15.7.2003

Abschlußbewertung

I. Allgemein

Als ein sinnvolles Softwareprojekt nahmen wir ein Content-Management-System in Angriff, das im Laufe der Projektdauer in ein verwendbares Redaktionssystem mit Veröffentlichungskomponente mutierte.

II. Erfahrungen mit der Gruppenarbeit

Bei diesem Projekt hatten wir eine unschöne personelle Krise zu bewältigen, was uns eine Menge Mehrarbeit und lange Nächte bescherte. Da wir gruppeninterne Meilensteine festgelegt hatten, deren Einhaltung wir im Hinblick auf ein gutes Gelingen des Projektes hohes Gewicht beimaßen, trafen wir die Entscheidung, die nicht geleisteten Anteile einiger Gruppenmitglieder selbst zu übernehmen. Durch dieses Vorgehen konnten wir unser Projekt - mit einigen Abstrichen in Umfang und Funktionalität –zu einem guten Ende führen. Während der Implementierung bekamen wir personelle Verstärkung, sodaß am Ende der beiden Semester doch ein brauchbares und einsatzfähiges CMS als Ergebnis der Projektarbeit entstanden war.

III. Auswirkung von Architektur- und Implementierungsentscheidungen

- Die Entscheidung, die Objektlisten in der Oberfläche zu halten und für die Anzeige und Manipulation der Objekte zu benutzen hat sich als nicht gut herausgestellt, da es auf diese Weise zu Inkonsistenzen zwischen dem Inhalt einer Liste und dem Zustand der Datenbank kommen kann. Dies führte im späteren Projektverlauf immer wieder zu Problemen, die teilweise umständlich umgangen werden mußten. Es wäre vermutlich einfacher gewesen, jedes Objekt zum Anzeigen und Ändern direkt aus der Datenbank zu holen.
- Die Verwendung typsicherer Listen war in unserem Fall etwas umständlich, da weder Struts noch Castor mit typsicheren Listen umgehen können. Für die Oberfläche wäre hier eine eigene Taglib (z.B. für das Iterieren der Listen auf der Oberfläche) sinnvoll, deren Umsetzung jedoch zu aufwendig gewesen.
- Der Einsatz des Erbauermusters in der Oberflächenschicht erwies sich als sinnvoll, da künftige Erweiterungen des Systems, wie etwa die Generierung von WAP-Seiten, relativ unkompliziert zu realisieren sind.

IV. Erfahrungen mit den eingesetzten Techniken

- **Jakarta-Struts** **1.0.2**
  Struts erleichtert das Leben sehr, da es viel Arbeit abnimmt. Die Grundlagen sind dank unzähliger Einführungen und Tutorien im Netz einfach zu erlernen, so daß der Einstieg in dieses MVC-Framework nicht schwerfällt. Dennoch ist zu empfehlen, sich mit der prinzipiellen Funktionsweise von Web-Anwendungen (http, Servlets, HTML etc.) gründlich vertraut zu machen, denn dieses Wissen ist unentbehrlich für die Entwicklung einer umfangreicheren Web-Applikation. So einfach der Einstieg in Jakarta-Struts ist, hat sich doch herausgestellt, daß die frühzeitige Einarbeitung in die verwendeten Techniken sehr sinnvoll war. Die Struts-Taglib war für unsere Zwecke ausreichend. Hervorzuheben ist hier, daß die Taglib von Struts zwar für manche Aufgaben etwas umständlich, jedoch wesentlich einfacher zu verwenden und weniger kompliziert ist als beispielsweise JSTL.

- **JavaScript**
  JavaScript erwies sich als unentbehrlicher Helfer, wenn es darum ging, die Bedienung einer Web-Oberfläche komfortabel zu gestalten. Allerdings sind hier Mehrbrowser-Tests unumgänglich, da einige Browser sehr eigene Auffassungen haben, wie JavaScript zu interpretieren ist.
- **Castor**
  ist ein einfach zu handhabendes OR-Mappingtools, dessen Verwendung bei einfachen Sachverhalten akzeptabel ist. Das Tool bietet flexible Einstellung der Mappingregeln. Bei komplexen Sachverhalten (Mehrfachvererbung und Polymorphie und rekursive Assozitionen) scheint es jedoch angeraten, ein kommerziell entwickeltes Tool zu verwenden. Wenn man mit den Castortypischen Eigenschaften und Schwächen vertraut ist und diese beim Softwareentwurf im „Hinterkopf" behält, kann man weniger komplexe Projekte sicher realisieren.
- **XSL-Stylesheetprocessing**
  ist eine mittlerweile verbreitete und bewährte Technik, die zur dynamischen Erzeugung verschiedenster Dokumentenformate aus XML verwendet wird. Abgesehen von der geringen Fehlertoleranz der XSLT-Prozessoren erscheint diese Technik robust und flexibel und lässt sich in Java gut realisieren.

V. Fazit

Während dieses Projektes machte sich unsere mangelnde Erfahrung mit Gruppen- und Projektarbeit bemerkbar. Eine vorausgehende und/oder begleitende Schulung in Projektorganisation wäre hier vermutlich hilfreich gewesesen. Ebenfalls hilfreich wäre, die in Projekten immer wieder zum Einsatz kommenden Techniken zumindest in Wahlpflichtveranstaltungen zu lehren.
Abschließend bleibt zu bemerken, daß wir Projektarbeit grundsätzlich als sehr sinnvoll ansehen. Solche Projekte sollten, in kleinerem Umfang, viel häufiger und spätestens mit Beginn des Hauptstudiums regelmäßig durchgeführt werden (Tip: weniger Klausuren, mehr Projekte!). SWP ist mit Sicherheit DIE sinnvollste Lehrveranstaltung des gesamten Studiums. Jedoch war der Arbeitsaufwand im Vergleich zum Lerneffekt unverhältnismäßig hoch.