

Integrative Softwaretechnik braucht Standards

Von Softwareprojekten wird heute ein schneller Return On Investment (ROI) verlangt. Neuentwicklungen werden entsprechend seltener – Vorrang hat die Frage, wie Einbindung und Erweiterung von Bestandssystemen einen kurzfristigen Geschäftswert schaffen können.

Die wichtigste Antwort der Softwaretechnik auf die Forderung nach schnellem ROI sind Bibliotheken, Komponenten und Frameworks. Doch diese einfache Kapseln und Wiederverwenden funktioniert bei Technologiebrüchen zwischen den Systemen – etwa durch verschiedene Programmiersprachen oder strukturell unterschiedliche Anwendungsschnittstellen (APIs) – nur noch aufwändig oder gar nicht mehr. Ein Beispiel ist die Integration von Datenbanken. Si-

cherlich lässt sich dies mit Hilfe von Bibliotheken wie Microsoft's ADO oder Java's JDBC implementieren. Aber der Aufwand, zwischen programm-sprachlichen Konzepten und der SQL-Schnittstelle der Datenbank zu vermitteln, obliegt dem Softwareentwickler – und er ist immens. Deshalb werden schon jetzt wann immer möglich höhere Konzepte verwendet. Im Java-Kontext ist die Objektpersistenz ein zentrales Thema. So werden Sun's Java Data Objects sowie Entity EJBs gerade im Zuge der Entwicklung von EJB 3.0 zu einem einheitlichen API zusammengeführt. Für sämtliche Plattformen gibt es darüber hinaus ausgereifte Persistenz-Frameworks. Ein weiteres, aktuelles Beispiel ist die Erstellung eines Webservice: Bestehender oder neuer Programmcode muss in einen Applikationsserver inte-

griert werden. Beiden Beispielen gemeinsam sind:

- **Abstraktion:** Eine deutliche Beschleunigung und Vereinfachung der Entwicklung ergibt sich aus der Abkehr von spezifischen Techniken wie SQL, Java-Methoden-Aufrufen oder das Protokoll HTTP und der Beschreibung auf semantisch höherer Ebene (Objektpersistenz, sprachunabhängige Schnittstelle).

- **Konkretisierung:** Die Abstraktion muss verlustfrei geschehen; ein konkretes, lauffähiges System ist weitgehend automatisch erzeugbar.

- **Konfiguration:** Bei allen Vorteilen entstehen doch auch wesentliche Kosten durch die Notwendigkeit, die Abstraktion verlustfrei zu halten. Im Falle der Objektpersistenz bedeutet dies eine Beschreibung der objekt-relationalen Abbildung, im Falle von Webservices die Beschreibung von Endpunkten, beides meist in Form von XML (zum Beispiel über WSDL). Aus Sicht der zu integrierenden Systeme handelt es sich bei diesen Informationen um Metadaten.

Allgemeine Sicht birgt auch Fallstricke

Während der Entstehung dieser integrierenden Techniken stand eher jeweils eine einzelne Aufgabe im Vordergrund, die durch Definition eines bestimmten Verfahrens oder Modells gelöst wurde. Dazu waren etliche Konfigurationsdateien in einem bestimmten Format nötig. Erst in jüngster Zeit wird ihr Einsatz vermehrt unter den genannten Aspekten der Modellbildung, Metadaten-Definition und Transformation betrachtet. Diese allgemeinere Perspektive schafft jedoch zwei Probleme:

Whitehorse

Unter der Bezeichnung Whitehorse bringt Microsoft einen Satz von **Modellierungstools** für die Windows-Plattform. In der ersten Implementierung in Visual Studio .NET 2005 liegt der Schwerpunkt auf Webservices, jedoch wurde auch viel Grundlagenarbeit geleistet. Herzstück ist ein allgemeines **Meta-Modellierungs-Framework** analog zur MDA, jedoch wird bei Microsoft eine stärkere Spezifizierung auf die Modellierung branchentypischer Prozesse angestrebt. *fm*

- Die Beschreibung der Transformation – sprich Metadaten in XML-Konfigurationsdateien in einem bestimmten Format – ist aufwändig. Es ist durchaus üblich, WDSL-Dateien im XML-Editor manuell zu bearbeiten. Zwar helfen viele herstellerspezifische Werkzeuge, etwa durch grafische Editoren oder Extraktion von Informationen aus Code. Aber in jedem Fall entstehen Kosten durch Konfiguration und Deployment.

- Das Format von Modell und Metadaten erzeugt selbst wieder eine Abhängigkeit. Zu schnell entstehen Bindungen an bestimmte Applikationsserver oder Persistenz-Frameworks. Beide Probleme können durch standardisierte Modelle und Transformationen wesentlich gelindert werden. Leider ist die Normierung naturgemäß ein langwieriger Prozess. Doch etliche Verfahren sind schon in Entwicklung oder Benutzung und werden entweder durch Marktdurchdringung oder Gremienarbeit Standards setzen. Die Model Driven Architecture (MDA) der OMG (Object Ma-

Entwicklungs-Glossar

- **ADO** (Active X Data Objects): Anwendungsschnittstelle von Microsoft, die unter Windows Zugriff auf Datenbankinhalte eröffnet.

- **JDBC** (Java Database Connectivity): Von Sun entwickeltes Paket, das über die Bereitstellung relationaler Datenbankobjekte und entsprechender Methoden den Zugriff aus Java-Applikationen auf beliebige Datenbanken ermöglicht.

- **JDO** (Java Data Objects): Bieten transparenten Zugriff auf unterschiedliche Datenquellen und machen den JDBC-Einsatz obsolet.

- **EJB** (Enterprise Java Beans): Serverseitige Komponenten zur Implementierung von Businesslogik und Integration von IT-Ressourcen. Kernbestandteil der Java 2 Enterprise Edition (J2EE), einer komponentenbasierten transaktionalen Ablaufumgebung.

- **WSDL** (Webservices Description Language): Beschreibungssprache für Webservices. Letztere repräsentieren IT-Komponenten mit einer plattformunabhängigen Schnittstelle, die Funktionen mit definierten Parametern und Rückgabewerten bietet.

- **XML** (Extensible Markup Language): Eine Metasprache für das Definieren von Dokumenttypen.

- **UML** (Unified Modeling Language): Grafisch orientierte Beschreibungssprache für die Modellierung von Software. *fm*

nagement Group), aufbauend auf UML 2.0 und der MOF (Meta Object Facility), ist sicherlich der allgemeinste und am besten fundierte Ansatz. Jedoch wird die OMG noch auf absehbare Zeit damit beschäftigt sein, einen Standard für allgemeine Modelltransformationen zu definieren. Noch gibt es kein Werkzeug, das tatsäch-

lich UML 2.0 auch auf Modellebene komplett unterstützt. Durch Microsofts Whitehorse-Initiative (siehe Kasten) wird sicher die Modellierung im Windows-Bereich verbreiteter werden. Im Java-Umfeld existieren bereits jetzt einige standardbasierte Tools bereit, welche die Integration verschiedener Techniken ermöglichen – Suns Java

Studio Creator, vormals Project Rave, ist ein Beispiel. Sowohl die Dotnet- als auch die Java-Welt (J2EE) bieten die Möglichkeiten, Metadaten direkt in den Code zu integrieren – als Attributes in Dotnet, als Annotations in Java 5.0. Dies ermöglicht einheitliche Transformationswerkzeuge und erspart dem Softwareentwickler den

Bruch zwischen Code und Konfiguration. Microsoft setzt dabei auf selbstdefinierte Metadaten, während Sun gerade im Rahmen des JCP (Java Community Process) Formate für verschiedene Anwendungsbe-reiche entwickelt.

*Stephen Kelvin,
Senior Software Architect,
Gentleware/fm*

Computer Zeitung 51/2004, Seite 18