
Projektdokumentation

Projektgruppe NT 2: XML/DXL

# Evaluation des Lotus XML-Toolkits

Office Systeme I

Professor Dr. Ludwig Nastansky

*Wintersemester 2002/2003*

*vorgelegt von:*

Alexander Lindhorst

Wirtschaftsinformatik

Markus Steckenborn

Wirtschaftsinformatik

# Inhaltsverzeichnis

# Abbildungsverzeichnis



# Tabellenverzeichnis

# 1. Einleitung

Die vorliegende Arbeit dokumentiert die im Rahmen des Projekts „NT 2: XML/DXL" aus der Veranstaltung „Office Systeme 1" bis zum Ende des Wintersemesters 2002/2003 erarbeiteten Ergebnisse. Ziel des Projektes ist die Erstellung eines Prototypen für die Nutzung des Lotus Notes Gruppenkalenders auf mobilen Endgeräten im Zeitraum von zwei Semestern. Ausgangspunkt für das Projekt ist die Evaluation des Lotus XML Toolkits, die im ersten Teil des Projekts (zweite Hälfte des Wintersemesters 2002/2003) stattfand.
Die vorliegende Arbeit beschäftigt sich schwerpunktmäßig mit eben dieser Evaluation; die Erstellung des Prototypen sowie die Erarbeitung des genauen Vorgehens auf dem Weg dahin sind Arbeitspakete des zweiten Semesters und bilden somit keinen Schwerpunkt in der vorliegenden Arbeit. Ein Überblick über das weitere Vorgehen wird im Verlauf der Arbeit zwar gegeben werden; dieser ist jedoch als Ausblick zu verstehen und kann sich durch neue Erkenntnisse noch grundlegend ändern.

# 2. Hintergrund der Evaluation

## *2.1. Bedeutung von XML*

Mit der Erfindung des World Wide Web (WWW) durch Tim Berners-Lee im Jahre 1989 (vergleiche [WebHistory]) wurde der Grundstein gelegt für die explosionsartige Ausweitung des Internets, die ab Mitte der neunziger Jahre die bisherige Art der Kommunikation grundlegend erweiterte und zum Teil veränderte. Mit dem exponentiellen Wachstum des WWW wurde auch die Hypertext Markup Language (HTML) als Beschreibungssprache für Dokumente im Web schnell weit verbreitet. HTML ist als Sprache nach den Regeln der Standardized General Markup Language (SGML) definiert (vergleiche [HTML]). Diese Definition erfolgt mit einer so genannten Document Type Definition (DTD), auf die im HTML-Dokument verwiesen wird.
Da es sich bei HTML um eine Beschreibungssprache handelt, die gleichzeitig mit den darzustellenden Daten auch die Beschreibung der Formatierung enthält, sind die Knotenelemente in der HTML DTD – so genannte Tags – durch eine starke Kopplung von Formatierung und Daten gekennzeichnet (vergleiche [HTML, „Guidelines for authoring"]). So werden zum Beispiel teilweise Dokumentteile in HTML der Positionierung wegen in Tabellen angeordnet, nicht um eine Korrelation der Tabelleninhalte auszudrücken. In solchen Extremfällen führt die Formatierungsumgebung unter Umständen zu einer Verzerrung der Bedeutung der Inhalte, da sie zum Teil nur existieren, um eine bestimmte Darstellung zu erzeugen (siehe auch [Suhl et al. 2001, Seite 44]). Dies steht einer automatisierten Datenextraktion im Wege.

Ein weiterer Nachteil von HTML besteht darin, dass HTML-Seiten in der Regel nicht durchgängig wohl geformt sind, das heißt, dass nicht jedes Tag aus einem öffnenden und einem schließenden Element besteht. Dadurch ist nicht immer klar, bis wohin in einem Dokument sich der Wirkungsbereich der Formatierungen eines Tags erstreckt; zum Teil bleibt dies der Interpretation des Darstellungsprogramms für HTML-Seiten (Browser) überlassen, das das jeweilige Dokument rendert. Auch dieser Aspekt ist eine große Hürde für die automatisierte Verarbeitung von Webdokumenten.

Ende der neunziger Jahre schenkte Microsoft der Verbreitung des Webs erhöhte Aufmerksamkeit und begann, Netscape, den bisherigen Marktführer für Webbrowser, massiv anzugreifen (siehe auch [WebStatistik]). Dies führte dazu, dass sowohl Microsoft als auch Netscape ihre jeweiligen Browser mit proprietären Tags erweiterten (vergleiche [Suhl et al. 2001, Seite 44]), die in der jeweils aktuellen HTML DTD nicht enthalten waren. Um diese Möglichkeiten auszureizen, durfte die DTD in Webdokumenten nicht mehr erscheinen, um eine Validierung zu verhindern, die wegen dieser proprietären Tags gescheitert wäre. Diese Auslassung verhindert eine automatisierte Validierung als Vorstufe zu einer automatisierten Weiterverarbeitung.

Die vorgenannten Aspekte machten also eine automatisierte Verarbeitung von Webdokumenten extrem schwierig, wenn nicht sogar unmöglich. Mit steigender Verbreitung kam dem Web immer größere Bedeutung bei der Lösung von Integrationsproblemen nach. Es schien die geeignete Plattform zu sein, mit Hilfe von nachrichtenbasierter Datenaustausch den historisch gewachsenen Bedarf nach Integration (vergleiche [Sailer, Seite 210 ff.]) verschiedener Technologien sättigen zu können. Dies resultierte in verstärkten Standardisierungbemühungen und dem Versuch der Elimination der Nachteile von HTML, um die für den automatisierten Austausch notwendige Verarbeitung zu ermöglichen.

Diese Bemühungen führten zur Entwicklung der eXtensible Markup Language (XML), einer Metasprache zur Formulierung von Formatierungssprachen mit speziellen Eigenschaften, die die vorgenannten Nachteile eliminieren (siehe [XML]). So sind XML-Dokumente immer wohl geformt, was ein automatisiertes Einlesen zum Beispiel mit Parsern ermöglicht. Ferner verfügen valide XML-Dokumente über eine DTD, mit der sie sich selbst beschreiben und an deren Struktur sie sich halten müssen und so ihre automatisierte Validierung ermöglichen (siehe [Suhl et al. 2001, Seite 46-47]. Dies bedeutet insbesondere, dass bei der automatisierten Verarbeitung ein XML-Dokument verlässlich bereits sofort nach dem Parsen als fehlerhaft zurückgewiesen werden kann. Dadurch wird vermieden, dass ein Fehler erst in einer späteren Phase beim Zugriff auf die Inhalte eines Dokuments auftritt und dabei potenziell den verarbeitenden Prozess in den Abgrund reißt.

Ferner sind in XML-Dokumenten die Daten von ihrer Darstellung abgekoppelt, da die Bedeutung ihrer Tags nicht an eine Darstellung gebunden ist, sondern sich erst im Kontext der verarbeitenden Anwendung erschließt (vergleiche [Suhl et al. 2001, Seite 46]).

### *2.2. Automatisierte Verarbeitung und Transformation von XML mit XSL*

Durch die klare Struktur der in XML formulierten Daten und durch ihre Fähigkeit zum automatisierten Einlesen und zur automatisierten Validierung können auf solche Daten vor allen Dingen auch Regeln angewendet werden, die diese Daten automatisiert in andere Formate überführen. Eine konsistente Form, solche Regeln zu formulieren, besteht in der Verwendung der eXtensible Stylesheet Language (XSL). Dabei handelt es sich um Transformationsvorschriften, die selbst in XML formuliert sind. Mit diesen Transformationsvorschriften ist es möglich, jedes Tag eines in XML formulierten Ausgangsdokuments in eine beliebige Zeichenkette zu überführen (vergleiche [XSL]). Damit ist es theoretisch möglich, XML-Dokumente in jedes beliebige Zielformat zu überführen. Beispiele für solche Transformationen sind zum Beispiel die Überführung von XML-Textdokumenten in das Portable Document Format (PDF) oder von in XML formulierten Zahlen in eine Scalable Vector Graphic (SVG).
Durch ihre Eigenschaften Wohlgeformtheit, Validierbarkeit und Daten-Darstellungs-Entkopplung eignen sich XML-Dokumente sehr gut zum automatisierten Austausch von Daten zwischen Systemen. Auf diesen Aspekt von XML wird im Kapitel 4 vertiefend eingegangen.

### *2.3. Lotus Notes/Domino und XML*

Lotus bot für die Notes/Domino Produkte schon in der Version R5 (genauenommen R5.02) einige – wenn auch eingeschränkte – Möglichkeiten für den Export von Daten als XML (Details zur Unterstützung von XML in den verschiedenen Versionen werden im Kapitel 3.5.2 erläutert). Ein eigener XML-Dialekt wurde definiert und von Lotus als Domino XML Language (DXL) bezeichnet. Die Definition dieses Dialekts wurde daraufhin in einer entsprechenden DTD definiert und stellte ab sofort den Ausgangspunkt für weitere Bemühungen zur Unterstützung von XML im Lotus Notes/Domino-Umfeld dar.
Einen Schritt in diese Richtung stellt das Lotus XML Toolkit dar, das im nächsten Abschnitt betrachtet werden soll.

## 3. Evaluation des XML Toolkits

Gegenstand des vorliegenden Abschnitts ist die detaillierte Beschreibung des Lotus XML Toolkits in der Version 1.0, das unter [ToolkitURL] kostenlos heruntergeladen werden kann.

### *3.1. Umfang des Toolkits*

Das Toolkit besteht aus einer selbst extrahierenden Archiv-Datei (`DXLTools10.exe`), die durch Ausführung das Verzeichnis `LotusXML` mit diversen Unterverzeichnissen anlegt. Die Unterverzeichnisse und ihr jeweiliger Inhalt können der Tabelle 1 entnommen werden:

*Tabelle 1: Inhalt des Lotus XML Toolkit*

| ***Unterverzeichnis*** | ***Inhaltsbeschreibung*** |
|---|---|
| *bin* | DOS Kommandozeilenwerkzeuge *dxlexport.exe* und *dxlimport.exe* zum Im- und Export von XML |
| *doc* | Dokumentation des Toolkits im HTML- und im NSF-Format |
| *include* | Header-Dateien für die Entwicklung mit C++ |
| *lib* | Bibliotheken zur Einbindung in eigene Applikationen: *DXLTools.jar* für Java sowie im Unterverzeichnis *mswin32* die Dateien *DXLTools10.dll* (Laufzeitbibliothek) und *DXLTools.lib* (Importbibliothek) zur Entwicklung mit C++ |
| *samples* | Beispielanwendungen zum Testen der Funktionalität |

Ferner befinden sich im Verzeichnis LotusXML noch einige Dateien mit Lizenzbestimmungen und weiteren Informationen über das Toolkit (*license.txt*, *relnotes.txt*, *readme.txt*) sowie die DTD zur Definition der Lotus DXL (*domino.dtd*).

## *3.2. Anforderung an das System*

Laut [ToolkitReadme] ist das Lotus XML Toolkit an die Ausführung auf einem System mit einer Win32-Plattform gebunden (Windows XP, Windows 2000, Windows NT 4.0 oder Windows 95/98). Ferner muss auf diesem System Lotus Notes oder Domino in der Version 5.0 oder höher installiert sein, da einige Bibliotheken aus diesen Programmpaketen benötigt werden.
Für die Entwicklung beziehungsweise die Übersetzung der Beispielapplikationen werden Compiler benötigt. Für C++ ist dies Microsoft Visual C++ 6.0 oder höher, für Java ein Java Development Kit in der Version 1.1.8 oder höher. Außerdem werden der Lotus XML Parser und der XSL Transformer benötigt, die ab Version 5.02 Bestandteil von Lotus Notes/Domino sind, ansonsten aber separat von der Lotus Website herunter geladen werden können.

## *3.3. Tools im Toolkit*

Das Toolkit enthält im Unterverzeichnis *bin* zwei Kommandozeilenwerkzeuge, die exemplarisch darstellen, welche Möglichkeiten sich bei der Entwicklung eigener Applikationen mit dem Toolkit ergeben. Bei diesen Werkzeugen handelt es sich um Programme zum Import von DXL-Dokumenten in Domino-Datenbanken (*dxlimport.exe*) sowie zum Export aus Domino-Datenbanken in DXL-Dokumente (*dxlexport.exe*). Ihre Funktionalität erschließt sich aus den Kommandozeilenschaltern und Parametern der Programme, die [ToolkitHilfe, „User's Guide" „About the DXL Import and DXL Export utilities"] bietet eine genaue Auflistung dieser Parameter mit entsprechenden Erklärungen.
Zusammenfassend kann gesagt werden, dass die beiden Tools mit diesen Schaltern eine große Anzahl verschiedener Vorgänge unterstützen. So bietet das Tool *dxlexport.exe* Möglichkeiten,

entweder ein einzelnes Element einer Datenbank über seine ID oder seinen Namen in DXL-Notation zu exportieren. Man kann aber auch ganze Elementklassen bis hin zu einer vollständigen Datenbank exportieren lassen. Dabei hat man bei einigen Elementtypen sogar Wahlmöglichkeiten, in welcher Form (also mit welchem Tag) diese exportiert werden.
Bei dxlimport.exe hingegen wird der zu importierende Inhalt durch die zu importierenden DXL-Daten gesteuert, so dass es in diesem Fall darauf ankommt, wie Konflikte zwischen bestehenden und importierenden Daten gelöst werden. Das Werkzeug unterstützt hier jeweils die vier Möglichkeiten ignorieren, Daten als neues Element anlegen, bestehende Elemente überschreiben sowie bestehende Elemente überschreiben und neue hinzufügen. Das Tool bietet diese Steuerungsmöglichkeiten separat für ACL-Informationen, Designelemente und Dokumente. Außerdem kann eingestellt werden, ob beim Import die Struktur der zu importierenden Daten durch einen validierenden Parser überprüft wird.
Beiden Werkzeugen ist gemeinsam, dass sie sowohl für lokal vorliegende Dateien eingesetzt werden können als auch für Daten, auf die sie über entfernte Domino-Server zugreifen müssen.

## *3.4. Bibliotheken im Toolkit*

Im Toolkit sind Bibliotheken enthalten, die ein Framework zum Im- und Export von DXL-Daten in beziehungsweise aus Domino-Datenbanken zur Verfügung stellen. Diese Bibliotheken können aus eigenen Anwendungen heraus für den Im- und Export von DXL-Daten verwendet werden. Sie enthalten dazu Funktionen, mit denen sie Datenbanken im NSF-Format von der Festplatte beziehungsweise Daten, die ihnen von Domino-Servern zur Verfügung gestellt werden, lesen und dann in XML überführen können beziehungsweise umgekehrt.
Die Bibliotheken stehen für die Sprachen Java und C++ zur Verfügung.

### 3.4.1. Bibliothek für Java

Die Bibliothek zum Einbinden in Java-Programme befindet sich Unterverzeichnis lib und heißt *`DXLTools.jar`*. Entgegen dem „Write Once, Run Anywhere" Paradigma von Java kann diese Bibliothek nicht zur Entwicklung von plattformunabhängigen Anwendungen herangezogen werden. Die in der Bibliothek enthaltenen Klassen greifen per Java Native Interface auf Funktionen in der Lauzeitbibliothek für C++ zurück; dadurch sind solche Java-Anwendungen an die Microsoft Win32-Plattform gebunden. Im Grunde genommen ist es hier angebracht, von einem Wrapper um die C++-Bibliotheken für die Sprache Java zu sprechen, da die Java-Klassen die Funktionalitäten des Toolkits nicht selbst implementieren.

### 3.4.2. Bibliotheken für C++

Für C++ sind im XML Toolkit zwei verschiedene Bibliotheken für Microsoft Win32-Plattformen enthalten, die Laufzeitbibliothek *`DXLTools10.dll`* und die Importbibliothek *`DXLTools.lib`*. Sie enthalten die eigentliche Funktionalität zum Lesen und Schreiben von proprietären Datenforma-

ten von Lotus Notes/Domino sowie zum Ausgeben und Einlesen von Daten im DXL-Format; deswegen wird die Laufzeitbibliothek von der oben angeführten Java-Bibliothek bei der Ausführung von Programmen benötigt.

## *3.5. XML-Fähigkeiten im Umfeld von Lotus Notes/Domino*

### 3.5.1. Klassen und ihre Funktionalitäten im Lotus XML Toolkit

Tabelle 2 listet die in den Bibliotheken enthaltenen Klassen und die enthaltenen Funktionalitäten auf; für eine detaillierte Auflistung der Methoden der Klassen sei auf [ToolkitHilfe, „Reference Guide"] verwiesen.

Anwendungen, die das Framework des Toolkits benutzen, müssen sich an ein generelles Vorgehen halten. Vor allen anderen Schritten muß ein Objekt der Klasse `DXLSession` instantiiert werden (bei Multithreading: ein Objekt pro Thread). Auf diesem Objekt muß vor weiteren Aktionen die Methode `init` (bei Multithreading: `initThread`) ausgeführt werden. Als nächstes wird eine Referenz auf die zu verwendende Datenbank initiiert (`DXLDatabase`) und nach Setzen eventueller Optionen kann ein Export oder Import durchgeführt werden. Vor Beendigung der jeweiligen Applikation muß die `DXLSession` mit der Methode `term` beendet werden.

Eine DXLSession kann beliebig häufig initialisiert und beendet werden, solange dies abwechselnd erfolgt und Abhängigkeiten von Threads berücksichtigt werden (vergleiche [ToolkitHilfe, „User's Guide" - „API program structure"]). Generell empfiehlt Lotus, Referenzen auf Objekte zwischen verschiedenen Threads nicht zu teilen; das heißt, dass jeder Thread auf seiner eigenen Objektinstanz arbeiten sollte.

*Tabelle 2: Klassen des Lotus XML Toolkit*

| ***Klassenname*** | ***In Java-Bibliothek*** | ***In C++-Bibliothek*** | ***Funktionalität*** |
|---|---|---|---|
| `DXLBase` | X | X | Oberklasse für alle anderen Klassen des Toolkits. |
| `DXLDatabase` | X | X | Repräsentiert eine Domino-Datenbank; wird sowohl für Import- als auch Exportoperationen benutzt. |
| `DXLException` | X | X | Oberklasse für `DXLImportException` und `DXLExportException`, die Fehlerinformationen an Applikationen weitergeben. |
| `DXLExporter` | X | X | Bietet Funktionen für den Export von Teilmengen von Domino-Datenbanken in DXL. |
| `DXLExportException` | X | X | Wird vom `DXLExporter` benutzt, um Fehlerstati an Applikationen weiterzugeben. |
| `DXLExportOptions` | X | X | Spezifiziert Optionen, die beim Export von Domino-Datenbanken in DXL benutzt werden. |

| | | | |
|---|---|---|---|
| DXLHandle | | X | Oberklasse für DXLDatabase, DXLException, DXLStream und DXLString |
| DXLImporter | X | X | Bietet Funktionen zum Import von DXL in eine Domino-Datenbank. |
| DXLImportException | X | X | Wird vom DXLImporter verwendet, um Fehlerstati an Applikationen weiter zu reichen. |
| DXLImportOptions | X | X | Spezifiziert Optionen, die beim Import von DXL in Domino-Datenbanken berücksichtigt werden. |
| DXLInputStream | | X | Datenstrom, der die Quelle für Einleseoperationen von DXL-Daten darstellt. |
| DXLOutputStream | | X | Datenstrom, der die Quelle für Ausgabeoperationen von Domino-Datenbanken darstellt. |
| DXLSession | X | X | Repräsentiert auf der Seite des Toolkits das Gegenstück zu einer Notes-Session; muss von allen Applikationen verwendet werden. |
| DXLStream | | X | Oberklasse von DXLInputStream und DXLOutputStream |
| DXLString | | X | Repräsentiert eine Zeichenkette im nativen Zeichencode, Unicode oder LMBCS (Lotus MultiByte Character Set, wird in Domino-Datenbanken verwendet) |

Die im XML Toolkit enthaltene Dokumentation führt die einzelnen Methoden der aufgelisteten Klassen auf. Diese Methoden sind so ausgelegt, dass ein Programm zum Beispiel im Falle eines Exports die erhaltenen Daten weiter verarbeiten kann. Die Daten stehen in Streams zur Verfügung und können in dieser Form in weiteren Schritten manipuliert werden. Auf diese Weise überlässt es das Toolkit dem Programmierer zu entscheiden, was das für die jeweilige Anwendung effizienteste Vorgehen ist. Zum Beispiel kann der Stream im Anschluss an einen Export per SAX oder per DOM geparst werden, je nachdem welche Anforderungen im Kontext der Applikation an Verarbeitungsgeschwindigkeit und Zugriffsart auf die Daten gestellt werden.

Die binär vorliegenden Werkzeuge des Toolkits *dxlexport.exe* und *dxlimport.exe* decken mit ihren vielfältigen Kommandozeilenschaltern einen Großteil der Möglichkeiten des Toolkits ab, so dass ihre Verwendung für einfache Projekte unter Umständen ausreicht. Ein großer Nachteil besteht in diesem Fall jedoch darin, dass unter realistischen Umständen häufig die Notwendigkeit gegeben ist, ihre Ausgaben erst in einer Datei abzuspeichern, bevor diese in einem weiteren Schritt wieder eingelesen und von einer anderen Applikation weiterverarbeitet werden können. Das auf Unix-Systemen in solchen Fällen häufig genutzte Piping-Konzept, bei dem Ausgaben eines Programms direkt als Eingabedaten an das nächste Programm weitergereicht werden, wird auf der

Win32-Plattform von zu wenig Applikationen unterstützt, um dies als häufig zur Verfügung stehende Möglichkeit in Betracht zu ziehen.

Wenn es darum geht, selektiv nur bestimmte Kombinationen von Elemente einer Datenbank zu exportieren, wird die Implementierung eigener Programme unter Verwendung der im Toolkit enthaltenen Funktionalität wird jedoch notwendig. Gibt es in einer Datenbank zum Beispiel drei Views und fünf Forms und es geht in einer konkreten Aufgabenstellung darum, nur den zweiten View und die dritte Form im DXL-Format zu exportieren, reichen die Fähigkeiten von `dxlexport.exe` nicht aus, da es entweder nur ganze Elementklassen (also alle Views oder alle Forms) oder nur ein benanntes Element exportieren kann. In einem solchen Fall kann eine XSL Transformation im Anschluss an einen DXL-Export den gewünschten Effekt erzielen, der aber die programmatische Auswahl mit Hilfe der Klassen des Toolkits aus Gründen der Performanz und des eingesparten Arbeitsschritts des erneuten Einlesens auf jeden Fall vorzuziehen ist.

### 3.5.2. Funktionalitäten im Vergleich zu bestehenden Lotus Notes/Domino Releases

Betrachtet man die Produkte im Umfeld der Lotus Notes/Domino Produktreihe bezüglich ihrer Bestandteile mit Fähigkeiten zur Verarbeitung beziehungsweise Generierung von XML, können die folgenden einzelnen Bestandteile identifiziert werden:

- Lotus XML Parser; bestehend aus einer Java-Bibliothek
- Lotus XSL Transformer, bestehend aus einer Java-Bibliothek
- URL Kommandos; dabei handelt es sich um Skripte, die beim Aufruf von Views über den in Lotus Domino integrierten Webserver an die jeweilige URL angehängt werden und die Daten des Views im DXL-Format ausgeben
- DXL Tools; unter diesem von den Verfassern selbst gewählten Begriff sind verschiedene Werkzeuge zur Bearbeitung von XML zusammengefasst: Importer, Exporter, Viewer und Transformer. Es besteht außerdem die Möglichkeit, mit Hilfe der mit Lotus Notes/Domino mitgelieferten Java-Klassen – dabei handelt es sich nicht um die unten erwähnten Java-Klassen für XML – auf der Ebene einzelner Dokumente DXL-Daten zu exportieren (vergleiche [Designerhilfe]). Da diese Möglichkeit aber auch im Exporter enthalten ist, wird sie hier nicht weiter betrachtet.
- LotusScript-Klassen für XML; dabei handelt es sich um einen Satz von LotusScript-Klassen mit Funktionen zum Im- und Export von DXL-Daten, zur XSL Transformation, für das Parsing mit DOM und SAX und zur selektiven Zusammenstellung einzelner zu exportierender Datenbankelemente. Einzelheiten können [Designerhilfe] entnommen werden.
- C++-Klassen für XML wie unter 3.5.1 beschrieben
- Java-Klassen für XML wie unter 3.5.1 beschrieben

Die Zuordnung der aufgeführten Bestandteile zu den jeweiligen Produkten ab Version 5.0 geht aus Tabelle 3 hervor.

*Tabelle 3: Aspekte der XML-Unterstützung in Lotus Notes/Domino Releases und dem Lotus XML Toolkit*

| ***Produkt*** | ***XML Parser*** | ***XSL Trans-former*** | ***URL Com-mands*** | ***DXL Tools*** | ***XML Lotus Script Klassen*** | ***XML C++ Klassen*** | ***XML Java Klassen*** |
|---|---|---|---|---|---|---|---|
| Lotus Notes/Domino vor Release 5.02 | - | - | - | - | - | - | - |
| Lotus Notes/Domino ab Release 5.02 | X | X | X | - | - | - | - |
| Lotus Notes/Domino ab Release 6.0 | X | X | X | X[1] | X | - | - |
| XML Toolkit | - | - | - | X[2] | - | X | X |

Die folgende Darstellung stellt diese Zusammenhänge noch einmal dar und erlaubt die Identifizierung von Schnittmengen in den Funktionalitäten. Da es vor der Version 5.02 keine auf XML bezogenen Fähigkeiten in Lotus Notes/Domino gibt, werden ältere Versionen in der Darstellung nicht berücksichtigt.

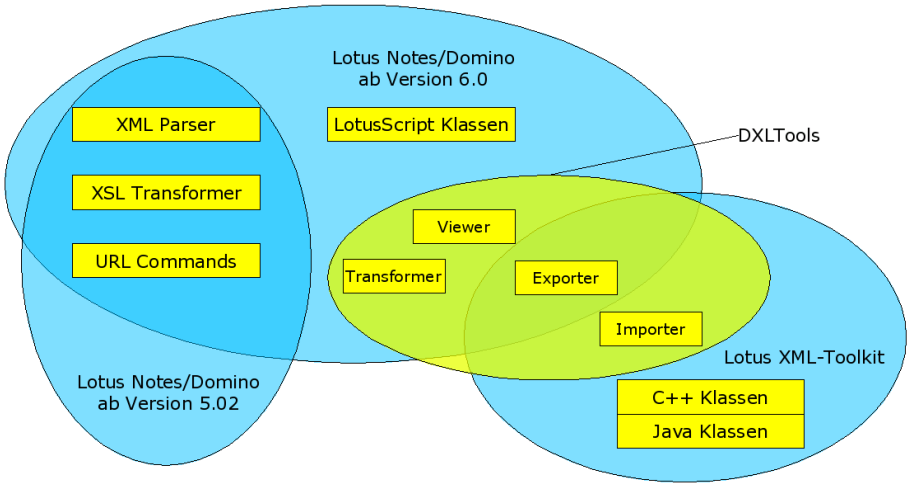


*Abbildung 1: XML-Unterstützung verschiedener Lotus-Produkte*

Zusammenfassend ist festzustellen, dass das Lotus Notes/Domino ab Version 6.0 bereits Möglichkeiten zur Arbeit mit XML beziehungsweise DXL „out of the box" mitbringt. Für spezielle, einmalig auftretende Aufgaben ist unter Umständen die Bearbeitung mit den DXL Tools ausreichend. Für wiederkehrende Aufgaben, bei denen automatisiert Transformationen und andere Manipulationen von DXL-Daten durchgeführt werden müssen, ist LotusScript das Mittel der Wahl. Diese Sprache stellt ab Version 6.0 Möglichkeiten zum Im- und Export von DXL-Daten sowie zur feingranularen Auswahl von zu bearbeitenden Objekten zur Verfügung. Allerdings ist die Bearbeitung durch ande-

1 Nur Exporter, Transformer und Viewer

2 Nur Exporter und Importer wie unter 3.3 beschrieben

re Programmierhochsprachen wie Java und C++ nur grobgranular möglich und die Unterstützung für diese Sprachen bezüglich XML lückenhaft.
Die Lücke zur Funktionalität von LotusScript wird für diese Sprachen durch das Lotus XML Toolkit geschlossen. Es werden durch das Toolkit feingranulare Selektionsmöglichkeiten bezüglich der zu bearbeitenden Elemente bereitgestellt, außerdem werden Möglichkeiten für die genannten Sprachen zum Import von Daten, die im DXL-Format vorliegen, in eine Domino-Datenbank gegeben. Das binäre Werkzeug *`dxlimport.exe`*, das als Beispielimplementierung solcher Möglichkeiten dient, kann außerdem wiederum für Aufgaben mit einmaligem Charakter herangezogen werden.
Dadurch dass mit den Werkzeugen und Klassen des Toolkits die Möglichkeit besteht, auf entfernte Daten über Server zuzugreifen, können dedizierte Maschinen für die XML-Verarbeitung aufgesetzt werden. Dadurch wird zum einen unterstrichen, dass die XML-Verarbeitung nicht zwangsweise an einen einzelnen Domino-Server gebunden ist; zum anderen kann dies helfen, eine nicht unerhebliche zusätzliche Belastung der Domino-Server vermeiden, auf die die jeweiligen Applikationen zugreifen.

# 4. Einbettung des Projekts im Kontext der IT

## *4.1. Situation*

Um sich der Bedeutung von XML in der Verbindung mit Lotus Notes/Domino bewusst zu werden, hilft zunächst ein Blick auf die Strategien und Trends aus IT-Branche. So prognostiziert August-Wilhelm Scheer, Gründer der IDS Scheer AG, für die Lage der IT-Branche in Deutschland:
"Das Thema Outsourcing gewinnt im Jahr 2003 bei Banken, Versicherern und Energieversorgern an Bedeutung. Man wird branchenübergreifend die Einsparpotenziale durch Customer-Relationship-Management (CRM) und SCM erkennen. [...] (Es) sind im Bereich UMTS erste konkrete Anwendungen zum Mobile Computing sichtbar. Obwohl noch in den Kinderschuhen, wird sich auch der deutsche EAI-Markt mittel- und langfristig gut entwickeln." [Computerwoche 2002] Diese Strategien und Unternehmensanwendungen können durch die Verwendung von XML zur Standardisierung von Daten und deren Austausch schneller und kostengünstiger umgesetzt werden. XML leistet so Integrationshilfe in den Bereichen CRM, SCM (Supply Chain Management), "Mobile Business/Computing" und EAI (Entreprise Application Integration). Andererseits ermöglicht es das Auslagern von Informations- und Kommunikationstechnologischen Aspekten ("IT-Outsourcing") durch den standardisierten Datenaustausch zwischen Anwendungen.
Weitere Anstrengungen von Unternehmen, die von XML unterstützt werden können, zielen auf EDI (Electronic Data Interchange) CSCW, Workflow Automatisierung, Katalogdaten- und Dokumentenmanagement ab. [ECIN 2000]

## *4.2. Trends im Umfeld von XML und Lotus Notes/Domino*

Zur Kopplung von Applikationen mit Lotus Notes/Domino besteht die Möglichkeit zum Export beliebiger Teile von Domino-Datenbanken in ein XML-Format beziehungsweise zum Import von Daten in Domino-Datenbanken. Dies stellt eine Schnittstelle dar, mit der innerhalb kürzester Zeit auf kostengünstige Art und Weise verschiedene andere Systeme an Lotus Notes/Domino angebunden werden können.

Deswegen bot Lotus für die Notes/Domino Produkte auch schon in der Version R5 einige – wenn auch eingeschränkte – Möglichkeiten für den Export von Daten als XML.

Mit XML als Zwischenformat bei der Konvertierung von Daten zwischen Lotus Notes/Domino und anderen Systemen können in Zukunft spezielle Konnektoren auf einfache Weise realisiert werden, indem sie entsprechend der DXL DTD programmiert werden. Im günstigsten Fall reicht es sogar, wenn ein Konnektor die Daten gemäß DXL von der Datenbank empfängt, von einem Parser einlesen, von einem Validierer überprüfen und von einem Transformer mit XSLT in das Zielformat überführen lässt. Der Programmieraufwand beschränkt sich in diesem Fall auf den Aufruf von einigen wenigen Methoden; eine fehleranfällige selbst zu implementierende Logik ist unnötig.

Da XML-Parser, XML-Validierer und XSLT-Transformer mittlerweile in vielen verschiedenen Programmierhochsprachen wie C++ und Java zur Verfügung stehen, gibt es eine große Auswahl von Werkzeugen mit denen eine solche Anbindung realisiert werden könnte. Da es sich bei XML um einen offenen und freien Standard handelt, für dessen Verwendung keine Lizenzen erworben werden müssen, gibt es auch eine entsprechend große Basis an Entwicklern, die sich der Erstellung solcher Tools annehmen können. Die Kombination aus vielen Werkzeugen und vielen Programmierern kann dazu führen, dass schon bald auf der Basis von XML eine große Anzahl von Werkzeugen zur Lösung der verschiedensten Probleme im Bereich der Konnektivität im Umfeld von Lotus Notes/Domino entstehen. Voraussetzung dafür ist allerdings, dass der Zugriff auf beliebige Teilmengen von Domino-Datenbanken per XML möglich ist. Als Schritt in diese Richtung hat Lotus das Lotus XML Toolkit zur Verfügung gestellt.

Bei Lotus Notes/Domino handelt sich um ein Büroinformationssystem, welches auch Informations- und Kommunikationssysteme (IKS) genannt wird. Solche Systeme bieten im B2B-Bereich nicht "die operative Unterstützung eines mehr oder weniger automatisch ablaufenden Massenbetriebs repetitiver und gut formalisierbarer betrieblicher Transaktionen" wie es beispielsweise das SAP-System anbietet (vgl. [Nastansky 95], Seite 182). Sie legen ihren Schwerpunkt auf die betriebliche Kommunikation und Steuerung von Arbeitsabläufen.

So entfallen die Bereiche, die auf Massentransaktionen fokusiert sind und zu denen dann EDI, Katalogdatenmanagement, CRM mit Data-Mining und SCM gehören. Jedoch in folgenden Bereichen kann Lotus Notes/Domino in Verbindung mit XML zu neuen Lösungen beitragen:

- EAI
- Workflow Automatisierung
- CSCW
- Mobile Business/Computing
- Dokumentenmanagement

Um nun eine EAI-Lösung zu konzipieren, die traditionelle, separate Anwendungen in verbundene und automatisierte E-Business-Lösungen integriert, bietet XML eine Möglichkeit zur Kopplung verschiedenenartiger Systeme. Durch einen unternehmensweiten vereinheitlichten XML-Standard können die an den Unternehmensprozessen beteiligten Informationssysteme Daten austauschen und so kann ein durchgängiger Prozess ohne Medienbrüche geschaffen werden. Unter dem Begriff Medienbruch wird in diesem Zusammenhang auch der Umstand verstanden, dass bereits digital vorliegende Informationen, die in Dateien von verschiedenen Programme enthalten sind, nicht automatisiert zu anderen Applikationen weitergereicht werden können. Um dies zu vermeiden, werden Dokumente aus den Domino-Datenbanken als XML-Dateien exportiert und mit XSL transformiert. Nachrichten bzw. Dokumente, die durch ein anderes System erzeugt wurden, können auf umgekehrtem Wege erst transformiert und dann importiert werden.

Damit bereits angesprochen ist die "Workflow Automatisierung", die sich bisher hauptsächlich nur in dem Bereich des CSCW ("Computer Supported Cooperative Work") wieder fand. Auch hier bieten XML-Schnittstellen zu bestehenden Anwendungssystemen Integrationsmöglichkeiten, so dass Workflows nicht nur innerhalb der Lotus Notes/Domino Umgebung, sondern auch darüber hinaus mit anderen Systemen innerhalb des Firmen Intranets.

Im Rahmen des "Mobile Business/Computing" wird durch XML Zugriff auf Informationen aus Domino-Datenbanken ermöglicht, ohne dass die gesamten Notes-Clients auf mobilen Endgeräten wie PDAs und Mobiltelefonen installiert werden müssen. Trotz der rasch fortschreitenden Verbesserung der Technik in Bezug auf Speicher und Übertragungsmöglichkeiten sind die bestehenden Lösungen für den Einsatz auf solch stark eingeschränkten Geräten nicht möglich und bedürfen neuer Konzepte. Hier findet das Produkt "Lotus Everyplace" sowie die XML-Fähigkeiten der neueren Lotus Notes/Domino Generation R5 und R6 Einsatzmöglichkeiten.

Weiterhin wird sich die Art des Dokumentenmanagements durch den Einsatz von XML stark ändern. Mit der Standardisierung von Dokumenten-Semantik und der Schaffung von XML-fähigen Agenten und Suchsystemen ist durch XML-Schnittstellen die gezielte Suche nach Informationen durchführbar(vgl. [ECIN 2000]). Die Agenten und Suchsystemen werden mittels neuer Web-Technologien z.B. Webspider als intelligente Softwareagenten implementiert. Dies kann auch auf Informationen angewendete werden, die als Dokumente in Domino-Datenbanken vorliegen und dann weltweit über das Internet verfügbar sind.

Somit lässt sich feststellen, dass XML zur Kopplung und als Schnittstelle zwischen Lotus Notes/Domino und anderen Applikationen im Unternehmensumfeld ein großes Einsatzfeld findet.

# 5. Einbettung des Projekts im Kontext des GCC

Das Projekt "NT2: XML/DXL" dient der Evaluation des Lotus XML Toolkits und nach Verständnis der Autoren der Beschreibung der daraus entstehenden Fähigkeiten der neueren Lotus Notes/Domino Produkte der Generation R5 und R6. Dieses Projekt steht in Beziehung zu einigen der derzeit laufenden Projekte. Diese Projekte und ihr Bezug zu XML beziehungsweise dem vorliegenden Projekt werden im Folgenden erläutert.

## *5.1. GCC 1: Entwicklung mobiler Applikationen mit Domino Everyplace*

Im Rahmen des Projektes GCC 1 soll eine mobile Applikationen zur Benutzung der Winfo-Foren Diskussionsdatenbank entworfen werden. Dabei wird eine Datenbankanwendung für einen Mobile Notes Client mit Lotus Everyplace entwickelt. Im Gegensatz dazu wird der im vorliegenden Projekt zu entwickelnde Prototyp keine Notes-Lösung auf dem Client, sondern eine voraussichtlich auf Java basierende Applikation benötigen. Die Notes/Domino-eigenen Replikationsmechanismen werden nicht benutzt, vielmehr werden per XML exportierte und transfomierte Daten an den Client übertragen. Daher werden im Gegensatz zu der Everyplace-Lösung keine Replikationen oder Offline-Zugriffe möglich sein, aber es wird auf Systemen mit geringerer Hardware-Anforderung laufen und einen Lösungsansatz unter Verzicht auf kommerzielle Werkzeuge finden.

## *5.2. DEK 3: Verfügbarkeit wichtiger Informationen auf Mobilen-Endgeräten*

Die wichtigsten Informationen der Fakultät der Wirtschaftswissenschaften sollen auch für mobile Endgeräte abrufbar sein. Um dieses Ziel zu erreichen, können die geforderten Dokumente oder Views aus Notes heraus mit dem XML Toolkit exportiert und an die mobilen Endgerät übermittelt werden. Mit einem Export im XML-Format und anschließender Transformation mittels XSL-Stylesheets können beliebige Zielformate erzeugt und auf dem Client dargestellt werden und es bleibt eine höchstmögliche Flexibilität erhalten.

## *5.3. G8 (2): Interportletkommunikation im Kontext des G8-Portals*

Die Kommunikation-/Interaktion zwischen Portlets des G8-Portals soll ermöglicht werden.

Für die Art des Datenaustausches zwischen den Portlets sollte ein einheitliches Format gewählt werden. Hier bietet sich XML an. Falls die Portlets auf Dokumente innerhalb der Lotus Notes/Domino Datenbanken zugreifen und direkt weitergeleiten, können jene auch direkt in einem XML-Format exportiert werden und so vereinfacht an andere Portlets weitergereicht werden. Dies ist nicht notwendig, wenn nur Links zwischen den Portlets ausgetauscht werden.

### *5.4. G8 (3): Möglichkeiten des Austausch von Notes-Dokumenten mit Domino 6*

Bei dem Projekt G8(3) soll untersucht werden, wie zwischen verschieden Portalen Portlets Informationen austauschen können. Dazu bieten die XML-Fähigkeiten der neuesten Generation R6 von Lotus Notes/Domino Produkten Lösungsmöglichkeiten. Allerdings sind diese Möglichkeiten auch schon unter R5 umsetzbar, falls das Lotus XML Toolkit und je nach Version weitere Komponenten installiert werden. Als Lösungsansatz ließen sich die Notes-Dokumente, die als Informationen an ein anderes Portal übermittelt werden sollen, im XML-Format exportieren, transformieren, und als serialisierter Datenstrom verschicken. Das empfangende Portlet kann über ein speziell entwickeltes Stylesheet die empfangenden Daten in ein für sich lesbares Format transformieren lassen und Einträge auswerten.

## 6. Anwendung der Erkenntnisse auf einen Prototypen

### *6.1. Beschreibung und Zielsetzung*

Im zweiten Semester soll im Projekt ein Prototyp zur Nutzung des Lotus Notes Gruppenkalenders auf mobilen Endgeräten entwickelt werden. Der Prototyp soll die Gruppenkalender-Funktionalität durch Online-Zugriff auf die entsprechenden Daten zur Verfügung stellen. Das heißt, dass die Daten nicht zwischengespeichert und dann unregelmäßig durch einen Synchronisationsmechanismus mit den entsprechenden Datenquellen abgeglichen werden; stattdessen werden die Daten während der Interaktion durch direkten Zugriff per Netzwerk auf die Datenquellen zur Verfügung gestellt.

Die Projektbeschreibung sah zunächst die Entwicklung eines Prototypen mit Lotus Domino Everyplace vor. Allerdings bedeutet dies eine Einschränkung auf die vom Produkt unterstützten Zielplattformen. Um eine solche Einschränkung zu vermeiden und um die zu übermittelnden Daten beeinflussen zu können, wird der Prototyp stattdessen auf einer selbst zu erstellenden Architektur aufsetzen.

Die folgenden Unterabschnitte beschäftigen sich damit, inwiefern XML beim Lösungsansatz Bedeutung zukommt, wie die Funktionalitäten aus dem XML Toolkit dabei genutzt werden können

und wie die notwendige Architektur aussieht. An dieser Stelle sei erneut darauf hingewiesen, dass es sich bei diesen Erläuterungen um einen Ausblick handelt, der sich durch neue Erkenntnisse grundlegend ändern kann.

### 6.1.1. Transferierte Daten

Programme, die auf mobilen Endgeräten zum Einsatz kommen, unterliegen normalerweise starken Einschränkungen bei der Rechenleistung und dem zur Verfügung stehenden Arbeitsspeicher. Es kommt also bei der Entwicklung eines solchen Prototypen darauf an, die zu verarbeitende Datenmenge möglichst gering zu halten. In einem Online-Szenario ist es zusätzlich wichtig, das zu übertragende Datenvolumen möglichst niedrig zu halten, um subjektiv ein Gefühl der flüssigen Verarbeitung beim Nutzer zu erzeugen. Beide Aspekte zusammen deuten darauf hin, dass der Beschränkung auf möglichst nur die Daten, die für die Funktionserfüllung unbedingt notwendig sind, das Hauptaugenmerk gilt.

Die für den Prototypen angestrebte Lösung besteht darin, die Daten aus dem Gruppenkalender im DXL-Format zu exportieren und diese mit XSLT auf die für das Endgerät notwendigen Elemente zu minimieren. Dadurch wird die durch das Endgerät darzustellende Datenmenge reduziert.

Um das mobile Endgerät dabei zu entlasten, wird zwischen Domino-Server und mobilem Endgerät eine weitere Verarbeitungsschicht eingefügt, die stellvertretend für das mobile Endgerät agiert: Sie nimmt proprietäre Daten vom Server entgegen, exportiert diese als DXL, transformiert sie mit XSL und leitet diese reduzierte Menge dann an das Endgerät weiter. Dadurch wird das Endgerät von der mit den verschiedenen Umformungen verbundenen Rechenlast befreit, die zu übermittelnde Datenmenge wird reduziert und die Speicherauslastung auf dem Endgerät klein gehalten.

Um möglichst viele verschiedene Endgeräte zu unterstützen, wird die Stellvertreterschicht – im weiteren Text als Proxy bezeichnet – möglichst flexibel gehalten, indem sie zwar Logik anstelle der Endgeräte anwenden kann, diese Logik aber nicht selbst mitbringt. Stattdessen wird ihr die jeweils notwendige Transformationslogik in Form eines vom Endgerät übermittelten XSL Stylesheets zur Verfügung gestellt. Dadurch kann eine für jedes Endgerät optimierte Transformation vorgenommen werden, ohne dass der Proxy das zu unterstützende Endgerät a priori kennen muss. Nach [Kao, Seiten 1-27 ff.] sind Offenheit – die Erweiterbarkeit eines Systems – und Skalierbarkeit – die Effizienzwahrung bei steigender Anzahl von Komponenten – wichtige Aufgabenstellungen beim Entwurf eines verteilten Systems. Durch das vorgestellte Vorgehen erfüllt das System diese Ansprüche.

### 6.1.2. Architektur

Die notwendige Architektur besteht aus drei Ebenen, die in der nachfolgenden Auflistung erläutert werden. Eine Erläuterung der konkreten Implementierungsvorgaben wird unter 6.1.3 gegeben.

- Ein mobiles Endgerät stellt den Client dar, der die Gruppenkalender-Funktionalität einem Nutzer zur Verfügung stellt. Auf dieser Ebene muss eine geeignete Client-Applikation entwickelt werden, die die erhaltenen Daten interpretiert und darstellt.
- Ein Serverprozess nimmt Anfragen der Clients entgegen und fragt als Proxy für die Clients das Daten-Backend ab, überführt die Antwort in DXL und verändert die Antwort des Backends mit einem vom Client übermittelten XSL Stylesheet so, dass sie den Anforderungen der Clients entspricht.
- Das Daten-Backend besteht aus einem Domino-Server. Dieser Server bedarf keiner weiteren Einstellungen, der Proxy tritt dem Server gegenüber als regulärer Notes-Client auf. Die Benennung des zu benutzenden Servers erfolgt durch den Client für jede Anfrage einzeln, so dass Proxy und Server nicht fest aneinander gekoppelt sind.

Diese Architektur ist – wie bereits erwähnt – lediglich ein Ausblick; neue Erkenntnisse können hier noch zu grundlegenden Änderungen führen. Die Phase „SOLL-Konzept“ wird diese Architektur entweder bestätigen oder zu einer Neudefinition führen. In Abbildung 2 wird die Architektur grafisch dargestellt.

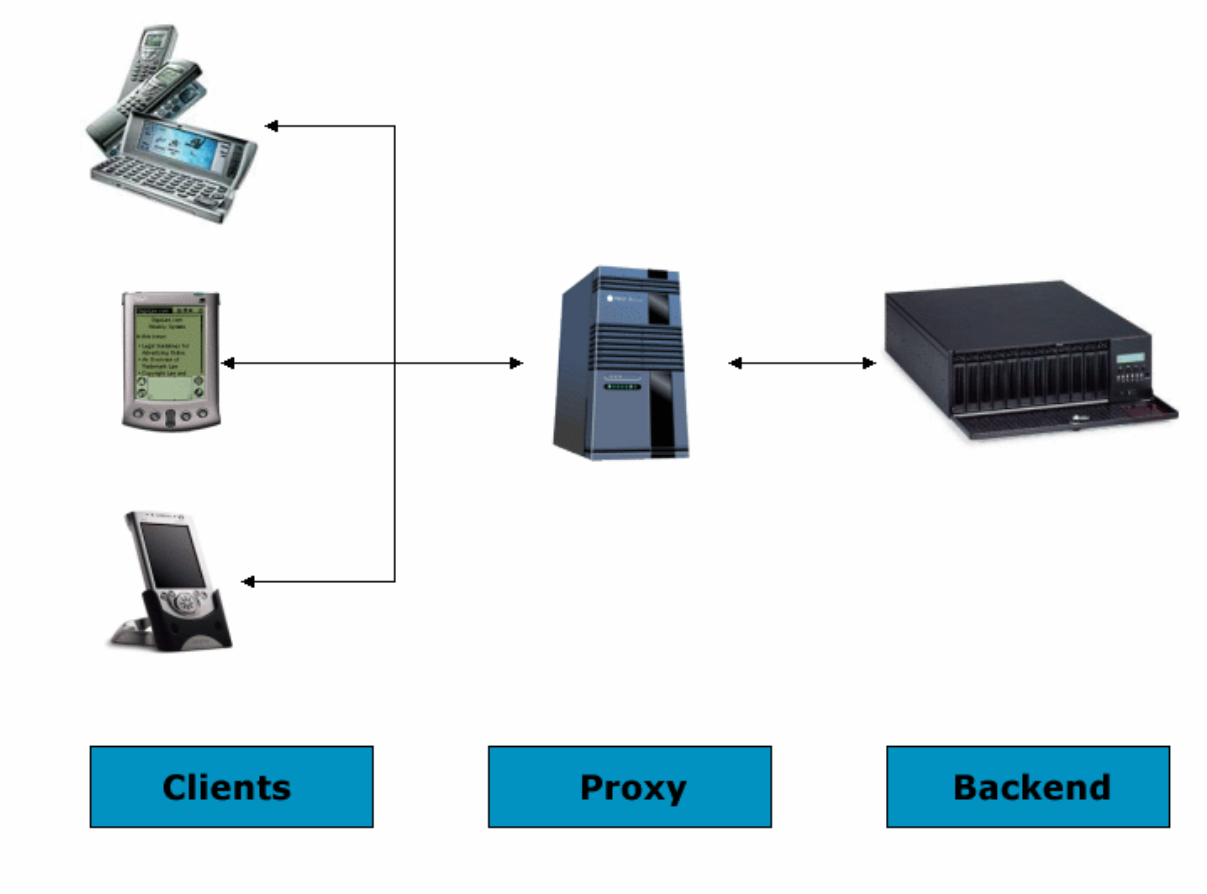


*Abbildung 2: Architektur für die Nutzung auf mobilen Endgeräten*

### 6.1.3. Implementierungsvorgaben

Nach derzeitigem Stand der Planung gelten die im Folgenden aufgeführten Vorgaben für die Implementierung des Prototypen:

- Als mobiles Endgerät wird ein Compaq Ipaq H3760 mit Familiar-Linux (vergleiche [Familiar]) und der grafischen Oberfläche OPIE (vergleiche [OPIE]) benutzt. Auf dem Endgerät steht eine Java Virtual Machine (vergleiche [CVM]) gemäß Java 2 Micro Edition (J2ME), Connected Device Configuration (CDC), Personal Profile (vergleiche [PersonalProfile]) zur Verfügung. Das Client-Programm wird für diese Laufzeitumgebung entwickelt.
- Als Proxy kommt ein Java Servlet zum Einsatz, das mit Hilfe des Lotus XML Toolkits die Ausgaben eines Domino-Servers im DXL-Format mit einem Stylesheet in ein clientspezifisches Format überführt und diese Daten dann an den Client ausliefert. Alternativ kann statt eines Java Servlets auch ein Java Programm per RMI diese Aufgaben übernehmen.
- Als Daten-Backend kommen beliebige Domino-Server in Frage, die eine Gruppenkalender-Funktionalität bieten. An ihnen sind keine Einstellungen zur Nutzung nötig.

## *6.2. Vorgehen nach Phasen*

Zur Entwicklung eines Prototypen, der die Funktionalitäten in Bezug auf XML demonstrieren kann, wird das so genannte "überlappende Phasenschema (zur Individualsoftwareentwicklung)" angewandt. Es beinhaltet die Phasen "Vorstudie", "IST-Analyse", "SOLL-Konzept", "Systementwurf", "Implementierung und Test" und "Einführung".

Die Vorstudie ist mit den Ergebnissen der XML/DXL-Evaluation abgeschlossen und kann als Erreichen des ersten Meilensteines mit der Zwischenpräsentation zum Februar 2003 betrachtet werden.

Die folgenden Phasen beginnen zeitversetzt, enden allerdings nicht vor der nächsten Phasen, sondern laufen gleichzeitig ab. Der Aufwand verteilt sich jedoch nicht gleichmäßig. Dies ist ein Charakteristikum des überlappenden Phasenschematas.

Bei der IST-Analyse geht es um die Feststellung des organisatorischen und technischen Ist-Zustandes. Dabei werden die Prozesse der Terminvereinbarung via Lotus Notes/Domino Groupcalendar bei einer fiktiven organisatorischen Einheit (OE) untersucht und in Form einer groben Systemerhebung festgehalten. Darin enthalten sind die Aufbauorganisation, die Ablauforganisation sowie die IT-Infrastruktur. Eine Schwachstellenanalyse wird nicht durchgeführt, da es sich um eine konstruierte OE handelt.

Nachdem der Grossteil der Analyse abgeschlossen ist, wird in der Entwicklung des SOLL-Konzepts der organisatorische Soll-Zustand zur elektronischen, mobilen Terminabsprache festgelegt. Dabei erfolgt eine detaillierte Beschreibung der Systemanforderungen aus Benutzersicht. Möglicherweise wird vorhandene Standardsoftware als Vergleichsbasis analysiert.

Im anschließenden Systementwurf wird die Systemarchitektur beschrieben. Dazu werden Benutzeroberflächen entworfen, die Datenmodellierung / Funktionsentwurf oder der Entwurf der Klassen, Methoden und Objektlebenszyklen vorgenommen. Eine mögliche Architektur bestände aus einem Servlet zur Steuerung der XML-Ex-/Importe aus den Lotus Notes/Domino Datenbanken heraus, das von einem Client aus angesprochen wird und die gewünschten Daten nach Transformation mit XSL-Stylesheets wieder an diesen zurückgibt. Darauf basierend sind drei Iterationen zur Entwicklung vorab geplant. Es gibt jeweils einen funktionstüchtigen Prototyp als Ergebnis der Iterationen.

1. Iteration
   Über einen Browser wird eine Anfrage an das Servlet geschickt, und eines Stylesheets, das als Parameter übergeben wurde, transformiert.
2. Iteration
   Eine entwickelte Applikation übermittelt ein eigenes Stylesheet für die Anfragen an das Servlet, und erhält von Servlet die Daten in der gewünschten Form zurück.
3. Iteration
   Die Applikation wertet aufgrund implementierter Logik die erhaltenen Daten aus, nimmt vom Benutzer Anweisungen entgegen, und übermittelt Informationen zur Eingabe in die Datenbank.

Dieses stellt eine vorläufige Möglichkeit dar, da der Systementwurf erst Ergebnisse aus den vorherigen Phasen benötigt.
Anschließend erfolgt die Phase "Implementierung und Test" mit dem Ziel der Erstellung des lauffähigen Systems. Dabei wird jeweils der Modulentwurf, die Codierung und Test vorgenommen sowie die Produktdokumentation fertig gestellt.
Die Einführung mit dem Ziel der Umstellung der Organisation und Übergang zur Arbeit mit dem neuen System entfällt, da im Rahmen des Projektes nur eine prototypische Entwicklung gefordert ist ein solcher Prototyp wird zur Abschlusspräsentation als letzter Meilenstein zur Verfügung stehen.

# 7. Fazit und Ausblick

XML wird sich allem Anschein nach auf absehbare Zeit zu dem dominierenden Standard für Webdokumente entwickeln. Die Unterstützung von XML ist daher zwingende Notwendigkeit für Produkte, die am Markt bestehen wollen.
Auch Lotus hat in der Version R6 seine Domino-Plattform mit Unterstützung für XML ausgestattet. Mit LotusScript steht Programmierern eine interpretierte Sprache zur Nutzung dieser Möglichkeiten zur Verfügung, mit den DXL Tools können kleinere Aufgaben händisch erledigt

werden. Für den externen Zugriff auf Domino – also von einem anderen Rechner oder aus einem anderen Programm heraus – ist die Unterstützung von Programmiersprachen nötig, die nicht auf die Interpretation innerhalb der Domino-Umgebung angewiesen sind. Die Unterstützung für die Sprachen Java und C++ sind aber in Bezug auf XML lückenhaft.

Das Lotus XML Toolkit behebt diesen Mangel und realisiert für Java und C++ fast den selben Umfang an Funktionen bezüglich XML wie für LotusScript. Damit ist der Grundstein gelegt für die Erstellung von Programmen, die auf Domino zugreifen, indem sie exportierte DXL-Daten auswerten. Auf diese Weise können auf einfache Art und Weise Zusatzfunktionalitäten realisiert werden, die Lotus nicht selbst unterstützt.

Für den mobilen Zugriff auf Domino-Server verfügt Lotus mit Everyplace über ein eigenes Produkt. Allerdings unterstützt dieses Produkt nur wenige Zielplattformen und es ist nicht klar, wie effizient die Datenreduktion für mobile Clients ist. Die Nutzung einer nicht unterstützten Client-Architektur stellt genau den oben beschriebenen Fall nicht gebotener Funktionalität dar, die durch eine eigene Applikation unter Nutzung von DXL implementiert wird.

An Hand der Gruppenkalenderfunktionalität wird im Sommersemester exemplarisch an einem Prototypen gezeigt werden, dass und wie für diese Aufgabenstellung unter Verwendung des Lotus XML Toolkits eine Lösung erarbeitet und umgesetzt werden kann.

# Glossar

| | |
|---|---|
| ACL | Access Control List, Zugriffskontrollliste |
| API | Application Programing Interface, Programmierschnittstelle |
| ASP | Application Service Providing, Dienstleistungen auf Anwendungsebene |
| B2B | Business To Business |
| Backend | Komponente einer Rechnerlandschaft, die im Hintergrund (ohne dass der Benutzer mit ihr in direkten Kontakt kommt) für andere Komponenten Dienstleistungen erbringt |
| Binaries | Ausführbare Dateien, die im Binärformat für eine Architektur und ein Betriebssystem vorliegen |
| Browser | Programm zur Darstellung von Webdokumenten |
| C++ | Programmierhochsprache |
| CSCW | Computer Supported Collabarative Work, wird in Bezug auf Software häufig als Synonym für Groupware benutzt. |
| DOM | Document Object Model; Modell für die Verarbeitung von XML-Dokumenten, bei dem den einzelnen Bestandteilen eines Dokurments Objekte zugeordnet werden |
| DTD | Document Type Definition, Strukturdefinition von XML-Dokumenten |
| DXL | Domino XML Language, von Lotus in XML definiertes Dokumentformat |
| EAI | Enterprise Application Integration, Zusammenführung der vierschiedenen Anwendungen im Unternehmen unter einheitlichem Zugriff |
| ERP | Enterprise Resource Planning, computergestützte Zuweisung von Resourcen an Komponenten des Produktionsprozesses |
| Framework | Konsistente Menge von Programmfunktionen zur Lösung von Problemen eines bestimmten Bereiches |
| HTML | Hypertext Markup Language, Auszeichnungssprache für Webdokumente |
| IKS | Informations- und Kommunikationssysteme |

| | |
|---|---|
| IT | Informationstechnik |
| Java | Programmierhochsprache |
| KMU | Kleine und mittelständische Unternehmen |
| Konnektor | Funkion zur Anbindung eines Systems an ein anderes |
| Legacy-System | Veraltetes oder auf andere Art nicht den aktuellen Standards entsprechendes System |
| LotusScript | Programmiersprache, die innerhalb von Lotus Notes/Domino interpretiert werden kann |
| Parser | Programm zum Einlesen und zur Überführung von XML Dokumenten in eine Speicherstruktur |
| PDA | Personal Digital Assistant, kleiner Computer mit Funkionen zum Management persönlicher Informationen und eventuell weiteren Funktionen |
| Proxy | Programm, das stellvertretend für andere Programme Aufgaben unter Verwendung einer eigenen Logik erledigt |
| RMI | Remote Method Invocation, Entfernter Methodenaufruf |
| SAX | Simple API for XML, ursprünglich eine reine Java-Programmierschnittstelle, die auf diesem Weg zu einem „de facto" Standard wurde und für andere Sprachen portiert wurde. |
| SGML | Standardized General Markup Language, Formatierungssprache |
| Stylesheet | Formatierungsanweisungen, die auf die Elemente eines Dokuments angewendet werden. |
| SVG | Scalable Vector Graphics, in XML formuliertes Grafikformat, dessen Darstellung mit mathematischen Methoden errechnet wird. |
| Tags | Textstruktur zur Auszeichnung eines Elements in Webdokumenten; ein Tag steht in spitzen Klammern, ein Element beginnt mit einem öffnenden und endet mit einem schließenden Tag (zum Beispiel `<b>gesperrt</b>`) |
| Toolkit | Sammlung von Programmen für einen bestimmten Aufgabenbereich |
| Webdokumente | Im Web veröffentlichte Dokumente, die überlicherweise mit einem Browser dargestellt werden; Webdokumente wurden ursprünglich in HTML geschrieben, mittlerweile allerdings zum Teil auch in XML. |

| | |
|---|---|
| Win32 | Microsoft Windows™ für 32-Bit Architekturen |
| WWW | World Wide Web |
| XML | Extended Markup Language, Sprache zur Definition von Dokumenten |
| XSL | Extended Stylesheet Language, in XML formulierte Sprache zur Erst von Stylesheets |
| XSLT | Teilbereich von XSL, der sich mit der Transformation von XML-Dokumenten beschäftigt |

# Quellen


[Computerwoche 2002] „Empower Germany": IT-Branche macht sich Mut COMPUTERWOCHE 51/52 vom 20.12.2002; *http://www.computerwoche.de/index.cfm?pageid=267&type=ArtikelDetail&id=80110535&cfid=8443404&cftoken=30410550&nr=10*; Computerwoche Verlag GmbH München [gesehen am 01.03.2003]

[CVM] CDC Virtual Machine for the Sharp Zaurus, J2ME Personal Profile Reference Implementation, *http://developer.java.sun.com/developer/restricted/pp4zaurus/*

[Designerhilfe] Hilfedokumentation zum Programm Lotus Designer, Release 6.0 September 26, 2002 (englisch), Unterpunkt „XML and Domino"

[ECIN 2000] XML – der Durchbruch für EDI? Vom 09.03.2000 *http://www.ecin.de/edi/xml* [gesehen am 01.03.2003]

[Familiar] Homepage von „The Familiar Linux Distribution", *http://familiar.handhelds.org*

[HTML] „Hypertext Markup Language (HTML) Home Page", *http://www.w3.org/MarkUp*

[Kao] Kao, O.: Grundlagen der verteilten Systeme, Folienskriptum zur Vorlesung „Verteilte Systeme" im Wintersemester 2002/2003

[Nastansky 95] Nastansky, L.: Message-Objekte und Team-Kommunikation. In: Fischer, et all. : Bausteine der Wirtschaftsinformatik, 1. Auflage

[OPIE] Homepage des Projekts „Open Personal Information Environment", *http://www.opie.info*

[PersonalProfile] Informationsseite J2ME Personal Profile, *http://java.sun.com/products/personalprofile*

[Sailer] Sailer, M.: „Anforderungen, Entwicklung und Trends im Bereich Enterprise Application Integration (EAI)", SerCon Gmbh, *http://www.competence-site.de/eaisysteme.nsf/D544A36C8611C335C1256A5A00*

*472B5C/$File/text_sailer_sercon.pdf* [gesehen am 11.03.2003]

[Suhl et al. 2001] Lena Suhl, Stephan Kassanke, Michael Scholz: Web Based Systems, Universität Paderborn, Wintersemester 2001/2002

[ToolkitReadme] README zum Lotus XML Toolkit, im Installationsverzeichnis des Toolkits unter *./readme.txt*

[ToolkitURL] Lotus Developer Domain – Sandbox – DXL Resources, *http://www.lotus.com/xmltoolkit*

[ToolkitHilfe] Hilfedokumentation im Lotus XML Toolkit, im Installationsverzeichnis des Toolkits unter *./LotusXML/doc/DXLTools10_html_index.htm*

[W3C] Homepage des World Wide Web Consortium, *http://www.w3.org*

[WebHistory] „A little History Of The World Wide Web", *http://www.w3.org/History.html*

[WebStatistik] „Wem gehört das Web? Statistiken und Schlussfolgerungen", *http://www.hrz.uni-wuppertal.de/infos/hrz-info/hrz-info-2002/node17.html*

[XML] „Extensible Markup Language", *http://www.w3.org/XML*

[XSL] „The Extensible Stylesheet Language (XSL)", *http://www.w3.org/Style/XSL*