Wiederverwendung von Komponenten in Web-Anwendungen

Martin Gaedke

Telecooperation Office (TecO), Universität Karlsruhe
gaedke@teco.uni-karlsruhe.de

Zusammenfassung: Das World-Wide Web (Web) konnte sich innerhalb kürzester Zeit als Plattform für verteilte Anwendungen etablieren. Zunehmend nutzen mehr und mehr Organisationen dieses Medium, um ihre Produkte und Dienstleistungen anzubieten. Web-Anwendungen für den elektronischen Handel sind nicht nur durch eine Anbindung an betriebliche Anwendungen geprägt, sondern erfordern vielmehr eine adäquate Umsetzung von betrieblichen Abläufen im Web. Das einfache, dokumentenbasierte Implementierungsmodell, das dem Web zugrunde liegt, behindert jedoch den Entwurf und insbesondere die Wartbarkeit und Evolution von Web-Anwendungen. Die Evolution einer Web-Anwendung erfordert den Zugriff auf Entwurfs- und Implementierungsartefakte, um diese ändern oder wiederverwenden zu können. Nach der Abbildung auf das grobgranulare Web-Implementierungsmodell ist dies jedoch nicht mehr möglich. Die Anforderungen, die an betriebliche Web-Anwendungen gestellt werden, können daher ohne einen disziplinierten Ansatz analog zur Softwaretechnik nicht erfüllt werden. In diesem Beitrag wird das objektorientierte WebComposition System als Unterstützung für die komponentenorientierte Entwicklung von Web-Anwendungen vorgestellt. Insbesondere wird gezeigt, wie dadurch eine disziplinierte Wiederverwendung feingranularer betrieblicher Layout-, Interaktions- und Navigationskomponenten umgesetzt werden kann.

Einleitung

Anfang 1990 wurde das World-Wide Web (Web, WWW) als dezentrales Informationsmedium für weltweit verteilte Forschungsgruppen konzipiert. Die Veröffentlichung von Informationen wird durch ein relativ einfaches Implementierungsmodell ermöglicht, das auf einen lokationsunabhängigen Zugriff auf dateibasierte Ressourcen beruht [5]. Für die ursprünglich verfolgte Idee des Webs ist diese Abstraktion sinnvoll gewesen, da Dateien der Granularität der zur Verfügung gestellten Informationen, wie z.B. Forschungsberichte, Personendarstellungen und andere Publikationen, entsprachen und sich dafür eigneten, unabhängig voneinander erstellt und gepflegt zu werden [4,8,17].

Durch die große und stetig anwachsende Zahl von Dokumenten [45] wird es für Informationsanbieter immer schwieriger Dokumente an die technologischen Möglichkeiten anzupassen, Informationen zu aktualisieren und zu klassifizieren sowie das Informationsangebot im ganzen adäquat zu vergrößern. Infolgedessen wird an neuen Sprachen, wie z.B. CSS, XML, XSL, XQL, RDF, XHTML [7,9,24,53], gearbeitet, die diese Problematik erleichtern sollen. Weitere Ansätze verfolgen das Ziel der Abbildung bestehender Daten in das Web-Implementierungsmodell zu ermöglichen, wie z.B. XML/EDI für den elektronischen Datenaustausch basierend auf dem EDIFACT-Ansatz [26,50].

Das WWW ist allerdings nicht auf ein rein passives Informations- bzw. Ressourcenangebot beschränkt. Durch die technologische Entwicklung wurde die Integration dynamischer Elemente erleichtert und führte zu einem sprungartigen Anstieg vielfältiger, interaktiver Anwendungen; hierzu zählen u.a. Angebote in der Aus- und Weiterbildung, im Unterhaltungssektor und kommerziellen Bereich; Elektronischer Handel (engl. Electronic Commerce) ist sogar zum Leitmotiv für einen der wichtigsten Wachstumsmärkte unserer Zeit geworden [11,13,51].

Web-basierte Anwendungen (*Web-Anwendungen*) sind zunehmend durch verteilte Prozesse, heterogene Systeme, komplexe Mensch-Maschine Schnittstellen und Navigationsstrukturen, grobgranulare Ressourcen (die ursprünglichen passiven Dokumente) als Implementierungsentitäten und kurze Lebenszyklen geprägt. Trotzdem wird dem Entwurf und der Evolution der zunehmend komplexer werdenden Web-Anwendungen durch eine wenig disziplinierte bzw. ad-hoc Vorgehensweise begegnet. Gellersen [21] weist darauf hin, daß die Erstellung von Web-Anwendungen gleichbedeutend mit Softwareentwicklung ist und somit die Erarbeitung softwaretechnischer Methoden für Entwicklung, Betrieb und Wartung von Anwendungen im Web nahelegt. In Analogie zur Softwaretechnik bezeichnet man in diesem Zusammenhang die Anwendung von Softwaretechnik für Entwicklung, Betrieb und Wartung von Anwendungen im World-Wide Web auch als *Web Engineering* [23].

Software wird selten von Grund auf neu entwickelt. Eine disziplinierte Vorgehensweise zur Wiederverwendung von Softwareartefakten führt zu Qualitäts- und Produktionssteigerung sowie Kostenminderung eines zu entwickelnden oder ändernden Softwaresystems [15,28,49]. Die Realisierung und insbesondere Evolution von Web-Anwendungen durch

Wiederverwendung von Softwareartefakten wird jedoch durch das dem Web zugrundeliegende grobgranulare Implementierungsmodell erschwert [30,31,32,40]. Dieses Modell führt dazu, daß die feingranularen *Softwareartefakte*, wie Algorithmen, Entwurfsmuster, Navigationsstrukturen, Layout-Elemente, Prozeßabläufe, etc. [12,29,48], aus denen eine Web-Anwendung besteht, in der Implementierung verloren gehen. Ein geordneter Zugriff auf einzelne Artefakte, um sie an neue Anforderungen anzupassen, ist daher nicht mehr möglich. Die Evolution von Web-Anwendungen kann durch die zunehmende Komplexität und kurzen Lebenszyklen nicht beherrschbar sein ohne ingenieurmäßige Ansätze zur Unterstützung der Wiederverwendung von Softwareartefakten im Web. Mit *Softwarewiederverwendung* bezeichnet man den Prozeß aus bestehender Software neue Softwaresysteme zu entwikkeln [43].
Durch die bisher fast ausschließliche Betrachtung des Webs als Informationsmedium muß eine auf Softwaretechnik und Web-Technologie aufbauende Disziplin, welche die Softwarewiederverwendung im Web berücksichtigt und fördert, erst noch den Voraussetzungen des WWW angepaßt und in vielen Bereichen neu entwickelt werden.

Ansätze zur Softwarewiederverwendung im Web Engineering

Obwohl die Softwaretechnik viele Verfahren, Modelle und Methoden zur Wiederverwendung hervorgebracht hat [47], finden sie im Web nur selten Anwendung. Softwarewiederverwendung wird in starkem Maße z.B. durch objektorientierte Sprachen, Rahmenarchitekturen (engl. Frameworks), Bibliotheken, Analyse-, Entwurfs und Implementierungsmuster (engl. Pattern) unterstützt [14,16,20,28].
Bieber kritisiert in [6] die steigende Anzahl an Anwendungen, die lediglich eine einfache Web-basierte Schnittstelle zur Verfügung stellen, ohne jedoch den Benutzer durch die Hypermedia-Funktionalität des WWWs bei der Navigation zu unterstützen. In der von ihm entwickelten Relationship-Analysis (RA) werden die Beziehungen einzelner Elemente zueinander elaboriert, um die Navigation innerhalb der Web-Anwendung domänenorientiert anzubieten. Viele der durch RA gefundenen Beziehungen können in ähnlichen Anwendungsdomänen wiederverwendet werden, wenn derartige Beziehungen modelliert und (wiederverwendbar) implementiert werden können.
Die traditionellen Hypertext-Methoden, wie HDM, OOHDM, RMM, SHDT, die in [25,27,46] verglichen werden, haben den modellbasierten Entwurf von Hypertext- und Hypermedia-Anwendungen zur Zielsetzung. Die Wiederverwendbarkeit von Teilentwürfen wird jedoch vernachlässigt. Lediglich für den speziellen Hypermedia-Entwurfsfall „Web-Anwendungen“ lassen sich Informationsstrukturen in OOHDM objektorientiert gliedern und modellieren. Schwabe und Rossi unterstützen einen wiederverwendungsorientierten Entwurf in OOHDM, indem sie die Modellierungssprache um neu auftretende Entwurfsmuster erweitern. Dennoch bleibt ein Entwurf mittels OOHDM zu abstrakt, um Web-Anwendungen daraus effektiv umsetzen zu können, bzw. fehlt eine in diese Richtung zielende Unterstützung.
Entwurfsmuster, insbesondere die von Gamma et al. [16] gesammelten objektorientierten Entwurfsmuster, sind in der Softwaretechnik sehr populär geworden. Das Ziel von Entwurfsmustern ist die Sammlung und Erhaltung von Entwurfserfahrung, so daß diese einfach wiederverwendet werden kann. Eine vollständige Menge von Entwurfsmustern zur Lösung von Aufgaben einer bestimmten Domäne bezeichnet man daher nach Christopher Alexander auch als Mustersprache. Der überwiegende Nutzen von Entwurfsmustern wurde in [41,42] experimentell gezeigt. Für die Domäne Web konnten bereits eine gewisse Zahl von Entwurfsmustern gefunden werden [18,20,46], jedoch erschwert die Granularität der Implementierungsentitäten des Web die Erfahrungen adäquat zu erfassen.
Foote beschreibt eine Mustersprache (engl. Patternlanguage) für die Wiederverwendung von „herkömmlicher“ Software [15]. Die Mustersprache umfaßt eine Menge bekannter softwaretechnischer Verfahren zur Unterstützung von Entwicklung, Wartung und Evolution von Anwendungen. Die Mustersprache beinhaltet u.a. Rahmenwerke, Komponentenbibliotheken, feingranulare Objekte und Sprachwerkzeuge. Diese Mustersprache kann jedoch nicht ohne erheblichen Aufwand für Web-Anwendungen angewandt werden, da die Voraussetzungen hierfür durch das Web-Implementierungsmodell nicht gegeben sind.
Der Einsatz der beschriebenen Softwaretechniken kann daher nur durch eine disziplinierte Vorgehensweise und eine unterstützende Implementierungstechnologie stattfinden.
Laufzeitumgebungen wie z.B. Apache [2], ASP [34], Frontpage [33], RMCase [10], Server-Side Includes [35], NetObjects [36], ColdFusion [1] und Systeme mit ähnlichen Erweiterungen [3,37,38,39,44] streben eine Unterstützung für die Entwicklung von Web-Anwendungen an, die Zielrichtung dieser Unterstützungssysteme ist jedoch meist durch dynamische Daten geprägt, z.B. durch die Integration von Ergebnissen einer Datenbankanfrage. Dieser Ansatz kann allerdings nur die datenorientierte Sicht auf der dem Web zugrunde liegenden Ressourcengranularität lösen. Eine darüber hinaus gehende Betrachtung wesentlicher Merkmale zur Implementierung von Anwendungen, wie z.B. Transaktionsbedingungen, betriebliche Prozeßabläufe, Föderation bestehender Anwendungssysteme, etc. wird vernachlässigt.
In neuerer Zeit wird der systematische und disziplinierte Einsatz von Techniken zur Wiederverwen-

dung durch komponentenorientierte Softwareentwicklung vereinfacht [43,48]. Komponentenorientierte Softwareentwicklung verfolgt dabei das Ziel Anwendungssysteme durch die wahlfreie Kombination oder Adaption von wiederverwendbaren Softwarebausteinen (Komponenten) zu ermöglichen. Eine *Komponente* wird hierzu definiert als eine wiederverwendbare, innerhalb eines Komponentensystems unabhängige und vermarktbare Software, die über dokumentierte Schnittstellen verfügt, in zur Zeit der Entwicklung unvorhersehbaren Kombinationen mit anderen Komponenten eingesetzt und an anwendungsspezifische Erfordernisse angepaßt werden kann sowie eine bestimmte Menge von Aufgaben einer Anwendungsdomäne implementiert.
In den folgenden Abschnitten wird ein komponentenbasierter Ansatz für das Web Engineering vorgestellt, so daß die Modellierung und Implementierung von Web-Anwendungen mit feingranularen Strukturen möglich wird. Da die Komponenten nach ihrer Implementierung erhalten bleiben, können sie für die Evolution oder die Entwicklung weiterer Web-Anwendungen genutzt werden.

Der WebCompostion Ansatz

Der WebComposition Ansatz basiert auf einer komponentenorientierten Modellierung von Web-Anwendungen [17,23]. Eine Komponente im WebComposition Modell wird hierbei als eine Code-Abstraktion einer beliebigen Zielsprache verstanden, z.B. HTML, Skript Code, VRML oder aber auch LaTex, PDF etc. Die Granularität einer WebComposition Komponente ist ausschließlich abhängig von der Art eines Entwurfsartefakts. 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 Modell als Komposition feingranularer Komponenten beschrieben. Die Evolution von Web-Anwendungen vollzieht sich durch Manipulation oder Hinzufügen von Komponenten, die während ihres gesamten Lebenszyklus persistent und zugreifbar in einem Komponentenspeicher, dem sog. Component Store, gehalten werden.
Das WebComposition Modell ist objektorientiert. Die Erzeugung neuer Komponenten kann daher durch die Wiederverwendung existierender Komponenten erleichtert werden. Hierzu werden grundlegende Merkmale von Objektorientierung, wie Vererbung und Aggregation, angeboten. Im Gegensatz zu den klassenbasierten Ansätzen, wie z.B. bei Java oder C++, beruht das WebComposition Modell auf einem Prototyp-Instanz Modell [52]. Prototyping ist eine Technik zur Wiederverwendung von Code. Jede definierte Komponente kann im Modell als Prototyp dienen oder referenziert werden, um so die Wiederverwendung bei der Beschreibung neuer Komponenten zu ermöglichen.
Durch die Modellierungsmöglichkeiten begegnet das WebComposition Modell der Forderung der Evolution und Wiederverwendung, indem es die Implementierung einer Web-Anwendung auf einer höheren und feingranularen Ebene ermöglicht.

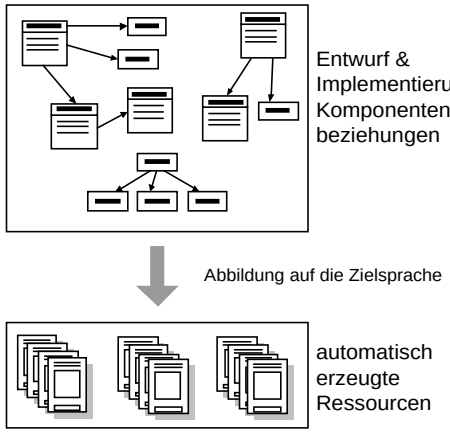

Abbildung 1 - Zusammenhang zwischen Modell und Ressourcen

In Abb. 1 ist die Abbildung von Komponenten auf Ressourcen dargestellt. Der Abbildungsmechanismus, der durch den sog. Ressource Generator zur Verfügung gestellt wird, nutzt dabei die Gesamtheit aller Komponenten, die für die Erstellung der Zielressourcen notwendig sind. Dadurch entfällt eine Implementierung direkt auf der Ebene einzelner Ressourcen.

WCML – Eine Beschreibungssprache für Komponenten im Web

Die WebComposition Markup Language (WCML) ist eine Anwendung von der eXtensible Markup Language (XML) [24,53]. Die Beschreibungssprache XML, die sich derzeit zum de-facto Standard für die Beschreibung von Daten im Internet etabliert, ermöglicht die Realisierung einer (Web-) Komponentenbeschreibungssprache auf der dem Web zugrunde liegenden Mechanismen. Diese Voraussetzung ist insbesondere im heterogenen Umfeld des WWW unabdingbar. Es ist evident, daß WCML die guten Eigenschaften von XML „erbt“, d.h. ist plattformunabhängig, einfach auszuwerten sowie streng im Sinne von syntaktisch und semantisch richtig.
Im WebComposition Modell werden Komponenten im Component Store verwaltet, da die Komponenten in Form von Beschreibungscode vorliegen, können Datenbanken, Dateien oder Web-Server als Component Store für Komponenten dienen und werden als virtual Component Store bezeichnet. Durch die URI-basierte Adressierung von virtual Component Stores können in WCML auch Komponenten anderer Component Stores wiederverwendet

werden. Die folgende Abbildung 2 verdeutlicht die Abbildung von Komponenten in eine Zielressource.

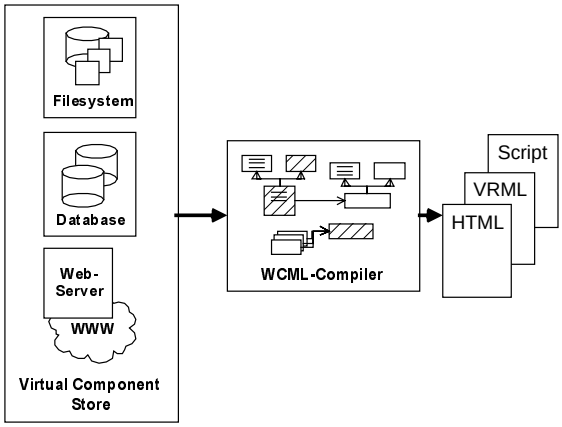

Abbildung 2 – Abbildung von WCML Komponenten in Ressourcen

Das folgende Beispiel verdeutlicht den Einsatz von WCML am Beispiel einer Servicekomponente. Eine Servicekomponente stellt einem Kunden eine Dienstleistung im Web zur Verfügung. Die Servicekomponente abstrahiert dabei von Navigationsunterstützung, firmeneinheitliches Layout, Benutzerinteraktion und weiteren Aktivitäten, die zur Erfüllung der Dienstleistung notwendig sind.

```
<?xml version='1.0' ?>
<!DOCTYPE wcml SYSTEM "wcml2.dtd">
<wcml>
<COMPONENT uuid=' StandardProduct'>
  <PROPERTY name='Waehrung'>DM
  </PROPERTY>
</COMPONENT>
<COMPONENT uuid='ProductService'>
  <PROTOTYPE is='StandardProduct'/>
  <PROTOTYPE is='Layout'/>
  <PROTOTYPE is='UserProcess'/>
  <PROTOTYPE is='OrderCode'/>

  <PROPERTY name='Name'>
     Dienstleistung
  </PROPERTY>
  <PROPERTY name='Preis'>
     59.99
  </PROPERTY>
</COMPONENT>
...
```

Das Beispiel zeigt die Beschreibung einer Komponente ProductService, die eine Dienstleistung zum Preis von 59.99 DM anbietet. Die Währungsinformation wird von der Komponente StandardProduct geerbt, was durch das TAG-Element PROTOTYPE und dem Attribut is='StandardProduct' beschrieben wird. Weitere Komponenten stellen das Layout, Navigation etc. zur Verfügung. Die Evolution der zur Verfügung gestellten Dienstleistungen kann einfach durch die prototypische Wiederverwendung der ProductService Komponente stattfinden. Änderungen an Navigation, Layout, Interaktionsmodell, etc. erfordern nun lediglich die Modifikation an einer ausgezeichneten Stelle/Komponente.

Stand der Arbeiten und Ausblick

Der WebComposition Ansatz und WCML wurden in Projekten zwischen dem TecO und industriellen Partnern (u.a. Daimler-Benz [18], Hewlett-Packard [19]) erfolgreich eingesetzt. Typische Probleme bei der Wartung und Evolution von Web-Anwendungen konnten gezeigt werden, die nur unzureichend von heutigen Entwicklungswerkzeugen unterstützt werden, z.B. Modellierung und Unterstützung von betrieblichen Abläufen mit Navigationsstrukturen.

Kompositionelle Wiederverwendung begründet sich auf der Idee möglichst viele Komponenten wiederverwenden zu können [43]. Komplexere Komponenten werden durch die Wiederverwendung feingranularer oder einfacher Komponenten gebildet. Hierzu ist es notwendig, daß Komponenten effizient gefunden werden können. Die Klassifikation von Komponenten, z.B. horizontale oder vertikale Klassifikationsschemata wie sie bei CORBA zu finden sind oder die Bildung von Bibliotheken, sind mögliche Ansätze.

Am TecO wird derzeit ein Repositorium (engl. Repository) entwickelt, um es als ein unterstützendes Werkzeug für das Web Engineering zur Verfügung zu stellen. Das Repositorium soll eine allgemeine Möglichkeit zur Referenzierung von Komponenten durch Metadaten ermöglichen. Das Auffinden von Komponenten kann dann durch verschiedene Klassifikationen, aber auch wissensbasierte Ansätze, wie z.B. adaptive Anfragen oder genetische Verfahren, die Wiederverwendung fördern.

WCML-Compiler

Beispiele und WCML-Compiler sind verfügbar unter:
http://www.teco.edu/~gaedke/webe

Literatur

1. Allaire Corp. Cold Fusion 2.0 Overview. Allaire Corp. Cambridge, MA, USA. 1997.
2. Apache (1996): *The Apache HTTP Server Project*. URL: http://www.apache.org/
3. Apple Computer, Inc. Hypercard User´s Guide. Apple Computer, Inc., Cupertino, CA, USA, 1987.
4. R. A. Barta; M. W. Schranz (1998): *JESSICA: an object-oriented hypermedia publishing processor*. In: Computer Networks and ISDN Systems 30(1998), Special Issue on the 7th Intl.

World-Wide Web Conference, Brisbane, Australia, p. 281-290.
5. T. Berners-Lee; R. Fielding; H. Frystyk. Hypertext Transfer Protocol (HTTP/1.0). RFC 1945, Mai 1996.
6. M. Bieber (1999): *Web Engineering*. HTF-7 Worksop, Helsinki. http://www.hci.oulu.fi/~kuutti/Bieber.htm 2.10.1999
7. T. Bray; J. Paoli; C. M. Sperberg-McQueen (Eds.) (1997): *Extensible Markup Language (XML)*. http://www.w3.org/TR/PR-xml.html. 12.06.98.
8. F. Coda; C. Ghezzi; G. Vigna; F. Garzotto (1998): *Towards a Software Engineering Approach to Web Site Development*. In Proceedings of 9th International Workshop on Software Specification and Design (IWSSD), Iseshima, Japan.
9. J. December; N. Randall (1994): *The World Wide Web Unleashed*. SAMS Publishing, Indianapolis, IN, USA.
10. A. Díaz; T. Isakowitz; V. Maiora; G. Gilabert (1995): *RMC: A Tool To Design WWW Applications*. The World Wide Web Journal, Issue One, Dezember 1995.
11. ECOM (Ed.) (1998): *Electronic Commerce - An Introduction*. http://ecom.fov.uni-mb.si/center/. 15.05.1998.
12. R. E. Fairley (1985). *Software Engineering Concepts*. McGraw-Hill.
13. K. Fellner; C. Rautenstrauch; K. Turowski (1999): *A Component Model for an Inter-organizational Agent-based Coordination*. To appear in: Information Resources Management Association (IRMA) International Conference, Hershey.
14. B. Foote (1995): *An Objective Look at Subjectivity*. Proceedings of OOPSLA'95 Workshop on Subjectivy in Object-Oriented Programming.
15. B. Foote; W.F. Opdyke (1995): *Lifecycle and Refactoring Patterns that Support Evolution and Reuse*. Edited by J.O. Coplien and D.C. Schmidt. Pattern Languages of Program Design I, Addison-Wesley.
16. E. Gamma; R. Helm; R. Johnson; J. Vlissides (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, *1994*.
17. M. Gaedke (1998): WebComposition: *Ein Unterstützungssystem für das Web Engineering*. In GI Softwaretechnik-Trends - Sonderheft zur Softwaretechnik´98 Konferenz, Paderborn.
18. M. Gaedke; M. Beigl; H. W. Gellersen (1998): *Mobile Information Access: Catering for Heterogeneous Browser Platforms.* In: Advances in Database Technologies, Lecture Notes in Computer Science (LNCS), Vol. 1552, Springer Verlag, November 16-19, 1998.
19. M. Gaedke; H. W. Gellersen; A. Schmidt; U. Stegemüller; W. Kurr (1999). *Object-oriented Web Engineering for Large-scale Web Service Management.* In: R. H. Sprague (Ed.) Proceedings of the 32nd Annual Hawaii International Conference On System Sciences, Maui, Hawaii, (CD-ROM).
20. D.M. Germán; D.D. Cowan (1999): *Three Hypermedia Design Patterns*. 2nd Workshop in Hypermedia Development: Design Patterns in Hypermedia at ACM Hypertext'99 Konferenz, Darmstadt. http://ise.ee.uts.edu.au/hypdev/ht99w/ 01.04.1998
21. H.-W. Gellersen (1997): *Web Engineering: Softwaretechnik für Anwendungen im World-Wide Web***.** In *Theorie und Praxis der Wirtschaftsinformatik*, Heft 196, Juli 1997.
22. H.-W. Gellersen and M. Gaedke (1999): *Object-Oriented Web Application Development.* In: IEEE Internet Computing Journal. Vol. 3, No. 1., pp. 60-68, January/February 1999. http://computer.org/internet/ic1999/w1toc.htm 01.04.1998
23. H.-W. Gellersen; R. Wicke; M. Gaedke (1997). *WebCompostion: an object-oriented support system for the Web engineering lifecycle*. In: Computer Networks and ISDN Systems 29, Special Issue on the 6th Intl. World-Wide Web Conference, Santa Clara, CA, USA, p. 1429-1437.
24. C. F. Goldfarb; P. Prescod (1998): *The XML Handbook*. Upper Saddle River.
25. F. Halasz; M. Schwartz (1994): *The Dexter Hypertext Reference*. Communications of the ACM, 37(2):30-39, Februar, 1994.
26. B. Harvey; D. Hill; R. Schuldt; M. Bryan; W. Thayer; D. Raman; D. Webber (1998): *Position Statement on Global Repositories for XML*. ftp://www.eccnet.com/pub/xmledi/repos710.zip p. 01.12.1998.
27. T. Isakowitz; E. A. Stohr; P. Balasubramaninan (1995). *RMM: A Methodology for Structured Hypermedia Design*. Communications of the ACM, Vol. 38, No.8, Aug. 1995, p. 34-44.
28. R.E. Johnson; B. Foote (1988): *Designing Reusable Classes*. In: Journal of Object-Oriented Programming, Vol. 1, Number 2, pp. 22-35
29. A. Kanerva; K. Keeker; K. Risden; E. Schuh; M. Czerwinski (1997): *Web Usability Research at Microsoft Corporation*. In J. Ratner, E. Grosse and C. Forsythe (eds.) Human Factors for World Wide Web Development, Lawrence Erlbaum, New York, United States of America. http://research.microsoft.com/research/ui/marycz/webchapter.html 08.12.1998
30. A. Kristensen (1998). *Template resolution in XML/HTML*. In: Computer Networks and ISDN Systems 30(1998), Special Issue on the 7th Intl. World-Wide Web Conference, Brisbane, Australia, p. 239-249.
31. D. Kristol; L. Montulli (1997). HTTP State Management Mechanism. RFC 2109.

32. D. A. Ladd; J. C. Ramming (1996): *Programming the Web: An Application-Oriented Language for Hypermedia Service Programming*. The 5th International World Wide Web Conference, Paris, France, 1996.
33. Microsoft Corp. (1997): FrontPage Home Page. http://www.microsoft.com/FrontPage/
34. Microsoft Corp. (1996): *Using Active Server Pages with Microsoft Internet Information Server 3.0*. White Paper. Microsoft Corp., Redmond, WA, USA.
35. NCSA (1995): *Tutorial on Server-Side Includes*. National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign, IL, USA. URL: http://hoohoo.ncsa.uiuc.edu/docs/tutorials/includes.html/
36. NetObjects Home Page, URL: http://www.NetObjects.com/
37. Netscape Communications Corp. (1996): *The Netscape ONE Development Environment Vision and Product Roadmap*. Netscape Communications Corp., Mountain View, CA, USA. http://home.netscape.com/comprod/one/white_paper.html
38. NeXT Software, Inc. (1996): *WebObjects Developer´s Guide*. NeXT Software, Inc. Redwood City, CA, USA.
39. J. Paoli (1996): *Cooperative Work On the Network: Edit the WWW!* The 5th International World Wide Web Conference, Paris, France.
40. T. A. Powell (1998): *Web Site Engineering*. Prentice-Hall Inc.
41. L. Prechelt; B. Unger; M. Philippsen; W. F. Tichy (1997): *Two Controlled Experiments Assessing the Usefulness of Design Pattern Information in Program Maintenance*. Submitted to Empirical Software Engineering. http://wwwipd.ira.uka.de/~prechelt/Biblio/patdoc_ese1999.ps.gz 01.04.1999
42. L. Prechelt; B. Unger; W. F. Tichy; P. Brössler; L. G. Votta (1999): *A Controlled Experiment in Maintenance Comparing Design Patterns to Simpler Solution*. Submitted to IEEE Transactions on Software Engineering. http://wwwipd.ira.uka.de/~prechelt/Biblio/patmain_tse1999.ps.gz
43. J. Sametinger (1997): *Software Engineering with Reuseable Components*. Springer Verlag.
44. W. J. Schoenfeldinger (1995). "*WWW MEETS LINDA" Linda for global WWW-based transaction processing systems.* In: Proceedings of 4th Intl. World-Wide Web Conference, Boston, Massachusetts, USA.
45. A. Schmidt; H.-W. Gellersen (1999): Suchen im Internet - Finden und Gefunden werden. In: HMD, Theorie und Praxis der Wirtschaftsinformatik, Heft 205, 36. Jahrg, Februar 1999.
46. D. Schwabe; G. Rossi; S. Barbosa (1996). *Systematic Hypermedia Design with OOHDM.* In Proceedings of the ACM International Conference on Hypertext, Hypertext '96, Washington.
47. I. Sommerville (1992): *Software Engineering*, Fourth Edition. Assison-Wesley.
48. C. Szyperski (1998): *Component Software - Beyond Object-Oriented Programming*. Addison-Wesley.
49. D. Ungar; R. B. Smith (1991). *SELF: The Power of Simplicity*. LISP and symbolic Computation: An International Journal, 4, 3, Kluwer Academic Publishers.
50. TMWG (Ed.) (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 01.12.1998.
51. K. Turowski (1999): *A Virtual Electronic Call Center Solution for Mass Customization*. In: R. H. Sprague (Ed.) Proceedings of the 32nd Annual Hawaii International Conference On System Sciences, Maui, Hawaii, (CD-ROM).
52. D. Ungar; R. B. Smith (1987). *Self: The power of Simplicity*, In: OOPSLA'87 Proceedings, p. 227-242.
53. World Wide Web Consertiun - W3C (1999): http://www.w3c.org/ 01.03.1999