Einsatz von XML zur Kontextspeicherung in einem agentenbasierten ubiquitären System

Faruk Bagci, Jan Petzold, Wolfgang Trumler und Theo Ungerer

Institut für Informatik
Universität Augsburg, 86159 Augsburg
{bagci,petzold,trumler,ungerer}@informatik.uni-augsburg.de

Abstract: „Ubiquitous Computing“ bezeichnet eine Zukunftsvision: Mit Mikroelektronik angereicherte Gegenstände sollen so alltäglich werden, dass die enthaltenen Rechner als solche nicht mehr wahrgenommen werden. Damit sich ein ubiquitäres System in hohem Maße auf den Menschen einstellen kann, benötigt es Information über seine Umgebung. Diese Information wird als Kontext bezeichnet und muss auch in kleinen, tragbaren Geräten darstellbar und zugreifbar sein. Diese Arbeit stellt eine Methode vor, wie Kontextinformation für ubiquitäre Systeme mit XML dargestellt und manipuliert werden kann.

1 Einleitung

Ubiquitäre Systeme sind eine Erweiterung der „eingebetteten Systeme“. Die letzteren sind Rechnersysteme zur Steuerung von technischen Abläufen, also z.B. zur Fertigungssteuerung in einer Fabrik, als Fahrkartenautomat oder als ABS im Auto. Als ubiquitäre (allgegenwärtige) Systeme bezeichnet man eingebettete Rechnersysteme, die selbstständig auf ihre Umwelt reagieren. Bei einem ubiquitären System kommt zusätzlich zu einem eingebetteten System noch Umgebungswissen hinzu, das es diesem System erlaubt, sich in hohem Maße auf den Menschen einzustellen. Die Benutzer sollen nicht in eine virtuelle Welt gezogen werden, sondern die gewohnte Umgebung soll mit Computerleistung angereichert werden, so dass neue Dienste entstehen, die den Menschen entlasten und ihn von Routineaufgaben befreien [Wei91, DG02].

Technisch gesehen sind für ein ubiquitäres System viele kleine, oftmals tragbare, in Geräten versteckte Mikroprozessoren notwendig, die über Sensoren mit der Umwelt verbunden sind und bisweilen auch über Aktuatoren aktiv in diese eingreifen. Ubiquitäre Systeme binden neue Geräte ein wie z.B. Handhelds, Mobiltelefone und am Körper getragene („wearable“) Rechner. Verbunden sind diese Rechner untereinander und mit dem Internet über drahtgebundene oder drahtlose Netzwerke.

Die Einbeziehung von Informationen aus der natürlichen Umgebung der Geräte stellt ein wesentliches Kennzeichen ubiquitärer Systeme dar. Die Berücksichtigung der Umgebung, des Kontexts, geschieht über die Erfassung, Interpretation, Speicherung und Verbindung

von Sensorendaten. Oftmals kommen Systeme zur orts- und richtungsabhängigen Informationsinterpretation auf mobilen Geräten hinzu. Das Gerät passt sich in seinem Verhalten der jeweiligen Umgebung an und wird damit ortssensitiv.

2 Eigenschaften und Anforderungen an eine Kontext-Modellierung

Ein Kontext beschreibt in ubiquitären Systemen den Zustand, der durch Auswertung von Informationen über die Umgebung eines Benutzers oder Systems entsteht [AEH+02]. Kontextinformationen können einzelne Zustandsinformationen wie „An“ oder „Aus“ im Falle einer Lampe betreffen, aber auch Zustände mit vielen komplexen Informationen beschreiben. Ein Beispiel für einen komplexen Kontext stellt ein Raumkontext dar, der die Nachbarräume, Personen im Raum, Zustand des Lichtes usw. umfasst. Kontexte können sich oftmals sehr schnell hintereinander ändern.

Die Kontext-Eigenschaften lassen sich wie folgt charakterisieren:

- verschiedene Wertigkeiten, z.B. binär, diskret, kontinuierlich, Intervall
- zeit- und ortsabhängig
- dynamisch
- untereinander korreliert
- zu komplexeren Kontexten zusammensetzbar

Neben diesen Eigenschaften müssen weiterhin die folgenden Anforderungen an eine Kontextmodellierung in Betracht gezogen werden:

- Austauschbarkeit zwischen verschiedenen vorher noch nicht bekannten Geräten, deshalb Aufbau auf einer Standarddarstellung,
- dezentrale Darstellung und Speicherung von Kontextinformationen,
- vielseitige Weiterverarbeitung auf rechenstarken PCs oder Workstations wie auch batteriebetriebenen PDAs und Wearable Computers,
- sprachunabhängiger Zugriff, da keine einheitliche Hardware und Software innerhalb eines ubiquitären Systems vorhanden sein muss,
- trotzdem hochstehende Such- und Manipulationsmöglichkeiten nötig.

3 Kontextmodellierung

Die Modellierung eines Kontexts in ubiquitären Systemen erfolgt in mehreren Stufen. Der einfachste Kontext, welcher eine einzelne Zustandsinformation beschreibt, besteht aus einem Bezeichner und einem Zustand. Durch Zusammenfügen mehrerer einfacher Kontexte

ensteht ein zusammengesetzter Kontext, welcher unter einer neuen Bezeichnung gekapselt wird. Mehrere zusammengesetzte Kontexte bilden schließlich einen komplexen Kontext. Die einzelnen Kontexte werden dabei wiederum unter einem Bezeichner gekapselt. Damit ergibt sich ein hierarchischer Aufbau, der als markierter Baum modelliert werden kann.

Gegeben sei eine Menge von Bezeichnern L sowie eine Menge von Zuständen S. Gesucht ist die Menge der Kontexte C. Ein Kontext $c \in C$ kann dann eine der folgenden Formen haben:

- Einfacher Kontext: $c := (L, S)$

  Beispiel: `Stromkontext = (Strom, an)`

- Zusammengesetzter Kontext: $c := (L, (L, S)^+)$

  Beispiel:
  ```
  Lichtkontext = (Licht, (Strom, an),
                         (Leistung, 60))
  ```

- Komplexer Kontext: $c := (L, C^+)$

  Beispiel:
  ```
  Raumkontext = (Raum, (Licht, (Strom, an),
                               (Leistung, 60)),
                       (Radio, (Strom, an),
                               (Lautstärke, 12),
                               (Sender, 3)))
  ```

Aufgrund dieser Modellierung, speziell des strukturierten Aufbaus von komplexen Kontexten, und der Anforderungen aus dem vorangegangenen Abschnitt haben wir uns für eine XML-basierte Umsetzung [W3C02] dieses theoretischen Modells entschieden. Anhand des Beispiels wollen wir diese Umsetzung aufzeigen:

```
<room name="Raum">
  <device name="Licht">
    <function name="Strom" value="an"/>
    <function name="Leistung" value="60"/>
  </device>
  <device name="Radio">
    <function name="Strom" value="an"/>
    <function name="Lautstärke" value="12"/>
    <function name="Sender" value="3"/>
  </device>
</room>
```

Das Beispiel zeigt einen Raumkontext, wie er innerhalb eines Gebäudekomplexes für jeden Raum existieren könnte. Der Raumkontext setzt sich hierbei aus den Kontexten verschiedener Geräte zusammen. Raumkontexte können lokal auf Rechnern in den jeweiligen Räumen verfügbar sein und aufgrund der standardisierten Darstellung zwischen diesen potentiell heterogenen Rechner ausgetauscht werden. Die Kontextinformationen werden von

den ubiquitären Systemkomponenten (z.B. den im nächsten Abschnitt beschriebenen mobilen Agenten) genutzt. Der Zugriff geschieht dabei über vordefinierte Methoden.

Eine alternative Methode der Kontextmodellierung stellt das Context Toolkit [SDA99] des GeorgiaTech dar. Der Ansatz beim Context Toolkit besteht darin, Software-Komponenten, so genannte „Context Widgets“, anzubieten, die Applikationen Zugriff auf Kontextinformationen aus der Operationsumgebung liefern. Vergleichbar mit GUI (Graphical User Interface) Komponenten, wie zum Beispiel Buttons, führen Context Widgets bestimmte Funktionen aus, wobei die Komplexität der Datenerfassung für den Benutzer völlig verborgen bleibt. Eine Anwendung kann über einen Kontext-Server auf gewünschte Context Widgets zugreifen. Dabei bieten sogenannte Interpreter eine Unterstützung, um aus verschiedenen Sensordaten zusammengestellte Kontexte zu erkennen, d.h. sie bündeln mehrere Context Widgets zu einer Komponente.

XML wird im Context Toolkit lediglich zur Kommunikation zwischen Client und Server eingesetzt. Die gesammelten Kontextdaten können in einer Datenbank abgespeichert werden, wobei hier keine einheitliche Modellierung der Kontexte ersichtlich ist. Durch den zentralen Server-Ansatz ist das Context Toolkit fehleranfällig. Sowohl die Kontexterfassung als auch die Speicherung wird zentral geregelt, so dass ein Ausfall des Servers den Stillstand des gesamten Kontextsystems bedeutet. Ein weiterer Nachteil ist, dass es keine generische Kontextbeschreibung existiert, die einen Austausch von Informationen unter unterschiedlichen Systemen ermöglichen würde.

4 Evaluierung der Kontextmodellierung in einem agentenbasierten ubiquitären System

Die XML-basierte Kontextmodellierung wird in einem agentenbasierten ubiquitären System zur Objektverfolgung in Räumen eines ubiquitär vernetzten Gebäudes eingesetzt.

Mobile Agentensysteme können Code, Daten und Ausführungszustand an einen entfernten Rechner transferieren und dort mit dem Programmablauf fortfahren. Der größte Vorteil von mobilen Agenten gegenüber zentralisierten serverbasierten Systemen ergibt sich dadurch, dass sie in höchstem Maße verteilt und dezentral arbeiten. Bei der Vision, dass das menschliche Umfeld immer mehr intelligente Geräte umfassen wird, erscheint eine dezentrale Lösung notwendig.

Alle Räume des Gebäudes sind mit vernetzten Rechnern ausgestattet, die jeweils mit den Sensoren und Aktuatoren des Raums verbunden sind. Jeder Rechner trägt somit einen „Raumkontext“. Die Objekte, z.B. Menschen, bewegen sich durch diese Räume und werden dabei von einem mobilen Agenten, der ihre persönliche Kontextinformationen trägt, von Raum zu Raum begleitet. Die persönliche Kontextinformation (Vorlieben, Personendaten usw.) kann aus Datenschutzgründen in dem Agenten gekapselt und für den Raumrechner nicht ohne weiteres zugreifbar sein. Auf dem Raumrechner treffen sich die persönlichen Agenten verschiedener Objekte, die sich in dem Raum befinden, und interagieren miteinander sowie mit dem Raumkontext.

Im Rahmen eines Forschungsprojekts an der Universität Augsburg wird der Einsatz mobiler Agenten für ubiquitäre Systeme erforscht. Ausgangsbasis stellt das mobile Agentensystem JMessengers [GKPU01, WKU01] dar, ein Java-basiertes mobiles Agentensystem, das ursprünglich für das verteilte Rechnen in heterogenen Rechnernetzen entworfen wurde. JMessengers zeichnet sich durch besonders leichtgewichtige Agenten aus und ermöglicht somit einen sehr schnellen Transport der lokalen Daten von einem Rechner zum nächsten. Das JMessengers-System wurde für den Einsatz in ubiquitären Systemen erweitert, so dass die Agenten XML-modellierte Kontextinformation von Rechner zu Rechner transportieren können. Ein Agent verfügt über vordefinierte Methoden, die den Zugriff auf die XML-Datei ermöglichen.

Während der Evaluierungsphase stellte sich heraus, dass insbesondere beim Einsatz von mobilen Geräten, wie PDAs, die XML-Zugriffe lange dauern und es dadurch zu Zeitverzögerungen kommen kann. Um dem entgegen zu wirken, wird beim ersten Eintritt des Agenten in den Raum die XML-Information in eine Datenstruktur transferiert. Diese Datenstruktur definiert ein Abbild des in der XML-Datei festgehaltenen Raumkontexts und ermöglicht einen schnellen und effektiven Zugriff auf die Kontextdaten. Sowohl Änderungen der Kontexte, als auch der Kontextstruktur (z. B. neue Geräte kommen hinzu) sind mit der Datenstruktur weiterhin möglich.

Als Java-basierte Lösung für das Lesen, Modifizieren und Zurückschreiben der XML-Informationen wurde das JDOM Softwarepaket [JDO02] verwendet. Da Kontexte i.A. relativ klein sind, ist der Einsatz auf PDAs ohne Probleme möglich. Durch die geringe Zugriffshäufigkeit und ihre geringe Größe können die XML-Dokumente mittels JDOM stets effizient verarbeitet werden.

Sobald der Agent eine Änderung in dem Raumkontext erkennt (beispielsweise eine Person betritt einen Raum oder der Zustand eines Gerätes ändert sich) aktualisiert er die Datenstruktur. Der Agent kennt dabei den Kontext der Person, den er „betreut“ und passt den Kontext des Raumes entsprechend an. Jede Aktualisierung wird sofort in die zugehörige XML-Datei geschrieben, damit der nächste Agent immer über das aktuelle Kontextwissen verfügt.

Da es unter Umständen vorkommen kann, dass sich viele verschiedene Agenten innerhalb eines Raums aufhalten und jeder den Kontext dieses Raums erfragen oder modifizieren kann, muss der Zugriff auf Kontextdaten synchronisiert werden. Um konkurrierenden Zugriff zu ermöglichen und Verklemmungen zu vermeiden, verfügt das JMessengers-System über verschiedene Synchronisationsverfahren [WKU01]. Diese werden in dem Kontextsystem erfolgreich eingesetzt.

Das Gebäudesystem wurde bisher auf einem Simulator realisiert. Räume, Personen, Geräte und deren Kontexte werden auf einem Grundrissplan visuell dargestellt. Tests mit mobilen und stationären Geräten, die eine Objekterkennung und -verfolgung mittels Infrarot (IrDA) und eine Übertragung der Agenten- und Kontextinformationen mittels Bluetooth durchführen, zeigen, dass die XML-Darstellung und -Zugriff keinen Engpass darstellen. Lesezugriffe auf Kontextinformationen können mittels der speziellen Datenstruktur, die sich im Speicher befindet, effizient ausgeführt werden. Nur Kontextänderungen lösen einen Schreibzugriff auf die zugehörige XML-Datei aus.

5 Zusammenfassung und Ausblick

Es wurde eine XML-Darstellung der Kontext-Informationen eines ubiquitären mobilen Agentensystems vorgestellt, die Implementierung im mobilen Agentensystem JMessengers beschrieben und eine Anwendung auf die Objektverfolgung in Räumen eines ubiquitär-vernetzten Gebäudes simuliert.

Die Erfahrungen zeigen, dass sich die gewählte XML-basierte Kontextmodellierung sehr gut für ubiquitäre Systeme, auf dezentralen, heterogen zusammengesetzten Systemplattformen eignet. Für einen schnellen Zugriff muss die XML-Datei allerdings in eine lokale Datenstruktur übertragen werden.

Als nächster Schritt ist eine prototypische Umsetzung des Systems innerhalb des Institutsgebäudes an der Universität Augsburg geplant.

Literatur

[AEH+02] Gregory D. Abowd, Maria Ebling, Guerney Hund, Hui Lei, and Hans-Werner Gellersen. Context-Aware Computing. In *IEEE Pervasive Computing*, Juli 2002.

[DG02] Nigel Davies and Hans-Werner Gellersen. Beyond Prototypes: Challenges in Deploying Ubiquitous Systems. In *IEEE Pervasive Computing*, Jan. 2002.

[GKPU01] M. Gmelin, J. Kreuzinger, M. Pfeffer, and Th. Ungerer. Agent-based Distributed Computing with JMessengers. In *I2CS Innovative Internet Computing Systems*, pages 131–145, Feb. 2001.

[JDO02] http://www.jdom.org, 2002.

[SDA99] Daniel Salber, Anind K. Dey, and Gregory D. Abowd. The Context Toolkit: Aiding the Development of Context-Enabled Applications. In *Proceedings of CHI'99*, pages 434–441, Mai 1999.

[W3C02] W3C. http://www.w3.org, 2002.

[Wei91] Mark Weiser. The Computer for the 21st Century. In *Scientific American*, pages 94–104, Sept. 1991.

[WKU01] U. Wolf, J. Kreuzinger, and Th. Ungerer. Synchronisation im JMessengers Agentensystem. In *PARS-Workshop, München*, pages 87–96, Okt. 2001.