Proc. Informatik 2004, 34. Jahrestagung der GI, 20.-24. Sept. 2004, Ulm,
GI Lecture Notes in Informatics, P-50, 2004, pp. 303-307

# Verarbeitung von Ontologien in mobilen Umgebungen

Timo Weithöner und Günther Specht
Universität Ulm, Fakultät für Informatik
{specht, weithoener}@informatik.uni-ulm.de

**Abstract:** Mobile Informationssysteme können erheblich von Ontologien profitieren, indem zuvor zusammenhanglose Fakten für den Anwender in Beziehung gebracht werden. In dieser Arbeit untersuchen wir drei mögliche Architekturvarianten zur mobilen Verarbeitung von Ontologien. Wir zeigen, dass ein Ansatz, beruhend auf einem mobilen Datenbanksystem, trotz seiner heutigen Limitationen in Zukunft zu bevorzugen sein wird. Bei der Verarbeitung von Ontologien auf mobilen Clients treten darüber hinaus interessante neue Anforderungen an Replikations- und Synchronisationsmechanismen auf, die um Unifikation erweitert werden müssen.

## 1 Motivation

Heutige mobile Informationssysteme machen dem Anwender in der Regel eine Menge von zuvor festgelegten Fakten zugänglich. Solche Fakten können zum Beispiel aus Tupeln einer Datenbank oder dem Inhalt von Webseiten zusammengesetzt sein. Ein mobiles Informationssystem, das zusätzlich in der Lage wäre, Beziehungen und Abhängigkeiten zu erkennen, könnte den Anwender deutlich effektiver unterstützen. Kern eines solchen Systems stellt eine formale Wissenrepräsentation – auch als Ontologie bezeichnet – dar, auf der ein Rechner automatisiert Schlussfolgerungen ziehen kann. Ontologien spielen eine zentrale Rolle im Wissensmanagement. Sie werden heute, auf Grund der Größe der Folgerungskomponenten ausnahmslos im stationären Umfeld eingesetzt. Möchte man die Vorteile von Ontologien auch in mobilen Umgebungen nutzen, stellen sich eine Reihe interessante (Architektur-)Fragen.

In dieser Arbeit werden nach einer kurzen Einführung in Ontologien (Abschnitt 2) drei verschiedene Architekturvarianten zur mobilen Ontologieverarbeitung vorgestellt (Abschnitt 3). Dabei zeigt sich, dass die bekannten Konzepte von Replikation und Synchronisation im Zusammenspiel mit Ontologien zu kurz greifen und um Unifikation erweitert werden müssen (Abschnitt 4). Abschnitt 5 fasst die Ergebnisse zusammen.

## 2 Wissensrepräsentation in Ontologien

Ontologien sind formale Wissensrepräsentationen, heute meist in Form von Beschreibungslogiken, die es Maschinen ermöglichen Wissen zu analysieren und zueinander in Beziehung zu setzen. Ziel ist, den Wissensaustausch sowohl zwischen Mensch und Ma-

schine als auch zwischen Maschinen untereinander zu formalisieren.

Dazu beschreibt eine Ontologie eine Menge von Individuen, die charakterisiert sind durch ihre Zugehörigkeit zu bestimmten Konzepten und die Ausprägung bestimmter Eigenschaften. Zwischen diesen sind Beziehungen und Abhängigkeiten definiert. Anders als in einem objektorientierten Modell dienen Klassen in einer Ontologie jedoch nicht der Typisierung von Objekten, sondern zur Klassifikation von Individuen abhängig von deren Eigenschaften [MM03].

Die vom W3C vorgeschlagene Web Ontology Language (OWL) [MvH04], stellt einen Standard zum Publizieren und Verteilen von Ontologien im World Wide Web dar. OWL basiert auf RDF Schema[1], das wiederum auf Basis des Resource Descriptor Framework[2] realisiert ist. Diese Sprachen nutzten XML zur Auszeichnung der Modellierungsprimitive.

Zur Verarbeitung von Ontologien kommen verschiedene Folgerungssysteme (auch Beweissysteme genannt) zum Einsatz. Mit unterschiedlichen Stärken und Schwächen können Tableaubeweiser, Prologsysteme oder deduktive Datenbanken zur Schlussfolgerung auf Ontologien verwendet werden.

Als Beispiele für solche Beweissysteme seien hier Racer [HM03] oder FaCT [Ho98] genannt. Alternativ dazu haben wir in [WLS03] einen Ansatz zur Speicherung und Verarbeitung von Ontologien in deduktiven Datenbanken beschrieben. Stets erfordert die Verarbeitung der Ontologie eine Konvertierung in die vom Folgerungssystem geforderte Notation. Dies kann zum Beispiel ein Tableauprogramm, ein Logik- oder ein Lispprogramm sein.

## 3 Architekturvarianten

Wir gehen von einem Anwendungsszenario aus, in dem eine Menge vergleichsweise leistungsschwacher mobiler Endgeräte mit begrenztem Speicher (PDAs) dazu eingesetzt wird, verschiedene Teilbereich einer gemeinsamen Ontologie zu nutzen und zu bearbeiten. Wir gehen weiterhin davon aus, dass die mobilen Endgeräte über ein drahtloses Netzwerk schwach mit einem fest vernetzen Backend verbunden sind. In diesem Backend stehen leistungsstarke Rechner zur Verfügung, die je nach Architektur beispielsweise ein zentrales Folgerungssystem aufnehmen können. Auch soll die Häufigkeit der Verbindungen zum Festnetz minimiert werden. Gründe hierfür könnten die schlechte Verfügbarkeit des drahtlosen Netzwerkes oder Kostenaspekte sein.

Insgesamt könnte das oben umrissene Szenario für ein medizinisches Informationssystem gelten, in dem das medizinische Personal mit PDAs ausgestattet ist, über die es Patienteninformationen abrufen und erfassen kann sowie gegebenenfalls bei der Diagnose und beim Verschreiben geeigneter Medikamente unterstützt wird. Dies sowohl in der Klinik oder Praxis als auch bei Hausbesuchen.

Im Folgenden sollen nun verschiedene Möglichkeiten dargestellt werden, wie ein System im oben beschriebenen Umfeld realisiert werden kann.

[1] RDF Schema: http://www.w3.org/TR/2004/REC-rdf-schema-20040210/

[2] RDF Concepts and Abstract Syntax: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

**Variante 1: Zentrales Folgerungssystem.** Der triviale Ansatz ist sicherlich die komplette Verwaltung der verwendeten Ontologien zusammen mit der Folgerungskomponente im Backend anzusiedeln und auf dem mobilen Endgerät lediglich einen schlanken Client zu installieren, der sich für alle Operationen mit dem Folgerungssystem verbindet. Ein solches System ließe sich beispielsweise mit Racer [HM03] im Backend realisieren. Wünschenswert wäre dabei die Verwendung eines standardisierten Kommunikationsprotokolls[3].

Nachteil: Ohne Verbindung zum Backend, kann der Client keine Informationen nutzen oder bearbeiten. Er muss demnach während der Verarbeitung permanent mit dem Netzwerk verbunden bleiben.

**Variante 2: Folgerungssystem auf dem Client.** Soll ein Arbeiten ohne permanente Verbindung zum Backend möglich gemacht werden, ist es erforderlich auf dem mobilen Endgerät ein Folgerungssystem zu installieren. Heute verfügbare Folgerungssysteme haben einen großen Hauptspeicherbedarf und erzeugen bei der Auswertung von Anfragen eine hohe Rechenlast. Sie sind daher für den Einsatz auf PDAs gänzlich ungeeignet und sind dort auch auf absehbare Zeit nicht zum Laufen zu bringen.

**Variante 3: Client mit mobilem Datenbanksystem.** Ontologien lassen sich auch in Datenbanksystemen speichern und abfragen. In [WLS03] haben wir gezeigt, dass sich deduktive Datenbanken dazu besonders gut eignen. Mit dem gegenwärtigem Zurückfließen deduktive Techniken in die relationalen Datenbanksysteme (SQL4), werden sich auch diese künftig dazu eignen. Dazu ist es erforderlich die Ontologie auf ein relationales Datenbankschema abzubilden auf dem dann Views und Funktionen definiert werden, die die Funktionalität des Resolutionssystems der deduktiven Datenbank implementieren. Erst mit dieser Ergänzung wird Folgern mittels einer relationalen Datenbank möglich.

In dieser Architekturvariante wird auf dem mobilen Endgerät ein leichtgewichtiges mobiles Datenbanksystem [MS04] mit oben skizzierter Folgerungskomponente installiert, das für den jeweils vom Anwender gewünschten Ausschnitt, alle relevanten Konzepte und Instanzen der Ontologie enthält. Die im Backend auf einem leistungsfähigen Server installierte zentrale Datenbank enthält die komplette Ontologie und verfügt über dementsprechend umfangreiche Folgerungskapazität.

Folgende Zahlen machen den Vorteil des Einsatzes von mobilen Datenbanksystemen klar: Eine typische DB2 Everyplace Installation benötigt auf einem PDA etwa 200kB Platz. Während eine Racer System – installiert auf einem normalen Windows Arbeitsplatz – 27MB Platz auf der Platte benötigt und bereits unmittelbar nach dem Start 30MB Hauptspeicher reserviert. Dies liegt darin begründet, dass mobile Datenbanken mit dem klaren Fokus auf einen kleinen Speicherfootprint entwickelt wurden, während Folgerungssysteme wie Racer rein auf Anfragegeschwindigkeit optimiert sind.

Weiterer Vorteil ist dabei, dass Replikation und Synchronisation durch das Datenbanksystem geregelt werden. Allerdings kann man bei der Auswahl der zu übertragenden Daten nicht den Regeln gewöhnlicher relationaler Datenmodelle folgen. Hierzu mehr in Abschnitt 4. Ergänzend ist sowohl beim Editieren der als auch bei vielen Anfragen an die

[3] Ein erster – für viele Anwendungen allerdings noch ungenügender – Entwurf eines solchen Protokolls ist DIG [BMC03].

Ontologie keine Onlineverbindung nötig. Die Architektur, beruhend auf einem mobilen Datenbanksystem, ist skalierbarer und universeller in ihrem Einsatz. So kann die Menge der replizierten Daten in Abhängigkeit von der Leistungsfähigkeit des Gerätes gesteigert werden bis zum kompletten Replikat der Ontologie.

Nachteile dieser Architektur liegen in der gegen Variante 1 höheren Leistungsfähigkeit, die von den mobilen Geräten erwartet wird. So muss sich mindestens ein mobiles Datenbanksystem mit Folgerungskomponente installieren und betreiben lassen.

## 4 Replikation und Synchronisation

Um auf dem mobilen Client Schlussfolgerungen aus einer Ontologie ziehen zu können, muss neben dem Folgerungssystem auch die Ontologie lokal verfügbar sein. Da wir von einer zentralen Ontologie ausgehen, muss diese Ontologie, oder besser Teile davon, an den mobilen Client repliziert werden. Änderungen des Clients müssen anschließend mit dem Original und Änderungen anderer Clients synchronisiert werden.

**Replikation.** In einem relationalen Modell erfolgt die Auswahl der zu replizierenden Tupel typischerweise über eine Selektion. So können beispielsweise die Datensätze aller Patienten, die von einem bestimmten Arzt betreut werden, auf den PDA dieses Arztes repliziert werden. Dieses Vorgehen ist bei einer auf Beschreibungslogik beruhenden Ontologie nicht möglich. Anstatt eine Selektion durchzuführen, muss ein Teilgraph ermittelt werden, der alle relevanten Konzepte, Eigenschaften und Instanzen enthält. Anders als im objektorientierten Modell handelt es sich dabei nicht um einen einfachen Teilbaum aller Subklassen einer Ausgangsklasse. Die über Beziehungen und Eigenschaften verbundenen Konzepte und die ihnen zuzuordnenden Instanzen bilden vielmehr einen teilinstanziierten Graphen. Die Größe, des an den Client zu replizierenden Teilgraphen, wird durch die vom Client tatsächlich benötigten Konzepte und Instanzen und die beim Client auszuführenden Anfragen beeinflusst. Dabei wird aus dem Graphen zunächst die “nächste Umgebung” der relevanten Elemente ausgewählt und repliziert. Die Größe des Datenpaketes, das bei diesem Prefetching an den Client übertragen wird, ist abhängig von dessen Leistungsfähigkeit. Kann mit einer so an den Client replizierten Teilontologie, eine dort gestellte Anfrage nicht beantwortet werden, können entweder weitere Teile der Ontologie auf den Client hochgeladen werden oder die Beantwortung der Anfrage an das Backend delegiert werden.

**Synchronisation.** Ändern zwei Clients dieselbe Ontologie, so müssen diese Änderungen im Backend zusammengeführt werden. In einem relationalen Modell werden die Originaltupel im Backend durch die, von den Clients geänderten Tupel, substituiert. Stehen die Änderungen unterschiedlicher Clients miteinander im Konflikt, so werden diese Konflikte automatisch oder manuell aufgelöst. In jedem Fall wird sich am Ende die Änderung eines Clients oder das Original durchsetzen. Alle anderen Versionen werden verworfen.

Im Falle einer Ontologie stehen nicht einzelne Tupel im Konflikt. Vielmehr übermitteln die Clients teilinstanziierte Teilgraphen aus der Ontologie. Bei Konflikten ist es nicht erforderlich, sofort eine Version zu verwerfen. Vielmehr kann zunächst versucht werden, die

beiden konkurrierenden Versionen miteinander zu unifizieren. Ist eine solche Unifikation möglich, sind die übermittelten Teile der Ontologie semantisch nicht miteinander in Konflikt. In folgendem Beispiel bearbeiten zwei Client unabhängig voneinander das Konzept $Patient$ und spielen ihre Änderungen an das zentrale Backend zurück:

| Client 1: | Client 2: |
|---|---|
| $Krank \sqsubseteq Patient$ | $NichtGesund \sqsubseteq Patient$ |

Dies führt zu keinem Konflikt, wenn die voneinander abweichenden Teile der Aussagen äquivalent sind ($Krank \equiv NichtGesund$). Es spielt dabei keine Rolle, ob der Sachverhalt in der Ontologie implizit oder explizit modelliert ist. Lassen sich die Aussagen unifizieren, können beide beibehalten werden und jeder Anwender kann mit "seiner Sicht der Welt" weiterarbeiten

## 5 Zusammenfassung

Wir haben in dieser Arbeit ein Anwendungsszenario für die mobile Ontologieverarbeitung skizziert und gezeigt, dass dabei eine Architektur beruhend auf einem mobilen Datenbanksystem einer Lösung mittels eines anderen Folgerungsystems prinzipiell überlegen ist. Dies liegt vor allem daran, dass keine dauernde Onlineverbindung zwischen Client und Folgerungssystem im Backend bestehen muss und dass mobile Datenbanksysteme Mehrbenutzerfähigkeit, Replikation und Synchronisation inhärent ermöglichen. Wir haben dann gezeigt, wie der von einen Client für eine bestimmte Anwendung benötigte Teil einer Ontologie bestimmt und repliziert werden kann und haben gezeigt, dass beim Synchronisieren konkurrierender Teilontologien auf eine Konfliktauflösung verzichtet werden kann, wenn die konkurrierenden Teile unifizierbar sind.

## Literatur


[BMC03] Bechhofer, S., Möller, R., und Crowther, P.: The DIG Description Logic Interface. In: *DL2003 International Workshop on Description Logics*. Rom, It. 2003.

[HM03] Haarslev, V. und Möller, R.: *RACER User's Guid and Reference Manual Version 1.7.7*. Concordia University, Motreal, Kan. and FH Wedel, Wedel. 2003.

[Ho98] Horrocks, I.: The FaCT system. In: de Swant, H. (Hrsg.), *Automated Reasoning with Analytic Tableaux and Related Methods: International Conference Tableaux'98*. volume 1397 of *LNAI*. S. 307–312. Springer. 1998.

[MM03] Maedche, A. und Motik, B.: Repräsentations- und Anfragesprachen für Ontologien – eine Übersicht. *Datenbank Spektrum*. 6/2003:43–53. 2003.

[MS04] Mutschler, B. und Specht, G.: *Mobile Datenbanksysteme*. Springer. Berlin. 2004.

[MvH04] McGuiness, D. L. und van Harmelen, F. OWL Web Ontology Language Overview. W3C Recommendation. 2004. http://www.w3.org/TR/owl-features/.

[WLS03] Weithöner, T., Liebig, T., und Specht, G.: Storing and Querying Ontologies in Logic Databases. In: *Proceedings of the Workshop "Semantic Web and Databases" at the VLDB 2003*. Berlin. 2003.