

Navigation auf XML-Dokumenten

Morad Ahmad,
FB 17 Mathematik/Informatik
Universität Kassel
D-34109 Kassel
{morad}@db.informatik.uni-kassel.de

Abstract: Die weite Verbreitung von mobilen Geräten und die Entwicklung von mobilen Kommunikationstechnologien ermöglichen den Einsatz solcher Geräte für Aufgaben, die über die reine Kommunikation hinaus gehen. In dieser Arbeit wird ein XML-basiertes Gesamtmodell vorgestellt, indem durch eine einheitliche Schnittstelle zu gemeinsamen Informationsräumen einen Zugang zu Informationen für Geräte mit unterschiedlichen Kapazitäten und Darstellungsmöglichkeiten ermöglicht wird. Das Navigieren auf gemeinsamen Informationsräumen wird durch die Anzeige der Präsenz und die Kontrolle der Nebenläufigkeit zu einer speziellen Form der Gruppenarbeit erweitert.

1 Einleitung

Die rapide Entwicklung von Techniken der mobilen Kommunikation (UMTS, WLAN) ermöglicht den Einsatz mobiler Geräte für Koordinierungsaufgaben räumlich verteilter Benutzer. In dieser Arbeit betrachten wir Anwendungen für *Synchrone Gruppenarbeit auf gemeinsamen Datenräumen*. Wegen den enormen Unterschieden zwischen den verschiedenen mobilen Geräten, sind Standards für Datenrepräsentation und Interaktion notwendig. Eine enorme Hilfe zur Entwicklung solcher Standards bietet die Metabeschreibungssprache XML (*eXtensible Markup Language* [W3C00]).

In dieser Arbeit wird in einer Mittelschicht eine einheitliche Navigationssprache auf XML-Dokumente realisiert, mit derer Hilfe verschiedene Geräte einen einheitliche Zugriff auf XML-basierten Datenquellen haben. Zur Darstellung der Daten bei den Anwendungen werden generische Stylesheets verwendet, die die entsprechende Visualisierung für jedes Gerät individuell erzeugen.

2 Das Modell

Die Grundidee ist die Bereitstellung von Informationen in einer einfachen hierarchischen Struktur und die Definition einer intuitiven *Operationssemantik* auf diesen Daten. Durch eine navigierende Schnittstelle wird die Lokalisierung eines Benutzers, und damit die

Bereitstellung von Awareness-Informationen, ermöglicht; der Cursor, mit dem er Operationen ausführt gibt seinen Aktivitätsbereich an. Sein Sichtbarkeitsbereich definiert sich durch die sichtbaren Objekte in seinem Fenster. In diesem Modell lassen sich vier relativ disjunkte Teilmodelle herausarbeiten:

Interaktions- und Datenmodell: Versteht man synchrone Gruppenarbeit als das gemeinsame Arbeiten auf strukturierten Datenräumen, dann bietet sich ein Interaktionsmodell an, das Navigation besonders unterstützt. Dazu führen wir den Begriff des Cursors ein, der auf ein Datenelement zeigt, nennen ihn aber hier *Finger*, um die Verwechslung mit dem mausgesteuerten Cursor der graphischen Anzeige oder einer Texteingabe zu vermeiden [Weg91b]. Mit einem Finger kann ein hierarchisches Dokument (XML-Dokument, DOM-Baum etc.) mit generischen Navigationsoperationen traversiert werden. Es ergibt sich bei einer navigierenden Interaktion zwangsläufig ein Datenmodell, bei dem die Objekte baumartig (hierarchisch) angeordnet sind.

Visualisierungsmodell: Aufbauend auf Erfahrungen aus Vorläuferarbeiten [Tha99] bietet es sich zunächst an, eine generische Darstellung in Tabellenform anzubieten, die die Struktur eines Dokuments anschaulich visualisiert. In diese Tabellensicht lassen sich wiederum die Finger aus dem Interaktions-Submodell gut projizieren. Zu beachten ist dabei, daß diese Tabellen in realen Anwendungen sehr groß werden. Daher muß man bei der Visualisierung berücksichtigen, daß nur einen Ausschnitt aus diesem Datenraum gezeigt wird. Im folgenden wird dieser Ausschnitt *Viewport* genannt und entspricht im wesentlichen einem Fenster, das über die Tabelle gescrollt werden kann. Neben der „generischen“ Visualisierung als Tabelle sollen weitere Darstellungen möglich sein, wobei sogar vorstellbar ist, daß verschiedene Teilnehmer einer Gruppenanwendung verschiedene, ggf. an ihre Endgeräte angepaßte Darstellungen des gemeinsamen Datenraums haben.

Awareness-Modell: Die Anzeige der Präsenz (*Awareness*) ist ein sehr wichtiges Element der Gruppenarbeit, denn sie gibt die Gruppenmitglieder Informationen über die Aktivitäten der anderen und schafft damit die Möglichkeit der Kooperation [Bür99]. Die sichtbaren Objekte im Viewport eines Benutzers geben seinen Interessenbereich an. Der Bereich, in dem der Benutzer agiert, ist eindeutig durch die Position seines Fingers bestimmt. Ähnlich dem sog. Fokus/Nimbus-Modell [BBF⁺94], werden aus diesen Daten Informationen darüber abgeleitet, die Auskunft darüber geben, welche Benutzer einander sehen, und welche die eigenen Aktivitäten beobachten. Awareness-Informationen werden nur an Benutzer gesendet, die den Finger des ausführenden Benutzers sehen [Ahm03].

Kontrolle der Nebenläufigkeit: Synchrone Gruppenarbeit auf strukturierten Datenräumen ist eine Form der Kollaboration, die sich für Verhandlungen, Koordinierungsaufgaben usw. besonders eignet [QSJ02]. Dabei werden von den Beteiligten der Gruppe Entscheidungen auf der Basis angezeigter Daten getroffen, die man als Transaktionen im üblichen Sprachgebrauch bezeichnen könnte. Transaktionen werden ausgehandelt, wobei bei offenen Verhandlungen Vorschläge, Forderungen, Angebote und Alternativen für alle frei einsehbar auf den Tisch kommen sollten. Gleichzeitig erwarten die Gruppenmitglieder, daß Festlegungen und Zusagen dokumentiert und eingehalten, bzw. umgesetzt werden. Die Parallelen zur Persistenz (*durability*) und zur Atomizität (*atomicity*) aus der DB-Transaktionsdefinition drängen sich deshalb auf. Wir verwenden den Begriff „visuelle

Transaktion“ für die Zusammenfassung von Aktionen eines an der Gruppenarbeit beteiligten Mitglieds. Der entscheidende Unterschied zu DB-Transaktionen ist die Aufgabe der Isolation zugunsten einer Signalisierung von Nebenläufigkeit (*concurrency awareness*) [GS87, BS95]. Im Gegensatz zu Datenbanksystemen sollen Teilnehmer bei der Gruppenarbeit sich mittels Awareness gegenseitig koordinieren. Unerwünscht sind nur die Auswirkungen des parallelen Betriebs, die aufgrund fehlender Informationen, etwa wegen veralteter Daten, versteckter Änderungen, inkonsistenter (Phantom-)Anzeigen usw. den einzelnen zu nicht revidierbaren Aktionen veranlassen, die er so nicht ausgeführt hätte, wenn er den aktuellen (korrekten) Zustand des Systems gekannt hätte.

3 Systemarchitektur

Das Modell kann in drei Hauptschichten gemäß Abbildung 1 aufgeteilt werden:

- User front end/Client
- Groupware-Server
- Daten- und Interaktion Server

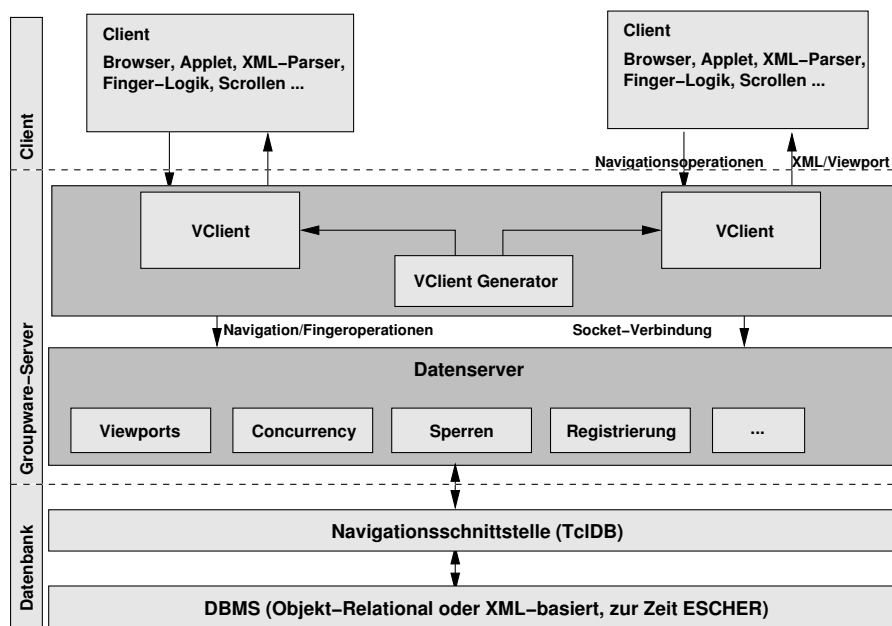


Abbildung 1: Systemarchitektur

3.1 Datenmodell

Der Informationsraum besteht aus sog. Applikationen. Jede Applikation kann mehrere XML-Dokumente und Schemata enthalten. Die Beschreibung der Struktur eines Dokuments wird durch ein XML Schema ausgedrückt. Zusätzlich werden bei jedem Element zwei weitere Attribute eingebaut [WA01]. Das Attribut `id` enthält eine eindeutige Identifizierung eines Elements. Das Attribut `nf2type` gibt die hierarchische Struktur des Dokuments wieder. Danach kann jedes Element mengenwertig (`sett` und `listt`), Tupel (`ptuplet` und `tupel`) oder atomar (`textt`, `chart`, `intt`, `floatt` und `imaget`) sein. Programm 3.1 zeigt ein XML-Dokument mit Daten über ein Projekt. Jedes Tupel der äußeren Menge `TASKS` enthält eine `TASKID`, kurze Beschreibung (`DESCRIPTION`) etc. und eine Menge der Tasks, von denen dieser Task abhängt (`REQUIRES`).

Programm 3.1 *Projects.xml* Daten über ein Software-Projekt:

```
<?xml version='1.0' encoding='' ISO-8859-1'' ?>
<TASKS nf2type='' sett'' >
  <TASK nf2type='' ptuplet'' >
    <TASKID nf2type='' textt'' > START </TASKID>
    <DESCRIPTION nf2type='' textt'' > project kickoff </DESCRIPTION>
    <DUR nf2type='' intt'' > 0 </DUR>
    <ES nf2type='' intt'' > 0 </ES>
    <LS nf2type='' intt'' > 0 </LS>
    <EF nf2type='' intt'' > 0 </EF>
    <LF nf2type='' intt'' > 0 </LF>
    <ISOTIME nf2type='' textt'' > 01-01-1998 </ISOTIME>
    <REQUIRES nf2type='' sett'' state='' empty'' > </REQUIRES>
  </TASK>
  <TASK nf2type='' ptuplet'' >
    ...
  </TASK>
  ...
</TASKS>
```

Zur Navigation auf ein Dokument dient ein Finger, mit dem Navigationsoperationen (`push`, `pop`, `next` und `back`) zur Traversierung des Baumes ausgeführt werden. Relativ zur Position eines Fingers können Lese- und Schreiboperationen, wie `get`, `is`, `insert` oder `delete` ausgeführt werden.

Dieser Ansatz ist aus dem Datenbankeditor `ESCHER` [Weg91a] auf XML-Dokumente übertragen. `ESCHER` wird immer noch für die interne Repräsentation der Daten eingesetzt, wobei der Übergang zur XML-Darstellung in der Mittelschicht liegt. Als Schnittstelle zu `ESCHER` dient die Skriptsprache `TclDB` [Ahm98].

3.2 Der Groupware-Server

Der Groupware-Server besteht aus einer Mittelschicht, die Aufgaben der Mehrbenutzerinteraktion und der Awareness-Signalisierung wahrnimmt. Wie in Abbildung 1 gezeigt, gibt es innerhalb des Groupware-Servers eine Trennung in den virtuellen Client (VClient) und den Datenserver.

Die weiteren Aufgaben, wie in Abbildung 1 angedeutet, können (von oben nach unten) wie folgt beschrieben werden: Die Benutzeroberfläche meldet sich bei dem zentralen Server (VClient-Generator) an. Dieser erzeugt eine Instanz des VClients, die einerseits eine (lokale) Verbindung zum Datenserver, andererseits einen Datenkanal zum in der Regel entfernten Client aufbaut. Diese Schicht bekommt von der Anwendung generische Navigations-, und Schreiboperationen, generiert daraus TclDB-Skripte und schickt sie an den Datenserver. Aus den Ergebnissen dieser Operationen generiert sie Visualisierungen, die als XML-Dokumente an den Client weitergeleitet werden. Der Datenserver schickt zusätzlich Awareness-Nachrichten an den VClient in Form von Funktionsaufrufen. Der VClient generiert daraus Visualisierungsnachrichten, und leitet sie an den Client weiter.

Für die Unterstützung von verschiedenen Gerätetypen mit unterschiedlichen Kapazitäten ist auf der Serverseite für jede Anwendung ein Client-Simulator (VClient) zuständig [JHE99, JK97]. Der VClient ist der serverseitige Verbindungsstück zum Client. Für jede laufende Applikation wird eine VClient-Instanz erzeugt. Sie dient als Übersetzer zwischen einem höheren Sprachsatz, der Schnittstelle zur Client-Applikation, und den technisch aufwendigen Operationen des Datenservers. Vom Client kommen kontextabhängige Schreib- und Navigationsoperationen auf XML-Dokumenten, die durch ein Protokoll festgelegt sind. Diese werden in TclDB-Skripte übersetzt und an den Datenserver weitergeleitet. Mit einem festgelegten Satz von Datenbankoperationen und Awareness-Abfragen kommuniziert der VClient mit dem Datenserver. Auf der anderen Seite schickt der Datenserver Visualisierungsnachrichten, die ebenso aus einem festgelegten Satz von Visualisierungsoperationen bestehen, die der VClient umsetzt. Diese reflektieren die Aktivitäten der anderen Benutzer.

Für die Entwicklung von Client-Applikationen ist eine Java-Schnittstelle implementiert. Eine Applikation bekommt Eingaben vom Benutzer und ruft entsprechende Methoden des VClients auf, um die Aktionen des Benutzers auszuführen. Auf der anderen Seite bekommt ein Client Visualisierungsnachrichten vom VClient in Form von standardisierten XML-Dokumenten. Für die Anzeige solcher Dokumente können Clients generische Stylesheets verwenden, die eine Standardvisualisierung produzieren. Die Interpretation dieser Dokumente wird von Klassen der Clientschnittstelle, oder vom VClient bei leistungsschwachen Geräten, durchgeführt. In letztem Fall werden HTML oder SVG verschickt.

Die Visualisierung wird anhand von XSLT-Stylesheets erzeugt. Dazu existieren zwei Arten von Stylesheets. *Generische Stylesheets* erzeugen eine Visualisierung aus der Struktur des Dokuments. D.h. sie können bei jedem Dokument verwendet werden. Dafür werden lediglich die Attributangaben in `nf2type` verwendet. Z. B. erzeugt das generische Stylesheet in Programm 3.2 eine geschachtelte HTML-Tabelle aus dem Dokument in Programm 3.1 (Abbildung 2).

Programm 3.2 *Generisches Stylesheet. Produziert eine geschachtelte HTML-Tabelle:*

```
<?xml version='1.0'?>
<xsl:stylesheet version='1.0'
  xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
<xsl:template match='/'>
  <html>
    <table align='center' border='0'>
      <xsl:apply-templates/>
    </table>
  </html>
</xsl:template>
<!-- Mengen und Listen -->
<xsl:template match='*[@nf2type='sett' or @nf2type='listt' or
  @nf2type='msett']'>
  <xsl:choose>
    <xsl:when test='@state='null''>
      <td align='center'>
        <xsl:choose>
          <xsl:when test='@nf2type='listt''>
            <xsl:value-of select = '$null-listt' />
          </xsl:when>
          <xsl:otherwise>
            <xsl:value-of select = '$null-sett' />
          </xsl:otherwise>
        </xsl:choose>
      </td>
    </xsl:when>
    <xsl:when test='@state='empty''>
      <td align='center'>
        ...
      </td>
    </xsl:when>
    <xsl:otherwise>
      <td align='left' valign='top'>
        <table align='center' border='0'>
          <xsl:apply-templates/>
        </table>
      </td>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
...
</xsl:stylesheet>
```

Spezielle Stylesheets dagegen werden für ein bestimmtes Schema entwickelt. Sie erzeugen eine spezielle Darstellung basierend auf dem Inhalt eines Dokuments. Abbildung 3 zeigt ein GANT-Diagramm in SVG, das durch ein spezielles Stylesheet erzeugt wurde.

Stunde	Projektphase	1	2	3	4	5	6	7	8	9	10	11	12
1	ANAL	1	1	1	1	1	1	1	1	1	1	1	1
2	ANAL	1	1	1	1	1	1	1	1	1	1	1	1
3	ANAL	1	1	1	1	1	1	1	1	1	1	1	1
4	ANAL	1	1	1	1	1	1	1	1	1	1	1	1
5	ANAL	1	1	1	1	1	1	1	1	1	1	1	1
6	ANAL	1	1	1	1	1	1	1	1	1	1	1	1
7	ANAL	1	1	1	1	1	1	1	1	1	1	1	1
8	ANAL	1	1	1	1	1	1	1	1	1	1	1	1

Abbildung 2: Darstellung als geschachtelte HTML-Tabelle

Objekte im Viewport eines Benutzers werden mit einer *Visuellen Sperre (V-Sperre)*, versehen, wodurch der Fokus eines Benutzers bestimmt wird. Die Gewinnung von Informationen über Fokus und Nimbus erweist sich dadurch als extrem einfach (siehe [Ahm03] für Details). Der Datenserver besitzt Funktionen, die Abfragen der Art „*Welche Benutzer sehen meinen Fokus (Nimbus) ?*“ oder „*Wieviele Benutzer sind in meinem Fokus aktiv ?*“ beantworten. Awareness-Informationen werden vom Datenserver zusammengestellt und bei Bedarf für die Benachrichtigung an den VClient in Form von Funktionsaufrufen geschickt. Dazu wird ein Protokoll festgelegt, das der VClient versteht. Der VClient trägt mit seinen Funktionen die Veränderungen in einer virtuellen Visualisierung ein und leitet diese Veränderungen in Form von XML-Dokumenten an den Client weiter. Die Client-Schnittstelle stellt Klassen für die Bearbeitung dieser XML-Dokumente und die Aktualisierung der Anzeige zur Verfügung.

In einer solchen kooperativen Umgebung können zwei Möglichkeiten der Koordination benutzt werden. Bei der ersten werden Benutzer Fingeroperationen sequentiell ausführen. Wir bezeichnen solche Operationen als *visuelle Operationen*, da sie aus der Visualisierung entstehen. In diesem Fall können Fehler eintreten falls, Operationen aus einer nicht aktuellen Visualisierung ausgeführt werden.

Zur Lösung solcher Konflikte wird ein Zeitstempleverfahren eingesetzt, daß eine Operation mit einer Auslösungszeit kleiner als die letzte Visualisierungsnachricht an dem ausführenden Benutzer zurückweist. Dazu übermittelt der Server bei jeder Visualisierungsnachricht einen Zeitstempel an dem Client. Diese wird vom Client an dem Server mit jeder Operation zurückgeliefert, woraus der Server die letztgesehene Nachricht bei der Auslösung einer Operation ermitteln kann.

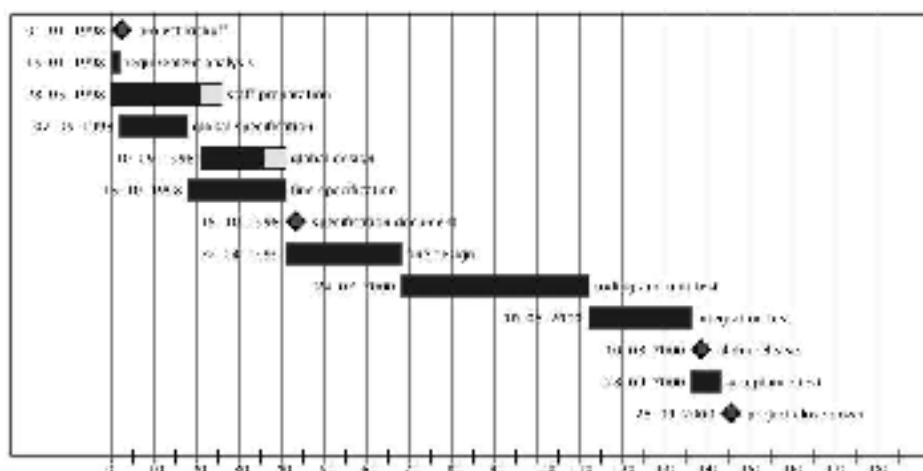


Abbildung 3: Darstellung als GANT-Diagramm

Bei der zweiten Möglichkeit werden mehrere Operationen vom Benutzer oder Anwendungsentwickler zu einer Transaktion zusammengefaßt. Solche Transaktionen (*visuelle Transaktionen*) unterscheiden sich von den sog. ACID-Transaktionen dadurch, daß sie *ohne Isolation* ablaufen. Sie werden vom Benutzer ausgeführt, der die Auswirkungen anderer Transaktionen beobachtet. Durch Awareness beeinflussen sich Benutzer also gegenseitig, so daß Entscheidungen zur Ausführung von Operationen innerhalb einer Transaktion vom Ablauf anderer Transaktionen abhängig ist.

Zur Lösung von Konflikten bei visuellen Transaktionen wird eine niedrigere Konsistenzstufe angestrebt (sog. Konsistenzstufe 2, siehe [GR93]). Danach werden write/write-Konflikte verhindert, read/write-Konflikte dagegen zugelassen. D.h. die Visualisierung von Daten, die von einer visuellen Transaktion bearbeitet werden, bevor diese beendet wird (*dirty read*), wird zugelassen. Dafür wird eine neue Sperre eingeführt (W-Sperre), die Leseoperationen zum Zweck der Visualisierung zuläßt, parallele Leseoperationen von anderen Transaktionen dagegen verhindert. Um diese Unterscheidung machen zu können wird eine neue Grundoperation (*readV*) eingeführt, die als „Lesen zum Visualisieren“ aufgefaßt wird, und dadurch von anderen Leseoperationen unterschieden wird.

4 Zusammenfassung und Weiterentwicklung

Der Groupware-Server wurde mit Tcl 8.4 mit den Erweiterungen Itcl und TclX auf Linux Maschinen implementiert. Der Zugriff auf die Datenbank wurde mit der Tcl-Schnittstelle zu ESCHER, TclDB [Weg91a, Ahm98], realisiert. Als XML-Parser diente das Paket tDOM 0.75. Die Client-Schnittstelle wurde mit JDK 1.3 implementiert. Das Paket Xerces diente

als XML-Parser, Xalan als Stylesheet-Prozessor und die „Open Source SVG Toolkit“ zur Darstellung von SVG-Dokumenten. Abbildung 4 zeigt eine Anwendung auf einem iPAQ 3970. Dabei wurde die Java Virtuelle Maschine von Jeode [Inc03] eingesetzt.



Abbildung 4: Benutzeranwendung auf dem iPAQ 3970 (Tabellendarstellung)

Es wurde ein Modell für synchrone Gruppenarbeit auf gemeinsamen Informationsräumen vorgestellt, das für den Einsatz von mobile Geräte sowie Arbeitsplatzrechner geeignet ist. Die aktuelle Implementierung kann unter <http://www.db.informatik.uni-kassel.de/~morad/eclient> beobachtet werden.

In der späteren Forschung wird der Einsatz von nativen XML-Datenbanken und die Übertragung des Fingermechanismus auf XML untersucht. Momentan wird der Einsatz von Xindice, mit XUpdate und XDNL, sowie der Einsatz von SOAP und XML-RPC erforscht. Eine Java-Anwendung wurde sowohl auf einem Desktop PC, als auch auf einem Pocket PC entwickelt. Es fehlten allerdings die Zeit und die personellen Möglichkeiten, ausgereifte spezielle Testanwendungen zu entwickeln oder den Einsatz bei Mobiltelefonen zu untersuchen. Eine Portierung auf Mobiltelefone mit J2ME ist ebenfalls Gegenstand unserer heutigen Forschung.

Literatur

- [Ahm98] Morad Ahmad. TcIDB: Entwurf und Implementierung einer Skriptsprache für den Datenbank-Editor ESCHER. Master's thesis, GhK-FB17, 1998.
- [Ahm03] Morad Ahmad. *Ein XML-basiertes Modell für synchrone Gruppenarbeit auf gemeinsamen Informationsräumen*. Dissertation, Universität Kassel, 2003. Eingereicht am 30.04.2003.

- [BBF⁺94] S. Benford, J. Bowers, L. E. Fahlen, J. Mariani, and T. Rodden. Supporting Cooperative Work in Virtual Environments. *The Computer Journal*, 37(8), 1994.
- [BS95] U. M. Borghoff and J. H. Schlichter. *Rechnergestützte Gruppenarbeit, Eine Einführung in verteilte Anwendungen*. Springer-Verlag, Berlin, Heidelberg, New York, 1995.
- [Bür99] Martin Bürger. *Unterstützung von Awareness bei der Gruppenarbeit mit gemeinsamen Arbeitsbereichen*. Herbert Utz Verlag, München, 1999.
- [GR93] Jim Gray and Andreas Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann Publishers, Inc., 1993.
- [GS87] Irene Greif and Sunil Sarin. Data Sharing in Group Work. *ACM Transactions on Office Information Systems*, 5(2):187–211, April 1987. Special Issue on Computer-Supported Cooperative Work.
- [Inc03] Insignia Solutions Inc. Insignia Jeode documentation. URL: <http://www.-insignia.com/content/products/jeodeRuntime.shtml>, 2003.
- [JHE99] Jin Jing, Abdelsalam Sumi Helal, and Ahmed Elmagarmid. Client-server computing in mobile environments. *ACM Computing Surveys (CSUR)*, 31(2):117–157, 1999.
- [JK97] Anthony D. Joseph and M. Frans Kaashoek. Building reliable mobile-aware applications using the Rover toolkit. *Wireless Networks*, 3(5):405–419, 1997.
- [QSJ02] Christoph Quix, Mareike Schoop, and Manfred Jeusfeld. Business data management for business-to-business electronic commerce. *ACM SIGMOD Record*, 31(1):49–54, 2002.
- [Tha99] Jens Thamm. *Visualisierungsverfahren zur Interaktion mit objekt-relationalen Datenbanken*. Dissertation, Universität Gh Kassel, August 1999.
- [W3C00] W3C. Extensible Markup Language (XML) 1.0 (Second Edition). URL: <http://www.w3.org/TR/2000/REC-xml-20001006>, October 2000.
- [WA01] Lutz Wegner and Morad Ahmad. Orthogonality in DBMS Design: the XML Approach. In Bob Werner, editor, *Proceedings of the International Workshop on Foundations of Models for Information Integration (FMII-2001)*, pages 66–78, Viterbo, Italy, Sept, 16-18 2001.
- [Weg91a] L. Wegner. Let the Fingers Do the Walking: Object Manipulation in an NF² Database Editor. In H. Maurer, editor, *Proceedings of the Symposium on New Results and New Trends in Computer Science, Graz, June 20–21, 1991*, number 555 in Lecture Notes in Computer Science, pages 337–358. Springer, 1991.
- [Weg91b] L. M. Wegner. Let the Fingers Do the Walking: Object Manipulation in an NF² Database Editor. In Hermann Maurer, editor, *Proceedings of New Results and New Trends in Computer Science*, volume 555 of *LNCS*, pages 337–358, Berlin, Germany, June 1991. Springer.