Objektorientierte Software-Entwicklung

Gregor Engels
Universität-GH Paderborn
FB 17 - Mathematik/Informatik
AG Datenbank- und Informationssysteme
Warburger Str. 100
33098 Paderborn
engels@uni-paderborn.de
http://www.uni-paderborn.de/cs/engels.html

In den 90er Jahren wurde das in den 70er und 80er Jahren vorherrschende Entwicklungsparadigma der *strukturierten Vorgehensweise* durch das Paradigma der *Objektorientierung* abgelöst. Nahezu zeitgleich wurde es in vielen verschiedenen Teildisziplinen der Informatik zum beherrschenden Thema. Bereits längere Zeit existierende Ansätze wurden auf einmal populär und existierende Ansätze wurden weiterentwickelt. Als Beispiele können hier objektorientierte Programmiersprachen, objektorientierte Benutzungsschnittstellen, objektorientierte Datenbanken und natürlich objektorientierte Software-Entwicklungsmethoden genannt werden.

Vor allem im Bereich der objektorientierten Software-Entwicklungsmethoden gab es in den 90er Jahren eine Vielzahl von Vorschlägen. Prominente Vertreter sind hier die Methoden der Autoren G. Booch, J. Rumbaugh, P. Coad, Shlaer/Mellor oder I. Jacobson. Alle diese Ansätze basieren im Prinzip auf denselben objektorientierten Grundsätzen und Konzepten. Hierzu gehören:

1. Systeme werden als Mengen von interagierenden Objekten aufgefaßt, die durch das Versenden von Nachrichten miteinander kommunizieren können.
2. Objekte sind Instanzen von Klassen und haben eine eindeutige Identität.
3. Klassen folgen der Idee der *abstrakten Datentypen (ADT)*. Damit werden im objektorientierten Ansatz, analog zum ADT-Ansatz, Struktur- und Verhaltensbeschreibung in einer Einheit, der Klasse, gekapselt. Klassen besitzen eine Schnittstelle nach außen, mit der Bezeichner exportiert werden. Konkrete Implementierungen sind nach außen verborgen („*information hiding*“).
4. Objekte können mit anderen Objekten in strukturellen Beziehungen stehen. Hierbei wird unterschieden zwischen Assoziationen, in denen alle beteiligten Objekte gleichwertig sind, und Aggregationen, durch die Enthaltenseinsbeziehungen beschrieben werden.
5. Mit Hilfe der Vererbungsbeziehung zwischen Klassen werden Spezialisierungen von Struktur- und/oder Verhaltensbeschreibungen beschrieben.

Software-Entwicklungsmethoden umfassen Konzepte, Sprachen und Techniken zur Vorgehensweise, um in einem konkreten Software-Entwicklungsprojekt die zunächst nur informell vorliegenden Anforderungen an ein Softwaresystem schrittweise in eine Implementierung umzusetzen. Dabei konzentrieren sich objektorientierte Methoden in erster Linie auf die beiden klassischen Phasen der *Anforderungsanalyse* und des *Entwurfs*. Da vor allem in der ersten Phase auch Fachgebietsexperten und damit nicht unbedingt nur Informatiker beteiligt sind, schlagen nahezu alle objektorientierten Methoden die Verwendung intuitiv zu verstehender Diagrammsprachen vor. Hierzu gehören als Weiterentwicklung von Entity-Relationship-Diagrammen sogenannte Klassendiagramme zur Modellierung der strukturellen Aspekte. Zur Modellierung der Verhaltensaspekte eines Systems werden verschiedene Diagrammformen wie z.B. Zustandsdiagramme, Harel'sche Statecharts, Sequenzdiagramme oder auch Datenflußdiagramme eingesetzt.

Der Boom an objektorientierten Methoden hat dazu geführt, daß eine Vielzahl unterschiedlicher Notationen vorgestellt wurde und somit in Software-Entwicklungsprojekten ein babylonisches Sprachwirrwarr entstand. Dies zerstörte die erhofften Vorteile einer „durchgängigen" objektorientierten Vorgehensweise, da Entwickler in jedem Projekt den dort verwendeten Dialekt und die dazugehörenden Werkzeuge lernen mußten. Entsprechend schwierig war es, Dokumente aus anderen Projekten wiederzuverwenden.

Darauf basierende Kostenfaktoren haben die Industrie dazu motiviert, einen einheitlichen Standard für objektorientierte Software-Entwicklungsmethoden anzustreben. Unter Federführung der drei Autoren Booch, Rumbaugh und Jacobson und unter Mitwirkung eines umfangreichen Firmenkonsortiums wurde in den letzten Jahren die objektorientierte Modellierungssprache UML (Unified Modeling Language) entwickelt und bei der Object Management Group (OMG) als Standardisierungsvorschlag eingereicht. Diese Sprache umfaßt eine Reihe von Diagrammsprachen, die in Analyse und Entwurf eingesetzt werden können. Hierzu gehören Use Case-Diagramme, Klassendiagramme, verschiedene Formen von Verhaltensdiagrammen wie Zustands-, Aktivitäten-, Sequenz- und Kollaborationsdiagramme, und schließlich Entwurfsdiagramme. Die öffentlich zugängliche Dokumentation von UML umfaßt eine Beschreibung der konkreten Syntax (Notation) und der durch ein Meta-Modell festgelegten abstrakten Syntax. Die Beschreibung der Semantik wird in textueller, informaler Form angegeben [UML 97].

Parallel zu der Diskussion über eine einheitliche objekt-orientierte Modellierungssprache ist in den letzten Jahren das Thema der Entwurfsmuster („*design patterns*") aktuell geworden [GoF 95]. Entwurfsmuster sind Muster für die Definition von Klassen und Beziehungen dazwischen, die in dieser Anordnung immer wieder in objektorientierten Modellierungen auftreten. Beispiele hierfür sind das Kompositum, mit dem hierarchische Strukturen modelliert werden, oder der Dekorierer, mit dem mögliche dynamische Erweiterungen der Funktionalität von Objekten modelliert werden. Die Verwendung von Entwurfsmustern unterstützt den Modellierungsvorgang und ermöglicht auch große Systeme, wie z.B. Frameworks, verständlich zu dokumentieren.

Der Vortrag gibt einen Überblick über den State-of-the-Art im Bereich der objektorientierten Software-Entwicklung. Hierzu gehört neben einer Übersicht über aktuelle Entwicklungen im Bereich der UML und der Entwurfsmuster auch eine kritische Darstellung der Stärken und Schwächen einer objektorientierten Software-Entwicklung. Insbesondere werden auch offene Fragen und aktuelle Forschungsthemen angesprochen, die u.a. auch in den Arbeitskreisen der GI FG 2.1.9 „Objektorientierte Software-Entwicklung" behandelt werden. Näheres hierzu findet man unter http://www.uni-paderborn.de/fachbereich/AG/engels/ag_dt/GI/gi-fg219.htm.

Literatur:

[UML 97] UML Documentation, http://www.rational.com/uml

[GoF 95] E. Gamma, R. Helm, R. Johnson, J. Vlissides: Design Patterns, Elements of Reusable Object-Oriented Software, Addison-Wesley 1995