Kontextsensitive Systeme – Stand der Technik

Bernhard Ehringer
Fachhochschule Oberösterreich – Fakultät für Informatik/Kommunikation/Medien
Masterstudiengang Mobile Computing
Softwarepark 11, 4232 Hagenberg, Österreich
bernhard.ehringer@fh-hagenberg.at

KURZFASSUNG

Systeme, welche den Kontext – also die aktuelle Situation des Benutzers – in Entscheidungen, Darstellungsformen oder Verhaltensmuster miteinbeziehen werden immer beliebter. Dies hat einerseits den Grund, dass es bereits ein breites Spektrum an Anwendungen – z. B. Google Latitude – gibt, welche etwa die Positionsinformation nutzen, um diverse Dienste anzubieten. Andererseits wird es Softwareentwicklern durch das Bereitstellen verschiedener Frameworks immer leichter gemacht, innovative kontextsensitive Anwendungen zu entwickeln.

Um als Anwendungsentwickler jedoch mit den Begrifflichkeiten *Kontext* und *Kontextsensitivität* vertraut zu werden, ist es in einem ersten Schritt notwendig, sich mit den diversen Definitionen zu beschäftigen. In in weiterer Folge kann ein Modell zum Abbilden der Kontextinformationen sowie das kontextsensitive System selbst – bei Bedarf basierend auf ein existierendes Framework – erstellt werden, wobei die bestehenden Konzepte zuerst geprüft und Vor- bzw. Nachteile gefunden werden müssen.

Dieser Artikel hat nun zum Ziel, genau diese Fragestellungen zu klären – Begriffe rund um das Thema Kontextsensitivität zu definieren sowie bestehende Datenmodelle und Frameworks vorzustellen, welche benutzt werden können, um eigene kontextsensitive Anwendungen zu erstellen.

1. EINLEITUNG

In viele Bereichen des täglichen Lebens können kontextsensitive Systeme dem Menschen helfen, den Tagesablauf zu planen, wichtige Entscheidungen richtig zu treffen und keine Termine zu versäumen. Solche Systeme können die unterschiedlichsten Ausprägungen haben – jedoch haben sie alle etwas gemeinsam: Das Benutzen von Informationen der aktuellen Situation – sei es die Position, die Temperatur, vorhandene Umgebungsgeräusche oder die aktuelle Tageszeit.

Da es aufgrund bestehender Frameworks, welche einen Großteil der benötigten Funktionen besitzen, nicht unbedingt notwendig ist, eine komplett eigene Systemarchitektur zu schaffen, ist es meist sinnvoll, ein vorhandenes Framework zu nutzen. Eine Auflistung sowie eine detaillierte Beschreibung dieser verschiedenen Frameworks zu geben, ist das Hauptziel dieser Arbeit. Des Weiteren sollen wichtige Begriffe definiert und Möglichkeiten zur Kontextmodellierung sowie Beispielprojekte vorgestellt werden.

Dieser Artikel stellt in Kapitel 2 zuerst einige wichtige Begrifflichkeiten vor. In einem weiteren Abschnitt werden Modellierungsmöglichkeiten vorgestellt, welche in Systemen mit Kontextbezug oft Verwendung finden. Kapitel 4 beschreibt existierende Frameworks und diverse Architekturmöglichkeiten, um Systeme dieser Art umzusetzen. Vor dem Resümee werden in Kapitel 5 noch einige bestehende Anwendungsbeispiele präsentiert.

2. GRUNDLAGEN

Im Vorfeld dieses Artikels werden – um eine einheitliche Auffassung diverser Begrifflichkeiten zu haben – einige dieser Fachbegriffe genauer definiert sowie deren Bedeutung erläutert.

2.1 Kontext

Der Begriff *Kontext* ist in der Literatur nur vage definiert und wird von vielen Menschen unterschiedlich interpretiert. Da alles und jedes in einem gewissen Kontext abläuft, ist es relativ schwierig, eine allgemeine Definition zu finden. Meist wird der Kontext mit dem Umfeld, in dem sich ein Objekt – etwa eine Person – befindet, assoziiert. Eine oft zitierte Definitionen des Begriffs wurde von Schilit [20] aufgestellt, welche sich auf drei wichtige Aspekte stützt – den Ort an dem man sich befindet, die Personen die sich in der Nähe aufhalten und die zur Verfügung stehenden Ressourcen. Dies besagt bereits, dass nicht nur der Ort eine Rolle spielt, sondern auch weitere nicht immer gleich bleibende Aspekte wie etwa der Lärmpegel, die Lichtverhältnisse oder die aktuelle soziale Situation den Kontext beeinflussen.

Des Weiteren wird der Begriff *Kontext* von Schilit in drei verschiedene Kategorien unterteilt, welche von Chen und Kotz [7] – fokussierend auf den Anwendungsbereich Mobile Computing – um zwei weitere Bereiche (*Time context*, *Context history*) ergänzt werden:

- **Computing context:** Informationen über zur Verfügung stehende Computer (z. B. Displayauflösung)
- **User context:** Informationen über den Benutzer (z. B. soziale Situation)

- **Physical context:** Physische Bedingungen (z. B. Vibration, Temperatur, Wetter)
- **Time context:** Zeitliche Informationen zur aktuellen Situation (z. B. Wochentag)
- **Context history:** Kontext-Informationen, die frühere Zeitpunkte beschreiben

Außerdem teilen sie den den Begriff des Kontext im Rahmen von Mobile Computing in zwei Arten – einerseits den aktiven Kontext, dessen Aspekte wichtig für die mobile Anwendung sind und diese direkt beeinflussen. Als zweite Art gibt es den passiven Kontext, welcher alle Aspekte des Umfelds indkludiert, die zwar wichtig sind, aber die Anwendung nicht direkt beeinflussen.

Als dritte betrachtete Definition von Kontext, welche auch im Rahmen dieser Arbeit verwendet wird, dient die von Dey und Abowd [2]. Dabei wird Kontext als jegliche Art von Information gesehen, welche die Situation einer Entität beschreibt. Eine Entität stellt in diesem Fall eine für die Interaktion zwischen Benutzer und Anwendung wichtige Person, einen Ort oder ein Objekt dar.

Neben der Begriffsdefinition wird von Dey und Abowd außerdem die Festlegung von Kontext-Kategorien als wichtig angesehen – vor allem für Anwendungsentwickler. Deshalb wurden folgende Kategorien definiert, welche hierbei auch als Primär-Kontext bezeichnet werden:

- **Lokalität:** Informationen zum (situationsbedingten) Ort
- **Identität:** Informationen über die im Mittelpunkt stehende Entität
- **Aktivität:** Informationen über die Aktionen, welche in der Situation passieren
- **Zeit:** Informationen über den aktuellen Zeitpunkt

Als zweite Schicht unter diesen vier Kategorien gibt es den Sekundär-Kontext, welcher aus Attributen besteht und über die Kategorien identifiziert werden kann. So ist etwa die Adresse einer Identität ein Element es Sekundär-Kontexts, welches über die Identität selbst (etwa den Namen in einem Telefonbuch) erreicht werden kann.

Da es in der Aufbereitung von Kontextinformationen diverse Fehlerquellen gibt ist es ebenfalls notwendig, Aussagen über die Qualität der Daten (Quality of Context, QoC) treffen zu können. Dabei wird von Buchholz die Qualität des Kontexts als Information bezeichnet, welche die Qualität der kontextbezogenen Daten beschreibt [6]. Jedoch ist hierbei zu beachten, dass sich die Qualität des Kontexts hierbei nur auf die Information selbst bezieht, nicht jedoch auf etwaige Komponenten, welche die Daten liefern.

2.2 Kontexterfassung

Die Grundlage einer jeden kontextsensitiven – also situationsangepassten – Anwendung bilden die Kontextelemente selbst, welche meist durch Key-Value-Paare abgebildet werden und mittels verschiedener Techniken erfasst werden können. Dabei muss laut Fuchs [11] zwischen Low- und High-Level-Kontextinformationen unterschieden werden.

Low-Level-Kontextinformationen stellen dabei Messwerte von Hardware-Sensoren dar. Des Weiteren werden die Low-Level-Kontextinformationen von Fuchs in mehrere Arten unterteilt:

- **Position:** Die Position stellt in vielen kontextsensitiven Systemen eine zentrale Rolle dar. Die nötigen Informationen können dabei unter Anderem mit Hilfe einer satellitenbasierter Ortung (GPS, Galileo) oder einer zellbasierten Lokalisierung (GSM/UMTS, WLAN) erfasst werden.
- **Zeit:** Sensoren zur Messung bzw. Erfassung der aktuellen Zeit sind in nahezu jeglichen elektronischen Geräten vorhanden – somit auch auf für kontextsensitive Systeme meist verwendeten Mobilgeräten.
- **Optische Informationen:** In diesem Bereich werden etwa Helligkeitssensoren eingesetzt, um zum Beispiel zwischen Tag und Nacht zu unterscheiden.
- **Akustische Informationen:** Um etwa auf Geräusche der Umgebung reagieren zu können, werden zumeist Mikrofone zur Messung der Akustik verwendet.
- **Sonstige:** Weitere für spezielle Anwendungsfälle interessante Informationen liefern etwa Pulsoxymeter, Beschleunigungssensoren oder berührungssensitive Elemente.

Unter High-Level-Kontextinformationen beschreibt Fuchs Werte, die nicht direkt von einem Sensor stammen sondern durch die Verarbeitung der Low-Level-Informationen entstehen. Beispiele dafür wären die aktuelle soziale Situation des Benutzers, die Gemütslage oder die gerade ausgeführte Tätigkeit. Da diese High-Level-Kontextinformationen jedoch mit Hilfe komplexer Regeln und/oder künstlicher Intelligenz erstellt werden, ist es sehr problematisch die richtigen Entscheidungen zu treffen.

Neben der Erfassung bzw. Erstellung von Low- und High-Level-Kontextelementen kann ebenfalls der Austausch von Daten mit anderen Systemen angedacht werden. Dies bedeutet etwa die Kontaktaufnahme mit anderen Computern, welche etwa mit Sensoren ausgestattet sind, die das eigene System nicht bietet. Nach dem Miteinbeziehen der räumlichen Differenz der beiden Systeme können beliebige Sensorwerte zwischen den Anwendungen ausgetauscht werden. Die so erhaltenen Daten können genauso wie durch eigene Sensoren erfasste Low-Level-Kontextinformationen behandelt und verarbeitet werden.

Sämtliche auf diese Arten erfassten Kontextinformationen bilden gemeinsam eine Kontextsituation, welche alle zu einem diskreten Zeitpunkt vorhandenen Umgebungseigenschaften inkludieren. Ein Beispiel dazu findet sich in Abbildung 1.

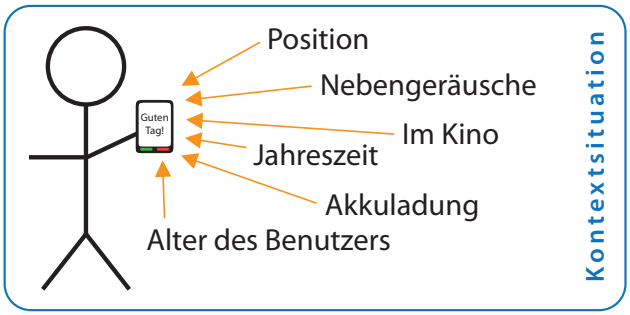

Abbildung 1: Erfassung der Kontextsituation

Dabei ist zu beachten, dass diese Kontextsituation im Laufe der Zeit Gültigkeit verliert, da sich der Kontext rundherum – eventuell sogar relativ schnell – ändert. Neben der aktuellen Kontextsituation kann auch der Verlauf bzw. die zeitlichen Veränderungen für die kontextsensitive Anwendung relevant sein. Dazu ist es möglich, einzelne Kontextinformationen oder die gesamte Kontextsituation zu diskreten Zeitpunkten persistent abzulegen. Somit kann auch später Zugriff auf bzw. Verarbeitung von früher aufgezeichneten Daten erfl

2.3 Kontextsensitivität

Der Kontext – also die Informationen einer bestimmten Situation – können nun herangezogen werden, um eine Anwendung oder ein System zu beeinflussen. Diese Beeinflussung bzw. Anpassung stellt die Kontextsensitivität dar und wird von Schilit [20] im Bezug auf Applikationen folgendermaßen definiert:

Kontextsensitive Anwendungen sind Systeme, die sich an den Einsatzort, die in der Umgebung befindlichen Personen und Ressourcen sowie die kontinuierliche Veränderung des Kontexts anpassen.

Des Weiteren definiert Schilit Kategorien kontextsensitiver Anwendungen, welche sich danach unterscheiden, entweder Informationen zu liefern oder Aktionen auszuführen sowie dies entweder manuell durch den Benutzer oder automatisch auszuführen.

Dey und Abowd [2] wiederum definieren kontextsensitive Systeme als Anwendungen, welche bestimmte Kontextinformationen nutzen, um dem Anwender situationsbedingt Informationen bzw. Dienste bereitzustellen. Dabei werden des Weiteren drei wichtige Funktionen kontextsensitiver Anwendungen aufgestellt, welche nahezu alle Systeme dieser Art bieten:

- Darstellung von Informationen bzw. Diensten
- Automatische Ausführung von Diensten
- Zuordnung von Kontextinformationen zu bestimmten Situationen zur späteren Verwendung

Des Weiteren wird Kontextsensitivität von Chen und Kotz [7] in zwei verschiedene Arten unterteilt – je nachdem wie die Kontextinformationen genutzt werden. Einerseits gibt es dabei die aktive Kontextsensitivität, welche dazu führt, dass sich Anwendungen automatisch an die ermittelte Kontextsituation anpassen. Die zweite Art – die passive Kontextsensitivität – dient hingegen nur dazu, die erforschten Kontextinformationen dem Anwender zu präsentieren bzw. diese persistent für einen späteren Zugriff abzulegen.

Ein Beispiel einer solchen aktiven Kontextsensitivität zeigt Abbildung 2 dar.

Abbildung 2: Aktive Kontextsensitivität

Dabei ist ersichtlich, dass die Anwendung basierend auf einem eingelesenen Kontextelement adaptiert wird – hierbei in Form des Begrüßungstextes je nach Tageszeit.

3. KONTEXT-MODELLIERUNG

Um Kontextinformationen in einer Anwendung darstellen und zwischen verschiedenen Systemen austauschen zu können, ist es notwendig, diese in einer einheitlichen Struktur abzulegen. Solche Strukturen enthalten i. d. R. den Typ der Kontextinformation, einen aktuellen Wert sowie Zusatzinformationen (Metadaten). Diese Metainformationen stellen strukturierte Daten über ein Objekt dar, mit Hilfe derer Funktionen ausgeführt werden können – etwa das Suchen von Personen mittels der Metainformation „Vorname“ [12].

Laut Strang und Linnhoff-Popien [23] gibt es einige weit verbreitete Datenmodelle, welche in den folgenden Abschnitten genauer vorgestellt werden.

3.1 Key-Value-Modelle

Dieses Datenmodell stellt ein rudimentäres und sehr einfaches Konzept dar, wie Kontextinformationen in Datenstrukturen abgebildet werden können. Dabei besteht jede Dateneinheit aus einem Key (Name der Kontextinformation) und einem Wert. Diese Art der Repräsentation der Informationen wurde zu Beginn der Entwicklung kontextsensitiver Systeme eingesetzt und findet bis heute vor allem in verteilten Systemen Verwendung [19]. Der Nachteil dieser Darstellungsform ist jedoch, dass diese für hochentwickelte Systeme keine geeignete Basis bieten, da hierbei der Wert der Kontexteigenschaft alleine nicht ausreicht. Ein Beispiel dieses Modells zeigt Abbildung 3.

```
<Position>48.36856;14.51409</Position>
<Date>2009-02-13</Date>
```

Abbildung 3: Key-Value-Modell

3.2 Markup-Scheme-Modelle

Im Unterschied zu den zuvor beschriebenen Modellen sind Markup-Scheme-Modelle hierarchisch aufgebaut und beinhalten Tags mit Attributen und Werten. Dies bedeutet, dass Kontextelemente somit durch meherere Werte beschrieben werden können. Beispiele für diese Art der Darstellung sind so genannte Profile – etwa das User Agent Profile (UAProf) oder das Composite Capabilities/Preferences Profile (CC/PP). Zu diesen Profilen, welche meistens XML-typisch aufgebaut sind, gibt es außerdem einige Erweiterungen. Diese bieten mehr Attribute und Möglichkeiten, Kontextdaten darzustellen. Abbildung 4 zeigt ein Beispiel einer Modellierung von Kontextdaten auf diese Weise.

```
<Hardware>
  <Screen size="48.36856;14.51409" colors="16mio" />
</Hardware>
<Software>
  <Platform os="Windows XP" memory="2gb" />
</Software>
```

Abbildung 4: Markup-Scheme-Modell

Durch die Eigenschaft, dass diese Modelle jedoch spezifisch für verschiedene Bereiche sowie limitiert in der Unterstützung von Kontextaspekten sind [18], ist es nicht in jedem System möglich, dieses Datenmodell zu verwenden.

3.3 Grafische Modelle

Mit Hilfe der Unified Modeling Language (UML) ist es möglich, Kontextaspekte grafisch abzubilden. Vor allem verschiedene Arten von UML-Diagrammen können dabei nützlich sein, um den visuellen Aspekt zu verdeutlichen. Bauer [4] geht in seiner Arbeit besonders auf das Modellieren von Kontext auf dem Gebiet der Luftfahrt ein. Eine laut Bauer mögliche Art, um Kontext in UML-Diagramme einzubinden, ist es, eigene Kontext-Elemente zu definieren und diese mit dem gewünschten Objekt mit Hilfe von Verbindungen zu verknüpfen. Somit soll es möglich sein, dass das Kontext-Objekt je nach aktueller Kontextsituation Informationen an das eigentliche Objekt weiterleitet. Ein Beispiel hierzu zeigt Abbildung 5, wobei auf das Objekt *Flugzeug* zwei Kontext-Aspekte einwirken – Turbulenzen und Luftfahrt-spezifische Konflikte (MTC, Medium Term Conflict).

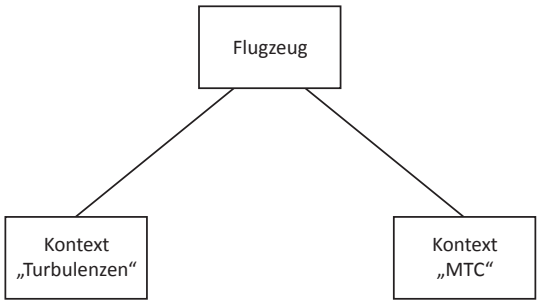

Abbildung 5: Grafisches Modell [4]

Neben UML gibt es weitere Möglichkeiten, um Kontext grafisch darzustellen. Hierzu zählt auch Object-Role Modelling (ORM), das laut Henricksen [13] besonderen Wert auf das Beschreiben von Tatsachen legt.

3.4 Objektorientierte Modelle

Durch die Verwendung von objektorientierten Techniken zur Darstellung von Kontextdaten können alle damit assoziierten Konzepte, zum Beispiel Kapselung, Vererbung und Wiederverwendbarkeit, verwendet werden [3]. Damit ist es möglich, für verschiedene Kontexttypen unterschiedliche Objekte zu erstellen – angepasst an deren jeweiligen Eigenschaften. Um die Interoperabilität zwischen mehreren Modulen zu gewährleisten, ist es nötig, die Schnittstellen der Kontextobjekte mittels Interfaces festzulegen.

Ein Beispiel eines solchen objektorientierten Ansatzes zur Verarbeitung von Kontextinformationen stellt das Framework *Hydrogen* [14] dar. Es besteht unter anderem aus Kontextelementen verschiedener Typen – dargestellt in Abbildung 6.

Dabei ist ersichtlich, dass es in oberer Ebene Objekte vom Typ *Context* gibt, welche eine komplette Kontextsituation darstellen. Des Weiteren gibt es Kontextobjekte, welche zum Beispiel Informationen zu einem Zeitpunkt, einem Ort oder einem Benutzer repräsentieren. Folgende fünf Typen von Kontextelementen werden derzeit vom System unterstützt (durch Ableiten von *ContextObject* beliebig erweiterbar):

- **Time:** Die aktuelle Uhrzeit
- **Location:** Repräsentiert die aktuelle Position, welche durch GPS-Koordinaten gespeichert wird

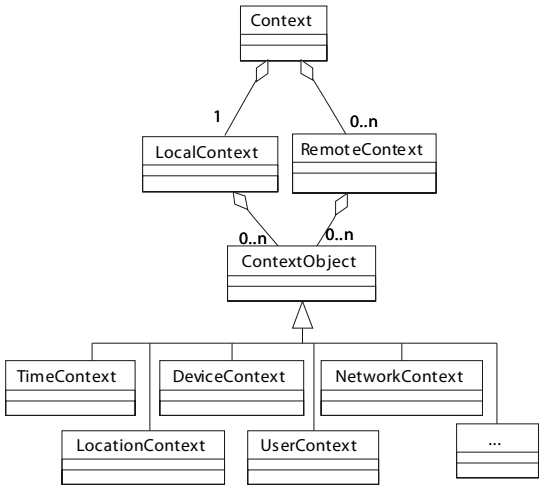

Abbildung 6: Objektorientiertes Modell [14]

- **Device:** Typ und ID des verwendeten Endgeräts
- **User:** Der Name des aktuellen Benutzers
- **Network:** Verfügbare Netzwerkschnittstellen

Ein Problem der objektorientierten Modelle ist es, dass die Objekte in verschiedenen Systemen wahrscheinlich unterschiedlich aussehen und somit auch die Schnittstellen anders sind [18]. Somit ist es schwierig, Systeme miteinander zu verbinden und Kontextelemente auszutauschen.

3.5 Logik-basierte Modelle

Modelle, welche nach diesem Ansatz erstellt werden, besitzen einen hohen Grad an Formalität, da dabei Kontext oftmals mit Hilfe von Fakten, Regeln und formalen Ausdrücken definiert wird [3]. Aufgrund dieser Tatsache ist es jedoch relativ einfach möglich, neue Elemente (High-Level-Kontextinformationen) zu erstellen, da dafür klare Regeln spezifiziert werden können. In ein logik-basiertes Modell werden im Normalfall neue Fakten eingespielt, alte gelöscht bzw. Daten aktualisiert.

Eine der ersten logik-basierten Modelle, das des Weiteren Vererbung ermöglicht, entwickelten McCarthy und Buvac [17]. Dabei werden Kontextelemente als abstrakte mathematische Elemente mitsamt gewissen Eigenschaften betrachtet. Als Basis-Relation wurde

$$ist(c,\ p)$$

definiert. Diese Formel beschreibt die Behauptung, dass die Aussage p im Kontext c wahr ist (*is true*). Damit könnte folgende Beispiel-Behauptung aufgestellt und geprüft werden:

ist(context-of(„Fachhochschule Hagenberg“), „Mobile Computing ist ein Masterstudiengang“)

Neben diesen Behauptungen können auch Werte, welche von der aktuellen Kontextsituation abhängen, abgefragt werden:

value(context-of(„Studiengang Mobile Computing“), „Anzahl an Studenten“) > 50

3.6 Ontologie-basierte Modelle

Als Ontologien werden im Bereich der Informatik strukturierte Konzepte bezeichnet, welche in Graphen abgebildet sind und mit Hilfe von Relationen (semantische Relation oder Verbindung bzw. Vererbung) verknüpft werden [18]. Ontologien werden hauptsächlich dazu verwendet, um das Wissen des behandelten Domänenbereichs – auch für Computer verstehbar – zu modellieren. Einsatz finden Ontologien etwa beim Austausch von Wissen, da die gewünschten Konzepte mitsamt Attributen gut strukturiert werden können. Da das Aussehen und die Struktur einer Ontologie immer vom Ersteller abhängig ist, gibt es für spezielle Wissensbereiche viele unterschiedliche, jedoch gültige Ontologien. Als Vorteile von Ontologien ist zu nennen, dass der Austausch von Daten erleichtert wird, da Doppeldeutigkeiten eliminiert werden. Außerdem erleichtern Ontologien durch ihre deskriptive Darstellungslogik das logische Denken der Anwender.

Als einer der ersten Ansätze, Kontext mittels Ontologien abzubilden, gilt das Konzept von Ötztürk und Aamodt [1]. Dabei geht es um die Analyse psychologischer Studien in Kombination mit kontextsensitiven Informationen.

Auch das W3C (World Wide Web Consortium) bietet eine Möglichkeit Ontologien zu erstellen. Mit Hilfe der Web Ontology Language (OWL) können Ontologien basierend auf dem Resource Description Framework (RDF) verfasst werden.

Ein Beispiel einer einfachen Ontologie illustriert Abbildung 7.

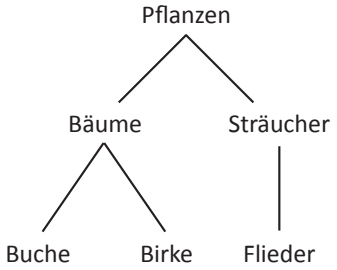

Abbildung 7: Ontologie zur Vegetation

Strang und Linnhoff-Popien kamen in ihrer Evaluierung der verschiedenen Datenmodelle zum Schluss, dass Ontologien die meisten Möglichkeiten bieten und die von ihnen aufgestellten Anforderungen an solche Konzepte am besten erfüllen. Korpipää [15] beschreibt diesbezüglich einige Anforderungen an Ontologie-Konzepte, welche im Zusammenhang mit Kontextsensitivität verwendet werden:

- **Einfach:** Um die Arbeit der Entwickler zu erleichtern sollen die Ausdrücke und Beziehungen so einfach wie möglich gehalten werden.
- **Flexibel und erweiterbar:** Um spätere Erweiterungen zu ermöglichen, soll die Ontologie flexibel im Bezug auf neue Kontextelemente und Beziehungen sein.
- **Generisch:** Die Ontologie soll verschiedene Typen von Kontextelementen unterstützen.
- **Aussagekräftig:** Um die Kontextinformationen so genau wie möglich erfassen zu können, soll die Ontologie so viele Elemente wie möglich in hohem Detailgehalt beinhalten können.

3.7 ContextUML

Als eine weitere Möglichkeit Kontext in Anwendungen zu modellieren gilt der Ansatz von Sheng und Benatallah [22]. Dieser basiert auf UML und stellt ein Metamodell zur Verfügung, welches Kontext, Dienste und kontextsensitive Mechanismen inkludiert. Aufgrund dieses Konzepts ist der Ansatz vor allem für die modellgetriebene Entwicklung von Anwendungen interessant, welche auf einem vom Systementwickler erstellten Modell basieren. Für Sheng und Benatallah ergibt sich daraus der Vorteil dass die Produktivität und Qualität wärend der Entwicklungsphase enorm gesteigert werden können. Der größte Pluspunkt eines solchen modellgetriebenen Systems ist jedoch die Tatsache, dass die Implementierung von Diensten dabei nahezu automatisch generiert werden kann – sogar für verschiedene Plattformen.

Abbildung 8 zeigt das Metamodell, welches die abstrakte Syntax von ContextUML darstellt und von zwei Sichtweisen betrachtet werden kann – der Kontextmodellierung sowie der Modellierung von Kontextsensitivität. Die erste Perspektive beinhaltet folgende Elemente des Metamodells:

- **Context:** Repräsentiert die Kontextdaten selbst und ist unterteilt in Low- (*AtomicContext*) und High-Level-Kontextinformationen (*CompositeContext*).
- **ContextSource:** Diese Klasse stellt die Quelle der Kontextdaten dar, wobei es wiederum zwei Typen gibt – *ContextService* und *ContextServiceCommunity*.

Neben diesen Elementen sind für die Modellierung von Kontextsensitivität weitere Module notwendig, welche im Folgenden aufgelistet sind:

- **CAMechanism:** Diese Objekte stellen ein Element dar, welches den Mechanismus der Kontextsensitivität formuliert. Ein Mechanismus kann zu Elementen des Typs *CAObject* zugeordnet werden und diese im weiteren Verlauf beeinflussen oder ändern.
- **CAObject:** Repräsentiert ein kontextsensitives Objekt – etwa eine Nachricht oder ein Dienst – welches situationsbedingt angepasst werden soll.
- **ContextBinding:** Als eine Art von *CAMechanism* erlaubt diese Klasse die automatische Zuordnung von Kontext zu kontextsensitiven Objekten.
- **ContextTriggering:** Eine weitere Art von *CAMechanism* ist das Modul *ContextTriggering*. Dies stellt die kontextsensitive Adaptierung von Objekten dar und inkludiert das Ausführen von Diensten aufgrund spezifischer Kontextinformationen. Dies wird mittels *ContextConstraints* und *Actions* erreicht, welche Bedingungen bzw. Aktionen einer bestimmten kontextsensitiven Adaption darstellen.

Auch wenn dieser Ansatz eigentlich für die Umsetzung kontextsensitiver Webdienste erarbeitet wurde, ist ein Einsatz auf anderen Gebieten durchaus denkbar, da es viele Bemühungen gibt, modellgetriebene Systeme zu entwickeln und dieses Projekt das erste seiner Art darstellt, welches auf die Umsetzung von kontextsensitiven Systemen mittels modellgetriebenen Architekturen (Model-Driven Architecture, MDA) eingeht.

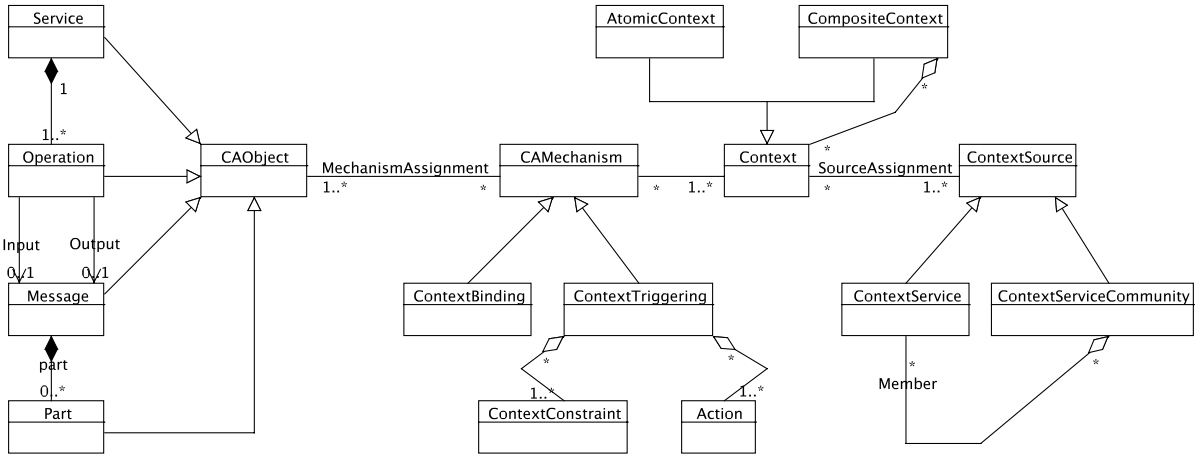

Abbildung 8: Metamodell von ContextUML [22]

3.8 Projekt: Simple Mobile Services

Ein weiterer auf den Modellierungssprache UML basierender Ansatz wurde im Projekt *Simple Mobile Services* erarbeitet [5]. Dieses hat zum Ziel, mit Hilfe eines das System inklusive Kontextinformationen beschreibenden Modells die Entwicklung von mobilen kontextsensitiven Diensten zu erleichtern. Um dies zu erreichen, wurde ein Ansatz gewählt, welcher sich in drei Ebenen gliedert – einem Metamodell, dem Modell des zu entwickelnden Systems selbst und der laufenden Instanz(en).

Das Erstellen von mobilen Diensten (Simple Mobile Services, SMS) kann dabei auf zwei verschiedene Arten erfolgen. Einerseits gibt es für erfahrene Systementwickler die Möglichkeit, das System bzw. den Dienst auf UML-Ebene mit allen verfügbaren UML-Techniken zu erstellen und anzupassen. Als zweiten Weg zu einer Anwendung gibt es für nicht so versierte Benutzer ein Authoring-Tool, wobei die Erstellung von Anwendungen hierbei ohne UML- bzw. programmiertechnisches Wissen erfolgen kann.

Die Architektur des Systems – bestehend aus den drei Schichten – ist in Abbildung 9 dargestellt.

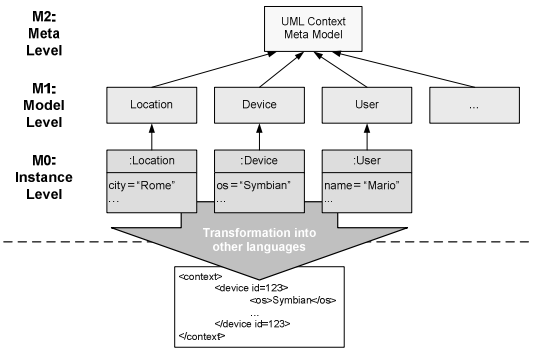

Abbildung 9: 3-Level-Architektur von SMS [5]

Die erste Ebene (M2) repräsentiert dabei das Kontext-Metamodell und definiert generische Elemente zum Modelliern von Kontext und Kontextsensitivität. Der mittlere Level (M1) verwendet die Komponenten der Meta-Ebene, um konkrete Klassen für das System zu erzeugen – etwa ein Objekt welches den Benutzer repräsentiert. Diese Ebene dient dazu, um alle in einer Anwendung benötigten Module bzw. Klassen zu erstellen und deren Eigenschaften zu definieren. Dies ist somit die Ebene bzw. das UML-Modell, welches der Systementwickler nach den Bedürfnissen der Anwendung erstellt bzw. ändert. Die dritte im Modell abgebildete Ebene (M0) stellt konkrete Instanzen vom in Ebene 1 definierten System zur Laufzeit dar. Die Instanzen beinhalten dabei zum Beispiel aktuelle Kontextinformationen.

Das Kontext-Metamodell dieses Ansatzes ist – wie in Abbildung 10 ersichtlich – relativ stark an das vorher behandelte ContextUML-Metamodell angelehnt.

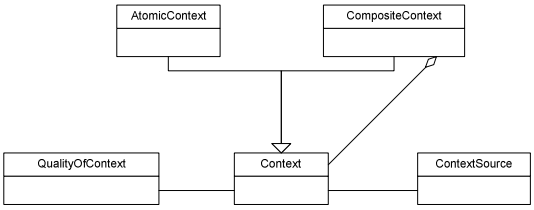

Abbildung 10: Kontext-Metamodell von SMS [5]

Wenige Unterschiede bestehen jedoch – so wurde etwa der Bereich rund um die Quelle der Kontextinformationen (*ContextSource*) vereinfacht. Außerdem wurde das Modell um ein Element zur Beschreibung der Qualität von Kontext erweitert (*QualityOfContext*), welches Parameter zur Aussage der Qualität beinhaltet.

4. SYSTEMARCHITEKTUREN UND FRAMEWORKS FÜR KONTEXTSENSITIVE ANWENDUNGEN

4.1 Anforderungen

Um Anwendungsentwicklern das Erstellen von kontextsensitiven Systemen zu erleichtern und eine Basis an oft benötigten Funktionen bzw. Modulen zu schaffen, ist es oft vorteilhaft, Frameworks zu kreieren, welche diese Aufgabe übernehmen und auf vierschiedenen Gebieten eingesetzt werden

können. Da Systeme mit Kontextbezug spezielle Anforderungen an die Anwendungsarchitektur haben, müssen auch Frameworks für kontextsensitive Anwendungen bestimmte Anforderungen erfüllen. Dey u. a. haben die Grundvoraussetzungen für solche Frameworks zusammengetragen und folgendermaßen definiert [8]:

- Mit Hilfe des Frameworks erstellte Anwendungen muss es möglich sein, auf gleiche Art Kontextinformationen und Benutzereingaben abzurufen
- Unterstützung von unterschiedlichen Plattformen und Programmiersprachen
- Unterstützung der Interpretation der Kontextdaten
- Unterstützung der Aggregation von Kontextinformationen
- Unterstützung von immer verfügbaren und unabhängig agierenden Kontext-Lieferanten
- Unterstützung beim Abspeichern der Kontext-Historie

4.2 Basismodelle

Für die Entwicklung von Softwareanwendungen gibt es zwei grundlegende Modelle – einen zentralisierten Ansatz sowie einen verteilten Ansatz [11]. Ersterer ist in Abbildung 11 dargestellt.

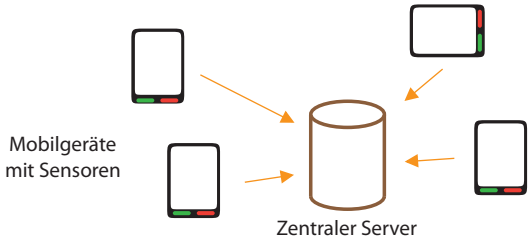

Abbildung 11: Zentralisierter Ansatz

Dabei gibt es ein zentrales Modul (z. B. einen Server), welches Daten – in diesem Fall Kontextinformationen von externen Sensoren – sammelt und für andere Module bereitstellt. Das Bereitstellen der Daten kann wiederum auf zwei verschiedene Arten erfolgen – entweder durch explizite Anfrage an die zentrale Einheit (*pull*) oder mittels automatischer Benachrichtigung aller an der zentralen Einheit angemeldeter Module (*push*).

Abbildung 12 zeigt den verteilten Ansatz eines Systems.

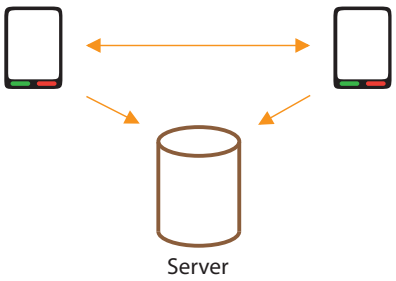

Abbildung 12: Verteilter Ansatz

Dabei ist im Gegensatz zum soeben beschriebenen Modell keinen zentralen Server o. Ä. vorgesehen, welcher als Datenspeicher fungiert. Stattdessen werden die Informationen an verschiedenen Orten – etwa an mehreren Mobilgeräten – erfasst und auch abgelegt. Somit kann die Anwendung oder der Benutzer selber entscheiden, welche Informationen überhaupt an andere Module weitergeleitet werden, was einen wichtigen Aspekt des Datenschutzes darstellt. Möchte der Anwender jedoch z. B. Dienste eines entfernten Servers nutzen, welcher bestimmte Kontextinformationen als Paramenter benötigt, so können diese zuerst übermittelt und die Ergebnisse der Anfrage abgerufen werden. Neben dem Vorteil, die verschickten Informationen durch den Anwender zu bestimmen, bietet dieser Ansatz auch die Lösung für einen möglichen Engpass am Server-Dateneingang des zentralisierten Ansatzes, da nicht alle Informationen versendet werden müssen.

4.3 Architekturen und Frameworks

4.3.1 Architektur von Schilit

In Anlehnung an das Projekt *ParcTab* entwickelte Schilit 1995 eine Systemarchitektur für kontextsensitive Systeme [9, 24]. ParcTab basiert auf der Kommunikation von Mobilgeräten (so genannten ParcTab) per Infrarot-Netzwerk, wodurch Nachrichten ausgetauscht und Objekte lokalisiert werden können. Die angesprochene Systemarchitektur besteht aus drei Hauptkomponenten, welche im Folgenden genauer erläutert werden:

- **Device Agents:** Dienen zur Verwaltung der mobilen Geräte und liefern Informationen darüber (z. B. Hardware-Ressourcen)
- **Active Maps:** Beinhalten die aktuellen Positionen der Mobilgeräte bzw. deren Benutzer auf einer Karte
- **User Agents:** Repräsentierten den Benutzer mitsamt seinen Eigenschaften

Das System ist wie in Abbildung 13 dargestellt aufgebaut.

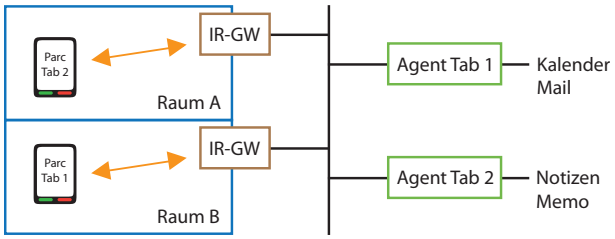

Abbildung 13: *ParcTab*-Systemarchitektur [24]

Dabei ist ersichtlich, dass die Mobilgeräte per Infrarot-Gateways (IR-GW) mit den jeweiligen Device-Agents kommunizieren, welche mit den Gateways in Verbindung stehen. Die Device-Agents haben sodann die Aufgabe, verschiedene Anwendungen an die Endgeräte anzubinden. Außerdem dienen sie dazu, Informationen zu verschicken, Anfragen zu bearbeiten und Positionsinformationen zur Verfügung zu stellen.

Mit Hilfe der Active-Maps, welche i. d. Ṙ. eine Grundkarte des jeweiligen Gebäudes darstellt, ist es nun möglich, die

verfügbaren Endgeräte auf einer Karte zu präsentieren sowie die Informationen über Positionsänderungen an Anwendungen weiterzureichen.

4.3.2 *Context Toolkit*

Das Context Toolkit [8], ein primär für Java entwickeltes Framework für die Erstellung kontextsensitiver Anwendungen, wurde von Dey u. a. entwickelt und gliedert sich in folgende Module:

- **Widget:** Repräsentiert einen abstrakten Zugriff auf einen Sensor und liefert dadurch die aktuellen Sensorwerte
- **Server:** Dient zum Sammeln der Kontextinformationen
- **Interpreter:** Ist für das richtige Interpretieren der Daten zuständig

Abbildung 14 zeigt die Relationen zwischen den einzelnen Systemkomponenten.

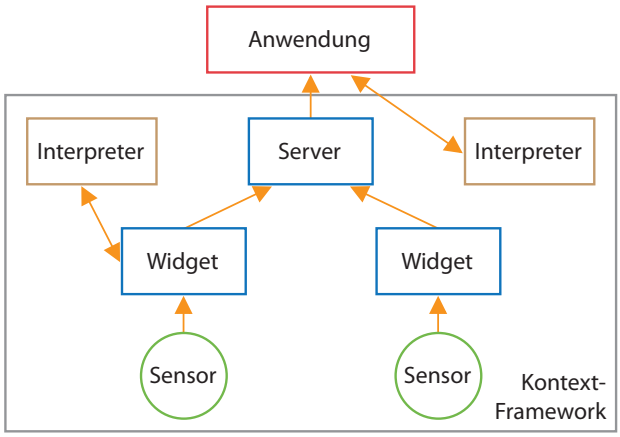

Abbildung 14: Architektur Context Toolkit [8]

Die Sensoren werden dabei durch die bereits angesprochenen Widget-Komponenten abgekapselt, um den Zugriff darauf zu vereinfachen. Die Widgets wiederum erhalten die Werte der Sensoren und geben diese an die betreffenden interessierenden Module weiter – dies geschieht über Callback- oder Anfrage-Mechanismen. Außerdem ermöglichen Widgets den Zugriff auf historische Daten – sie speichern also die Kontextinformationen zu definierten Zeitpunkten ab.

Der in der Grafik abgebildete Server sammelt die von den Widgets gelieferten Kontextinformationen (z. B. alle Daten einer Person). Dies erleichtert die Arbeit eines Anwendungsentwicklers enorm, da dieser sodann alle verfügbaren Informationen eines Objekts auf einmal abrufen kann, was wiederum mittels den zuvor genannten Mechanismen geschieht.

Interpreter können von Widgets, Servern und Anwendungen gleichermaßen benutzt werden um Kontextinformationen zu interpretieren. Beispiele hierfür wären eine vorhandene Position einem Bezirk zuzuordnen oder mittels Position, Benutzerinformationen und Umgebungsgeräuschen abzuleiten, ob sich der Anwender gerade in einer Besprechung oder im Straßenverkehr aufhält.

Mit Hilfe der vorhandenen Module des Frameworks, welche mittels HTTP und XML kommunizieren, ist das relativ einfache Erstellen von kontextsensitiven Anwendungen möglich. Durch das Abstrahieren der Sensoren ist es außerdem dem Entwickler überlassen, weitere Kontextinformationen ins System einzubinden. Dies war beim zuvor beschriebenen Systemansatz von Schilit nicht der Fall – hierbei war als einzige Information die Position des Benutzers verfügbar.

4.3.3 *Hydrogen*

Ein weiteres Framework für die Erstellung kontextsensitiver Anwendungen ist das bereits in Abschnitt 3 angesprochene *Hydrogen*-Framework [14], dessen Architektur in Abbildung 15 dargestellt ist.

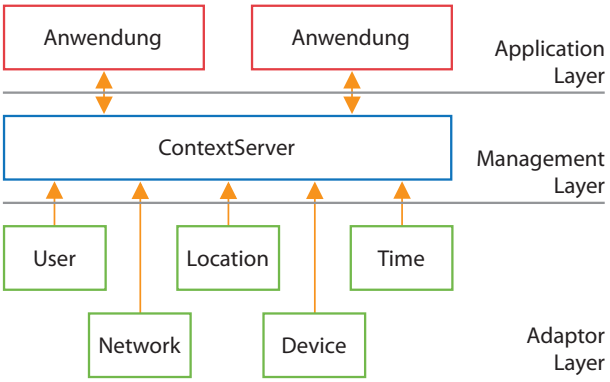

Abbildung 15: Architektur Hydrogen [14]

Das System ist – im Gegensatz zum Context Toolkit – in mehrere Ebenen (Layer) unterteilt, welche verschiedene Aufgaben übernehmen:

- **Adaptor Layer:** Dieser Layer ruft Daten von den jeweiligen im System vorhandenen Sensoren ab und erweitert diese physikalischen Informationen bei Bedarf um logische Kontextinformationen, welche gesammelt an den übergeordneten Management-Layer weitergegeben werden.
- **Management Layer:** Der *ContextServer* stellt als einzige Komponente diesen Layer dar und dient dazu, alle vom Adaptor Layer gelieferten Kontextinformationen zu speichern. Des Weiteren ist es diesem Layer möglich, mit anderen Systemen in Kontakt zu treten und Informationen (Kontextdaten) auszutauschen. Um den Anwendungen diese gesammelten Daten weiterreichen zu können, sind zwei Mechanismen vorgesehen – eine *pull*-basierte Methode auf Basis einer spezifischen Anfrage sowie ein *push*-basiertes Verfahren bei dem die Anwendungen über neue Daten umgehend informiert werden.
- **Application Layer:** Alle Anwendungen, welche das Framework – also die beiden unteren Ebenen – benutzen, stellen diesen Layer dar.

Ein Vorteil dieses Frameworks ist die Tatsache, dass durch das Anordnen der drei Ebenen auf einem Endgerät die Fehlerquellen eines Netzwerks (Verbindungsabbruch etc.) elmininiert werden. Außerdem ist es möglich, über den *ContextServer* allen interessierenden Anwendungen Zugriff auf alle Kontextinformationen zu bieten. Somit ist es nicht notwendig, dass jede Applikation die Sensoren auf unterster Ebene ansprechen muss.

4.3.4 CASS

Anders als alle bisher beschriebenen Frameworks nutzt Context-Awareness Sub-Structure (CASS) einen so genannten Middleware-Ansatz um das System umzusetzen [3, 10]. Die Architektur des von Fahy und Clarke entwickelten Systems ist in Abbildung 16 dargestellt.

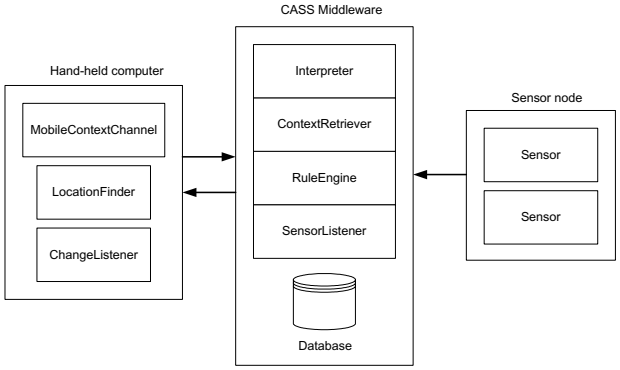

Abbildung 16: Architektur CASS [3]

Das System besteht aus drei Hauptkomponenten – einem mobilen Endgerät, der Middleware selbst, welche auf einem Server ausgeführt wird, und den Sensorknoten. Diese drei Elemente enhalten wiederum Module und werden im Folgenden näher erläutert:

- **Hand-held computer:** Stellen die mobilen Endgeräte dar, welche kontextsensitive Anwendungen ausführen und mit dem CASS-Middleware-Server über eine drahtlose Schnittstelle kommunizieren. Um Kommunikationsunterbrechungen zwischen Server und Endgerät zu verbergen, ist es möglich, alle vom Server abgerufenen Kontextinformationen am Endgerät lokal abzulegen.

- **CASS Middleware:** Das Hauptmodul dieses Systems ist die CASS Middleware. Diese dient zunächst dazu, die von den *Sensor nodes* gelieferten Daten einzulesen (*SensorListener*) und abzuspeichern (*Database*). Um die so abgelegten Daten wieder abzurufen, wird das Modul *ContextRetriever* verwendet. Ähnlich wie im Context Toolkit werden *Interpreter* eingesetzt, um Kontextinformationen zu interpretieren. Die letzte Komponente der Middleware – *RuleEngine* – wird von Fahy selbst nicht genauer erläutert.

- **Sensor node:** Dieses Element repräsentiert eine externe Komponente, welche Zugriff auf einen oder mehrere Sensoren hat. Die von den Sensoren gelieferten Werte werden vom Sensor node ausgelesen und an die CASS-Middleware weitergegeben.

Der primäre Vorteil dieses Server-basierten Ansatzes ist, dass die gesamte Erfassung der Kontextinformationen nicht am Endgerät selbst sondern an externen Modulen stattfindet, wodurch wichtige Ressourcen (Rechenleistung, Speicherplatz) für die kontextsensitive Anwendung selbst verwendet werden können.

4.3.5 FuzzySpace

In ihrem Artikel stellen Schmidt und Gellersen eine weitere Architektur für Systeme mit Kontextbezug dar [21]. Dabei lehnt sich das Modell an so genannte *Unscharfe Mengen* (*Fuzzy Logic*) an, deren Grundidee es ist, die Zuordnung von Objekten zu Mengen nicht nur binär (ja oder nein) sondern auch *unscharf* zuzulassen. Dies bedeutet, Objekte können dabei mit einer bestimmten Relevanz einem bzw. mehreren Bereichen zugeordnet werden – diese Zuordnung geschiet dabei z. B. mittels Dreiecks- oder Trapezfunktionen.

Im Rahmen der kontextsensitiven Systeme wird das Verfahren der Unscharfen Mengen dazu verwendet, die räumliche und zeitliche Relevanz für jeden gewünschten Ort bzw. Zeitpunkt zu berechnen.

Die *FuzzySpace*-Architektur für kontextbezogene Systeme ist wie in Abbildung 17 dargestellt aufgebaut.

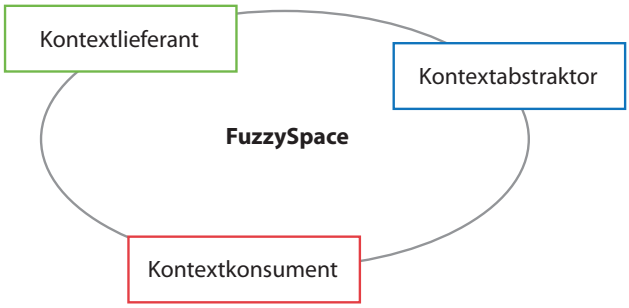

Abbildung 17: Architektur FuzzySpace [21]

Die drei Komponenten – *Kontextlieferant*, *Kontextkonsument* und *Kontextabstraktor* – kommunizieren über den so genannten *FuzzySpace*, welcher auf Tupel sowie Unscharfen Mengen basiert und einen Kontextraum für die vorhandenen Objekte realisiert. Dadurch kann allen Objekten eine gewisse zeitliche und räumliche Relevanz zugeordnet werden. Über einen Abfragemechanismus können interessierende Elemente vom FuzzySpace abgerufen werden. Die drei über den FuzzySpace gekoppelten Systemkomponenten werden im Folgenden genauer erläutert:

- **Kontextlieferant:** Stellt für andere Module Kontextinformationen bereit, welche von Sensoren oder anderen Geräten erstellt und eingelesen werden. Wichtig ist, dass alle Kontextinformationen einen Zeit- und Ortsbezug besitzen.

- **Kontextkonsument:** Alle Module, welche in irgend einer Form Kontext nutzen, werden diesem Element zugeordnet. In erster Linie sind dies kontextsensitive Anwendungen, welche die Informationen der Kontextlieferanten benutzen, um etwa Anwendung anzupassen.

- **Kontextabstraktor:** Repräsentiert ein Modul, welches sowohl als Kontextkonsument als auch als Kontextlieferant fungiert und dafür verwendet wird, aus bestehenden Kontextdaten zusätzliche Informationen abzuleiten bzw. zu berechnen. Das Konzept ist dabei ähnlich zu den im Context Toolkit integrierten Interpretern, welche ebenfalls mit Hilfe vorhandenen Fakten neue Daten ermitteln.

Durch die relativ klare Trennung von Kontextlieferanten und -konsumenten in diesem Ansatz bietet dieser die Möglichkeit, die Module eigenständig und unabhängig voneinander zu entwickeln, zu testen und in einem letzten Schritt zusammenzuführen. Dies bedeutet einerseits, dass durch die Aufteilung des Systems in kleinere Komponenten die Entwicklung begünstigt wird, andererseits ist es dadurch auch relativ einfach möglich, die Module auszutauschen und neue ins System zu integrieren.

5. BEISPIELE FÜR KONTEXTSENSITIVE SYSTEME

Um zu zeigen, welche Möglichkeiten kontextsensitive Syteme bieten, sollen in diesem Abschnitt einige Beispielprojekte vorgestellt werden.

5.1 Health Coach

Das von Philips entwickelte Projekt *Health Coach* dient laut Ensing [9] in erster Linie dazu, Personen über den aktuellen Fitness- und Gesundheitszustand sowie mögliche bessere Ernährungsweisen zu informieren. Dazu werden fünf Sensoren verwendet – Temperatur, Gewicht, Puls, Blutdruck und körperliche Aktivitäten.

Im aktuellen Entwicklungsfortschritt liest ein Computer die Sensorwerte ein, interpretiert diese und speichert diese Werte mitsamt Zeitstempel persistent ab. Möchte sich der Benutzer über seine aktuelle körperliche Verfassung informieren, so werden alle im System gespeicherten Werte herangezogen und daraus mit teils komplexen Formeln der aktuelle Fitnesszustand berechnet.

Das System besteht zum einen aus der Health Coach-Applikation selbst, zum anderen aus einem Modul zum Einlesen der Sensorwerte (*Base Station*). Diese übernimmt alle notwendigen Berechnungen, beinhaltet die Datenbank für das Abspeichern der Werte sowie das Wissen wie die gemessenen Sensorwerte zu interpretieren sind. Über eine Netzwerkschnittstelle ist die Base Station mit der eigentlichen Health Coach-Anwendung verbunden, welche etwa auf einem PDA (Personal Digital Assistant) oder einem Fernsehgerät präsentiert werden kann.

5.2 MOBIlearn

Der von Lonsdale u. a. [16] beschriebene Ansatz beschäftigt sich damit, das Projekt *MOBIlearn* um kontextsensitive Ansätze zu erweitern. Das eigentliche Projekt hat zum Ziel, möglichst viele Dienste und Anwendungen für das Lernen mittels Mobilgeräten anzubieten. Um jedoch auf die jeweiligen Geräte mit deren speziellen Eigenschaften (Displayauflösung etc.) einzugehen, ist es notwendig, einen kontextsensitiven Ansatz zu integrieren, um zur richtigen Zeit das korrekt aufbereitete Material für den Benutzer zu liefern.

Durch den Ansatz, den zu liefernden Inhalt auf Basis der aktuellen Kontextinformationen zu wählen, ergeben sich die Vorteile, dass der Benutzer nicht lange nach dem gewünschten Inhalt suchen muss sowie dass das System nicht auf Benutzereingaben angewiesen ist, sondern auch im Hintergrund Aktivitäten ausführen kann.

Nicht nur Lern-Inhalte sondern auch passende Dienste wie die Kontaktaufnahme mit Personen, welche aktuell auf das gleiche Gebiet fokussiert sind, sollen so durch die Konextinformationen selektiert und dem Benutzer präsentiert werden.

5.3 Lomotain

Das prototypisch an der Fachhochschule Hagenberg entwickelte System *Lomotain* befasst sich damit, ein Framework für kontextsensitive Anwendungen zu erstellen, die vor allem in Tourismusregionen Einsatz finden sollen und etwa interaktive Spiele darstellen. Dabei bedient man sich GPS (Global Positioning System) als Grob-Ortungsverfahren und NFC (Near Field Communication) als genaue Positionsbestimmung durch das Einlesen von RFID-Tags (Radio Frequency Identification). Da die jeweiligen Spielszenarien nur an bestimmten Orten vorhanden sind (z. B. in der Linzer Innenstadt), werden per Grob-Ortung die Anwendungen bestimmt, welche gestartet werden können. Der Benutzer bekommt i. d. R. nach dem Start der Applikation verschiedene Aufgaben (z. B. das Suchen eines bestimmten historischen Gebäudes), die mit dem Einlesen der dort angebrachten RFID-Tags automatisch komplettiert werden.

Ein in UML erstelltes Spiel-Szenario dient der Anwendung als Basis, welches per XML und drahtloser Kommunikation an die jeweiligen Endgeräte übertragen und dort aufbereitet wird. Dieses Szenario enthält neben den zu erfüllenden Aufgaben auch weitere Inhalte wie Texte, Bilder oder Videos, welche dem Benutzer an zuvor definierten Zeitpunkten präsentiert werden.

Neben einem Einzel-Spieler-Modus ist es des Weiteren möglich, mehrere Benutzer in ein Spiel einzubinden sowie die je nach Erfüllung der Aufgabe erhaltenen Punkte in einer Highscore-Liste zu hinterlegen. Nicht nur Stadtführungen können somit interessanter gestaltet werden, sondern auch Erlebnisparks können das System nutzen, um das Gebiet mit interessanten Aufgaben aufzuwerten und interaktive Dienste sowie ansonsten nicht verfügbare Informationen bereitzustellen.

6. RESÜMEE

Kontext bzw. kontextsensitive Informationen können jegliche Art von Daten darstellen, die für die jeweilige Anwendung gerade wichtig ist. Vorwiegend werden jedoch die aktuelle Position, Informationen über den Benutzer, die gegenwärtige Zeit sowie passierende Aktionen verwendet, um Anwendungen in Darstellung bzw. Logik situationsgerecht anzupassen.

Mit Hilfe verschiedener Modelle können Kontextinformationen innerhalb von Anwendungen dargestellt bzw. zwischen Systemen ausgetauscht werden. Dabei ist es möglich, auf einfache Key-Value-Modelle oder komplexere Konzepte zurückzugreifen.

Diverse bestehende Frameworks zeigen Möglichkeiten auf, wie Informationen der Umgebung in Anwendungen integriert und somit kontextsensitive Systeme umgesetzt werden können. Hierbei gibt es verschiedene Methoden – etwa den objektorientierten Ansatz des Context Toolkits, ein aus verschiedenen Layern bestehendes Konzept wie Hydrogen es verwendet, eine Middleware-basierte Architektur oder eine auf Unscharfe Mengen basierende Vorgehensweise. Jedes dieser Modelle hat verschiedene Vor- und Nachteile, alle jedoch ermöglichen es, Anwendungen zu entwickeln, welche die Einflüsse der aktuellen Situation – eben die Kontextinformationen – miteinbeziehen, um dem Benutzer die bestmöglichen Informationen liefern oder die idealen Dienste anbieten zu können.

7. REFERENCES

[1] A. Aamodt. Towards a model of context for case-based diagnostic problem solving. In *Context-97: Proceedings of the interdisciplinary conference on modeling and using context*, pages 198–208, Rio de Janeiro, 1997.

[2] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307, London, UK, 1999. Springer-Verlag.

[3] M. Baldauf, S. Dustdar, and F. Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, pages 263–277, June 2007.

[4] J. Bauer. Identification and modeling of contexts for different information scenarios in air traffic. Master's thesis, Technische Universität Berlin, March 2003.

[5] G. Broll, H. Hußmann, G. N. Prezerakos, G. Kapitsaki, and S. Salsano. Modeling context information for realizing Simple Mobile Services. In *Mobile and Wireless Communications Summit, 2007. 16th IST*, pages 1–5, 2007.

[6] T. Buchholz, A. Küpper, and M. Schiffers. Quality of context information: What it is and why wee need it. In *Proceedings of the 10th International Workshop of the HP OpenView University Association*, July 2003.

[7] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical report, Dartmouth College – Department of Computer Science, Hanover, NH, USA, 2000.

[8] A. K. Dey, D. Salber, G. D. Abowd, and M. Futakawa. An architecture to support context-aware applications. Technical report, Georgia Institute of Technology, 1999.

[9] J. Ensing. Software architecture for the support of context aware applications, 2002. Preliminary study.

[10] P. Fahy and S. Clarke. CASS – A middleware for mobile context-aware applications. In *Workshop on Context Awareness, MobiSys*, 2004.

[11] F. Fuchs. Kontextsensitive Dienste. Technical report, TU München, Fakultät für Informatik, January 2002.

[12] J. Greenberg. Metadata and the world wide web. In *Encyclopedia of Library and Information Science*, pages 1876–1888. CRC Press, 2003.

[13] K. Henricksen, J. Indulska, and A. Rakotonirainy. Generating context management infrastructure from high-level context models. In *In 4th International Conference on Mobile Data Management (MDM) - Industrial Track*, pages 1–6, 2003.

[14] T. Hofer, W. Schwinger, M. Pichler, G. Leonhartsberger, J. Altmann, and W. Retschitzegger. Context-awareness on mobile devices - the Hydrogen approach. In *HICSS '03: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9*, page 292.1, Washington, DC, USA, 2003. IEEE Computer Society.

[15] P. Korpipää and J. Mäntyjärvi. *An Ontology for Mobile Device Sensor-Based Context Awareness.* ACM, 2003.

[16] P. Lonsdale, C. Baber, M. Sharples, and T. N. Arvanitis. A context-awareness architecture for facilitating mobile learning. *Learning With Mobile Devices – Research and Development*, pages 79–85, 2004.

[17] J. McCarthy and S. Buvac. Formalizing context (expanded notes). Technical report, Stanford University, Stanford, CA, USA, 1994.

[18] M. Miraoui, C. Tadj, and C. ben Amar. Context modeling and context-aware service adaptation for pervasive computing systems. In *International Journal of Computer and Information Science and Engineering (IJCISE)*, volume 2, 2008.

[19] M. Samulowitz, F. Michahelles, and C. Linnhoff-Popien. CAPEUS: An architecture for context-aware selection and execution of services. In *Proceedings of the IFIP TC6 / WG6.1 Third International Working Conference on New Developments in Distributed Applications and Interoperable Systems*, pages 23–40, Deventer, The Netherlands, 2001. Kluwer, B.V.

[20] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *In Proceedings of the Workshop on Mobile Computing Systems and Applications*, pages 85–90. IEEE Computer Society, 1994.

[21] A. Schmidt and H.-W. Gellersen. Modell, Architektur und Plattform für Informationssysteme mit Kontextbezug. *Informatik – Forschung und Entwicklung*, 16(4):213–224, 2001.

[22] Q. Z. Sheng and B. Benatallah. ContextUML: A UML-based modeling language for model-driven development of context-aware web services development. In *ICMB '05: Proceedings of the International Conference on Mobile Business*, pages 206–212, Washington, DC, USA, 2005. IEEE Computer Society.

[23] T. Strang and C. Linnhoff-Popien. A context modeling survey, September 2004.

[24] R. Want, B. N. Schilit, N. I. Adams, R. Gold, K. Petersen, D. Goldberg, J. R. Ellis, and M. Weiser. An overview of the ParcTab ubiquitous computing experiment. *IEEE Personal Communications*, 2:28–43, 1995.