WebComposition: Ein Unterstützungssystem für das Web Engineering

Martin Gaedke

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

Zusammenfassung

Das dokumentenbasierte Implementierungsmodell des Web beeinträchtigt den Entwurf und die Wartung von Webanwendungen. Die Wartung ist eine aufwendige und fehleranfällige Aufgabe, da auf viele Entwurfsentscheidungen nach ihrer Implementierung nicht mehr direkt zugegriffen werden kann. Trotzdem setzt sich nur allmählich die Erkenntnis durch, daß für die Entwicklung und Wartung von Webanwendungen ein disziplinierter Ansatz analog zum Software Engineering notwendig ist. In diesem Beitrag wird das WebComposition System als Unterstützung für das Web Engineering vorgestellt. Das System beruht auf einem objektorientierten Komponentenmodell für Webanwendungen und schafft die Voraussetzung für den feingranularen Zugriff auf die Komponenten während des gesamten Lebenszyklus einer Webanwendung sowie die automatische Generierung von Ressourcen für Webserver aus dem Modell heraus.

1 Motivation

Das World-Wide Web (WWW) ist aus Sicht der Softwaretechnik eine neue Anwendungsdomäne. Zunächst als Informationssystem für weltweit verteilte Forschungsgruppen entworfen, konnte sich das WWW innerhalb kürzester Zeit als Plattform für Hypertextanwendungen vom akademischen bis hin zum kommerziellen Umfeld etablieren und ist mittlerweile (fälschlicherweise) zu einem Synonym für das Internet geworden. Motiviert durch die steigenden Nutzerzahlen treibt die Wirtschaft zusammen mit der akademischen Forschung die Entwicklung von neuen weborientierten Standards voran, bei denen es sich fast durchweg um de facto Standards handelt, da die Rate der Einführung von Neuerungen selbst für die vergleichsweise schnellen Standardisierungsprozesse im Bereich Internet zu hoch ist. Es ist nicht damit zu rechnen, daß die Entwicklung von Webanwendungen in Zukunft leichter werden wird, da Ansprüche und Bedingungen an die Qualität der Anwendungen gestiegen sind und weiter steigen. Während es anfangs ausreichte, "online zu sein", ist es inzwischen besonders für kommerzielle Anbieter im Web wichtig, sich ansprechend und auf den einzelnen Kunden speziell angepaßt zu präsentieren, um die Erfüllung des Ziels der Institution oder des Unternehmens voranzutreiben. Dem Entwurf und der Wartung der zunehmend komplexer werdenden Webanwendungen wird (bisher) jedoch mit einer wenig disziplinierten bzw. ad-hoc Vorgehensweise begegnet. In [GeWG97] wird darauf hingewiesen, daß die Erstellung von Webanwendungen gleichbedeutend mit Softwareentwicklung ist und somit die Erarbeitung softwaretechnischer Methoden für Entwicklung, Betrieb und Wartung von Anwendungen im Web nahelegt. Diese neue Disziplin, im folgenden Web Engineering (oder kurz WebE) genannt, verspricht eine Kostenreduktion bei der Entwicklung und vor allem bei der Wartung. Dies wird dadurch erreicht, daß der Lebenszyklus als solcher planbar wird und somit hilft die Produktqualität zu sichern. Vereinfacht betrachtet, können die Aufgaben des Web Engineering mehreren Entwicklungsphasen zugeordnet werden, die denen "herkömmlicher" Softwareentwicklung entsprechen. Hierbei wird von den folgenden Phasen ausgegangen:

- *Analyse.* In der Analysephase werden Informationen über die Anwendungsdomäne gesammelt und bewertet. Hierbei ist insbesondere zu analysieren, welcher Art der zu präsentierende Informationsraum ist. Die anzusprechende Zielgruppe beeinflußt hierbei sowohl die benutzte Sprache und Darstellung als auch die Art der Bedienung der Webanwendung.

- *Entwurf.* Ziel des Entwurfsprozesses ist es, Inhaltselemente des durch die Analyse gefundenen Informationsraumes genau zu definieren und ihnen eine logische Struktur zu geben. Ergänzend werden Inhalts- und Strukturelemente auf Ressourcen, das sind MIME-Typ qualifizierte Dokumente, aufgeteilt und die konkrete Navigationsstruktur zwischen Elementen bzw. Ressourcen bestimmt. Darüberhinaus resultiert aus dem Entwurfsprozeß die Definition wiederverwendbarer bzw. die Wiederverwendung bestehender Entwufsartefakte, z.B. von Layoutelementen wie Logos oder Navigationsleisten, um der Webanwendung ein einheitliches Look&Feel zu verleihen.

- *Implementierung.* Die Implementierungsphase zeichnet sich durch die Abbildung von Inhalt und Strukturen auf Ressourcen aus. Die während der Entwurfsphase definierten Navigationsstrukturen müssen durch HTML-Links in den dazugehörigen Seiten umgesetzt werden. Auf Strukturen der Anwendung, die während der Entwurfsphase gewonnen wurden, kann nach ihrer Abbildung auf Ressourcen nicht mehr als eigenständige Entwurfsartefakte zugegriffen werden. Der Verlust von Strukturinformationen bzw. der Information über Entwurfsartefakte bei der Umsetzung in Implementierungsartefakte erschwert daher die spätere Wartung der Webanwendung.

- *Betrieb und Wartung.* Webanwendungen zeichnen sich gegenüber herkömmlicher Software nach ihrer initialen Implementierung durch eine hohe Änderungsdynamik sowohl der Inhalte als auch der Struktur aus. In dieser Phase wird die Anwendung häufig sehr schnell einer weiteren Analyse- und Implementierungsphase unterworfen.

Obwohl die beschriebenen Phasen weitestgehend mit denen herkömmlicher Softwareentwicklung vergleichbar sind, unterscheiden sie sich in ihrem zeitlichen Rahmen und in der Häufigkeit der durchlaufenen Zyklen. Während die Wartung herkömmlicher Software hauptsächlich durch das Beseitigen von Fehlern bestimmt ist, die Funktionsweise und die Benutzerschnittstelle eines Programmes aber weitgehend unverändert bleiben, befinden sich Webanwendungen in einem Zustand kontinuierlicher Weiterentwicklung. Typische Probleme der Wartung einer Webanwendung, also erneute Analyse, Entwurf und ergänzenden Implementierung, sind z.B. Änderung replizierter HTML-Fragmente, die Manipulation an Strukturen, die Änderung der Aufteilung auf Ressourcen oder die Änderung der Navigationsstruktur. Die Orientierung an einem bestimmten Hypertextmodell (z.B. an dem Dextermodell [HaSc94] ist wegen der unterschiedlichen Ausprägungen einer Webanwendung wenig sinnvoll. Die Beschreibung einer Webanwendung sollte daher auf einer allgemeineren Ebene erfolgen, wobei aber die Möglichkeit der Bildung höherer Abstraktionen zur Implementierung bestimmter Hypertextmodelle gegeben sein sollte. Die allgemeine Zielsetzung bei WebComposition ist die Schaffung einer Grundlage für Web Engineering, d.h. ein disziplinierter Ansatz analog zum Software Engineering für Entwicklung, Betrieb und Wartung von Webanwendungen, bestehend aus aufeinander abgestimmten konstruktiven Maßnahmen der Qualitätssicherung. Dazu zählen u.a. die Strukturierung von Entwicklung und Wartung in einem Prozeßmodell, der gezielte Zugriff auf Entwurfsentscheidungen über den gesamten Lebenszyklus einer Webanwendung hinweg sowie Modelle zur Beschreibung von Entwurfsartefakten [Gell97] und Methoden zur Wiederverwendung von daraus resultierenden Artefakten.

2 Stand der Entwicklung im Web Engineering

Webanwendungen können sehr unterschiedliche Ausprägungen haben. Die Bandbreite reicht von statischen, dokumentenorientierten Hypertextsystemen bis hin zu interaktiven, verteilten Anwendungen (z.B. realisiert mit Hilfe von Formularen und CGI-Skripten). Hinsichtlich der unterschiedlichen Ausprägungen existieren verschiedene Werkzeuge, die bei der Implementierung von statischem Hypertext, bei der Integration von Datenbanken als Informationssysteme oder bei der Entwicklung von hochdynamischen Anwendungen (z.B. sessionbasierte Anwendungen) unterstützen. Die Realisierung eines statischen Hypertext Systems ist dadurch charakterisiert, daß nur statische Links und Seiten angewendet werden. Im Vergleich hierzu sind bei einem Webinformationssystem die Seiten dynamisch, lediglich die Linkstruktur bleibt statisch. Dynamische Webanwendungen zeichnen sich durch die dynamische Generierung sowohl der Seiten als auch die Linkstruktur aus. Die ersten Web Engineering Werkzeuge stellten verschiedene Arten der Entwicklungsunterstützung zur Verfügung. So unterstützen einige Werkzeuge die Erstellung einzelner Seiten, andere unterstützen das Management von Websites [Micr98a] und wieder andere beinhalteten grundlegende Formen der Abstraktion, z.B. arbeitet das Entwicklungswerkzeug Fusion von NetObjects zur Erstellung dokumentenorientierter Webs mit dem Begriff "Style", der alle Standardelemente des Seitenlayouts (Grafiken, Textattribute usw.) umfaßt. Die Art bzw. der Zeitpunkt der *Ressourcengenerierung* ist ein anderes Kriterium, das zur Klassifikation von Werkzeugen dienen kann. Von statischer Generierung spricht man, wenn zum Zeitpunkt der Erstellung der Webanwendung der Code jeder Ressource vollständig definiert ist und in einem Datenspeicher abgelegt werden kann. Im Gegensatz dazu steht die dynamische Generierung, bei der eine Ressource erst zum Zeitpunkt der Anforderung durch den Server erzeugt wird. Wichtige Vertreter dieser Klasse sind Server-Side Includes (SSI) und Server-Side Scripting. Beim SSI [fSAa95] können mehrere Dateien in eine HTML-Datei eingelesen werden. Dieser Mechanismus wird zumeist durch eine Art Präprozessor, der direkt im Webserver integriert ist, realisiert. Weitergehende Konzepte, wie Server-Side Scripting, ermöglichen das einbetten von Befehlen in HTML-Ressourcen, die serverseitig vor Auslieferung der Ressourcen von einem Skriptinterpreter ausgeführt werden. Stellvertreter dieser Kategorie sind insbesondere Active Server Pages [Micr98b] und PHP/FI [Lerd98]. Eine weitere Klasse von Werkzeugen unterstützt die *Entwicklung dynamischer Anwendungen*. Hierzu zählt das WebObjects System [Soft96], daß ebenfalls auf dem Prinzip von Server-Side Scripting basiert. Im Unterschied zu diesem stellt WebObjects dem Entwickler neben Laufzeitsystem und Skriptinterpreter auch eine umfangreiche Klassenbibliothek zur Verfügung, der eine objektorientierte Architektur für Webanwendungen zugrundeliegt. Allerdings unterstützt WebObjects nicht die Organisation und Verwaltung von Ressourcen, zudem ist die Wiederverwendung von Objekten nur eingeschränkt möglich, da sie in statischen HTML-Templates eingebettet sind. Mit Ausnahme von RMCase [DIMG95] sind alle betrachteten Werkzeuge auf eine der beiden Möglichkeiten der Ressourcengenerierung festgelegt, wobei die dynamische Generierung überwiegt. RMCase soll den gesamten Lebenszyklus einer Webanwendung von Design über Implementierung bis hin zu Betrieb und Wartung unterstützen. Es basiert auf der in [IsSB95] beschriebenen Relationship Management Method (RMM) und ist an sich unabhängig von einer bestimmten Hypermediaplattform. Es ist zu erwähnen, daß RMCase wegen seines zugrunde liegenden Datenmodells RMM nicht für alle Arten von Webanwendungen gleichermaßen geeignet ist. Es orientiert sich stark an E-R-Modellierung und ist dementsprechend für die Abbildung von Inhalten relationaler Datenbanken geeignet.

Zusammenfassend ist festzustellen, daß dateibasierte Ressourcen zur Modellierung von Webanwendungen insbesondere im Hinblick auf Wiederverwendung und Wartbarkeit ungeeignet sind, da sie als "Modellierungseinheit" zu grob sind. Die Entwurfsentscheidung der Strukturierung der Inhalte auf Ressourcen ist nach der Implementierung als solche nicht mehr zugreifbar. Für Web Engineering und insbesondere für eine bessere Unterstützung des Betriebs und der Wartung einer Webanwendung sind feiner granulare Strukturen erforderlich, um Web Engineering über den gesamten Lebenszyklus einer Webanwendung anzuwenden.

3 WebComposition Component Model

WebComposition tritt den Forderungen an Entwicklung und Wartbarkeit entgegen, indem es die Implementierung einer Webanwendung auf einer fein granularen Ebene ermöglicht, auf der sämtliche Informationen des Entwurfs zugreifbar und manipulierbar sind.

Die Abbildung auf Ressourcen wird dann durch die Gesamtheit aller Objekte, die als Komponenten bezeichnet werden, im Modell einer konkreten Webanwendung definiert (zum Zusammenhang zwischen Modell und Ressourcen, siehe Abbildung 1). Eine Implementierung direkt auf der Ebene einzelner Ressourcen entfällt. Eine Webanwendung wird durch eine Menge von Komponenten repräsentiert.

3.1 Komponenten

In dem Projekt WebComposition wurde ein implementierungsunabhängiges, objektorientiertes Component Model entwickelt. Das Modell bildet die Grundlage für das WebComposition Laufzeitsystem, mit dem das Ziel verfolgt wird, eine bessere Plattform für die Beschreibung, Wiederverwendung und Wartung von Webanwendungen zu schaffen. Das Component Model stellt Komponenten als uniformes Modellierungskonzept für Entwurfsartefakte in Webanwendungen zur Verfügung. Die Idee des Component Model ist die objektorientierte Darstellung einer Webanwendung über ihren gesamten Lebenszyklus als Basis für ein abstraktes Entwurfsmodell und für das Laufzeitmodell. Durch das Component Model wird zum einen eine automatische aber auch noch im Detail manipulierbare Abbildung auf Ressourcen unterstützt, während es auf der anderen Seite besser mit objektorientierten Konzepten oder höheren Abstraktionen korrespondiert als das Webimplementierungsmodell. Eine Webanwendung wird im Modell als hierarchische, strukturelle Dekomposition von Komponenten beschrieben. Auf der höchsten Hierarchieebene können Komponenten dabei ganzen Teilwebs entsprechen, ferner korrespondieren sie mit einzelnen Ressourcen oder Fragmenten von Ressourcen. Auf dieser Stufe der Hierarchie können Komponenten anwendungsspezifische Abstraktionen darstellen, wie z.B. Firmenlogo, Navigationsleiste, Werbespalte, oder aber auch nur Wertepaare, die das Design der Seite beeinflussen, z.B. Hintergrundfarbe und Überschriftsgröße. Die Blätter in der Komponentenhierarchie werden als Primitive bezeichnet, Beispiele hierfür sind einfache Werte u.a. vom Typ Datum, Integer, Text, oder URL. Komponenten sind unterscheidbare Objekte, die einen Zustand in Form von typisierten Attributen, die Properties genannt werden und einfachen Name-Wert Paaren entsprechen, und ein Ver-

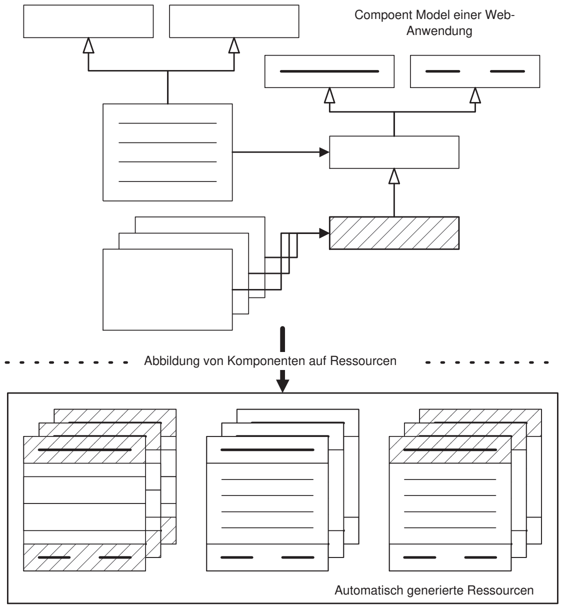

Abbildung 1: Component Model und Webimplementierungsmodell

halten als Menge von Methoden besitzen. Mit Hilfe der verschiedenen Methoden löst eine Komponente eine Reihe von Aufgaben, die entsprechend ihrem Einsatzgebiet zu Diensten zusammengefaßt werden. Eine Komponente stellt mindestens zwei Basisdienste bereit:

Darstellungsdienst. Durch die Methoden *generateCode* und *update* wird eine Repräsentation der Komponente in Abhängigkeit von ihrem Zustand erzeugt. Bei dem daraus resultierenden Code kann es sich um den Teil einer Ressource, einer ganzen Ressource oder sogar mehrerer Ressourcen handeln.

Persistenzdienst. Dieser wird durch die Methoden *getProperties* und *setProperties* angestoßen, die dazu dienen, den Zustand einer Komponente zu serialisieren bzw. zu deserialisieren. Dadurch wird es möglich, Komponenten in einem Datenspeicher festzuschreiben, sie also persistent zu halten, und sie später aus diesem wieder zu laden.

Diese Basisdienste ermöglichen es, eine Menge von Komponenten zu definieren, die zusammen ein ganzes Web erzeugen, und diese zusammen in Dateien oder einer Datenbank zu speichern, wie die folgende Abbildung 2 darstellt.

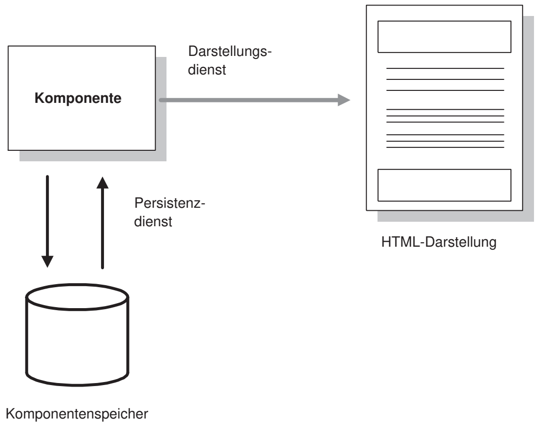

Abbildung 2: Von Komponenten erbrachte Dienste

3.2 Vererbung

Bei objektorientierten Modellen dient der Mechanismus der Vererbung zur Ableitung neuer Objekte auf der Basis existierender, indem deren Definitionen übernommen und erweitert werden. Für die Ableitung neuer Komponenten aus alten wurde ein Prototyping-Modell gewählt, wie es in [UnSm91] für die Programmiersprache *SELF* beschrieben ist. Es verspricht für die Modellierung von Entwurfsartefakten geeigneter zu sein, als die klassenbasierte Vererbung gängiger objektorientierter Sprachen wie Smalltalk oder Java, da viele Instanzausprägungen disjunkt sind. Beim Prototyping-Modell von WebComposition wird keine Unterscheidung zwischen Klassen und Instanzen vorgenommen. Jede Komponente kann gleichzeitig als Instanz benutzt werden und als ein Prototyp für andere Komponenten dienen, denen sie ihr Verhalten und ihren Zustand zur Verfügung stellt. Die Beziehung ist-Unterklasse und ist-Instanz-von-Klasse der klassenorientierten Vererbung werden hier also auf eine einzige Relation *nutzt-Prototyp* abgebildet. Jede Komponente kann ihren Zustand bzw. ihre Propertymenge durch mehrere Prototypen ergänzen. Ob auf ein Prototyp-Attribut oder ein lokal definiertes Attribut zugegriffen wird, ist für die Methoden der Komponente transparent. Allerdings sind die Werte von Prototyp-Attributen weiterhin nur einmal innerhalb des jeweiligen Prototyps vorhanden (vergleichbar mit Klassenvariablen in Programmiersprachen wie Smalltalk oder Java). Eine Änderung der Attribute einer Komponente wirkt sich transitiv auf den Zustand aller von ihr als Prototyp referenzierten Komponenten aus. Prototyp Attribute können von lokalen Attributen gleichen Namens überladen werden. Hinsichtlich der Repräsentation der Komponenten durch HTML-Code, kann man sich Prototypen als eine Art HTML-Schablone denken, die eine bestimmte Struktur vorgibt und Bereiche definiert, in denen konkrete Inhalte eingefügt werden können. Der Vorteil von Prototypen ist ähnlich dem der gemeinsamen Nutzung von Komponenten zur Replikation von Repräsentationen. In diesem Fall werden die Prototypen und damit ihr Zustand und Verhalten gemeinsam genutzt (Component Sharing). Das Component Sharing reduziert den Entwicklungsaufwand durch Wiederverwendung von Prototypen und führt zu einer Erleichterung der Wartung, da gemeinsam benutzte Properties an nur einer Stelle geändert werden müssen.

4 WebComposition System

Die Nutzung des Component Model für Entwicklung und Wartung von Webanwendungen erfordert eine Systemunterstützung. Ein WebComposition System stellt persistenten Speicher für das Component Model zur Verfügung und verwaltet den feingranularen Zugriff auf die Entwurfsartefakte während des gesamten Lebenszyklus der Webanwendung. Außerdem bildet das System das Component Model inkrementell auf dateibasierte Ressourcen ab, so daß das System für den verwendeten Webserver transparent bleibt. Die System Architektur zur Umsetzung dieser Aufgaben ist in der Abbildung 3 dargestellt.

Den Schwerpunkt des Systems bildet der Component Store, der für die persistente Speicherung der Komponenten sorgt. In der ersten Implementierung des WebComposition System basiert der Component Store auf einem relationalen Database Management System (RDBMS). Das Prototyping-Modell, das dem Component Model zugrunde liegt, läßt sich sehr einfach in einer relationalen Datenbank implementieren, so daß trotz des objektorientierten Ansatzes kein OODBMS für die Datenhaltung benötigt wird. Zu den Diensten des Component Store gehören neben Persistenz- und Konsistenzerhaltung des Component

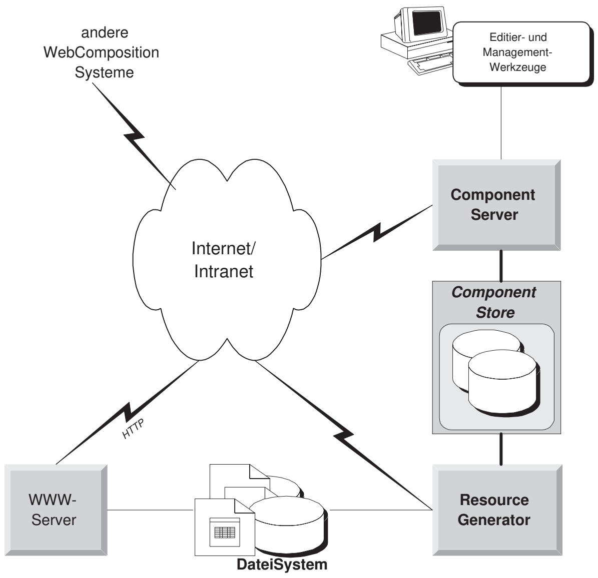

Abbildung 3: Das WebComposition System

Model auch Dienste wie Revisionsmanagement, um die Entwicklungsgeschichte jeder Komponente aufzuzeichnen.

Der Component Server dient dem geordneten Zugriff auf das im Component Store gehaltene Component Model, so daß Werkzeuge und Anwendungen über eine festgelegte Schnittstelle auf die Komponenten zugreifen können. Der Zugriff auf Komponenten kann im Rahmen der Wartung erfolgen, um explizite Änderungen an einer Anwendung vorzunehmen, oder aus der Anwendung selbst, indem durch den Webserver Prozesse angestoßen werden, die mit dem Component Store kommunizieren. Der Zugriff auf das eigene Komponentenmodell durch eine Webanwendung kann z.B. dazu genutzt werden, um Komponenten dynamisch zu erzeugen. Das WebComposition System fungiert somit nicht nur als Repositorium in einer Entwicklungsumgebung sondern auch als im Betrieb manipulierbare Anwendungsplattform.

Der Resource Generator des WebComposition System realisiert die Integration des Component Model mit dem eigentlichen Web, in dem eine Anwendung schließlich abläuft. Der Dienst des Resource Generator ist die automatische Abbildung einer im Modell beschriebenen Anwendung auf die Ressourcen im Dateisystem, auf denen Standard-Webserver operieren können. Das Dateisystem fungiert dabei quasi als Cache zwischen Web und Component Model. Die Abbildung erfolgt durch die Nutzung des Darstellungsdienstes von Komponenten und unter Berücksichtigung der zwischen den Komponenten bestehenden Abhängigkeiten, die durch Prototyp-Instanz-Beziehungen, Referenzierungen und Revisionen auftreten. Der Resource Generator ermöglicht neben der Erzeugung der gesamten Anwendung auch die inkrementelle Erzeugung von Teilen einer Anwendung. Die inkrementelle Erzeugung kann durch eine benutzergesteuerte (z.B. nach Wartungsarbeiten), ereignisgesteuerte (z.B. bei Eintreffen einer Bedingung) oder zeitgesteuerte (z.B. zur regelmäßigen Aktualisierung einer Implementierung bei fortlaufender Weiterentwicklung am Component Model) Aktivierung angestoßen werden. Durch die ereignisgesteuerte Erzeugung (hier bei dem Ereignis der Änderung) von Ressourcen läßt sich beispielsweise erreichen, daß dynamische Ressourcen nicht mehr bei jedem Webzugriff erzeugt werden müssen, sondern zwischen tatsächlich auftretenden Veränderungen statisch vorgehalten werden können. Das WebComposition System ist ein Unterstützungssystem für das Web Engineering. Es ist keine integrierte Entwicklungsumgebung sondern eine Plattform, die Werkzeugen und Anwendungen einen feingranularen Zugriff auf Webanwendungen während ihres gesamten Lebenszyklus ermöglicht. Die tatsächliche Arbeit mit dem WebComposition System sollte durch Autorenwerkzeuge und Editoren zur Manipulation von Komponenten unterstützt werden, darüber hinaus ist die Integration existierender Werkzeuge sinnvoll.

5 Stand der Arbeiten und Ausblick

Ein erster Prototyp des WebComposition Systems wurde implementiert und kam zur Erstellung interner Webseiten zum Einsatz. Der Prototyp arbeitet in einem heterogenen Umfeld basierend auf Windows NT und UNIX. Die plattformabhängigen Systemkomponenten sind ein Webserver unter UNIX und der Component Store, der auf dem Microsoft SQL-Server realisiert wurde. Der Component Store unterstützt neben dem Zugriff auf die Komponenten, Revisionskontrolle, Mehrbenutzermanagement und ein einfaches Transaktionsmanagement zur Konfliktbehandlung beim gleichzeitigen Zugriff auf Komponenten. Der Component Server und Resource Generator wurden in Java implementiert, wobei der Zugriff auf das DBMS mit JDBC/ODBC realisiert wurde. Darüber hinaus wurden die ersten Erfahrungen aus WebComposition für den Betrieb und die Wartung des an Ressourcen sehr umfangreichen Baden-Württemberg-Servers [Bade97] eingebracht. Die Entwicklung dynamischer Webanwendungen auf Basis eines an das WebComposition Component Model angelehnten Modells wurde im Projekt MobilBot mit Daimler-Benz Research erfolgreich erprobt [Gaed97]. Durch verschiedene strukturelle Änderungen während des Betriebs konnten typische Aufgaben der Wartung einer Webanwendung gezeigt werden, die nur unzureichend von heutigen Entwicklungswerkzeugen unterstützt werden, wie z.B. die Modifikation von Navigationsstrukturen, die Zusammenführung vieler verknüpfter Seiten auf eine Seite oder die Layoutänderung von Mitteilungsfeldern auf verschiedenen Seiten. Durch den im WebComposition Modell beschriebenen Ansatz konnte der Wartungsaufwand auf das Ändern weniger Komponenten reduziert werden.

Das entwickelte Laufzeitsystem legt eine Grundlage für das Web Engineering, also die Voraussetzung für die Anwendbarkeit von Softwaretechnik für Entwicklung, Betrieb und Wartung von Anwendungen im World-Wide Web. Die durch das System zur Verfügung gestellten Möglichkeiten müssen dazu genutzt werden den Anwendungsentwickler durch eine zu integrierende Entwicklungsumgebung während des gesamten Lebenszyklus einer Webanwendung zu unterstützen. Dies impliziert, daß Methoden, Werkzeuge und weitere Mittel der Softwaretechnik auf ihre Anwendbarkeit bezüglich eines disziplinierten Web Engineerings untersucht werden müssen. Aktuelle Untersuchungen behandeln Fragen aus dem Bereich der Wiederverwendung von Komponenten sowie die Beschreibung von Komponenten. Für die Wiederverwendung von Lösungsansätzen oder Komponenten prüfen wir insbesondere den Einsatz von Design Patterns [GHJV95]. Im Rahmen eines Projektes zur systemunabhängigen Beschreibung von Komponenten definieren wir eine Sprache auf Basis von XML [Cons97] und erforschen mögliche XML-Anwendungen. In einem Kooperationsprojekt mit Hewlett-Packard evaluieren wir den Nutzen von Komponenten zur automatischen Generierung von Webserviceangeboten und zur Lösung von Internationalisierungsproblemen.

Literatur

[Bade97] Baden-Wuerttemberg. Web-Server des Landes Baden-Wuerttemberg. In *http://www.baden-wuerttemberg.de/*, Januar 1997.

[Cons97] World Wide Web Consortium. W3C architecture domain: Extensible Markup Language (XML). In *http://www.w3.org/XML/*, April 1997.

[DIMG95] A. Diaz, T. Isakowitz, V. Maiora und G. Gilabert. RMC: A Tool To Design WWW Applications. In *The World Wide Web Journal, Issue One*, Dezember 1995.

[fSAa95] National Center for Supercomputing Applications at the University of Illinois at Urbana-Campaign. NCSA: Tutorial on Server-Side Includes. In *http://hoohoo.ncsa.uiuc.edu/docs/*, September 1995.

[Gaed97] M. Gaedke. ActiveX: Componentware im Internet. *Theorie und Praxis der Wirtschaftsinformatik* 34(197), September 1997.

[Gell97] H.-W. Gellersen. Web Engineering: Softwaretechnik fr Anwendungen im World-Wide Web. *Theorie und Praxis der Wirtschaftsinformatik* 34(196), Juli 1997.

[GeWG97] H.-W. Gellersen, R. Wicke und M. Gaedke. WebComposition: An Object-Oriented Support System for the Web Engineering Lifecycle. In *Proc. of the 6th International World-Wide Web Conference (WWW6)*, Santa Clara, April7–12 1997.

[GHJV95] E. Gamma, R. Helm, R. Johnson und J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. In *Addison-Wesley Publishing Company, Reading MA, USA*, Dezember 1995.

[HaSc94] F. Halasz und M. Schwartz. The Dexter Hypertext Reference. In *Communications of the ACM, 37(2):30-39*, Februar 1994.

[IsSB95] T. Isakowitz, T. Stohr und A. Balasubramanian. RMM: A Methodology for Structured Hypermedia Design. In *Communications of the ACM*, August 1995.

[Lerd98] R. Lerdorf. Server Parsed Scripting language with RDBMS support. In *http://modules.apache.org/*, Februar 1998.

[Micr98a] Microsoft. FrontPage Home Page. In *http://www.microsoft.com/frontpage*, Februar 1998.

[Micr98b] Microsoft. Using Active Server Pages with Microsoft Internet Information Server. In *http://www.microsoft.com/iis*, Februar 1998.

[Soft96] NeXT Software. WebObjects Developers Guid. Januar 1996.

[UnSm91] D. Ungar und R. B. Smith. SELF: The Power of Simplicity. *LISP and symbolic Computation: An International Journal* 4(3), 1991.