Föderierung betrieblicher Anwendungssysteme auf der Grundlage Web-basierter Dienste

Martin Gaedke[*], Klaus Turowski[+] und Jörn Rehse[#]

[*] *Telecooperation Office (TecO), Universität Karlsruhe*
Vincenz-Prießnitz Str. 1, 76131 Karlsruhe, Germany
Tel.: +49 (721) 69 02-79, FAX: -16, e-mail: gaedke@teco.edu

[+] *Otto-von-Guericke-Universität Magdeburg, Arbeitsgruppe Wirtschaftsinformatik*
Postfach 4120, 39016 Magdeburg, Germany
Tel.: +49 (391) 67 1-83 86, FAX: -12 16, e-mail: turowski@iti.cs.uni-magdeburg.de

[#] *Augartenstraße 58, 76137 Karlsruhe, Germany*
e-mail: rehse@websupport.de

Abstract. Das World Wide Web (Web) ist in einem Wandel vom einem Informationsmedium zu einer Plattform für verteilte (betriebliche) Anwendungen begriffen. Mehr und mehr Unternehmen gehen dazu über Waren und Dienstleistungen über das Web anzubieten. Darüber hinaus nutzen sie das Web als Mittel, um ihre oft heterogenen betrieblichen Anwendungssysteme zu föderieren. Das Web wird so gleichermaßen als Vertriebskanal und Integrationsplattform genutzt. Dementsprechend ist auch der Begriff *Produkt* überladen. So kann mit Produkt einerseits das hergestellte Ergebnis einer Unternehmung gemeint sein, oder andererseits, aus dem Blickwinkel der Softwaretechnik, das Web-basierte Ergebnis der Föderierung von Anwendungssystemen verschiedener Unternehmen. Um ein derartiges Produkt zu erstellen, ist es notwendig, Dienste, die von verschiedenen (betrieblichen) Anwendungssystemen angeboten werden, zu einem neuen, Web-basierten Dienst, dem Produkt, zusammenzustellen. Dazu schlagen wir die Verwendung einer generischen Föderierungsschicht vor und beschreiben deren Realisierung unter Nutzung der WebComposition Markup Language (WCML). Um darüber hinaus eine Kommunikation zwischen den beteiligten Anwendungssystemen zu unterstützen, welche die betriebswirtschaftliche Semantik bewahrt, ergänzen wir die generische Föderierungsschicht um eine statische Föderierungsschicht.

Keywords: E-Commerce; Web Engineering; XML; EDI; Component-Based Web-Application

1 Einleitung

Einhergehend mit seinem unerwarteten Wachstum seit Beginn seines Bestehens wandelt sich das World Wide Web (WWW, Web) von einem dezentralen Informationsmedium in eine Plattform für komplexe, verteilte Anwendungen. Für seine ursprüngliche Aufgabe, die Veröffentlichung von Informationen im wissenschaftlichen Bereich (Forschungsberichte, wissenschaftliche Arbeiten, Kontaktinformationen etc.) war das World Wide Web geschaffen und geeignet. Die inzwischen im Web realisierten Dienste sind jedoch umfangreiche Anwendungen, unter anderem zur Aus- und Weiterbildung, zur Unterhaltung und aus dem kommerziellen Bereich. Hierbei sind insbesondere Anwendungen, die Geschäftsprozesse über die Grenzen eines Unternehmens hinweg unterstützen sowie der „elektronische Handel“ (engl. *Electronic Commerce, e-commerce*) zu nennen.
E-Commerce wird allgemein definiert als „jede Form von geschäftlicher Transaktion in dem die teilnehmenden Parteien elektronisch anstatt durch physikalischen Austausch *(physical exchange)* oder direkten physikalischen Kontakt *(physical contact)* interagieren“ (ECOM 1998, S. 2). Die einfachste Form des E-Commerce ist die Präsenz eines Unternehmens im World Wide Web. Ferner lassen sich E-Commerce Anwendungen in vier Gruppen unterteilen: *business-to-business* (Unternehmen mit Unternehmen), *business-to-consumer* (Unternehmen mit Verbraucher), *business-to-administration* (Unternehmen mit Behörden) und *consumer-to-administration* (Verbraucher mit Behörde).
E-Commerce im business-to-business Bereich gehört schon seit einiger Zeit zum Alltag vieler Unternehmen, während sich die Abwicklung von Geschäftsvorfällen mit Behörden über elektronische Medien noch in den Anfängen befindet. E-Commerce Anwendungen aus dem business-to-consumer Bereich haben durch die Entwicklung des World Wide Web sehr schnell an Bedeutung gewonnen.
Eine besondere Rolle spielen hierbei die E-Commerce Anwendungen, die Dienste aus verschiedenen Abteilungen eines Unternehmens oder mehrerer Unternehmen integrieren (Gaedke et al. 1999). Hierzu zählen Dienste *(Services)* verschiedener Anwendungen (z. B. Produktionsplanungssysteme, Systeme zur Abwicklung von Bestellungen, Lagerverwaltung, etc.) oder Datenbanken. Die Zusammenfassung dieser Dienste kann zu einem neuen Dienst führen. Die dem neuen Dienst zu Grunde liegenden Dienste werden vor dem Dienstnehmer (dem Kunden) versteckt. Erreicht wird dies durch die Einführung von *Föderierungsschichten*, welche die Services und damit die ihnen zu Grunde liegenden Applikationen föderieren.
In diesen Föderierungsschichten werden Ansätze aus dem Bereich des *Web Engineering* eingesetzt. Die Begründung der Disziplin des Web Engineering in den letzten Jahren, erklärt sich aus der Problematik der Erstellung komplexer Anwendungen für das World Wide Web (Web-Anwendungen) und der damit verbundenen Erkenntnis, daß die Entwicklung für das Web gleichbedeutend mit Softwareentwicklung ist (Barta/Schranz 1998), (Coda et al. 1998), (Gellersen/Gaedke 1999). Web Engineering bezeichnet die Anwendung von Softwaretechnik für Entwicklung, Betrieb und Wartung von Anwendungen im World Wide Web.
In der Praxis erfolgt die Entwicklung für das World Wide Web oftmals nicht nach den Methoden der Softwaretechnik, sondern in einem unkoordinierten ad-hoc Prozeß (Powell/Jones/Cutts 1998). Dies liegt unter anderem daran, daß das Implementierungsmodell des World Wide Web die Umsetzung moderner softwaretechnischer Methoden verhindert. Das Implementierungsmodell des World Wide Web ist immer noch auf seine ursprüngliche Aufgabe als verteiltes, dokumentenorientiertes Informationssystem zugeschnitten. Die Entwicklung komplexer Anwendungen leidet deshalb unter dem einfachen grob-granularen Implementierungsmodell des World Wide Web.
Zur Überbrückung der Lücke zwischen Entwurf und Implementierung für das Web wurde der objektorientierte WebComposition Ansatz eingeführt (Gellersen/Wicke/Gaedke 1997). Neben der gesteigerten semantischen Ausdrucksfähigkeit erlaubt WebComposition die Nutzung der Vorteile der Wiederverwendung durch Komposition von bereits erstellten Softwareartefakten *(Compositional Reuse)* (Frakes et al. 1994). Zur Wiederverwendung durch Komposition (im Gegensatz zur Wiederverwendung durch Generierung) vgl. z. B. (Szyperski 1998).

Die Notwendigkeit der Föderierung von Services (und damit der ihnen zu Grunde liegenden Applikationen) erhöht die Komplexität von Web-Anwendungen noch weiter. Bei der Erstellung von Web-Anwendungen, die mit föderierten Services arbeiten, stellen die Beschränkungen des Implementierungsmodells des World Wide Web also ein besonderes Hindernis dar.
In diesem Artikel stellen wir eine plattformunabhängige Architektur vor, welche die Föderierung von Web-basierten Anwendungen auf der Grundlage des Dienste-Paradigmas durch einen generischen Ansatz unterstützt und die benötigte Flexibilität und Granularität für die Anwendung softwaretechnischer Methoden für die Entwicklung und Evolution solcher Anwendungen bietet.
Um dies zu erreichen besteht die Architektur im Kern aus zwei Schichten. In beiden Schichten wird auf verschiedene Art und Weise XML als Basistechnologie verwendet. Die *generische Föderierungsschicht (Generic Federation Layer, GFL)* basiert auf dem WebComposition Ansatz. Hier kommt insbesondere die WebComposition Implementierungstechnologie WCML *(WebCompositon Markup Language)* zum Einsatz. Bei WCML handelt es sich um eine XML Anwendung. Die GFL unterstützt die Wiederverwendung von Komponenten, erzwingt die strenge Trennung von Inhalt, Layout, Navigation und Geschäftsprozessen[1] und erlaubt die Verwendung von Softwareartefakten beliebiger Granularität in der Implementierung eines Services im Web. Sie unterstützt die Föderierung von aus WebComposition-Komponenten zusammengesetzten Services.
Damit aus verschiedenen Geschäftsanwendungen ein neuer Dienst entstehen kann, müssen auch diese Anwendungen föderiert werden. Dazu dient die *statische Föderierungsschicht (Static Federation Layer, SFL).* Sie bietet Möglichkeiten für den Austausch von Geschäftsdaten zwischen Applikationen *(Electronic Data Interchange, EDI).* Zur Realisierung des Datenaustauschs wird auch hier XML verwendet.
Im folgenden Abschnitt wird die Problematik, welche die Entwicklung des WebComposition Ansatzes motiviert hat, aufgezeigt. Danach werden die Grundlagen von WebComposition, welche die technischen Möglichkeiten für die Implementierung von komplexen Services und ihre Föderierung bilden, vorgestellt. Insbesondere wird die Rolle von WCML, und damit von XML, für die Implementierung von Services erklärt. Im darauf folgenden Abschnitt wird das Service Konzept und die damit einhergehende Trennung von Prozessen, Layout, Navigation und Inhalt präzisiert. Im Anschluß daran wird die Arbeitsweise der GFL veranschaulicht. Im Weiteren erklären wir die Funktionsweise der SFL und wie XML bei der Kommunikation mit betrieblichen Anwendungssystemen und ihrer Föderierung verwendet werden kann. Vor den Schlußfolgerungen und dem abschließenden Ausblick auf unsere weitere Arbeit präsentieren wir eine komplexe E-Commerce Anwendung, in der die hier vorgestellten Ansätze bereits verwirklicht wurden.

2 Komponentenorientierte Softwareentwicklung für das World Wide Web

In diesem Abschnitt wird zunächst erläutert, wie die Bestrebung Ergebnisse aus der Softwaretechnik auf die Entwicklung für das World Wide Web zu übertragen, die Entwicklung des WebComposition Ansatz motiviert hat. Hierbei stehen die Granularitätsproblematik zwischen Entwurfsmodell und Implementierungsmodell sowie die Wiederverwendung von Softwareartefakten (Komponenten) im Vordergrund. Im zweiten Teil dieses Abschnitts wird dann der WebComposition Ansatz selbst und der Nutzen von XML (in Form von WCML) für WebComposition dargelegt.

[1] Dies geht über die im Web Engineering sonst übliche Trennung von Inhalt und Layout hinaus.

2.1 Komponententechnologie für das Web

Für die Anwendung von Softwaretechnik (Sommerville 1982) und insbesondere von komponentenbasierter Softwaretechnik (Sametinger 1997), (Szyperski 1998) *(Component Based Software Engineering, CBSE)* für das Web gibt es zwei Kernargumente. Diese sind die Erleichterung der Wiederverwendung und die Möglichkeit, feingranulare Konzepte in der Implementierung zu repräsentieren. Die Erleichterung der Wiederverwendung ist auch in der herkömmlichen Softwaretechnik einer der Hauptbeweggründe für die Einführung von CBSE. Das Granularitätsproblem jedoch, also die Überführung eines feingranularen Entwurfs auf das grobgranulare Implementierungsmodell des WWW, ist eine Problematik, die inhärent durch das Implementierungsmodell des Web bestimmt wird.
In der Softwaretechnik existieren verschiedene Arten der Wiederverwendung. Neben der Wiederverwendung von Code gibt es die Wiederverwendung von wiederkehrenden Entwurfsmustern *(Patterns)* und Konzepten (Gamma 1995). Im Bereich der Wiederverwendung von Code unterscheidet man die Wiederverwendung durch Komposition und durch Generierung. Im ersten Fall werden Anwendungen durch das Zusammenfügen von existierenden Komponenten erzeugt. Im zweiten Fall handelt es sich um parametrisierbare Applikationsrahmen oder Codegeneratoren. Obwohl die Softwaretechnik viele solche Verfahren, Modelle und Methoden hervorgebracht hat (Sommerville 1982), finden sie im Web nur selten Anwendung.
In der Praxis der Softwareentwicklung für das Web finden zur Zeit Laufzeitumgebungen Anwendung, welche die Wiederverwendung von HTML (Hypertext Markup Language) Fragmenten zumeist durch dynamische Generierung von Ressourcen aus Datenbanken oder andere Techniken, die auf dynamische Daten basieren, in den Vordergrund stellen. Typische Vertreter solcher Unterstützungssysteme sind beispielsweise ASP (Microsoft 1997b), Server-Side Includes (NCSA 1995), FrontPage (Microsoft 1997a) und NetObjects (NetObjects 1997). Hierbei wird die Ressourcengranularität nur in der datenorientierten Sicht erhöht. Eine darüber hinausgehende Betrachtung wesentlicher Merkmale zur Implementierung von Anwendungen wie z. B. Transaktionsbedingungen, Geschäftsprozesse, Föderierung bestehender Anwendungssysteme, etc. wird vernachlässigt. Es wird im besten Fall eine Trennung von Inhalt und Layout durchgeführt. Geschäftsprozesse werden nicht gesondert behandelt. Oft werden bei serverseitigen Skripttechniken sogar Inhalt, Layout und Prozesse (durch Skriptcode modelliert) in einer einzigen Ressource zusammengefaßt.
In neuerer Zeit wird der systematische Einsatz von Techniken zur Wiederverwendung durch komponentenorientierte Softwareentwicklung diskutiert, vgl. z. B. (Sametinger 1997), (Szyperski 1998). Komponentenorientierte Softwareentwicklung verfolgt dabei das Ziel Anwendungssysteme durch wahlfreie Kombination oder Adaption von wiederverwendbaren Softwarebausteinen (Komponenten) zu ermöglichen. Eine Komponente ist ein wiederverwendbarer, abgeschlossener und vermarktbarer Softwarebaustein, der Dienste über eine wohldefinierte Schnittstelle zu Verfügung stellt und in zur Zeit der Entwicklung unvorhersehbaren Kombinationen mit anderen Komponenten einsetzbar ist (Turowski 1999, S. 4)
Von der systematischen Wiederverwendung mit Hilfe von Komponenten ist eine Reduktion des Aufwands bei der Entwicklung von Anwendungen bei gleichzeitiger Steigerung der Qualität zu erwarten (Lim 1998). Für die Ausnutzung dieser Vorteile ist die Systematisierung der Wiederverwendung allerdings Voraussetzung. Hierzu muß ein adäquater technischer Rahmen vorliegen, der die Entwicklung mit Komponenten unterstützt. Dies war ein Grund für die Entwicklung der WebComposition Markup Language als Grundlage für die Repräsentation von Komponenten und ihren Schnittstellen für das Web, um so eine Basis für die systematische Wiederverwendung durch Komposition zu schaffen.
Ein weiterer Grund für die Entwicklung des WebComposition Ansatzes war die Problematik der Granularitätslücke zwischen den Entwurfsmodellen und dem Implementierugsmodell im Web. Die

dokumentenorientierte (Datei-basierte) Implementierung für das Web verhindert die Modellierung von feingranularen Konzepten im Implementierungscode. Die Information, welcher Teil einer Implementierungsressource zum Beispiel ein Objekt der Benutzerschnittstelle, ein Dialog, Teil eines Geschäftsprozesses oder nur ein grafisches „*Corporate Identity*“ Element ist, läßt sich nicht in einer solchen Ressource repräsentieren. Feingranulare Modellierungseinheiten und Konzepte gehen daher bei der Abbildung auf die grobgranularen Implementierungsressourcen verloren. Bei der Abbildung des Entwurfs auf die Zielressourcen geht die Entwurfsinformation verloren, da sie sich nicht im Implementierungsmodell darstellen läßt. Konzepte, die während des Entwurfs eine Einheit gebildet haben, müssen zum Beispiel oft in mehrere Zielressourcen kopiert werden, während unabhängige Konzepte in einer einzigen Ressource zusammengefaßt werden müssen. Ein typisches Beispiel ist eine Navigationsleiste, die auf mehreren HTML-Seiten vorkommen soll. Kopiert man den Code, so ist die Information, daß es sich auf jeder Seite um die selbe Leiste handelt, aus der Implementierung nicht mehr zu erkennen.
Besonders bei der Wartung im evolutionären Lebenszyklus einer Web-Anwendung führt dies zu großen Problemen, da die den Entwurfsentscheidungen zu Grunde liegenden Konzepte in der Implementierung nicht mehr zu erkennen sind (Gaedke 1998). Das Zusammenschmelzen und Klonen von konzeptuell getrennten Einheiten bei der Abbildung auf das Implementierungsmodell stellt auch ein Problem für die Wiederverwendung solcher Einheiten dar, da sie in der Implementierungsressource nicht immer als zusammenhängendes Fragment auftauchen.

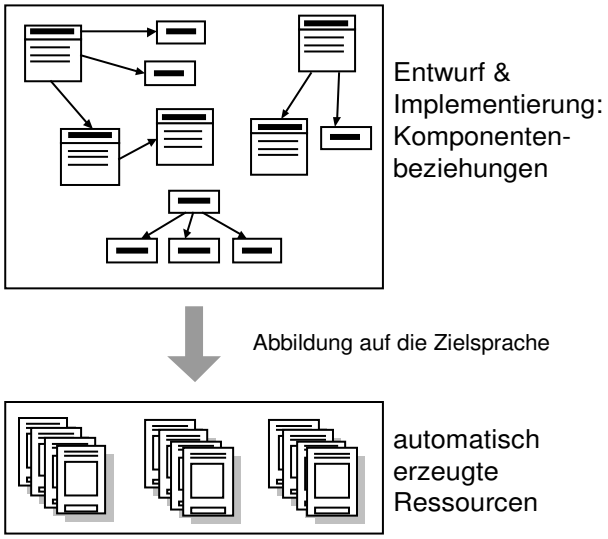

Abbildung 1: Zusammenhang zwischen Modell und Ressourcen

Zwar wurden Methoden entwickelt, welche die Abbildung abstrakter Konzepte und feingranularer Entwurfseinheiten auf das Web erlauben (OOHDM (Schwabe/Rossi/Barbosa 1996), RMM (Isakowitz/Stohr/Balasubramaninan 1995), JESSICA (Barta/Schranz 1998), TML (Kristensen 1998)), dennoch können auch sie die Probleme, die sich durch den Informationsverlust bei der Abbildung auf das Implementierungsmodell des Webs ergeben, nur bedingt beheben.
Im folgenden Abschnitt wird ein komponentenbasierter Ansatz für das Web Engineering vorgestellt, der die Modellierung und die Implementierung von Web-Anwendungen mit feingranularen Strukturen erlaubt. Der Ansatz unterstützt die Wiederverwendung durch Komposition, da die Komponenten nach ihrer Implementierung erhalten bleiben und so für die Evolution oder die Entwicklung weiterer Web-Anwendungen genutzt werden können.

2.2 WebComposition

Der WebComposition Ansatz basiert auf dem Zusammensetzen von Web-Anwendungen aus Komponenten. WebComposition ist die Grundlage der GFL, welche die Föderierung von Services unterstützt. Komponenten im WebComposition Ansatz werden hierbei als eine Codeabstraktion einer beliebigen Zielsprache verstanden, z. B. HTML, Skriptcode, VRML (*Virtual Reality Markup Language*), WML (*Wireless Markup Language*) oder auch LaTeX Quellcode oder PDF. Die Granularität einer WebComposition Komponente ist ausschließlich von der Art eines Entwurfsartefakts abhängig. Eine Komponente kann z. B. einfache HTML-Elementeigenschaften, wie die Größe einer zu benutzenden Schrift, aber auch komplexe Aufgaben, wie betriebliche Abläufe oder Umsetzungen von Entwurfsmustern, beschreiben oder eine Komposition existierender Komponenten sein. Web-Anwendungen werden im WebComposition Ansatz als Komposition feingranularer Komponenten beschrieben. Die Evolution von Web-Anwendungen vollzieht sich durch die Manipulation oder durch das Hinzufügen von Komponenten, die während ihres gesamten Lebenszyklus persistent und zugreifbar in einem Komponentenspeicher, dem *Component Store*, gehalten werden.

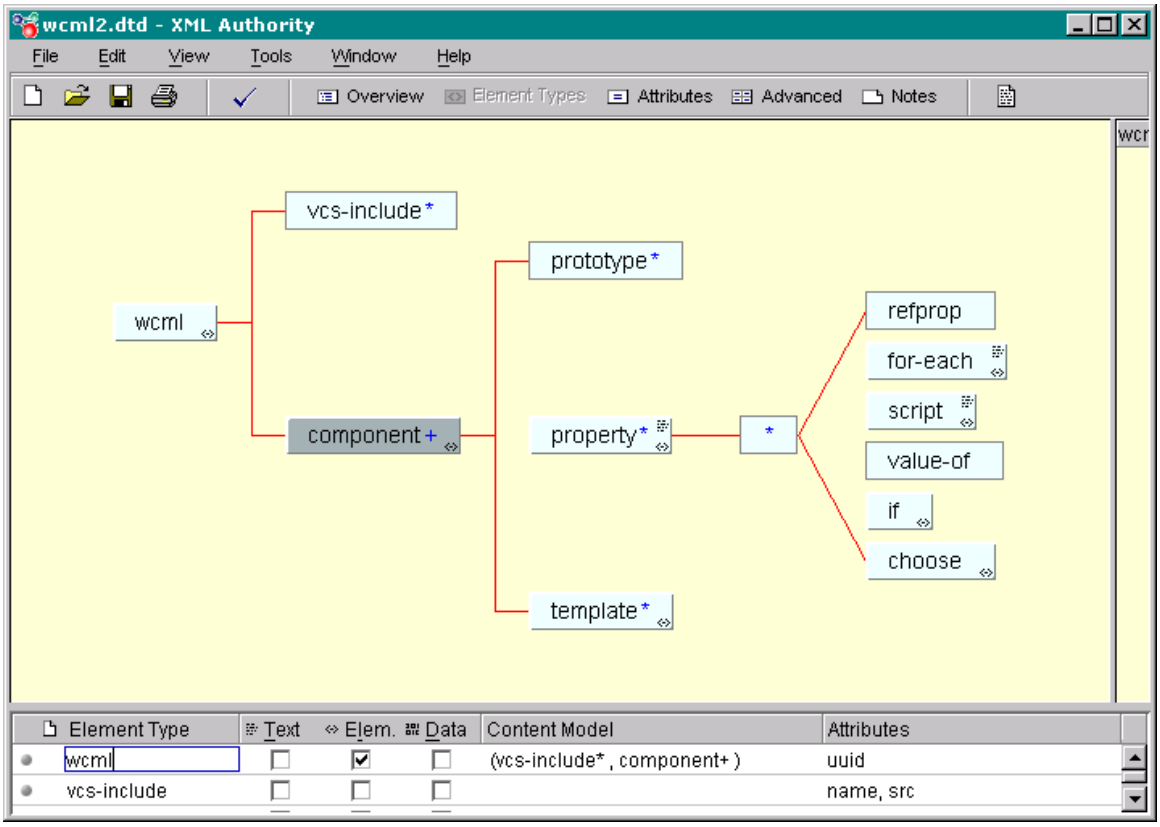

Abbildung 2: Die WCML DTD

Der WebComposition Ansatz ist objektorientiert. Die Erzeugung neuer Komponenten kann daher durch die Wiederverwendung existierender Komponenten erleichtert werden. Hierzu werden grundlegende Merkmale von Objektorientierung, wie Vererbung, Aggregation und Polymorphie angeboten. Im Gegensatz zu den klassenbasierten Ansätzen, wie z. B. bei Java und C++, verwendet der WebComposition Ansatz die Prototyp-Instanz-Vererbung (Ungar/Smith 1987). Die Erzeugung von Exemplaren aus Prototypen und ihrer Modifikation durch Überladen von Eigenschaften ist eine

Technik zur Wiederverwendung von Code. Jede definierte Komponente kann im Modell als Prototyp dienen und referenziert werden, um so die Wiederverwendung bei der Beschreibung neuer Komponenten zu ermöglichen.
Der WebComposition Ansatz begegnet den im letzten Abschnitt genannten Problemen bei der Wiederverwendung für das Web und der Abbildung von feingranularen Entwurfseinheiten auf ein dokumentenbasiertes Implementierungsmodell, indem es die Implementierung einer Web-Anwendung auf einer feingranularen Ebene zuläßt und die Arbeit mit Komponenten unterstützt.
In Abbildung 1 ist die Abbildung von Komponenten auf Ressourcen dargestellt. Der Abbildungsmechanismus, der durch den sog. Ressourcen Generator zur Verfügung gestellt wird, nutzt dabei die Gesamtheit aller Komponenten, die für die Erstellung der Zielressourcen notwendig sind. Dadurch entfällt die Implementierung direkt auf der Ebene einzelner Ressourcen.

2.3 Die WebComposition Markup Language

Die WebComposition Markup Language (WCML) ist eine Anwendung der eXtensible Markup Language (XML) (Gaedke/Schempf/Gellersen 1999). Die Auszeichnungsssprache XML ermöglicht die Beschreibung einer Auszeichnungssprache durch die Angabe einer *Document Type Definition* (DTD). Für WCML existiert eine solche DTD. Eine andere XML Anwendung ist zum Beispiel XSL (World Wide Web Consortium 1999). Daten, die konform zu einer gegebenen DTD vorliegen, heißen gültiges (*valid*) XML. Entsprechen Daten nur den allgemein in XML festgelegten Strukturregeln, aber wurden sie nicht mit einer bestimmten DTD assoziiert, so spricht man von wohlgeformten XML (*well-formed*). Der Vorteil, der durch den Einsatz von XML erzielt wird, ist das man Daten mit Metainformationen (den XML-Elementen) versehen kann und darüber hinaus beliebige XML-Daten auf ihre Syntax bezüglich gegebener DTD überprüfen kann. Verschiedene XML Parser, die zusammen mit beliebigen DTDs beliebige XML Dokumente auf ihre Gültigkeit überprüfen können, gibt es von mehreren Herstellern, z.B. XML4Java von IBM, XMLParser von Sun oder der MSXML Parser von Microsoft.

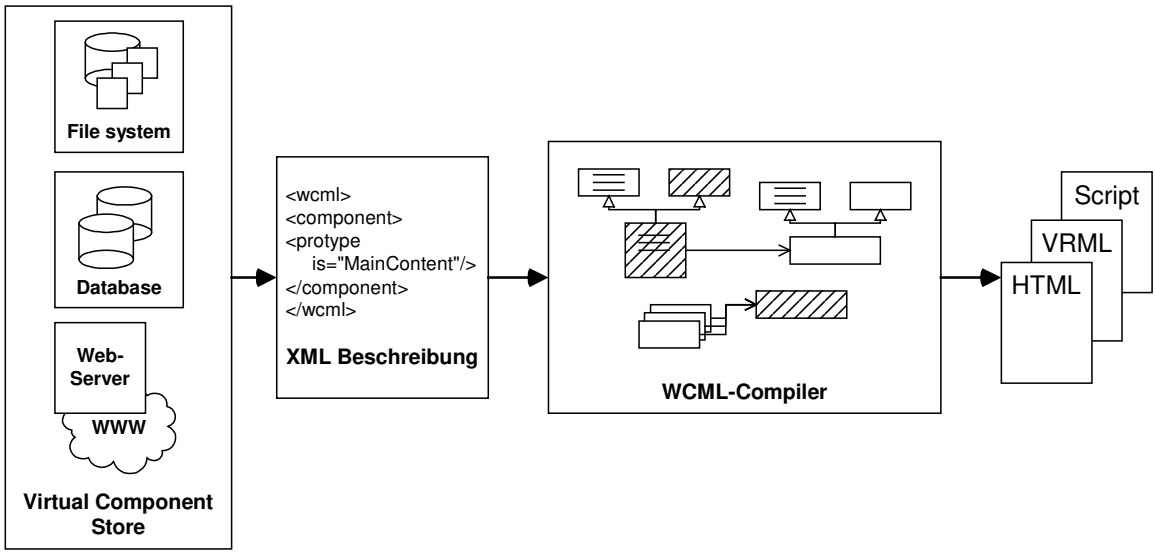

Abbildung 3: Einsatz des WCML-Compilers

Die WCML DTD, die in Abbildung 2 dargestellt ist, beschreibt die Syntax für WebComposition Komponenten. Es ist evident, daß WCML die positiven Eigenschaften von XML “erbt”. WCML ist

plattformunabhängig, einfach auszuwerten. Darüber hinaus unterstützen die meisten Standardparser strenge Syntaxprüfung anhand von DTDs.
Im WebComposition Ansatz werden Komponenten in einer Datenbank, dem *Component Store* verwaltet. Da die Komponenten in Form von Beschreibungscode (XML) vorliegen, können Datenbanken, aber auch Dateien oder Web-Server als Component Store für Komponenten dienen; daher werden die Komponentenspeicher auch als *Virtual Component Stores* bezeichnet. Durch die URI-basierte Adressierung von virtuellen Component Stores können in WCML auch Komponenten anderer Component Stores wiederverwendet werden. Die Abbildung 3 verdeutlicht die Abbildung von Komponenten in eine Zielressource.
Der WCML-Compiler verwendet eine von einem Dritthersteller entwickelte XML Parserkomponente. Diese Komponente liest die vom Virtual Component Store gelieferte WCML Komponenten, erzeugt einen Parsebaum und übernimmt die Syntaxprüfung anhand der WCML DTD. Der Compiler übernimmt dann die Ausführung der Präsentationsmethoden der WCML Komponenten und die Auflösung von Referenzen und Vererbungsbeziehungen zwischen Komponenten. Die Komponenten, die vom Component Store ausgeliefert werden, werden vom WCML Compiler auf das Implementierungsmodell des World Wide Web (bzw. auf eine andere Zielsprache) abgebildet. Die Abbildung verdeutlicht auch die Möglichkeit der Nutzung eines Virtual Component Stores unter Verwendung eines Webservers, der Komponenten durch das Netz zur Verfügung stellt.
Die Implementierungstechnologie WCML und der WebComposition Ansatz dienen der Konstruktion des GFL, welche die Föderierung von Services im Web erlaubt. Die Verwendung von WCML für die Föderierung von Services wird im nächsten Abschnitt erläutert.

3 Service-Basierte Föderierung im Web

In diesem Abschnitt wird zuerst das Service Konzept und die Funktionsweise der GFL erklärt. Eine besondere Rolle spielt hierbei die Verwendung der im letzten Abschnitt vorgestellten WebComposition Konzepte und die XML-basierte Sprache WCML. In der GFL wird die Trennung von Layout, Inhalt, Navigation und der Implementierung der (Geschäfts-)Prozesse durch das Service Konzept sichergestellt. Im zweiten Teil dieses Abschnitts wird die statische Föderierungsschicht (SFL) vorgestellt. Auch sie verwendet XML, um Föderierungsaufgaben durchzuführen. Sie ermöglicht die Anbindung von (heterogenen) betrieblichen Anwendungen unter Beibehaltung der jeweiligen betriebswirtschaftlichen Semantik.

3.1 Das Service Konzept

Betrachtet man die über Web-Technologie (sei es im Internet oder in einem Intranet) zugänglichen Geschäftsanwendungen, so dient die Anwendung in den meisten Fällen dem Benutzer dadurch, daß sie es ihm erlaubt über das Netz Produkte zu bestellen oder Informationen abzurufen. Man kann von den Angeboten im Web abstrahieren und die Tätigkeit von Web-basierten Anwendungen, als das zur Verfügung stellen einer oder mehrerer Dienstleistungen bezeichnen. Jedes Angebot oder jede Information kann als Service verstanden werden, den das Betreiberunternehmen als Dienstleistung anbietet (vgl. **Abbildung 4**). Die Kombination solcher Services ist wiederum eine Dienstleistung bzw. ein Service des Unternehmens. Ein Service als Modellierungseinheit beinhaltet den textuellen oder grafischen Inhalt, d.h. die Information, die das Unternehmen mit dem Service assoziiert. Er beinhaltet außerdem das Layout und die Navigation mit dem diese Informationen zur Verfügung gestellt werden sollen, sowie Anweisungen darüber, wie mit dem Benutzer interagiert werden soll, d.h. welche Folgen seine Aktionen haben. Die Implementierung der (Geschäfts-)Prozesse ist ein weiterer wichtiger Bestandteil, d.h. die technische Beschreibung die Interaktion mit den Geschäftsanwendungen, z. B. ERP-Systemen (*Enterprise Resource Planning Systems*). Im Sinne der Softwaretechnik kann ein

Service durch ein Objekt repräsentiert werden, dessen innerer Zustand vom Benutzer verändert werden kann. Aussehen und Benutzerschnittstelle des Service werden während der Laufzeit durch Web-Seiten bestimmt. Die Definition des Ablaufs erfolgt in Prozesskomponenten im Back-End. Bei dem Benutzer eines Services braucht es sich nicht unbedingt um einen Menschen handeln. Durch die Trennung von Daten, Präsentation und Prozeß, kann ein Service auch anderen Dienstnehmern, z. B. anderen Web-Servern oder Geräten, maschinen-verständlich zugänglich gemacht werden.
Mit WCML kann ein Service aus den folgenden Komponenten zusammengesetzt werden:

- *Service Content:* Eine WCML Komponente, welche die Eigenschaften (u. a. den textuellen Inhalt) des Service beschreiben.
- *Service Layout und Navigation:* WCML Komponenten die definieren, mit welchem Layout der Service angezeigt werden soll und wie die Informationen dem Benutzer zugänglich gemacht werden sollen, also wie z. B. durch die Seiten navigiert werden kann. Es kann sich beispielsweise um eine Komponente handeln, die das Web-Entwurfsmuster *Decorator* implementiert, um den Service entsprechend der Corporate Identity des Unternehmens anzuzeigen. Darüber hinaus kann das Web-Entwurfsmuster *Guided Tour* zur „geführten“ Navigation über die Seiten zum Einsatz kommen.
- *Service Processing:* Eine Folge von Aufrufen an Komponenten zur Weiterverarbeitung in den Geschäftsanwendungen. Diese Aufrufe spiegeln den automatisierten Teil des Geschäftsprozesses wieder. Sie verwenden die statische Föderierungsschicht, um auf (Legacy-)Anwendungssysteme zuzugreifen.

Ein Service ist selbst eine Komponente, da er aus anderen Komponenten zusammengesetzt wird. Die Komponente kann als Prototyp für neue Komponenten dienen und somit wiederverwendet werden. Applikationen, die auf dem Service orientierten Modell mit WCML basieren, können föderiert werden, in dem unabhängige, vorhandene Services in neuen WCML Komponenten zu neuen Services zusammengeführt werden.

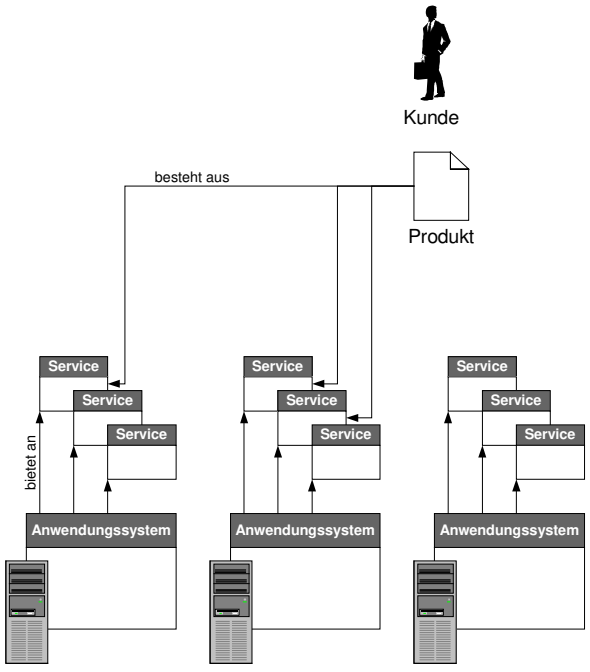

Abbildung 4: Zusammenhang zwischen Produkt und Dienst (Service)

Das folgende Beispiel verdeutlicht die Definition von Services mit WCML. Eine Servicekomponente soll ein Produkt (hier ein Mobiltelefon) zum Verkauf anbieten. Der Service muß das Telefon beschreiben (Content), die Information im Browser darstellen (Layout und Navigation) und den Einkaufsprozeß für das Telefon definieren (Process).
Jede Komponente ist in WCML durch einen eindeutigen Namen (UUID) identifiziert. Die Beschreibung des Service zum Verkauf von Mobiltelefonen (SellMobilePhone) basiert dabei auf einer Komponente zur Beschreibung von Standardprodukten. Die Komponente StandardProduct definiert durch die sogenannten *Properties* die Eigenschaften, die alle Produkte gemeinsam haben. Ein Property kann dabei ein Attribut oder eine parameterlose Funktion der Komponente darstellen.

```
<?xml version='1.0' ?>
<!DOCTYPE wcml SYSTEM "wcml2.dtd">
<wcml>
<COMPONENT uuid='StandardProduct'>
      <PROPERTY name='currency'>EUR</PROPERTY>
      <PROPERTY name='unitName'>piece</PROPERTY>
      <PROPERTY name='unitNamePlural'>pieces</PROPERTY>
      ...
</COMPONENT>
...
```

Diese Komponente kann jetzt von der Content Komponente des SellMobilePhone Services als Prototyp verwendet werden, um alle für das Mobiltelefon relevanten Daten hinzuzufügen.

```
...
<COMPONENT uuid='CellphoneModel123'>
      <PROTOTYPE is='StandardProduct'/>
      <PROPERTY name='bandInfo'>dual</PROPERTY>
      <PROPERTY name='price'>229.99</PROPERTY>
      ...
</COMPONENT>
...
```

Durch das Erben von StandardProduct übernimmt die Komponente CellphoneModel123 die Eigenschaften des Prototyps. Dadurch wird klar, daß die Preisangabe 229.99 in Euro (EUR) angegeben ist. Die Präsentations- und Prozeßkomponenten werden ebenfalls in WCML definiert. Nun bleibt nur noch die Zusammenführung in der endgültigen Service Komponente. Hierzu wird die Möglichkeit der geordneten Mehrfachvererbung *(Ordered Multiple-Inheritance)* von WCML verwendet.

```
...
<COMPONENT uuid='SellMobilePhone'>
      <PROTOTYPE is='CellphoneModel123'/>
      <PROTOTYPE is='Layout'/>
      <PROTOTYPE is='Navigation'/>
      <PROTOTYPE is='OrderProcess'/>
</COMPONENT>
...
</wcml>
```

Durch Austausch der Content Komponente (CellphoneModel123) könnte der gleiche Service zum Verkauf anderer Mobiltelefone verwendet werden. Ändert man die Prozeßkomponente, so kann der zugrundeliegende Geschäftsprozeß verändert werden, um zum Beispiel das Telefon in einer „Bundling"-Aktion anzubieten. Durch das Ändern der Layout- und Navigationskomponente kann die Darstellung des Services verändert werden.
Neue Services, die sich von bestehenden Services nur durch ihren Inhalt unterscheiden, können ihrerseits durch einen Service erzeugt werden, der beispielsweise in einem Formular im Web-Browser die Eigenschaften (Properties) des neuen Services abfragt. Services, die neue Services aus bestehenden Services erzeugen heißen Service-Fabriken (*Service-Factories*). Die Instanziierung durch dedizierte Objekte (Services) ist ein häufig angewandtes Verfahren, daß auch als erzeugendes Entwurfsmuster (*Creational Design Pattern*) bekannt ist (Gamma 1995). Bei der Verwendung eines solchen Services kann die Wiederverwendung ohne weitere WCML-Programmierung erfolgen. Es reicht die Verwendung des Formulars. Dies ist aber nicht der einzige Vorteil beim Einsatz einer Service-Fabrik. Das Entwurfsmuster macht ein System unabhängig von der Art und Weise wie Objekte auf einer gewissen Plattform erzeugt werden (Kapselung von plattformabhängigen Eigenschaften). Service-Fabriken sind ein wichtiges Element für die Evolution von Web-Anwendungen, da sie die Wiederverwendung von Services durch nicht technisch geschultes Personal erlauben, Kosten reduzieren und die mittlere Qualität der Services erhöhen.
Im folgenden Beispiel wird anhand eines Beispiels erläutert, wie die objektorientierten Eigenschaften von WebComposition Komponenten die Föderierung von Services (Web-Anwendungen) unterstützen. Hierbei entstehen aus der Kombination der Services neue Services und damit neue Web-Anwendungen.

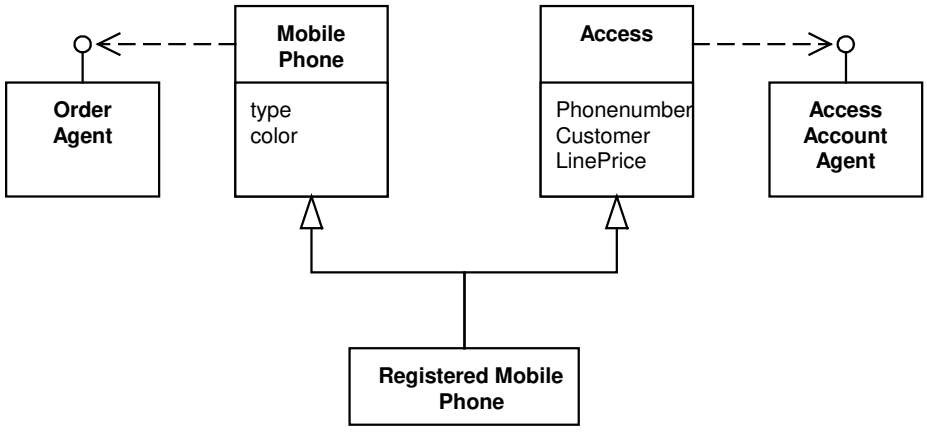

Abbildung 5: Föderierung von Services mit Hilfe der Mehrfachvererbung

Im Beispielszenario wird von den Leistungen zweier Unternehmen unter Nutzung des Service-Ansatzes abstrahiert. Anschließend werden diese zu einem neuen Service (bzw. spezialisierten Produkt) föderiert. Das eine Unternehmen bietet mit einem Service Mobiltelefone im Web zum Verkauf an. Das andere Unternehmen ist ein Netzprovider, das einen Service betreibt mit dem Verträge über den Netzzugang im Internet abgeschlossen werden können. Die Verarbeitung der Kundendaten geschieht durch Geschäftsanwendungen in den jeweiligen Firmen.
Soll nun ein dritter Service im Web angeboten werden, in dem Mobiltelefone und Verträge (d. h. mit Karte) als Paket angeboten werden, so kann dies durch Wiederverwendung der bestehenden Services durch Vererbung geschehen. Der neue Service *Registered Mobile Phone* (siehe Abb. 5) ist eine Web-Anwendung, die durch die Ausnutzung der objektorientierten Eigenschaften (hier die

Mehrfachvererbung) von WCML mit nur minimalen zusätzlichen Codierungsaufwand entstanden ist. Die Menge der Komponenten des neuen Services konstituiert in diesem Fall die generische Föderierungsschicht.
Angepaßt werden müssen im neuen Service nur noch die Navigation und das Layout. Üblicherweise bietet der Mobilfunk Service-Provider solche Paketangebote an. Es könnte deshalb die Corporate Identity Komponente verwendet werden, die auch im *Access* Service verwendet wird. Insgesamt bleibt festzustellen, daß ein neuer Service durch die Wiederverwendung existierender Komponentenbeschreibungen aus den entsprechenden WCML Component Stores erzeugt werden kann.
Die Verarbeitung der bei Verwendung der Services anfallenden Geschäftsdaten erfolgt in den Geschäftsanwendungen der Unternehmen. Hierbei kann es sich um heterogene Umgebungen mit einer Reihe von Legacy-Systemen handeln. Diese Systeme sind in der Abbildung 5 durch den *Order Agent* und den *Access Account Agent* repräsentiert. Neben der Föderierung von Services ist hier die Föderierung der Geschäftsanwendungen notwendig. Die Kommunikation mit diesen Anwendungen muß mit einem einfachen, standardisierten und verteilten Mechanismus erfolgen, damit die auf den Mechanismus aufsetzenden Services wiederverwendbar sind. Insbesondere kann für den Geschäftsvorfall eine Business-To-Business Kommunikation zwischen den Anwendungen beider Unternehmen notwendig werden. Im nächsten Abschnitt wird hierzu die statische Föderierungsschicht (SFL) eingeführt, die verbreitete Standards für die Kommunikation zwischen Geschäftsanwendungen zur einfachen Föderierung verwendet. Sie ermöglicht die Kommunikation zwischen verteilten, heterogenen Geschäftsanwendungen und den Services der generischen Föderierungsschicht. Hierbei werden bereits vorhandene Techniken für den elektronischen Datenaustausch (*Electronic Data Interchange, EDI*) zusammen mit XML zum Einsatz gebracht, welche die betriebswirtschaftliche Semantik der auszutauschenden Geschäftsdaten bewahren. Die Umsetzung der Anwendungsdaten in ein allgemeines Austauschformat wird durch einen verteilten, gemeinsamen Speicher, der nach dem „Tuplespace“-Prinzip arbeitet, ermöglicht.

3.2 Die statische Föderierungsschicht: Anwendungsföderierung und zwischenbetriebliche Integration

Auch die Föderierung von Geschäftsanwendungen, sowie die dafür notwendige Kommunikation zwischen den Geschäftsanwendungen verschiedener Unternehmen, läßt sich am „Mobiltelefon-Beispiel“ darstellen. Im folgenden Abschnitt wird gezeigt, wie die statische Föderierungsschicht der vorgestellten Architektur diese Aufgaben löst.
In obigem Beispiel werden zwei verschiedene Services durch Vererbung zu einem neuen Service kombiniert. Die ursprünglichen Services werden im Hintergrund durch die Geschäftsanwendungen der beteiligten Unternehmen erbracht. Die neu entstandene, im World Wide Web zugängliche Dienstleistung besteht aus den Dienstleistungen *Erwerb eines Mobiltelefons* und *Abschluß eines Vertrages mit einem Provider*. Zwischen den beiden unterschiedlichen Erbringern dieser Teildienstleistungen finden zur Erbringung der kombinierten Dienstleistung Kommunikationsvorgänge statt, an denen der Kunde nicht beteiligt ist. Dieser Informationsfluß dient unter anderem der Organisation der physischen Auslieferung des Produkts des föderierten Dienstes (dem Mobiltelefon mit gültiger Karte), da Karte und Telefon zur gleichen Zeit ausgeliefert werden sollen. Die Tatsache, daß der neue Service aus zwei einzelnen Services besteht, soll vor dem Kunden versteckt werden. Ebenso müssen zwischen den Geschäftspartnern Abrechnungsdaten ausgetauscht werden, da ein Teil der Kosten für das Gerät üblicherweise vom Provider übernommen wird.
Der Geschäftsprozeß hat sich also im Vergleich zu den ursprünglichen Dienstleistungen verändert. Eine Business-To-Business Kommunikation ist hinzu gekommen. Geschäftsprozesse werden als fachliche Komponenten (Fachkomponenten) realisiert, die nun wiederum auf die Funktionen der

statischen Föderierungsschicht für die Kommunikation von Geschäftsapplikationen zurückgreifen können (Wiederverwendung der Kommunikationsmöglichkeiten).
Um den Informationsaustausch auf effiziente Art und Weise automatisieren zu können und ein hohes Maß an Wiederverwendung zu erreichen, muß die Bindung der Dienste mit den Geschäftsanwendungen, die für die Auslieferung eines Produkts verantwortlich sind, und der Geschäftsanwendungen untereinander, auf semantischer Ebene geschehen.
In der vorliegenden Architektur wird diese Problematik nicht durch spezialisierte Verbindungssoftware für die jeweilig zusammenarbeitenden Softwaresysteme gelöst, was im schlechtesten Fall eine quadratische Anzahl an Protokollkonvertern erfordern würde. Statt dessen wird von der statischen Föderierungsschicht ein allgemeines Austauschformat für Geschäftsanwendungen verwendet, um die Kommunikation zu ermöglichen, so daß lediglich eine Komponente pro primärer Dienstleistung benötigt wird. Der Einsatz eines allgemeinen Austauschformats kann den Aufwand für die Kommunikation also von quadratischer auf lineare Komplexität absenken, vgl. z. B. (Becker/Rosemann 1993, S. 20-23). Ursprünglich kommt diese Idee aus dem Bereich des *Computer Integrated Manufacturing (CIM),* um Entwurfsdaten zwischen verschiedenen Unternehmen auszutauschen. In diesem Zusammenhang spricht man von *CIM Interface Systems.* Die Voraussetzung für die Ausnutzung dieses Vorteils ist aber die vollständige und korrekte Implementierung des Standards in den verbleibenden Adapterkomponenten, welche die Daten der Geschäftsanwendungen in das Austauschformat übertragen.
Aus diesen Überlegungen ergeben sich drei Voraussetzungen für die Umsetzung der statischen Föderierungsschicht.

- Alle Benutzer des allgemeinen Austauschformats müssen sich auf eine gemeinsame *Ontologie* geeinigt haben. Eine Ontologie ist ein gemeinsames Vokabular, das zur Beschreibung der relevanten Geschäftsvorfälle geeignet ist.
- Spezielle *Kommunikationsdienste* müssen von jeder Geschäftsanwendung angeboten werden. Sie dienen der Übertragung der Ausgabe der Geschäftsanwendung in das gemeinsame Vokabular und umgekehrt. Diese Kommunikationsdienste erledigen die Kommunikation einzelner Geschäftsanwendungen mit der statischen Föderierungsschicht. Ihre Implementierung kann durch Adapterkomponenten erfolgen.
- Die *statische Föderierungsschicht* muß einen Übermittlungsdienst für Nachrichten *(Messages)* zwischen Anwendungssystemen zur Verfügung stellen. Sie übermittelt die im allgemeinen Austauschformat formulierten Nachrichten von Anwendungsystemen (bzw. ihren Adapterkomponenten) an andere Anwendungsysteme.

Für die geschilderte Problemstellung relevante Standardisierungsversuche sind aus dem Bereich der zwischenbetrieblichen Integration unter dem Stichwort EDI bekannt. EDI stellt einen Weg zur Verbesserung der Kommunikation im Business-To-Business Bereich dar. Hierzu wird in diesem Beitrag ein Ansatz vorgestellt. Die Vorteile von EDI liegen vor allem in der Schaffung organisationsbezogener informationeller Mehrwerte (Kuhlen 1996, s. bes. S. 90), die sich z.B. in Form einer verbesserten Aufbau- und Ablauforganisation sowie Zeit- und Kostenersparnis niederschlagen.
Als wichtigster Standard für den zwischenbetrieblichen elektronischen Datenaustausch ist das von den Vereinten Nationen vorgeschlagene UN/EDIFACT (Electronic Data Interchange for Administration, Commerce and Transport) (UN 1995) zu nennen. Gleichwohl konnte sich EDIFACT bisher noch nicht im erwarteten Umfang durchsetzen. Neben grundsätzlichen Problemen, wie dem Fehlen bestimmter semantischer Regeln, z.B. für Mengen- oder Verpackungseinheiten, und der impliziten Annahme unternehmensübergreifend einheitlicher Geschäftsprozesse und Geschäftsszenarios (Zbornik 1996, S. 92f.), hat dies insbesondere wirtschaftliche und organisatorische Ursachen (vgl. z. B. Goldfarb/Prescod 1998, S. 106-110), z. B. hohe Einstiegskosten (insbesondere für kleine und mittlere Unternehmen), geringe Flexibilität bei der Erweiterung oder Änderung von Nachrichtenformaten, langsame Anpassung des Standards an sich ändernde Geschäftsprozesse, Festschreibung bestimmter Geschäftsprozesse und Geschäftsregeln sowie relativ geringe Verbreitung.

Der Behebung der erstgenannten Probleme widmen sich die Vorhaben Open-EDI/object oriented-EDI (TMWG 1998), Universal Data Element Framework (UDEF) (Harvey et al. 1998, S. 25f.), Basic Semantic Repository (BSR) sowie dessen Nachfolger BEACON (Steel 1997). Bei diesen Vorhaben steht folglich die Definition einheitlicher Geschäftsszenarios und semantischer Regeln im Vordergrund.
Demgegenüber wird der zweitgenannte Problemkomplex schwerpunktmäßig durch die XML/EDI-Initiative adressiert (Peat/Webber 1997), bei welcher der Nutzen, der durch die Verwendung der Extensible Markup Language (Bray/Paoli/Sperberg-McQueen 1997) gestiftet wird, im Vordergrund steht. Die XML/EDI-Initiative zielt dabei vorrangig auf die Senkung der EDI-Einstiegskosten und eine Erhöhung der Flexibilität ab.
Das folgende Codebeispiel zeigt ein Fragment einer XML Nachricht. Sie fragt die Verfügbarkeit eines Produkts an. Die Feldbezeichner (Elementnamen) sind so gewählt, daß ihre Bedeutung für Menschen leicht erkennbar ist. Die Flexibilität des XML Formats liegt unter anderem darin, daß der Empfänger nur jene Elemente verarbeiten muß, die für ihn relevant sind. Eine vollständige Implementierung des gemeinsamen Vokabulars ist für ihn also nicht notwendig. Die Extraktion der gewünschten Daten aus einer XML Nachricht leistet der XML Parser. Hierfür ist jede gängige Parserkomponente geeignet.

```
<MESSAGE>
      <TYPE>Request for quote</TYPE>
      <REQUEST-DATE>15.03.1999</REQUEST-DATE>
      ...
      <PRODUCT>
            <EAN>536247732722</EAN>
            <DESCRIPTION>CellPhoneModel123</DESCRIPTION>
            <COLOR>black</COLOR>
            ...
      </PRODUCT>
</MESSAGE>
```

Die Syntax eines solchen Datenaustauschs kann in einer DTD festgelegt werden. Liegt den dabei gewählten Bezeichnern (d. h. dem gemeinsamen Vokabular) ein Standard oder eine von Sender und Empfänger akzeptierte Vereinbarung zugrunde, dann gewinnt eine Nachricht, die auf einer solchen DTD basiert, für den Empfänger eine gewisse Semantik, da ein Feldbezeichner (ein XML-Tag) unmittelbar auf die Bedeutung des Inhalts hinweist.
Die Grundidee von XML-basierten EDI, nämlich die Kapselung von Nutzdaten durch (standardisierte) XML-Elemente, wird im folgenden aufgegriffen. Dazu kommen in einem ersten Schritt die durch den UN/EDIFACT-Standard vorgegebenen Segmentnamen als standardisierte Feldbezeichner zur Sicherstellung der Kommunikation zum Einsatz. Sie ersetzen die XML-Elemente im vorherigen Beispiel, die so gewählt wurden, daß sie von einem menschlichen Leser leicht interpretiert werden können. Im folgenden Codebeispiel ist dargestellt, durch welche XML-Elemente das Element `REQUEST-DATE` aus der vorherigen Beispielnachricht ersetzt wird.
Die Bestimmung der Namen der XML-Elemente erfolgt anhand der Liste der UN/EDIFACT Segmentnamen, aus der hier ein Auszug dargestellt ist:

```
DTM DATE/TIME/PERIOD
    To specify date, time, or period.
 ...
 2005 Date/time/period qualifier,              M, an..3
  ...
  137 Document/message date/time, Date/time when a
      document/message is issued.
```

```
 ...
2380 Date/time/period,                          C, an..35
...
2379 Date/time/period format qualifier,         C, an..3
...
 102 CCYYMMDD, Calendar date: C = Century; Y = Year;
                              M = Month;   D = Day.
 ...
```

Hieraus ergibt sich folgendes XML-Fragment. Die Bedeutung der so übermittelten Nachricht lautet: „Die Anfrage ist vom 15. März 1999."

```
<DTM DTM2005="137">
  <DTM2380>19990315</DTM2380>
  <DTM2379>102</DTM2379>
</DTM>
```

Die Namen der dazu genutzten XML-Elemente entsprechen den UN/EDIFACT Definitionen. Die Segmentnamen sind in der Liste der UN/EDIFACT Segmentnamen durch Fettdruck hervorgehoben. Übergeordnete Segmentnamen (`DTM`) werden direkt als XML-Element-Name übernommen. Die XML-Element-Namen von abhängigen Segmentnamen (`2005`, `2380` und `2379`) werden um den Namen des übergeordneten Segmentnamens ergänzt (`DTM2005`, `DTM2380` und `DTM2379`). Notwendige Attribute (`2005`) werden zu Parametern des übergeordneten XML-Elements, wahlfreie Attribute (`2380` und `2379`) werden als eigene, untergeordnete XML-Elemente behandelt.
Die Wahl des UN/EDIFACT Standards als Grundlage der Namensvergabe stellt insofern keine Einschränkung dar, als die von verschiedener Seite durchgeführten und oben genannten Repository-Projekte die Rückwärtskompatibilität sicherstellen - also die Herstellung einer Beziehung zwischen den neuen Termen und den UN/EDIFACT-Kategorien erlauben (vgl. z. B. Harvey et al. 1998, s. bes. S. 10-12).
Die Umsetzung der Kommunikationsunterstützung geschieht durch einen *Tuplespace*, ähnlich zu dem Vorschlag in (Schoenfeldinger 1995). Ein Tuplespace ist ein verteilt organisierter Datenraum. Im Gegensatz zu anderen Implementierungen ermöglicht der für die statische Föderierungsschicht genutzte Tuplespace jedoch die Speicherung und Verarbeitung von XML-Daten, die im folgenden der Terminologie entsprechend XML-Tupel genannt werden. Hierdurch ist es möglich auch heterogene Tuplespaces zu verbinden, da die Kommunikation auf dem Hypertext Transfer Protokoll (HTTP) beruht und die Tupel plattformunabhängig in XML vorliegen. Durch die Beschreibung der Tupel in XML können diese auch auf ihre Korrektheit mittels einer DTD überprüft werden. Obwohl die Überprüfung anhand einer DTD nicht zwingend ist, kann die Überprüfung auf strukturelle Integrität der Daten insbesondere zur Erhöhung der Sicherheit bei der Kommunikation zwischen Geschäftsanwendungen beitragen. In unserem Szenario sind die XML-Tupel vorwiegend XML/EDI Daten, ein Datenaustausch für andere Zwecke der anwendungsübergreifenden Kommunikation ist aber auch denkbar und möglich. Der XML-Tupelspace stellt für die Kommunikation lediglich die drei Funktionen *out*, *in* und *read* zur Verfügung, die im folgenden beschrieben werden. Die Adressierung eines Tuplespaces findet durch eine Uniform Ressource Locator (URL) statt, die in der Signatur der Funktionen als *XML tuplespace URL* bezeichnet wird.

- *out(XML tuplespace URL, DTD, XML tuple)*
  Die Funktion *out()* schreibt ein XML-Dokument bzw. ein XML-Tupel, insbesondere die XML/EDI Daten in den XML-Tuplespace. Bereits existierende XML-Tupel werden hierbei nicht überschrieben. Wird die Funktion mit ein Angabe einer DTD aufgerufen, so werden die Daten bezüglich ihrer Struktur überprüft und bei korrekter Struktur mit der DTD im Tuplespace

abgespeichert. Diese Information kann bei Suchanfragen dazu genutzt werden, nur nach Daten einer bestimmten DTD.

- *in(XML tuplespace URL, DTD, XML tuple-query)*
  Die Funktion *in()* liest ein XML-Tupel aus dem XML Tuplespace aus, falls ein XML-Tupel zu einer gegebenen Suchanfrage paßt. Durch Angabe einer DTD kann darüber hinaus der Suchraum auf XML-Daten, die der DTD entsprechen eingeschränkt werden. Nach der Rückgabe eines gefundenen XML-Tupels befindet sich dieses nicht mehr im XML-Tuplespace.
- *read(XML tuplespace URL, DTD, XML tuple-query)*
  Die Funktion *read()* arbeitet nach dem gleichen Prinzip wie die Funktion *in()* jedoch bleiben gefundene XML-Tupel dem XML-Tuplespace erhalten, d. h. sie werden nicht gelöscht.

Die Beschreibung der Suchanfrage geschieht durch die *XML Query Language* (XQL), die in (Robie/Lapp/Schach 1998) vorgeschlagen wird. XQL beschreibt eine Notation für die Suche in XML-Dokumenten, ähnlich zu der Structured Query Language (SQL) für relationale Datenbanken. Durch XQL und die Berücksichtigung der Strukturelemente wird es möglich drei Klassen von Suchanfragen zu spezifizieren:

- *Eindeutig*
  Eindeutige XQL Suchanfragen sind XQL-Anfragen, die für die Suche eindeutige Kriterien, wie z. B. Schlüsselattribute, verwenden. Das im folgenden beschriebene XML-Tupel mit dem eindeutigen Attribut *ID* könnte durch die XQL-Suchanfrage `/PRODUCT/[@ID="pxz-23"]` gefunden werden:

  ```
  <PRODUCT ID='pxz-23'>
     <EAN>230239844531</EAN>
     <DESCRIPTION>Mobile Phone</DESCRIPTION>
  </PRODUCT>
  ```

- *Formal*
  Formale Suchanfragen sind XQL-Anfragen, die die Struktur von XML-Daten berücksichtigen. Hierbei kann es vorkommen, daß mehrere XML-Tupel der Anfrage entsprechen. Die folgende Suchanfrage liefert alle XML-Tupel, die ein mobiles Telefon beschreiben:

  ```
  /PRODUCT/DESCRIPTION[.="Mobile Phone"]
  ```

- *Semantisch*
  Bei dieser Form der Suchanfragen werden nicht nur die beschriebenen Daten berücksichtigt, sondern vielmehr ihre Beschreibung. In der folgenden Suchanfrage werden XML-Tupel gesucht, die als Produkt gekennzeichnet sind und die darüber hinaus eine EAN bzw. ein EAN-Element besitzen

  ```
  /PRODUCT[./EAN]
  ```

Da die Kommunikation mit dem XML-Tuplespace mittels HTTP durchgeführt wird, können auch etablierte Verfahren, die für herkömmliche Web-Anwendungen eingesetzt werden, angewendet werden. So ist die Sicherung des Datenaustauschs mit kryptografischen Verfahren, die im Web als Standard eingesetzt werden, denkbar. Die Implementierung des XML-Tuplespace kann durch die Verwendung eines herkömmlichen Web-Server sowie eines XML-Parsers durchgeführt werden.
In diesem Abschnitt wurde der Einsatz von Standardtechnologie, insbesondere der Kombination von XML/EDI mit UN/EDIFACT, für die Föderierung von Legacy-Systemen in einem Business-To-Business Szenario aufgezeigt. Im folgenden Abschnitt werden die vorgestellten Technologien und Einsatzmöglichkeiten an einem realen Beispiel veranschaulicht.

4 Das Service Konzept in einer E-Commerce Applikation

In einem Projekt zwischen dem Telecooperation Office (TecO) an der Universität Karlsruhe und der Hewlett-Packard GmbH (HP) wurde eine Service-orientierte E-Commerce Anwendung basierend auf dem WebComposition Ansatz mit WCML vollständig implementiert. Ziel des Projekts war es, ein Unterstützungssystem zu entwickeln, daß Entwicklung, Management, Wartung und Evolution von Services im heterogenen Umfeld des europäischen Intranets der Firma HP ermöglicht. Im Rahmen dieses Projekts wurde eine Architektur zur Beschreibung und Föderierung unterschiedlicher Dienste entwickelt, hierzu wurde das oben beschriebene Schichtenmodell umgesetzt. Für die Realisierung der Föderierungsschicht wurde jedoch als verteilter Datenraum ein Datenbankmanagementsystem eingesetzt. Für die Anbindung von Legacy-Systemen, z. B. Remedy und anderen Verwaltungs- und Warenwirtschaftssystemen, wurden entsprechende Agenten bzw. Adapter entwickelt, die über mit dem Datenraum bzw. indirekt mit den Services kommunizieren können.

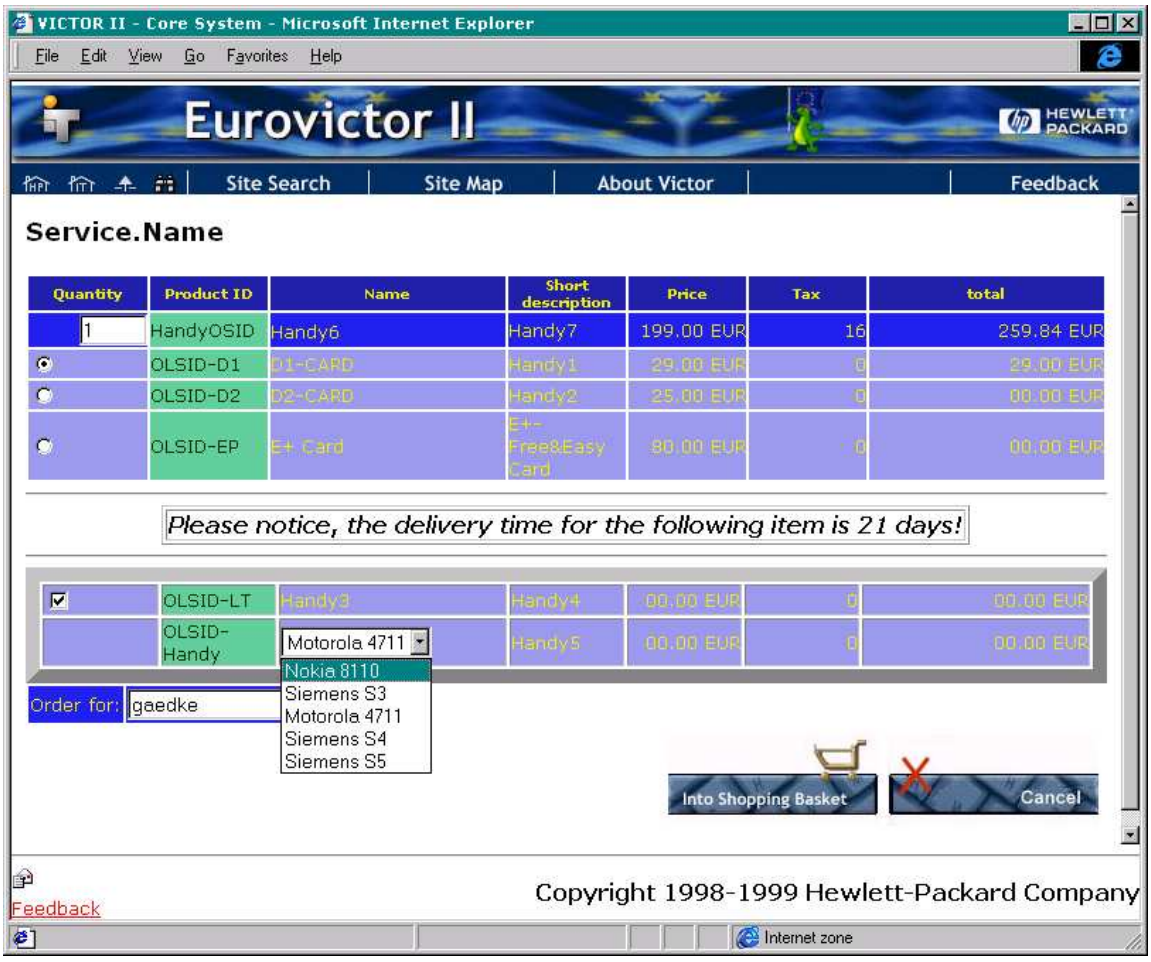

Abbildung 6: Föderierte Dienste grafisch hervorgehoben

Durch Nutzung von XML und der objektorientierten Eigenschaften von WCML können viele der Services wiederverwendet werden, so werden im internationalen Einsatz insbesondere die Übersetzung von Services und die Anpassung an nationale Layouts durch Vererbung und Referenzierung gelöst. Die Föderierung von Services wird insbesondere durch flexible Navigations-

Komponenten und den Föderierungsschichtansatz gefördert. Mittlerweile werden in nahezu allen europäischen Ländern Services verteilt entwickelt und über das am TecO entwickelte Eurovictor II System zur Verfügung gestellt.
Abbildung 6 wurde durch den Prototyp-Instanz Mechanismus, dem Vererbungsprinzip des WebComposition Ansatzes, erzeugt und anschließend durch die Manipulation von der Darstellungs- und Navigationskomponente geändert. In der Abbildung wurden die beiden föderierten Dienste, die diesem Dienst zugrunde liegen, grafisch hervorgehoben.

5 Zusammenfassung und Ausblick

Das Web ist zu einer Plattform für Anwendungen geworden, die es ermöglicht, die Grenzen der Heterogenität zu überwinden. Damit ist es geeignet, um die Föderierung betrieblicher Anwendungssysteme zu unterstützen, die im Zuge der Virtualisierung von Unternehmen zunehmend an Bedeutung gewinnt. Zwischenbetriebliche Integration und die zeitlich befristete Kooperation von Unternehmen zum Zwecke der gemeinsamen Leistungserstellung erfordern zum einen einen Ansatz, um (Web-basiert) Dienste Kunden gegenüber zugänglich zu machen, und zum anderen, einen Ansatz, um eine zwischenbetriebliche Kommunikation sicherzustellen, welche die betriebswirtschaftliche Semantik bewahrt. Beide Problemstellungen wurden in diesem Beitrag adressiert und unter dem Dach einer gemeinsamen Föderierungsarchitektur behandelt.
Zur Lösung des erstgenannten Problems wurde eine generische Föderierungsschicht vorgeschlagen, die auf der Verwendung von WCML aufsetzt und damit Vorteile einer komponentenorientierten Softwareentwicklung aufweist, so daß neben einer gestiegenen Flexibilität besonders positive Zeit- (kürzere Time-to-market) und Kosteneffekte (geringere Entwicklungskosten) zu erwarten sind.
Die statische Föderierungsschicht fokussiert auf die zweite Problemstellung. Auch sie setzt neben der Verwendung von WCML auf Standardtechnologien und –Datenaustauschformaten (XML, XML/EDI, Java, UN/EDIFACT) auf. Neben dem prinzipiellen Vorteil, der einfachen Möglichkeit zur Integration von (Legacy-)Anwendungssystemen, wird so eine weite Verbreitung begünstigt.
Insgesamt liegt so ein offener Ansatz zur Föderierung betrieblicher Anwendungssysteme vor, der auf dem Paradigma des Einsatzes Web-basierter Dienste beruht. So können beispielsweise auch Ansätze berücksichtigt werden, bei denen Ontologien in einem zentralen Repository im Web liegen und automatisch abgefragt werden können, z. B. die auf XML basierende BizTalk Initiative (BizTalk 1999).

Literatur

Barta, R. A.; Schranz, M. W. (1998): JESSICA: an object-oriented hypermedia publishing processor. Computer Networks and ISDN Systems 30(1998)(Special Issue on the 7th Intl. World-Wide Web Conference, Brisbane, Australia), S. 239-249.

Becker, J.; Rosemann, M. (1993): Logistik und CIM: Die effiziente Material- und Informationsflußgestaltung im Industrieunternehmen. Berlin.

BizTalk (1999): BizTalk Homepage. http://www.biztalk.org/. Zugriff: 03.06.1999.

Bray, T.; Paoli, J.; Sperberg-McQueen, C. M. (1997): Extensible Markup Language (XML). http://www.w3.org/TR/PR-xml.html. Zugriff: 12.06.1998.

Coda, F.; Ghezzi, C.; Vigna, G.; Garzotto, F. (1998): Towards a Software Engineering Approach to Web Site Development9th International Workshop on Software Specification and Design (IWSSD). Ise-shima, Japan.

ECOM (1998): Electronic Commerce - An Introduction. http://ecom.fov.uni-mb.si/center/. Zugriff: 15.05.1998.

Frakes, W. B.; IEEE Computer Society. Technical Council on Software Engineering.; Sociedade Brasileira de Computação.; Association for Computing Machinery. (1994): Third International Conference on Soft-

ware Reuse: Advances in Software Reusability : proceedings, November 1-4, 1994, Rio de Janeiro, Brazil. Los Alamitos, Calif.

Gaedke, M. (1998): WebComposition: Ein Unterstuetzungssystem fuer das Web Engineering. GI Softwaretechnik-Trends 18 Heft 3(Special Issue Softwaretechnik (ST´98), Paderborn Germany).

Gaedke, M.; Gellersen, H.-W.; Schmidt, A.; Stegemüller, U.; Kurr, W. (1999): Object-oriented Web Engineering for Large-scale Web Service ManagementThirty-Second Annual Hawaii International Conference On System Sciences (HICSS-32). Island of Maui, USA.

Gaedke, M.; Schempf, D.; Gellersen, H.-W. (1999): WCML: An enabling technology for the reuse in object-oriented Web EngineeringPoster-Proceedings of the 8th International World Wide Web Conference (WWW8). Toronto, Ontario, Canada.

Gamma, E. (1995): Design patterns : elements of reusable object-oriented software. Reading, Mass.

Gellersen, H.-W.; Gaedke, M. (1999): Object-Oriented Web Application Development. IEEE Internet Computing 3(1), S. 60-68.

Gellersen, H.-W.; Wicke, R.; Gaedke, M. (1997): WebCompostion: an object-oriented support system for the Web engineering lifecycle. Computer Networks and ISDN Systems 29 (1997)(Special Issue on the 6th Intl. World-Wide Web Conference, Santa Clara, CA, USA), S. 1429-1437.

Goldfarb, C. F.; Prescod, P. (1998): The XML Handbook. Upper Saddle River.

Harvey, B.; Hill, D.; Schuldt, R.; Bryan, M.; Thayer, W.; Raman, D.; Webber, D. (1998): Position Statement on Global Repositories for XML. ftp://www.eccnet.com/pub/xmledi/repos710.zip. Zugriff: 01.12.1998.

Isakowitz, T.; Stohr, E. A.; Balasubramaninan, P. (1995): RMM: A Methodology for Structured Hypermedia Design. Communications of the ACM 38, No. 8(Aug. 1995), S. 34-44.

Kristensen, A. (1998): Tempate resolution in XML/HTML. Computer Networks and ISDN Systems 30 (1998)(Special Issue on the 7th Intl. World-Wide Web Conference, Brisbane, Australia), S. 239-249.

Kuhlen, R. (1996): Informationsmarkt: Chancen und Risiken der Kommerzialisierung von Wissen. 2. Aufl., Konstanz.

Lim, W. C. (1998): Managing Software Reuse: A Comprehensive Guide to Strategically Reengineering the Organization for Reusable Components. Upper Saddle River, NJ.

Microsoft (1997a). FrontPage. Redmond, WA, USA: Microsoft.

Microsoft (1997b). Using Active Server Pages with Microsoft Internet Information Server 3.0. Redmond, WA, USA: Microsoft.

NCSA (1995). Tutorial on Server-Side Includes. Illinois, USA: National Center for Supercomputing Applications at the University of Illinois at Ur-bana-Champaign.

NetObjects (1997). NetObjects Home Page: NetObjects Inc.

Peat, B.; Webber, D. (1997): Introducing XML/EDI: "The E-business Framework". http://www.geocities.com/WallStreet/Floor/5815/start.htm. Zugriff: 01.12.1998.

Powell, T. A.; Jones, D. L.; Cutts, D. C. (1998): Web site engineering : beyond Web page design. Upper Saddle River, NJ.

Robie, J.; Lapp, J.; Schach, D. (1998): XML Query Language (XQL). http://www.w3.org/TandS/QL/QL98/pp/xql.html. Zugriff: 07/07/1999.

Sametinger, J. (1997): Software engineering with reusable components. Berlin ; New York.

Schoenfeldinger, W. J. (1995): "WWW MEETS LINDA" Linda for global WWW-based transaction processing systems4th Intl. World-Wide Web Conference. Boston, Massachusetts.

Schwabe, D.; Rossi, G.; Barbosa, S. (1996): Systematic Hypermedia Design with OOHDMACM International Conference on Hypertext' 96. Washington, USA.

Sommerville, I. (1982): Software Engineering. London ; Reading, Mass.

Steel, K. (1997): The Beacon User's Guide: Open Standards for Business Systems. http://www.cs.mu.oz.au/research/icaris/beaug1.doc. Zugriff: 01.12.1998.

Szyperski, C. (1998): Component Software: Beyond Object-Oriented Programming. 2. Aufl., Harlow.

TMWG (Hrsg.) (1998): Reference Guide: "The Next Generation of UN/EDIFACT": An Open- EDI Approach Using UML Models & OOT (Revision 12). http://www.harbinger.com/resource/klaus/tmwg/TM010R1.PDF. Zugriff: 01.12.1998.

Turowski, K. (1999): Ordnungsrahmen für komponentenbasierte betriebliche Anwendungssysteme. In: K. Turowski (Hrsg.): Tagungsband des 1. Workshops Komponentenorientierte betriebliche Anwendungssysteme (WKBA 1). Magdeburg, S. 3-14.

UN (Hrsg.) (1995): United Nations Directiories for Electronic Data Interchange for Administration, Commerce and Transport. http://www.unece.org/trade/untdid/Welcome.html. Zugriff: 01.12.1998.

Ungar, D.; Smith, R. B. (1987): Self: The Power of SimplicityOOPSLA '87., S. 227-242.

World Wide Web Consortium (Hrsg.) (1999): Extensible Stylesheet Language (XSL) Specification: W3C Working Draft 21 Apr 1999. http://www.w3.org/TR/WD-xsl/. Zugriff: 01.07.1999.

Zbornik, S. (1996): Elektronische Märkte, elektronische Hierarchien und elektronische Netzwerke: Koordination des wirtschaftlichen Leistungsaustausches durch Mehrwertdienste auf der Basis von EDI und offenen Kommunikationssystemen, diskutiert am Beispiel der Elektronikindustrie. Konstanz.