1 Einleitung

Ende der 80er, Anfang der 90er Jahre war ein neues Schlagwort in der Software-Entwicklung vermehrt zu hören: Objektorientierung. Dabei wurde die Objektorientierung mitunter als neue oder gar revolutionäre Idee dargestellt, die dem Software Engineering eine grundlegend neue Basis gibt. Doch so neu waren die Ideen, auf denen die Objektorientierung beruht, gar nicht. Als Mitte der 80er Jahre beispielsweise **Brad Cox** Buch *Object-Oriented Programing - An Evolutionary Approach* erschien, war das keineswegs die Geburtsstunde der objektorientierten Programmierung. Die zugrundeliegenden Ideen waren schon vorher bekannt. Dennoch waren es lange Zeit nur einige wenige Pioniere, die sich an dieses neue Gebiet heranwagten. Die breite Masse - hier für die Software-Industrie insgesamt verwendet entdeckte erst im Laufe der 90er Jahre, dass hinter objektorientierter Programmierung (OOP) mehr als vage Ideen oder akademische Spielereien stecken.

Mit der Entwicklung objektorientierter Sprachen wie Eiffel (Bertrand Meyer), C++ (Bjarne Stroustrup) und Objective-C (Brad Cox) ging die Entwicklung objektorientierter Methoden einher. Auch die Auswirkungen der Objektorientierung auf die gesamte Vorgehensweise, also die Einbeziehung von Analyse (OOA) und Design (OOD), finden sich nicht erst in neueren Veröffentlichungen zu diesem Thema, wenngleich sie heute stärker betont werden als früher. Dazu gehören auch weitergehende Vorschläge, nämlich im Zusammenhang mit der Objektorientierung den Software-Entwicklungszyklus insgesamt umzuorganisieren. Was schließlich den endgültigen Durchbruch brachte und dazu führte, dass die objektorientierte Vorgehensweise heute nicht nur ein allgemein anerkanntes, sondern auch ein meist positiv bewertetes Verfahren des Software-Engineerings ist, dürfte im nachhinein kaum noch auszumachen sein. Die objektorientierte Programmierung ist daher weniger eine Revolution als vielmehr das (derzeitige) Ende einer Folge evolutionärer Schritte, wie bereits der Untertitel von **Brad Cox** Buch nahelegte.

Trotzdem lässt sich festhalten, dass das Paradigma[1] der Objektorientierung eine neue Art ist, Probleme zu verstehen und Lösungen zu finden. Die objektorientierte Vorgehensweise hat vielfach frühere Entwurfs- und Implementierungsmethoden abgelöst und erweitert und gilt als eine der vielversprechendsten Techniken des Software-Engineerings. Allerdings bleiben auch bewährte Prinzipien erhalten, während zugleich neue Prinzipien hinzukommen. Als Defizite konventioneller Methoden wurden vor allem genannt, dass der Problemlösungsbereich mit unterschiedlichen Konzepten beschrieben wird (semantische Lücke), dass es keine Unterstützung bei inkrementeller Erweiterung des Systems (Variantenbildung) und zu wenig Unterstützung für systematische Wiederverwendung vorhandener Software gibt.

Zu den von der Objektorientierung besonders betonten Prinzipien zählen:

- Datenabstraktion
- Modularisierung
- Hierarchisierung
- Kapselung (Information hiding)
- Vererbung
- Polymorphie

Mit Objektorientierung soll eine bessere Anpassbarkeit an veränderte Benutzeranforderungen erlangt werden. Vorhandene Software soll besser erweiterbar und in anderen Projekten wiederverwendbar sein und schließlich erhofft man damit, die Entwicklungskosten bei der Softwareproduktion zu reduzieren.

1.1 Was bedeutet objektorientiert?

Die Kernidee der Objektorientierung ist die Verschmelzung von Daten und Verarbeitung. Der im Deutschen üblichen Bezeichnung EDV als Abkürzung von Elektronischer Datenverarbeitung ist deutlich zu entnehmen, worum es dabei vor allem geht:

1 (griech.) Beispiel, Muster

Daten und Verarbeitung.

Viele der früheren Verfahren des Software-Engineerings konzentrierten sich also darauf, geeignete Daten- und Verarbeitungsstrukturen zu finden, und zwar getrennt voneinander. Genau diese Trennung wird in der objektorientierten Vorgehensweise aufgehoben. Daher enthalten Objekte sowohl Daten als auch Verarbeitung. Das einzelne Objekt versteckt die Daten im Inneren, es kapselt sie, und hat sozusagen eine nach außen für Datenzugriffe undurchdringliche Hülle. Nach außen sichtbar ist nur das, was das Objekt an Verarbeitung - oder in objektorientierter Sprechweise Methoden - anbietet.

Die vom Objekt zur Verfügung gestellten Verarbeitungsmethoden werden von anderen Objekten oder durch äußere Ereignisse, wie etwa Benutzereingaben, aktiviert (Abb. 1.I). Ein objektorientiertes Programm lässt sich daher als ein System kommunizierender Objekte auffassen, die sich gegenseitig benachrichtigen, bestimmte Methoden auszuführen, um ihre Daten im Inneren zu verarbeiten. Daher lässt sich formulieren:

Objekte sind aktive Datenstrukturen.

Ein Objekt weist also einen **Zustand** (Daten) und ein **Verhalten** (Verarbeitungsmethoden) auf. Der Auftrag an ein Objekt, eine seiner Methoden auszuführen, wird in objektorientierter Sprechweise auch **Senden einer Nachricht** genannt (siehe Abb. 1.1).

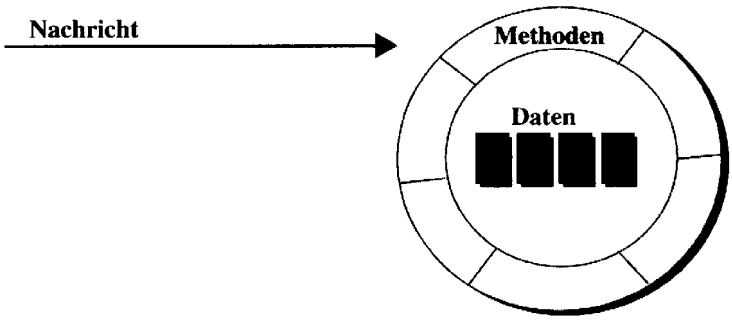

Abb. 1.1: Ein Objekt

Objekte, die dasselbe Verhalten aufweisen und dieselbe Menge von Zuständen speichern können, gehören ein und derselben Klasse an, Die Klasse stellt also den Bauplan dar, der gleichen Objekten zugrunde liegt. Sie ist eine Art Typbeschreibung der Objekte. Jedes Objekt gehört einer Klasse an. Das Konzept der Klasse (des Objekttyps) leitet sich aus dem Konzept der abstrakten Datentypen ab.

Objektorientiertes Programmieren ist ein Programmierstil.

Der objektorientierte Programmierstil wird durch Sprachen wie Java, C++, Smalltalk und Objective-C besser, durch konventionelle Programmiersprachen wie C, Pascal, Basic schlechter unterstützt.

Auf den ersten Blick mögen die Vorteile der Objektorientierung gar nicht so groß erscheinen, da es lediglich um die Verschmelzung von Daten und Verarbeitung geht. Jedoch gibt es eine Reihe von Problemen, die verschwinden oder zumindest erheblich reduziert werden:

Jedes Objekt ist prinzipiell nur für seine eigenen Daten verantwortlich. Wie es diese intern organisiert, ist allein seine Sache und betrifft niemals die anderen Objekte.

Da alle Datenmanipulationen durch das Objekt selbst erfolgen, ist unmittelbar klar, wo der „Schuldige“ für eventuelle Dateninkonsistenzen zu finden ist. Die Fehlersuche und -behebung wird sich damit oft auf ein einzelnes Objekt konzentrieren.

Änderungsanforderungen betreffen in der Praxis meist Daten und Verarbeitung gleichermaßen. In einem objektorientierten Programm gehen Änderungen nicht in zwei getrennte Hierarchien ein, sondern wirken sich in der Regel auf wenige Objekte aus.

Auch der Mensch denkt in Objekten. In der Problemspezifikation könnten etwa Begriffe wie Kunde, Lieferschein usw. auftreten. Für diese Begriffe gibt es in den Objekten eines objektorientierten Programms eine direkte, als natürlich empfundene Entsprechung.

Allgemein lässt sich sagen, dass durch Objektorientierung die Begriffswelt des Programms ein Stück näher an die Problembeschreibung herangerückt wird, und sich die Tätigkeit der Programmiererinnen und Programmierer vereinfacht, so wie das in der Vergangenheit bei jedem entscheidenden Fortschritt - etwa dem Übergang von Assembler auf Hochsprachen - der Fall war.

Aus „Objektorientierter Softwareentwurf mit UML“; G. Bannert, M. Weitzel; Addison-Wesley, 1999

Objektorientierung ist aber weit mehr als nur ein Programmierkonzept. Es handelt sich um einen eigenen Denkstil, um eine bestimmte Art der Herangehensweise an Problemstellungen. Viele Softwareprojekte scheitern an dem Versuch, die herkömmliche Vorgehensweise bei der Softwareentwicklung einfach auf objektorientierte Projekte zu übertragen. Um erfolgreich objektorientiert zu programmieren reicht es aber nicht, objektorientierte Programmiersprachen einzusetzen. Es ist ein objektorientiertes Denken erforderlich. Dies gilt auch und vor allem in den frühen Phasen des Entwicklungsprozesses - in der Analyse und im Entwurf. Die objektorientierte Programmierung verlangt eine Anpassung des Entwicklungsprozesses und der eingesetzten Methoden an den Denkstil - und nicht umgekehrt.

1.2 Objekte und Klassen

Die Objektorientierung ist einer der wenigen Fälle, in denen der Einsteiger gegenüber dem erfahrenen Programmierer einen kleinen Vorteil besitzt: Er ist noch nicht in der Denkweise verhaftet, die klassische Programmiersprachen erfordern. Prozedurale Programmiersprachen wie z.B. Pascal erziehen dazu, in Abläufen zu denken. In der realen Problemstellung zu beobachtende Abläufe werden Schritt für Schritt in Algorithmen umgesetzt, um etwa betriebliche Prozesse in einem Programm zu automatisieren. Die Objektorientierung entspricht mehr der üblichen menschlichen Denkweise, indem sie reale Objekte aus der abzubildenden Umwelt identifiziert und in ihrer Art beschreibt. Es werden Kategorien gebildet, Zusammenhänge dargestellt und neue Objekte aus bekannten Kategorien abgeleitet.

Stellen Sie sich vor, Sie lesen in der Zeitung die folgende Meldung:

Auf Samoa ist eine neue Baumart entdeckt worden. Aufgrund ihrer propellerförmigen Krone gaben die Forscher ihr den Namen „Propeller Baum“. Es handelt sich um einen Laubbaum mit rötlichem Stamm, der eine Höhe von etwa drei bis fünf Metern erreicht. Seine Blätter haben eine dunkelgrüne Färbung.

Welches Bild ist in Ihrem Kopf entstanden, während Sie diese Meldung gelesen haben? Durch die Beschreibung in der Meldung dürften Sie eine ungefähre Vorstellung von dem „Propeller-Baum“ haben. Vielleicht entspricht sie in etwa der folgenden Abbildung.

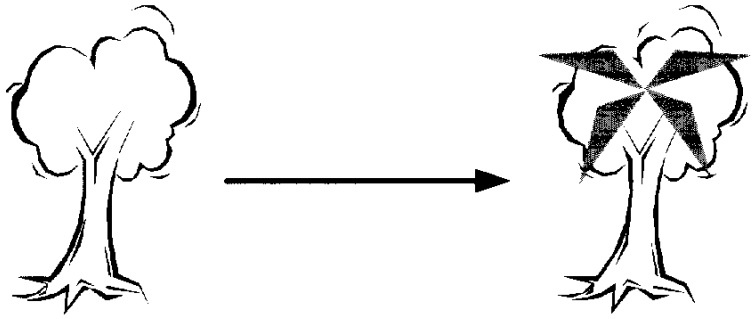

Abb. 1.2: Der·Propeller-Baum

Sie kennen bereits eine Vielzahl von Baumarten. Sie wissen auch, dass ein Baum aus Wurzeln, einem Stamm und einer Krone besteht. Diese Bestandteile müsste demnach auch der „Propeller-Baum“ besitzen. Sie haben also das Bild des neu entdeckten Baumes zunächst grob aus einer Ihnen bekannten Kategorie (Bäume) abgeleitet. Es handelt sich um einen Baum mit propellerförmiger Krone. Sie können somit auch das Aussehen der Krone aus Ihnen bekannten Formen ableiten. Gleiches gilt für die Farbe des Stammes sowie der Blätter.

Diese Denkweise ist eines der Grundprinzipien der Objektorientierung. Objekte der Realität werden mit ihren spezifischen Eigenschaften beschrieben. Alle Objekte mit gemeinsamen Eigenschaften werden zu Kategorien gruppiert. In der Objektorientierung heißen diese Kategorien Klassen, die zugehörigen Objekte bezeichnet man als Instanzen einer Klasse.

Was sind Objekte? Alles kann als Objekt betrachtet werden. Schauen Sie sich einfach einmal um! Objekte sind überall um Sie herum: Ihr Tisch, Ihr Heft, Ihr Stift – oder eben die Bäume draußen vor dem Fenster.

Zur vollständigen Beschreibung einer Klasse gehören drei Komponenten:

- Name
- Zustand und
- Verhalten.

Sie definieren eine Klasse, indem Sie ihr zunächst einen Namen geben, über den sie eindeutig identifizierbar ist. Jede Klasse in Ihrem objektorientierten Modell besitzt einen eigenen Namen, um sie von anderen Klassen unterscheiden zu können. Dann benennen Sie sämtliche Eigenschaften, die Sie den Objekten dieser Klasse zuschreiben. Dabei können statische (**Zustand**) und dynamische Eigenschaften (**Verhalten**) unterschieden werden. Es handelt sich in der Regel um eine rein subjektive Definition. Das heißt, Sie definieren nur die Eigenschaften, die Sie selbst in dem gegebenen Kontext für relevant halten.

Zustand

Statische Eigenschaften von Objekten und Klassen werden als deren Zustand bezeichnet. Es handelt sich dabei um Attribute, mit denen Objekte der Realität beschrieben werden. Eine Klasse Auto hat also beispielsweise die Attribute besitzer, typ, farbe und geschwindigkeit.

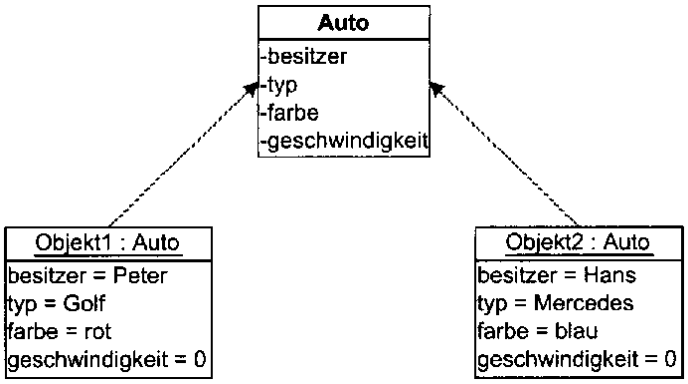

Abb. 1.3: Zustand von Objekten

Dieser Klasse könnten nun als Instanzen zwei Objekte angehören, wie in der Abbildung 1.3 veranschaulicht: Peters roter Golf (Objekt1) und ein blauer Mercedes, der Hans gehört (Objekt2). Beide Autos stehen in der Garage, daher ist ihre Geschwindigkeit gleich Null.

Die Attributwerte eines Objekts können sich ändern. Wenn Peter seinen Golf etwa an Karl verkauft, so ändert sich der Wert des Attributs besitzer des entsprechenden Objekts von „Peter“ in „Karl“. Lässt Hans seinen Mercedes grün lackieren, ändert sich der Attributwert der farbe von Objekt2 von „Blau“ in „Grün“. Die Attribute eines Objekts bezeichnen also stets seinen aktuellen Zustand.

Verhalten

Neben der Zustandsbeschreibung ist es bei den meisten Objekten sinnvoll, auch etwas über ihr Verhalten zu wissen. Mit dem **Verhalten** werden mögliche Aktionen (so genannte Methoden oder Operationen) des Objekts beschrieben. Im Beispiel der Autos sind sinnvolle Methoden „Fahren“ und „Bremsen“.

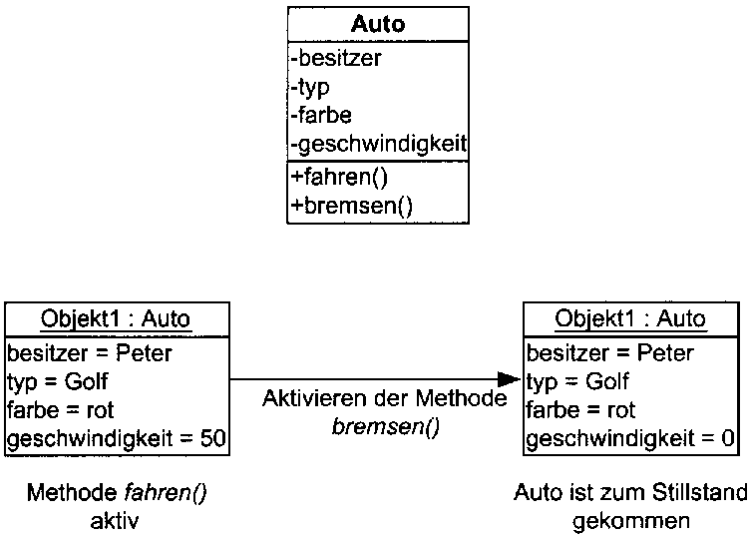

Abb. 1.4: Verhalten von Objekten

Nehmen Sie an, Peters roter Golf fährt gerade mit einer Geschwindigkeit von 50 km/h. Der aktuelle Zustand von Objekt1 entspricht dann der linken Seite der Abb. 1.4. Die Methode fahren ist aktiv.

Wird jetzt die Methode bremsen aktiviert, dann verringert sich die Geschwindigkeit des Autos und das Objekt ändert seinen Zustand (geschwindigkeit = 0). Diese verändere Situation wird durch die rechte Seite der Abbildung 1.4 symbolisiert.

Ein Objekt ist also eine abstrakte Darstellung der Realität. Indem Sie ein Objektmodell erstellen, bilden Sie einen für Sie relevanten Ausschnitt aus der realen Umwelt ab. Veränderungen dieser Umwelt können Sie in Ihrem Modell durch entsprechende Änderungen von Objektzuständen nachvollziehen.
Damit kennen Sie nun alle Bestandteile von Objekten. Jedes Objekt besitzt einen **Namen**, mit dem es eindeutig identifiziert werden kann, eine **Zustandsbeschreibung** durch Attribute und Attributwerte sowie ein spezifisches **Verhalten**, das in Methoden definiert wird. Gleichartige Objekte, also Objekte mit ähnlichen Zuständen und ähnlichem Verhalten, werden zu Klassen zusammengefasst. In dem Beispiel der Abbildung 1.3 werden die Objekte „Peters roter Golf“ (Objekt1) und „Hans blauer Mercedes“ (Objekt2) in der Klasse Auto zusammengefasst. Sie sind Instanzen der Klasse Auto.

1.3 Kommunikation zwischen Objekten

Die Zusammenfassung von **Name**, **Zustand** und **Verhalten** eines Objekts bezeichnet man als **Kapselung**. Jedes Objekt besitzt mit diesen drei Merkmalen alle notwendigen Informationen zu seiner vollständigen Beschreibung. Dies stellt einen wesentlichen Unterschied zur prozeduralen Programmierung dar, in der Daten und Funktionen getrennt sind.

Eine weitere Besonderheit der Objektorientierung, die aus dem Umstand der Kapselung resultiert, ist die Kommunikation zwischen Objekten über **Nachrichten**. Um den Zustand eines Objekts zu manipulieren, können Sie nicht einfach direkt in das Objekt eingreifen und den entsprechenden Attributwert ändern. Sie müssen zunächst eine **Methode** aufrufen, die dann die gewünschte Manipulation durchführt.

Die Kapselung aller benötigten Informationen in einem Objekt ist ein wichtiges Prinzip, das einen modularen Aufbau objektorientierter Software fördert. Durch klar definierte Schnittstellen lassen sich Aufgaben zwischen verschiedenen Objekten teilen.

Wenn Peter sein fahrendes Auto zum Stillstand bringen möchte, dann tritt er auf die Bremse, und sein roter Golf verringert die Geschwindigkeit, bis diese schließlich den Wert Null erreicht. Er kann nicht einfach eine Geschwindigkeit von Null vorgeben, um sofort zum Stillstand zu kommen. Genauso muss in einem objektorientierten Modell dieses Vorgangs die Methode bremsen aktiviert werden, die das Attribut geschwindigkeit entsprechend manipuliert. Die folgende Abbildung veranschaulicht den Vorgang.

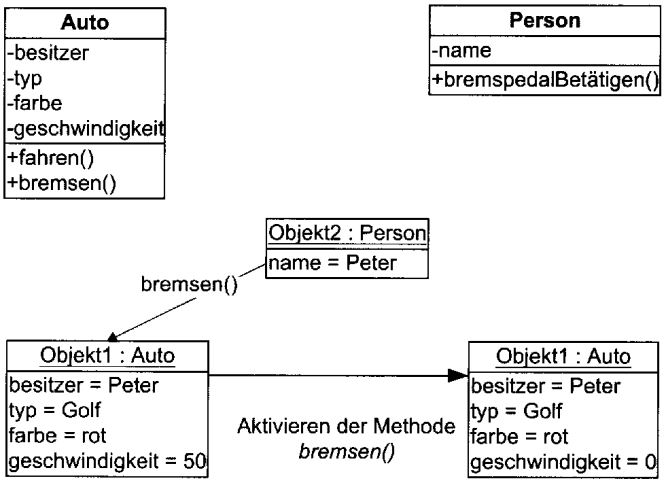

Abb. 1.5: Kommunikation zwischen Objekten

In der objektorientierten Terminologie lässt sich die obige Abbildung wie folgt beschreiben: Das Objekt2 (Peter) sendet über seine Methode bremspedalBetätigen (definiert in der Klasse Person) eine Nachricht an das Objekt1 (Peters Golf) und aktiviert damit dessen Methode bremsen. Dieser „Umweg“ über die Objektmethoden kommt Ihnen vielleicht umständlich vor; er macht aber durchaus Sinn. Die Kapselung ermöglicht einen streng modularen Aufbau objektorientierter Programme. Einzelne Module (hier: Objekte) eines Programms können verändert werden, ohne dass die übrigen Objekte davon in irgendeiner Weise beeinträchtigt werden. Wenn Peters Freundin Petra mit seinem roten Golf fährt, funktioniert die Bremse genauso wie vorher, obwohl die Methode bremsen nun von einem anderen „Objekt“ aufgerufen wird. Auch eine Modifikation des Autos, beispielsweise das Austauschen der Bremsscheiben, hat keinerlei Einfluss auf das Grundprinzip des Bremsens und damit auf das Zusammenspiel der beiden Objekte Fahrer und Auto.

Ein weiterer Vorteil des modularen Aufbaus objektorientierter Programme ist die Mehrfachverwendbarkeit der Module. Sie können Objekte bzw. ganze Klassen in verschiedenen Programmen wiederverwenden.

Aus *Das Einsteigerseminar UML*; Thomas Erler; verlag moderne industrie Buch AG, Landsberg; 2000 – 2001.