Palladio Componentmodel

Entwurfsbeschreibung

Marko Hoyer
Marko.Hoyer@informatik.uni-oldenburg.de

5. Oktober 2005

Inhaltsverzeichnis

1 Einleitung

2 Architektur

In diesem Kapitel wird die derzeitige Architektur des Komponentenmodells vorgestellt. Sie setzt sich aus den in Abbildung 1 dargestellten und im Folgenden kurz erläuterten Bestandteilen zusammen. An der Umrandung der Blöcke ist abzulesen, ob diese bereits entworfen oder lediglich als Erweiterungen geplant sind. Details und Informationen zur Umsetzung der Bestandteile des Komponentenmodells sind Inhalt der folgenden Kapitel dieses Dokuments.

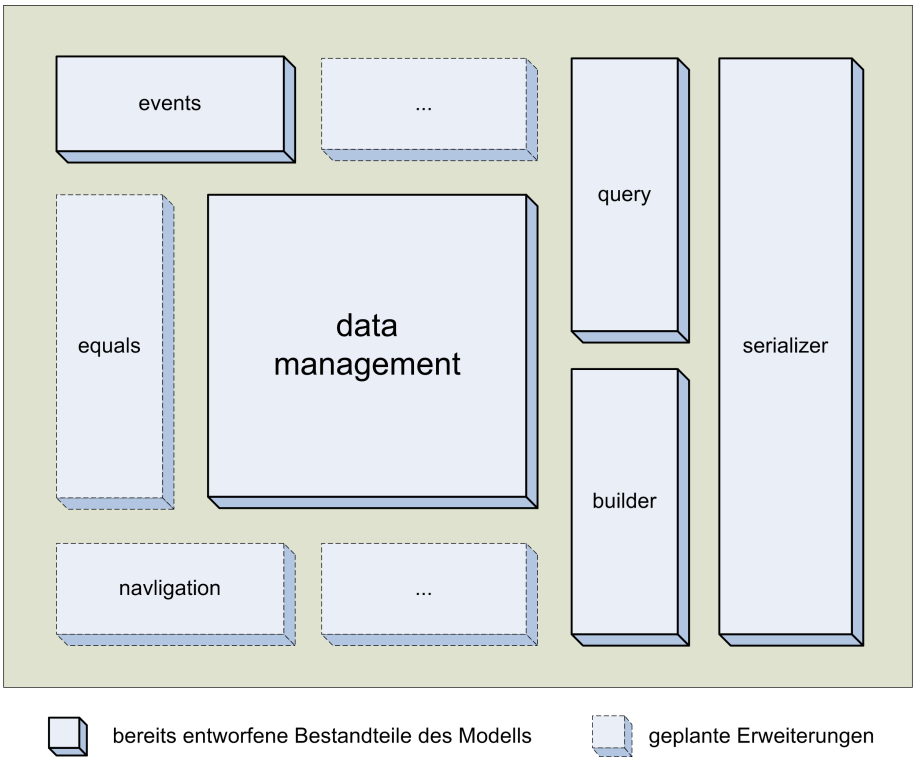

Abbildung 1: Architektur des Komponentenmodells

Zentrum der Architektur bildet die in der Abbildung mit *data management* bezeichnete Datenhaltung. Sie dient der lokalen Speicherung der Entitäten und Relationen des Modells zur Laufzeit der nutzenden Anwendung. Hierfür stellt sie Möglichkeiten zum Lesen und Schreiben der Daten zur Verfügung. Die Konsistenz der Daten ist in dieser Schicht lediglich in Bezug auf die verwendeten Datenstrukturen zu gewährleisten. Semantische Fehler im Sinne des theoretischen Komponentenmodells sind von der Datenhaltung zu tollerieren,

um unabhängig von möglichen Änderungen dieser Semantik zu bleiben. Aufgrund dessen ist der das Modell nutzenden Anwendung keine Möglichkeit zu gewähren, direkt auf die Datenhaltung zuzugreifen, da sonst Korrektheit im Sinne des theoretischen Modells nicht mehr gewährleistet werden kann. Zugriff ist erst nach Überprüfung durch entsprechende Zwischenschichten zu gestatten.

Schreibende Änderungen der Anwendung am Modell sind hierbei durch den in der Architekur mit *builder* bezeichneten Block vorgesehen. Dieser stellt eine Infrastruktur bereit, welche Änderungen an verschiedenen Stellen des Modells zulässt und den korrekten Aufbau gemäß dem theoretischen Modell sicherstellt. Es können an dieser Stelle bereits durch geschickte Wahl der Zugriffsmethoden Fehler ausgeschlossen werden. Eine Möglichkeit der Umsetzung dieser Schicht unter Beachtung der Hierarchie des Komponentmodells wird in Kapitel 5 vorgestellt.

Der lesende Zugriff auf das Modell kann je nach Bedarf durch verschiedene Schichten erfolgen. Die in Abbildung 1 mit *query* bezeichnete Schicht dient der Abfrage von Attributen der Entitäten und der Beziehungen zwischen diesen. Eine direkte Abfragemöglichkeit der Datenhaltung ist prinzipiell möglich, jedoch aufgrund fehlenden Wissens über das theoretische Modell unpraktikabel. Abstraktionen in diesem Sinn sind also ebenfalls Aufgabe der Abfrageschicht.

Die persistente Speicherung des Modells ist Aufgabe der Serialisierungsschicht. Diese soll nach Möglichkeit unabhängig von der genauen Art der Speicherung bleiben. Sowohl der Ex- und Import von Xml-Dateien, wie auch die Speicherung in einer relationalen Datenbank sollen möglich sein. Ebenfalls ist die binäre Speicherung in einem eigenen Datenformat denkbar. Eine Entkopplung vom Kern des Komponentenmodells durch austauschbare Module bietet hierbei die größte Flexibelität. Der Zugriff der Serialisierungsschicht auf die lokal gehaltenen Daten kann auf zwei unterschiedliche Arten realisiert werden, direkter Zugriff auf den Kern oder indirekter Zugriff über die Abstraktionsschichten *query* und *builder*. Vorteil der ersten Variante ist die freie Wahl der Zugriffsmethoden auf die Daten. Je nach Art der Speicherung können Anfragen gezielt angepasst werden, um effizient schreiben oder lesen zu können. Nachteil hierbei ist jedoch die Prüfung der Konsistenz und Korrektheit des Modells. Diese müsste redundant zur *builder*-Schicht implementiert werden. Weiterhin wird eine Abhängigkeit zwischen Kern und Serialisierungsschicht geschaffen, die im Falle der zweiten Variante nicht besteht. Die Nutzung der beiden Abstraktionschichten im zweiten Fall bilden weiterhin eine gute Möglichkeit zur effizienten Qualitätssicherung. So schlagen Serialisierungstests fehl, wenn sich entweder Fehler in der Serialisierungsschicht

selber oder in der Implementierung der Zugriffsmethoden befinden.

Der Benachrichtigung der nutzenden Software bei Änderungen des Modells dient die in der Abbildung mit *events* bezeichnete Schicht. Ziel hierbei ist die Bereitstellung vollständiger Überwachungsmöglichkeit des Modells ohne direkte Verbindung zu den Quellen der Veränderung. So sollen sowohl die Änderungen des Modells durch die Builder als auch durch die Deserialisierung überwacht werden. Setzen sich zur Wahrung der Konsistenz und Korrektheit Aktionen transitiv fort, so sind auch diese Änderungen der Überwachung mitzuteilen.

Die in Abbildung 1 mit *navigation* und *equals* bezeichneten Module sind als Erweiterungen geplant. Zweck des ersten Moduls ist die Navigation durch das Komponentenmodell z.B. entlang des Kontrollflusses unter Verwendung verschiedenster Strategien. Das zweite Modul befasst sich mit der Äquivalenz von Bestandteilen eines Modells oder gar von gesamten Modellen. Die Möglichkeit der unterschiedlichen Definition von Äquivalenz anhand verschiedener Kriterien ist hierbei wichtigste Anforderung an dieses Modul.

Die Architektur kann je nach Bedarf beliebig um Module und Schichten erweitert werden. Diese können entweder direkt oder indirekt über Abstraktionsschichten auf den Kern zugreifen. Bei direkten schreibenden Zugriffen ist wie oben bereits erläutert zu beachten, dass die Konsistenz im Sinn des theoretischen Modells eingehalten werden muss. Keinesfalls dürfen Erweitungsmodule der nutzenden Anwendung den direkten Zugriff auf den Kern ermöglichen. Ist direkter Zugriff nicht zwingend erforderlich, so sind die durch die entsprechenden Abstraktionsschichten angebotenen Schnittstellen vorzuziehen.

In den folgenden Kapiteln werden die für die aktuelle Version des Komponentenmodells entworfenen Module und Schichten vorgestellt, Entwurfsentscheidungen begründet und bekannte Probleme erläutert.

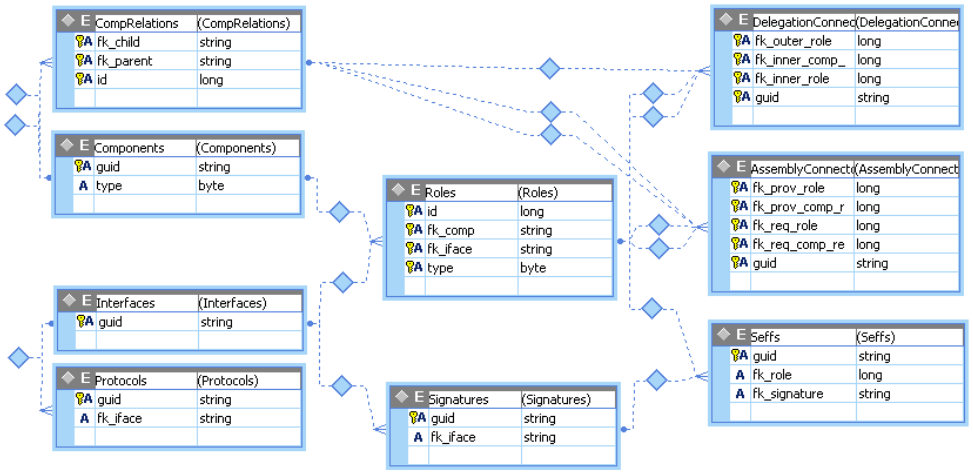

Abbildung 2: .NET Dataset des Modellkerns