Architektur und Realisierung eines verteilten XML-basierten Informationssystems

Sergej Melnik

melnik@informatik.uni-leipzig.de
Institut für Informatik, Universität Leipzig

September, 1998

Abstract

Die rasante Entwicklung von XML (eXtensible Markup Language) als neue Web-Sprache eröffnet weitreichende Möglichkeiten zur flexiblen Realisierung von verteilten Informationssystemen. Dieser Beitrag beschreibt den grundlegenden Aufbau und die Erfahrungen, die bei der Entwicklung eines universitären Informationssystems auf der Basis von XML gesammelt wurden. Durch eine zur Zeit noch kaum in der Praxis eingesetzte Technologie-Kombination konnten viele Schwierigkeiten heutiger Informationssysteme mit geringem Aufwand umgangen werden.

1 Einleitung

WWW-basierte Informationssysteme müssen zahlreiche Anforderungen erfüllen. Sie sollen skalierbar und erweiterbar sein, eine präzise inhaltliche Suche ermöglichen, eine einheitliche Schnittstelle bieten und auf offenen Standards basieren. Die WWW-orientierten Client-Server Anwendungen, Java-Technologie und verteilte objekt-orientierte Middleware-Plattformen konnten vielen dieser Kriterien mit Erfolg standhalten [Uma97].

Dabei wurde der Strukturierung, Archivierung und dem Austausch von Informationen und Wissen häufig ungenügend Rechnung getragen. Der neue Standard des World Wide Web Konsortiums — eXtensible Markup Language (XML) — soll dem eine Abhilfe schaffen und ist zur Zeit von vielen visionären Vorstellungen und Erwartungen umgeben [KR97, Bos97, Mic98, PR98]. Allerdings liegen noch kaum Erfahrungen mit der Realisierung XML-basierter Informationssysteme vor, die ihre konkreten Vorteile und Beschränkungen aufzeigen. In diesem Beitrag stellen wir Entwurf und Realisierung eines universitären Informationssystems vor. Der Einsatz von XML wurde dadurch motiviert, daß die bisherige Umsetzung [Mel97] die Anforderungen an das System nur unzureichend abdecken konnte, insbesondere im Hinblick auf eine große Anzahl von Informationsproduzenten und eine flexible Anpassbarkeit auf die Bedürfnisse jeweiliger Einrichtungen.

Das System ist auf die Unterstützung des Lehr- und Forschungsbetriebs der Universität Leipzig ausgerichtet und ermöglicht die Erfassung und präzise Suche über Veranstaltungen, Publikationen, Studiengänge, Mitarbeiter u.ä. der Universität. Wegen einer beträchtlichen potentiellen Benutzeranzahl (über 5000 Mitarbeiter, ca. 25000 Studierenden) und sehr inhomogenen zu repräsentierenden universitären Strukturen wären traditionelle Datenbanktechniken nur mit unvertretbarem Aufwand einsetzbar (allein die Benutzerverwaltung wäre mit einem extremen Administrationsaufwand verbunden). Deshalb haben wir uns entschlossen, diese Anwendung auf einer datenbankgestützten Java-basierten XML-Plattform aufzubauen.

Unser Ansatz zeichnet sich durch mehrere Charakteristiken aus, die die gewünschte Funktionalität des Systems im wesentlichen bestimmen:

1. Verteilte Daten werden gesammelt und für eine globale Recherche bereitgestellt.

Dieses Grundprinzip wird von den heutigen Suchmaschinen im WWW extensiv genutzt. Es wurde beispielsweise im objekt-relationalen Informationssystem der australischen Universitäten eingesetzt [Eng98] und dient als Basis für viele anderen interessanten Anwendungen [PR98]. Neu sind allerdings folgende Aspekte: durch Einsatz von XML läßt sich eine sehr detaillierte inhaltliche Strukturierung der Informationen sowie deren flexible Verknüpfung untereinander erreichen. Die Ergebnisse der Suchanfragen werden in XML geliefert und können als Grundlage für weitere Verarbeitung dienen bzw. von anderen Dokumenten per Transklusion[1] eingebunden werden. Die Suchdienste können domänen- und funktionsspezifisch mehrfach vorhanden sein und entweder Indexierung (Abspeicherung relevanter Suchattribute und Verweise auf ursprüngliche Daten) oder Archivierung (lokale Spiegelung komplexer Datenobjekte) einsetzen.

2. Generierung von Mehrwert-Informationen.

Neben der Recherche ist die Erstellung von Mehrwert-Informationen ein wichtiger Schwerpunkt der Arbeit. Als Mehrwert-Informationen bezeichnen wir selbstenthaltene oder transklusionsbasierte XML-Dokumente, die aufgabenspezifisch aus dem globalen Datenbestand bei Bedarf automatisch generiert werden. Beispiele dafür sind Forschungs- und Jahresberichte sowie Vorlesungsverzeichnisse. Die Generierung kann unter Zuhilfenahme der oben genannten Suchdienste erfolgen, wobei die per Transklusion eingebundenen Informationsobjekte inhärenterweise immer auf dem aktuellsten Stand präsentiert werden.

3. Client-basierte Datenpräsentation und -verarbeitung.

Strukturierte Daten werden von einem mit mobiler Logik ausgerüsteten Client aus verschiedenen Quellen zusammengefügt und benutzersensitiv präsentiert. Der Benutzerkontext kann auf dem Client-Rechner abgespeichert werden. Die Ergebnisse der Client-basierten Datenverarbeitung können ebenfalls für spätere Nutzung lokal abgelegt werden. Web-orientierte Anwendungen, die dieses Prinzip im genannten Umfang einsetzen sind uns nicht bekannt.

Der nächste Abschnitt gibt einen kurzen Überblick über XML und unsere Sicht auf seine Bedeutung für verteilte Informationssysteme. Danach gehen wir unmittelbar auf das Design und Implementierung des Systems ein, insbesondere darauf, wie die drei oben beschriebenen Prinzipien umgesetzt werden.

2 XML im Überblick

Das World Wide Web wurde durch die Idee erschaffen, strukturierte Dokumente (HTML) mit Hilfe von leicht zu merkenden Hyperlinks (URLs) miteinander zu verknüpfen und durch ein einfaches Protokoll (HTTP) auszutauschen. Die Struktur der Dokumente beschränkte sich dabei auf die visuelle Auszeichnung der darin enthaltenen Informationen. Zwar menschenlesbar, entziehen sich jedoch die HTML-Dokumente einer automatischen inhaltlichen Verarbeitung, deren Notwendigkeit mit wachsendem Daten- und Kommunikationsumfang sehr akut geworden ist [All97]. Der spartanische Wortschatz des HTML reichte nicht mehr aus. Eine mächtigere universale Web-Spra-

1. Als *Transklusion* bezeichnet man ein transparentes Einbinden von Teilen anderer Dokumente in das Zieldokument. Diese erscheinen so, als ob sie im Zieldokument physisch enthalten wären (Zitat durch Verweis ohne Kopieren). Dieser Begriff wurde von Ted Nelson geprägt.

che wurde benötigt, um Informationen beliebiger Art semantisch aufzubereiten. XML entstand als eine offene Spezifikation des W3C [W3C], die auf SGML (ISO 8879) aufbaut, *dem* Industrie-Standard für strukturierte Archivierung und Austausch von Daten.

Die elementaren Bausteine eines XML-Dokumentes sind hierarchisch angeordnete XML-Elemente, die aus Attribut-Wert–Paaren, Text- oder Binärdaten bestehen. Die Etikettierung von XML-Elementen kann nicht nur durch einen Titel bzw. Namen erfolgen, sondern kann auch eine Namensraumangabe beinhalten, mit deren Hilfe über die Zugehörigkeit des Elementes zu einer bestimmten Domäne bzw. Ontologie entschieden werden kann. XML-Elemente können zu informationellen Einheiten [Kuh91] zusammengefügt werden, die wir als *Informationsobjekte* bezeichnen wollen. Die Informationsobjekte können sich aus Elementen verschiedener Domänen zusammensetzen, wie z.B. Medizin, CAD usw.

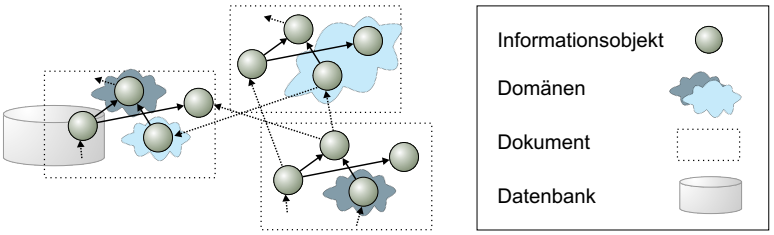

Abb. 1: Informationsobjekte, verteilt über die Grenzen von Dokumenten, Datenbanken und Domänen hinaus

Das Verweis-Konzept von XML (XML Linking Language, XLL) sieht eine für ein vollwertiges Hypertext-System kritische Eigenschaft [Eng90] vor, die in HTML nicht vorhanden war: nämlich die Möglichkeit, jedes Element innerhalb eines Dokumentes zu adressieren, unabhängig davon, ob dies vom Autor vorgesehen wurde oder nicht. In HTML sind Verweise in das Innere eines Dokumentes nur bedingt möglich, wenn der Autor dafür an der entsprechenden Stelle einen sog. *Anchor* eingefügt hat. Die Dokumentengrenzen werden in XML immer transparenter (siehe Abbildung 1). Zum einen ist es für die Anwendung häufig irrelevant, ob ein Informationsobjekt tatsächlich im Dokument enthalten ist, oder nur durch einen Verweis referenziert wird. Zum anderen werden zunehmend mehr Dokumente erst bei der Anfrage aus Datenbanken generiert. Aus diesen Gründen kann man vom Dokumentenbegriff abstrahieren und von einem Netzwerk aus verteilten, durch Hyperlinks verknüpften Informationsobjekten sprechen. Wird in XML ein wissensbasiertes Datenmodell repräsentiert, so läßt sich dieses auch als ein verteiltes semantisches Netzwerk auffassen.

In Abbildung 2 haben wir die Eigenschaften von XML und die Hoffnungen, die man mit dessen Einsatz verbindet, in komprimierter Form zusammengefaßt. Viele von diesen Vorteilen sind allerdings erst durch die Verwendung von mobilen Software-Modulen (z.B. Java, JavaScript [Net98]) ermöglicht worden. Aus der Sicht des Software-Engineering kann die neue Web-Sprache mit den zugehörigen APIs (XML DOM [W3C]) als eine *Middleware-Komponente für die Datenintegration* aufgefaßt werden.

3 Architektur des Systems

Eine wichtige Funktion einer Universität ist die Produktion von *Wissen*. Dieses Wissen schlägt sich in wissenschaftlichen Arbeiten der Universitätsangehörigen sowie im laufenden Vorlesungsbetrieb nieder. Es effizient aufzubereiten und einem breiten Kreis von Forschern, Studenten und der Industrie zur Verfügung zu stellen ist eine erfolgskritische Aufgabe der Universität. Jedoch

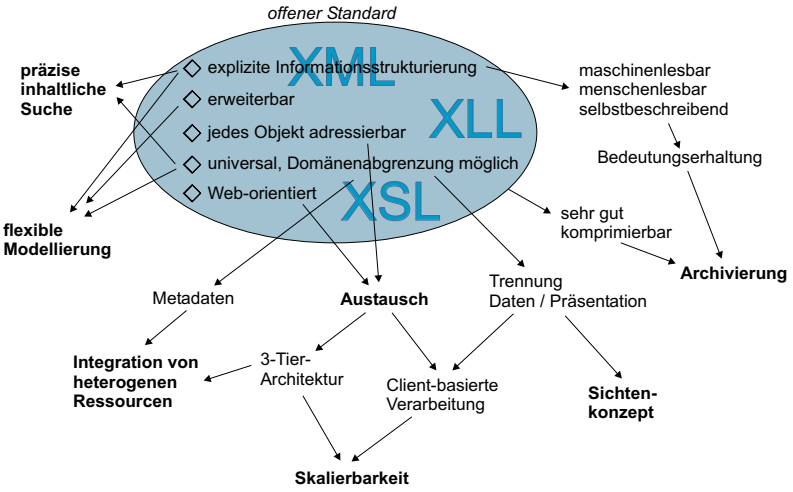

Abb. 2: Das Versprechen von XML

erweist sie sich als eine komplexe Fragestellung, da sehr viele Personen und Einrichtungen an der Wissensproduktion beteiligt sind.

Unter anderem wurden an das System folgende Anforderungen gestellt:

- Erfassung der Informationen über Mitarbeiter, Veranstaltungen, Studiengänge, Publikationen, Einrichtungen usw. Diese sollen bei Bedarf auch *mehrsprachig* aufbereitet werden können.
- Globale inhaltsbasierte Suche über alle relevanten Informationsobjekte.
- Möglichkeit der automatischen Erstellung von Mehrwertinformationen, wie Jahres- und Forschungsberichten, Vorlesungsverzeichnissen usw.
- Anhand des Veranstaltungsangebotes sollen Studenten in der Lage sein, ihre Stundenpläne per Mausklick fakultätsübergreifend zu erstellen und zu verwalten.

Wir haben die Entscheidung getroffen, einen zentralisierten Ansatz [Mel97] aufzugeben und ein vollständig dezentrales System zu entwickeln, dem die Prinzipien der hypermedialen Datenorganisation zugrunde liegen [Eng90]. Sowohl die Daten als auch die Anwendungslogik werden zwischen einzelnen Benutzern und Rechnern verteilt. Ein bedeutsames Argument dafür war die Notwendigkeit der Editierbarkeit der Daten und der damit verbundene Administrationsaufwand, der für die Verwaltung von Zugriffsrechten anfallen würde. Ein weiteres Kriterium war die Komplexität des Datenmodells, das sich von vornherein nicht lückenlos beschreiben ließ. Eine wesentliche Rolle spielte auch die benötigte Leistungsfähigkeit der Web-Anbindung, die bei zahlreichen potentiellen Benutzern mit Schreibrechten und einer Reihe externer Benutzer mit Leserechten schnell zum Flaschenhals hätte werden können.

Die grundlegende Informationsarchitektur des Systems wird in Abbildung 3 dargestellt. Diese baut auf vergleichbaren Prinzipien auf, die dem WWW zum Aufschwung verholfen haben: die Nutzer besitzen ihre Informationen im wahrsten Sinne des Wortes, es gibt kein zentrales Datenverwaltungssystem, von dem sie abhängen; der Veröffentlichungsvorgang besteht darin, daß sie die aufbereiteten Informationen dort unterbringen, wo darauf über eine URL zugegriffen werden kann.

Der Weg der Daten von Informationsproduzenten (Mitarbeitern der Universität) zu deren Konsumenten setzt sich aus Datenaufbereitung (Abbildung 4), Sammlung von verteilten Informationsob-

jekten (Abbildung 5) und deren nutzersensitiven Präsentation und Verarbeitung zusammen (Abbildung 6). Diese drei Schritte werden wir nachfolgend diskutieren.

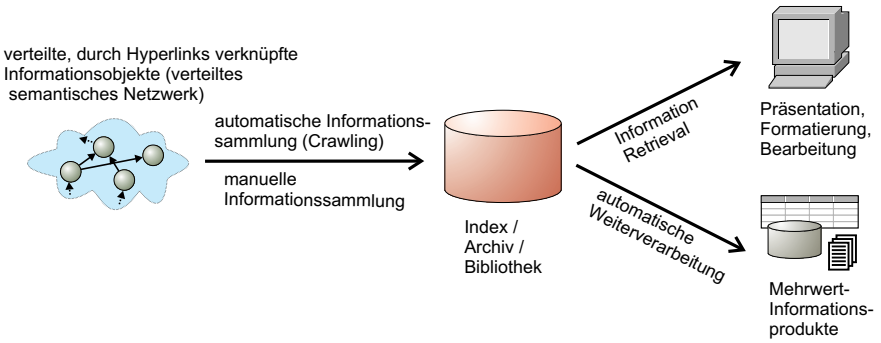

Abb. 3: Informationsarchitektur des Systems

Informationsaufbereitung

Jeder potentielle Benutzer des Systems ist in der Lage, ohne Voranmeldung am Informationsangebot teilzunehmen. Bei einer klassischen Architektur, in der die Daten in einer Datenbank i.e.S. verwaltet werden, erfordert jeder neue Nutzer die Einrichtung eines neuen Benutzerprofils. In unserem Szenario hingegen (siehe Abbildung 4) werden die Daten im Dateisystem des jeweiligen Informationsanbieters in Form von XML-Dateien dezentralisiert erstellt und gehalten. Dabei brauchen die Vorteile eines DBMS nicht notwendigerweise aufgegeben zu werden. Durch eine anschließende Datenaggregation läßt sich eine globale Sicht auf die Daten wiederherstellen.

Die mobile Anwendungslogik, die in Form eines XML-fähigen Java-Applets auf den Web-Client des Benutzers geladen wird, ermöglicht die Erstellung und Speicherung von semantisch strukturierten Daten im lokalen Dateisystem. Die erforderliche Interaktion mit dem Anwendungs-Server beschränkt sich auf einen Zugriff für den gesamten Editiervorgang. Die lokal entstandenen Daten (ein Teilnetz von Informationsobjekten) werden durch einen Verweis von einer glaubwürdigen Instanz auf ein Einstiegsobjekt dieses Teilnetzes bekanntgemacht und authentifiziert. Beispielsweise, wenn eine Mitarbeiterin einer universitären Einrichtung am Informationsangebot teilnehmen möchte, ist nur eine einmalige Eintragung des Verweises auf ihre aufbereiteten Daten in die Liste der Mitarbeiter dieser Einrichtung notwendig. Somit werden ihre Daten bekanntgemacht und über die zugehörige Einrichtung authentifiziert. Jede anschließende Änderung der Daten ihrerseits (z.B. Änderung der Sprechzeiten oder Verschiebung des Ortes einer Veranstaltung) wird in einem gewissen Zeitabstand global sichtbar.

Entsprechend der XML-Spezifikation läßt sich zu jedem XML-Element angeben, in welcher Sprache (Deutsch, Englisch usw.) sein Inhalt aufbereitet wurde. Auf diese Weise können die international relevanten Informationen wie z.B. Vorlesungsbeschreibungen, Forschungsschwerpunke u.ä. sprachsensitiv publiziert werden.

Die bei Bedarf ladbaren Software-Komponenten erlauben eine kontinuierliche Erweiterung der Funktionalität der Anwendung und dienen als Grundlage einer einfach wartbaren und skalierbaren Software-Architektur [Net98]. Dabei ist die Erweiterbarkeit des XML-basierten Datenmodells von entscheidender Bedeutung: bei der Datenmodellierung ist es oft der Fall, daß bestimmte Zusammenhänge nicht von vornherein berücksichtigt werden oder nicht bekannt sind. Eine dezentralisierte Datenarchitektur setzt deshalb voraus, daß eine Objektinstanz ein eigenständiges Leben führen kann. Dies bedeutet, daß eine Änderung dieser Objektinstanz weder die Objektklasse noch

alle anderen bestehenden Objektinstanzen betreffen darf. Die klassischen Datenmodelle (relationale, objektorientierte Datenbanken) sind in Hinblick darauf erheblich weniger flexibel, da diese auf einem festverdrahteten Datenmodell aufbauen. Der Einsatz von XML und mobiler Komponenten vervollständigen einander, indem sowohl die Anwendungslogik als auch das Datenmodell mit wechselnden Anforderungen an die Anwendung evolvieren können.

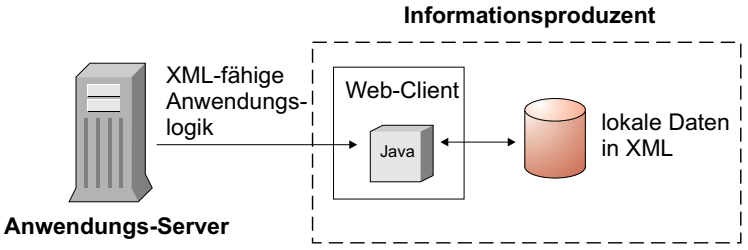

Abb. 4: Aufbereitung lokaler Informationen

Datenaggregation und -sammlung

Die verteilt aufbereiteten Informationen werden in periodischen Abständen von einem Crawler durchsucht und gesammelt (siehe Abbildung 5), ähnlich zu der Vorgehensweise eines Web-Roboters. Jedoch mit dem wesentlichen Unterschied, daß der Crawler das Datenvolumen nicht blind abarbeitet, sondern anhand eines feinkörnigen Datenmodells in der Lage ist, die benötigten Informationen domäne- und funktionsspezifisch durchzusieben. Die universitäre Domäne kann weiterhin nach Funktionen aufgegliedert werden, wobei jeder Crawler einen spezifischen Auftrag erfüllen kann, wie z.B. Sammlung der Adressen aller Mitarbeiter der Universität. Auf die von einem Crawler aggregierten Daten kann anschließend über eine Anfrageschnittstelle zugegriffen werden. Die gewünschte Anfragemächtigkeit und -syntax bestimmen das Suchprofil des jeweiligen Aggregations-Servers. Soll beispielsweise eine auf Forschungsschwerpunkten basierende Suche nach Mitarbeitern der Universität möglich sein, so soll diese Information mitgesammelt werden.

Je nach Bedarf können bei der Aggregation der Informationen drei verschiedene Strategien verfolgt werden (in der Reihenfolge aufsteigender Komplexität):

1. *Indexierung*: die für ein Suchprofil relevanten Attribute der Informationsobjekte werden samt dem Verweis auf das Originalobjekt in einem RDBMS gespeichert. Das ist das einfachste und schnellste Verfahren, das eine globale Recherche über die gesamte Domäne ermöglicht.
2. *Spiegelung*: zu jedem Informationsobjekt wird ein Duplikat erstellt. Diese Methode garantiert eine Verfügbarkeit der letztaktuellen Version des Objektes und lindert den *broken-link*–Effekt bei anschließenden Suchanfragen. Da die Aggregation domänen- und funktionsspezifisch gestaltet werden kann, hält sich der zu spiegelnde verteilte Datenbestand in Grenzen. Die gesammelten XML-Daten können entweder in einem universalen DBMS, einer Volltext-Datenbank oder in einem Dokumenten-Management-System abgelegt werden.
3. *Archivierung* in einer digitalen Bibliothek stellt die komplizierteste, aber auch die vielversprechendste Variante dar [Les97, CGM98]. Die gefundenen Informationsobjekte werden wie in der vorangegangenen Methode gespiegelt. Allerdings mit dem Unterschied, daß nicht nur die letzte Version des Objektes gespeichert wird, sondern eine Versionsverwaltung betrieben wird. Dabei wird jedes Objekt mit bereits vorhandenen Versionen verglichen (zu Algorithmen siehe [CGM97]). Beim Feststellen einer Änderung wird eine neue Version des Objektes sowie die

Querverweise zwischen der aktuellen und der vorangegangenen Version angelegt. Dieses Verfahren macht die Informationsobjekte über lange Zeit referenzierbar. Auch wenn sich die aktuelle Version mehrmals geändert hat, läßt sich der gewünschte Stand zurückverfolgen.

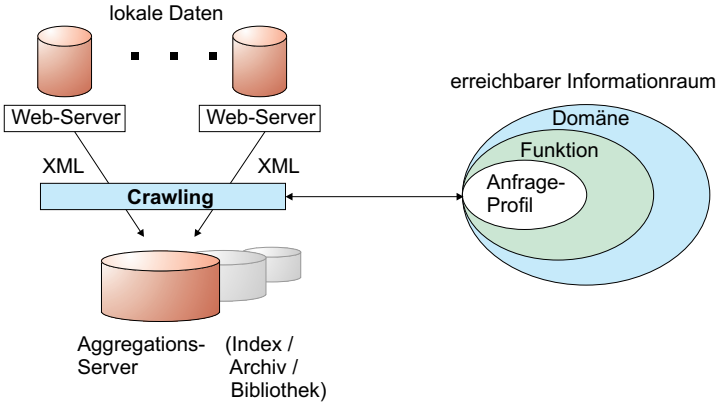

Abb. 5: Aggregation der verteilten Informationen

Der Aggregationsvorgang umfaßt nicht nur die Sammlung und evtl. Speicherung der gefundenen Objekte, sondern auch solche Aufgaben wie die Überprüfung deren Authentizität und Konsistenz, die im Falle einer datenbank-basierten Datenverwaltung normalerweise durch das DBMS übernommen werden. Diese Aufgaben können zum Teil bereits bei der Datenaufbereitung von der mobilen Anwendungskomponente erledigt werden. Beispiel: wird von einer Mitarbeiterin eine Publikation referenziert, die sie verfaßt haben soll, so kann dies durch die Überprüfung der in der Publikation enthaltenen Verweisen auf die Autoren bestätigt bzw. in Frage gestellt werden.

Präsentation und Verarbeitung

Die Schritte der Aufbereitung und Aggregation liefern eine Datenkonstellation, bei der die Originalversionen von Informationsobjekten über verschiedene Web-Server und Plattformen verteilt sind (siehe Abbildung 6). Über eine Anfrageschnittstelle können die Kopien der Informationsobjekte bzw. Verweise auf ihre Originalabbilder geliefert werden, wobei die Anfrage in einer URL kodiert werden kann. Es sei noch einmal betont, daß alle Daten in XML ausgetauscht werden, was ihre automatische Weiterverarbeitung ermöglicht. Mit Hilfe eines XML-fähigen Java-Moduls können nun verschiedene Benutzer im vorliegenden Informationsraum navigieren.

Die vorliegende Architektur weist eine Reihe von Vorteilen auf. Die Ergebnisse der Suchanfragen lassen sich per Transklusion (siehe Fußnote auf Seite 2) in weitere Dokumente einbinden. Damit wird eine Suchanfrage auf eine transparente Weise im Navigationsmodus ausgelöst. Die auf eine explizit gestellte oder implizit ausgelöste Suchanfrage gelieferten Informationsobjekte (bzw. Verweise darauf) können vom Benutzer im Navigationsmodus ins Detail erforscht werden. Das Wiederverwendungspotential der Daten vermindert Redundanz und Inkonsistenzen und fördert eine aufschlußreiche Vernetzung von Informationsobjekten. Wie aus der Abbildung hervorgeht, lassen sich 100% der Präsentation der XML-Daten auf dem Client-Rechner bewerkstelligen. Dies entlastet die Aggregations-Server und gewährleistet eine hohe Skalierbarkeit, wobei auf den Anwendungs-Server nur ein Zugriff pro Sitzung benötigt wird.

Client-basierte Verarbeitung erweist sich ebenfalls als vorteilhaft. Die benutzerspezifischen Einstellungen wie Sprache, Startseite usw. lassen sich lokal abspeichern und bei der nächsten Sitzung neu laden. Damit war es auch machbar, allen Studierenden die Möglichkeit zu geben, ihre Stun-

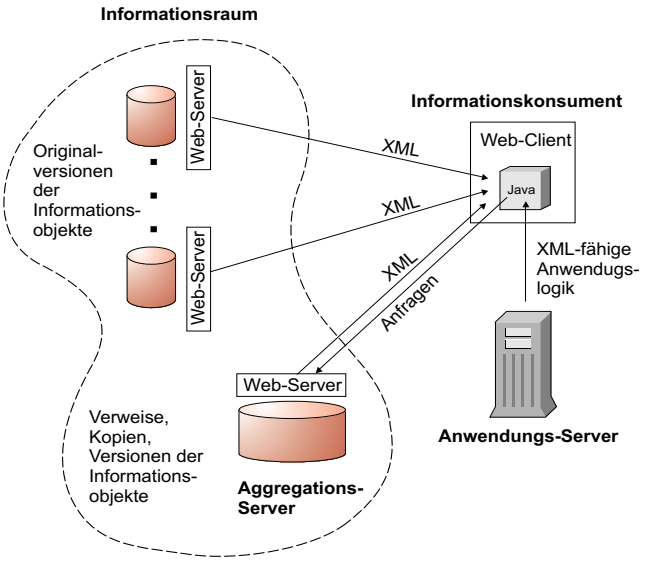

Abb. 6: Präsentation und Verarbeitung der Informationen

denpläne über alle Studienjahre fakultätsübergreifend zusammenzustellen und zu verwalten. Diese Art der Interaktion würde bei einer serverseitigen Datenverarbeitung eine ungeheure Last erzeugen.

Mehrwert-Informationen

Wie bereits erwähnt, ist die automatische Generierung von Mehrwert-Informationen neben der inhaltsbasierten globalen Recherche eines der wesentlichen Ziele des Systems. Die Erstellung von umfangreichen Dokumenten, an denen viele Autoren beteiligt sind wie z.B. ein kommentiertes Vorlesungsverzeichnis ist eine anspruchsvolle administrative Aufgabe, die viel Koordinationsaufwand erfordert [Mel97]. Auf der Grundlage eines genügend detaillierten Datenmodells und eines breiten Datenbestands lassen sich nicht nur komplexe relationale Zusammenhänge feststellen (mittels Data Mining und KDD [CHY96]), sondern auch gut strukturierte Dokumente erzeugen, an die man sich vorher gar nicht herangewagt hätte, wie z.B. ausgewählte bereichsübergreifende Verzeichnisse.

Die Erzeugung von maßgeschneiderten Dokumenten benötigt einen mit der Erstellung der Präsentationslogik vergleichbaren Programmieraufwand, der jedoch bei häufig wiederkehrenden Aufgaben durchaus gerechtfertigt wird. Dabei kann auf die von den Aggregations-Servern zusammengefaßten Sichten durch URL-kodierte Anfragen zurückgegriffen werden. So dient beispielsweise eine Anfrage, die das Lehrveranstaltungsangebot einer Einrichtung (Vorlesungen, Übungen, Seminare usw.) für das gegebene Semester auflistet, als Grundlage für die Erstellung eines kommentierten Vorlesungsverzeichnisses. Danach werden für jede gefundene Veranstaltung ihre strukturierte Beschreibung, Literaturhinweise, Teilnehmerkreis, Ort, Zeiten usw. ausgabegerecht formatiert, wobei die Verweise auf die beteiligten Personen und Einrichtungen automatisch verfolgt werden.

4 Implementierungsaspekte

Der derzeitige Stand der Realisierung des als ISLE (Informationssystem der Universität Leipzig) bezeichneten Systems umfaßt u.a. (a) einen Aggregations-Server, der alle relevanten Informationsobjekte indexiert, grundlegende transitive Beziehungen auflöst (Publikationen der Mitarbeiter einer Einrichtung können z.B. als Veröffentlichungen der Einrichtung angesehen werden) und eine fakultäts- und semesterübergreifende Suche erlaubt, (b) eine Navigationsstruktur, die alle Facetten des Datenmodells berücksichtigt, (c) client-basierte Stundenplanerstellung und (d) automatische Generierung der kommentierten Vorlesungsverzeichnisse. Das System befindet sich in der Test-Phase und ist zur Zeit unter http://lipsia.informatik.uni-leipzig.de/isle/ verfügbar.

Ein Hauptproblem ist die derzeit noch fehlende native Unterstützung von XML von den gängigen Browsern. eXtensible Style Language (XSL [W3C]) zusammen mit ECMA-Script als Scripting-Sprache (ECMA-262, ein Standard für Java-Script) sollen die Erstellung von Präsentationssoftware erheblich vereinfachen. Eine hauseigene Java-Script–basierte Implementierung der XML-Layout-Formatierung hat jedoch eine Reihe von Performanz- und Sicherheitsproblemen aufgeworfen. Deshalb haben wir eine Realisierung in Java vorgezogen. Eine knappe Hälfte des ca. 250 KB großen Moduls, das zum Client übertragen wird, macht ein validierender XML-Parser aus. Eine vergleichsweise kompakte Präsentationskomponente konnte dadurch erstellt werden, daß die Ausgabeformatierung und Benutzerinteraktion über HTML erfolgt und nicht durch den Einsatz von grafischen Elementen. Die angeklickten Verweise aktivieren das im Hintergrund laufende Java-Modul, welches verteilte XML-Daten über HTTP herholt und für die Ausgabe kontextsensitiv aufbereitet.

Einige Zeit haben wir die Umsetzung des XML Document Object Model (DOM [W3C]) mit Hilfe von CORBA intensiv verfolgt. Ein XML-fähiger CORBA-Server erlaubt es, XML-Dokumente gleichzeitig von mehreren Benutzern zu bearbeiten. Es zeigte sich jedoch, daß das XML DOM nur mit bedingtem Erfolg für eine verteilte Objekt-Plattform eingesetzt werden kann, weil wegen einer hohen Granularität der Objekte ein beträchtlicher Netzverkehr zustande kommt, der die Leistungsfähigkeit erheblich beeinträchtigt. Umfangreiche, von mehreren Benutzern editierbare Ressourcen können in unserem Architekturmodell durch eine Variante des Check-out/Check-in–Konzeptes nachgebildet werden, auf die wir hier nicht näher eingehen.

Ein weiteres Hindernis stellt die mangelhafte Standardisierung des Sicherheitsmodells für Java-Applets, die auf verschiedene Netzressourcen und lokale Dateisysteme zugreifen sollen. Aufgrund dessen ist das System zur Zeit ausschließlich mit Netscape Communicator nutzbar, das ein flexibles aber leider proprietäres Sicherheitskonzept zur Verfügung stellt.

5 Fazit

XML verspricht, viele der angesprochenen Bedürfnisse heutiger Systeme auf eine elegante Art zu lösen. Obwohl die Werkzeug-Grundlage noch nicht ausgereift ist, zeigten uns die Erfahrungen, die wir bei der Entwicklung eines Informationssystems für die Universität Leipzig gesammelt haben, daß man bereits heute durch den Einsatz dieser Technologie signifikante Vorteile erzielen kann.

Zu nennen sind hierbei insbesondere folgende Vorzüge, welche unser Ansatz durch die Verwendung der oben beschriebenen Architektur gewonnen hat: Dank client-basierter Verarbeitung, unbegrenzter Speicherungskapazität verteilter Rechner und mehrfach vorhandenen Aggregations-Servern mit abgrenzbaren Aufgaben ist die Benutzeranzahl beliebig skalierbar. Durch semantische Auszeichnung der Informationen ist eine präzise Recherche möglich. Wiederverwendung der

Daten wird durch Maschinenlesbarkeit und offene Standards erleichtert. Selbstverwaltung und Transklusion der Informationsobjekte fördern Redundanzvermeidung, Aktualität und Korrektheit der Daten. Daneben zählen einheitliche Präsentation, Mehrsprachigkeit sowie die Möglichkeit der inkrementellen Erweiterung des Datenmodells und der Anwendungslogik zu den weiteren positiven Merkmalen des Systems.

Abgesehen von den Schwierigkeiten, die mit der zur Zeit fehlenden Browserunterstützung von XML verbunden sind, stellen Authentifizierung und Konsistenzerhaltung der verteilten Informationen weitere Herausforderungen an das System dar.

6 Ausblick

In [PR98] wird die Vermutung geäußert, daß die ersten zwei (von drei) Phasen des Übergangs von XML zum dominierenden Datenaustauschformat für das WWW wie folgt ablaufen werden:

1. Strukturierte Daten und Suchergebnisse werden in XML zur Weiterverarbeitung geliefert.
2. Das Anfragevermögen von Web-Ressourcen wird in XML beschrieben, wodurch deren automatische Erschließung möglich wird.

Die erste Phase ist in unserem System vollständig realisiert. Die zweite ist für die von uns in Angriff genommene Domäne (Universität) noch von geringerer Bedeutung, da weltweit wenig Daten für globale Recherchen verfügbar sind. Anders sieht es im Bereich von elektronischen Bibliotheken aus. Der Dokumenten-Server der Universität Leipzig[1], der ebenfalls von uns entwikkelt wird, exportiert seine gesamten Metadaten in XML und erleichtert damit seine Integration in umfassendere Suchdienste.

Eine weitere aus unserer Sicht interessante Entwicklung besteht in der Aggregation und Archivierung der feinkörnigen Informationsobjekten nach den Prinzipien einer elektronischen Bibliothek (Daten werden nie gelöscht, sondern durch neuere Versionen ergänzt). Der Einsatz von XML erlaubt es, neben der physischen Archivierung von Daten die bedeutungserhaltenden Aspekte verstärkt anzugehen.

Danksagung

Zum Dank verpflichtet bin ich Herrn Prof. Dr. E. Rahm für seine kontinuierliche Unterstützung und Motivation beim Entwurf des Systems. Ich möchte ebenfalls Herrn Dr. D. Sosna und Herrn Dr. Ch. Wolff für ihre wertvollen Hinweise und Anregungen danken.

Literatur

All97 Charles Allen. *WIDL: Automating the Web with XML*, in World Wide Web Journal, Volume 2, Number 4, 1997.

Bos97 Jon Bosak. *XML, Java, and the Future of the Web*, 1997. http://sunsite.unc.edu/pub/sun-info/standards/xml/why/xmlapps.htm

1. Verfügbar unter http://dol.uni-leipzig.de

BBP97 V. Balasubramanian, A. Bashian, D. Porcher. *A Large-Scale Hypermedia Application using Document Management and Web Technologies*. Proc. of the 8th ACM Conf. on Hypertext, Southhampton, UK. ACM Press 1997.

CGM97 Sudarshan S. Chawathe, Hector Garcia-Molina. *Meaningful Change Detection in Structured Data*, SIGMOD 1997.

CGM98 Arturo Crespo, Hector Garcia-Molina. *Archival Storage for Digital Libraries*. 3rd ACM Conf. on Digital Libraries, 1998.

CHY96 M.-S. Chen, J. Han and P. S. Yu. Data Mining: An Overview from Database Perspective, IEEE TKDE 8 (6), 1996

Eng90 Douglas C. Engelbart. *Knowledge-Domain Interoperability and an Open Hyperdocument System*, Proc. of the Conf. on CSCW, Los Angeles, 1990. http://www.bootstrap.org/augment-132082.htm

Eng98 Carlos F. Enguix. *Database querying on the WWW: UniGuide, an object-relational search engine for Australian universities*. 7th WWW Conf., 1998

FLM98 D. Florescu, A. Levy and A. Mendelzon. *Database Techniques for the World-Wide Web: A Survey.* ACM SIGMOD Records, Sep. 1998

KR97 Rohit Khare and Adam Rifkin. *Capturing the State of Distributed Systems with XML*, World Wide Web Journal. Special Issue on XML, Volume 2, Number 4, 1997.

Kuh91 Reiner Kuhlen. *Hypertext: ein nicht-lineares Medium zwischen Buch und Wissensbank*. Springer, 1991

Les97 Michael Lesk. *Practical Digital Libraries: Books, Bytes and Bucks*. Morgan Kaufmann Publishers, 1997.

Mau93 Frank Maurer. *Hypermediabasiertes Knowledge Engineering für verteilte wissensbasierte Systeme*. Infix, 1993

Mel97 Sergej Melnik. EPOS: Electronic Publishing mit OODBS und SGML. LDV-Forum, Bd. 14, Nr. 1, S. 17-32, 1997.

Mic98 Microsoft Corp. *XML: Enabling Next-Generation Web Applications*, White Paper, 1998. http://www.microsoft.com/xml/articles/xmlwp2.htm

Net98 Netscape Corp. *Crossware: Cross-Everything Applications*, 1998. http://www.netscape.com

PR98 STS Prasad and Anand Rajarman. *Virtual Database Technology, XML, and the Evolution of the Web.* Bulletin on Data Engineering, IEEE, Vol. 21, No. 2, 1998.

Uma97 A. Umar. *Application (Re)Engineering: Building Web-Based Applications and Dealing With Legacies*. Prentice Hall, 1997.

W3C World Wide Web Consortium. XML, XSL, XLL und XML DOM Spezifikationen. http://www.w3.org