Ableitung komplexer Kontextinformationen und deren Anreicherung aus externen Quellen

Harald Wonneberger
wonne@rhrk.uni-kl.de

Technische Universität Kaiserslautern

Zusammenfassung In diesem Seminar möchte ich herausstellen, welche Möglichkeiten es gibt, durch Sensoren das Umfeld des Menschen zu analysieren und durch verschiedene Mechanismen rohe Sensordaten zu komplexen Kontextinformationen zu aggregieren. Weiterhin stelle ich grundlegende Modelle vor, wie Kontextinformationen generiert, verarbeitet und weitergeleitet werden können. Außerdem stelle ich Möglichkeiten dar, wie die abgeleiteten Informationen mit externen Datenquellen kombiniert werden können, um daraus weitere Informationen abzuleiten.

1 Dumme Computer?

Wir schreiben das 21. Jahrhundert und befinden uns angeblich im Zeitalter des Internets, der Informationsflut, im Zeitalter der Computer. Die Frage, die sich stellt, ist, warum Computer, mehr als 70 Jahre nach ihrer Erfindung durch Konrad Zuse, immer noch dumme Maschinen sind, die vom Menschen mühsam mit Informationen gefüttert werden müssen? Menschen kommunizieren seit langem nicht mehr nur mit anderen Menschen in ihrer Umgebung, sondern sind gezwungen, die Präsenz von Maschinen und insbesondere von Computern zu akzeptieren. Vielen Menschen ist der Umgang mit Computern immer noch unangenehm, was unter anderem daran liegt, dass man einem Computer eben nicht die Hand geben kann, man einem Computer nicht in die Augen schauen kann, und man auch keine Gefühlsregung des Computers erkennen kann. Dieser Informationsfluss, der unterbewusst zwischen Menschen stattfindet, ist zwischen Mensch und Maschine bisher nicht vorhanden.

Schon 1991 beschrieb Weiser in seiner Vision der Computer des 21. Jahrhunderts Konzepte allgegenwärtiger Computer, die sich unbemerkt in das Leben integrieren und im Hintergrund agieren [We91]. 16 Jahre später ist die technologische Entwicklung stark voran geschritten; dennoch ist die Vision von Weiser längst nicht eingetreten. Stattdessen haben wir Menschen uns mit der Situation abgefunden, dass eine unterbewusste Kommunikation zwischen Mensch und Maschine nicht existiert.

Das enorme Potential für die automatische Erkennung von Umgebungsinformationen durch den Computer erkennt man schon an einfachen Fragestellungen. Warum muss ich im Bus jedes Mal meinen Studentenausweis vorlegen? Warum muss ich in der Mensa meine Mensakarte vorlegen? Warum muss ich immer

noch einen Autoschlüssel benutzen? Warum muss ich in jeder Besprechung mein Handy manuell leise stellen? An folgender Situation sei die Verwendung von Kontextinformationen dargestellt:

> *In einem Raum sitzen mehrere Personen an einem Tisch. Eine Person steht neben einer Leinwand und spricht. Ein Videokonferenzsystem ist aktiviert und mehrere Notebooks werden verwendet.*

Ein kontextbewusstes System kann in dieser Situation anhand mehrerer Sensoren einfache Kontextinformationen ermitteln, die anschließend zu komplexen Kontextinformationen integriert werden. Das System erkennt, wie viele Personen am Tisch sitzen, und identifiziert die Person, die gerade vor der Leinwand spricht. Aus dem Terminkalender dieser Person kann das System sogar das Thema des Vortrags ermitteln. Durch weitere Raumsensoren und historische Daten kann die komplexe Kontextinformation abgeleitet werden, dass eine Besprechung stattfindet. Durch die Verwendung drahtloser Kommunikationsmedien wie z. B. Bluetooth könnte der Klingelton jedes Mobiltelefons im Raum deaktiviert werden, ganz automatisch und ohne dass der Besitzer dies vergessen könnte.

2 Die Kontextwelt

Im oben genannten Beispiel kommen bereits Wörter wie z. B. Kontext, Kontextbewusstsein sowie komplexe Kontextinformationen vor. Diese Begriffe kann man sich zwar näherungsweise herleiten, dennoch existieren in der Fachliteratur zahlreiche, zum Teil sehr unterschiedliche Definitionen. Im Folgenden möchte ich daher kurz auf die Entwicklung der Definitionen eingehen, um anschließend eine Definition anzugeben, welche im weiteren Verlauf der Seminararbeit Anwendung finden wird.

2.1 Kontext

Definitionen der Fachliteratur stellen den Kontext als statische Informationen dar [DA99]. Daraus ergibt sich, dass Informationsbausteine wie z. B. der Ort, eine bestimmte Person oder die Zeit von vielen Autoren als Bestandteile des Kontextes angegeben werden. Diese und andere Definitionen sind allerdings nicht ausreichend, da die Situation, in der eine Kontextinformation ermittelt wird, nicht berücksichtigt wird. Da es von der jeweiligen Situation abhängt, ob eine Information Kontext ist oder nicht, scheint die folgende Definition von Dey sinnvoller:

> *Kontext ist jede Information, die dazu verwendet werden kann, die Situation einer Entität zu charakterisieren. Eine Entität ist hierbei eine Person, ein Ort oder ein Objekt, die/der/das als relevant für die Interaktion zwischen Benutzer und Anwendung eingestuft wird. Hierbei können auch Benutzer und Anwendung eine Entität darstellen [DA99].*

Daraus ergibt sich, das nur solche Informationen Kontextinformationen darstellen, welche einen Einfluss auf die aktuelle Situation nehmen können. Im Weiteren ergeben sich daraus vier verschiedene primäre Kontextarten. Hier nennt Dey *Ort*, *Identität*, *Aktivität* und *Zeit*. Bezogen auf das mögliche kontextsensitive System des Konferenzbeispiels werden alle vier Kontextarten verwendet. Der Ort aller Personen im Raum wird bestimmt, die Identität der vortragenden Person ist dem System bekannt, die Benutzung von Notebooks stellt eine erkannte Aktivität dar und Zeit spielt durch die Integration von Termindaten eine Rolle.

2.2 Kontextbewusstsein

Reagieren Anwendungen auf Kontextinformationen, so spricht man im Allgemeinen von Kontextbewusstsein. Die Fachliteratur unterscheidet hierbei zwei unterschiedliche Ansätze. Zum einen spricht man von kontextbewussten Anwendungen, wenn diese sich an den jeweiligen Kontext *anpassen*. Zum anderen werden auch solche Anwendungen als kontextbewusst aufgefasst, die vorliegende Kontextinformationen *verwenden*. Die folgende Definition von Dey ist etwas weiter gefasst, damit bereits bestehende Anwendungen aus der Kategorie der kontextsensitiven Anwendungen nicht ausgeschlossen werden. Hierzu gehört beispielsweise ein Gerät, das lediglich die vorhandenen Kontextinformationen auf einem Display ausgibt. Auch ein solches System ist kontextbewusst, auch wenn es seinen eigenen Zustand nicht an den Kontext anpasst:

> *Ein System ist genau dann kontextbewusst, wenn es Kontext verwendet, um dem Anwender in Abhängigkeit seiner Tätigkeit Informationen oder Dienste anzubieten [DA99].*

Um im weiteren Verlauf kontextbewusste Anwendungen besser einordnen zu können, wird die Menge der kontextbewussten Anwendungen in drei Klassen unterteilt. Hierbei nennt Dey Anwendungen, die dem Benutzer Informationen *präsentieren*, einen bestimmten Dienst automatisch *ausführen*, und solche, die Informationen mit Kontextinformationen *anreichern*, um diese zu einem anderen Zeitpunkt wieder aufzufinden [DA99]. In unserem Konferenzbeispiel wird bisher nur ein Dienst, nämlich die Deaktivierung des Mobiltelefons automatisch ausgeführt. Eine Präsentation könnte durch eine automatische Mitteilung auf dem Display aller Teilnehmer erfolgen. Die Anreicherung von Informationen könnte durch die Speicherung der ermittelten Kontextinformationen in Zusammenhang mit den vorgestellten Präsentationsfolien erfolgen.

2.3 Komplexe Kontextinformationen

Im obigen Konferenzbeispiel existieren viele verschiedene Sensoren, um Kontextinformationen aus der Umgebung zu gewinnen, wie z. B. zur Positionsbestimmung oder zur Messung der Stromversorgung. Sensoren liefern jedoch noch

keine abstrahierten Daten, sondern lediglich elektronische Signale. Diese müssen in einem weiteren Schritt zu einer Information abgeleitet werden. Dieser Schritt kann einen einzelnen oder mehrere Sensoren abstrahieren. Dadurch können unscharfe oder sich widersprechende Signale verarbeitet bzw. korrigiert werden. Liegen abstrahierte Sensordaten in Form von einfachen Kontextinformationen vor, so lassen sich diese wiederum abstrahieren und integrieren, wodurch sukzessive komplexere Informationen entstehen. Die Schritte der Informationsabstraktion lassen sich in einer baumartigen Struktur wiedergeben (siehe Abbildung 1) [Mä03].

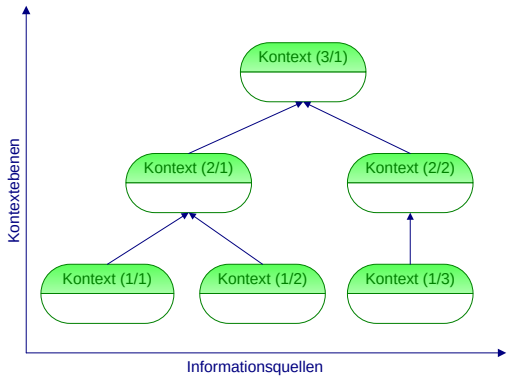

Abbildung 1. Zusammensetzung komplexer Kontextinformationen

So bestehen Kontextinformationen einer Ebene aus einfacheren Kontextinformationen bzw. Sensordaten der darunter liegenden Ebenen. Weiterhin beachtet Mäntyjärvi, dass Sensordaten sowie Kontextinformationen erst im Zeitverlauf entstehen. So kann beispielsweise eine komplexere Information erst dann ermittelt werden, wenn bestimmte Daten angefallen sind, wodurch die Tatsache berücksichtigt wird, dass nicht zu jeder Zeit sämtliche Daten verfügbar sind (siehe Abbildung 2.

In unserem Beispiel liefern Sensoren beispielsweise Informationen über Belastungen des Bodens. Hieraus könnten Informationen über die Anzahl der Personen im Raum gewonnen werden. Integriert man diese Information mit anderen abstrahierten Signalen wie z. B. dem Zustand der Videokonferenzanlage, so kann man in einem weiteren Schritt die komplexe Information „Konferenz findet statt“ ableiten.

Eine grundlegende Frage, die sich in diesem Rahmen ergibt, besteht in der Abgrenzung zwischen einfachen und komplexen Kontextinformationen. Liefern

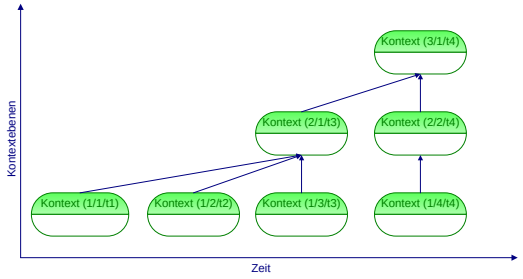

Abbildung 2. Zusammensetzung komplexer Kontextinformationen im Zeitverlauf

beispielsweise drei Sensoren Signalstärken, so kann die Position innerhalb der Signalfelder bestimmt werden. Somit wäre hier die Position schon eine komplexere Information, da die rohen Sensordaten abstrahiert wurden. Hingegen liefert ein GPS-Sensor schon die genauen Positionsdaten, die nicht weiter abstrahiert werden müssen. In diesem Zusammenhang könnte man wieder von einfachen Kontextinformationen sprechen. Eine klare Abgrenzung scheint nicht trivial und muss daher situationsabhängig entschieden werden.

Mäntyjärvi entwirft ein Schichtenmodell, nach dem Kontextinformationen in drei Schichten eingeteilt sind. Auf der untersten Schicht existieren nur reine Sensordaten wie z. B. Temperatur oder Koordinaten. Einfache Kontextinformationen werden durch eine erste Abstraktion erzeugt. Auf dieser Ebene liegen interpretierte Informationen, z. B. warm oder kalt, vor. Nach einer weiteren Abstraktion entstehen die so genannten komplexen Kontextinformationen. Diese zeichnen sich durch eine Beschreibung einer Situation aus. Als Beispiel kann man hier „Konferenz findet statt“ bzw. „Mittagessen findet statt“ nennen. In Abbildung 3 erkennt man auf der linken Seite den Informationsfluß sowie die verschiedenen Ebenen der Kontextinformationen auf der rechten Seite.

2.4 Problemfeld Kontext

Schon heute sind Technologien vorhanden, um Kontext zu erkennen, abzuleiten und weiterzuverwenden. Sensoren wie z. B. RFID können schon bald sehr günstig hergestellt werden, und die Rechenleistung mobiler Endgeräte steigt stetig. Dennoch gibt es nur sehr wenige kontextbewusste Anwendungen, die meist auf ein bestimmtes Gebiet oder einen Raum beschränkt sind. Dey spricht Probleme an, die mit kontextbewussten Anwendungen einhergehen [DA99]. Kontexterkennung erfordert neuartige Technologien, die sich noch im Entwicklungsstadium befinden. Durch nicht-standardisierte Hardware wird der Entwicklungsprozess von kontextbewussten Anwendungen erschwert. Hinzu kommt die Abstraktion von Sensordaten zu Kontextdaten, welche sich von Anwendung zu Anwendung

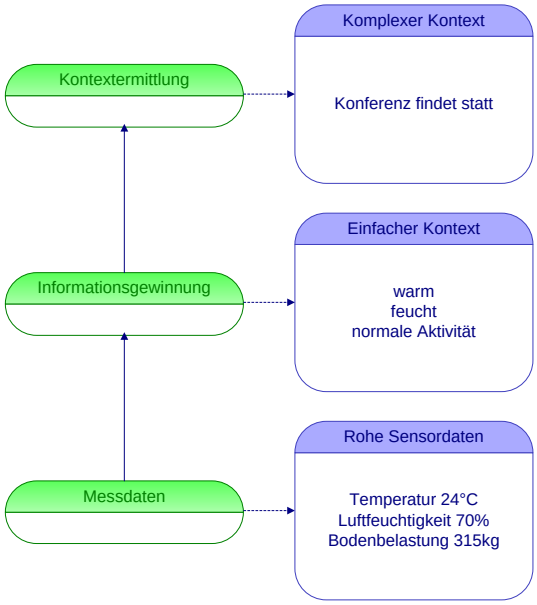

Abbildung 3. Abstraktionsebenen von Kontextinformationen

unterscheidet, sowie die Schwierigkeit, Kontextdaten aus mehreren heterogenen und eventuell verteilten Sensoren zu aggregieren. Durch die ständige Veränderung der Umgebung und die Notwendigkeit, historische Daten zu speichern und wiederzuverwenden, ergeben sich vor allem in mobilen Anwendungen weitere Probleme.

3 Grundlegende Kontextsysteme

Allgegenwärtige Computertechnik wurde erstmals von Weiser 1991 aufgegriffen [We91]. Seitdem haben sich viele Autoren mit dem Thema beschäftigt und Systeme entwickelt, um kontextbewusste Systeme, so genannte Kontextsysteme, implementieren zu können. Schilit teilte erstmals 1995 ein Kontextsystem in drei Komponenten auf [Sc95]. Dey entwickelte 2000 ein auf Java basierendes System, das verschiedene Elemente eines allgemeinen Kontextsystems definierte [De00]. Dieser Ansatz wurde 2004 von Chen aufgegriffen und durch moderne Technologien wie z.B. Ontologien erweitert [CF03]. Diese Systeme werden hier kurz vorgestellt, um einen Einblick in die Architektur und vor allem in die Komponenten von Kontextsystemen zu erlangen. Einzelne Komponenten werden anschließend detaillierter betrachtet.

3.1 Schilits Systemarchitektur

Schilit [Sc95] teilt ein grundlegendes Kontextsystem in die drei Komponenten *Person*, *Gerät* und *Kontext* ein.

Die Komponente Person enthält dabei sämtliche relevanten Informationen, um eine reale Person zu identifizieren. Diese Informationen werden in so genannten Benutzer-Agenten gespeichert. Dadurch kann die Privatsphäre des einzelnen Benutzers bewahrt werden, da jeder Benutzer seinen Agenten beliebig anpassen und private bzw. öffentliche Informationen definieren kann. Geräte werden durch Geräte-Agenten repräsentiert, welche wiederum dynamische Geräteinformationen verwalten. Voraussetzung ist hierbei eine Trennung der Informationen über das Gerät selbst, welche möglichst nah am Gerät oder im Gerät selbst gespeichert werden, von Informationen über die Konnektivität des Gerätes. Als dritte Komponente dient die aktive Karte, die den öffentlichen Kontext darstellt.

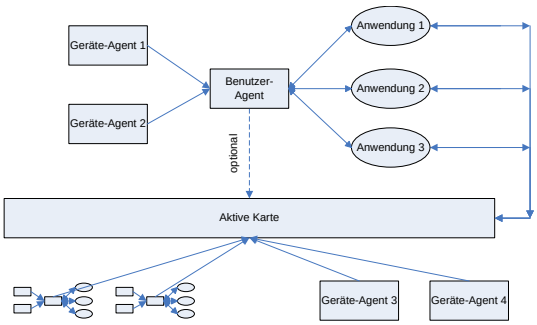

Abbildung 4. Komponenten und deren Interaktion

Geräte-Agenten repräsentieren beispielsweise einen PDA oder einen Arbeitsplatzrechner. Bei Wechsel des Kontextes eines Geräte-Agenten wird die Änderung bekannt gegeben. Je nachdem, ob der Geräte-Agent zu einem Benutzer-Agenten gehört oder ein öffentliches Gerät ist, wird die Information, wie in Abbildung 4 zu erkennen, an den Benutzer-Agenten bzw. an die aktive Karte übertragen. Der Benutzer-Agent speichert private Informationen wie z. B. Aufenthaltsort einer Person. Je nach Einstellung der Privatsphäre werden private Informationen durch die aktive Karte veröffentlicht. Weiterhin stehen Anwendungen sowohl mit dem Benutzer-Agenten, als auch mit der aktiven Karte in Verbindung. Die aktive Karte dient weiterhin auch für den Austausch von Informationen zwischen verschiedenen Benutzer-Agenten, da Benutzer-Agenten keine Möglichkeit haben, direkt miteinander zu kommunizieren.

Das von Schilit erstellte Framework stellt eine grundlegende Architektur zur Verfügung, um kontextsensitive Anwendungen zu erstellen. Somit konnten sich Softwareentwickler erstmals auf eine Architektur stützen und erste Anwendungen implementieren.

3.2 Deys Kontext-Toolkit-Framework

Dey erweitert in seiner Arbeit die Architektur von Schilit. Während sich Schilit jedoch auf die Ermittlung von Kontextinformationen von Benutzern und Geräten konzentrierte, fokussiert Dey seine Arbeit auf die Entwicklung eines Systems, das auf einfache Art und Weise erweitert werden kann. Dey baut sein Kontextsystem so modular auf, dass einzelne Komponenten erweitert werden können, um z. B. eine neue Kontextinformation darzustellen. Weiterhin identifiziert Dey Probleme der Architektur von Schilit, zu denen unter anderem die mangelnde Erweiterbarkeit, die fehlende automatische Erkennung von Agenten, die fehlende Speicherung von Kontextinformationen sowie die fehlende Unterstützung für die Kontextkomponenten *Zeit* und *Aktivität* zählen.

Die Architektur von Dey setzt sich aus einer Klassenhierarchie zusammen, die eine Basis-Klasse enthält sowie die davon abgeleiteten Klassen Widget, Interpreter, Discoverer und Aggregator, wie aus Abbildung 5 zu entnehmen. Die Basis-Klasse enthält sämtliche Methoden zur Kommunikation zwischen den einzelnen Komponenten.

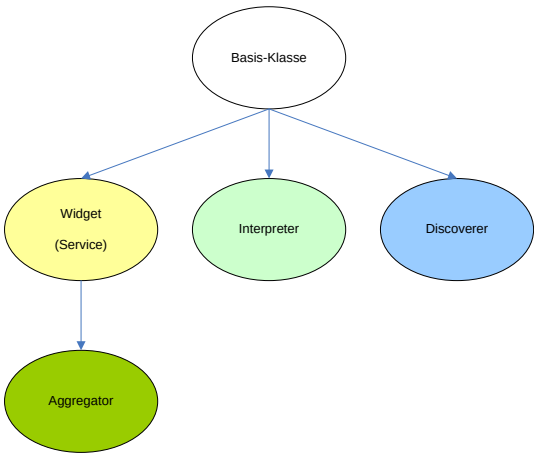

Abbildung 5. Hierarchie des Kontext-Toolkit

Die Interaktion der einzelnen Komponenten kann der Abbildung 6 entnommen werden. Widgets greifen auf vorhandene Sensordaten zu, verarbeiten diese jedoch mit Hilfe von Abstraktion weiter. Ob der Aufenthaltsort einer Person über eine Kamera oder einen RFID-Sensor ermittelt wurde, spielt für das Widget keine Rolle mehr. Informationen werden von einem Widget nur dann weitergegeben, wenn diese benötigt werden. Ist eine Anwendung nur am Gebäude interessiert, in dem sich eine Person aufhält, so liefert das Widget nur beim Wechsel des Gebäudes die entsprechende Information. Um Informationen weiterzugeben,

stellen Widgets eine standardisierte Schnittstelle zur Verfügung, auf die Anwendungen zugreifen können. Services, als Komponente des Widgets, ermöglichen die Ausgabe von Informationen. Wird beispielsweise durch ein Kontextsystem erkannt, dass in einem Raum eine Konferenz stattfindet, jedoch noch nicht alle Personen eingetroffen sind, so kann eine Benachrichtigung an die fehlenden Personen gesendet werden. Sämtliche Kontextinformationen, die ein Widget ermittelt hat, werden im Widget gespeichert. So können Anwendungen auch auf historische Daten zugreifen.

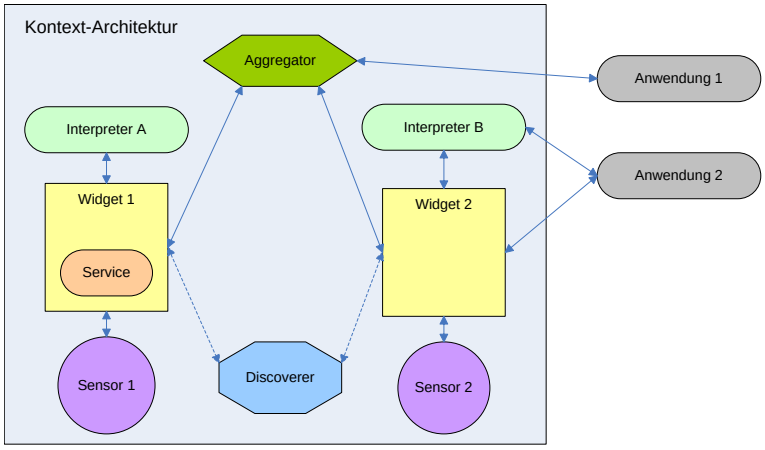

Abbildung 6. Interaktion der Komponenten

Eine weitere Vereinfachung des Systems ist durch die Aggregation von Kontextinformationen gegeben. Aufgrund der Tatsache, dass Anwendungen oftmals mehr als eine Kontextinformation verwenden, bieten Kontext-Aggregatoren die Möglichkeit, verschiedene Widgets zu vereinen. Widgets beziehen sich nur auf einen einzelnen Sensor, beispielsweise einen Sensor für den Aufenthaltsort einer Person. Im Unterschied dazu ermittelt ein Aggregator Informationen zu einer kompletten Entität, wie z. B. einer Person. Anwendungen können somit bei einem Aggregator eine Menge von Informationen über eine Entität abrufen.

Interpreter verwenden Daten der Widgets. Hierbei wird nochmals von den Daten des Widgets abstrahiert. Liefert das Widget beispielsweise Informationen darüber, in welchem Raum sich eine Person befindet, so kann der Interpreter von dieser Information auf andere, wie z. B. den Namen des Gebäudes schließen. Interpreter abstrahieren also einfache Kontextinformationen der Widgets zu komplexen Kontextinformationen.

Der Discoverer dient als Verwaltungseinheit von Widget, Aggregator und Interpreter. Sämtliche Komponenten des Toolkits registrieren sich beim Discoverer,

wodurch diese verfügbar gemacht werden. Neue Anwendungen können Informationen über die im System verfügbaren Ressourcen beim Discoverer erfragen, ohne dass vorher bekannt sein muss, welche Komponenten existieren. Auch neue Komponenten des Toolkits, z. B. ein neues Widget, können sich beim Discoverer registrieren und ihre Funktionalitäten anmelden. Dadurch ist das System modular aufgebaut und kann zu jeder Zeit beliebig erweitert werden.

4 Methoden der Sensorabstraktion

Wie bereits in den vorherigen Kapiteln beschrieben, ist die Abstraktion von Sensordaten bzw. einfachen Kontextinformationen ein wesentlicher Bestandteil von kontextbewussten Anwendungen. Die einzelne Information hat noch keinen großen Wert, wenn der Gesamtzusammenhang nicht klar ist, d.h. die umgebende Situation nicht erkannt wird.

Im Ausgangsbeispiel wird die Konferenz-Situation aus einigen wenigen Sensordaten abgeleitet. Auch in den Systemen von Dey, Schilit und Chen existieren Mechanismen, um komplexe Kontextinformationen abzuleiten. Grundsätzlich ist davon auszugehen, dass in einer kontextbewussten Umgebung viele verschiedene redundante Sensoren existieren, die eine fehlerbehaftete bzw. unscharfe Ausgabe liefern. Um dennoch aus ungenauen Daten Kontextinformationen abzuleiten, müssen die Sensordaten interpretiert werden. Grundsätzlich lassen sich die verfügbaren Methoden wie von Luo in vier Kategorien einteilen [LY02]. Hierzu gehören Methoden der Signalabschätzung, Klassifikationsmethoden, Inferenzmethoden und Methoden der künstlichen Intelligenz. In der Literatur existieren zahlreiche einfache Methoden, um Sensordaten zu abstrahieren. Dazu gehören Verfahren wie z. B. die Bildung des Mittelwertes einzelner Sensordaten, die Berechnung der Standardabweichung, mittlerer quadratischer Fehler und ähnliche. Diese und andere können bei Mäntyjärvi nachgeschlagen werden [Mä03]. Im Folgenden konzentriere ich mich jedoch auf die Methoden der künstlichen Intelligenz, speziell auf Fuzzy-Logik und neuronale Netze, da diese keine harte Abgrenzung zwischen Sensordaten erzwingen, sondern einen fließenden Übergang erlauben [RN03].

4.1 Fuzzy-Logik

Grundlage der Fuzzy-Logik ist der soeben angesprochene fließende Übergang zwischen Sensordaten und den daraus resultierenden Wertebereichen. Liefert ein Sensor beispielsweise Daten zur Helligkeit eines Raums, so werden diskrete Signale an das System geliefert. Aus der Sicht eines Computers könnte dieses Signal in drei Bereiche eingeteilt werden. Daraus resultiert, dass bis zu einem bestimmten Wert a der Raum als *dunkel*, ab dem Wert a bis zum Wert b der Raum als *hell* und ab einem Wert b als *sehr hell* gilt. Für einen Menschen ist diese klare Unterteilung jedoch nicht typisch; es existiert normalerweise ein fließender Übergang zwischen dunkel und hell. Der Wert, ab dem ein Raum hell ist, kann nicht genau definiert werden.

Ein Fuzzy-System besteht im Wesentlichen, wie auch Abbildung 7 zu entnehmen ist, aus drei Komponenten. Hierzu gehören Fuzzifizierung, Inferenzsystem und Defuzzifizierung. Während der Fuzzifizierung werden Sensordaten aufgrund der Zugehörigkeitsfunktionen, die eine Fuzzy-Menge bilden, abstrahiert. Ausgabewerte der Zugehörigkeitsfunktionen werden mit Hilfe von Regeln im Inferenzsystem zu mehreren Ergebniswerten verarbeitet. Anschließend werden diese Ergebniswerte während der Defuzzifizierung zu einem konkreten Ergebniswert zusammengefasst. Der so entstandene Ergebniswert kann als Stellgröße wiederum Einfluß auf das beobachtete System nehmen.

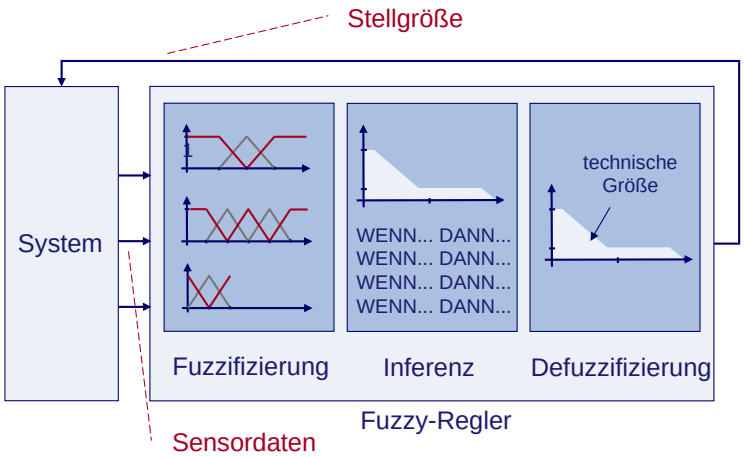

Abbildung 7. Komponenten eines Fuzzy-Systems

In unserem Beispiel gibt es für jeden Sensor drei Zugehörigkeitsfunktionen, für den Helligkeitssensor je eine für den Zustand dunkel, hell und sehr hell. Die Werte der Zugehörigkeitsfunktionen liegen zwischen 0 und 1, wobei 0 keine Zugehörigkeit und 1 eine vollkommene Zugehörigkeit bedeutet. Es ist jedoch auch möglich, eine Zughörigkeit von z.B. 0,8 anzugeben, wodurch wiederum der fließende Übergang zwischen Werten dargestellt wird. Beispiele verschiedener Modellierungen von Zugehörigkeitsfunktionen sind Abbildung 8 zu entnehmen.

Ein weiterer Sensor, der die Temperatur im Raum misst, wird ebenfalls in drei Zugehörigkeitsfunktionen unterteilt, hier sind die Bereiche *kalt*, *warm* und *heiß*. Typische Zugehörigkeitsfunktionen für beide Sensordaten sind beispielsweise Dreiecks-, Trapez-, oder Gauß-Funktionen. Ein Fuzzy-System enthält jedoch neben den Eingabedaten auch Ausgabedaten. Diese sind ebenfalls durch

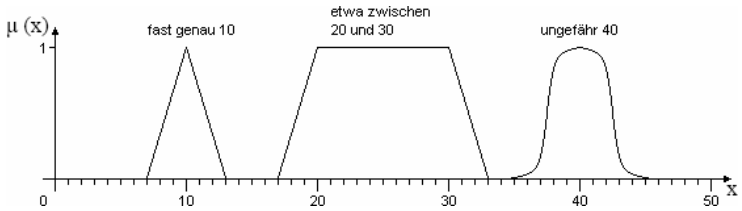

Abbildung 8. Beispiele verschiedener Zugehörigkeitsfunktionen

Zugehörigkeitsfunktionen modelliert. Als Beispiel wäre hier die Regelung der Temperatur eines Raums zu nennen, die in die Bereiche *kühle*, *nichts tun* und *heize* unterteilt ist.

In Anlehnung an die klassische Logik existieren auch in der Fuzzy-Logik Regeln, die Schlussfolgerungen ermöglichen. Fuzzy-Regeln sind typischerweise mit *und* bzw. *oder* verknüpft und stellen die wesentliche Komponente des Inferenzsystems dar. Folgende Regeln seien für das oben beschriebene Fuzzy-System definiert:

1. WENN Helligkeit hell UND Temperatur kalt DANN heize
2. WENN Helligkeit dunkel UND Temperatur kalt DANN tue nichts
3. WENN Helligkeit sehr hell UND Temperatur heiß DANN kühle

Welche Regeln angewendet werden, wird durch die vorliegenden Sensordaten bestimmt. Für unser Beispiel seien die Zugehörigkeitsfunktionen für Helligkeit und Temperatur folgendermaßen definiert. Die Funktion $W()$ gibt hierbei den Wert der Zugehörigkeitsfunktion an.

$$W(dunkel) = 0,4; W(hell) = 0,8; W(sehrhell) = 0,9$$

$$W(kalt) = 0,8; W(warm) = 0,4; W(heiss) = 0,2$$

Die anwendbaren Regeln ergeben sich daraus, indem für jede und-verknüpfte Regel das Minimum und für jede oder-verknüpfte Regel das Maximum der Sensorwerte als Ergebnis liefert.

1. $W(heize) = min\{W(hell) = 0,8, W(kalt) = 0,8\} = 0,8$
2. $W(tuenichts) = min\{W(dunkel) = 0,4, W(kalt) = 0,8\} = 0,4$
3. $W(kuehle) = min\{W(sehrhell) = 0,9, W(heiss) = 0,2\} = 0,2$

Als Ergebnis der Regelauswertung erhalten wir nun Ergebniswerte für die Zugehörigkeitsfunktion der Ausgabewerte. Diese sind für eine maschinelle Weiterverarbeitung jedoch noch nicht geeignet, da kein konkreter Ergebniswert vorliegt. Daher wird in einem letzten Schritt die so genannte Defuzzifizierung vorgenommen, in der die mehreren Ergebniswerte in einen konkreten Ergebniswert umgerechnet werden. Hierfür lässt sich aus Gründen der Einfachheit die Berechnung des Flächenschwerpunktes, wie in Abbildung 9 zu erkennen, heranziehen. Als Ausgabewert erhält man eine technische Größe, die weiterverarbeitet werden kann.

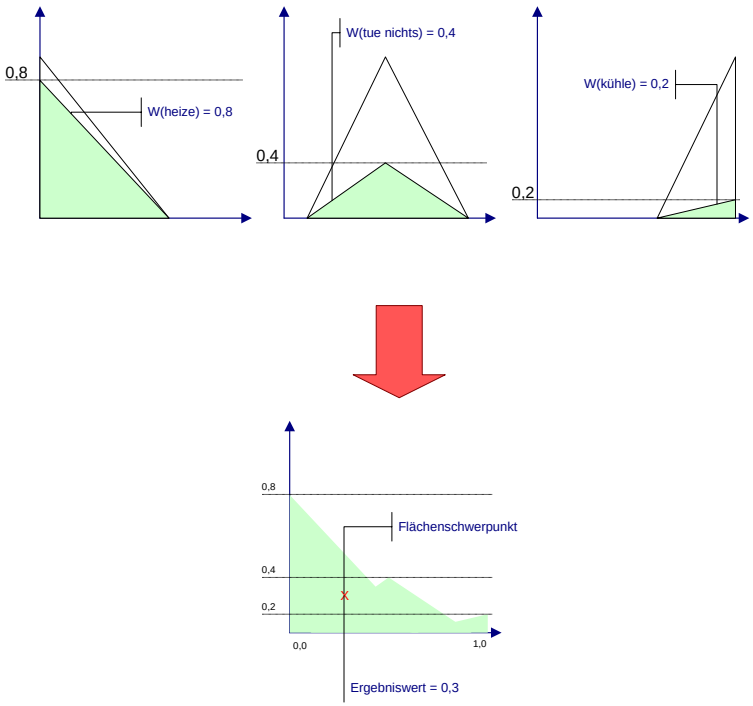

Abbildung 9. Berechnung des Flächenschwerpunktes

Erkennung komplexer Situationen Die bisherigen Ausführungen zu Fuzzy-Systemen abstrahieren Sensordaten zu komplexeren Informationen wie z. B. Temperaturdaten – im Allgemeinen Daten, die ein bestimmtes System regeln sollen. Einen weiteren Schritt der Abstraktion muss man jedoch vollziehen, wenn beispielsweise bestimmte Situationen erkannt werden sollen. Auch ein solches

Abstraktionsniveau lässt sich mit einem Fuzzy-System erreichen, wenn die Sensordaten entsprechend spezifiziert sind [TT06]. Die Fuzzy-Logik erlaubt es auf einfache Art und Weise Regeln festzulegen, die eine bestimmte Situation beschreiben. Um eine Situation zu charakterisieren, werden im Allgemeinen Positionsdaten, Zeitinformationen und Geräteinformationen benötigt. Die Sensoren, die nötig sind, um die genannten Eigenschaften einer Situation zu erfassen, sind im Allgemeinen mit einer Ungenauigkeit behaftet. So kann beispielsweise mit einem Lokalisierungssystem die Position einer Person nur bis zu einer bestimmten Genauigkeit ermittelt werden. Da die genauen Koordinaten bereits fehlerbehaftet sein können, kann auch die abstrahierte Information, in welchem Raum sich eine Person aufhält, fehlerbehaftet sein. So kann man annehmen, dass sämtliche Sensordaten nur mit einer bestimmten Wahrscheinlichkeit ausgewertet werden können. Thomson definiert in seiner Arbeit verschiedene Situationen. Hier werden Informationen zu Positionen, Anzahl der Personen, sowie den Zuständen verschiedener Geräte definiert (siehe Abbildung 10). Weitere Situationen können von bereits definierten Situationen abgeleitet werden, wodurch es zu sich überlappenden Situationsbeschreibungen kommt (siehe Abbildung 11). Da diese Überlappungen für die Erkennung der korrekten Situation ausgeschlossen werden sollen, wird ein Fuzzy-System angewendet.

```
role: speaker
entities: p:Person
expressions:
  p.location has type ‘Speaker area’

role: audience member
entities: p:Person
expressions:
  p.location has type ‘Audience area’

role: presentation equipment
entities: c:Computer
expressions:
  presentation software is active
    application on c
```

```
situation: presentation
roles:
  s:speaker
  a:audience member
  e:presentation equipment
expressions:
  s.cardinality = 1
  a.cardinality >= 3
  e.cardinality >= 1
  ‘Room’ of s.p.location = ‘Room’ of a.p.location
  ‘Room’ of s.p.location = ‘Room’ of e.c.location
```

Abbildung 10. Einfache Situationsbeschreibung

Das Fuzzy-System wird angewendet, um die Ungenauigkeiten der abstrahierten Sensordaten zu verwenden. Dadurch werden Situationen mit einer bestimmten Wahrscheinlichkeit erkannt. Für jeden Eingabewert gibt es einen Ungenauigkeitsfaktor, der angibt, mit welcher Wahrscheinlichkeit die Sensordaten korrekt sind. Soll beispielsweise erkannt werden, ob in einem Raum eine Party stattfindet, so sind Informationen über die Position von Personen nötig. Werden sämtliche Positionsdaten vom selben System ermittelt, so könnte sich eine Wahrscheinlichkeit von $0,9$ ergeben, dass die Sensordaten korrekt sind. Ein besonderer Vorteil der Fuzzy-Logik bei der Interpretation solcher Daten ergibt sich

```
role: PhD supervisor
inherits: meeting attendee
entities: phd:Person
expressions: p supervises phd

role: PhD student
inherits: meeting attendee
expressions: p is a PhD student

customisation: John PhD meeting
customises: PhD meeting
expressions:
  sup.p.id = JOHN_ID ->
    'Room' of sup.p.location = L10.01
```

```
situation: PhD meeting
inherits: meeting
roles:
  stu:PhD student
  sup:PhD supervisor
expressions:
  stu.cardinality = 1
  sup.cardinality >= 1
  stu.p.id = sup.phd.id
  'Meeting area' of stu.p.location =
    'Meeting area' of sup.p.location
```

Abbildung 11. Beschreibung einer abgeleiteten Situation

darin, dass bei der Verknüpfung von Sensordaten entweder das Maximum oder das Minimum verwendet wird. Im Gegensatz dazu stehen andere Verfahren, die Wahrscheinlichkeiten multiplikativ verknüpfen, wodurch sich die Wahrscheinlichkeit einer Party mit zunehmender Personenanzahl verringern würde. Das System von Thomson verwendet monotone Zugehörigkeitsfunktionen, die eine Wahrscheinlichkeit zwischen 0 und 1 repräsentieren.

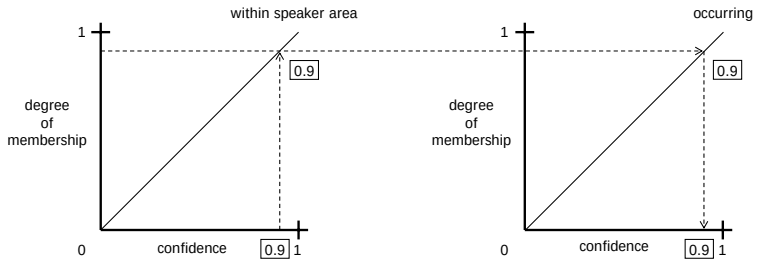

Abbildung 12. Monotone Zugehörigkeitsfunktion

Werden mehrere Sensordaten und deren Wahrscheinlichkeiten miteinander verknüpft, so können die oben spezifizierten Situationen mit einer bestimmten Wahrscheinlichkeit, wie in Abbildung 13 dargestellt, erkannt werden. Hierzu werden die Zugehörigkeitsfunktionen der relevanten Eingabedaten miteinander verknüpft, indem das Minimum der Ergebniswerte ermittelt wird. Aus der Berechnung geht der Eingabewert der Zugehörigkeitsfunktion der definierten Situation hervor. Durch Abbildung des Eingabewertes, mit Hilfe der Zugehörigkeitsfunktion der gewählten Situation, auf die Ausgabewerte lässt sich die Wahrscheinlichkeit ermitteln, mit der die entsprechende Situation stattfindet.

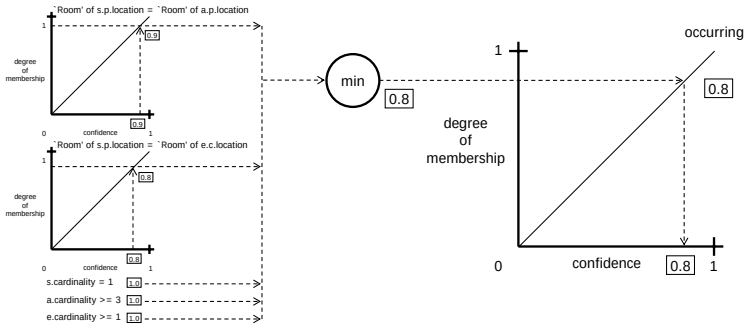

Abbildung 13. Erkennung komplexer Situationen

Thomson stellt im Weiteren eine Architektur vor, mit der die Erkennung von Situationen erreicht werden soll. Seine aus mehreren Agenten bestehende Architektur enthält Geräte-Agenten (DA) und Personen-Agenten (PSA). Ein Server-Agent (ASA) stellt die zentrale Sammelstelle der Kontextinformationen, die von kontextsensitiven Anwendungen (SAA) verwendet werden können, dar (siehe Abbildung 14).

Die Fuzzy-Komponente ist hier im Server-Agenten verankert. Dieser wertet die Wahrscheinlichkeiten der Sensordaten aus und gibt diese an Anwendungen weiter. Eine Referenzimplementierung von Thomson hat ergeben, dass die Realisierung eines solchen Systems mit geringer Rechenleistung möglich ist. Eine Erweiterbarkeit des Systems ist durch die Definition neuer Regeln in der Fuzzy-Komponente gegeben.

4.2 Künstliche neuronale Netze

Der Forschungsbereich der künstlichen Intelligenz beschäftigt sich unter anderem mit der Erkennung von Mustern durch neuronale Netze [RN03]. Neuronale Netze sind Nachbildungen des menschlichen Gehirns, wobei die grundlegende Funktion der Neuronen im Computer nachgebildet wird. Ein mögliches Einsatzgebiet von kleinen neuronalen Netzen, den Perzeptronen, ist die Bilderkennung. Hier werden Eingabedaten der sensorischen Einheiten durch die Eingabeschicht direkt zu Ausgabedaten der Ausgabeschicht umgewandelt (siehe Abbildung 15).

Im Gegensatz dazu steht ein mehrschichtiges neuronales Netz, das durch die Verknüpfung mehrerer neuronaler Zellen über ein System von Ebenen komplexe Eingabedaten zu Ausgabedaten umwandeln kann. Im Bereich der Kontexterkennung können neuronale Netze ebenfalls angewendet werden, da hier viele verschiedene Eingabedaten existieren, aus denen eine Kontextinformation gewonnen werden soll. Im Gegensatz zu anderen Verfahren, speziell auch zu dem

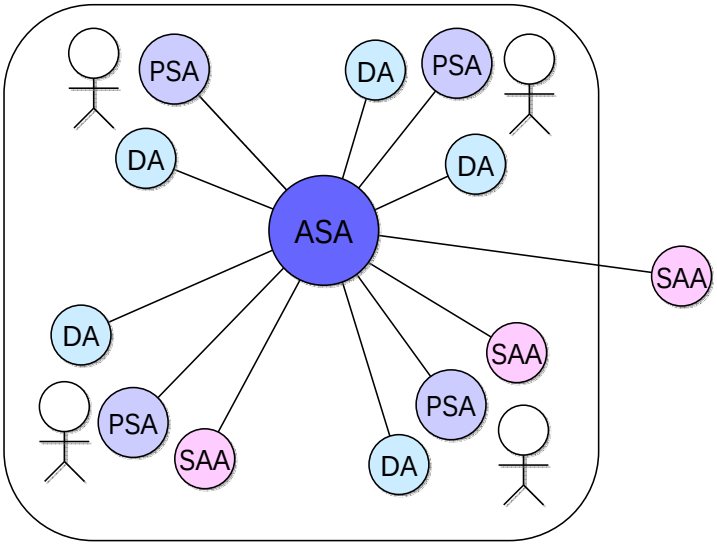

Abbildung 14. Architektur der Situationserkennung

oben vorgestellten Verfahren der Fuzzy-Regler, kann ein neuronales Netz lernen und somit sein Verhalten an neue Gegebenheiten anpassen.

Ein neuronales Netz besteht im wesentlichen aus drei Komponenten: Dazu gehören zum einen künstliche neuronale Zellen, die ihren Zustand aufgrund von gewichteten Eingaben berechnen. Des Weiteren spielt die Topologie des Netzes, d. h. die Anordnung und Verbindung der künstlichen neuronalen Zellen untereinander eine große Rolle. Hierbei wird zwischen teilweise und vollständig vernetzten Topologien unterschieden. Als letzte Komponente spielen die Gewichtungen der Eingabekanäle jeder künstlichen neuronalen Zelle eine entscheidende Rolle, da durch die Veränderbarkeit dieser Gewichte ein Lernprozess im neuronalen Netz stattfindet.

Eine künstliche neuronale Zelle i verarbeitet die gewichteten Eingabedaten zu einem Ausgabesignal net_i, der Propagierungsfunktion. Diese entsteht durch das Skalarprodukt aus Eingabedaten und entsprechenden Gewichtungen.

$$net_i = e_i \circ g_i$$

Das Ergebnis der Propagierungsfunktion fließt in die so genannte Aktivierungsfunktion F_i ein, welche den Aktivierungszustand einer künstlichen neuronalen Zelle, in Abhängigkeit der vorherigen Aktivierung angibt. Wird ein bestimmter Schwellenwert überschritten, so wird ein Ausgabesignal an alle nachgeschalteten künstlichen Neuronen gesendet.

$$A_i^t = F_i(net_i, A_i^{t-1})$$

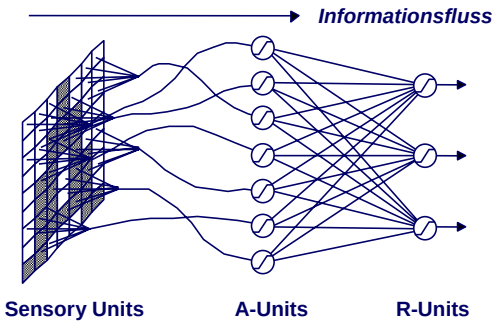

Abbildung 15. Aufbau eines Perzeptrons

Die Ausgabefunktion f_i bestimmt auf Basis der Aktivität einer Zelle, welches Signal a_i über die Ausgabeverbindung ausgegeben werden soll.

$$a_i = f_i(A^t)$$

Um ein Perzeptron auf eine bestimmte Aufgabe zu trainieren, muss das neuronale Netz zuerst eine Lernphase durchlaufen. Hier werden Eingabewerte vorgegeben und die daraus resultierenden Ausgabewerte mit den erwarteten Werten verglichen. Der Unterschied, der zwischen erwartetem und tatsächlichem Ausgabewert entsteht, ist der Fehler der Netzes. Dieser Fehler kann anschließend rückwärts in das Netz propagiert werden, damit jede am Fehler beteiligte Zelle ihre Gewichtungen so anpassen kann, dass der Fehler geringer wird. Eine Anpassung der Gewichtungen könnte mit folgender Formel geschehen, wobei α den Lernfaktor und damit die Lerngeschwindigkeit angibt.

$$g_{ij} = g_{ij} + \alpha \cdot Fehler_i \cdot e_i$$

Nach einer ausreichenden Lernphase entsprechen die Resultate in etwa den erwarteten Ergebnissen. Das Wissen, wie die Ausgabewerte erzeugt werden, ist in neuronalen Netzen nicht explizit definiert. Vielmehr bestimmt die Konfiguration der Topologie und der Gewichte, wie die Ausgabe erzeugt wird.

Da ein einschichtiges Perzeptron nur eine begrenzte Anzahl von Daten verarbeiten kann, kann man diese durch zusätzliche verborgene Schichten erweitern, um größere Mengen an Eingabedaten verarbeiten zu können. Hierbei bedarf es einer genauen Konfiguration bzgl. der Anzahl der Schichten und der Anzahl der Neuronen auf jeder Schicht. Im Unterschied zum Perzeptron wird die Aktivierungsfunktion mehrschichtiger neuronaler Netze mit Hilfe einer Sigmoidfunktion beschrieben:

$$A(x) = \frac{1}{(1 + e^{-x})}$$

Die verschiedenen Ausgaben von verborgenen Schichten sowie der Ausgabeschicht werden durch die Ausgabefunktion beschrieben.

$$Ausgabe := A(G \cdot Eingabe)$$

Einen wesentlichen Unterschied bildet die Lernregel, mit der während der Lernphase Fehler in das Netz zurückgegeben werden können, um Gewichtungen anzupassen. Jedes Neuron passt seine Gewichte entsprechend seinem Anteil am Gesamtfehler an. Dazu wird folgende delta-Lernregel verwendet:

$$delta := (Ergebnis - Ausgabe) \cdot A'(G \cdot Eingabe)$$

Die Anpassung der Gewichte erfolgt ähnlich zum Perzeptron durch folgende Formel:

$$g_i = g_i \cdot \alpha \cdot delta \cdot e_i$$

Grundsätzliche Schwierigkeiten existieren bei der Konstruktion von neuronalen Netzen. Die Anzahl der Schichten und Neuronen hat einen wesentlichen Einfluss auf die Funktion des Netzes. Standardmäßig wird bei neuronalen Netzen eine verborgene Schicht verwendet, über die Anzahl der Neuronen dieser Schicht kann man keine genauen Angaben machen. Je nach Anwendung unterscheiden sich diese Parameter. Grundsätzlich kann man jedoch sagen, dass bei zu wenigen neuronalen Zellen keine komplexen Funktionen erlernt werden können, sowie dass bei zu vielen Neuronen die Generalisierungsfähigkeit des Netzes, d. h. die Fähigkeit zur Ermittlung korrekter Ergebnissen bei nicht trainierten Eingabedaten, leidet. Ebenfalls zu beachten ist, dass die Anzahl der Ausgabewerte kleiner sein sollte als die Anzahl der Eingabedaten.

Konkrete Anwendung Eine konkrete Anwendung finden neuronale Netze, wenn eine Vielzahl unterschiedlicher Sensordaten zu einer komplexen Kontextinformation integriert werden soll. Herkömmliche Verfahren benötigen meist einen größeren Rechenaufwand für die Berechnung eines ähnlich genauen Ergebnisses, wohingegen künstliche neuronale Netze durch ihre parallele Berechnung und ohne die Kenntnis von Daten zur Berechnung, nur einen geringen Rechenaufwand erzeugen.

Neuronale Netze werden eingesetzt, um beispielsweise unter einer Menge von 400 verfügbaren Kommunikationsprovidern den Besten herauszusuchen. Jeder Provider ist durch neun Eigenschaften, zu denen unter anderem Entfernung, Kosten und Verfügbarkeit gehören, gekennzeichnet [AM06]. Somit kann aus den einfachen Kontextinformationen, die die Situation beschreiben, in der sich der Anwender zur Zeit befindet, eine komplexe Kontextinformation „aktuell bester Provider“ ermittelt werden.

Beispielhafte Anwendung Für unser Konferenzbeispiel kann man sich folgende Anwendung neuronaler Netze vorstellen: In unserem Konferenzraum seien Sensoren installiert, die Informationen zu den Eigenschaften des Raums bzgl. der folgenden Fakten ermitteln: Licht, Personenanzahl, Rauchgehalt der Raums, Umgebungslautstärke, Beamertemperatur, Nutzungsdauer des Videokonferenzsystems, Kohlendioxidgehalt der Luft und Blutdruck sämtlicher Personen. Diese acht Sensoren liefern ständig sehr unterschiedliche Daten, aus denen ermittelt

werden soll, ob in dem Raum zur Zeit eine Konferenz oder eine Party stattfindet. Dazu könnte ein neuronales Netz entwickelt werden, das als Ausgabe Werte zwischen 1 und 0 liefert. 1 bedeutet hierbei „Konferenz findet statt“, 0 bedeutet, „Party findet statt“. Eine beispielhafte Konfiguration des neuronalen Netzes ist in Abbildung 16 dargestellt.

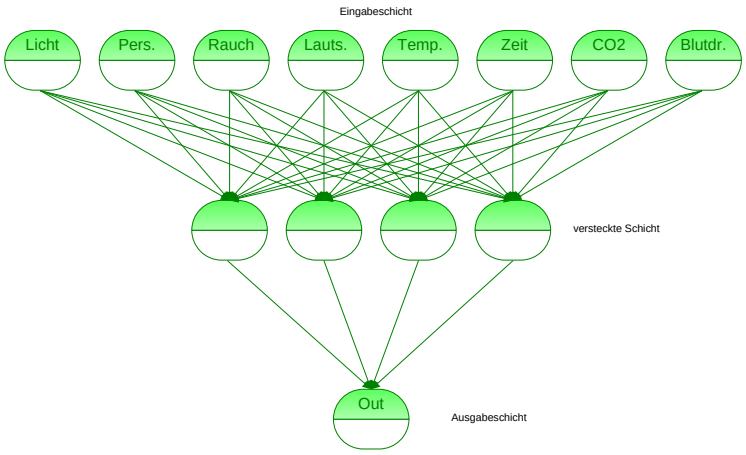

Abbildung 16. Beispielhaftes neuronales Netz für Kontexterkennung

Nach dem Training des neuronalen Netzes kann dieses die Sensordaten interpretieren. Der Vorteil des neuronalen Netzes liegt darin, dass bei der hohen Anzahl der Sensoren und dementsprechend hoher Anzahl möglicher Situationen, nicht alle möglicherweise auftretenden Situationen zuvor trainiert werden können, aber dennoch die Auswertung nicht trainierter Situationen möglich ist. Das neuronale Netz wird nach der Trainingsphase Werte zwischen 0 und 1 ausgeben. Hier könnte man entsprechend abschätzen, dass zwischen 0 und 0.5 eine Party, und zwischen 0.5 und 1 eine Konferenz stattfindet.

5 Ergänzung externer Daten

In den vorherigen Kapiteln haben wir uns damit beschäftigt, wie ein grundlegendes Kontextsystem aufgebaut sein kann. Weiterhin haben wir exemplarische Methoden zur Abstraktion von Sensordaten kennengelernt. Somit stehen uns zu diesem Zeitpunkt komplexe Kontextinformationen zur Verfügung, die von Anwendungen weiterverwendet werden können. Die Kontextinformationen alleine haben jedoch noch keinen großen Wert. Erst durch die Integration externer Daten und durch den aktiven Austausch der Kontextinformationen mit anderen

Systemen können Kontextinformationen effektiv weiterverwendet werden. Kontextsensitive Umgebungen werden in Zukunft nicht mehr nur in kleinen Testumgebungen vorhanden sein, sondern ein miteinander verknüpftes Netzwerk bilden. Die Kommunikationsplattform für ein solches Netzwerk besteht schon lange: Das Internet. Die vorhandene Infrastruktur des Internets, sowie die zunehmende Bandbreite und Verfügbarkeit der Anschlüsse wie z. B. WLAN, ermöglichen bald einen ständigen Zugriff auf das Internet. Laut Kotz wird das Internet in Zukunft exponentiell weiterwachsen, so dass viele Menschen einen ständigen Zugriff auf ein Hochgeschwindigkeitsnetz haben, das sie zu jeder Zeit mit einer Flut an Informationen versorgen kann [KG99]. Daher dient das Internet als Infrastruktur für die Versorgung kontextsensitiver Anwendungen mit zusätzlichen Daten.

5.1 Notwendigkeit externer Daten

Die Einbeziehung externer Daten ist für ein Kontextsystem von wesentlicher Bedeutung, da Kontextinformationen alleine noch keinen großen Nutzen aufweisen. In unserem anfänglichen Konferenzbeispiel werden externe Daten benötigt, um die Konferenzsituation zu erkennen und das Thema des Vortrags zu ermitteln. Dies geschieht mit Hilfe der Daten aus dem persönlichen Terminplan der teilnehmenden Personen. Durch die Einbeziehung dieser zusätzlichen externen Daten, kann die Situation detaillierter beschrieben werden.

Beim Einbezug externer Daten kann man zwischen Datenbeständen unterscheiden, die im Kontextsystem verfügbar sind, wie z. B. die Termindatenbanken der Mitarbeiter einer Firma, und Datenbeständen, die außerhalb des Kontextsystems liegen. Hierzu gehören im Internet verfügbare Datenbanken, wie z. B. digitale Straßenkarten oder Suchmaschinen. Datenbanken, die innerhalb eines Kontextsystems liegen, können leicht ausgelesen werden, da ein direkter Zugriff auf das System möglich ist und Schnittstellen bekannt sind. Anders sieht dies bei öffentlichen Datenbanken aus. Hier müssen Schnittstellen standardisiert sein, damit die Daten verwendet werden können.

5.2 Web Services

Standardisierte Schnittstellen zum Austausch von Daten existieren im Bereich der Web Services. Ein Web Service kann durch einen Client automatisch gefunden werden, er bearbeitet Anfragen eines Clients und ermöglicht den Austausch von Daten zwischen Client und Web Service. Web Services bieten somit unter anderem Dienstleistungen in Form von Informationen an.

Die Service-Umgebung Ein Web Service wird, wie in Abbildung 17 zu erkennen, durch drei Komponenten realisiert. Hierzu gehören Service-Broker, Service-Konsument und Service-Anbieter. Web Services können sich mit Hilfe des Universal Description, Discovery and Integration Standards (UDDI) beim Service-Broker anmelden. Der Service-Broker verwaltet sämtliche Web Services und ermöglicht somit ein Wiederauffinden eines Services durch einen Konsumenten.

Der Service-Anbieter stellt einen Web-Service zur Verfügung und beschreibt diesen mit Hilfe der Web Service Description Language (WSDL). Dieser auf XML basierte Standard beschreibt, welche Daten durch den Web Service verfügbar sind, welche Übertragungsprotokolle verwendet werden, sowie weitere Detailinformationen des Web Services. Als letzte Komponente existiert der Service-Konsument. Dieser möchte den Dienst des Service-Anbieters nutzen und muss daher eine Kommunikation mit dem Service-Anbieter aufbauen. Dies geschieht mit Hilfe des Simple Object Access Protocol (SOAP), einem Netzwerkprotokoll, mit dessen Hilfe Daten zwischen Systemen ausgetauscht und entfernte Prozeduren aufgerufen werden können [KK04].

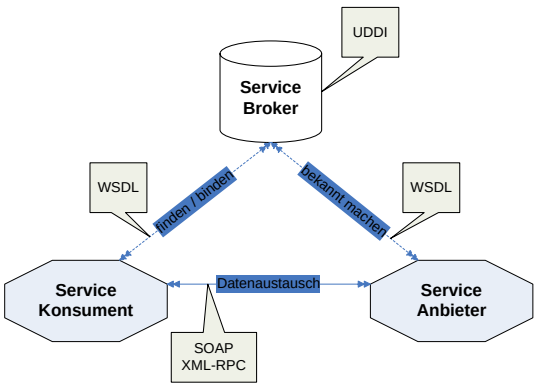

Abbildung 17. Komponenten eines Web Service

Trotz der vorhandenen Standards existieren Probleme, die den großflächigen Einsatz von Web Services bisher noch verhindern. Wesentliche Probleme bestehen im Bereich von Sicherheit und Privatsphäre, in der Verknüpfung mehrerer Web Services, sowie in der mangelnden Unterstützung von Semantik durch Web Services [WH04].

Web Services tauschen Daten über SOAP aus. SOAP basiert jedoch auf dem HTTP-Protokoll und ist daher nicht gesichert. Eine Anwendung einer sicheren Verbindung über HTTPS bzw. SSL ist nicht möglich, da die vorliegende Kommunikation nicht eine Ende-zu-Ende-Verbindung ist, sondern Daten von einer Anwendung zur nächsten übertragen werden. Die Sicherung der Verbindung durch eine Firewall ist ebenfalls nicht möglich, da der HTTP-Port üblicherweise nicht durch die Firewall geschützt ist. Daher sind Web Services auf Anwendungsebene sämtlichen Angriffen von außen ausgesetzt.

Weiterhin ist eine Komposition verschiedener Web Services durch den Web-Service-Standard nicht definiert. Da in heutigen Geschäftsprozessen jedoch oftmals mehrere Partner zusammenarbeiten, ist eine Verknüpfung mehrerer Web Services notwendig.

Als letzter Punkt sei die mangelnde Semantik von Web Services genannt. Web Services sind durch den WSDL-Standard beschrieben. Dieser definiert unter anderem, welche Daten der Web Service als Eingabe benötigt, bzw. als Ausgabe liefert. Nicht definiert ist die Semantik, die eine Interpretation der Daten ermöglicht.

Web Services und Kontextinformationen Im Rahmen dieser Arbeit erscheint es nicht nur interessant, wie externe Daten verwendet werden können, sondern auch in welchem Zusammenhang gewonnene Kontextinformationen mit der Gewinnung externer Daten stehen. Kontextinformationen werden auf Seite des Anwenders – sprich seitens des Service-Konsumenten – ermittelt. Hier existieren Sensoren und Verfahren zur Sensorabstraktion. Die externen Daten liegen typischerweise auf der Seite des Service-Anbieters und somit entfernt von den ermittelten Kontextinformationen. Durch den Austausch von Kontextinformationen könnte jedoch nicht nur die Ermittlung der externen Daten verbessert werden, sondern auch direkt der Kontext des Service-Konsumenten aktualisiert sowie erweitert werden.

Die Implementierung eines Kontext-Frameworks von Keidl, das auf der Anwendung von Web Services basiert, ermöglicht die Verwendung von Kontextinformationen [KK04]. Grundidee des Frameworks ist, die Bearbeitung von Kontextinformationen nicht vom Web Service selbst durchführen zu lassen. Statt dessen werden Schnittstellen zu Kontext-Plugins und Kontext-Services verwendet, um Kontextinformationen zu verarbeiten. Diese Ablösung der Kontextverarbeitung vom Web Service hat den Vorteil, dass die Kontextkomponenten auch von anderen Web Services verwendet werden können, und der Änderungsaufwand bei der Entstehung neuer Anforderungen an den Web Service gering bleibt.

Die Kontextinformation des Kontext-Konsumenten wird mittels einer erweiterten SOAP-Nachricht an den Web Service übermittelt. Die standardmäßig in die beiden Teile Header und Body eingeteilte SOAP-Nachricht wird so modifiziert, dass der SOAP-Header zusätzliche Kontextinformationen übertragen kann. Die Verwendung des Headers zur Übertragung von Kontextinformationen hat den Vorteil, dass bei der Anfrageverarbeitung durch Web Services, die Kontextinformationen nicht verarbeiten können, dieser Teil der Nachricht, anders als beim SOAP-Body, ignoriert werden kann. Der SOAP-Kontext-Header kann aus mehreren Kontext-Blöcken bestehen, die wiederum durch bestimmten Kontext-Typ, z. B. eine Ortsangabe, definiert werden (siehe Abbildung 18).

Die Kontextinformation des Konsumenten kann durch den Web Service bearbeitet werden. Die übertragenen Originaldaten können durch Daten des Web Services angereichert oder modifiziert werden. Nach der vollständigen Bearbeitung durch den Web Service wird der veränderte Kontext wieder an den Konsumenten zurückgesendet. Dieser kann dann entscheiden, welche Teile des veränderten Kontextes in den aktuellen Kontext übernommen werden. Bei der Bearbeitung von Kontextinformationen durch den Web Service unterscheidet das Framework zwischen der *expliziten* und der *automatischen* Bearbeitung der Informationen.

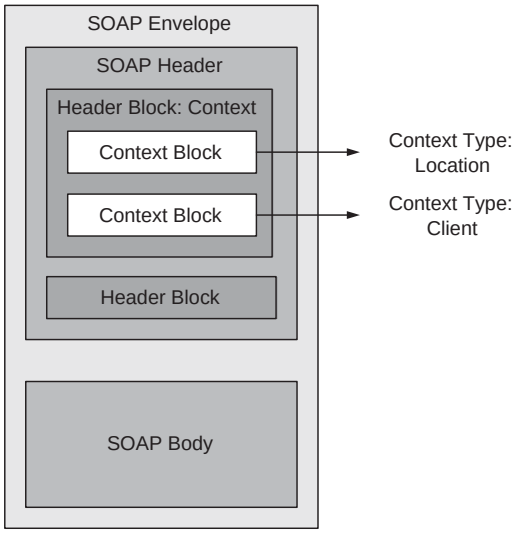

Abbildung 18. Kontext in einer SOAP-Nachricht

Werden Kontextinformationen explizit durch den Web Service bearbeitet, so geht die oben angesprochene Wiederverwendbarkeit und Erweiterbarkeit der Komponenten verloren, da Verarbeitung im Web Service selbst definiert ist. Bei der automatischen Bearbeitung werden Kontextinformationen durch die dem Web Service zugrunde liegende Web-Service-Plattform weiterverarbeitet (siehe Abbildung 19). Hierbei werden die Daten des SOAP-Headers bei der Übertragung vor- bzw. nachbearbeitet.

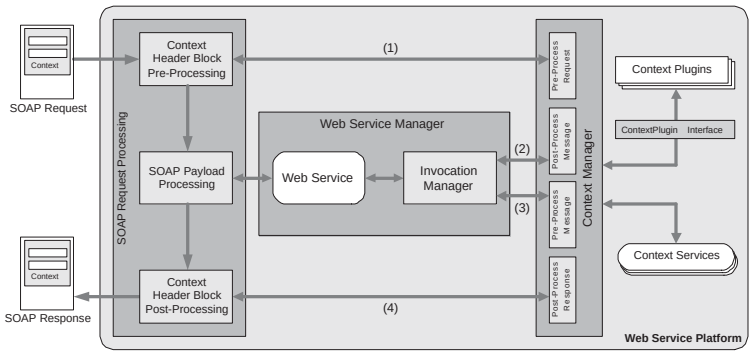

Abbildung 19. Keidels Web-Service-Plattform

Bearbeitet die Web-Service-Plattform die Kontextinformationen, so werden sowohl Kontext-Plugins als auch Kontext-Services verwendet. Kontext-Plugins werden in einen Web Service registriert und müssen lokal verfügbar sein. Es besteht dennoch die Möglichkeit, Kontext-Plugins in verschiedenen Web Services zu verwenden, wenn lokale Kopien erstellt werden. Im Gegensatz dazu sind Kontext-Services im Internet verfübar und in einem UDDI-Verzeichnis registriert. Ein Web Service, der der Web-Service-Plattform zugrunde liegt, hat demnach die Möglichkeit, Kontextinformationen mit Hilfe bereits bestehender Web Services zu verarbeiten. Hierzu reicht eine Anfrage an das UDDI-Verzeichnis, um entsprechende Services aufzufinden. Im Gegensatz zu herkömmlichen Methoden, bei denen die verwendeten Web Services fest kodiert sind, bedient sich Keidl einer dynamischen Serviceauswahl [Ke04]. Jeder Web Service ist im UDDI-Verzeichnis mit einem so genannten *tModel* registriert. Dieses definiert Eingaben und Ausgaben eines Services. Benötigt ein aufgerufener Web Service weitere Services, so stellt dieser eine Anfrage an das UDDI-Verzeichnis, und wählt mit Hilfe von Bedingungen passende Services aus. Als Bedingungen können beispielsweise der Ort des Webservice oder die Rückgabewerte eines Web Service dienen. Des Weiteren schafft die dynamische Serviceauswahl Redundanzen, die beim Ausfall einzelner Services oder bei Unerreichbarkeit des Netzwerkes eine weitere Bearbeitung ermöglichen.

Screenscraping Web Services Web Services bieten standardisierte Schnittstellen, um Informationen verfügbar zu machen. Betrachtet man das Internet als globalen Informationsspeicher, so muss man feststellen, dass nur ein geringer Teil der verfügbaren Informationen strukturiert oder sogar in Form eines Web Services vorliegt. Einschlägige Datenquellen wie z. B. Google oder Amazon bieten bereits Schnittstellen in Form von Web Services an. Dennoch bleibt der größte Teil des Internets hinter unstrukturierten Webseiten verborgen und ist somit für die Anreicherung von Kontextinformationen nicht verfügbar.

Eine mögliche Lösung für die Einbeziehung von Informationen statischer Webseiten ist die so genannte Screenscraping-Technologie, bei der eine Software Daten verschiedener semi-strukturierter Webseiten sammelt und strukturiert ausgibt. Die Entwicklung eines Web Service, der eine Screenscraping-Komponente enthält, ermöglicht die Verbindung von Web Service, Kontextanreicherung und statischer, semi-strukturierter Webseiten. Der von Oostenrijk exemplarisch entwickelte Web Service ermöglicht, Daten eines Vorlesungsverzeichnisses zu analysieren und bereitzustellen. Probleme ergeben sich hierbei jedoch in der Geschwindigkeit der Anfragebearbeitung, da durch die interne Verlinkung vieler Webseiten ein hohes Datenvolumen entsteht, das eine zeitnahe Gewinnung zusätzlicher Daten nahezu unmöglich macht [Oo04].

5.3 Szenario

Auch in unserem anfänglichen Konferenzbeispiel lässt sich die Bedeutung externer Datenquellen zur Kontextanreicherung erläutern. Nachdem das vorhandene

Kontextsystem erkannt hat, dass eine Besprechung stattfindet, könnten Informationen über den Vortragenden gesammelt werden. Da sich die Teilnehmer untereinander nicht kennen, sollen die gesammelten Informationen bereitgestellt werden, um eine Übersicht über die verschiedenen Arbeitsfelder zu geben. Hierfür existieren verschiedene Datenquellen. Eine systeminterne Datenbank gibt Aufschlüsse über die Termine der Person und wann diese für Fragen zur Verfügung stehen kann. Eine externe Datenquelle, die durch einen Web Service repräsentiert wird, liefert weitere Informationen zum Vortragsthema und erläutert beispielsweise spezielle Fachbegriffe. Als letztes wird die in Form einer HTML-Seite verfügbare Liste der Veröffentlichungen des Vortragenden durch einen Screenscraper analysiert und strukturiert. Diese Daten werden zusammengefasst und den Teilnehmern als erweiterter Kontext präsentiert.

Dieses Szenario stößt so jedoch an die Grenze des Machbaren. Auch wenn die einzelnen Technologien zu Ermittlung von externen Daten anhand der vorliegenden Kontextinformationen existieren, so wird es schwierig, einem solchen System zu erklären, wann es welche Dokumente verwenden soll. Der Computer weiß im Gegensatz zum Menschen nicht, welche Informationen nützlich sind, und welche nicht. Eine Zusammenfassung von Informationen ist so leicht nicht umzusetzen. Auch hier müsste eine Maschine wichtige von unwichtigen Informationen unterscheiden können – und das ist ebenfalls noch Zukunftsmusik.

6 Dumme Menschen, dümmere Computer!

Computer sollen das Leben des Menschen erleichtern. Wir Menschen erfinden jeden Tag neue Technologien, die diesen Zweck erfüllen sollen. Dasselbe Ziel hat auch die Umsetzung von kontextbewussten Systemen. Im Laufe dieser Seminararbeit habe ich herausgestellt, was genau ein Kontext ist und wie ein kontextbewusstes System arbeitet. Viele bestehende Frameworks bauen aufeinander auf, um die Implementierung solcher Systeme zu vereinfachen. Viele Probleme, wie z. B. unzureichende Konnektivität, unhandliche mobile Endgeräte und unzureichende technologische Entwicklungen treten immer weiter in den Hintergrund. Methoden zur Sensorabstraktion sowie zur Integration externer Daten sind seit langem bekannt. Dennoch – die Eingangsfragen sind immer noch nicht beantwortet: Warum werden kontextsensitive Systeme nicht eingesetzt? Ist nicht der Computer zu dumm, sondern der Mensch, weil er sich sein Leben schwerer macht als nötig? Wollen wir Menschen überhaupt, dass ein Computer Informationen über uns sammelt und an Dritte weitergeben kann? Wollen wir einem Computersystem die Kontrolle über unsere Umgebung überlassen? Können wir einem solchen System überhaupt vollständig vertrauen, angesichts der Tatsache, dass Software immer fehlerbehaftet ist? Wo genau ist der Unterschied zwischen Kontexterkennung und ständiger Überwachung?

Meiner Meinung nach sind diese wesentlichen Fragen nicht einfach zu beantworten. Tatsache ist jedoch, dass durchaus technologische Entwicklungen existieren, um kontextsensitive Systeme auch in größeren Gebieten zu installieren. Daher scheint für mich ein Hauptproblem in der mangelnden Akzeptanz durch den

Anwender selbst zu bestehen. Erst wenn die Ängste der Anwender überwunden sind, sowie verlässliche Standards für die Wahrung der Privatsphäre existieren, wird es meiner Meinung nach einen Markt für kontextsensitive Anwendungen geben und werden so genannte Killer-Applikationen entwickelt werden, die den vermehrten Einsatz solcher Systeme vorantreiben.

Ein weiteres Problem liegt jedoch auch auf der Seite der Computer. Wie soll ein Computer entscheiden können, welche Information in welchem Moment für den Menschen von Nutzen ist? Wie soll ein Computer wesentliche von unwesentlichen Informationen unterscheiden? Wie soll uns ein Computer die Arbeit abnehmen, Informationen aus einem Text zu entnehmen und auf ein Minimum zusammenzufassen? Hier existieren bisher noch keine ausgereiften Lösungen, um dem Menschen Arbeit abzunehmen.

Bis diese und andere Probleme behoben sind, bleibt uns anscheinend nichts anderes übrig, als sich damit abzufinden, dass auch in Zukunft in fast jeder Besprechung mindestens ein Handy klingelt.

Literatur

[We91] Weiser, M.: The Computer for the 21st Century, Scientific American 265, No. 3, pp. 94–104, 1991

[De00] Dey, A.: Providing Architectural Support for Building Context-Aware Applications, Georgia Institute of Technology, 2000

[DA99] Dey, A., Abowd, G.: Towards a Better Understanding of Context and Context-Awareness, In Handheld and Ubiquitous Computing: First International Symposium, pp. 304, 1999

[CF03] Chen, H., Finin, T., Joshi, A.: An Intelligent Broker for Context-Aware Systems, University of Maryland, 2003

[CF04] Chen, H., Finin, T., Joshi, A.: A Context Broker for Building Smart Meeting Rooms, In Proceedings of the Knowledge Representation and Ontology for Autonomous Systems Symposium, pp. 53–60, 2004

[Sc95] Schilit, W.: A System Architecture for Context-Aware Mobile Computing, Doctoral Thesis. UMI Order Number: UMI Order No. GAX95-33659, Columbia University, 1995

[LY02] Luo, R., Yih, C., Su, K.: Multisensor Fusion and Integration: Approaches, Applications, and Future Research Directions, IEEE Sensors Journal, vol. 2, no. 2, pp. 107–119, 2002

[Mä03] Mäntyjärvi, J.: Sensor-based Context Recognition for Mobile Applications, University of Oulu, VTT Electronics, Espoo. 118 p. + app. 60 p. VTT Publications : 511, 2003

[MS02] Mäntyjärvi, J., Seppänen, T.: Adapting Applications in Mobile Terminals Using Fuzzy Context Information, In Mobile HCI, LNCS 2411, Springer-Verlag Berlin Heidelberg pp. 95–107, 2002

[BV02] Battiti, R., Villani, A., Nhat, T.: Neural Network Models for Intelligent Networks: Deriving the Location from Signal Patterns, Universita di Trento, In Proceedings of AINS, 2002

[TT06] Thomson, G., Terzis, S., Nixon, P.: A Model and Architecture for Situation Determination, University of Strathclyde, Glasgow, UK, In Proceedings of the Conference of the Center For Advanced Studies on Collaborative Research, ACM Press, New York, 2006

[RN03] Russel, S., Norvig, P.: Artificial Intelligence: A Modern Approach, Prentice Hall, 2003

[AM06] Al-Masri, E., Mahmoud, Q.: A Context-Aware Mobile Service Discovery and Selection Mechanism using Artificial Neural Networks, University of Guelph, In Proceedings of the 8th international Conference on Electronic Commerce, vol. 156. ACM Press, New York, pp. 594–598, 2006

[KG99] Kotz, D., Gray, R.: Mobile Agents and the Future of the Internet, Dartmouth College, SIGOPS Oper. Syst. Rev. 33, 3, pp. 7–13, 1999

[KK04] Keidl, M., Kemper, A.: Towards Context-Aware Adaptable Web Services, Universität Passau, WWW2004, New York, USA, ACM 1-58113-912-8/04/0005, 2004

[Ke04] Keidl, M.: Metadata Management and Context-based Personalization in Distributed Information Systems, Technische Universität München, 2004

[WH04] H. Wang, J. Huang, Y. Qu, J. Xie: Web Services: Problems and Future Directions, Journal of Web Semantics, vol. 1, pp. 17, 2004

[Oo04] Oostenrijk, A.: Screen Scraping Web Services, Department of Computer Science, Radboud University of Nijmegen, 2004