XML-basierte Techniken für Location-Based Services

Martin BREUNIG, Thomas BRINKHOFF, Wolfgang BÄR und Jürgen WEITKÄMPER

Zusammenfassung

Die Verwendung der „eXtensible Markup Language“ (XML) stellt einen generellen Trend zur Repräsentation von Daten dar. Sie ist insbesondere für Anwendungen interessant, die auf einen standardisierten Datenaustausch über das Internet angewiesen sind. Aufgrund dieser Entwicklung stellt sich die Frage, wie XML-basierte Techniken für Location-Based Services und bewegte Geoobjekte verwendet bzw. erweitert werden können. In diesem Artikel geben wir einen Überblick über XML-basierte Techniken zur Modellierung und Verwaltung von Geodaten. Dies umfasst sowohl die Speicherung der Daten in Datenbanksystemen als auch deren Darstellung auf mobilen Endgeräten. Anhand eines XML-basierten Routenplaners zeigen wir schließlich die Nutzungsmöglichkeiten der hier vorgestellten Konzepte auf.

1 Einleitung

Zur Zeit lässt sich eine Verlagerung von stationären hin zu mobilen Informationssystemen beobachten. Mobile Anwendungen haben einen Ortsbezug und sind daher im allgemeinen auf die Übermittlung raumbezogener Daten angewiesen ("Location-Based Services", LBS). Oft sind in diesem Kontext auch bewegte Geoobjekte von Relevanz.

Einen generellen Trend stellt die Verwendung von XML zur Repräsentation von Daten dar. Zur Zeit lässt sich kaum ein Anwendungsfeld benennen, in dem nicht XML als Basis von mehr oder weniger standardisierten Datenmodellen eingeführt wird. Dies gilt auch für den Bereich der Repräsentation und Visualisierung von Geodaten. Als Beispiel ist die „Geography Markup Language“ zu nennen (OGC 2001b).

Unter Berücksichtigung dieser beiden Entwicklungen geben wir hier einen Überblick über XML-basierte Techniken zur Modellierung und Verwaltung von Geodaten. Abschnitt 2 beschäftigt sich mit dem Einsatz von XML für Geodaten. Das Thema des dritten Abschnitts ist die Verwaltung XML-basierter Daten in Datenbanksystemen. Im vierten Abschnitt wird die Grundstruktur einer Architektur zur Nutzung von XML für Location-Based Services vorgestellt und auf die Entwicklung entsprechender Viewer auf mobilen Endgeräten eingegangen. Anhand eines XML-basierten Routenplaners zeigen wir schließlich im Abschnitt 5 die Nutzungsmöglichkeiten der vorgestellten Konzepte auf. Die Darstellung endet mit einem kurzen Ausblick auf künftige Arbeiten.

Breunig M., Brinkhoff T., Bär W., Weitkämper J.: „XML-basierte Techniken für Location-Based Services“, in: A. Zipf, F.J. Strobl (Hrsg.): *Geoinformation mobil - Grundlagen und Perspektiven von Location Based Services*, Wichmann-Verlag, 2002, S. 26-35.

2 XML für Geodaten

2.1 Die „Geography Markup Language“ GML

Das „Feature Geometry Model“ des OGC (OGC 2001a) definiert ein abstraktes Datenmodell zur Repräsentation von Geodaten, das ab der Version 5 dem ISO-Standard 19107 „Geographic Information - Spatial Schema“ entspricht (ISO 2001). Auf dieser Basis hat das OGC als eine Implementierung das „Simple Feature Model“ vorgeschlagen, das sich auf zweidimensionale Geoobjekte beschränkt und Punkte grundsätzlich geradlinig verbindet. Dieses Modell findet zum Beispiel in Geodatenbanken Anwendung (OGC 1999).

Für die XML-basierte Repräsentation von Geodaten hat das OGC die „Geography Markup Language“ (GML) (OGC 2001b) vorgeschlagen. Neben der Repräsentation von räumlichen Informationen und deren Beziehungen soll GML eine Grundlage für Internet-GIS bieten und eine Integration von raumbezogenen und nicht-räumlichen Daten unterstützen, insbesondere für den Fall, dass die nicht-räumlichen Daten auch XML-kodiert sind. Grundlage von GML ist das Simple Feature Model. Um Beziehungen zwischen Objekten und deren Geometrieattributen zu repräsentieren, gibt es innerhalb von GML eine Menge von vordefinierten Beziehungsattributen: `location`, `position`, `centerOf`, `centerLineOf`, `edgeOf`, `extentOf` und `coverage`. Das Beispiel in Abbildung 1 zeigt eine XML-Repräsentation einer Stadt, die u.a. zwei Geometrien besitzt.

```
<city>
   <name>Oldenburg</name>   <id>3403000</id>   <population>153531</population>
   <gml:location>
      <gml:Point> <gml:coordinates>8.2275,53.1375</gml:coordinates> </gml:Point>
   </gml:location>
   <gml:extentOf>
      <gml:Polygon> <gml:outerBoundaryIs>
            <gml:LinearRing>
               <gml:coordinates> 8.214289,53.087776, 8.296767,53.145781,
                 8.307031,53.153128, 8.299069,53.171417, 8.279517,53.191205,
                 8.24101,53.201628, 8.213507,53.204358, 8.158582,53.177491,
                 8.153453,53.165192, 8.143793,53.136619, 8.214289,53.087776
               </gml:coordinates>
            </gml:LinearRing>
      </gml:outerBoundaryIs>  </gml:Polygon>
   </gml:extentOf>
</city>
```

Abb. 1: Beispiel für ein GML-Dokument.

2.2 „Scalable Vector Graphics“ (SVG)

Um Vektordaten im Internet übermitteln und darstellen zu können, gibt es bislang noch keinen anerkannten Standard. Dies führt dazu, dass Geodaten-Server oftmals entweder die Daten vor einer Übertragung in Rasterbilder umwandeln oder dass proprietäre Datenformate eingesetzt werden. Wegen der beschränkten Übertragungsgeschwindigkeit ist gerade für Location-Based Services die Übertragung von Rastergrafiken keine adäquate Lösung. Ein Lösung aus diesem Dilemma kann die vom WWW-Konsortium (W3C) verabschiedete Empfehlung zum Grafikstandard „Scalable Vector Graphics“ (SVG) darstellen (W3C 2001). Dieses XML-basierte Datenformat besitzt eine Reihe von Eigenschaften, die die

Repräsentation und Visualisierung von kartografischen Informationen unterstützt (NEUMANN&WINTER 2000):

- SVG unterstützt Vektordaten und erlaubt die Einbettung von Rasterbildern.
- SVG erlaubt die Gruppierung von Objekten, also die Bildung von Layern.
- Lokale Koordinatensysteme können definiert werden.
- Eine Ereignisbehandlung kann über JavaScript eingebettet werden.

Als geometrische Primitive werden Rechtecke, Kreise, Ellipsen, Strecken, Streckenzüge und Polygone unterstützt. Das `path`-Element erlaubt die Definition komplexerer Geometrien wie z.B. Multi-Polygone mit Löchern. Ein Beispielsdokument und die Visualisierung dieses und eines anderen SVG-Dokuments wird in Abbildung 2 gezeigt.

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<?xml-stylesheet href="graphic.css" type="text/css"?>
<svg width="120" height="120">
  <g transform="scale(0.5)">
    <line x1="20" y1="20" x2="44" y2="54" style="stroke:red;"/>
    <rect x=60 y=10 width=24 height=34 style="stroke:rgb(0,0,0); fill:none;" />
    <g id="ct" transform="translate(30,40)" style="stroke:red; fill:rgb(0%,75%,0%);">
      <rect x="30" y="12" width="24" height="34" transform="translate(-30,-20)" />
      <circle cx="50" cy="52" r="24" style="visibility:normal;" />
    </g>
    <rect x="70" y="52" width="24" height="24" rx="9" ry="9" style="fill:yellow;"/>
    <text x="0" y="70" style="font-size:7;">TEXT 1</text>
    <line x1="30" y1="90" x2="100" y2="90" style="stroke:#0000ff;"/>
    <text x="30" y="90" style="font-family:Arial; text-decoration:underline;
          font-style:italic; font-weight:bold; text-anchor:start; font-size:16;">
       Hello </text>
    <ellipse cx="5" cy="55" rx="20" ry="40" class="green" />
    <polyline points="50,50 75,50 90,75 120,80 90,60" />
    <polygon points="15,5 30,1 35,15 15,10" transform="translate(30,0)"
             style="fill:aqua; stroke:maroon;" />
    <path d="M 5 5 L 20 5 l 0 15 h -15 Z M 10 10 L 10 15 L 15 15 L 15 10 Z"
          style="fill:pink; stroke:teal;" />
  </g>
</svg>
```

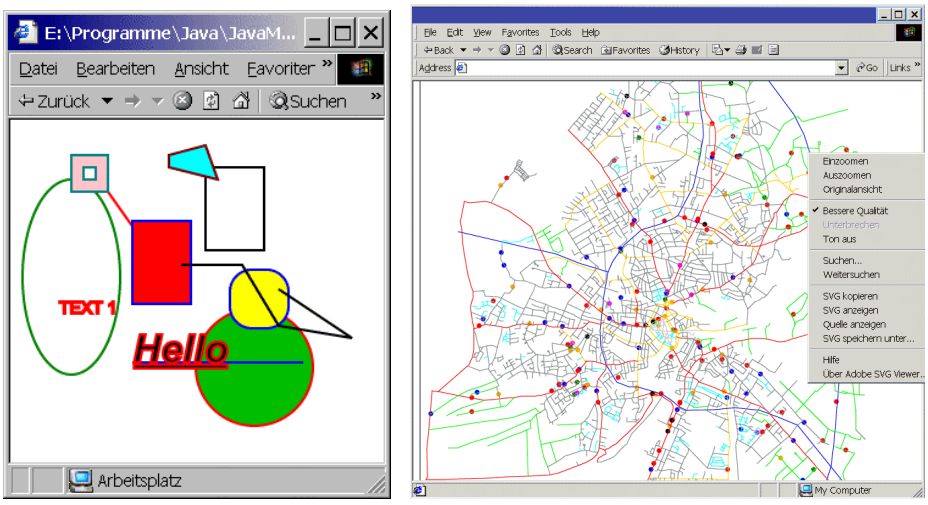

Abb. 2: Beispiele für ein SVG-Dokument und für visualisierte SVG-Dokumente.

Über entsprechende Viewer können SVG-Dokumente in normalen HTML-Seiten visualisiert werden. Für herkömmliche Internet-Anwendungen kann erwartet werden, dass SVG künftig ohne zusätzliche Plug-Ins unterstützt wird. Für LBS ist es aus Effizienzgründen nicht zweckmäßig, die volle Definition von SVG zu unterstützen. Dieser Thematik widmet sich die z.Zt. im Stadium der „Candidate Recommendation" befindliche Version 1.1 der SVG-Spezifikation (W3C 2002b). Dabei wurde der Beobachtung Rechnung getragen, dass die Endgeräte im Hinblick auf Rechenleistung, Speicher- und Übertragungskapazität sehr weit differieren. Ausgehend von einer Modularisierung von SVG wurden zwei „Profile" speziell für mobile Endgeräte definiert (W3C 2002c). SVG Tiny (SVGT) zielt auf sehr eingeschränkte Geräte wie Smartphones ab, SVG Basic (SVGB) richtet sich an leistungsfähigere, wie z.B. PDAs der neuesten Generation.

3 Verwaltung XML-basierter Daten in Datenbanksystemen

Nach (WATERFELD 2001) sprechen wir von einem XML-Datenbanksystem, wenn das Datenbanksystem XML als Datenmodell vollständig versteht. Das Schema wird also in XML definiert und das DBS ist in der Lage, beliebige XML-Instanzen zu speichern und mit einer XML-basierten Anfragesprache wieder zurückzuliefern. Dies impliziert die Nutzung heutiger Standards wie XML Schema und XPath, die für die Beschreibung von Schemata in Datenbanken und für die hierarchische Navigation in Knoten eines Dokumentes (Pfadausdrücke) entwickelt wurden.

3.1 Architekturen von XML-Datenbanksystemen

Vergleichbar mit der Einführung objektrelationaler und objektorientierter Datenbanksysteme (DBS) kann auch bei XML-Datenbanksystemen die Architektur in zwei Kategorien eingeteilt werden (WATERFELD 2001):

- Aufsatzlösungen auf bestehenden Datenbanksystemen und
- „native" XML-Datenbanksysteme.

Während bei der ersten Kategorie ein existierendes Datenbanksystem „on the top" für die Verwaltung von XML-Daten erweitert wird, handelt es sich bei der zweiten um ein vollständig neues Datenbanksystem zur Verwaltung von XML-Daten. Offensichtlich wird ein dritter theoretisch möglicher Ansatz, nämlich die Erweiterung eines existierenden relationalen oder objektorientierten Datenbanksystems „from the side", also auf verschiedenen Ebenen wie Anfragesprache, Zugriffsstrukturen etc., bisher nicht verfolgt. Die Mehrzahl der existierenden Datenbanksysteme haben Aufsatzlösungen auf relationalen oder objektorientierten DBS realisiert. Es gibt jedoch auch erste native XML-Datenbanksysteme.

Zur ersten Kategorie gehören beispielsweise SQL-Server 2000 und Oracle 8i. Der Vorteil besteht darin, dass bereits bewährte Datenbanktechnologie genutzt werden kann. Allerdings wird dies durch den damit einhergehenden Abbildungsverlust auf andere Datenmodelle erkauft, was insbesondere bei großen Datenmengen zu mangelnder Effizienz führen kann. Ein Vertreter der zweiten Kategorie von Datenbanksystemen ist Tamino DB. Dieses DBS speichert XML-Dokumente in einer für XML optimierten Form, ohne dabei andere Datenmodelle zu benutzen. Es stellt auch für die physikalische Datenspeicherung XML-Datenbankstrukturen bereit. Der XPath-Standard wurde so erweitert, dass er als Anfrage-

sprache auf Mengen von XML Instanzen genutzt werden kann. Mit der Verabschiedung einer XQuery Recommendation (W3C 2002a) – derzeit Working Draft Status – des W3C wird in zukünftigen Versionen auch eine standardisierte Anfragesprache für XML-Dokumente unterstützt werden. Erste prototypische Implementierungen basierend auf dieser Spezifikation existieren bereits.

3.2 Verwaltung von Geodaten: Stand der Technik

Nach Wissen der Autoren haben bisher weder die Vertreter der Aufsatzlösungen auf bestehenden Datenbanksystemen noch die der nativen Datenbanksysteme Lösungen zur Verwaltung von Geodaten in XML-Datenbanksystemen entwickelt. Prinzipiell bieten sich zwei verschiedene Lösungsmöglichkeiten für eine Realisierung an:

- die Erweiterung von XQuery um Anfragetypen für Geodaten und damit einer direkten Unterstützung durch das jeweilige Datenbanksystem oder
- eine Aufsatzlösung für Geodaten in der jeweiligen API des Datenbanksystems.

Die erste Möglichkeit ist die effizientere, da die Erweiterungen auf niedrigerer Datenbankebene eingebaut werden können. Außerdem kann so im Datenbanksystem ein „Baukasten“ für verschiedene Klassen von Geo-Anwendungen realisiert werden (z.B. 2D/3D-Anwendungen). Dies hat den Vorteil, dass die Klassen nicht bei jeder Anwendung neu implementiert werden müssen (Wiederverwendbarkeit des Codes). Der Nachteil ist jedoch, dass die Datenbankhersteller zusätzlich von der ständigen Weiterentwicklung des jeweiligen Geo-Standards abhängig sind und dieser aufgrund der vielseitigen Nutzergruppen in den Geowissenschaften in kleineren Zeitabständen geändert bzw. angepasst werden müsste. Die zweite Möglichkeit ist hier freier, da jeweils nur für die aktuelle Anwendungsklasse eine Anpassung der XML-Standards für Geodaten vorgenommen werden muss. Allerdings ist die Portierbarkeit auf andere Anwendungsklassen erschwert. Im folgenden geben wir ein Beispiel für die zweite Möglichkeit der Realisierung.

3.3 Eine Erweiterung der Tamino API zur Verwaltung von Geodaten

Eine einfache Erweiterungsmöglichkeit für native XML-Datenbanksysteme besteht in der Erweiterung ihrer jeweiligen APIs zum Zugriff auf das Datenbanksystem. Im vorliegenden Fall wurde die Java-API des Datenbanksystems Tamino erweitert. Die Erweiterung kann in zwei Bereiche unterteilt werden. Der erste Teilbereich beinhaltet eine von dem jeweiligen Datenbanksystem unabhängige Implementierung eines räumlichen Index auf der Basis eines R-Baumes mit einer einfachen Unterstützung von Transaktionen. Der zweite Bereich ist die Erweiterung der Tamino API um Interfaces und Klassen für die Verwaltung und den Zugriff auf raumbezogene XML-Daten.

Die innerhalb der Tamino API für den Zugriff auf XML-Daten zuständigen Interfaces und Klassen wurden in der Geo-Erweiterung in ihren Definitionen bis auf das Einfügen der Bezeichnung „Geo“ in die Interface-, Klassen- und Methodensignaturen zur Unterscheidung der beiden API, eins zu eins übernommen. Die Implementierung der Interfaces der Geo-Erweiterung erfolgte durch die Umhüllung der bestehenden Zugriffsklassen. Innerhalb der Klassen der Geo-Erweiterung erfolgt dadurch jeweils eine Vorverarbeitung der Daten und eine anschließende Weitergabe der Daten an die Implementierung der umhüllten Klasse der ursprünglichen Tamino API.

In der derzeitigen Realisierung werden alle XML-Dokumente, die ein definiertes Element „BoundingBox“ beinhalten in einen räumlichen Index aufgenommen. Weiterhin wurde die derzeitig in Tamino verwendete XML Query Language innerhalb der Geo-Erweiterung um die Anfragetypen `intersects`, `contains` und `nearestOf` für räumliche Anfragen erweitert. Die Speicherung von GML-Dokumenten ist derzeit in der Tamino DB noch nicht möglich, da nicht alle von GML benötigten XML Schema-Elemente unterstützt werden.

4 XML für Location-Based Services

4.1 Die Grundstruktur

Ausgangspunkt der weiteren Betrachtungen ist, GML als Basis-Kodierung von XML-basierten Geodaten zu verwenden. Allerdings muss man dabei berücksichtigen, dass GML nicht alle relevanten Aspekte einer Anwendung abdecken kann. So fehlen zum Beispiel Konzepte, um Beschriftungen geeignet in ein Dokument einzubetten (EVANS 2001). Auch bedarf es für die Repräsentation von bewegten Geoobjekten an Erweiterungen (ZIPF&KRÜGER 2001). Ein so ergänztes GML wird im weiteren als GML^{+} bezeichnet.

Auf der Client-Seite wird als Darstellungsformat SVG erwartet. Dieses muss allerdings zwei Prozessen unterworfen werden: Der erste Schritt ist eine Reduktion auf Elemente, die für die Repräsentation von GML^{+} erforderlich sind, um eine (effiziente) Unterstützung auf mobilen Endgeräten zu ermöglichen. Damit kann die Komplexität von SVG entscheidend verringert werden, ohne an notwendiger Ausdruckskraft zu verlieren. Der andere Prozess ist eine Erweiterung von SVG. So lassen sich einige kartografische Konzepte wie die Generalisierung mit SVG zur Zeit nur unzureichend bzw. nur sehr umständlich über eingebettete Skripte realisieren (NEUMANN&WINTER 2000). Auch sind Erweiterungen zur effizienten Unterstützung von bewegten Geoobjekten erforderlich. Das entsprechend eingeschränkte und erweiterte Format wird im folgenden als SVG$^{\pm}$ bezeichnet. Der dafür erforderliche Sprachumfang kann in der kommenden Version 1.1 von SVG als ein eigenständiges Profil in Anlehnung an SVG Tiny bzw. SVG Basic beschrieben werden.

Der Übergang von GML^{+} zu SVG$^{\pm}$ erfolgt auf der Server-Seite. Von besonderen Interesse für LBS ist naturgemäß die aktuelle Position des mobilen Endgeräts bzw. von anderen Endgeräten. Diese Positionen werden für zwei Aufgaben benötigt: 1. für die Kartendarstellung auf dem Endgerät und 2. für die XML-basierte Formulierung von Anfragen an einen entsprechenden Informationsdienst. Abbildung 3 gibt einen Überblick über die beschriebene Grundstruktur.

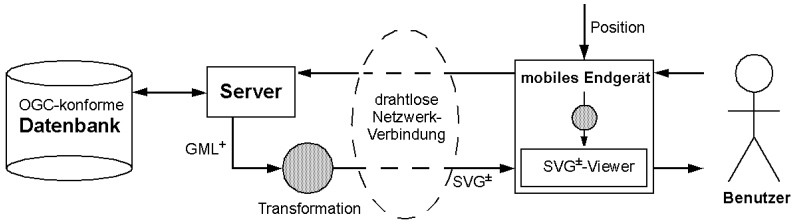

Abb. 3: Grundstruktur für die Verwendung von XML in LBS.

Die Berücksichtigung unterschiedlicher Clients erfolgt serverseitig durch geräteklassenspezifische Stylesheets (XSL-Transformationen) (W3C 1999), die die Transformation der GML-Daten in das erforderliche SVG-Profil oder auch nicht-grafische XML-Dialekte übernehmen. Einzelheiten zur Architektur werden in (BRINKHOFF&WEITKÄMPER 2001) beschrieben.

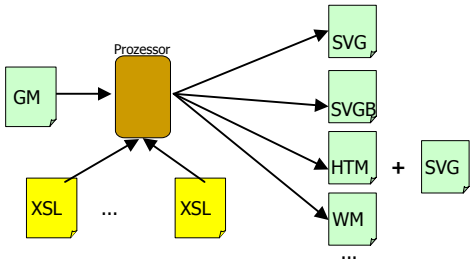

Abb. 4: Gerätespezifische XSL-Transformationen.

4.2 Implementierung von Viewern für mobile Endgeräte

Auf einem PC ist die Verarbeitung von XML-Dokumenten relativ unproblematisch. Es existieren die entsprechenden Programmbibliotheken und die verwendeten Rechner sind ausreichend rechenstark. Für mobile Endgeräte stellt sich die Situation anders dar: Zur Zeit finden 3 unterschiedliche Betriebssysteme (PocketPC auf Basis von Windows CE, PalmOS und Epos) Verwendung. Bei Mobiltelefonen ist die Lage noch gerätespezifischer. Um auf all diesen Geräten einen Viewer für SVG$^{\pm}$ zu entwickeln, kommt als Betriebssystem- und Hardware-unabhängige Programmiersprache insbesondere Java in Frage. Allerdings sind Java-Programme deutlich langsamer in ihrer Ausführung als vergleichbare C++-Anwendungen. Diese Schwäche ist für mobile Kleingeräte nicht unproblematisch.

Für PDAs hat Sun die Java-Variante „PersonalJava“ entworfen. Personal Java besitzt Java 1.1.8 als Basis (SUN 2000b). Für Mobiltelefone wurde von Sun das „Mobile Information Device Profile” (MIDP) für die „Java 2 Platform, Micro Edition (J2ME)“ definiert (SUN 2000a). Die Klassenbibliothek des MIDP ist eine (zu) sehr eingeschränkte Untermenge der Bibliothek von Java 2. So erlaubt diese Bibliothek weder die Darstellung gefüllter Polygone noch unterstützt sie Gleitkomma-Arithmetik.

Die Verarbeitung von XML-Dokumenten erfordert, diese zu parsen. Dafür existiert unter Java 2 die "Java API for XML Processing (JAXP)" (SUN 2001). Allerdings sind Implementierungen von JAXP viel zu komplex und auf den oben genannten Java-Plattformen nicht ablauffähig. Das Open-Source-Projekt kXML (ENHYDRA 2002) stellt hingegen einen geeigneten XML-Parser bereit, der sowohl unter PersonalJava als auch unter dem MIDP einsetzbar ist. Die übersetzten Klassen benötigen nur 37 KB.

Auf dieser Basis wurden sowohl unter PersonalJava als auch dem MIDP zwei Viewer für SVG$^{\pm}$ entwickelt, wobei der Satz der unterstützen SVG-Elemente die Darstellung aller transformierten GML-Elemente ermöglicht (BRINKHOFF 2002). Zu Vergleichszwecken wurde eine Implementierung unter C++ für PDAs mit dem Betriebssystem PocketPC entwickelt. Abbildung 5 zeigt die PersonalJava-Implementierung auf einem PocketPC-PDA und die MIDP-Implementierung auf einem Emulator.

Abb. 5: Die Implementierungen von Viewern für SVG$^{\pm}$.

5 Anwendung

Die in Abschnitt 3.3 vorgestellte Geo-Erweiterung von Tamino DB soll künftig für die Entwicklung eines XML-basierten Fahrrad-Routenplaners genutzt werden. Die Anwendung soll die Routenplanung auf verschiedenen Detailstufen (z.B. Fahrrad-Fernwegenetz – lokale Fahrradwege) durch geeignete Datenstrukturen unterstützen. Zur effizienten Speicherung der Routen bieten sich für die Wegesuche erweiterte und optimierte Ansätze hierarchischer Graphen an (HIMSOLT 1996; BUCHHOLZ& RIEDHOFER 1997). Im einzelnen sind für die Routenplanung die folgenden Typen von Anfragen auf Graphen zu unterstützen:

- Kürzeste-Wege-Anfragen und kontextspezifische Anfragen zur Wegesuche (z.B. unter Berücksichtigung der Anzahl der Sehenswürdigkeiten, geringen Steigung, Benutzerprofile).
- Navigation in Graphen (Vorgänger-, Nachfolgeknoten und –kanten etc.).
- Suche in hierarchischen Graphen (Unterscheidung verschiedener Detailebenen des Wegenetzes.
- Räumliche Selektion von Teilgraphen (contains und intersects).

Die vom Serversystem generierte Route, sowie das Streckennetz zur Übersicht und weitere touristische Informationen können durch XSL-Transformation (W3C 1999) direkt von den in der Datenbank gespeicherten XML-Dokumenten in SVG-Graphiken umgewandelt und übertragen werden.

Für eine endgültige Realisierung dieses Konzeptes auch auf mobilen Endgeräten sind jedoch noch weitere Entwicklungen im Bereich der Viewer, insbesondere bei der Interaktion mit den Graphikelementen, nötig. Hier sollte eine Auswahl von Start-, End- und Zwischenpunkten für die Anfrageformulierung einer Routenplanung interaktiv aus dem visualisierten SVG-Dokument möglich sein. Dies ist derzeit jedoch nur sehr eingeschränkt mittels JavaScript in HTTP-Browsern möglich.

6 Ausblick

In zukünftigen Arbeiten gilt es, die vorgestellten XML-basierten Techniken für Location-Based Services weiter zu vervollständigen und in die Gesamtarchitektur zu integrieren. Dabei sind insbesondere folgende Fragen zu untersuchen:

- die Optimierung des Zusammenspiels der verschiedenen Komponenten,
- Untersuchungen zur Effizienz,
- die Anfrage und Übertragung von bewegten Geoobjekten,
- die Entwicklung von standardisierten Bausteinen zur Implementierung von LBS,
- die Integration z.B. als Web-Service in die z. Zt. entstehenden Internet-Frameworks wie Microsoft .NET und SUN ONE.

Schließlich sind die Ergebnisse anhand einer konkreten Anwendung zu evaluieren.

Literatur

Brinkhoff T. (2002): *Java Support for XML-Represented Simple Features on Mobile Devices*. Technischer Bericht, Institut für Angewandte Photogrammetrie und Geoinformatik, Oldenburg, 2002.

Brinkhoff T. & Weitkämper J. (2001): *Continuous Update Queries within an Architecture for Querying XML-Represented Moving Objects*. In: Proceedings 7th International Symposium on Spatial and Temporal Databases (SSTD), Redondo Beach, CA, 2001, Lecture Notes in Computer Science, Vol. 2121, Springer, S. 136-154.

Buchholz F. & Riedhofer B. (1997): *Hierarchische Graphen zur kürzesten Wegesuche in planaren Graphen.* Techn. Report Nr. 1997/13, Institut für Informatik, Universität Stuttgart, 12 S.

Enhydra.org (2002): *kXML*, http://kxml.enhydra.org/index.html

Evans J.D. (2001): *Discussion Paper: XML for Image and Map Annotations (XIMA), Draft Candidate Interface Specification*, Juni 2001, http://www.opengis.org/techno/ discussions/01-019.pdf

Himsolt M. (1996): *Hierarchical Graphs for Graph Grammars.* In: Abstract Proc. 5th Int. Workshop on Graph Grammars and their Application to Computer Science, S. 67-70.

ISO (2001): *Draft International Standard ISO/DIS 19107 - Geographic Information - Spatial Schema*, 2001.

Neumann A. & Winter A. (2000): *Kartographie im Internet auf Vektorbasis, mit Hilfe von SVG nun möglich*. In: carto:net, Version 2.0, 09/2000, http://www.carto.net/papers/svg/index.html

OGC (1999): *OpenGIS Simple Feature Specification for SQL, Revision 1.1*, Mai 1999, http://www.opengis.org/techno/implementation.htm

OGC (2001a): *The OpenGIS Abstract Specification, Topic 1: Feature Geometry, Version 5*, Mai 2001, http://www.opengis.org/techno/abstract.htm

OGC (2001b): *Geography Markup Language (GML), Version 2.0, OpenGIS Implementation Specification*, Februar 2001, http://www.opengis.org/techno/implementation.htm

Sun Microsystems Inc. (2000a): *Mobile Information Device Profile (JSR-37), JCP Specification Java 2 Platform, Micro Edition, Version 1.0a*, Dez. 2000.

Sun Microsystems Inc. (2000b): *PersonalJava Application Environment Specification, Version 1.2a (Final)*, Nov. 2000.

Sun Microsystems Inc. (2001): *Java API for XML Processing, Version 1.1, Final Release*, Febr. 2001, ftp://ftp.java.sun.com/pub/jaxp/89h324hruh/jaxp-1_1-spec.pdf

Waterfeld W. (2001): *Realisierungsaspekte eines XML Datenbanksystems.* In: Proceedings 9. GI-Tagung Datenbanksysteme in Büro, Technik und Wissenschaft (BTW), Oldenburg, Informatik aktuell, Springer, S. 479-484.

W3C (1999): *XSL Transformations (XSLT) Version 1.0, W3C Recommendation*, November 1999, http://www.w3.org/TR/xslt

W3C (2000): *Document Object Model (DOM) Level 2 Core Specification, W3C Recommendation*, November 2000, http://www.w3.org/TR/DOM-Level-2-Core/

W3C (2001): *Scalable Vector Graphics (SVG) 1.0 Specification, W3C Recommendation*, September 2001, http://www.w3.org/TR/SVG/

W3C (2002a): *XQuery 1.0: An XML Query Language, W3C Working Draft*, April 2002, http://www.w3.org/TR/xquery/

W3C (2002b): *Scalable Vector Graphics (SVG) 1.1 Specification, W3C Candidate Recommendation*, April 2002, http://www.w3.org/TR/SVG11/

W3C (2002c): *Mobile SVG Profiles: SVG Tiny and SVG Basic, W3C Candidate Recommendation*, April 2002, http://www.w3.org/TR/SVGMobile/

Zipf A. & Krüger S. (2001).: *TGML - extending GML by Temporal Constructs - A Proposal for a Spatiotemporal Framework in XML*. In: Proceedings of the 9th ACM International Symposium on Advances in Geographic Information Systems, Atlanta, GA, USA, November 2001, S. 94-99.