Thesenpapier
CampusSource Developer Forum
O. Ehlert, 10.12.2002

Mit der steigenden Nutzung von objektorientierten und komponentenorientierten Modellierungen und Entwicklungsumgebungen sowie der breiten Nutzung von Open-Source für Projekte und Publikation deren Entwicklungsmethodiken hat sich der Prozeß der Softwareentwicklung in den letzten Jahren kontinuierlich verändert.

Trotz inzwischen zahlreichen vorliegenden Praxiserfahrungen ist Softwareentwicklung für das Themengebiet E-Learning wenig dokumentiert, ein Erfahrungsaustausch über die Entwicklungsprozesse ist daher erforderlich. Er sollte helfen den Bedarf, die Rahmenbedingungen und möglichen Dienstleistungen sowohl für den Entwicklungsprozeß als auch für Open-Source-Plattformen zu eruieren und zu fixieren.
Eine wichtige Frage ist es, inwieweit die vorhandenen Entwicklungsprozesse für das Umfeld E-Learning geeignet sind und welche spezifischen Besonderheiten berücksichtigt werden müssen.

Thesen zur aktuellen Softwareentwicklung im Bereich E-Learning:
- Eine Vielzahl von Randbedingungen, auch nicht technischer Art, muß in den Entwicklungsprozeß integriert werden (Nutzungsrandbedingungen, Einführungsstrategien, Personalfragen).
- Vielfache Parallelentwicklung von Funktionalitäten und resultierende ungenutzte Synergien
- Differenzierte und komplexe Funktionen bedingen, dass zur Nutzung der Synergien vielschichtige Modelle mit verschiedenen Realisierungsebenen bzw. Abstraktionsebenen implementiert werden müssen (z.B. Datenbankschnittstelle, GUI-Entwicklung, User-Verwaltung).
- Einzelne Funktionalitäten lassen sich ungleich effizient auf den unterschiedlichen Abstraktionsebenen realisieren (z.B. hätte eine gemeinsame Funktion für Internationalisierungsdienste auf Realisierungsebene von Webservices zuviel Overhead)
- Aktuelle Abstraktionsmechanismen ermöglichen erhöhte (maximale) Unabhängigkeit der Entwicklertätigkeit durch hohe Entkopplung der Implementierungsabhängigkeiten.

Für nachhaltige Projekte sind neben den softwaretechnischen Randbedingungen auch andere (externe) Faktoren maßgebend und müssen demzufolge in den Projektprozeß integriert werden (Standards, Personal, Einsatzumgebung).
Komplexe Arbeitsumgebungen erfordern erhöhten Einarbeitungsaufwand auf Seiten der Entwicklung. Einfach gehaltene Systeme sind bei Personalfluktuation u.U. daher besser geeignet.
Standards und Spezifikationen unterstützen gemeinschaftliche Entwicklungsprozesse, erfordern aber auch eine vorgeschaltete Verifkation auf deren Eignung und Optimierungs-potenzial.

Viele ähnliche oder gleiche Funktionalitäten werden von allen in CampusSource vertretenen Anwendungen benötigt: z.B. Sicherheit / Kryptografie, Verzeichnisdienste / Anwenderver-waltung, Internationalisierung / Lokalisierung, aber z.B. auch Factories, Persistence, Caching, Scheduling, Templates oder einfach nur MIME-Funktionalitäten.
Beispiel: Für die Internationalisierung/Localization wird eine Abstraktionsschicht für Texte und Sprachen benötigt und ist bereits vielfach implementiert.
Beispiel: Für alle Projekte wird eine User-Verwaltung / Verzeichnisdienst benötigt.

Die aufgeführten Teilaspekte und Abstraktionsschichten sind von der Architektur horizontal zur eigentlichen (angenommenen vertikalen) Anwendung ausgerichtet und könnten eigentlich in mehreren Projekten genutzt werden. Voraussetzung ist allerdings der Einsatz

von Entwicklungsmethodiken mit entsprechenden ausdrucksstarken Differenzierungs-möglichkeiten.

Teilweise stehen für vielfach benötigte Funktionen fertige Bibliotheken oder Source-Code als Public-Domain zur Verfügung; folgende Probleme ergeben sich:
- Vielzahl der angebotenen Teilfunktionalitäten, Unübersichtlichkeit des Angebotes
- Fehlende oder aufwändige Beurteilung der Nutzungsqualität für den eigenen Code

Synergien durch direkte gemeinsame Nutzung von Sourcen sind möglich, aber schwierig zu erreichen, Probleme sind z.B.:
- vorhandene Bibliotheken bzw. Komponenten sind für speziell für die jeweilige Anwendungen erstellt und in der Regel nicht generisch entworfen und nutzbar
- heterogene Entwicklungsumgebungen bedingen für gemeinsame Nutzung wiederum Mehrfachimplementierungen (z.B. Jakarta-Turbine-Fulcrum ist Java/Servlet optimiert und kann nicht für PHP-Systeme genutzt werden)
- Teilaspekte/Abstraktionsschichten sind nicht Kernziel der einzelnen Entwicklungsgruppen (Optimum wäre: eine Entwicklergruppe arbeitet konzentriert am Verzeichnisdienst, aber dies erzeugt wiederum häufig nicht gewollte Abhängigkeiten)
- kontinuierlicher Anpassungsaufwand bei neuen Versionen der Teilfunktionen

Die Nutzung von Komponentenstandards (CORBA, DCOM, EJB), ermöglicht Synergien durch gemeinsame Nutzung von fertigen Komponenten/Modulen, z.B. auch aus dem Bereich Open-Source: KDE (DCOP), Mozilla (XPCOM).
Wichtige Voraussetzungen bzw. Probleme sind:
- Durchgängige Nutzung eines einheitlichen Komponentenstandards
- Projektarchitektur muß weitestgehend auf Basis des Komponentenstandards ausgelegt sein
- Projektarchitektur muß sinnvolle Komponentenzerlegung und -schnittstellen aufweisen

Derzeitiger Trend in der Softwareentwicklung ist die lose Kopplung von komplexen Serveranwendungen durch ein einheitliches Protokoll bzw. Schnittstelle (Beispiel anhand Webservices: Einbindung bzw. Nutzung von Legacy-Anwendungen aus dem Bereich Geschäftsprozesse durch Implementierung einer Webservices-Abstraktionsschicht).
Der Hauptvorteil dabei, ist die maximale Entwicklerunabhängigkeit durch die lose Kopplung der Projektteile und Auflösung von Implementierungsabhängigkeiten; sowohl beim Erstentwurf als auch bei der Weiterentwicklung
Wichtige Voraussetzungen bzw. zu realisierende Teilaspekte:
- dynamische Verknüpfung von Teilanwendungen (z.B. über das Internet)
- Kommunikation erfolgt plattformübergreifend (z.B. mit XML)
- Interoperabilität (unabhängig von darunter liegender Architektur)

Exkurs: Synergien durch Nutzung gemeinsamer Schnittstellen oder Protokolle können auf verschiedenen Ebenen/Schichten realisiert werden und müssen aufgrund unterschiedlicher Vor- und Nachteile entsprechend differenziert betrachtet werden:
- Datenbankzugriffsebene, Abstraktion oder gemeinsame Zugriffe
- Komponentenebene (entsprechende Komponententechnologie und Schnittstellenbildung)
- Anwendungsebene, Webservices, XML

Beispiel-Szenario „User-Verwaltung über Verzeichnisdienst/LDAP“:
Für das Szenario ergebt sich für eine gemeinsame Nutzung im wesentlichen das Problem der Modellierung der Datenstruktur und daraus folgend das Problem der Absprache der User-Datenstrukturen (in LDAP).

Weitere Themen und offene Fragen sind (Bereiche: Entwicklung, Content, Support):

Einschätzung der eingesetzten Methodiken und Technologien anhand von Erfahrungsberichten, z.B.:

- Entwicklungsprozeß (Compiler, make/ant, Editoren, CVS)
- Integrierte Entwicklungsumgebungen
- Erfahrungen zum Einarbeitungsaufwand in Entwicklungsumgebungen
- Erfahrungen zur Nutzung von vorhandenen Open-Source-Prozessen (mozilla)
- Integration von ausgabeunabhängigem Content (z.B. Basis XML)
- Unterstützung zur Modularisierung von Content
- Integration der Unterstützung der Lernenden
- Erfahrungsberichte synchrone und asynchrone Supportsysteme
- Erfahrungen zur Einführung von Lösungen

Mit welche Entwicklungsstrategien/-prozessen und Entwicklungsmethodiken liegen gute Erfahrungen vor, wo lagen Probleme?