UML - Teil 2

In PCNEWS-76 stellten wir Ihnen die ersten drei Diagramme der Unified Modeling Language vor. Teil zwei der Artikel-Reihe soll die sogenannten Interaktions- und Package-Diagramme vorstellen.

Thomas Obermayer

Use-Case-Diagramme, Klassendiagramme und Objektdiagramme (vorgestellt in Teil 1 der Artikel-Reihe) dienen zur statischen Beschreibung von Systemen. Anwendungsfälle, Attribute und Fähigkeiten werden durch Grafiken dargestellt und so sehr gut veranschaulicht. Das Verhalten von Systemen wird in diesen Diagrammen nicht geschildert. Soll eine Handlung bzw. ein dynamischer Ablauf beschrieben werden, so braucht es neue Diagramme, namentlich "Interaktionsdiagramme".

Die *Unified Modeling Language* kennt zweierlei Interaktionsdiagramme; einerseits das "Sequenzdiagramm", andererseits das "Kollaborationsdiagramm". Beide werden in diesem Artikel beschrieben. Anschließend erhalten Sie eine kurze Vorstellung so genannter "*Package*-Diagramme".

Das Sequenzdiagramm

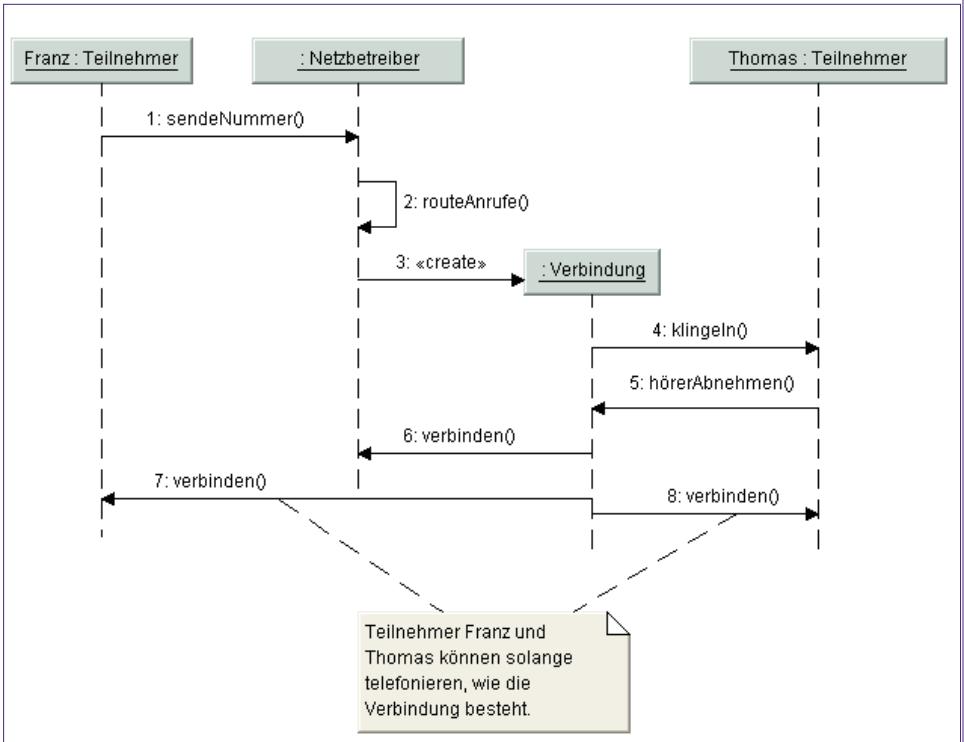
Sowohl das Sequenzdiagramm, als auch das Kollaborationsdiagramm schildern die Zusammenarbeit („Kollaboration") zwischen Objekten. Es werden also konkrete Fälle dargestellt, weshalb Klassen und Multiplizitäten hier nicht zu finden sind ***(vgl. „UML – Teil 1")***. Anders als im Objektdiagramm versucht man nun, den Ablauf eines Prozesses festzuhalten. Der zeitliche Verlauf des Nachrichtenaustausches zwischen Objekten steht hier im Vordergrund.

Die Objekte werden durch Rechtecke visualisiert. Von ihnen gehen senkrechte Lebenslinien (auch *„Life Lines"*) weg, die durch gestrichelte Linien dargestellt werden. Nachrichten zwischen Objekten werden durch waagrechte Pfeile zwischen den Objektlebenslinien eingetragen. Ober diesen Pfeilchen werden die Nachrichtennamen in der Form „`nachricht(argumente)`" notiert. Nachrichten, die als Antworten deklariert sind, erhalten die Form „`antwort:=nachricht()`". Möchte man Nachrichten an etwaige Bedingungen binden, so lässt sich dies in der Schreibweise „`[bedingung] nachricht()`" darstellen. Iterationen werden durch ein Sternchen „`*`" vor dem Nachrichtennamen beschrieben.

Man kennzeichnet die aktive Beteiligung eines Objektes an einer Interaktion durch einen Balken auf der Lebenslinie. Jedoch unterstützen einige CASE-Tools und UML-Editoren diese Notation nicht (auch in unserer Abbildung finden sie derartige Balken nicht).

Objekte können während des des zeitlichen Ablaufes erzeugt und gelöscht werden. Ein Objekt wird erzeugt, indem ein Pfeil mit der Aufschrift `<<create>>` auf ein neues Objektsymbol trifft und zerstört, indem seine Lebenslinie in einem Kreuz endet.

Mit Hilfe von Sequenzdiagrammen ist es möglich, fertige Programme vollständig auf dem Papier darzustellen.



Sequenzdiagramm: Objekte, Lebenslinien, Nachrichten - Darstellung einer Telefonverbindung

Das Kollaborationsdiagramm

Das Kollaborationsdiagramm zeigt die einzelnen Objekte und ihre Zusammenarbeit untereinander. Dabei steht im Gegensatz zum Sequenzdiagramm der zeitliche Ablauf dieser Interaktionen im Hintergrund. Es werden die für den Programmablauf wichtigen kommunikativen Aspekte zwischen den einzelnen Objekten ereignisbezogen dargestellt.

Der zeitliche Verlauf der Interaktionen wird lediglich durch eine Nummerierung der Nachrichten („Botschaften") symbolisiert. Inhaltlich unterscheidet sich dieses Diagramm vom Sequenzdiagramm nicht wesentlich; es werden bloß andere Schwerpunkte beleuchtet.

Die einzelnen Objekte werden wieder als Rechtecke eingezeichnet. Diese werden durch Assoziationslinien miteinander verbunden, auf welchen die Nachrichten, bestehend aus einer durchgehenden zeitlichen Nummerierung, den Namen der Nachrichten, etwaigen Antworten und ihren möglichen Argumenten, platziert werden.

Ein Pfeil deutet die Richtung der Nachricht vom Sender zum Empfänger an. Antworten werden in der Form „`antwort:=nachricht()`" symbolisiert.

Die zeitliche Markierung erfolgt mittels aufsteigender Durchnummerierung der einzelnen Nachrichten, angefangen bei eins.

Werden innerhalb eines Vorganges, den eine empfangene Nachricht interpretiert, neue Nachrichten verschickt, so erhalten diese neuen Nachrichten eine durch einen Punkt von der Hauptnummerierung getrennte Unternummerierung (z.B. „`1.1:`"). Somit kann die sogenannte „Verschachtelungstiefe" einer Nachricht innerhalb anderer Nachrichten nachvollzogen werden. Die Nummerierung wird durch einen Doppelpunkt abgeschlossen (einige CASE-Tools und UML-Editoren geben den Doppelpunkt nicht an). Auf die Kennzeichnung von Verschachtelungstiefen wurde in der Abbildung bewusst verzichtet.

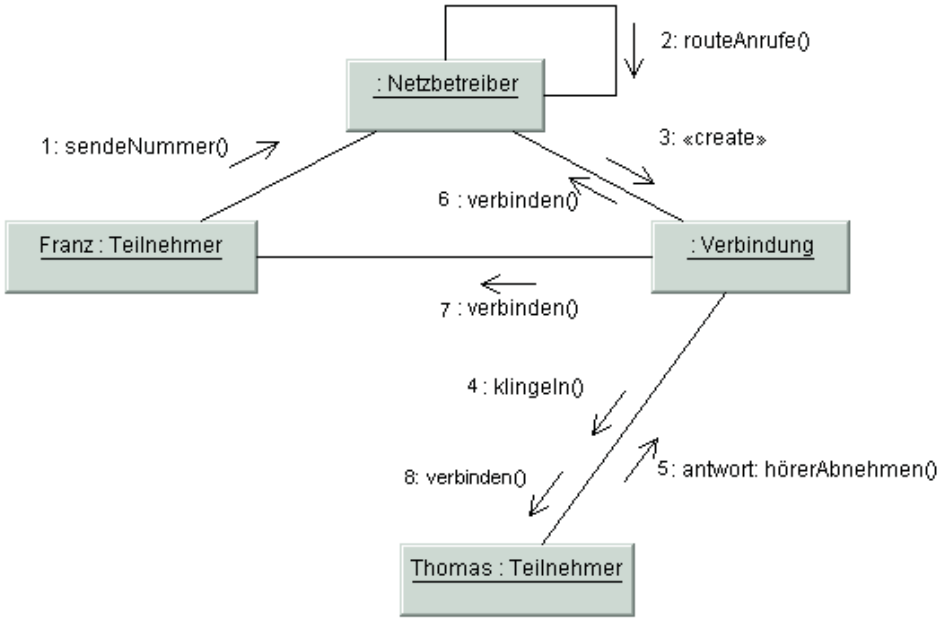
Mit Hilfe eines Sternchens „`*`" hat man die Möglichkeit, anzugeben, dass Nachrichten mehrfach versendet werden.

In einem Kollaborationsdiagramm können Objekte erzeugt und zerstört werden. Objekte die gerade erzeugt wurden, wer-

den mit `<<create>>`, Objekte die im Kontext zerstört werden, durch `<<destroy>>` markiert. Objekte, die erzeugt und später wieder vernichtet werden, sind mit `<<transient>>` zu kennzeichnen.

Ein Vergleich der grafischen Abbildungen zum Sequenzdiagramm und zum Kollaborationsdiagramm wird Ihnen schnell die unterschiedliche Betonung der beiden Darstellungsformen näher bringen. Sie werden weiters feststellen, dass sich die beiden Diagrammformen inhaltlich nicht wesentlich unterscheiden.

Die UML führt weit mehr Möglichkeiten zur Ausführung dieser Diagramme ein. Als Entwickler von UML-Modellen können Sie selbst entscheiden, wie ausführlich Sie diese gestalten. Nützliche Diagramme müssen also nicht zwingend vollständig sein! Oft ist es zielführender, nur die wesentlichen Merkmale eines Modells zu beleuchten.



Kollaborationsdiagramm: Objekte, Assoziationslinien, Nachrichten – Darstellung einer Telefonverbindung

Das *Package*-Diagramm („Paket-Diagramm“)

Um verschiedene Modellelemente, wie z.B. Klassen oder Anwendungsfälle nach ihren physischen oder logischen Zusammenhängen zu gruppieren, bietet sich das *Package*-Diagramm der UML an.

Man überlegt, wie sich einzelne Elemente am besten zu wiederverwendbaren Modulen bzw. Bibliotheken vereinen lassen und teilt diesen einen Namensraum (einen Paketnamen) zu. Pakete können hierarchisch gegliedert werden, also ihrerseits weitere Pakete enthalten.

Eine Klasse besitzt immer ihr Stammpaket, sie kann aber auch in verschiedenen anderen Klassen vorkommen. Sie wird dann in der Schreibweise `Paketname::Klassenname` zitiert und besitzt so einen eindeutigen Namen.

Durch das Auftreten von Klassen eines Paketes in einem anderen entstehen Abhängigkeiten zwischen den Paketen. Gut modellierte Systeme weisen nur wenige dieser Abhängigkeiten von Paketen auf.

Das Konzept der Paketbildung eignet sich z.B. sehr gut für unternehmensweite Modellierung, bei der jeder Abteilung ein eigener Namensraum zugeteilt wird. Definieren beispielsweise zwei Abteilungen Klassen mit identischen Namen, so kommt es zu keinen Konflikten, da jede Klasse einem eigenen Paket zugeordnet ist und über ihren vollständigen Bezeichner (`Paketname::Klassenname`) identifiziert wird.

Ein Paket wird im *Package*-Diagramm als Aktenregister dargestellt, welches den Namen des jeweiligen Paketes enthält. Werden innerhalb des Symbols Modellelemente angezeigt, steht der Name auf der Registerlasche, andernfalls innerhalb des Rechteckes. Manche CASE-Tools und UML-Editoren führen leicht abgewandelte Formen dieser Notation ein.

Die Darstellungen in diesem Teil der UML-Reihe wurden mit ObjectDomain 3.0 (http://www.objectdomain.com/) erstellt und anschließend leicht mit einem Grafikprogramm modifiziert.

Bei ObjectDomain 3.0 handelt es sich um ein CASE-Tool, das aus Diagrammen Programmcode erstellen kann (JAVA, C++, Python) und umgekehrt in der Lage ist, aus Programmcode (JAVA, C++, Python) UML-Diagramme zu erstellen *(„Reverse-Engineering“)*. Dies ermöglicht so genanntes *„Roundtrip-Engineering“*, bei dem der Entwickler abwechselnd modelliert und programmiert.

Man ist in der Lage, ein erstes Modell zu erstellen und dieses in Programmcode umzuwandeln. Anschließend erweitert man den Programmcode schrittweise und aktualisiert das Modell entsprechend (automatisch, durch Reverse-Engineering des CASE-Tools). Man fährt fort, bis ein fertiges Modell mit zugehörigem Code entstanden ist.

In der nächsten Ausgabe von PCNEWS stellen wir Ihnen das Aktivitätsdiagramm, das Zustandsdiagramm und die Implementierungsdiagramme der UML vor.

Package-Diagramm: Klassen, Pakete