UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft
Fakultät für Elektrotechnik, Informatik und Mathematik

Existierende Systeme II: Generierende Ansätze

Ausarbeitung zum Seminar der Projektgruppe

Generierung von Web-Anwendungen aus visuellen Spezifikationen

Jens Siebert

Paderborn, den 25. Juli 2004

Inhaltsverzeichnis

Abstrakt

Web-Anwendungen werden heute meist mit Hilfe von integrierten Entwicklungsumgebungen erstellt. Diese unterstützen einen Entwickler während seiner Arbeit und helfen ihm auch bei größeren und komlexen Projekten den Überblick nicht zu verlieren. Ein anderer Ansatz zur Erstellung von Web-Anwendungen sind Generatoren. Hier wird die Web-Anwendung mit Hilfe einer Spezifikation entworfen und anschließend automatisch als Prototyp durch den Generator erstellt. In dieser Ausarbeitung werden drei bereits existierende Generator-Systeme für Web-Anwendungen, ProGUM-Web, T-WEB und BioPro, vorgestellt. Dabei wird auch auf die Besonderheiten der einzelnen Systeme eingegangen.

1 Einleitung

Mit der starken Verbreitung des Internets hat auch die Nutzung von Webbasierten Anwendungen zugenommen. Mit der Hilfe von Web-Anwendungen kann ein Anbieter von Dienstleistungen seine Dienste über das Internet den Benutzern zur Verfügung stellen. Da diese Form der Bereitstellung von Dienstleistungen immer populärer wird, werden die bereit gestellten Anwendungen immer komplexer. Um die Komplexität bei der Entwicklung solcher Anwendungen möglichst gering zu halten, gibt es verschiedene Ansätze. Eine sehr häufig genutzte Methode zur Entwicklung von Web-Anwendungen basiert auf so genannten Integrated Development Environments (IDEs). Solche IDEs ermöglichen es, das Grundgerüst einer Web-Anwendung automatisch erstellen zu lassen. Die Erstellung kann dabei durch Parameter, die der Entwickler anpassen kann, gesteuert werden. Auch behält der Entwickler durch eine grafische Darstellung der Struktur der Anwendung die Übersicht über das gesamte Projekt. Beispiele für solche IDEs sind IBM WebSphere Application Developer [1], Oracle JDeveloper [2] oder SUN Java Studio [3].

Neben Code für die grafische Darstellung der Anwendung im Browser des Benutzers enthält eine Web-Anwendung Code für die Ausführung der eigentlichen Aufgaben der Anwendung. Dieser Code enthält meist Teile, die in vielen verschiedenen Anwendung immer wieder vorkommen. Dazu gehört zum Beispiel der Code für eine Datenbankabfrage oder Code zur Berechnung der grafischen Darstellung oder der Steuerung der Anwendung. Häufig kommt dieser Code mehrfach in einer Anwendung vor, so dass es sinnvoll ist, ihn möglichst automatisch erzeugen zur lassen, bzw. in Bibliotheken zusammen zu fassen. Diese automatische Erzeugung wird heute schon von den meisten IDEs bewerkstelligt.

Ein anderer, weit verbreiteter, Ansatz ist die Verwendung von so genannten Frameworks. Diese Frameworks fassen immer wieder auftauchenden Code einer Web-Anwendung zu einer Bibliothek zusammen deren Funktionen vom Entwickler der Anwendung benutzt werden können. Die Funktionen sind dabei so allgemein gehalten, dass sie zwar eine bestimmte Aufgabe erfüllen, jedoch durch Parameter steuerbar bleiben. Neuere Frameworks unterstützen den Entwickler bei der Erstellung von Anwendungen nach dem Model-View-Controller Prinzip. Diese Technik ermöglicht eine strikte Trennung zwischen der Darstellung (View), der steuernden Anwendung (Controller) und der Datenbasis (Model). Dies ermöglicht neben der einfacheren Wartung der Anwendung auch die Tren-

nung nach Entwicklungsaufgaben. So kann die Darstellung von einem Web- oder Grafik-Designer mit wenig Kenntnissen über die Entwicklung von Web-Anwendungen erstellt werden, während der eigentliche Ablauf der Anwendung durch einen Programmierer erstellt wird. Existierende Frameworks sind Struts [4] oder das Spring-Framework [5]. Viele dieser Frameworks lassen sich in IDEs integrieren, so dass auch solche Anwendungen mit Unterstützung durch Werkzeuge erstellt werden können.

Der in dieser Ausarbeitung betrachtete Ansatz ist die Erstellung von Web-Anwendungen durch Generierung. Bei dieser Technik wird die Anwendung durch eine Spezifikation erstellt, mit deren Hilfe ein Generator einen Prototyp der Anwendung generiert. Anschließend kann dieser Prototyp bis zur endgültigen Anwendung weiter entwickelt werden. Im folgenden Abschnitt wird die Technik der Generierung von Web-Anwendungen genauer erläutert. Anschließend werden drei bereits existierende Generator-Systeme mit unterschiedlichen Ansätzen vorgestellt.

2 Generierung von Web-Anwendungen

Zur Generierung von Anwendungen existieren heute bereits viele Generator-Systeme. Diese System arbeiten alle nach den gleichen Prinzipien.

Zunächst wird eine textuelle Spezifikation der zu erstellenden Anwendung geschrieben. Der Generator bekommt diese Spezifikation als Eingabe und generiert daraus einen Prototypen der Anwendung in der gewünschten Ziel-Programmiersprache. Dieser Ansatz bietet zunächst einige Vorteile, hat aber auch gewisse Nachteile.

Ein Vorteil dieses Ansatzes ist, dass der generierte Code keine Fehler enthält, sofern der Generator korrekt arbeitet. In der fertig gestellten Anwendung auftretende Fehler können also nur noch durch Code hinzugekommen sein, den ein Entwickler in die generierte Anwendung integriert hat. Außerdem ist der generierte Code meist so erstellt worden, dass er möglichst leicht wartbar ist um spätere Änderungen an der Anwendung zu vereinfachen.

Dieser Ansatz hat jedoch auch den Nachteil, dass die Spezifikationssprachen in der Regel sehr komplex und schwer zu erlernen sind. So muss sich ein Entwickler, der einen solchen Generator einsetzten möchte, zunächst in die Spezifikationssprache einarbeiten.

Um diesen Einarbeitungsaufwand zumindest etwas zu verringern, kann eine solche Spezifikation in einer visuellen Spezifikationssprache erstellt werden. Bei diesem Ansatz wird die Spezifikation der zu generierenden Anwendung mit Hilfe von grafischen Symbolen und Konstrukten erstellt.

Durch diesen Ansatz sind auch weniger erfahrene Entwickler in der Lage Generator-Systeme einzusetzen, da solche visuellen Spezifikationssprachen in der Regel leicht zu erlernen sind. Außerdem verbessert dieser Ansatz auch die Kommunikation mit Entwicklern, die sich in der Spezifikationssprache nicht auskennen, da die Struktur einer Anwendung und die zentralen Abläufe meist leicht zu erkennen sind.

Weiterhin wird der Entwickler bei der Erstellung der visuellen Spezifikation von dem System unterstützt. So kann der visuelle Editor, mit dem die Spezifikation erstellt wird, den Entwickler auf eventuell vorhandene Fehler schon während des Entwurfs der Anwendung hinweisen bzw. bestimmte Konstrukte

der visuellen Sprache, die zu Fehlern in der Anwendung führen würden, nicht zulassen.

Der Ansatz der Generierung von Anwendungen aus visuellen Spezifikationen lässt sich auch auf den Bereich der Web-Anwendungen übertragen.

Für die Generierung von Web-Anwendungen werden dem Generator-System neben der Spezifikation als Eingabe noch zusätzliche Templates übergeben. Diese können zum Beispiel immer wiederkehrende Design-Elemente der einzelnen Webseiten, wie Firmen-Logos, enthalten.

Mit der Spezifikation und den Templates als Eingabe kann der Generator nun einen Prototyp der Web-Anwendung generieren, mit dem bereits erste Tests durchgeführt werden können. Meist muss die generierte Anwendung jedoch noch durch weiteren Code ergänzt werden, der nicht vom Generator erzeugt werden kann.

Es existieren heute bereits viele solcher Generator-Systeme speziell für Web-Anwendungen. Viele dieser Systeme verfolgen den Ansatz, eine Spezifikation, die mit der Modellierungssprache UML erstellt wurde, in den Prototypen einer Web-Anwendung umzusetzen. Im folgenden werde drei solcher Systeme vorgestellt, von denen lediglich eines diesen UML-Ansatz verwendet. Die anderen Systeme verwenden eigene visuelle Darstellung der zu generierenden Web-Anwendung.

3 Vorstellung der betrachteten Systeme

In diesem Abschnitt werden drei existierende System zur Generierung von Web-Anwendungen vorgestellt.

Das System ProGUM-Web [6], das an der Universität Paderborn entwickelt wurde, verwendet die Modellierungssprache UML um die Spezifikation der zu generierenden Web-Anwendung zu erstellen. Die Zielsprache dieses Systems ist die Skriptsprache PHP.

Das T-Web-System [7] des Tokyo Institute of Technology benutzt so genannte Web-Transition-Diagramme zur Darstellung der Web-Anwendung. Die erstellte Anwendung benutzt die JSP/Servlet-Technologie der Firma SUN Microsystems.

BioPro [8] ist ein Generator, welcher an der Universität Tokushima entwickelt wurde. Dieser Generator benutzt einen Bild-basierten Ansatz mit Komponenten zur Darstellung der Anwendung. Die Zielsprache dieses Systems ist ebenfalls Java.

3.1 ProGUM-Web

Das ProGUM-Web-System verwendet einen Workflow-basierten Ansatz für die Entwicklung von Web-Anwendungen (siehe Abbildung 1).

Im Modelling-Workflow wird die Web-Anwendung mittels eines UML-Werkzeuges, wie z.B. Borland Together, modelliert. Das so entstandene Modell der Anwendung wird anschließend in das UML-Austauschformat XMI exportiert. In dieser Form kann das Modell in den ProGUM-Web-Generator importiert werden. Dieser Generator arbeitet gleichzeitig als ein Repository, welches die Dateien des Projektes zentral verwaltet. So wird zusätzlich Teamarbeit durch den Generator unterstützt.

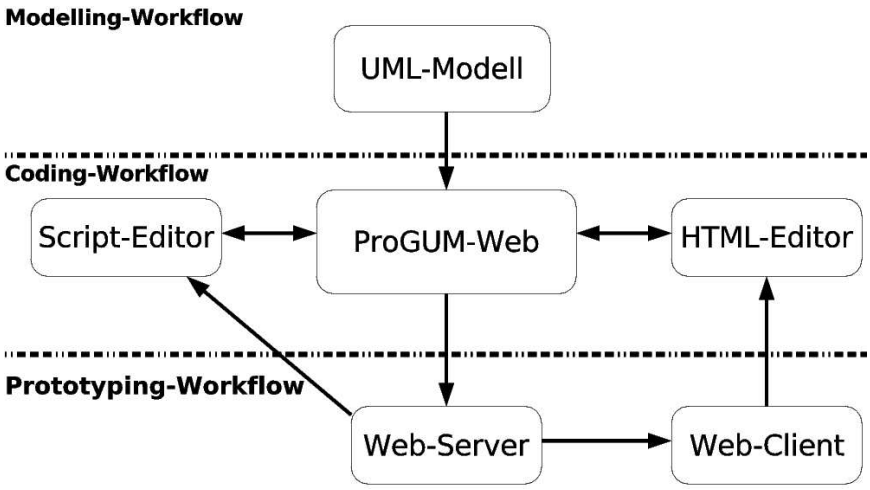

Abbildung 1: Aufbau des ProGUM-Web-Systems

Innerhalb des Coding-Workflow wird die Anwendung entwickelt. Dazu können die Entwickler zusätzliche Templates, wie HTML-Dateien, Grafiken oder PHP-Skripte in das Repository einpflegen und mit den entsprechenden Teilen der Anwendung verknüpfen.

Im darauf folgenden Prototyping-Workflow wird die Anwendung durch den ProGUM-Web-Generator aus den im Repository vorhandenen Teilen erstellt. Dieser Anwendungs-Prototyp kann anschließend auf einem Web-Server installiert und getestet werden. Bei diesen Tests auftretende Fehler können dann sofort im Repository korrigiert und ein neuer Prototyp erstellt werden. So entsteht ein Kreislauf zwischen dem Coding-Workflow und dem Prototyping-Workflow.

3.2 T-WEB-System

Das T-WEB-System verwendet so genannte Web-Transition-Diagramme, wie in Abbildung 2 zu sehen, um eine Web-Anwendung zu spezifizieren. Diese Web-Transition-Diagramme bestehen aus Knoten und Kanten, wobei es unterschiedliche Typen von Knoten und Kanten gibt.

Die beiden wichtigsten Kantentypen sind der einfache Hyperlink, sowie der Dataflow-Link. Eine Hyperlink-Kante entspricht einem HTML-Hyperlink, welcher auf eine andere Web-Seite verweist. Der Dataflow-Link entspricht einem HTTP-Submit. Hierbei werden Daten, zum Beispiel aus einem Formular, vom Web-Browser zu einem Web-Server geschickt, welcher die Daten entsprechend auswertet. In einem Web-Transition-Diagramm werden die zu übermittelnden Daten durch die Beschriftung an der Dataflow-Link-Kante repräsentiert.

Ein Web-Transition-Diagramm kann eine Vielzahl unterschiedlicher Knotentypen enthalten. Die Wichtigsten sind hier die Fixed Web-Page-Node, die Output Web-Page-Node, die Server-Program-Node, die Database-Node, sowie die Client-Browser-Node.

Die Fixed Web-Page-Node und die Output Web-Page-Node repräsentieren unterschiedliche Typen von Web-Seiten. Eine Fixed Web-Page-Node entspricht

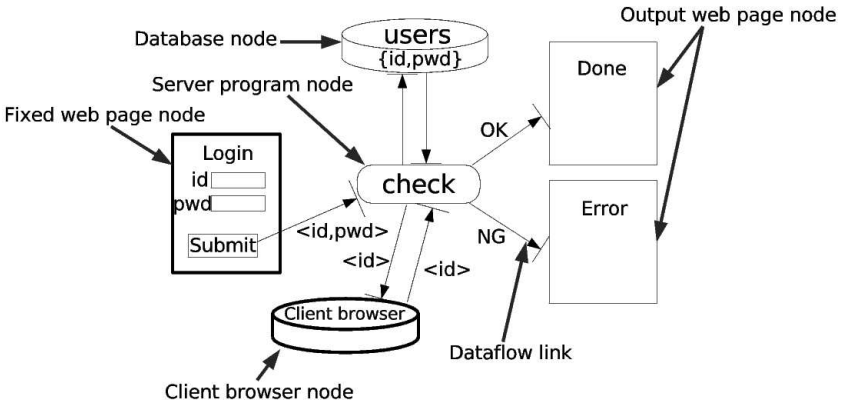

Abbildung 2: Beispiel für ein Web-Transition-Diagramm

dabei einer statischen Web-Seite, die zur Laufzeit der Anwendung nicht mehr verändert wird. Die Output Web-Page-Node repräsentiert eine, zumindest teilweise, durch ein serverseitiges Programm generierte Web-Seite.

Die Server-Program-Node steht für ein serverseitiges Programm. Im Falle des T-WEB-Systems sind hiermit Java-Programme gemeint. Diese Programme verarbeiten die Daten, welche durch einen Dataflow-Link übermittelt werden und generieren, basierend auf den Ergebnissen der Verarbeitung, eine neue Web-Seite.

Die Database-Node repräsentiert eine Datenbanktabelle. Durch setzen entsprechender Dataflow-Links kann mit diesem Knotentyp eine Datenbankanfrage modelliert werden.

Der Web-Client des Benutzers der Web-Anwendung wird durch die Client-Browser-Node repräsentiert. Hiermit kann unter Anderem die Speicherung von Daten im Web-Client mittels Cookies modelliert werden.

Es stehen noch viele weitere Knotentypen für die Web-Transition-Diagramme zur Verfügung, auf die ich aber hier nicht näher eingehen möchte. Eine Beschreibung dieser Knotentypen findet sich in [7].

Das T-WEB-System setzt sich aus den Komponenten Generator und Editor zusammen (siehe Abbildung 3). Im T-WEB-Editor wird die Web-Anwendung mit Hilfe der Transition-Web-Diagramme modelliert. Der Editor prüft bereits bei der Eingabe die Konsistenz der Anwendung. Wird zum Beispiel an einem Dataflow-Link ein Parameter definiert, der in der Quellseite nicht existiert, so erkennt der Editor dies als Fehler und der Benutzer wird darauf hingewiesen, dass die Anwendung inkonsistent ist.

Im Editor können bereits vorhandene Templates mit den einzelnen Knotentypen verknüpft werden. So kann zum Beispiel ein Fixed Web-Page-Knoten mit einer bereits existierenden HTML-Seite verknüpft werden oder ein Server-Program-Knoten mit einem existierenden Java-Servlet.

Der T-WEB-Generator erstellt aus der Beschreibung des Web-Transition-Diagramms und den vorhandenen Templates anschließend den Prototypen der modellierten Web-Anwendung.

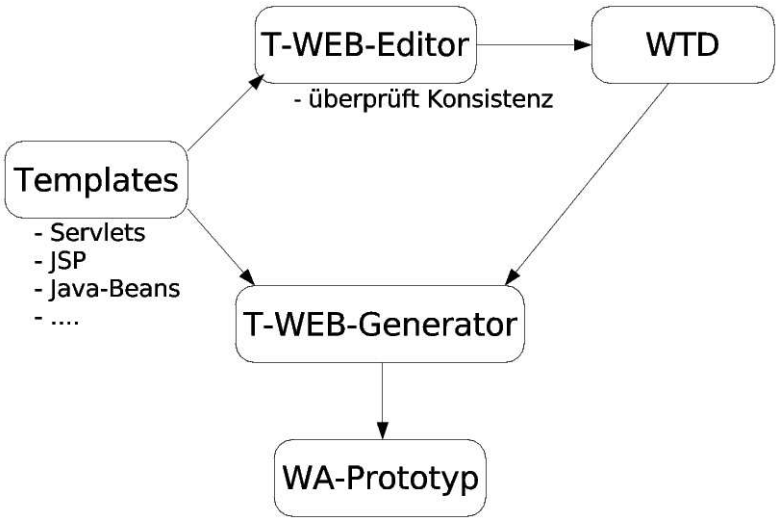

Abbildung 3: Aufbau des T-Web-Systems

3.3 BioPro-System

Das BioPro-System verwendet einen visuellen, bildorientierten Ansatz zur Spezifikation einer Web-Anwendung (siehe Abbildung 4).

Im BioPro-Editor wird zwischen drei grundlegenden Elementen für die Erstellung einer Web-Anwendung unterschieden. Zum Einen existieren die so genannten Programm-Tabellen. Diese dienen als Container für Daten, welche in der Web-Anwendung verwendet werden können. Diese Daten sind in Tabellenform, wie in einer Datenbank, abgelegt. Die Programm-Tabellen werden entweder vom Entwickler der Web-Anwendung entworfen und mit Daten gefüllt, oder können auch aus einer bereits bestehenden Datenbank erstellt werden. Im ersten Fall erstellt der BioPro-Generator ein entsprechendes Java-Objekt, welches die vom Entwickler festgelegten Daten, sowie Zugriffsmethoden für diese Daten enthält. Wird eine Programm-Tabelle aus einer existierenden Datenbank erstellt, so generiert der Generator die entsprechenden Datenbankabfragen und Zugriffsmethoden für die Daten.

Die eigentlichen Web-Seiten werden im Web-Page-View entworfen. Diese Ansicht ist eine Darstellung, die der fertigen Web-Seite bereits sehr ähnlich sieht. Die Gestaltung der Web-Seite wird mit Komponenten, wie Textfeldern oder Schaltflächen realisiert. Daten aus den Spalten einer Programm-Tabelle werden durch Platzhalter in die Web-Seite eingefügt. Diese Platzhalter sind im Editor durch eine Linie mit der entsprechenden Spalte der Programm-Tabelle verknüpft. Der entsprechende Code für den Zugriff der Web-Seite auf die Daten in der Programm-Tabelle wird durch den Generator erstellt.

Zu jeder Web-Seite im BioPro-System gehört eine so genannte Web-Source. Hier wird der Code eingefügt, welcher nicht vom Generator erzeugt werden kann. Dies kann zum Beispiel Code für eine Berechnung sein.

Der Aufbau einer Anwendung, welche mit dem BioPro-System erstellt wurde, entspricht dem Model-View-Controler-Prinzip, wie in Abbildung 5 darge-

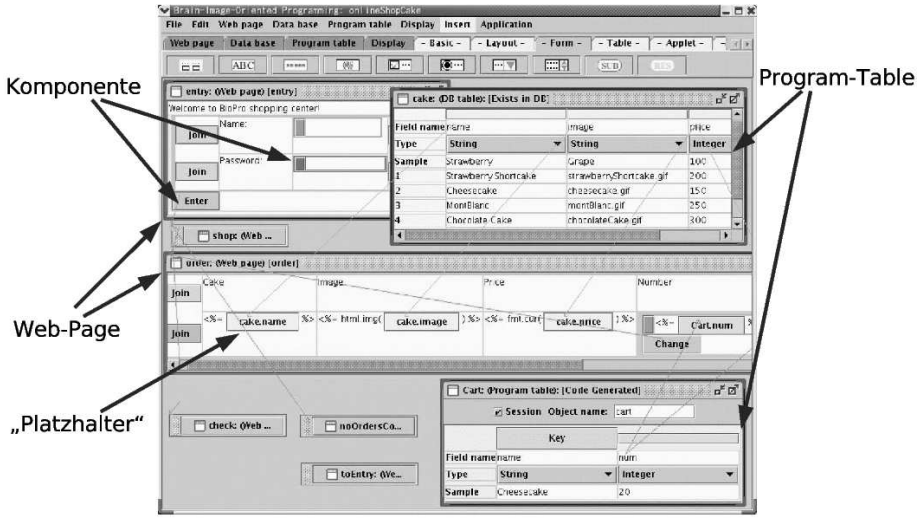

Abbildung 4: Beispielsitzung des BioPro-Systems

stellt. Das Datenmodell besteht bei einer solchen Anwendung aus den Programm-Tabellen, bzw. Datenbank-Tabellen. Die View wird durch die im BioPro-Editor entworfenen Web-Pages dargestellt. Als Controller dient schließlich die zu jeder Web-Page vorhandene Web-Source, welche den generierten und den vom Entwickler eingefügten Anwendungscode enthält.

Eine Besonderheit des BioPro-Systems sind die so genannten „Actions”. Hierbei handelt es sich um eine Technik, mit deren Hilfe man Aktionen für den Zugang zu einer Web-Seite festlegen kann. Welche Aktion dabei ausgeführt wird, ist davon abhängig, von welcher Quell-Seite der Benutzer auf die Ziel-Seite zugreift. Abbildung 6 soll dies verdeutlichen. Ein Anwendungsbeispiel wäre, eine für nicht registrierte Benutzer gesperrte Web-Seite. Würde ein Benutzer, der sich noch nicht bei der Web-Anwendung registriert oder angemeldet hat, versuchen auf die gesperrte Web-Seite zuzugreifen, so würde er von der entsprechenden Aktion zunächst auf eine Anmelde-Seite weitergeleitet. Versucht jedoch ein angemeldeter Benutzer von einer anderes Web-Seite innerhalb der Web-Anwendung auf die zugriffsbeschränkte Web-Seite zuzugreifen, so wird die entsprechende Aktion dies registrieren und den Benutzer zur Ziel-Seite weiterleiten.

4 Zusammenfassung

Die hier betrachteten Systeme zur Generierung von Web-Anwendungen enthalten viele interessante Konzepte für die Erstellung von Web-Anwendung mittels visueller Spezifikation.

Zunächst fällt auf, dass die Verwendung der Modellierungssprache UML kein Allheilmittel für die Spezifikation von Web-Anwendungen ist. UML bietet zwar den Vorteil, dass sich heutige Software-Entwickler meist schon in der Verwendung dieser Sprache auskennen und so lange Einarbeitungszeiten entfallen, jedoch ist UML nicht speziell auf die Spezifikation von Web-Anwendungen ausge-

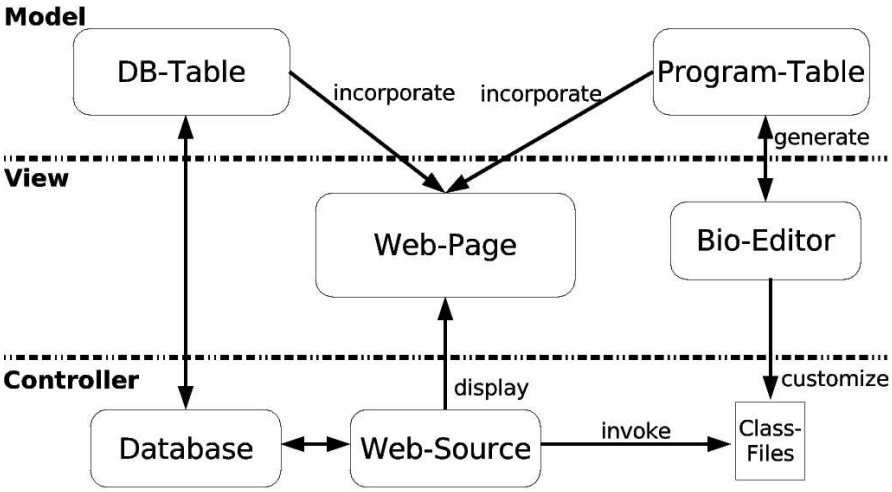

Abbildung 5: Aufbau einer BioPro-Anwendung

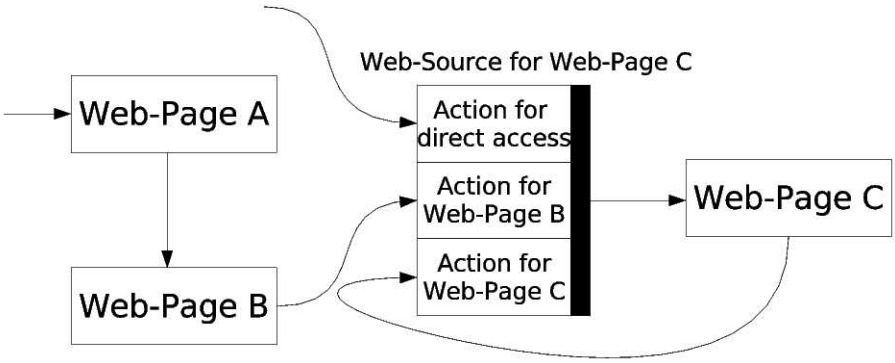

Abbildung 6: Beispiel für Actions im BioPro-System

legt. So kommt es vor, dass einige bestimmte Konzept und Konstrukte aus dem Bereich der Web-Anwendungen nur schwer oder gar nicht mit UML dargestellt werden können. Daher bietet es sich an, sich an anderen Konzepten, wie den Web-Transition-Diagrams oder dem visuellen Ansatz von BioPro zu orientieren. Da beide Systeme speziell für die Spezifikation von Web-Anwendungen ausgelegt sind, lassen sich mit ihren visuellen Sprachen wesentlich exaktere Spezifikationen erstellen. Auch lässt sich mit diesen Systemen besserer Code generieren, da die Generatoren wesentlich besser auf die visuelle Sprache abgestimmt werden können.

Weitere interessante Konzepte finden sich in den Editoren der hier betrachteten Systeme, beziehungsweise in den einzelnen Generatoren. Eines dieser Konzepte sind die Konsistenzüberprüfungen im Editor des T-WEB-Systems. Diese Überprüfungen führen zu weniger Fehlern in einer Anwendung, da der Entwickler bereits während des Entwurfs der Anwendung auf eventuelle Fehler aufmerksam gemacht wird und diese korrigieren kann.

Die Technik der „Actions" im BioPro-System zählt ebenfalls zu diesen interessanten Konzepten. Mittels dieser Technik lässt sich der Seitenfluss einer Web-Anwendung einfach und vor allem übersichtlich in Abhängigkeit von verschiedenen Zuständen modellieren.

Viele der hier vorgestellten Konzepte können als Inspiration für eigene Ideen verwendet werden. Insbesondere für das im Rahmen der Projektgruppe zu erstellende System, können einige dieser Ideen sicherlich aufgegriffen werden.

Literatur

[1] IBM WebSphere Application Developer, http://www-306.ibm.com/software/awdtools/studioappdev/

[2] Oracle JDeveloper, http://otn.oracle.com/products/jdev/index.html

[3] SUN Java Studio, http://wwws.sun.com/software/sundev/jde/index.html

[4] Apache Struts Framework, http://struts.apache.org

[5] Spring Framework, http://www.springframework.org

[6] M. Lohmann, S. Sauer, T.Schattkowsky, ProGUM-Web: Tool Support for Model-Based Development of Web Applications, Universität Paderborn, 2003

[7] K. Jamroendararasame, T. Suzuki, T. Tokuda, A Diagram Approach to Automatic Generation of JSP/Servlet Web Applications, Tokyo Institute of Technology

[8] T. Shimomura, A page-transition framework for image-oriented Web programming, University of Tokushima