Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung AW2

Maik Weindorf

Self-Managing & Context-Aware
Mobile Computing

Fachbereich Elektrotechnik und Informatik
Department of Electrical Engineering and Computer Science

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Aufgrund wachsender Komplexität, werden Computersysteme zunehmend unbeherrschbar. Mit der zunehmenden Verbreitung leistungsfähiger mobiler Geräte, ergeben sich zusätzliche Anforderungen an Computersysteme. Das Thema "Self-Managing & Context-Aware Mobile Computing" wird daher zunehmend relevant. Auf diesem Gebiet gibt es bisher kaum etablierte Standards und es sind noch viele Fragen unbeantwortet. Diese Ausarbeitung gibt einen Überblick über grundlegende Konzepte und stellt verschiedene Forschungsarbeiten vor. Der Fokus liegt dabei nicht ausschließlich auf Mobile Computing, da die grundlegenden Konzepte sowohl für mobile, als auch für nicht-mobile Umgebungen gelten.

"Self-Management" und "Context-Awareness" sind zwei Themengebiete, die in enger Beziehung zueinander stehen. Laut [Schmidt et al., 1998] beschreibt Context[1]:

> "...a situation and the environment a device or user is in."

Context-Awareness bedeutet somit das Bewusstsein über diese Situation und Rahmenbedingungen.

Self-Management ist ein Oberbegriff für eine ganze Reihe von Themen, wie z.B. "self-configuration", "self-healing", "self-protection" und "self-optimization".

In Kapitel 2 folgt eine nähere Betrachtung der Begriffe.

[1] In dieser Ausarbeitung wird durchgängig der Begriff "Context" verwendet. Von einer Verwendung des deutschen Begriffs "Kontext" wird aus Konsistenzgründen abgesehen.

Kapitel 2

Grundlagen

In diesem Kapitel werden die Begriffe ”Self-Management” und ”Context-Awareness” näher betrachtet. Anschließend wird die Entwicklung dieser Begriffe erläutert.

2.1 Self-Management

Wie schon in Kapitel 1 erwähnt, handelt es sich bei dem Begriff ”Self-Management” um einen Oberbegriff für eine ganze Reihe von Themen. In der Literatur findet man in diesem Zusammenhang häufig Begriffe wie ”self-configuration”, ”self-healing”, ”self-protection” und ”self-optimization” als Unterthemen von Self-Management. Synonyme wie ”self-x” und ”Self-CHOP”, aber auch verwandte Themen wie ”Autonomic Computing” (siehe 3.1), ”Adaptive Computing”, ”Organic Computing”, ”selfware” und weitere. Die Grenzen sind dabei meist fließend. Es gibt keine genaue Definition von ”Self-Management”. Eine einfache Umschreibung für Self-Management ist z.B.: ”Die Fähigkeit eines Systems, sich ohne menschliche Einwirkung selbst zu beeinflussen, um definierte Ziele zu erreichen.”

Es stellt sich die Frage, wie sich Aspekte von Self-Management aus Software-Engineering Sicht realisieren lassen. Auf einem hohen Abstraktionsniveau lassen sich im Wesentlichen ”interne” und ”externe” Ansätze unterscheiden (siehe 2.1.1 und 2.1.2). Natürlich sind auch Kombinationen möglich.

2.1.1 extern

Ein System wird ”von außen” analysiert und beeinflusst. Das System muss dafür definierte Schnittstellen anbieten oder in einen ”Wrapper” verpackt werden (vgl. 3.1). Die Vorteile dieses Ansatzes sind, dass sowohl die Einbindung von legacy Systemen ermöglicht wird,

als auch eine Orchestrierung durch hierarchische Manager erreicht werden kann. Weiterhin können mehrere Systeme von einem Manager verwaltet werden. Der größte Nachteil dieses Ansatzes ist, dass der Manager nur über externe Schnittstellen Einfluss auf das von ihm verwaltete System nehmen kann. Die Möglichkeiten sind somit unter Umständen sehr begrenzt.

Ein Manager und die von ihm verwalteten Systeme können als neues Gesamtsystem betrachtet werden. (siehe Abbildung 2.1)

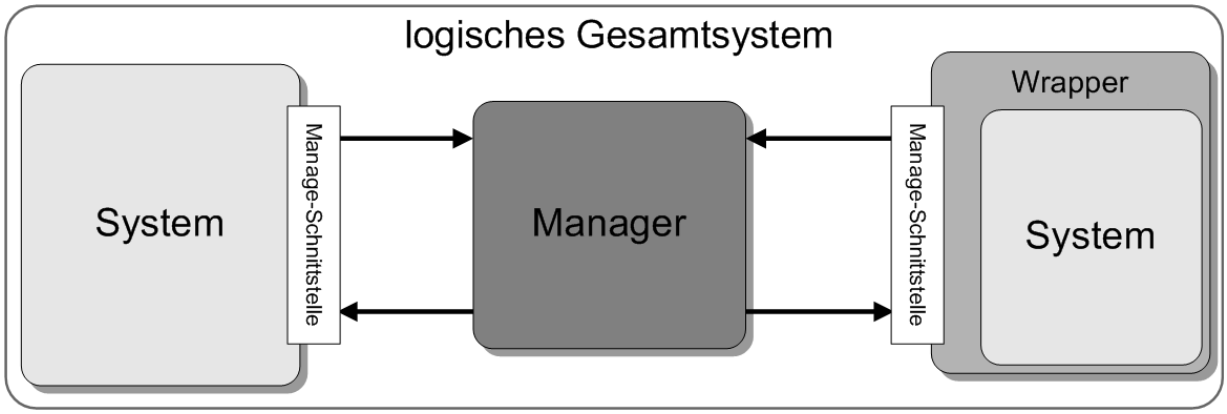

Abbildung 2.1: Manager extern

2.1.2 intern

Ein System analysiert "sich selbst". Schnittstellen für externe Manager können, müssen aber nicht nach außen angeboten werden (solche Schnittstellen sind für Orchestrierung sinnvoll). Dieser Ansatz bietet potentiell mehr Möglichkeiten der Systembeeinflussung, als externe Ansätze (siehe 2.1.1). Der Nachteil dieses Ansatzes ist, dass er nur bei Neuentwicklungen praktikabel ist, da die Manager Komponente ins Systemdesign integriert werden muss (siehe Abbildung 2.2).

2.2 Context-Awareness

"There is more to Context than Location" [Schmidt et al., 1998]. Context-Awareness in mobilen Umgebungen wird häufig gleichgesetzt mit Location-Awareness. Es gibt jedoch wesentlich mehr verwertbare Context-Informationen über die reine Location Information hinaus. In [Schmidt et al., 1998] wird eine Unterteilung in menschliche Faktoren und physikalische Umgebung vorgenommen. Menschliche Faktoren beschreiben dabei z.B. Benutzereinstellungen, soziales Umfeld und die konkrete Aufgabe eines Nutzers (Task). Die physikalische Umgebung hingegen beschreibt unter Anderem Umgebungsbedingungen, wie z.B.

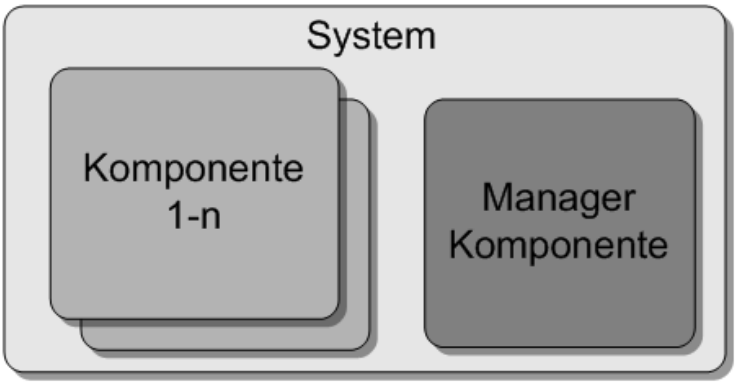

Abbildung 2.2: Manager intern

Licht, Temperatur und Beschleunigung, sowie Infrastruktur Informationen und natürlich auch die Location. All diese Faktoren lassen sich messen, bzw. ermitteln und unterschiedlich granular betrachten. So kann man beispielsweise Intensität und Wellenlänge als messbare Faktoren von Licht auffassen. Hofer [Hofer et al., 2003] hingegen nimmt eine Unterteilung in physikalischen und logischen Context vor. Dabei beschreibt der physikalische Context "low level" Informationen, wie z.B. eine GPS-Position oder eine Temperatur. Der logische Context hingegen beschreibt "high level" Informationen, wie beispielsweise Straßennamen oder Temperatur-Umschreibungen, wie "heiß" und "kalt". Somit wird schnell klar, dass die Definitionen von Context weder einheitlich noch eindeutig sind.

Es drängt sich die Frage auf, welches Ziel mit der Einbeziehung von Context in die Logik von Anwendungen erreicht werden soll. Eine aktuelle Studie des BSI [BSI, 2006] nennt als Folge von Context-Awareness einen erhöhten Komfort für die Nutzer und auf längere Sicht einen weiteren Schritt in Richtung Autonomie.
(vgl. Autonomic Computing [Ganek und Corbi, 2003] [IBM, 2001])

2.3 Entwicklung

Self-Management und Context-Awareness sind keine neuen Themen. Besonders Fehlererkennung, Fehlertoleranz und Selbstkonfiguration sind aus dem Server und DB Bereich schon lange bekannt (vgl. Load-Balancer, etc.). Der Trend geht jedoch hin zu einer globaleren Sicht auf Systemlandschaften. Autonomic Computing (siehe 3.1) ist ein gutes Beispiel für eine aktuelle Betrachtung dieser Themen.

Der Begriff Context-Awareness gewinnt zunehmend an Bedeutung. Wie schon beschrieben (2.2), ist seine Definition nicht eindeutig. Es entsteht leicht der Eindruck, dass es sich bei "Context-Awareness" um ein völlig neues Thema handelt. Der Begriff wurde jedoch schon Mitte der Neunziger Jahre geprägt. Es stellt sich also die Frage, warum dieses Thema momentan so viel Beachtung findet.

Wenn man die aktuellen Entwicklungen betrachtet (siehe 3), wird deutlich, dass der Trend im Bereich der Context-Awareness dahin geht, viele verschiedene Context-Informationen zu nutzen. Dazu werden die verschiedenen Informationen meist akkumuliert und in abstraktere Informationen überführt. Aufschwung bekommt das Thema zusätzlich durch die zunehmende Verbreitung leistungsfähiger mobiler Geräte.

Kapitel 3

Aktuelle Arbeiten

In diesem Kapitel werden einige Forschungsarbeiten und Konzepte vorgestellt, die das Ziel verfolgen, Aspekte von Self-Management und/oder Context-Awareness zu verwirklichen. Es wird dabei jeweils erläutert, welche Aspekte beachtet, bzw. vernachlässigt werden. Zusätzlich wird die Eignung für mobile Umgebungen bewertet.

3.1 Autonomic Computing

Autonomic Computing (AC) ist ein vor allem von IBM geprägter Begriff [IBM, 2001]. Bisher handelt es sich bei AC vor allem um eine Vision. Ausdrückliche Ziele von AC sind "self-configuration", "self-healing", "self-protection" und "self-optimization". Wie in Kapitel 2 beschrieben, handelt es sich dabei um Unterthemen von Self-Management. Im Folgenden werden einige Komponenten beschrieben, die laut IBM für eine AC Architektur notwendig sind und auf Aspekte von Self-Management und Context-Awareness untersucht. Für eine nähere Betrachtung von AC sei an dieser Stelle auf die AW1 Ausarbeitung verwiesen [UbiComp].

Autonomic Manager Autonomic Managers (AM) sind die wichtigsten Komponenten einer AC Architektur (siehe Abbildung 3.1). In ihnen steckt die eigentliche "Intelligenz" des Systems. Sie bieten folgende Funktionen:

- Monitor: Sammelt Informationen über Ressourcen, die vom AM verwaltet werden. Die gesammelten Informationen werden akkumuliert und gefiltert. Die so aufbereiteten Informationen (Symptome) werden an die "Analyze" Funktion übergeben.
- Analyze: Analysiert die Symptome und entscheidet, ob Änderungen notwendig sind. (basierend auf Wissen aus Knowledge Source)

- Plan: Erstellt einen Plan, wie die geforderten Änderungen umgesetzt werden können. Dies können einfache Befehle, bis hin zu komplexen Workflows sein.
- Execute: Führt die im Plan festgelegten Änderungen aus. Dies erfolgt über die Effector Schnittstelle der verwalteten Ressourcen.

Zusammen stellen diese Funktionen einen ”Control Loop” dar. Es gibt einen ständigen Zyklus von messen des ”Ist” Zustandes, Änderungsmaßnahmen zur Erreichung eines ”Soll” Zustandes und wiederum über die Messung des ”Ist” Wertes eine Kontrolle der Auswirkungen dieser Änderungsmaßnahmen. AM vereinen also Context-Awareness und Self-Management. Dieser Ansatz korrespondiert mit externen Self-Management Ansätzen (siehe 2.1.1).

Autonomic Manager können zudem hierarchisch angeordnet werden. D.h. Autonomic Manager können andere Autonomic Manager verwalten. Die übergeordneten Autonomic Manager werden dabei als ”Orchestrating Autonomic Manager” bezeichnet, die untergeordneten Autonomic Manager als ”Touchpoint Autonomic Manager”. Allerdings spielen Orchestrating Autonomic Manager derzeit kaum eine Rolle, da noch nicht klar ist, wie abstrakte Ziele eines Orchestrating Autonomic Manager in konkrete Ziele für Touchpoint Autonomic Manager überführt werden können. Selbst IBM nennt keine konkreten Anwendungsbeispiele.

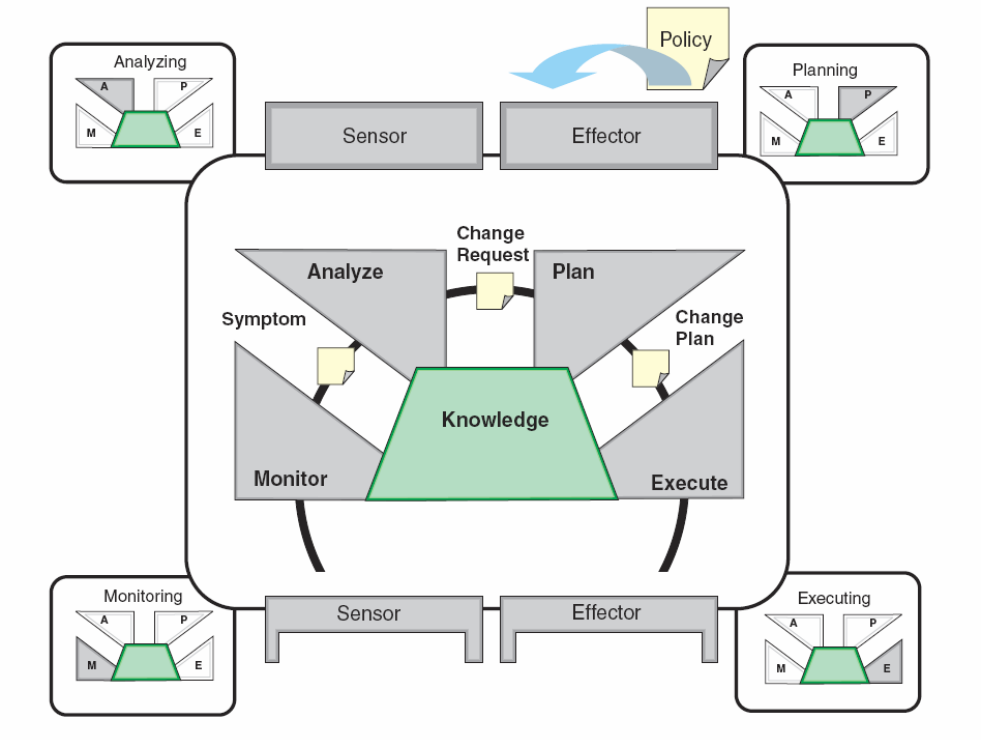

Abbildung 3.1: Autonomic Manager [IBM, 2005]

Touchpoint Ein Touchpoint kann als Wrapper für eine Managed Resource verstanden werden. Er stellt eine definierte Schnittstelle (Sensor/Effector) zur Verfügung, über die die Managed Resource kontrolliert und beeinflusst werden kann. Wie in Kapitel 2.1.1 beschrieben wurde, benötigt man für die Anbindung von Systemen an einen Manager definierte Schnittstellen. Diese Schnittstellen müssen entweder vom System selbst angeboten werden oder durch einen Wrapper nachgerüstet werden. Touchpoints sind dafür ein guter Ansatz.

Für Context-Awareness besitzen Touchpoints keine zentrale Bedeutung. Sie dienen hauptsächlich der Aufbereitung und Weitergabe von Informationen und Befehlen, besitzen jedoch selbst kaum Context-Informationen.

Managed Resources Eine Managed Resource ist ein Hard- oder Softwarekomponente, die "gemanaged" werden kann. Infrage kommt hier praktisch jede konventionelle Systemkomponente (z.B. Server, Datenbank, Application Server, Anwendungen, ect.). Die Managed Resources können dabei durchaus über Fähigkeiten zum Selbsmanagement verfügen.

Knowledge Source Eine Knowledge Source (bzw. Knowledge Service oder K-Service) ist eine Art Registry oder Datenbank, in der z.B. Symptome, Policies oder Pläne gespeichert werden können. Dieses gespeicherte Wissen wird für die verschiedenen Funktionen der Autonomic Manager genutzt. Dabei wird das Wissen der Knowledge Sources ständig durch die Autonomic Manager erweitert. Eine Knowledge Source ist somit für Context-Awareness von elementarer Bedeutung.

Eignung für mobile Umgebungen AC ist konzeptionell auf große Unternehmensweite Systeme ausgelegt, was mobile Komponenten nicht explizit ausschließt. Mobile Computing ist jedoch eindeutig nicht der Schwerpunkt dieses Konzeptes. Grundlegende Ansätze, wie Autonomic Manager und Touchpoints lassen sich aber ohne weiteres auf Mobile Computing übertragen.

3.2 Hydrogen Context-Framework

Abbildung 3.2 zeigt die Grobarchitektur des Frameworks [Hofer et al., 2003]. Es handelt sich bei dem Hydrogen Context-Framework um ein Java basiertes Framework. Es ist auf eine modulare Anbindungen verschiedenster Sensoren (bzw. Informationsquellen) und Anwendungen ausgelegt. Zentraler Punkt des Frameworks ist ein Context Server, der die verschiedenen Informationen der Sensoren akkumuliert und den Anwendungen in abstrahierter Form zur Verfügung stellt.

Wie der Name schon vermuten lässt, liegt der Schwerpunkt bei diesem Ansatz auf Context-Awareness, wobei die Framework Komponenten nicht für die endgültige Interpretation der

Context-Informationen verantwortlich sind. Self-Management Ansätze sind beim Hydrogen Context-Framework nicht vorgesehen. Die Verantwortung liegt letzten Endes bei den angebundenen Applikationen. Da das Framework in Java geschrieben wurde, kann man davon ausgehen, dass es nicht ohne weiteres auf allen mobilen Geräten lauffähig sein wird. Die modulare Struktur kann jedoch als Vorlage für eine äquivalente Implementierung für mobile Geräte dienen.

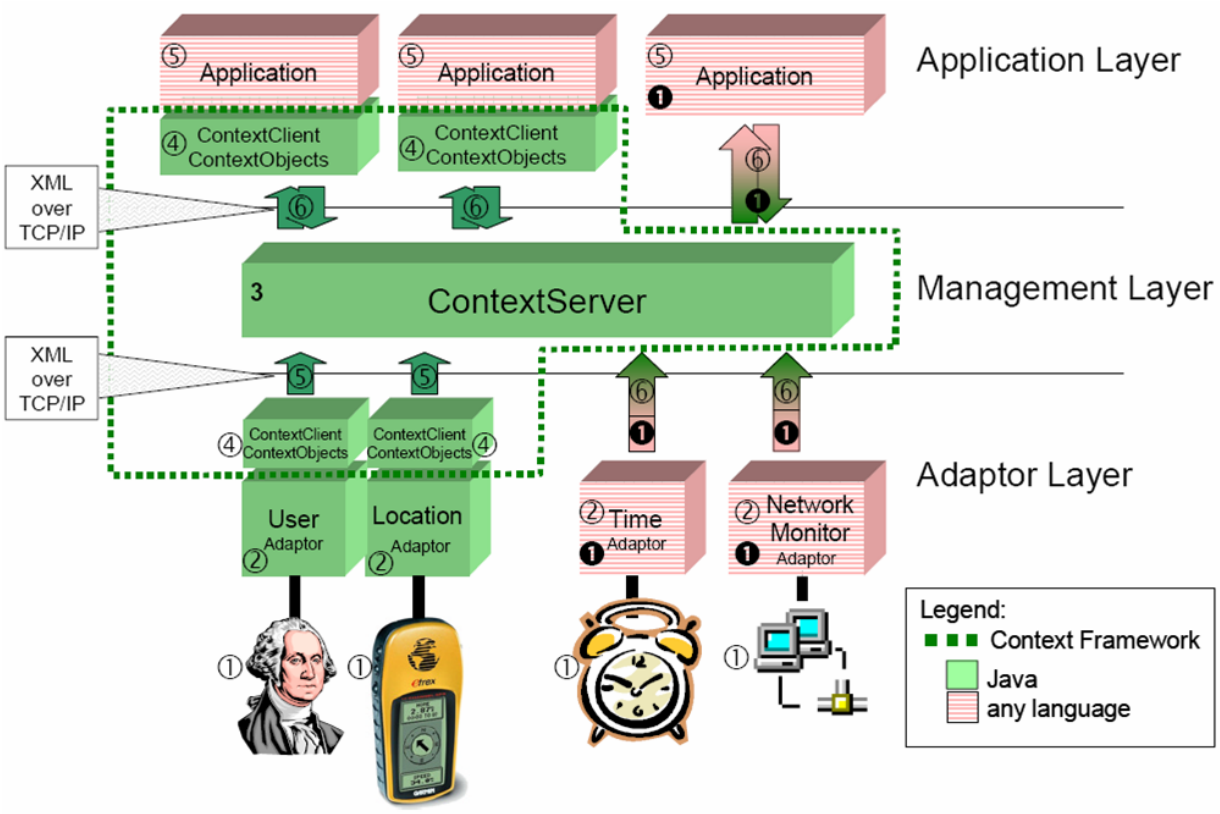

Abbildung 3.2: Hydrogn Context Framework [Hofer et al., 2003]

3.3 Sensor-based Context-Awareness

...for Adaptive PDA User Interfaces [Schmidt et al., 1998]. Diese Arbeit beschäftigt sich, mit den Einsatzmöglichkeiten von Sensoren zur automatisierten Anpassung von PDA Displays, bzw. der darauf dargestellten Benutzerschnittstellen. Als Beispiele werden Lichtsensoren genannt, anhand derer eine automatische Anpassung der Display Helligkeit an das Umgebungslicht ermöglicht wird. Zusätzlich werden Lagesensoren betrachtet, mit deren Hilfe sich z.B. die Darstellung auf einem Display stets ”richtig” herum drehen lässt. D.h. wenn ein Benutzer einen PDA dreht, muss er die Darstellung nicht manuell anpassen.

Es handelt sich bei diesem Ansatz also um eine Kombination aus Context-Awareness und Self-Management. Da der Fokus dieses Ansatzes liegt explizit auf mobilen Geräten. Eine gesonderte Betrachtung der Eignung für mobile Umgebungen erübrigt sich somit.

3.4 AW2 Projekt

Das AW2 Projekt "Pervasive Gaming Framework" (Projektbericht in Kürze zu finden auf [UbiComp]) beschäftigt sich mit ortsbezogenen Spielen auf mobilen Geräten. Ein wichtiger Aspekt dabei ist Location-Awareness, also ein Unterthema von Context-Awareness. Das Framework stellt Positionsinformationen zur Verfügung und reagiert auf Positionsänderungen. Anwendungen, bzw. Spiele die auf dem Framework basieren, können diese Informationen nutzen, um ihr Verhalten zu beeinflussen. Die Abbildungen 3.3 und 3.4 zeigen die Grobarchitektur des Framworks. Wie den Abbildungen zu entnehmen ist, verfügen sowohl Client, als auch Server über einen Context Manager, der Context-Informationen verwaltet.

Das Framework bietet selbst keine Self-Management Funktionalitäten. Die Verantwortung für die Verwertung von Context-Informationen liegt bei den konkreten Anwendungen. In der aktuellen Implementierung bestehen die Context-Informationen hauptsächlich aus Positionsinformationen. Konzeptionell ist das Framework auf beliebige Context-Informationen ausgelegt. Für den Austausch von Context-Informationen zwischen Client und Server, wird bei jeder Kommunikation ein erweiterbares Context Objekt übertragen.

Das Framework wurde unter Microsoft .Net Compact Framework und C# entwickelt und speziell für mobile Umgebungen ausgelegt. Eine gesonderte Betrachtung der Eignung für mobile Umgebungen erübrigt sich somit auch hier.

3.5 Weitere interessante Arbeiten

Im Folgenden werden weitere Arbeiten im Umfeld von Self-Management und Context-Awareness vorgestellt. Für eine genauere Betrachtung sei an dieser Stelle auf die angegebenen Quellen verwiesen.

3.5.1 SOLAR

...Design and Implementation of a Large-Scale Context Fusion Network [Chen et al., 2004]. Diese Arbeit beschäftigt sich mit weit verteilten Sensoren und der Akkumulation von Context-Informationen. Der Ansatz basiert auf P2P Technologien und flexiblen Graphen.

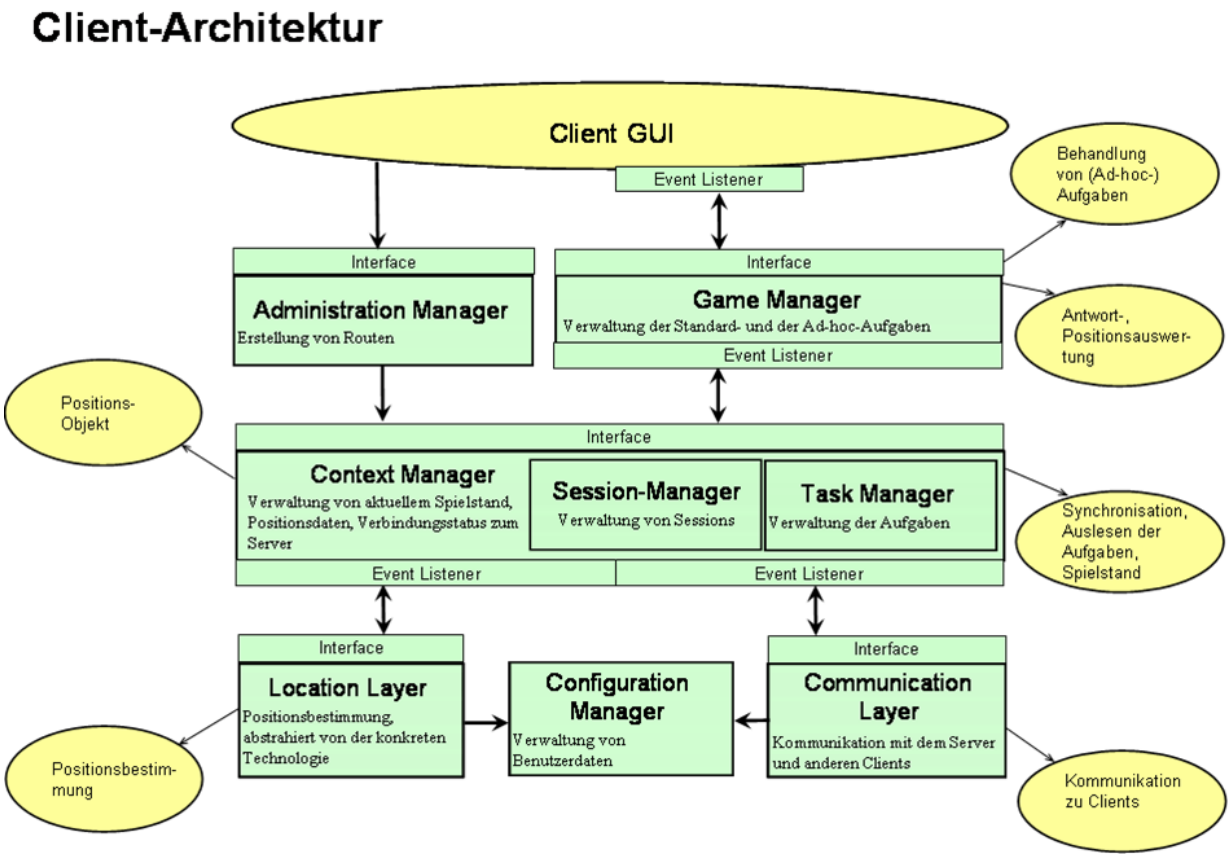

Abbildung 3.3: Pervasive Gaming Framework (Client Architektur)

3.5.2 Call Forwarding

...Active Badge location System ACM, 1992 (beschrieben durch [Chen und Kotz, 2000]). Wie schon in Kapitel 2.3 beschrieben, sind Self-Management und Context-Awareness keine neuen Themen. Das Active Badge location System von 1992 ist eines der ersten Beispiele für den Einsatz entsprechender Technologien. Bei diesem System wurde die Position von Mitarbeitern anhand eines Badge Systems getracked. Anrufe für bestimmte Personen wurden dann automatisch an ein Telefon in ihrer Nähe weitergeleitet.

3.5.3 Context-aware Service Protocol

Das Context-aware Service Protocol [Tan et al., 2003] ist ein auf dem SIP Standard basierendes Protokoll für die Übertragung von Context-Informationen. Es gibt bisher keine einheitlichen Standards für das Format von Context-Informationen und deren Übertragung. Das Context-aware Service Protocol stellt einen Versuch dar, diese Situation zu ändern.

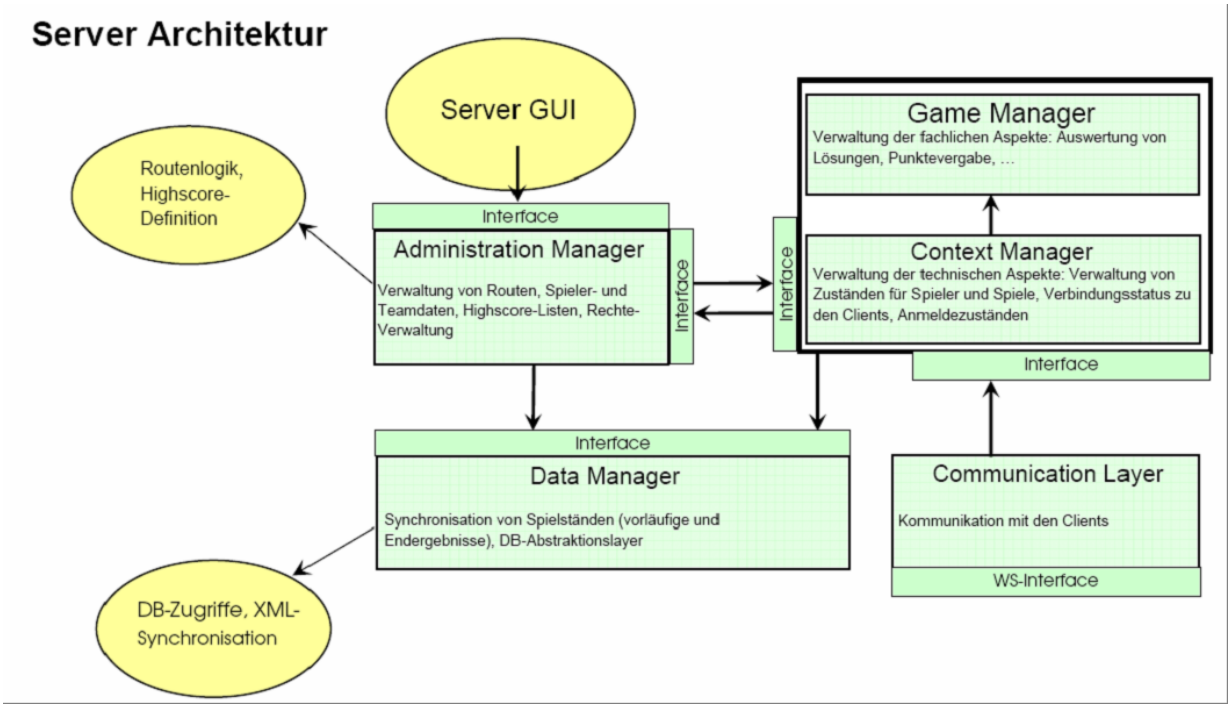

Abbildung 3.4: Pervasive Gaming Framework (Server Architektur)

3.5.4 A Formalism for Context-Aware Mobile Computing

Ein grundlegendes Problem von Context-Awareness ist die Überführung von ”low level” Context-Informationen in ”high level” Context-Informationen (siehe 2.2). In den meisten Arbeiten wird auf dieses Thema nicht eingegangen. A Formalism for Context-Aware Mobile Computing [Yan und Sere, 2004] beschreibt einen formalen Ansatz für die Überführung von Context-Informationen.

Kapitel 4

Fazit

Context-Awareness und Self-Management werden zunehmend relevant. In Kapitel 3 wurde gezeigt, dass sich diese beiden Themen ergänzen. Systeme, die über Context-Informationen verfügen, diese aber nicht für Self-Management nutzen, verschenken einen großen Teil des Potentials, der in der Kombination dieser Themen steckt. Ein System, welches über Self-Management Fähigkeiten verfügt, ohne Context-Informationen ermitteln zu können, verschenkt hingegen nicht nur Potential, sondern ist kaum zu realisieren.

Mobile Umgebungen sind ein ideales Einsatzgebiet Context-Awareness und Self-Management, da hier typischerweise weit mehr Context-Informationen zu finden sind, als in nicht-mobilen Umgebungen. Für die Anwender ist eine deutliche Verbesserung der Usability und eine Erweiterung der Funktionalität erreichbar.

Es bleibt abzuwarten, welche Konzepte, Standards und Technologien sich im Bereich der Context-Awareness und des Self-Management etablieren werden. Das Potential dieser Themen wird bisher nur in geringem Maße ausgeschöpft.

Literaturverzeichnis

[Ganek und Corbi, 2003] A.G. Ganek, T.A. Corbi: **The dawning of the autonomic computing era**, IBM Systems Journal, 42(1), (2003).

[Schmidt et al., 1998] A. Schmidt, M. Beigl, H.-W. Gellersen: **There is more to Context than Location**, IEEE (1998).

[Hofer et al., 2003] T. Hofer et al.: **Context-Awareness on Mobile Devices the Hydrogen Approach**, IEEE (2003).

[Yan und Sere, 2004] L. Yan, K. Sere: **A Formalism for Context-Aware Mobile Computing**, IEEE (2004).

[Barkhuus und Dey, 2003] L. Barkhuus, A. Dey: **Is Context-Aware Computing Taking Control Away from the User?**, Proceedings of UbiComp (2003).

[Chen und Kotz, 2000] G. Chen, D. Kotz: **A Survey of Context-Aware Mobile Computing Research**, IEEE (2000).

[Chen et al., 2004] G. Chen et al.: **Design and Implementation of a Large-Scale Context Fusion Network**, IEEE (2004).

[Tan et al., 2003] C.-P. Tan et al.: **Context-aware Service Protocol**, IEEE (2003).

[BSI, 2006] Bundesamt für Sicherheit in der Informationstechnik: **Pervasive Computing: Entwicklungen und Auswirkungen**, BSI (2006). - ISBN 3-922746-75-6

[IBM, 2001] ibm.com: **Autonomic Computing**, (2001). Im Internet zu finden unter <http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf> (Februar 2007)

[UbiComp] UbiComp @ Informatik.HAW Hamburg. Im Internet zu finden unter <http://users.informatik.haw-hamburg.de/ ubicomp/projects.html> (Februar 2007)

[IBM, 2005] ibm.com: **An architectural blueprint for autonomic computing.**, (2005). Im Internet zu finden unter <http://www03.ibm.com/autonomic/blueprint.shtml> (Februar 2007)