarbeit

Ein verteiltes Agentensystem zur Koordination sozialer Interaktion

Stichworte

Verteilte Systeme, Agenten, Terminplanung, Kalender, Persönliche Assistenten

Kurzzusammenfassung

In dieser Arbeit wird eine verteilte Agentengesellschaft vorgestellt, welche in der Lage ist heterogene Benutzergruppen bei der Koordination von sozialer Interaktion proaktiv zu unterstützen.

Bisherige Forschungen oder kommerzielle Umsetzungen von Groupware oder Kalendern haben ihren Fokus entweder auf eine homogene Anwenderschaft gelegt oder sind rein passive Systeme.

Das vorgelegte Konzept ist ein Schritt zu mehr sozialer Interaktion und bietet einige Ideen, welche zeigen wie ein Computer unterstützend dem Menschen helfen und somit Zeit schenken kann.

Florian Burka

Title of the paper

A System Of Distributed Agents To Coordinate Social Interaction

Keywords

Agents, Calendaring, Time Management, Scheduling, Personal Assistans

Abstract

This thesis deals with event management by the means of a society of distributed agents. A system that actively helps to coordinate a heterogeneously mixed user base and proactively schedule events.

Current research only covers groupware systems for closed user groups and coordination of those. Commercially available software on the other hand offers either inherently passive systems or systems for homogenous user groups.

This work presents a step towards a more proactive event planning for heterogeneous user groups. It tries ideas to illustrate the usage of computers focused on time saving and supporting humans.

Inhaltsverzeichnis

1 Einführung

1.1 Motivation

Termine in heterogenen Gruppen[1] zu koordinieren, ist eine beliebig komplex werdende Aufgabe. Bestehende Systeme bieten entweder eine unterstützte Koordination von zentral verwalteten Benutzergruppen, oder sie bieten eine Plattform, über welche sich heterogene Benutzergruppen Termine mitteilen können.

Zwischenmenschliche Interaktion wird in einer sozial verkümmernden Gesellschaft durch 'modische', zeitgemäße Mittel verstärkt. Durch Technik den Menschen zu unterstützen und ihm das Leben in einer immer schneller werdenden Gesellschaft zu erleichtern, die Herausforderung.

Wie schon Pattie Maes 1994 in *Agents that reduce work and information overload* (Maes (1994)) beschrieben hat, ist unsere Interaktion mit dem Computer heutzutage durch direkte Manipulation gekennzeichnet, bei der der Nutzer die Aktionen selbst anstößt.
Ihr Ansatz ist es, autonome Agenten einzusetzen und den Menschen durch einen persönlichen Assistenten zu unterstützen, welcher mit dem Menschen und anderen Assistenten interagiert und sich dabei derselben Arbeitsumgebung wie sein Mensch bedient.

Diese Arbeit soll einen Teil des Weges beschreiten, der das langfristige Ziel hat, Software in die eigene Körperlichkeit zu integrieren und die Interaktion mit Computern nicht mehr als Interaktion mit Fremdsystemen wahrzunehmen. [2]

[1] Als heterogene Gruppe sind in dieser Arbeit Gruppen bezeichnet, welche nicht einer Organisationseinheit, wie einer Firma, zugehören. Homogene Gruppen hingegen werden aus einem Benutzerkreis gebildet bei dem jeder Benutzer den anderen kennt oder Zugriff auf diesen hat.

[2] Bei Spiegel (2009) findet sich eine interessante Beschreibung dieses Themas.

1.2 Szenario

Einmal in der Woche Rennradfahren: Die Koordination eines solchen Termins ist mit vielseitigen Herausforderungen behaftet.

In einer heterogenen Gruppe, bei der sich nicht alle Teilnehmer persönlich kennen oder gar die persönlichen Termine anvertrauen, müssen die Termine mit viel Kommunikationsaufwand geplant werden. Des Weiteren kann der Kommunikationsaufwand durch Absagen oder eine Terminverschiebung stark steigen.

Dieses Szenario wird maßgeblich durch diese Arbeit führen. Weitere mögliche Szenarien sind unter 5.1 zu finden.

1.3 Gliederung

Zu Beginn der Arbeit wird auf ihre **Zielsetzung** (2) eingegangen sowie eine Abgrenzung gegenüber bestehenden Arbeiten vorgenommen.

Auf die **Grundlagen** (3) wird im folgenden Kapitel eingegangen. Zu den erläuterten Grundlagen gehören Informationen über Agenten (3.1), Wissen über Zeitlogik (3.2.1) sowie gebräuchliche Austauschformate über Kalenderinformationen (3.2.2).

Das darauf folgende Kapitel **Analyse** (4) beginnt mit einem Überblick über Forschungsergebnisse aus dem Bereich der agentenunterstützten Terminkoordination (4.1). Darauf folgt eine kurze Übersicht über kommerziell erhältliche Lösungen zu diesem Thema (4.2). Das Kapitel schließt mit einer Betrachtung der aus den bisherigen Umsetzungen gewonnenen Erkenntnissen ab.

Das Kapitel **Design** (5) beschäftigt sich mit einer genaueren Analyse von möglichen Szenarien (5.1), den daraus resultiernden Use Cases (5.2) und der Modellierung der Anwendung (5.3).

In dem Kapitel **Realisierung** (6) wird auf die verwendeten Technologien (6.1) sowie die Umsetzung der Anwendung eingegangen.

Den **Schluss** (7) bildet ein kurzes Fazit (7.1) sowie ein Ausblick (7.2) auf mögliche nächste Schritte für eine weitere wissenschaftliche Betrachtung des gewählten Themenkomplexes.

2 Zielsetzung

Ziel ist es, eine Software zu entwerfen, welche hilft, bei der Planung von Terminen nicht zusätzlich Zeit aufzuwenden, nein, vielmehr Zeit zu gewinnen. Eine Kalenderplanung, welche die soziale Interaktion zwischen Menschen und dadurch das Miteinander in unserer Gesellschaft verbessert.

Hierfür können Menschen ihre Wünsche und Vorlieben an die Software mitteilen und diese hilft ihnen, einen Zeitplan zu schaffen, welcher möglichst vielen der gegebenen temporalen und nichttemporalen Randbedingungen genügt. Des Weiteren unterstützt die Software den Anwender dabei, diese Randbedingungen festzulegen.

Die Software soll benutzt werden um

- *Jemanden*
- zu einer bestimmten *Zeit*
- an einem *Ort*
- zu einem *Thema*
- sowie einer bestimmbaren *Sichtbarkeit* des Ereignisses

zu finden.

Im Rahmen dieser Masterarbeit soll diese Software prototypisch umgesetzt werden.

Die Planung von Terminen soll von Agenten unterstützt werden. Agenten kann Zugriff auf die private Terminplanung gewährt werden, sodass diese untereinander mögliche Termine aushandeln und dem Anwender Vorschläge unterbreiten können.

Die Interaktion mit dem Anwender soll über eine Webanwendung erfolgen, welche sich moderner Webtechniken (3.3) bedient.

2.1 Anforderungen

In diesem Abschnitt sind zu betrachtende Anforderungen an die anvisierte Anwendung aufgeführt.

2.1.1 funktionale Anforderungen

Terminverwaltung

Die Grundfunktionalität der Anwendung besteht in der Terminverwaltung.

Hierzu gehört die Verwaltung der Termine mit all ihren Eigenschaften wie in Abbildung 5.3 dargestellt.

Einladungen verschicken

Einladungen zu Terminen sollen von dem System über verschiedene, je Kontakt unterschiedliche, Wege verschickt werden können (Siehe auch bei 2.1.2 Erweiterbarkeit).

Kontakte verwalten

Kontakte zu anderen Benutzern des Systems, sowie Kontakte, die lediglich über externe Schnittstellen wie Mail oder SMS integriert sind, müssen innerhalb der Anwendung von dem Anwender verwaltet werden können.

Integration von und in externe Kalender

Kalender, welche in anderen Systemen gepflegt werden, sollen in die Anwendung integriert und von der Anwendung genutzt werden können. Des Weiteren soll der durch die Benutzung der Anwendung entstehende Kalender in andere Kalendersysteme integriert werden können. Dadurch soll es dem Anwender möglich sein, seine bisherigen Gewohnheiten beizubehalten und lediglich ein neues Werkzeug zur Terminplanung zur Hand zu haben.

Automatische Absagen

Falls ein Anwender für bestimmte Termine für eine absehbaren Zeitraum keine Zeit hat, muss dieser dieses im System hinterlegen können.

So können zum Beispiel Informationen über einen Urlaub oder eine Krankheit zu einer automatischen Absage führen.

Gruppierung von Terminen sowie deren Eigenschaften

Die Anwendung soll die Gruppierung von Terminen sowie deren Eigenschaften (siehe Abbildung 5.3) unterstützen.

Diese Gruppen können für vielfältige Erweiterungen gebraucht werden.

- Vereinfachter Zugriff auf Gruppen von Personen oder anderen Termineigenschaften.
- Automatische Generierung von Vorschlägen für Eigenschaften zu einem Termin sobald dem System erste Daten bekannt gemacht werden.
- Gruppierung von Sicherheitseinstellungen oder Einstellungen im Allgemeinen.

Priorisierung

Termine sollen mit einer Priorität versehen werden können. Die Gruppierungsfunktion hilft, die Termine automatisch zu priorisieren, sodass auftretende Konflikte automatisch eskaliert werden können.

Heterogene Benutzergruppen

Die Anwendung soll stark heterogene Benutzergruppen unterstützen.

Benutzer, die untereinander keine Vertrauensbasis haben, sollen bei der gemeinsamen Teilnahme an einem Termin oder bei der gemeinsamen Benutzung eines Doorman (5.3.2) ihre Privatsphäre wahren können. Dennoch sollen sie in der Lage sein, durch die Unterstützung des Systems, einen möglichst für alle passenden Termin zu finden.

Privatsphäre

Der Anwender muss die Möglichkeit haben, die Wahrung seiner Privatsphäre sicherzustellen.

Hierzu müssen dem Anwender Werkzeuge an die Hand gegeben werden, welche ihm eine einfache Übersicht über die Verfügbarkeit seiner Daten geben. Diese Werkzeuge müssen auch ein einfaches Einschränken oder Erweitern der Rechte ermöglichen und hierfür Detail- und Übersichts-Ansichten bereitstellen, welche die Rechte in der entsprechenden Granularität anpassen können.

Randbedingungen für die Kalenderverwaltung

Mögliche Randbedingungen für die Verwaltung eines Kalenders sollten von dem System berücksichtigt werden. Unter anderem:

- Der Terminkalender soll möglichst wenig fragmentiert sein, allzu große zeitliche Lücken zwischen Terminen sind zu vermeiden.
- Die Reisezeiten bei Terminen an verschiedenen Orten sollten minimiert werden.

2.1.2 nichtfunktionale Anforderungen

Einfachheit, Usability

Simple things should be simple, complex things should be possible.

Alan Kay

Die Anwendung soll dem Anwender einen einfachen Einstieg ermöglichen. Die alltäglichen Use-Cases sollen intuitiv erreichbar sein, komplexe Anforderungen an das System sollen aber umsetzbar sein.

Die Grundfunktionalität sollte ein unbedarfter Anwender verstehen und zu benutzen wissen.

Die Interaktion mit dem System sollte 'einfach funktionieren'. Hierzu gehört unter anderem, dass keine unnötigen Daten erhoben werden und keine unnötigen Registrierungsprozeduren erforderlich sind.

Erweiterbarkeit

Erweiterungen sollen leicht in die bestehende Anwendung integriert werden können.

Dieses soll dadurch erreicht werden, dass die Grundfunktionalität von dem Basisset an Agenten zur Verfügung gestellt wird, weitere Agenten aber jederzeit in das System eingebunden werden können, wodurch beliebig komplexe Anforderungen integriert werden können.

Prototyp[1]-Agenten oder einfach durch den Anwender programmierbare Agenten sind hierfür eine gute Erweiterungsmöglichkeit.

Integration von Personen ohne Internet

Personen ohne Internetzugang oder ohne Computer sollen auch eine Möglichkeit haben über bevorstehende Termine informiert zu werden sowie dem Veranstalter ihre Zu- oder Absage zukommen zu lassen.

Offline Client

Die Anwendung muss auch offline verfügbar sein und Änderungen entgegennehmen. Diese Änderungen müssen dann, sobald wieder eine Verbindung zum Internet besteht, synchronisiert werden.

Performance

Als Anforderungen an die Performance der Anwendung zählen:

- Latenz - die Reaktion auf eine Benutzereingabe soll möglichst schnell erfolgen. Ist keine Anfrage zum Server notwendig und können die Veränderungen auch asynchron übermittelt werden, ist dem Anwender das Ergebnis sofort zu präsentieren.
- Skalierbarkeit - die Anwendung sollte auch bei steigenden Benutzerzahlen performant bleiben. Hierzu gehört auch die Wahl einer Architektur, welche entsprechend skalierungsfähig ist.

[1] Analog zu Gamma u. a. (1995) Seite 117ff

Privatsphäre

Bei den Termindaten der Anwender handelt es sich um hochsensible Informationen. Wegen der automatischen Verarbeitung und Weitergabe dieser Informationen an vertrauenswürdige Kontakte ist es besonders wichtig, dass das System diese Daten vertrauenswürdig Verarbeitet und schützt.

Integration bestehender Schnittstellen

Die Diversität der Systeme im Internet ist Segen und Fluch zugleich. Um als Anwendung in dieser Artenvielfalt zu überleben, bietet sich eine enge Intergation mit bestehenden Standards an. Hierunter fallen unter anderem eine Integration mit sozialen Netzwerken wie Facebook[2], aber auch eine Integration von OpenSocial[3], OpenId[4] oder auch die Nutzung von PubSubhubbub[5].

Weitere nichtfunktionale Anforderungen

Folgende Anforderungen sollen im Rahmen dieser Arbeit nicht genau betrachtet werden, sind aber bei einer Umsetzung nicht außer Acht zu lassen:

- Sicherheit
- Wartbarkeit
- Rechtliche Rahmenbedingungen
- Internationalisierung
- Kulturelle oder politische Anforderungen
- Integrierbarkeit in bestehende Angebote

[2] http://www.facebook.com/
[3] http://code.google.com/apis/opensocial/, API zur Interaktion zwischen Sozialen Netzen
[4] Ein System zur Authentifizierung über unabhängige 'identity provider' http://openid.net/
[5] http://code.google.com/p/pubsubhubbub/

2.2 Abgrenzung

Es gibt schon viele wissenschaftliche Ansätze zur Planung von Meetings (4.1) und auch kommerzielle Anwendungen zur Koordination von Kalendern (4.2). Jedoch zielen diese entweder auf stark homogene Gruppen oder bieten kaum Unterstützung bei einer automatischen Terminfindung.

Mein Ziel ist es, eine Anwendung zu entwerfen, welche genau diese Lücke füllt und stark heterogenen, gemischten Gruppen proaktive Unterstützung bei der Terminfindung bietet.

2.3 Umsetzung

Für die prototypische Umsetzung sollen zunächst lediglich folgende Anforderungen umgesetzt werden:

funktionale Anforderungen

- Terminverwaltung
- Einladungen verschicken

nichtfunktionale Anforderungen

- heterogene Benutzergruppen
- Erweiterbarkeit
- Performance

3 Grundlagen

3.1 Agenten

An Multi-Agent-System enhances overall system performance, specifically along the dimensions of computational efficiency, reliability, extensibility, robustness, maintainability, responsiveness, flexibility, and reuse. (CMU)

Agenten[1] bieten Unterstützung bei lose gekoppelter organisationsübergreifender Teamarbeit.

Hierbei interagieren die Agenten sowohl mit dem Menschen, untereinander oder mit verschiedenen Diensten.

Die Interaktion erfolgt dabei zwischen

- Agent - Agent
- Agent - Service
- Agent - Human

Ein wichtiger Aspekt hierbei ist das Vertrauen zwischen den jeweiligen Interaktionspartnern.

Eine mögliche Kategorisierung von Agenten ist, diese danach zu unterteilen, ob sie proaktiv oder reaktiv agieren. Proaktive Agenten lösen von selbst Aktionen aus, reaktive Agenten tun dies lediglich als Reaktion auf Sensoreingaben.

In dieser Arbeit wird das *proaktive Verhalten der Anwendung*, abweichend hierzu, als ein Verhalten bezeichnet, welches nicht direkt von einem Menschen angestoßen wurde, sondern auf der Einstellung eines Agenten beruht. Zum Beispiel die automatische Absage eines Termins während einer Krankheit oder die automatische Generierung eines Alternativvorschlags für einen Termin.

[1] Eine ausführlichere Beschreibung von Agenten ist bei Tennstedt (2008) zu finden.

3.1.1 Agentennetzwerke

Verschiedene Architekturen für die Verteilung von Agenten werden unter anderem von Zhang u. a. (2007) und Zhong u. a. (2003) untersucht. Die erste Arbeit zu diesem Thema war *The Agent Network Architecture (ANA)* (Maes (1991)).

In *From Computer Networks to Agent Networks* (Zhong u. a. (2003)) werden Ideen für eine Architektur zur Realisierung einer *global distributed computing architecure* vorgestellt. Zhang u. a. (2007) gehen dabei besonders auf Aspekte wie die Nebenläufigkeitskontrolle, die Mobilität und die Intelligenz der Agenten ein.

In Barabasi und Albert (1999) werden Agentennetzwerke auf drei charakteristische Ausprägungen (von Netzwerken) hin unterschieden (Burka (2008)):

- High Clustering - Netzwerke, welche viele Cluster von miteinander verknüpften Knoten bilden. Die Cluster untereinander sind jedoch eher lose oder gar nicht miteinander gekoppelt. In der in dieser Arbeit entworfenen Anwendung müssten die ersten sich bildenden Beziehungen zwischen Anwendern diese Eigenschaft aufweisen.
- Scale Free - Scale Free Netzwerke entsprechen einer Verteilung nach einem Potenzgesetz. So gibt es einige Knoten mit sehr vielen Verbindungen und viele Knoten, welche diese nur über sehr weite Wege erreichen.
- Small World - Das Small World Phänomen von Karinthy (1929) stellte die von Milgram (1967) bestätigte Theorie auf, dass zwei beliebige Personen auf dieser Welt über eine Kette von maximal sechs Bekannten untereinander verbunden sind. Small World Netzwerke sind ähnlich aufgebaut wie die Beziehungen zwischen den Menschen unserer Welt.

3.1.2 Agentenkommunikation

Zhang und Liu (2006) untersuchen die Netzwerkbildung von Agenten in einem Service orientierten Netzwerk. Hierfür wurden die Eigenschaften sozialer Netzwerke auf Agentennetzwerke übertragen und anhand folgender Metriken die Beziehungen zu anderen Agenten bewertet.

- Fähigkeiten: Die Fähigkeit, bestimmte Aufgaben unter den gegebenen Voraussetzungen zu erfüllen.
- Gemeinschaftlicher Profit: Der Profit, den die Agenten bei bisherigen Transaktionen gemacht haben
- Kooperationsverhalten: Die Bereitschaft, für einen nur geringen Profit zu kooperieren.

- Partnerschaftsgrad: Die Häufigkeit, mit der zwei Agenten miteinander kooperieren.

Um ein möglichst performantes Agentennetzwerk zu erhalten, wurden verschiedene Strategien für die Partnerwahl untersucht:

- Strategie I: Die Agenten mit dem größten Partnerschaftsgrad werden erwählt.
- Strategie II: Ein Partner wird anhand des maximalen gemeinschaftlichen Profits ermittelt.
- Strategie III: Es werden die kooperativsten Partner für die Zusammenarbeit gewählt.
- Strategie IV: Ein Agent wählt einen Agenten, der möglichst viele Partner hat.

Die Untersuchungen haben ergeben, dass die Auswahl von kooperativen Partnern mit einem großen gemeinschaftlichen Profit die beste Performance bietet.

Solche Netzwerke, die Menschen mit einschließen, hybride Mensch-Agent Netzwerke, wurden unter anderem von Sycara und Sukthankar (2006) untersucht.

3.1.3 Netzwerkbildung durch Agenten

Netzwerkbildung durch Agenten bezieht sich auf die Bildung eines Netzwerkes, für welches die Agenten selbst ohne Bedeutung sind und in dem diese nur die vermittelnde Rolle spielen. Hierbei können die Agenten für eine gute Vermittlung der Daten Sorge tragen oder aber selbst die Daten mitführen und an ihren Bestimmungsort transportieren.

Um Skalierungsprobleme bisheriger *ad hoc mobile networks* zu umgehen, stellt Denko (2003) eine Architektur vor, welche Agenten verwendet, um Pakete zu routen.

Paysakhov u. a. (2004) zeigen, wie man unter Verwendung von Agenten die Sicherheit und Zuverlässigkeit in einem *ad hoc mobile network* erhöhen kann.

3.1.4 Vertrauen und Sicherheit

In einer Infrastruktur, in der viele Agenten von vielen Anwendern miteinander kommunizieren, sind Aspekte wie Sicherheit und Vertrauen von großer Wichtigkeit.
Dafür gibt es verschiedene Ansätze, um die Vertrauenswürdigkeit eines Gegenübers zu bewerten (Burka (2008)).

- Web of trust
- Social trust

- Trust propagation
- Friend of a Friend

Es ist wichtig, dass die Agenten eines Menschen wissen, welche der ihnen anvertrauten Informationen über ihren Meister sie als private und welche als öffentliche Information behandeln sollen. Genauso ist es wichtig, dass die Agenten berücksichtigen, welche Teilinformationen für die Weitergabe von Informationen wirklich wichtig sind.

3.1.5 Advocate Agents

In Ketter u. a. (2008) wird eine Architektur für Advocate Agents vorgestellt, die die Vorlieben 'ihres' Anwenders kennen lernen sollen und dann helfen sollen, Entscheidungen zu treffen.

Ketter u. a. (2008) wollen Agenten schaffen, die den Menschen nicht nur als eine statistische Variable sehen, die aufgrund ihres Alters, des Einkommens, ihres Wohnorts, anderen Menschen ähnelt und dadurch ihre Entscheidungen begründen, dass andere Menschen mit ähnlichen Voraussetzungen gleiche Entscheidungen getroffen haben.

Es wird vielmehr beschrieben, wie eine Architektur aussehen kann, die den Menschen beobachtet und anhand seiner persönlichen Aktivitäten ein Benutzermodell (Rosseburg (2009)) aufbaut um 'seinem' Menschen in Zukunft besser beiseite stehen zu können.

Die vorgestellte Architektur basiert auf einem Messaging Bus, an den Module angebunden werden (siehe Abbildung 3.1). Folgende Module bilden die Basis dieser Architektur:

- *Information Retrieval Module* zur Gewinnung von Informationen.
- *Knowledge Representation and Inference Module* zur Verarbeitung von vorhanden Roh-Informationen zu semantisch angereicherten Informationen.
- *Agent Collaboration Module* für die Zusammenarbeit mit anderen Agenten.
- *Recommendation Module* mit dem der Advocate seinem Meister Vorschläge oder neue Informationen präsentieren kann.
- *High Performance Processing Module*, um aufwändige Arbeiten zu erledigen oder diese zum Beispiel in die *Cloud* auszulagern.
- *Security* welches sowohl für Fragen der Sicherheit wie Sicherheitsdomänen, Digitale Signaturen, Verschlüsselung zuständig ist, als auch die Agenten um den Aspekt Sicherheit erweitert.

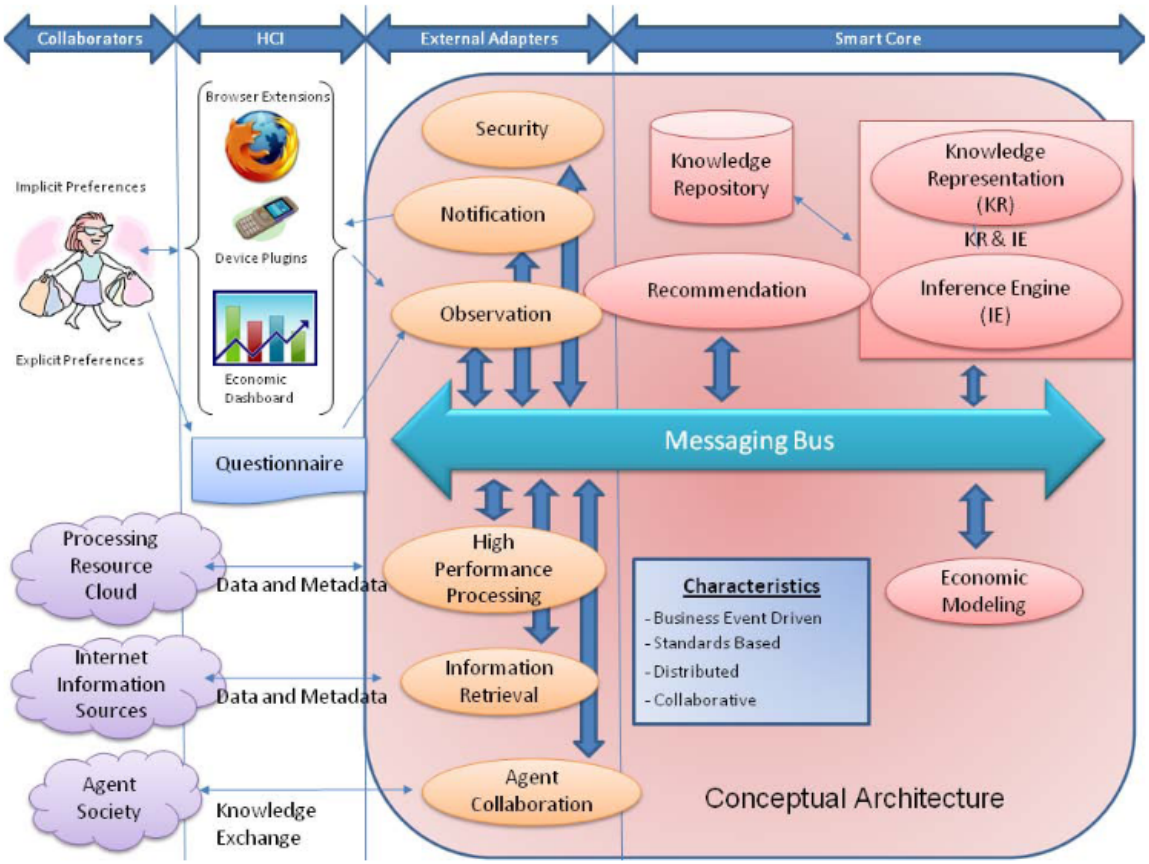

Abbildung 3.1: Die Architektur von Ketter u. a. (2008) macht sich die Vorteile eines ESB (siehe hierzu Josuttis (2007)) zunutze.

- *Economic dashboards*, um dem Benutzer die Möglichkeit zu geben, die Entscheidungen des Agenten zu bewerten, um damit zukünftige Vorschläge so zu lenken, dass diese den Benutzer zufriedener stellen.

Die Infrastruktur für die Advocate Agents besteht aus einer vereinheitlichten Kommunikationsschicht, Protokoll Standards für nahtlosen Wissenstransfer, einer Sammlung an Transformationsregeln um bekannte Formate für die Agenten verständlich zu machen, sowie einer Datenverarbeitungskomponente.

3.2 Zeit

3.2.1 Zeitlogik

Um zeitliche Verläufe abbilden zu können, bedarf es einer Logik, welche Operationen auf zeitlichen Relationen ermöglicht.

Allen (1983) hat hierfür ein grundlegendes Modell geschaffen, welches den Eigenarten von Zeitlogik Rechnung trägt. Die beschriebene Zeitlogik basiert auf vier grundlegenden Anforderungen.

- Die Darstellung soll signifikante Ungenauigkeit erlauben. Temporales Wissen kann unabhängig von einem absoluten Datum sein (A findet vor B statt).
- Die Darstellung soll Unsicherheit von Informationen ermöglichen, falls die exakte Beziehung zwischen zwei Vorgängen nicht bekannt ist, aber einige Einschränkungen dieser Beziehung bekannt sind.
- Die Darstellung soll verschiedene Auflösungen der Genauigkeit der Zeitdarstellung ermöglichen.
- Das Modell unterstützt die Persistenz von Daten. Einmal ergründetes Wissen bleibt gültig.

Um diese Anforderungen auf ein Zeitmodell abzubilden, benutzt Allen Zeitintervalle sowie deren Relationen zueinander. Hierfür definiert er 7 Basisrelationen sowie deren Inverse.

- X before Y - Y endet, bevor das X beginnt.
- X meets Y - Der Endzeitpunkt von X ist der Startzeitpunkt von Y.
- X overlaps Y - X beginnt vor Y und endet nach dem Beginn, aber vor dem Ende von Y.
- X during Y - X findet vollständig während Y statt.
- X starts Y - Zwei Intervalle starten gleichzeitig, X endet jedoch vor Y.
- X finishes Y - Zwei Intervalle enden gleichzeitig, X beginnt jedoch nach Y.
- X equals Y - Beide Intervalle starten und enden gleichzeitig, sind sozusagen gleich.

Da die Berechnung von Allens Zeitlogik NP-vollständig ist, gibt es verschiedene Modelle, die versuchen, die Komplexität auf ein berechenbares Maß zu beschränken, dabei aber die Mächtigkeit nur so wenig wie möglich einzuschränken.

Nebel (1997) hat mit dem ORD-Horn Subset eine Möglichkeit für eine eingeschränkte Zeitlogik geliefert, welche bei dem Zielkonflikt zwischen Komplexität und Mächtigkeit die Berechenbarkeit in den Fokus nimmt.

3.2.2 Kalender

Kalender sind seit langem Teil von Softwareanwendung und seit vielen Jahren standardisiert.

- RFC 2445[2](Dawson und Stenerson (1998)) definiert ein einheitliches Austauschformat für die Darstellung von Terminen, To-dos, Journaleinträge sowie Verfügbarkeitsinformationen.
- RFC 2446[3] (Silverberg u. a. (1998)) definiert die Interaktionen zwischen Kalendern.
- RFC 2447 (Dawson u. a. (1998)) definiert den Austausch von Kalenderinformationen per Mail.

Eine etwas ausführlichere Übersicht über diese Standards ist zu finden bei Dawson (1997).

Durch die Verwendung dieser Standards kann eine Softwareanwendung auf eine große Menge an kompatibler Software zurückgreifen und eine nahezu reibungslose Interoperabilität sicherstellen. Dazu gehört auch der Austausch von Zeitanfragen sowie deren Beantwortung.

[2] inzwischen ersetzt durch RFC 5545 (Desruisseaux (2009))
[3] inzwischen ersetzt durch RFC 5546 (Daboo (2009))

3.3 Technologien

3.3.1 NoSQL

NoSQL Datenbanken verfolgen das Ziel, gut skalierbare, verteilte Datenbanken für große Datenmengen zu sein. Dabei wird auf viele Eigenschaften von relationalen Datenbanken verzichtet und ein meist schemaloser Ansatz gewählt.

Eric Brewer hat zu diesem Thema das CAP Theorem aufgestellt (Brewer (2000)), welches besagt, dass man nicht drei der folgenden Eigenschaften für eine Web-Anwendung vollständig erfüllen kann.

- *Consistency* Änderungen durch mehrere Operationen werden atomar durchgeführt.
- *Availability* Die Anwendung ist verfügbar.
- *Partition tolerance* Die Anwendung funktioniert weiter, falls die sie ausführenden Knoten partitioniert werden.

In Gilbert und Lynch (2002) wurde dieses Theorem untersucht und gezeigt, dass maximal zwei dieser drei Eigenschaften vollständig erfüllt werden können.

Da die von ACID (Haerder und Reuter (1983)) geforderten Eigenschaften (*atomicity, consistency, isolation, durability*) sehr strenge Randbedingungen festlegen, sind die diametral entgegengesetzten Eigenschaften BASE (*basically available, soft state, eventually consistent*) (Pritchett (2008)) vorgestellt worden. Diese Eigenschaften lassen es zu, dass eine Datenbank nicht zu jedem Zeitpunkt einen definierten Konsistenzzustand hat, sondern, dass die Konsistenz sich ständig im Fluss befindet. Diese Eigenschaft ist in vielen Anwendungen akzeptabel[4] und ermöglicht eine viel höhere Skalierbarkeit.

Einhergehend mit anderen Eigenschaften von Persistenz (*eventually consistent*) wird eine andere Speicherung von Daten erforderlich, um eine gute horizontale Partitionierung[5] zu ermöglichen. So werden Daten denormalisiert und passend für die Anwendungsfälle gespeichert. Dadurch ist ein Zusammenführen (*JOIN*) von Daten nicht erforderlich und für einen Anwendungsfall wird im Idealfall ein Datensatz aus einer Partition, von einem Server, geholt.

[4] Ein neuer Tweet auf Twitter wird zum Beispiel nicht sofort und gleichzeitig bei allen Followern angezeigt.
[5] Siehe auch Hoff (2009).

3.3.2 HTML5

Websockets[6] sind eine HTML5-Erweiterung welche eine bi-direktionale Verbindung zwischen einem (Web-)Server und einem Browser ermöglichen. Um ohne Verwendung von Websockets Server-Push (Kanitkar und Delis (1998)) zu ermöglichen, müssen Browser bisher regelmäßig den Server anfragen oder eine HTTP-Verbindung offen halten[7].

Webstorage[8] ist eine Möglichkeit per JavaScript in einem Browser auf dem Client-Rechner Daten zu speichern. Diese Form der Speicherung ist die Erweiterung von Cookies (RFC 2965 Kristol und Montulli (2000)) und ermöglicht neue Anwendungsfälle, zum Beispiel das Speichern von Daten auf dem Client nur für den Context einer Session.

Offline Web applications[9] ermöglichen eine offline verfügbare Webapplikation (2.1.2). Durch die Verwendung von Webstorage können Änderungen in dieser Applikation gespeichert werden und, sobald die Anwendung wieder eine Verbindung zum Internet herstellen kann, synchronisiert werden.

3.3.3 GWT

GWT[10], das Google Widget Toolkit, ist ein Werkzeugkasten welcher die Entwicklung von komplexen Browser-basierten Anwendungen erleichert.
GWT übersetzt in Java programmierte Anwendung in Javascript, welches darauf im Browser ausgeführt werden kann.
Durch die Verwendung von GWT kann eine Webanwendung einmal programmierte Logik sowohl im Frontend als auch im Backend nutzen.

[6] http://dev.w3.org/html5/websockets/
[7] http://infrequently.org/2006/03/comet-low-latency-data-for-the-browser/
[8] http://dev.w3.org/html5/webstorage/
[9] http://www.w3.org/TR/html5/offline.html
[10] http://code.google.com/intl/de-DE/webtoolkit/

4 Analyse

Dieses Kapitel beschäftigt sich mit der Analyse vorausgehender Arbeiten, sowie den daraus folgenden Implikationen.

Zunächst wird ein Überblick über vorhandene Forschungsarbeiten (4.1) im Bereich der Terminplanung mit Agenten gegeben. Darauf folgt eine Übersicht über eine Auswahl an kommerziellen Kalenderlösungen [1].

Darauf folgt die Einordnung sowohl der vorgestellten Lösungen, als auch der angestrebten Lösung.

Bisherige kommerzielle Umsetzungen (4.2) für die Unterstützung von Terminplanung haben nur wenige der Forschungsergebnisse aufgegriffen. Diese gewonnenen Erkenntnisse und Akzeptanzkriterien dienen als Grundlage für die folgenden Kapitel Design (5) und Realisierung (6).

4.1 Bisherige Forschungsarbeiten

Im Folgenden werden Forschungsprojekte, die sich damit beschäftigt haben, unter Verwendung von Agententechnologie die Terminkoordination von geschlossenen Benutzergruppen zu verbessern und den Anwendern den Büroalltag zu vereinfachen, vorgestellt.

4.1.1 Collaborator

Collaborator (Bergenti u. a. (2004)) ist eine webbasierte Groupware, welche durch die verwendeten Technologien die Flexibiltät von Interaktion zwischen Menschen fördert. In Collaborator hat jeder Mensch persönliche Agenten, die dem Anwender helfen, die eigenen Kalender zu verwalten. Des Weiteren können die persönlichen Agenten im Sinne des Anwenders Termine aushandeln, Termine klassifizieren sowie die Vorlieben des Anwenders lernen.

[1] Eine größere Auswahl an verfügbaren Kalenderlösungen findet sich unter http://www.mozilla.org/projects/calendar/links.html

Um zwischen den Agenten zu vermitteln, gibt es sogenannte *Session Manager Agents*. Diese Agenten sind dafür zuständig, eine Liste an Teilnehmern für einen Termin zu erstellen sowie einen für die Teilnehmer passenden Termin zu finden. Im Falle einer Verschiebung oder Absage übernimmt der *Session Manager Agents* die Aufgabe, alle Teilnehmer hierüber zu infomieren.

Für die Umsetzung von Collaborator wurde JADE[2] als Agentenframework eingesetzt. JADE wurde unter Verwendung von BlueJade in einem JBoss Application Server eingesetzt. In dieser Umgebung interagieren Personal Agents der Endanwender mit Session Manager Agents. Personal Agents lassen sich über eine Webanwendung vom Anwender verwalten. Die Terminkoordination wird an die Sesssion Manager Agents deligiert, welche sich mit anderen Session Manager Agents abstimmen.

4.1.2 MATT

MATT (Grosso u. a. (2005)) ist ein Groupware-Calender-System, welches die Vorteile von agentenorientierter Softwareentwicklung nutzt. MATT basiert auf einem hierarchischem Kalendermodell, welches sozialen Strukturen sowie Prioritäten von Terminen und Personen untereinander abbildet.

Jedem Anwender wird ein *Personal Assistant Agent* zugeteilt, welcher den Kalender des Anwenders betreut, sowie den Anwender an Termine errinnert. Weitere Agenten mit ihren Aufgaben sind:

- der *Communicator* ist für die Kommunikation zwischen den Agenten zuständig
- der *Arranger* findet passende Termine
- der *Manager* organisiert und steuert die *Arranger*
- der *Role Manager* bestimmt die Wichtigkeit eines Termins oder einer Person
- der *Reminder* errinnert an bevorstehende Termine
- der *Synchronizer* synchronisiert Offline Clients

Die Clients greifen per Webservices auf den Server zu. Es gibt einerseits Online-Clients, die nur mit Zugriff auf den Server funktionieren als auch Offline-Clients, sogenannte Smart Clients, welche eine leichtgewichtige Version des persönlichen Assistenten auf dem Server darstellen, die lediglich um die Funktion beschnitten sind, Verhandlungen durchzuführen. Die Smart Clients verfügen über einen Cache, der die Daten vom Server zwischenspeichert und so auch Offline verfügbar hat.

[2] http://jade.tilab.com/

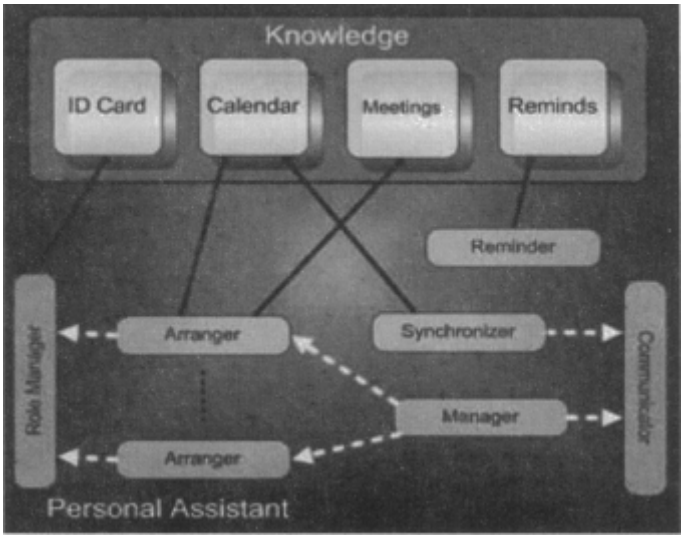

Abbildung 4.1: Die Architektur von MATT

MATT unterstützt bei der Terminfindung durch eine automatische Auswertung der Verfügbarkeiten unter Berücksichtigung von deren Prioritäten.

Es wurde folgendes festgestellt:

- Die standardisierte Kommunikationslösung in Form von Webservices vereinfacht den Zugriff auf das System.
- Offline Clients bieten eine Integration von schlecht angebundenen Teilnehmern.
- Die agentenorientierte Softwarentwicklung hat bei der Entwicklung viele Vorteile gebracht. Besonders die Fähigkeit zu Kooperation und Verhandlung hat in Konkurrenzsituationen zu guten Lösungen geführt.

4.1.3 EventMinder

EventMinder (Smith und Lieberman (2009)) ist ein Kalendersystem, welches dem Anwender ermöglicht, natürlichsprachlich Ziele für ein Termin festzulegen. EventMinder extrapoliert dann aus vorhandenem Wissen fehlende Fakten und bestimmt Abhängigkeiten zwischen Terminen und ist in der Lage, alternative Terminvorschläge zu unterbreiten.

4.1.4 PTIME

PTIME (Personalized Time Management Agent) (Berry u. a. (2006)) ist ein personalisierter Zeit-Management-Agent, welcher an der SRI International[3] in Californien als Teil einer größeren mehrjährigen Initiative, persönliche Assistenten zu realisieren, entwickelt wurde. PTIME ist ein Teil von PAL (*Personal Assistant that Learns*) im Rahmen von CALO[4].

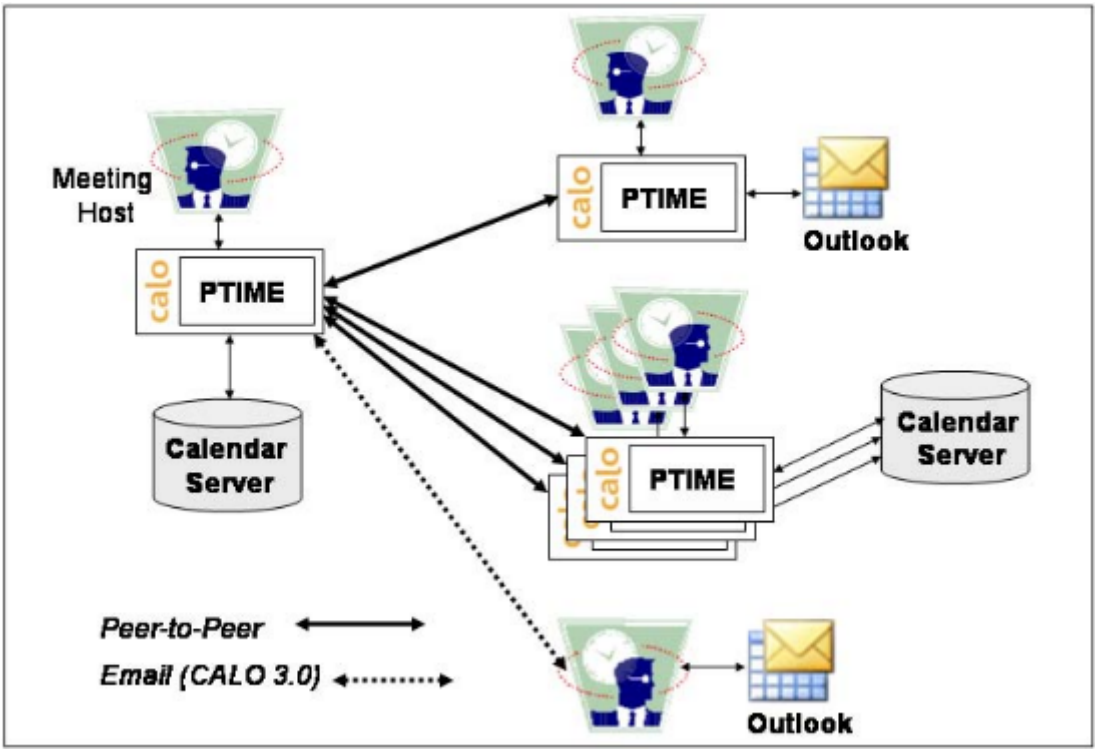

Abbildung 4.2: Das Kommunikationsmodell der PTIME-Agenten

Jeder Anwender bekommt einen Agenten zur Seite gestellt, der mit anderen Agenten Meetings verabreden kann. Dabei informiert der Agent seinen Anwender über bevorstehende Meetings und lernt dessen Preferenzen. Die Lösung ist voll integriert mit kommerziell erhältlichen Kalenderlösungen.

Berry u. a. (2006) arbeiten insbesondere zwei Faktoren als Herausforderung für die Umsetzung einer Zeitplanung heraus.

- Zeitplanung ist eine sehr persönliche Angelegenheit, die von Person zu Person unterschiedlich gehandhabt wird.

[3] Ein privates Forschungsunternehmen, ehemals Stanford Research Institute (http://www.sri.com/).
[4] *Cognitive Assistant that Learns and Organizes (http://caloproject.sri.com/)*

- Anwender wollen die Kontrolle über ihre Methode der Zeitplanung nicht aufgeben.

Die Arbeit an PTIME hat zu folgenden Erkenntnissen geführt:

- Zeitplanung ist nur ein Teil einer persönlichen Zeit-Management-Lösung.
- Verschiedene Kommunikationswege sind notwendig, um erfolgreich Termine zu vereinbaren.
- Vertrauen in den persönlichen Assistenten muss durch diesen erarbeitet werden.
- Der Assistent sollte unterstützend arbeiten und nicht versuchen, den Anwender zu ersetzen.

4.1.5 RhaiCAL

RhaiCAL (Rich Human-Agent Interaction for a Calender Scheduling Agent) (Faulring und Myers (2005)) ist ein System, welches an der Carnegie Mellon University als Teil von RADAR[5] entwickelt wurde.

Ziel von RhaiCAL ist es, neue Wege zu finden, dem Anwender eine natürlichsprachliche Eingabe seiner Wünsche zu ermöglichen und eine neuartige Darstellungsform von Vorschlägen durch den Agenten zu evaluieren, mit dem Ziel, ein hochqualitatives Benutzerinterface für die Terminplanung zu schaffen. Dabei sollen die Vorlieben und Wünsche der Anwender berücksichtigt werden.

Faulring und Myers (2005) sind dabei zu folgenden Erkenntnissen gelangt:

- Falls eine Interaktion mit einem Agenten nicht vollendet wurde, ist der Wunsch des Anwenders weiterhin unklar. Die Anwender wünschen sich eine Erklärung für die Entscheidungen des Agenten.
- Es wurde ein Weg gefunden, die Interaktion mit Agenten einfach zu gestalten und Fehler bei der Interpretation der natürlichsprachlichen Eingabe der Nutzer zu korrigieren und dabei dem Agenten Rückmeldung über die gemachten Fehler zu geben.

[5] Reflective Agent with Distributed Adaptive Reasoning http://www.radar.cs.cmu.edu/

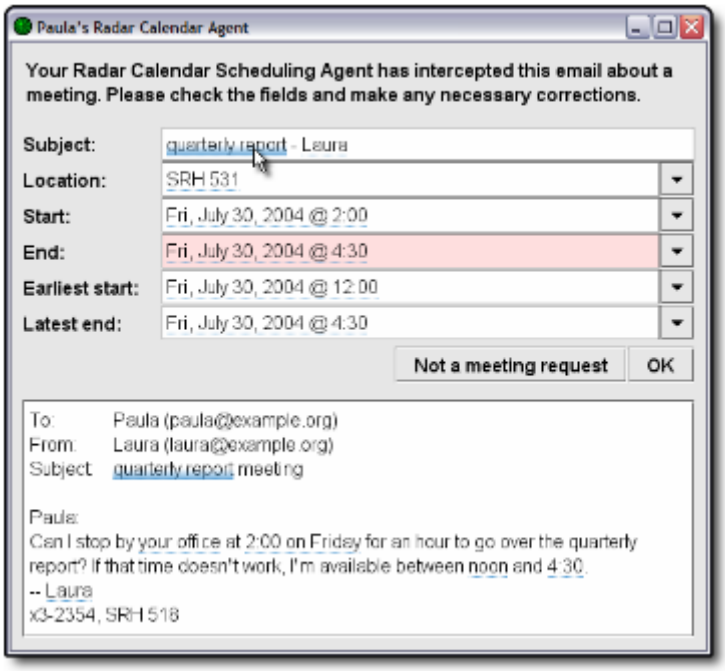

Abbildung 4.3: Der Klärungsdialog von RhaiCAL durch welchen die Interpretation des Agenten bestätigt sowie korrigiert werden kann

4.2 Kommerzielle Lösungen

Kommerzielle Lösungen lassen sich in drei Felder unterteilen:

- reine Kalenderanwendungen
- Groupware
- Social Kalender

Social Kalender wie Plancast[6] oder Meetup[7] bieten eine enge Integration in Social Networking Sites wie Facebook und Twitter, sind aber im Endeffekt bisher nicht mehr als eine Mailingliste für Termine.

Groupware Kalender sind etwa Sogo[8] oder Open Xchange[9]. Diese bieten eine erweiterte Terminsuche, sind aber für geschlossene Benutzergruppen (wie zum Beispiel in Firmen) gedacht.

[6] http://plancast.com
[7] http://www.meetup.com
[8] http://www.sogo.nu
[9] http://www.open-xchange.com

Einfache Kalender wie der Google Calendar oder der Yahoo Calender bieten lediglich das Verschicken von Terminen, die Absage, die Annahme sowie eine Übersicht über bevorstehende Termine.

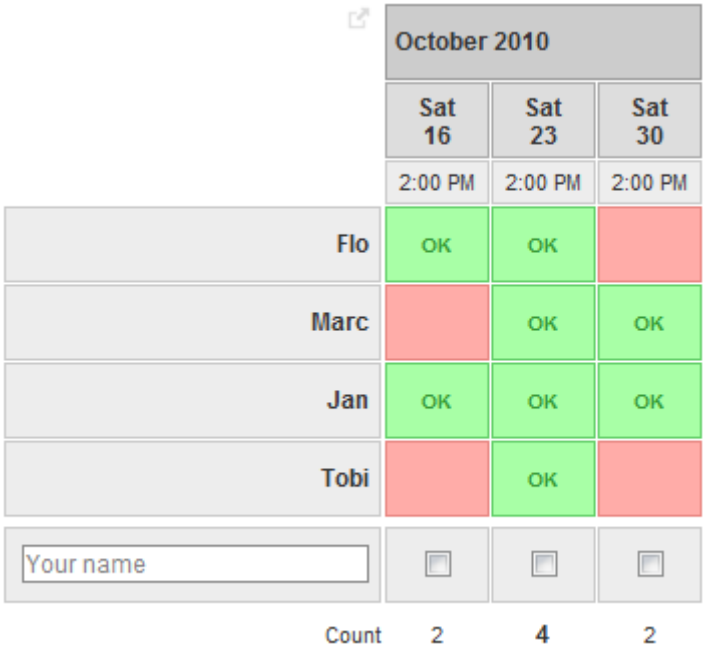

Abbildung 4.4: Terminabstimmung mit Doodle

Des Weiteren gibt es noch einfache Lösungen wie Doodle[10] (siehe Abbildung 4.4). Doodle bietet heterogenen Gruppen eine Plattform um Abstimmungen vorzunehmen. Diese Abstimmungen können unter anderem speziell für Termine gemacht werden. Damit bietet Doodle heterogenen Gruppen eine einfache Plattform, um gemeinsam Termine zu finden.

4.3 Einordnung bisheriger Arbeiten

In Abbildung 4.5 ordne ich die bisher vorgestellten Lösungen nach zwei Kriterien ein.

- Homogenität der Benutzergruppe
- Proaktivität der Anwendung

[10] http://www.doodle.com/

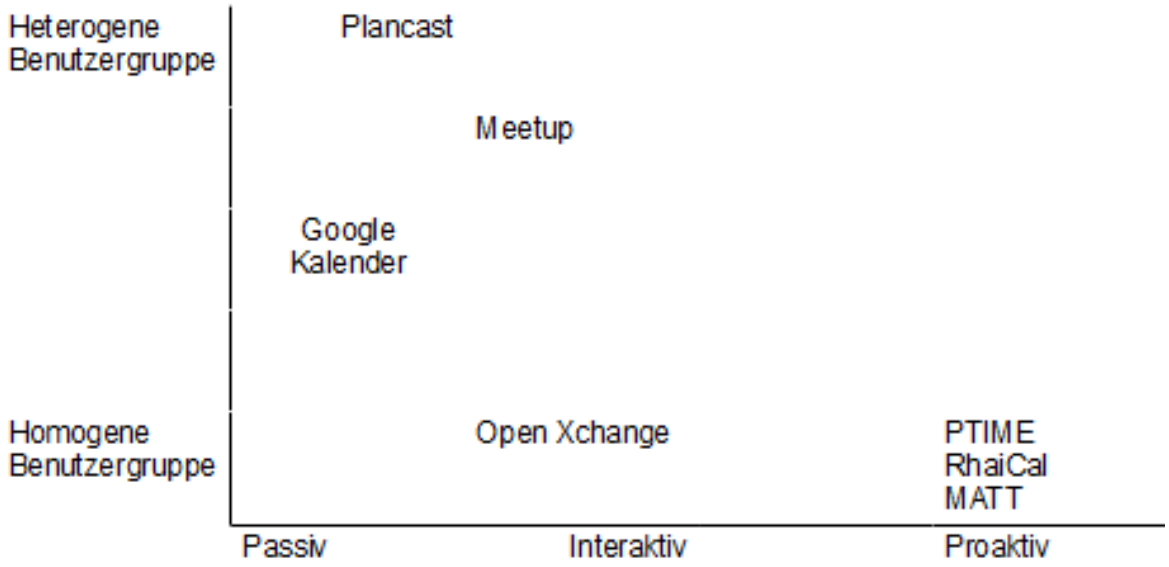

Abbildung 4.5: Einordnung bisheriger Anwendungen

Homogenität

Als Zielgruppe für homogene Benutzergruppen sehe ich nahezu sämtliche bisherigen Groupwarelösungen als auch die vorgestellten Forschungsarbeiten. Für stark unterschiedliche Benutzergruppen sind Lösungen wie Plancast oder Meetup gedacht.

Proaktivität

Verfügbare Anwendungen bieten bisher keine Proaktivität, wenn doch, beschränken sie sich auf Dinge wie Terminzeitsuche oder automatisches Benachrichtigen bei neuen Terminen. In der Forschung gibt es jedoch viele weitreichende Ansätze, um das System Proaktiv zu gestalten und dem Anwender mehr zu helfen.

Fazit

Proaktive Anwendungen für heterogene Benutzergruppen aber gibt es nicht. Daher wird in dieser Arbeit genau diese Lücke betrachtet.

4.4 Akzeptanzkriterien

Aus den untersuchten Arbeiten ergeben sich Schlüsse, die genutzt und umgesetzt werden wollen. Diese Erfahrungen helfen die Koordination von Terminen unter Zuhilfenahme von Agenten zu realisieren.

- Unterstützung anstelle von Entmündigung, aber die Delegation automatisierbarer Aufgaben ermöglichen und anbieten.
- Viele Kommunikationskanäle müssen angeboten werden.
- Die Privatsphäre der Nutzer muss respektiert werden.
- Die Terminplanungsgewohnheiten der Nutzer müssen respektiert werden, der Nutzer muss nicht neue Wege der Terminplanung lernen, sondern wird in seiner Terminplanung unterstützt.
- Die Lösung muss sowohl Offline, als auch Online verfügbar sein.
- Bestehende Standards sollten genutzt werden, sowohl für die Integration von Fremdsoftware, als auch für Interaktionsmuster.

In den Kapiteln 5 (Design) und 6 (Realisierung) wird gezeigt, wie diese Erkenntnisse genutzt werden können.

4.4.1 Motivation der Benutzer

Der Spieltrieb der Anwender kann genutzt werden, um die Interaktion mit dem System zu fördern und um den Anwender zu mehr Benutzung des Systems zu ermutigen.

Hierfür sind verschiedene Belohnungssysteme wie Achievements oder Badges und Statistiken vorgesehen.

Für die Ermittlung der Statistiken oder Badges kann ein Anwender die jeweiligen dafür zuständigen Agenten an seinen persönlichen Benutzeragenten binden.

4.5 Komplexität

In diesem Abschnitt erfolgt eine Betrachtung möglicher auftretender Komplexität bei der Berechnung von möglichen Terminen als auch bei der Benutzung der Software.

Die technische Komplexität der Kommunikation bei einer Terminabsprache zwischen vielen Beteiligten muss begrenzt werden.

4.5.1 Terminfindung

Aus Sicht des Anwenders

Die durch die Verwendung der Anwendung entstehende Komplexität soll nicht größer sein als die Komplexität, einen Termin ohne die Anwendung zu planen. Dafür müssen die häufig verwendeten Use Cases sehr einfach zu erreichen sein. In der fertig gestellten Anwendung muss dann eine Analyse der Usability erfolgen, um diesen Punkt abschließend zu ergründen.

Aus Sicht der Anwendung

Die Berechnung von möglichen Terminen kann beliebig komplex werden.

Wenn Termine zwischen vielen, miteinander nicht vertrauten Personen gefunden werden müssen, welche dazu noch über unterschiedliche Doorman koordiniert werden, ist mit den vorgestellten Konzepten (5) kein zentraler Zugriff auf alle vorhandenen Daten möglich.

Wenn dazu noch die unter 3.2.1 vorgestellten Möglichkeiten in Betracht gezogen würden, würde die Komplexität weiter steigen.

Bei einer Beschränkung auf einfaches Aushandeln von möglichen Terminen mit den unter 3.2.2 vorgestellten Konzepten und einem frühzeitigen Abbrechen bei auftretenden Schleifen sollte die Komplexität jedoch beherschbar sein.

4.5.2 Konfiguration

Die Einstellungen der Anwendung bezüglich von Automatismen welche von Advocate Agents sowie Doorman und Event Managern bereit gestellt werden können kann sehr komplex werden. Dies betrifft ebenso die Einstellungen bezüglich der Privatsphäre.

Um diese Komplexität zu reduzieren, können auf Übersichtsseiten ähnliche Einstellungen gruppiert werden, sodass der Anwender zum Beispiel schnell eine Übersicht bekommt, wieviele seiner Daten öffentlich sind. Über diese Seite ist dann über eine Zoom Funktion eine genauere Betrachtung der einzelnen Einstellungen möglich.

5 Design

5.1 Szenarien

Ein Szenario mit dem durch diese Arbeit geführt wird, ist das des Rennradfahrens. Alternative Szenarien sind die Terminplanung in einer Arztpraxis, die Unterstützung bei der sozialen Interaktion in einem Haus, als Doorman oder Hilfe bei der Koordination einer Band.

5.1.1 Rennradfahren

Eine lose Gruppe von drei bis zehn Personen fährt ein bis zwei mal die Woche an wechselnden Tagen Rennrad. Die Teilnehmer kommen aus verschiedenen Bekanntenkreisen und kennen einander meist nur durch das Rennradfahren.
Das Finden eines Termins erfordert die Kommunikation und das Vertrauen mehrerer Gruppenmitglieder, um einen für diese Gruppe passenden Termin zu finden.
Eine Teilung der Gruppe ist denkbar, wenn zu viele an einem Termin teilnehmen wollen oder aber die terminlichen Gegebenheiten dafür sprechen.

5.1.2 Terminplanung Arztpraxis

In einer Arztpraxis müssen Tag für Tag sehr viele Termine mit Patienten gemacht werden. Diese Patienten haben Vorlieben, manche Patienten sind Notfallpatienten und haben eine höhere Priorität. Des Weiteren haben die Patienten individuelle Randbedingungen wie Arbeitszeiten oder andere Termine.

Durch die Kommunikation mit den jeweiligen Advocate Agents (5.3.2) ist es dem Agenten der Praxis möglich, bei der Terminplanung zu unterstützen. Falls ein Termin kurzfristig frei wird, können die Agenten untereinander schnell jemanden finden, der Zeit hat und es von seiner aktuellen Position aus noch pünktlich zum freigewordenen Termin schafft.

5.1.3 Doorman Szenario

Der Doorman, wie unter 5.3.2 beschrieben, hilft einer Gruppe bei der Koordination und unterstützt somit die soziale Interaktion der Gruppe.

Der Doorman in seiner ursprünglichen Bedeutung als Hüter oder Helfer eines Mehrfamilienhauses kann vielfältige Aufgaben unterstützen.

- Die Koordination eines regelmäßigen Nachbarschaftstreffens wird unterstützt.
- Falls jemand es nicht schafft, sein Kind rechtzeitig vom Kindergarten abzuholen, wird nach vertrauenswürdigem Ersatz gesucht.
- Ähnliche Interessen zweier Personen können ein Bekanntmachen dieser Personen nahe legen[1].
- Die Nachbarn können sich bei kleinen Einkäufen von selten besuchten Geschäften gegenseitig unterstützen, wenn sie über die Wünsche der Anderen in Kenntnis gesetzt werden.

5.1.4 Band

Eine kleine Band mit einer festen Fangemeinde hat regelmäßig Auftritte. Ein Veranstalter fragt bei einem Bandmitglied nach, ob ein Termin für die Band möglich ist. Darauf fragt dieses Bandmitglied die anderen Mitglieder und meldet sich wiederum bei dem Veranstalter mit dem Ergebnis. Sobald die Veranstaltung steht, werden die Fans über das bevorstehende Konzert informiert.

[1] Natürlich nur unter starker Berücksichtigung ihrer Privatsphäre.

5.2 Use Cases

In diesem Abschnitt wird die Interaktion des Anwenders mit dem System beschrieben.

Der Anwender hat, wie in Abbildung 5.1 dargestellt, verschiedene Interaktionsmöglichkeiten mit dem System.

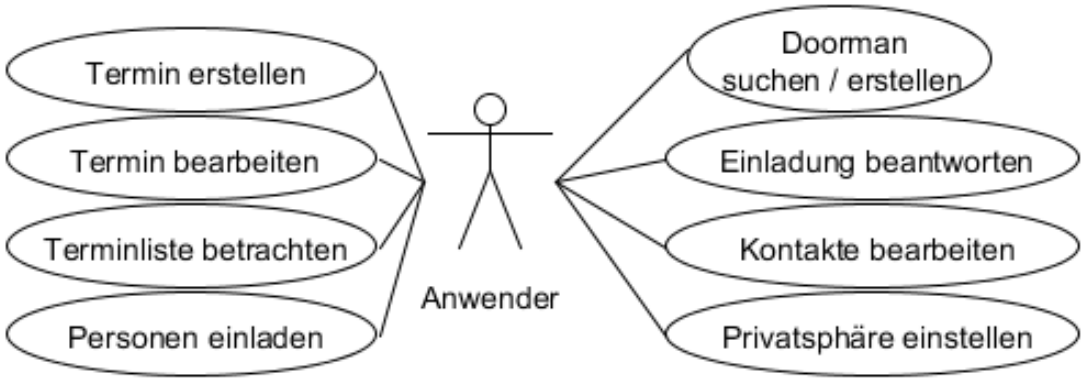

Abbildung 5.1: Use-Cases: Anwender

Termine

Der Anwender kann Termine erstellen, betrachten oder bearbeiten. Hierbei kann er den Titel, den Ort, die Sichtbarkeit, das Datum und die Dauer des Termins festlegen sowie Teilnehmer einladen.

Der Anwender kann sich Listen von Terminen anzeigen lassen. Diese können entweder seine eigenen Termine oder aber Termine sortiert nach bestimmten Einschränkungen sein. Hierfür könnten beispielsweise alle öffentlichen Termine zum Thema 'Rennradfaren in Hamburg' angezeigt werden. In dieser Ansicht können dem Anwender Überschneidungen von Terminen sowie berechnete Wegzeiten zwischen den Terminen angezeigt werden.

Kontakte

Der Anwender kann Kontakte hinzufügen, lesen, bearbeiten oder löschen.

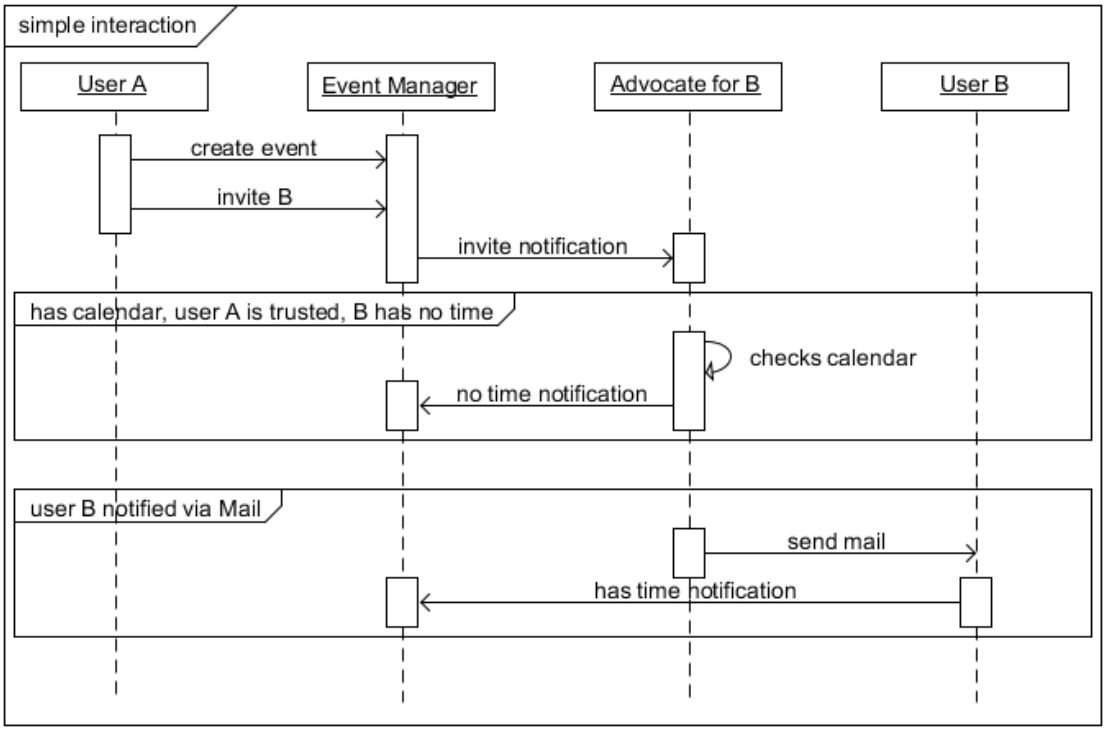

Abbildung 5.2: Sequence Diagramm für zwei einfache Anwendungsfälle

Hierbei kann es sich sowohl um Kontakte zu anderen Anwendern des Systems als auch zu externen Kontakten sein. Das Adressbuch enthält die Benachrichtigungsmöglichkeiten, wie zum Beispiel SMS oder Mail, für diese Kontakte.

Doorman

Des Weiteren kann ein Anwender einen Doorman (5.3.2) erstellen, suchen oder kontaktieren wenn es darum geht, für ein bestimmtes Thema einen Doorman zu finden.

Einstellungen

Ein Anwender hat die Möglichkeit die Anwendung nach seinen Bedürnissen zu konfigurieren. So gibt es Einstellmöglichkeiten zu folgenden Punkten:

- Privatsphäre
- Benachrichtigungsoptionen

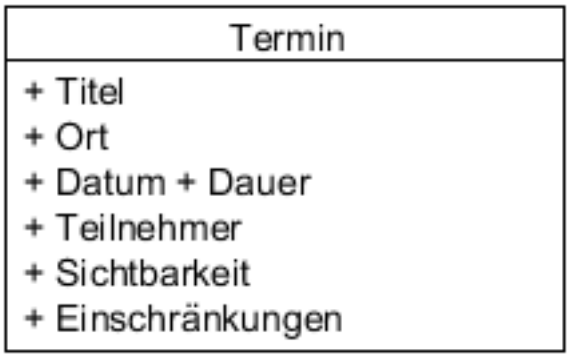

Abbildung 5.3: Die Attribute eines Termins

- Integration von externen Kalendern

5.3 Modell

5.3.1 Daten

Die Zeitplanung wird abgebildet durch Agenten, die miteinander über Veranstaltungen und deren Eigenschaften, sowie die Einschränkungen der Eigenschaften diskutieren.

Der veranstaltende Agent kontrolliert die Einschränkungen, die mit einem Termin einhergehen. Dadurch kann die Verhandlung bei entsprechenden Vorgaben davon abweichen, den größten gemeinsamen Gewinn zu erzielen.

Mögliche Einschränkungen für eine Veranstaltung können einfach oder komplex sein, wie zum Beispiel:

- In Hamburg, am 10.10.2010 ab 10 Uhr, nicht zu weit von mir Zuhause
- einmal die Woche
- Sonntags 20:15 bei jemanden 'der nen Fernseher' hat
- bei IKEA
- in x Minuten beim Kindergarten meiner Tochter, diese bitte abholen, nur vertrauenswürdige Personen
- Zusagen bitte bis in zwei Wochen - zwecks Reservierung
- mindestens x Teilnehmer

5.3.2 Agenten

In dieser Arbeit dienen die Agenten als

- Interaktionsmöglichkeit für den Endnutzer
- Entscheidungsunterstützung durch die Integration von verschiedenen Datenquellen[2]
- Koordinationshilfe bei heterogenen Gruppen
- Wahrer der Privatsphäre
- Nachrichtenbote bei der Verteilung von Einladungen, Absagen und Teilnahmebenachrichtigungen

Hierbei gibt es mehrere Agenten, die diese Rollen wahrnehmen.

[2] Kalender, evtl. Routen, Reservierungen, Wettervorhersagen

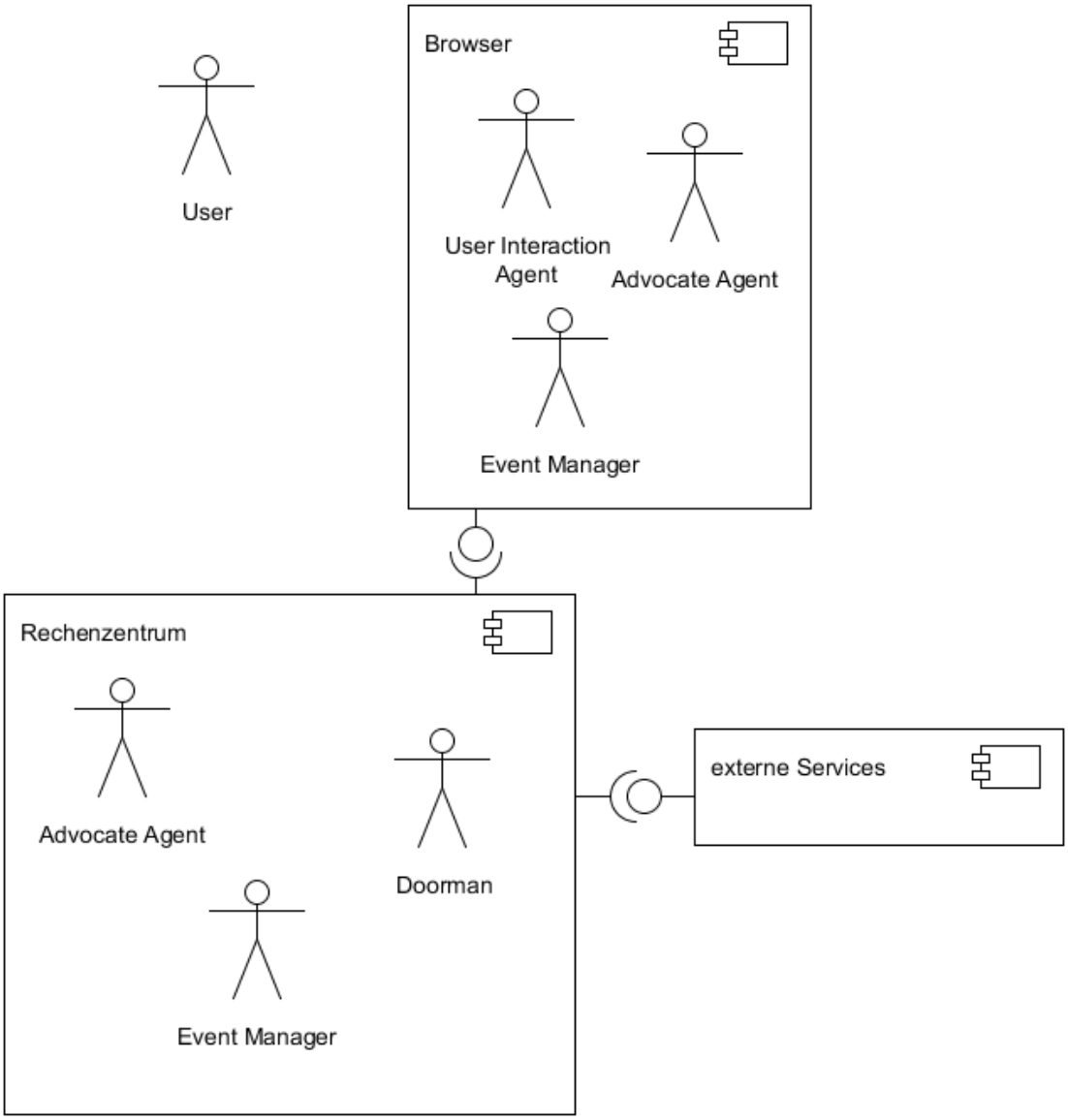

Abbildung 5.4: Eine Übersicht mit den wichtigsten Agenten

User Interaction Agent

Der User Interaction Agent fungiert als Schnittstelle für die Interaktion des Anwenders mit den Agenten des Systems. Hierfür ermöglicht dieser Agent dem Anwender die Ausführung der in Abbildung 5.1 gezeigten Use Cases.

Advocate Agent

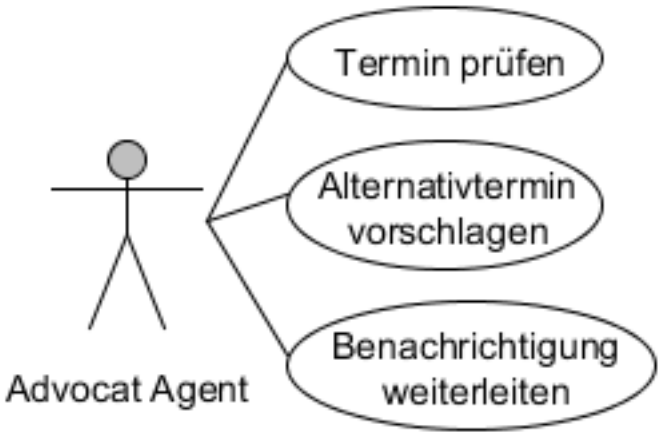

Abbildung 5.5: Use-Cases: Advocate Agents

In Ketter u. a. (2008) wird eine Architektur für Advocate Agents vorgestellt, die die Vorlieben ihres Menschen kennen lernen sollen und ihm dann helfen sollen, Entscheidungen zu treffen (siehe auch 3.1.5).

Ganz in diesem Sinne soll jedem Anwender ein Agent beiseite gestellt werden. Dieser Agent ist die rechte Hand des Anwenders, kennt dessen Vorlieben, dessen Kalender, dessen Freunde und hilft dabei, den Alltag zu organisieren. Er hilft das Leben etwas einfacher zu machen und weniger Zeit für die Planung von Terminen zu benötigen. Dem Anwender sollen Freiräume geschaffen werden, mehr von seinen Wünschen in Erfüllung gehen zu lassen. Dabei kann er auf die Wünsche bezüglich der Randbedingungen der Zeitplanung, wie in 2.1.1 beschrieben, Rücksicht nehmen.

Hierfür verwaltet der Advocate Agent den Kalender des Anwenders, er kennt die üblichen Ruhezeiten, die Gewohnheiten und Vorlieben. Zur Befriedigung dieser Vorlieben schlägt der

Advocate Agent dem Anwender gegebenenfalls Doorman (5.3.2) vor, welche beim Koordinieren von Terminen oder beim Finden von Gleichgesinnten helfen können.

Der Advocate Agent kennt die Vorstellung von Privatsphäre, die sein Anwender hat und hilft ihm, diese zu bewahren, besonders in der direkten Interaktion des Advocate Agents mit anderen Agenten.

Doorman

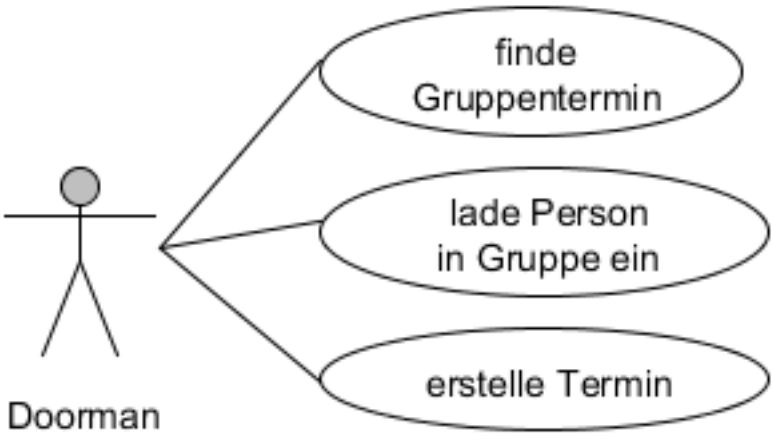

Abbildung 5.6: Use-Cases: Doorman

Der Name ist dem Doorman in amerikanischen Häusern entlehnt, welcher das soziale Leben des Hauses im Auge hat und falls gewünscht, Hilfestellungen bietet.

Der Doorman ist analog bei Bergenti u. a. (2004) als *Session Manager Agents* bezeichnet.

Der Doorman ist die *gute Seele* einer Gruppe von Personen. Der Doorman hilft Gruppen von Personen, sich zu koordinieren und gemeinsame Termine zu finden. Um dieses zu ermöglichen, kann der Anwender einem Doorman erweiterten Zugriff auf den eigenen Advocate Agent ermöglichen.

Ein Doorman ist ein Agent, welcher Gruppen von Personen als gemeinsame Vertrauensbasis dient. Hierbei hilft der Doorman, die Gruppe zu organisieren und hat erweiterten Zugriff auf die einzelnen Teilnehmer. Hierdurch kann der Doorman zwischen den Teilnehmern einer Gruppe gezielt vermitteln.

Event Manager

Ein Event Manager kümmert sich um die Verwaltung eines Termins. Hierzu gehört, dass der Event Manager sich darum kümmert, dass die Einschränkungen eines Termins eingehalten werden. Der Event Manager kann beauftragt werden, Personen zu einem Termin einzuladen.

Ein Event Manager kann unter anderem von einem Doorman bestellt werden, um das wöchentliche Rennradfahren für “nächsten Dienstag“ zu planen.

Der Event Manager verwaltet eine Historie eines Termins. Hieraus lässt sich erschließen, welcher Anwender welche Veränderung vorgenommen hat.

Statistik Agent

Der Statistik Agent sammelt Daten und kann von dem Anwender genutzt werden, um Statistiken über seine Interaktion mit dem System zu sammeln.

Agenten beim Anwender

Die Agenten können sowohl auf Serverseite als auch direkt beim Anwender aktiv werden und ihre Aktionen ausführen. Hierdurch lassen sich viele Aktionen schon direkt beim Anwender ausführen, obwohl es sich um eine Webanwendung handelt.

5.3.3 Services

MessageHub

Das MessageHub ist für die Verteilung von Nachrichten an verschiedene Teilnehmer zuständig. Die jeweiligen Doorman und Advocate Agents haben hierdurch die Möglichkeit über verschiedene Kommunikationswege mit dem Anwender zu kommunizieren.

Mögliche Kommunikationswege sind zum Beispiel:

- Mail
- SMS
- Brief
- Fax
- Chat[3]

Die erweiterten Integrationsmöglichkeiten wie Brief oder Fax ermöglichen das Erreichen von Personen ohne Internet (2.1.2). Die jeweiligen Antworten müssen von dem Ersteller des Termins entgegengenommen und entsprechend eingetragen werden. (Es sei denn, es findet sich ein Service, der dies anbietet.)

Privacy

Der Privacy Service bietet einem Agenten die einfache Möglichkeit, Daten auf das Notwendige zu beschränken. So können zur Terminzeitsuche die Daten auf den angefragten Zeitraum sowie die reinen Terminzeiten beschränkt werden.

Der PrivacyService bietet die Funktion, Daten auf ihren notwendigen Minimalgehalt zu beschränken. Hierzu gehört beispielsweise, dass zum Finden eines gemeinsamen Termins nur die zur Verfügung stehenden Zeiträume notwendig sind, aber nicht die Kenntnis, welche Termine in der übrigen Zeit stattfinden.

[3] Über das Einbinden von vorhandenen Chat-Clients wie icq, Skype, Google-Talk, Facebook Chat, ..

Tags

Um das Erstellen einer ganzen Taxonomie zu umgehen und den Anwendern trotzdem eine einfache Möglichkeit zu geben, ihre Termine zu sortieren, sowie um ähnliche Termine zu finden, gibt es einen TagService. Dieser bietet Anwendern die Möglichkeit Termine, Kontakte oder Orte mit Tags zu versehen. Diese Tags können sowohl vom Anwender manuell angelegt werden, als auch von Agenten vorgeschlagen werden.

Findet zum Beispiel immer in einer ähnlichen Gruppe am selben Ort ein Termin mit jeweils dem selbem Titel statt, kann dieser von einem Agenten automatisch mit einem Tag versehen werden, sodass die regelmäßigen Teilnehmer als Vorschlag für zukünftige Termine an dem selben Ort zur Verfügung stehen.

Adressbuch

Das Adressbuch bietet dem Anwender die Möglichkeit, seine Kontakte zu verwalten. Hier kann der Nutzer die Zugriffsberechtigungen anderer Nutzer auf sein Profil festlegen.

Kalender

Ein Service, welcher das Einbinden sowie den Export von Kalendern via iCal (3.2.2) ermöglicht.

Mail Empfänger

Ein Service, der eingehende Mails annimmt und so das direkte Antworten auf Terminanfragen und dem Anwender somit einen einfacheren Einstieg in die Verwendung der Software ermöglicht.

5.3.4 Kommunikation

Die Agenten kommunizieren über Veränderungen von Eigenschaften und Einschränkungen, die meist einem Kontext zugeordnet sind.
Ein Kommunikationskontext kann zum Beispiel die Kommunikation mit einem Event Manager über ein bevorstehenden Termin sein. Alternativ kann es sich um die Kommunikation mit einem Doorman oder einem Advocate Agent handeln.

Die Kommunikation ist an Operational Transforms (Ellis und Gibbs (1989)) angelehnt, welche unter anderem bei Google Wave verwendet wurde.

Veränderungen werden als Änderungsbefehl[4] zu einem Kontext an die beteiligten Parteien weitergeleitet. Diese Art der Weitergabe ermöglicht eine einfache Realisierung von einer Veränderungshistorie.

Die Verteilung der Nachrichten erfolgt über einen Event-Bus, welcher mittels eines Transport Agents an den Server angebunden ist. Der Transport der Nachrichten zwischen dem Server und den Clients erfolgt als JSON Nachricht und ermöglicht so eine Anbindung neuer Clientanwendungen.

5.3.5 Benutzerinterface

Das Protokoll für die Kommunikation zwischen den Agenten soll die Kommunikation zwischen einem Agenten seinem Besitzer bewerkstelligen können.

So können die Anwendungen für den Endanwender als Agent implementiert werden, der direkt über ein Benutzerinterface verfügt und die Anweisungen an den 'richtigen' Agenten weiter gibt und gegebene Einschränkungen gleich auswerten kann.

Dieser Agent könnte erweiterte Dienste anbieten, wie Hilfe beim Finden von Adressen oder gar beim Ausfüllen helfen, indem er die bisherigen Gewohnheiten als Grundlage nimmt.

[4] Analog zum Command Pattern (Gamma u. a. (1995) Seite 233ff.)

5.4 Architektursichten

Es gibt verschiedene Betrachtungsweisen der gewählten Architektur.

5.4.1 Agentenarchitektur

Die seit etwa 2000 stagnierende Geschwindigkeit der einzelnen Prozessorkerne wird durch eine Erhöhung der Parallel arbeitenden Prozessoren ausgeglichen (Catanzaro und Keutzer (2010)). Die hieraus resultierende Anforderung von Parallelität in Computerprogrammen lässt sich mit Agentensystemen gut umsetzen.

Die Verteilung von Agenten auf die Clientsysteme der Endanwender ist eine Möglichkeit, die dort vorhandenen Ressourcen zu nutzen und eine hilfreiche Architektursicht, um verteilte Systeme zu betrachten.

Die Agenten dienen aus Architektursicht gut dazu, in sich abgeschlossene Teile der Gesamtsoftware zu entwerfen. Diese Softwareteile können durch eine verteilte Agentenarchitektur beliebig migriert werden und auf unterschiedlichen Plattformen (siehe auch 6.1.2) laufen.

6 Realisierung

In diesem Kapitel werden die zur Umsetzung der Anwendung verwendeten Technologien dargestellt.

Die realisierte Software konzentriert sich eher auf die technischen als auf die funktionalen Anforderungen.

6.1 Technologie

6.1.1 Java

Java hat sich als Programmiersprache für Webanwendungen bewährt.
Jetty[1] ist ein weit verbreiteter Java HTTP-Server und Servlet Container, welcher als einer der ersten HTTP-Server WebSockets(3.3.2) unterstützte und dessen Entwickler auch aktiv an der Standardisierung beteiligt waren.

6.1.2 Agenten und GWT

Durch die Verwendung von GWT im Frontend kann der entwickelte Java-Code für die Agenten problemlos auf den Computern der Endanwender ausgeführt werden, ohne dass dort eine Installation von Software erforderlich ist.

6.1.3 Datenbank

Als Datenbank wurde MongoDB, ein Document Store, eine NoSQL Datenbank gewählt. MongoDB unterstützt schemalose Speicherung von Daten und bietet einen sehr schnellen, indizierten Zugriff auf diese. Dazu unterstützt MongoDB Sharding und bietet gute Skalierbarkeit.

[1] http://jetty.codehaus.org/jetty/

6.1.4 Kommunikation

Zur Kommunikation zwischen Server und Client werden Nachrichten im JSON-Format[2] ausgetauscht. Dies ermöglicht eine Kommunikation welche sowohl mit GWT als auch mit der Datenbank gut harmoniert. So sind JSON Nachrichten unter Javascript und GWT direkt nativ verarbeitbar und die Kommunikation zwischen der Anwendung und der MongoDB erfolgt per BSON[3], einer binären Form von JSON, welche sich leicht ineinander überführen lassen.

Die GWT Anwendung hat mehrere Möglichkeiten von dem Server Aktualisierungen zu erhalten:

- Polling - die Anwendung fragt in regelmäßigen Zeitabständen beim Server nach, ob es neue Daten gibt. Dies erzeugt auf Serverseite eine relativ hohe Last und die Daten kommen je nach Abfrageintervall zeitverzögert beim Anwender an.
- Bayeux - durch die Verwendung des Bayeux Protokolls mit Cometd[4] ist es möglich mittels einer offen gehaltenen HTTP-Session ein Server-Push zu realisieren, welche mit nur wenig Verzögerung Aktualisierungen zum Anwender zu bekommen. Des Weiteren werden hierbei weniger Zugriffe je Anwender notwendig (die Verbindungen werden bis zu 30 Sekunden offen gehalten) und es gibt serverseitige Implementierungen welche von dem sonst bei Java-Serlvets üblichem “eine Connection - ein Thread“[5] abweichen und die Threads während der Wartezeiten freigeben.
- WebSockets (3.3.2) sind die neueste Methode für Server-Push und bieten auch die einfachste Kommunikationsmöglichkeit, werden jedoch bisher nur durch wenige Browser unterstützt.

In der prototypischen Umsetzung wurde lediglich Polling als Kommunikationsform realisiert.

6.1.5 Zeit

Die in 3.2.1 gezeigten möglichen zeitlichen Relationen wurden im Rahmen dieser Arbeit nicht umgesetzt. Für die prototypische Umsetzung wurde auf die unter 3.2.2 beschriebenen Mechanismen zurückgegriffen.

Diese Reduktion wurde durchgeführt, da die Komplexität der Anwendung, sowohl für die Berechnungen, als auch für die Usability stark gestiegen wäre.

[2] http://json.org/
[3] http://bsonspec.org/
[4] http://cometd.org/
[5] Dies ist auf anderem Wege erst mit Servlet 3.0 Async oder mit speziellen APIs vom Servlet Container, wie Continuations bei Jetty, möglich.

6.2 Architektur

In diesem Abschnitt wird die Architektur aus der Sicht der verwendeten Technologien und aus der softwareseitigen Umsetzung beschrieben.

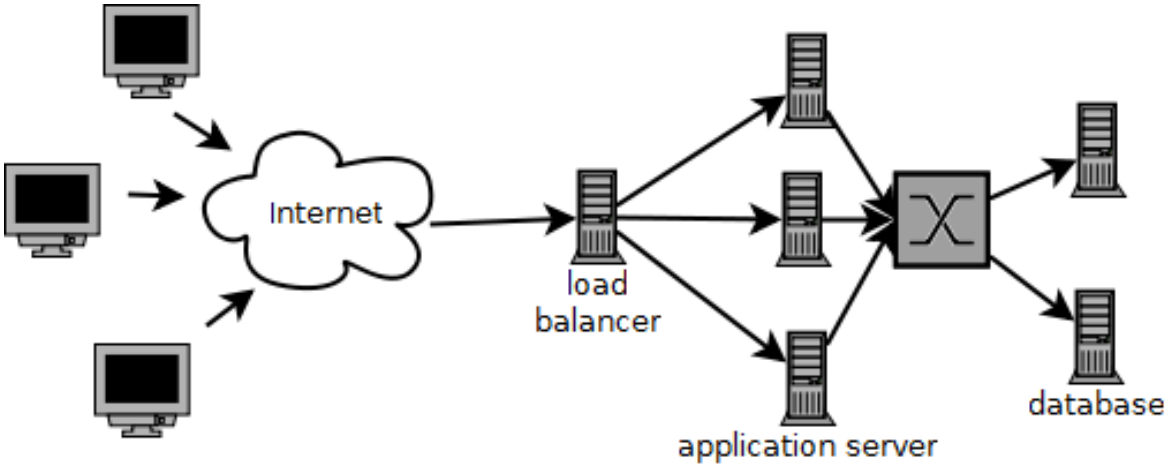

Abbildung 6.1: Der Aufbau der einzelnen Komponenten

Die Anfragen der Anwender werden von einem *Load Balancer* auf mehrere *Application Server* verteilt, welche wiederum auf eine Datenbank zugreifen.

Als *Load Balancer* wird nginx[6] verwendet. Die Wahl fiel auf nginx, da dieser viele Vorteile bietet, welche eine gute Skalierbarkeit der Anwendung versprechen[7].

Als *Application Server* wird der Servlet Container Jetty eingesetzt.

Ein Anwender lädt sich die GWT-Anwendung in seinen Browser. In dem Browser des Anwenders interagieren darauf mehrere Agenten, welche sich fehlende Daten asynchron vom Server schicken lassen können.

6.2.1 Skalierbarkeit

Um die Skalierbarkeit der entwickelten Webanwendung zu gewährleisten, wurden einige Architekturentscheidungen getroffen.

[6] http://nginx.org/
[7] siehe auch http://nginx.org/en/

- *Load Balancer* - durch die Verwendung von nginx als *Load Balancer* können die Anfragen auf viele Anwendungsserver verteilt werden. Des Weiteren bietet nginx eine Entlastung der Anwendungsserver durch die Verwaltung von Keep-Alive-Verbindungen.
- Datenbank - MongoDB (6.1.3) bietet als Vertreter der NoSQL Datenbanken Unterstützung für Sharding und bietet somit eine Grundlage als gut skalierendes Datenbanksystem.
- GWT im Frontend - durch die Verwendung von GWT im Frontend kann, obwohl es sich, um eine Web-Anwendung handelt, ohne Bruch in der Programmiersprache, Funktionalität welche zunächst für die Ausführung auf dem Server vorgesehen war, leicht auf den Client gebracht werden und somit den Server entlasten.

Durch die Bündelung von Aktivitäten bei einzelnen Agenten erhöht sich zudem die Lokalität der Daten und somit die Effizienz des Gesamtsystems. Dies könnte auch gezielt zur Migration von Agenten verwendet werden, indem Agenten, welche viel miteinander kommunizieren, auf die selben Server Instanzen migrieren.

Erste Untersuchungen der entwickelten Anwendung auf mehreren virtualiserten Servern haben gezeigt, dass die bisherige Umsetzung linear mit zunehmender Clientanzahl skaliert und selbst bei einer Auslastung der verfügbaren Ressourcen nur linear langsamere Antwortzeiten entstehen.

Eine Untersuchung, ob die Konzepte auch bei der Skalierung über viele reale Server skalieren, steht noch aus.

7 Schluss

7.1 Fazit

Die hier vorgestellte Lösung beschreibt eine verteilte Agentengesellschaft, welche in der Lage ist, heterogene Benutzergruppen bei der Koordination von sozialer Interaktion proaktiv zu unterstützen.

Bisherige Forschungen (4.1) oder kommerzielle Umsetzungen (4.2) von Groupware oder Kalendern haben ihren Fokus entweder auf eine homogene Anwenderschaft gelegt oder sind rein passive Systeme.

Das vorgelegte Konzept ist ein Schritt zu mehr sozialer Interaktion und bietet hoffentlich auch einige Ideen, welche zeigen, wie ein Computer unterstützend dem Menschen helfen und somit Zeit schenken kann.

Umsetzung

Die Idee einer proaktiven und vorausschauenden Agentengesellschaft ist in der prototypischen Umsetzung eher reaktiv umgesetzt worden. Die Umsetzung zeigt die grundsätzlichen Funktionen und demonstriert eine Tragfähigkeit der Konzepte.

Evaluation

Um eine Auswertung zu machen, in wieweit die vorgestellten Konzepte und Ideen Früchte tragen, müsste die Anwendung weiter entwickelt werden und in die freie Wildbahn ausgesetzt werden.

Danach ließen sich auch folgende Fragen beantworten:

- Was für Netze bilden die Agenten untereinander?
- Wie lässt sich das System für den Anwender verständlich darstellen?

- Wie lässt sich die Komplexität verstecken, ohne dem Anwender Informationen vorzuenthalten?
- Wie sehr sind die Anwender bereit, die Kontrolle über ihre Terminplanung aus der Hand zu geben?

7.2 Ausblick

In diesem Abschnitt möchte ich abschließend weitere Ideen aufzeigen, die sich mit den hier vorgestellten Konzepten untersuchen lassen.

7.2.1 Zeitprobleme verteilt lösen

Durch die verteilte Architektur und die Möglichkeit, Agenten in den Browsern der Anwender laufen zu lassen, können komplexe Aufgaben auch in dieser Architektur verteilt werden. Interessant wäre es zum Beispiel, zu versuchen, die Lösung zum Finden von passenden Terminen bei komplexen Voraussetzungen auf diese verteilten Ressourcen zurückzugreifen zu lassen.

7.2.2 Datenalterung

Alte Termine könnten automatisch nach einiger Zeit altern. Hierfür könnten diese mehr und mehr in Ihrem Detailreichtum reduziert werden und nach einiger Zeit ganz verschwinden. Anwender, die einen Termin nicht gelöscht sehen wollen, müssten diesen rechtzeitig archivieren. Alternativ könnten vergangene Termine, die häufiger betrachtet werden länger bestehen bleiben. Dies wäre ein Versuch, die Menge an gesammelten Daten zu beschränken.

7.2.3 Ressourcenverwaltung

Als primäres Ziel dieser Arbeit kann man die Unterstützung bei der Verwaltung der Zeitplanung von Menschen sehen. Eine erweiterte Form wäre die Verwaltung von Ressourcen im Allgemeinen. Hierzu könnte das Planen von Besprechungsräumen oder eventuell benötigten weiteren Ressourcen wie einem Beamer gehören. Diese Ressourcen würden jeweils einen Advocate Agent (5.3.2) zugeordnet, welcher die Verwaltung relativ eigenständig übernehmen könnte, dabei aber auch mit einem oder mehreren Verantwortlichen Rücksprache halten kann.

7.2.4 Dynamische Kalender

Termine müssen nicht immer nur zu bestimmten Zeitpunkten beginnen oder enden, sondern können vielmehr voneinander abhängen. Ein einfaches Beispiel hierfür wäre eine Arztpraxis, welche die Patiententermine mit der Software plant (siehe auch 5.1.2). Hierbei wäre es sowohl möglich, online Termine zu vergeben, als auch dem Patienten Wartezeit zu ersparen, das Absagen von Terminen zu erleichtern und auch andere Personen je nach deren Möglichkeiten auf frei gewordene Termine vorzurücken.

7.2.5 Agentengesellschaften

Durch die Verwendung von Industriestandards zur Interaktion zwischen Agenten können so mit zunehmender Verbreitung nicht nur ganze Gesellschaften von Agenten enstehen, sondern neue Marktplätze und Wirtschaftsräume[1]. Für die hier vorgestellte Lösung wäre eine Integration mit föderiert arbeitenden Kalendern sowie die Integration von Dienstleistungen durch Fremdanbieter denkbar[2].
Diese Idee wurde von Luck u. a. (2005) schon ausführlicher beschrieben.

[1] Wie es auch von Agentlink propagiert wurde (Luck u. a. (2005)).
[2] Zum Beispiel die Integration eines Brief-Versands für Einladungen.

Literaturverzeichnis

[Allen 1983] ALLEN, James F.: Maintaining knowledge about temporal intervals. In: *Commun. ACM* 26 (1983), Nr. 11, S. 832–843. – ISSN 0001-0782

[Barabasi und Albert 1999] BARABASI, A-L. ; ALBERT, R.: Emergence of scaling in random networks. In: *Science* 286 (1999), S. 509–512

[Bergenti u. a. 2004] BERGENTI, Federico ; MARI, Marco ; GARIJO, Mercedes: Collaborator - Enabling Enterprise Collaboration through Agents. In: *WETICE '04: Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*. Washington, DC, USA : IEEE Computer Society, 2004, S. 41–46. – ISBN 0-7695-2183-5

[Berry u. a. 2006] BERRY, Pauline ; PEINTNER, Bart ; CONLEY, Ken ; GERVASIO, Melinda ; URIBE, Tomás ; YORKE-SMITH, Neil: Deploying a personalized time management agent. In: *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. New York, NY, USA : ACM, 2006, S. 1564–1571. – ISBN 1-59593-303-4

[Brewer 2000] BREWER, Eric A.: Towards robust distributed systems. (Invited Talk). In: *PODC*, Principles of Distributed Computing, Oregon, 2000

[Burka 2008] BURKA, Florian: *Agenten in Netzwerken*. HAW-Hamburg, 2008

[Catanzaro und Keutzer 2010] CATANZARO, Bryan ; KEUTZER, Kurt: Parallel computing with patterns and frameworks. In: *XRDS* 17 (2010), Nr. 1, S. 22–27. – ISSN 1528-4972

[CMU] CMU: Carnegie Mellon University. . – URL http://www.cs.cmu.edu/~softagents/multi.html

[Daboo 2009] DABOO, C.: *iCalendar Transport-Independent Interoperability Protocol (iTIP)*. RFC 5546 (Proposed Standard). Dezember 2009. – URL http://www.ietf.org/rfc/rfc5546.txt

[Dawson u. a. 1998] DAWSON, F. ; MANSOUR, S. ; SILVERBERG, S.: *iCalendar Message-Based Interoperability Protocol (iMIP)*. RFC 2447 (Proposed Standard). November 1998. – URL http://www.ietf.org/rfc/rfc2447.txt

[Dawson und Stenerson 1998] DAWSON, F. ; STENERSON, D.: *Internet Calendaring and Scheduling Core Object Specification (iCalendar)*. RFC 2445 (Proposed Standard). November 1998 (Request for Comments). – URL http://www.ietf.org/rfc/rfc2445.txt. – Obsoleted by RFC 5545

[Dawson 1997] DAWSON, Frank: Emerging Calendaring and Scheduling Standards. In: *Computer* 30 (1997), Nr. 12, S. 126–128. – ISSN 0018-9162

[Denko 2003] DENKO, Mieso K.: The use of Mobile Agents for Clustering in Mobile Ad Hoc Networks. In: *SAICSIT*, 2003

[Desruisseaux 2009] DESRUISSEAUX, B.: *Internet Calendaring and Scheduling Core Object Specification (iCalendar)*. RFC 5545 (Proposed Standard). September 2009 (Request for Comments). – URL http://www.ietf.org/rfc/rfc5545.txt. – Updated by RFC 5546

[Ellis und Gibbs 1989] ELLIS, C. A. ; GIBBS, S. J.: Concurrency control in groupware systems. In: *SIGMOD Rec.* 18 (1989), Nr. 2, S. 399–407. – ISSN 0163-5808

[Faulring und Myers 2005] FAULRING, Andrew ; MYERS, Brad A.: Enabling rich human-agent interaction for a calendar scheduling agent. In: *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*. New York, NY, USA : ACM, 2005, S. 1367–1370. – ISBN 1-59593-002-7

[Ferber 2001] FERBER, Jacques: *Multiagentensysteme*. Addison-Wesley, 2001. – ISBN 3-8273-1679-0

[Gamma u. a. 1995] GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph ; VLISSIDES, John: *Design patterns: elements of reusable object-oriented software*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1995. – 117 S. – ISBN 0-201-63361-2

[Gilbert und Lynch 2002] GILBERT, Seth ; LYNCH, Nancy: Brewer's Conjecture and the Feasibility of Consistent Available Partition-Tolerant Web Services. In: *In ACM SIGACT News*, 2002, S. 2002

[Grosso u. a. 2005] GROSSO, A. ; VECCHIOLA, C. ; COCCOLI, M. ; BOCCALATTE, A.: A multiuser groupware calendar system based on agent tools and technology. In: *Proc. Int Collaborative Technologies and Systems Symp*, 2005, S. 144–151

[Haerder und Reuter 1983] HAERDER, Theo ; REUTER, Andreas: Principles of Transaction-Oriented Database Recovery. In: *ACM Computing Surveys* 15 (1983), S. 287–317

[Hoff 2009] HOFF, Todd: An Unorthodox Approach To Database Design : The Coming Of The Shard. (2009). – URL http://highscalability.com/unorthodox-approach-database-design-coming-shard

[Josuttis 2007] JOSUTTIS, Nicolai M.: *SOA in Practice*. O'Reilly, 2007

[Kanitkar und Delis 1998] KANITKAR, V. ; DELIS, A.: Real-Time Client-Server Push Strategies: Specification and Evaluation. In: *RTAS '98: Proceedings of the Fourth IEEE Real-Time Technology and Applications Symposium*. Washington, DC, USA : IEEE Computer Society, 1998, S. 179. – ISBN 0-8186-8569-7

[Karinthy 1929] KARINTHY, Frigyes: *Chain-Links*. 1929

[Ketter u. a. 2008] KETTER, Wolfgang ; BATCHU, Arun ; BEROSIK, Gary ; MCCREARY, Dan: A Semantic Web Architecture for Advocate Agents to Determine Preferences and Facilitate Decision Making. In: *ICEC*, 2008

[Kristol und Montulli 2000] KRISTOL, D. ; MONTULLI, L.: *HTTP State Management Mechanism*. RFC 2965 (Proposed Standard). Oktober 2000. – URL http://www.ietf.org/rfc/rfc2965.txt

[Luck u. a. 2005] LUCK, M. ; MCBURNEY, P. ; SHEHORY, O. ; WILLMOTT, S.: *Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing)*. AgentLink, 2005

[Maes 1991] MAES, Pattie: The Agent Network Architecture (ANA). In: *SIGART*, 1991

[Maes 1994] MAES, Pattie: Agents that reduce work and information overload. In: *Communications of the ACM* 37 (1994), S. 30–40

[Milgram 1967] MILGRAM, Stanley: The small world problem. In: *Psychology Today* 1 (1967), S. 61

[Nebel 1997] NEBEL, Bernhard: Solving hard qualitative temporal reasoning problems: Evaluating the efficiency of using the ORD-Horn class. In: *Constraints* 1 (1997), S. 175–190. – URL http://dx.doi.org/10.1007/BF00137869. – 10.1007/BF00137869. – ISSN 1383-7133

[Paysakhov u. a. 2004] PAYSAKHOV, M. ; ARTZ, D. ; SULTANK, E. ; TEGLI, W.: Network Awareness for Mobile Agents on Ad Hoc Networks. In: *AAMAS*, 2004

[Pritchett 2008] PRITCHETT, Dan: BASE: An Acid Alternative. In: *Queue* 6 (2008), Nr. 3, S. 48–55. – ISSN 1542-7730

[Rosseburg 2009] ROSSEBURG, Kai: *Benutzermodellierung*. HAW-Hamburg, 2009

[Silverberg u. a. 1998] SILVERBERG, S. ; MANSOUR, S. ; DAWSON, F. ; HOPSON, R.: *iCalendar Transport-Independent Interoperability Protocol (iTIP) Scheduling Events, BusyTime, To-dos and Journal Entries*. RFC 2446 (Proposed Standard). November 1998 (Request for Comments). – URL http://www.ietf.org/rfc/rfc2446.txt. – Obsoleted by RFC 5546

[Smith und Lieberman 2009] SMITH, Dustin A. ; LIEBERMAN, Henry: Recognizing and using goals in event management. In: *CHI '09: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*. New York, NY, USA : ACM, 2009, S. 4525–4530. – ISBN 978-1-60558-247-4

[Spiegel 2009] SPIEGEL, Bernt: *Die obere Hälfte des Motorrads*. Motorbuch Verlag, 2009. – 118ff S

[Sycara und Sukthankar 2006] SYCARA, Katia ; SUKTHANKAR, Gita: Literature Review of Teamwork Models. In: *CMU-RI-TR-06-50*, 2006

[Tennstedt 2008] TENNSTEDT, Sven: *Agentenbeschreibung: Expressiv AI (Façade) im Vergleich zu traditionellen Ansätzen*. HAW-Hamburg, 2008

[Wooldridge 2002] WOOLDRIDGE, Michael: *MultiAgent Systems*. Wiley, 2002. – ISBN 0-471-49691-X

[Zhang u. a. 2007] ZHANG, Shiwu ; LEUNG, Clement H. ; RAIKUNDALIA, Gitesh K.: Performance Evaluation of Agent Network Topologies based on the AOCD Architecture. In: *AINAW*, 2007

[Zhang und Liu 2006] ZHANG, Shiwu ; LIU, Jiming: From Local Behaviors to the Dynamics in an Agent Network. In: *WI 2006 Main Conference Proceedings*, 2006

[Zhong u. a. 2003] ZHONG, Guoqiang ; TAKAHASHI, Kenichi ; AMAIYA, Satoshi ; MATSUNO, Daisuke ; MINE, Tsunenori ; AMAMIYA, Makoto: From Computer Networks to Agent Networks. In: *HICSS*, 2003

Alle Verweise auf Quellen im Internet wurden am 10.10.2010 auf ihre Aktualität überprüft.

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 14. Oktober 2010	
Ort, Datum	Unterschrift