Darstellung von Assoziationen

Wie bereit aus Kapitel 1 bekannt, beschreiben Assoziationen Beziehungen zwischen Objekten, die zwischen Klassen modelliert werden. Zunächst soll die Modellierung binärer Assoziationen in UML betrachtet werden.

Rein aus Modellierungssicht müssen Objekte nicht über ihre Beziehungen informiert sein. Diese können relativ selbständig existieren. So wird man über nähere Verwandtschaftsverhältnisse meist informiert sein, doch bei fernerer Verwandtschaft ist das schon fraglich. Doch selbst bei der Vater-Kind-Beziehung gibt es Fälle, in denen die betroffenen Personen nicht informiert sind, obwohl die Beziehung existiert.

Bei der objektorientierten Spezifikation ist das etwas anders. Hier werden Assoziationen mit dem Begriff Navigierbarkeit verknüpft. Damit soll ausgedrückt werden, dass man sich entlang der Beziehungen von einem Objekt zu dem oder zu den in Beziehung stehenden Objekten bewegen kann. Ein Objekt ist mindestens über die Beziehung informiert und von ihm kann man sich zu dem anderen bewegen. In einem solchen Falle spricht man von einer unidirektionalen Assoziation. Sind beide Objekte einer binären Assoziation über das andere Objekt informiert, so ist die Beziehung bidirektional.
In einem Banksystem steht ein Konto auch in Beziehung zu der dort modellierten Klasse Kunde.

Bild 2.24 Unidirektionale Assoziationen zwischen Kunde und Konto

Ein Kunde besitzt bei einer Bank mindestens ein Konto und ein Konto kann einem oder mehreren Kunden gehören. Die Navigation ist von beiden Seiten möglich. Ein Kunde ist also über die zugehörigen Konten informiert und umgekehrt sind aus jedem Konto die Eigentümer ersichtlich.
Liegen zwei unidirektionale Assoziationen in verschiedenen Richtungen zwischen zwei Klassen vor, so kann dies unter Umständen auch als bidirektionale Assoziation modelliert werden.
Jede bidirektionale Assoziation kann umgekehrt durch zwei unidirektionale Assoziationen ausgedrückt werden. Für das Beispiel aus Bild 2.24 ergibt sich das Modell in Bild 2.25

Bild 2.25 Assoziation zwischen Kunde und Konto

In UML ist nicht genau definiert, ob eine Assoziation, wie in Bild 2.25 angegeben, eine bidirektionale Assoziation ist oder ob die Navigationsangabe nicht spezifiziert ist und sowohl unidiektional in eine der beiden Richtungen oder bidirektional sein kann. Einige Autoren wie Fowler und Oesterreich tendieren zu der zweiten Annahme. Sie gehen von einer Unterspezifikation aus, verlieren damit aber die Möglichkeit der Spezifikation einer bidirektionalen Assoziation. Sie müssen stets zwei unidirektionale Assoziationen notieren, was auch nicht immer der Übersichtlichkeit dient. Für die explizite Notation einer bidirektionalen Assoziation wäre die folgende Notation denkbar.

Bild 2.26 Bidirektionale Assoziation zwischen Kunde und Konto

Hier wird explizit ausgedrückt, dass die Navigation in beide Richtungen möglich sein soll. In UML wird nur die Nutzung eines Dreiecks vorgeschlagen, dessen Spitze die Richtung der Assoziation angibt. Dieses Dreieck ist ein reiner Kommentar und daher spricht nichts dagegen, im Bedarfsfalle auch zwei Dreiecke als Kommentar zu nutzen.

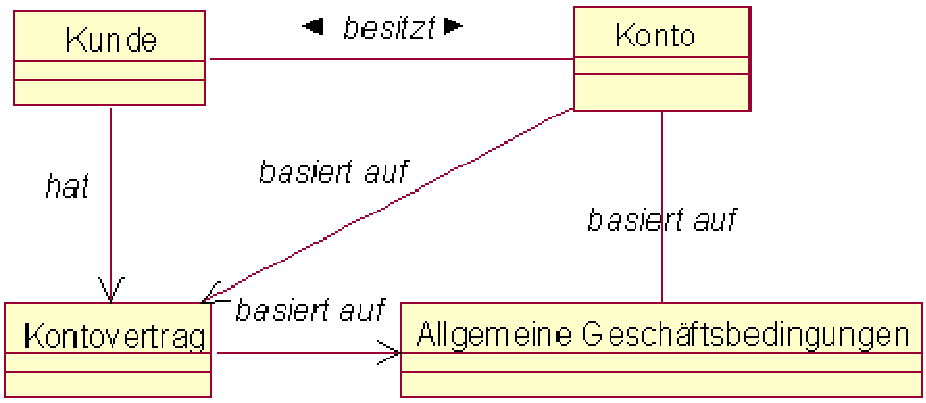

Bild 2.27 Kombinationen von Assoziationen

Bild 2.27 drückt aus, dass ein Kunde einen Kontovertrag hat, auf dem die Abwicklung eines Kontos beruht. Der Kontovertrag basiert auf den allgemeinen Geschäftsbedingungen. Damit stützt sich auch die Verwaltung des Kontos auf diese Geschäftsbedingungen. Da es sich um eine abgeleitete Assoziation handelt, kann auf sie verzichtet werden. Die Navigation von einem Konto zu den allgemeinen Geschäftsbedingungen kann auch über den Kontovertrag erfolgen.
Auf die Assoziation zwischen Konto und Kontovertrag könnte auch verzichtet werden, da vom Konto über Kunde zu Kontovertrag navigiert werden kann. Mitunter ist es aber durchaus sinnvoll, derartige Informationen doppelt zu verwalten und wenn es nur darum geht, die Navigation zu vereinfachen. In dem Falle muss aber sichergestellt werden, dass die Navigation von Konto zu Kontovertrag den gleichen Vertrag liefert, wie die Navigation über den Kunden. Dazu kann man Zusicherungen formulieren.

Darstellung von Zusicherungen

Im aktuellen Beispiel sind zwei logische Ausdrücke vorstellbar, die die Konsistenz der Informationen sichern. Der Klasse Kunde kann der Ausdruck

{Kunde.Kontovertrag=Kunde.Konto.Kontovertrag}

zugeordnet werden. Er besagt, dass die Navigation von Kunde zu Kontovertrag das gleiche Ergebnis liefert, wie die Navigation von Kunde über Konto zu Kontovertrag.
Eine Zusicherung mit gleichem Inhalt kann auch der Klasse Konto zugeordnet werden. Dabei handelt es sich um den Ausdruck

{Konto.Kontovertrag=Konto.Kunde.Kontovertrag}.

Mit derartigen Zusicherungen können Abhängigkeiten zwischen Assoziationen gut beschrieben werden. Explizit ist in UML noch eine weitere Art von Zusicherung festgeschrieben, die dazu genutzt werden kann, auszudrücken, dass ein Objekt nur eine von zwei möglichen Beziehungen eingehen kann.

Darstellung von Oder-Assoziationen

Wenn in einem Banksystem zwischen Privatkunden und Geschäftskunden unterschieden wird, dann kann auch ein Konto eine entsprechende Zugehörigkeit haben. Eigentümer eines Kontos können eine oder mehrere Privatpersonen sein. Es sind aber auch ein oder mehrere Unternehmer als Eigentümer denkbar. Inhaber eines Kontos dürfen aber nicht gleichzeitig ein Privatkunde und ein Geschäftskunde sein.

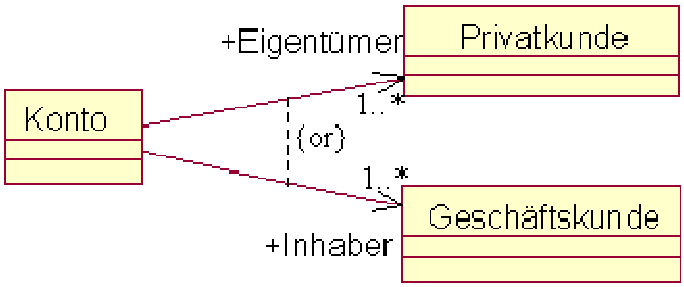

Bild 2.28 Oder-Assoziation

Durch die Zusicherung "{or}" wird verdeutlicht, dass nur eine Assoziation für ein Konto zutreffend ist. Wichtig ist dabei, den Rollen der beiden in die Assoziationen einbezogenen Klassen unterschiedliche Namen zu geben.
In dem dargestellten Fall darf nicht in beiden Fällen die Rolle Eigentümer genutzt werden. Erklären kann man diese Einschränkung durch die spätere Implementation. Damit die Objekte der Klasse Konto Referenzen zu den Privat- und Geschäftskunden verwalten können, ist in der Klassendefinition jeweils ein Array enthalten, welches die Referenzen aufnehmen kann. Der Name des Arrays ist der Name der Rolle, den die Objekte spielen. In Java würde die Klasse Konto folgendes Aussehen haben:

```
public class Konto {

   public Privatkunde Eigentümer[];
   public Geschäftskunde Inhaber[];
   ...
}
```

Die Zusicherung gewährleistet die Wertebelegung nur eines der beiden Arrays. Liegen in beiden Assoziationen die gleichen Rollennamen vor, gibt es natürlich einen Konflikt.
Mitunter ist es besser, auf die Angabe der Rollennamen zu verzichten, da in diesem Falle der Name der Klasse als Rollenname angenommen wird. Damit ist gesichert, dass nicht der gleiche Name vorliegt, da die Klassennamen unterschiedlich sind.
Die Angabe des gleichen Rollennamens in beiden Fällen mag fachlich durchaus sinnvoll erscheinen, darf aber in UML nicht spezifiziert werden.

Darstellung von attributierten Assoziationen

Mitunter möchte man Assoziationen auch gewisse Eigenschaften zuordnen. Beispielsweise könnte bei einem Konto über die Assoziation festgelegt werden, um welche Art von Konto es sich handelt, welcher monatliche Betrag als Gutschrift erfolgen muss, welcher Verfügungsrahmen besteht und wie die Verzinsung aussieht. Man spricht in diesem Falle auch von attributierten Assoziationen.

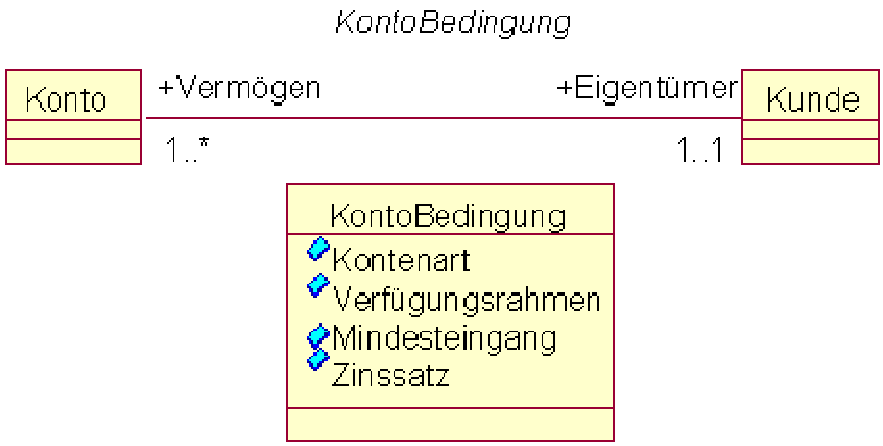

Bild 2.29 Attributierte Assoziation mit Assoziationsklasse

In diesem Modell hat ein Konto genau einen Eigentümer. Dieser kann aber mehrere Konten haben. Eines muss er aber mindestens besitzen. Die Eigenschaften der Assoziation werden durch eine gesonderte Klasse modelliert. Der Name der Klasse muss mit dem Namen der Assoziation übereinstimmen. Ihre Attribute sind gleichzeitig die Assoziationsattribute.

Eine Assoziationsklasse spielt eine Sonderrolle. Sie kann nicht, wie alle anderen Klasse unabhängige Objekte erzeugen. Die von ihr erzeugten Objekte sind von der Existenz anderer Objekte abhängig. Sie kapseln Informationen, die diesen anderen Objekten nicht direkt zugeordnet werden können, sondern genau der Beziehung zwischen ihnen entsprechen.

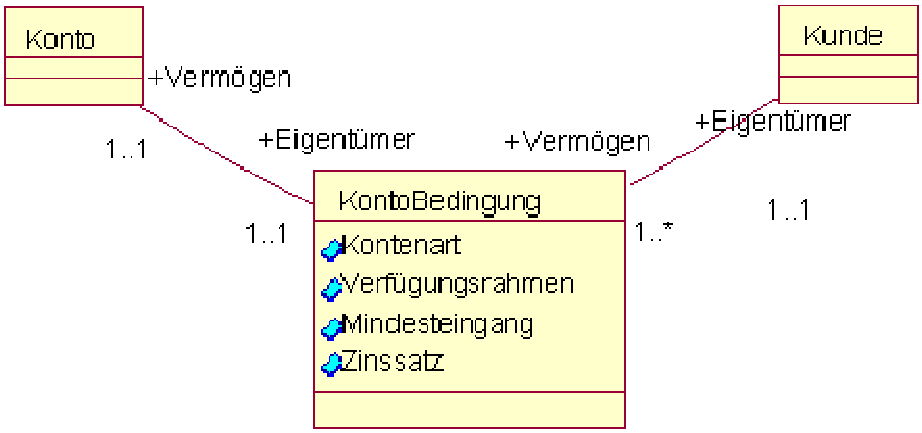

Bild 2.30 Klassen zur Modellierung von Assoziationen

Die Attribute der Klasse KontoBedingung beschreiben die Beziehung zwischen den Klassen Konto und Kunde. Besonders zu beachten sind die Multiplizitäten. An den Klassen Konto und Kunde entsteht 1..1. Es existiert also aus Sicht der späteren Objekte genau ein Beziehungsobjekt. Die ursprüngliche Multiplizität an den Klassen steht nun an der Assoziation mit der jeweils anderen Klasse. Dort wiederum stehen sie an der Seite der Beziehungsklasse. Die an der Klasse KontoBedingung links stehende Multiplizität 1..1 stammt von der Klasse Kunde und die rechts stehende Multiplizität 1..* stammt von der Klasse Konto.

Darstellung von Aggregationen und Kompositionen

Aggregation und Komposition wurden bereits im Einführungskapitel behandelt und in der Notation von UML dargestellt. Beides sind spezielle Assoziationen. Sie werden zwischen Klassen modelliert, repräsentieren aber Beziehungen der Objekte.

Wenn diese Beziehung zwischen den Objekten sehr eng wird, dann modelliert man nicht mehr eine allgemeine Assoziation, sondern eine Aggregation. Werden die Objekte unzertrennlich, dann ist die Komposition die richtige Form der Darstellung. In diesem Fall beendet die Existenz des Aggregates auch die Existenz aller Teilobjekte.

Der Unterscheidung zwischen beiden Assoziationsformen ist mitunter nicht ganz einfach. Sie muss aus der Sicht des jeweiligen Projektes getroffen werden und kann nicht globale Entwicklungen bereits berücksichtigen.

Ein Beispiel ist die Modellierung des menschlichen Körpers. Hier verwendet man sicher die Komposition für die Modellierung der Hände, da sie fester Teil des Körpers sind. Wenn man allerdings Transplantationen in die Betrachtung einbezieht, so treffen die oben aufgeführten Regeln nicht mehr ganz zu und man müsste eine Aggregation nutzen.
Solche globalen Entwicklungen müssen in einem konkreten Projekt nicht berücksichtigt werden.

Man sollte allerdings nicht zu viele Gedanken über den Unterschied zwischen allgemeiner Assoziation, Aggregation und Komposition verschwenden. Die Konsequenzen für die weitere Softwareentwicklung sind nicht so entscheidend. Andere Dinge sind da wichtiger.

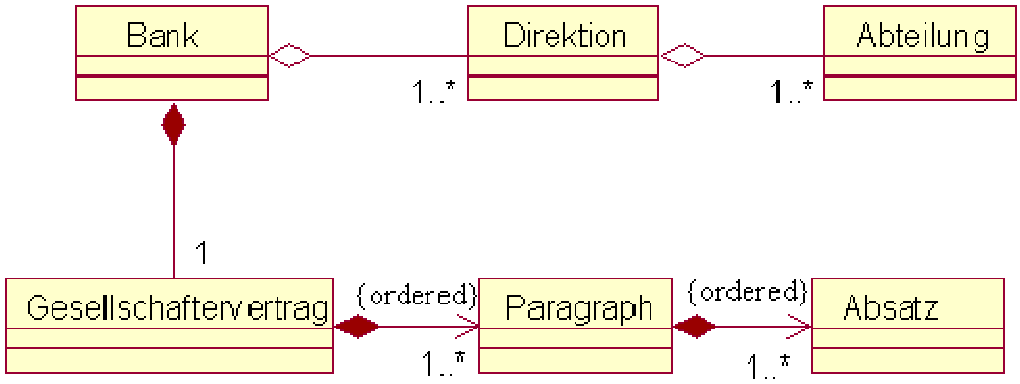

Bild 2.31 Aggregation versus Komposition

Bei dieser Modellierung wurde die Beziehung zwischen Gesellschaftervertrag und Bank so eng modelliert, da der Vertrag bei Auflösung der Bank auch nicht mehr existent ist. Physisch liegt er als Papier zwar weiter vor. Seine Bedeutung hat er aber verloren. Der Vertrag selbst besteht aus Paragraphen und diese wiederum aus Abschnitten. Der Pfeil besagt, dass eine Navigation nur vom Ganzen zu den Teilen erfolgen kann. Die Teilobjekte wissen damit nicht, zu welchen Aggregaten sie gehören. Mit der Zusicherung "ordered" wird ausgedrückt, dass die Teilobjekte nicht in beliebiger Reihenfolge Teil des Ganzen sind. Wonach geordnet wird, kann separat modelliert werden, erfolgt an Hand der Werte der Teilobjekte und ist im beschriebenen Fall klar.
Hier ist die Modellierung der Komposition sicher ziemlich eindeutig. Wenn Paragraphen gelöscht werden, sind natürlich automatisch die Absätze gelöscht und wenn der Gesellschaftervertrag insgesamt gelöscht wird, dann gilt das ebenso für die Paragraphen. Wichtig ist auch die sonstige Weiterleitung (Propagieren) von Methoden. Wird ein Gesellschaftervertrag formatiert oder gedruckt, so werden auch die Paragraphen und Abschnitte formatiert und gedruckt. Das Aggregat leitet die Ausführung einer Methode an seine Teile weiter.
Diese Eigenschaft ist der Grund, warum bei organisatorischen Strukturen häufig die Aggregation und nicht die Komposition modelliert wird. In Bild 2.31 betrifft das die Direktionsbereiche und die Abteilungen. Man könnte argumentieren, dass das Auflösen der Bank zum Auflösen der Direktionen

führt. Eine Direktion kann aber in einer anderen Bank weitergeführt werden. Bei dem Verhältnis Direktion zu Abteilung wird das vielleicht noch deutlicher. Hier kann ein Direktionsbereich aufgelöst werden, ohne dass Abteilungen betroffen sind. Sie arbeiten unter anderen Direktionen eventuell weiter. Auch das Propagieren der Methode erfolgt nicht automatisch. Werden die Direktionen umstrukturiert, so folgt daraus nicht unbedingt ein Umstrukturieren der Abteilungen. Ganze Abteilungen können weiterhin in bestehender Form anderen Direktionen zugeordnet werden. Die Umstrukturierung kann auch nur einzelne Abteilungen treffen.

Organisatorische Strukturen und Mitgliedschaft in Vereinen und Gesellschaften sollten als Aggregation modelliert werden.
Für die Komposition gibt es in UML drei verschiedene Darstellungsformen. Für das Beispiel des Gesellschaftervertrages sind sie in Bild 2.32 angegeben.
Den allgemeinen Aufbau der Darstellungen enthält Bild 2.33.

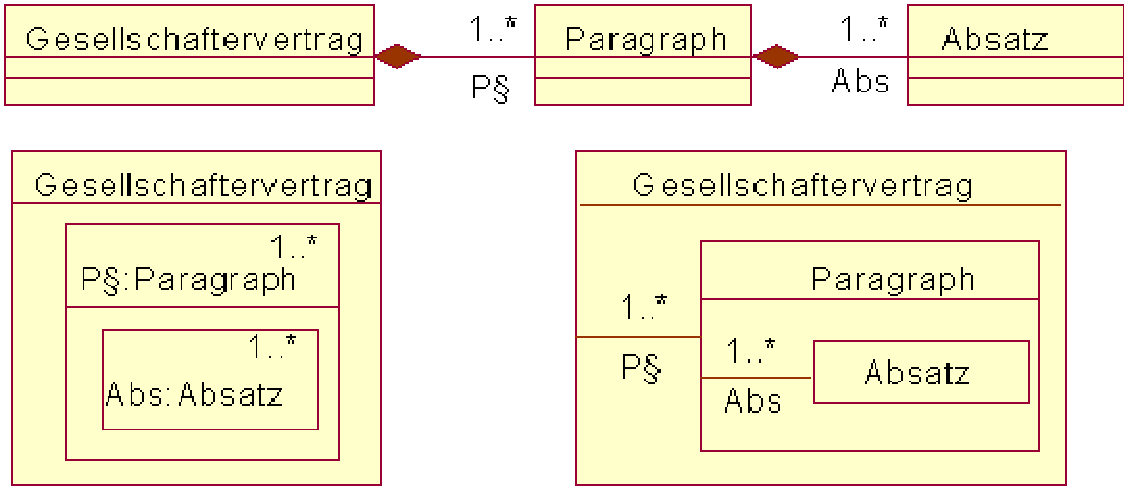

Bild 2.32 Darstellung der Komposition für Gesellschaftervertrag

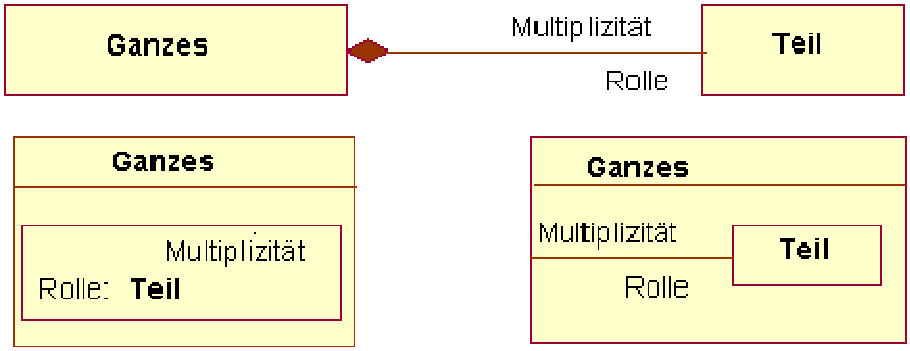

Bild 2.33 Darstellungsformen der Komposition

Welche Art der Darstellungsform genutzt wird, ist den Softwareentwicklern freigestellt. In den meisten Fällen wird jedoch die obere Notation mit dem Drachenviereck genutzt, da sie wohl die übersichtlichste ist. Eine Ausnahme bilden generische Klassen. Bei ihnen wird häufig die untere rechte Darstellungsform genutzt, um besser zu sehen, welche Teile der Spezifikation mit der generischen Klasse direkt zu tun haben. Auf generische Klassen wird in einem späteren Abschnitt noch genauer eingegangen.