Übersicht der UML Diagramme

Die Unified Modeling Language (UML) ist eine Sprache zur Beschreibung von Softwaresystemen. Der Grundgedanke der UML besteht darin, eine einheitliche Notation für viele Einsatzgebiete zu haben.

Die UML gilt heute als Standard für Analyse und Design Objektorientierter Anwendungen. Sie bietet ein Gerüst, um so unterschiedliche Applikationen wie Datenbankanwendungen, Echtzeitsysteme oder Grafikprogramme einheitlich darstellen zu können. Die Aufgabe der UML ist Spezifikation, Visualisierung und Dokumentation von Modellen für Softwaresysteme.

Die UML besteht im Wesentlichen aus neun verschiedenen Diagrammtypen. Die einzelnen Diagramme besitzen verschiedene graphische Elemente, deren Semantik genau festgelegt ist. Mit diesem Dokument erhalten Sie eine Übersicht der einzelnen UML-Diagramme und ihrer Anwendung. Sie finden bei jedem Diagrammtyp eine genaue Beschreibung der jeweiligen Verwendung, ergänzt durch ein praktisches Beispiel.

Diagrammtypen

Die Modellelemente der UML werden nach Diagrammtypen gegliedert. *Jedem Diagrammtyp liegt eine bestimmte Betrachtungsweise eines Softwaresystems zu Grunde.* Die einzelnen Diagramme lassen sich nach diesen Betrachtungsweisen in vier Bereiche gliedern:

Betrachtungsweise: Requirements (Anforderungen an das System)

• Use-Case-Diagramm	Akteure, Szenarios

Betrachtungsweise: Statische Sicht (logischer Aufbau des Systems)

• Klassendiagramm	Klassen, Beziehungen
• Paketdiagramm	Strukturierung der Darstellung
• Kollaborationsdiagramm	Zusammenwirken von Komponenten

Betrachtungsweise: Dynamische Sicht (Interaktionen, Abläufe im System)

• Aktivitätsdiagramm	Ablaufmöglichkeiten
• Sequenzdiagramm	Objekte, Interaktionen
• Zustandsdiagramm	Internes Verhalten von Objekten

Betrachtungsweise: Implementierung

• Komponentendiagramm	Innere Struktur von Objekten
• Verteilungsdiagramm	Einbettung von Objekten in eine Umgebung

1. Use-Case Diagramm

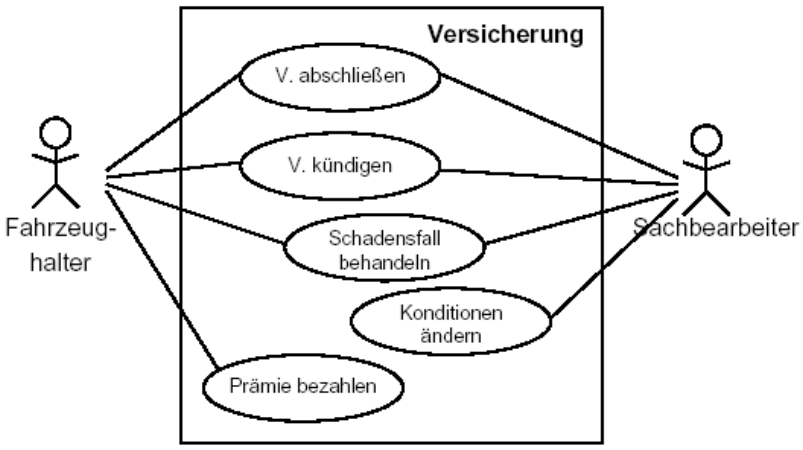
⇨ Use-Case Diagramme beschreiben das Zusammenwirken von Personen (Aktoren) mit einem System.

Use-Case Diagramme werden auch als Anwendungsfall-Diagramme bezeichnet. Unter einem Use-Case wird eine typische Handlung verstanden, die ein Benutzer mit dem System ausführt, z. B. "Versicherung abschließen". In das Use-Case-Diagramm werden die Use-Cases als Ellipsen eingezeichnet. Use-Cases können beliebig kompliziert und umfangreich sein. Verbindungen zwischen den Use-Cases und den Aktoren werden durch Linien hergestellt. Damit wird angezeigt, welche Aktoren an dem entsprechenden Use-Case beteiligt sind. Die Aktoren werden nach ihrem Rollenverhalten im Bezug auf das System und nicht nach ihrer Identität unterschieden. Im unten angeführten Beispiel könnte z. B. der Sachbearbeiter auch selbst ein Fahrzeughalter sein.

Use-Cases werden durch Szenarios beschrieben (Normalfälle + Problemfälle). Sie sind die Grundlage für die Erstellung des Systems und müssen daher vollständig sein. Darüber hinaus bilden sie die Grundlage für das Testen des Systems nach der Erstellung.

Während sich die Use-Cases selbst seit ihrer Einführung als extrem nützlich erwiesen haben, ist der Nutzen der Use-Case-Diagramme umstritten. Die textuelle Beschreibung der Szenarios ist in jedem Fall erforderlich und in den meisten Fällen hinreichend. Das Diagramm kann darüber hinaus dazu dienen, die Zusammenhänge zu verdeutlichen.

Beispiel: KFZ-Versicherung
Die verschiedenen Use-Cases für eine KFZ-Versicherung sind:
– Versicherung abschließen
– Versicherung kündigen
– Versicherter bezahlt Prämie
– Sachbearbeiter behandelt Schadensfall
– Sachbearbeiter ändert Konditionen



Quelle: Dr. Michael Hahsler, Abteilung fuer Informationswirtschaft, WU Wien

2. Klassendiagramm

⇨ Klassendiagramme beschreiben die statische Struktur von Objekten in einem System und ihre Beziehungen untereinander.

Klassendiagramme sind der zentrale Bestandteil der UML und auch zahlreicher Objektorientierter Methoden. Wie die Klassen ermittelt werden, darüber gibt die UML keine Auskunft; hierfür gibt es andere Techniken, z. B. CRC-Karten (Abk. für "Class, Responsibility and Collaboration") oder die Substantiv-Technik. Die UML beschränkt sich darauf, Notation und Semantik zu beschreiben.

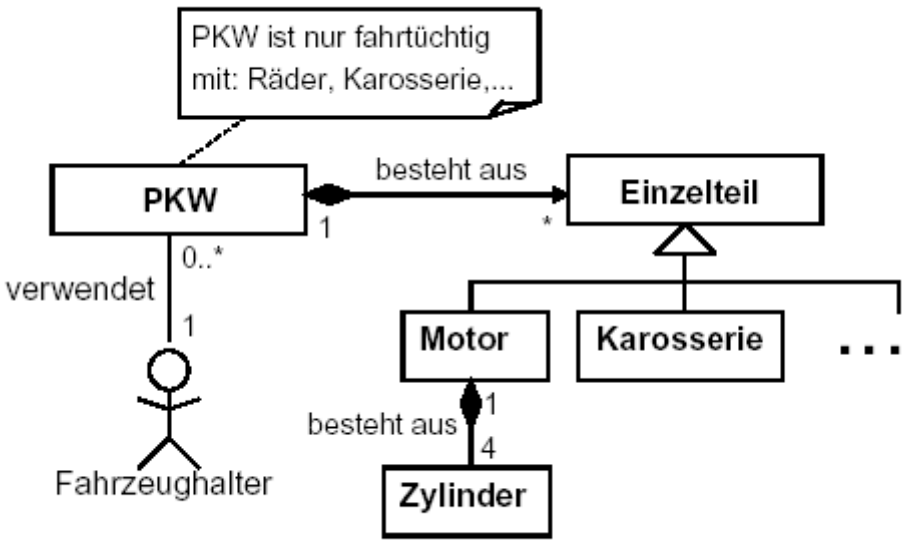
Klassendiagramme haben zwei wesentliche Elemente:

- *Objekt:* ein Konzept, eine Abstraktion oder ein Gegenstand mit klarer Abgrenzung und präziser Bedeutung, z. B. der "rote Apfel".
- *Klasse:* ein Gruppe von Objekten mit ähnlichen Eigenschaften, z. B. "Äpfel".

Eine strenge visuelle Unterscheidung zwischen Objekten und Klassen entfällt in der UML. Objekte werden von den Klassen dadurch unterschieden, dass ihre Bezeichnung unterstrichen ist, sonst sind die Symbole gleich. Auch können Klassen und Objekte zusammen im Klassendiagramm auftreten.

Das Klassendiagramm beschreibt die statische Struktur der Objekte in einem System sowie ihre Beziehungen untereinander. Das nachfolgende Beispiel verdeutlicht die Anwendung.

Beispiel: PKW



Quelle: Dr. Michael Hahsler, Abteilung fuer Informationswirtschaft, WU Wien

Das zentrale Element im Klassendiagramm sind die Klassen. Sie werden als Rechtecke dargestellt. Die gefundenen Klassen werden durch Linien miteinander verbunden. Diese Linien stellen die Elemente Assoziation, Aggregation, Komposition und Vererbung dar.

Die *Assoziation* stellt eine allgemeine Beziehung zwischen zwei Klassen dar - über die Realisierung wird dabei nichts ausgesagt.

Eine besondere Assoziation ist die *Aggregation*, die durch eine Raute an der Linie dargestellt wird. Sie gibt an, dass eine Klasse in einer anderen Klasse "enthalten" ist (Ist-Teil-von-Beziehung).

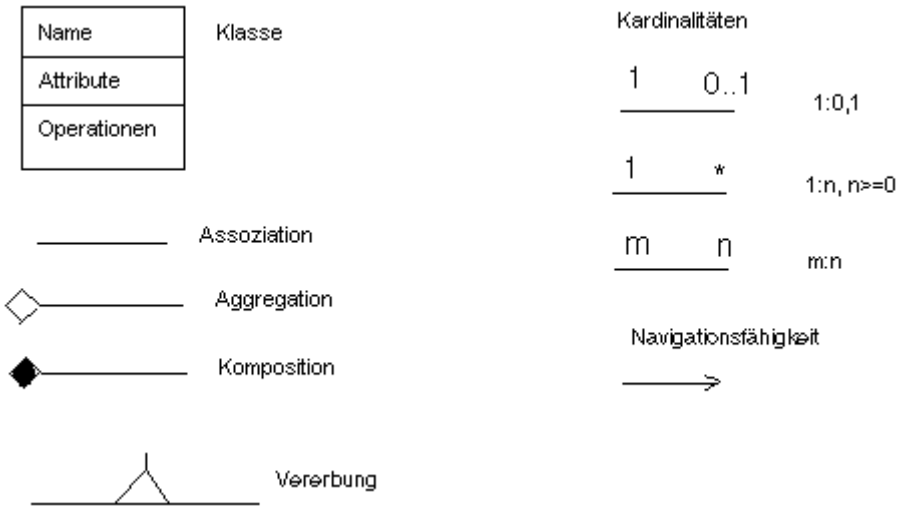
Die *Komposition* ist eine stärkere Form der Aggregation, die durch eine ausgefüllte Raute dargestellt wird (z.B. zwischen Motor und Zylinder). Sie gibt an, dass eine Klasse aus anderen „besteht“.

An einer Assoziation können noch Multiplizitäten, d. h. Zahlen oder Zahlbereiche, angegeben werden. Diese bestimmen die Anzahl der Objekte, die miteinander in Beziehung stehen. Beispielsweise ist ein PKW genau einem Fahrzeughalter zugeordnet, während ein Fahrzeughalter keinen oder mehrere PKW verwenden kann.

Jede Assoziation kann eine Richtung besitzen; hierfür wird ein Pfeil am Ende der Assoziation angebracht. Zugriffe können dann nur in Pfeilrichtung erfolgen. Man verwendet deshalb den Begriff der Navigationsfähigkeit (engl. Navigability).

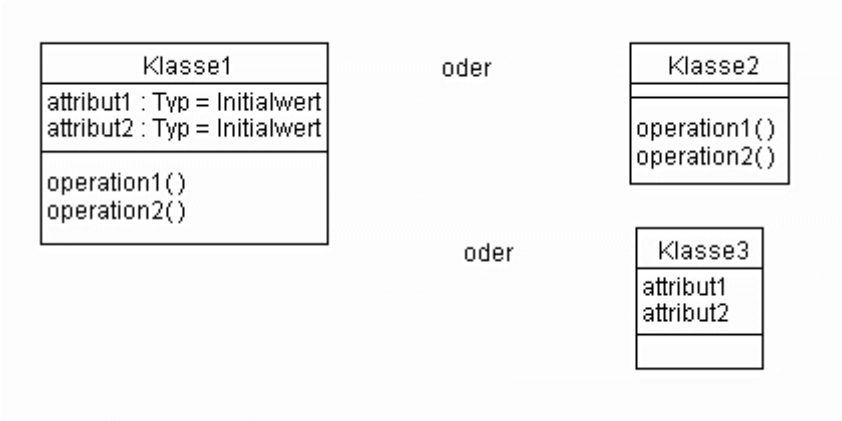
Die *Vererbung* stellt eine Verallgemeinerung von Eigenschaften dar - sie wird auch als Spezialisierung und Generalisierung oder "Ist-ein"-Beziehung bezeichnet. Zum Beispiel ist ein Motor ein Einzelteil. Ein Einzelteil hat generelle Eigenschaften, und ein Motor spezialisiert diese.

Hier die wichtigsten Elemente eines Klassendiagramms:



Klassen werden durch Rechtecke dargestellt, die den Namen der Klasse und/oder die Attribute und Operationen der Klasse enthalten. Klassenname, Attribute und Operationen werden durch eine horizontale Linie getrennt. Der Klassenname steht im Singular und beginnt mit einem Großbuchstaben (z.B. Apfel). Attribute können näher beschrieben werden, z.B. durch ihren Typ, einen Initialwert und Zusicherungen. Sie werden aber mindestens mit ihrem Namen aufgeführt. Operationen können ebenfalls durch Parameter, Initialwerte, Zusicherungen usw. beschrieben werden. Auch Sie werden mindestens mit ihrem Namen aufgeführt.

Hier der typische Aufbau von Klassen:



3. Paketdiagramm

⇨ Paketdiagramme dienen der Strukturierung der verschiedenen Darstellungen.

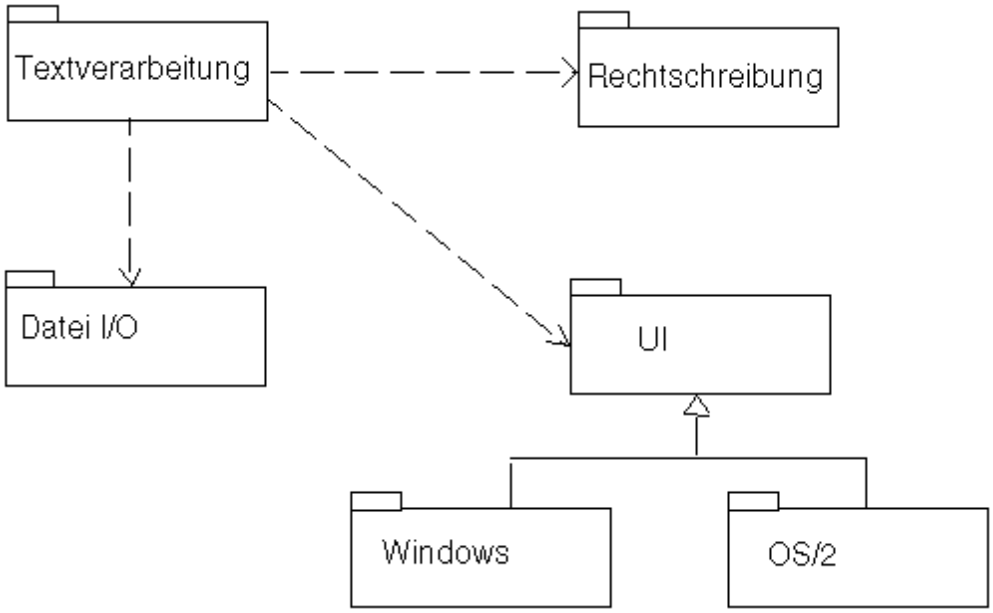
Pakete ermöglichen einen guten Überblick über ein Gesamtsystem. Insbesondere für größere Systeme ist eine solche Strukturierung sehr wichtig.

In einem Paketdiagramm (engl. Package-Diagram) werden Gruppen von Diagrammen oder Elementen zusammengefasst. Dies ist besonders wichtig für den Überblick über das System und die Aufteilung in verschiedene Übersetzungseinheiten.

Ein Paketdigramm besteht im Wesentlichen aus Paketen (dargestellt durch große Rechtecke mit kleinem Rechteck links oben) und Abhängigkeiten (den gestrichelten Pfeilen). Eine Abhängigkeit gibt an, dass bei einer Änderung des Pakets an der Pfeilspitze das Paket am anderen Ende der gestrichelten Linie eventuell geändert bzw. neu übersetzt werden muss.

Nachfolgend ein Beispiel eines Paketdiagramms für eine Textverarbeitung:

Beispiel: Textverarbeitung



4. Kollaborationsdiagramm

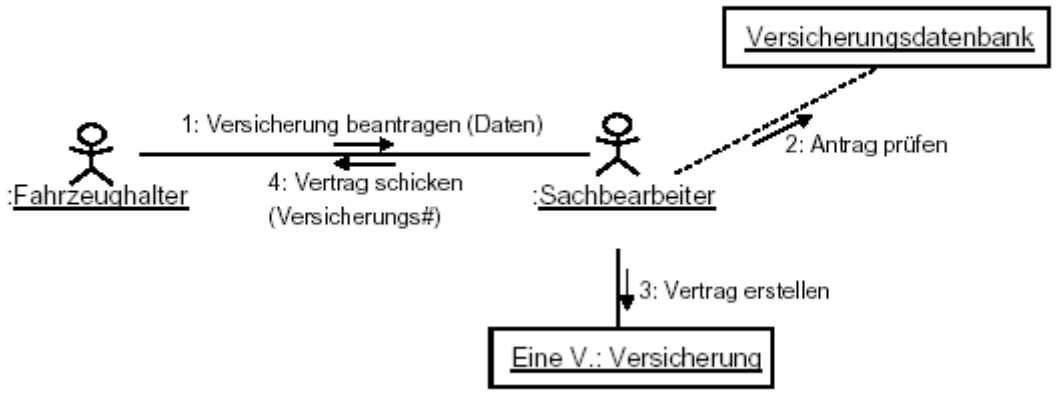
⇨ Kollaborationsdiagramme stellen einzelne Objekte und ihre Interaktion (auf Ereignisse bezogen) dar.

Im Kollaborationsdiagramm steht, im Vergleich zum Sequenzdiagramm, nicht der zeitliche Ablauf der Interaktionen im Vordergrund - vielmehr werden die für den Programmablauf wichtigen kommunikativen Aspekte zwischen den einzelnen Objekten auf Ereignisse bezogen dargestellt.

Der zeitliche Verlauf der Interaktionen wird durch eine Nummerierung der Nachrichten symbolisiert. Das Kollaborationsdiagramm kann für die Darstellung von Entwurfs-Sachverhalten benutzt werden und, in etwas detaillierterer Form, von Realisierungssachverhalten. Es beinhaltet stets kontextbezogene begrenzte Projektionen des Gesamtmodells.

Die einzelnen Objekte werden als Rechtecke dargestellt. Diese werden durch Assoziationslinien miteinander verbunden, auf denen dann die Nachrichten, bestehend aus einer durchgehenden zeitlichen Nummerierung, dem Namen der Nachrichten und Antworten sowie ihren möglichen Argumenten verzeichnet werden. Ein Pfeil zeigt die Richtung der Nachricht vom Sender zum Empfänger. Die zeitliche Nummerierung erfolgt mittels Durchnummerierung der einzelnen Nachrichten angefangen bei 1.

Beispiel: KFZ-Versicherung
Szenario „Versicherung abschließen“: Ein Fahrzeughalter beantragt eine KFZ-Versicherung. Der Antrag wird vom zuständigen Sachbearbeiter geprüft. Nach erfolgreicher Prüfung wird ein Vertrag abgeschlossen.



Quelle: Dr. Michael Hahsler, Abteilung fuer Informationswirtschaft, WU Wien

5. Sequenzdiagramm

⇨ Sequenzdiagramme stellen die einzelnen Objekte und ihre Interaktion (auf zeitlichen Ablauf bezogen) dar.

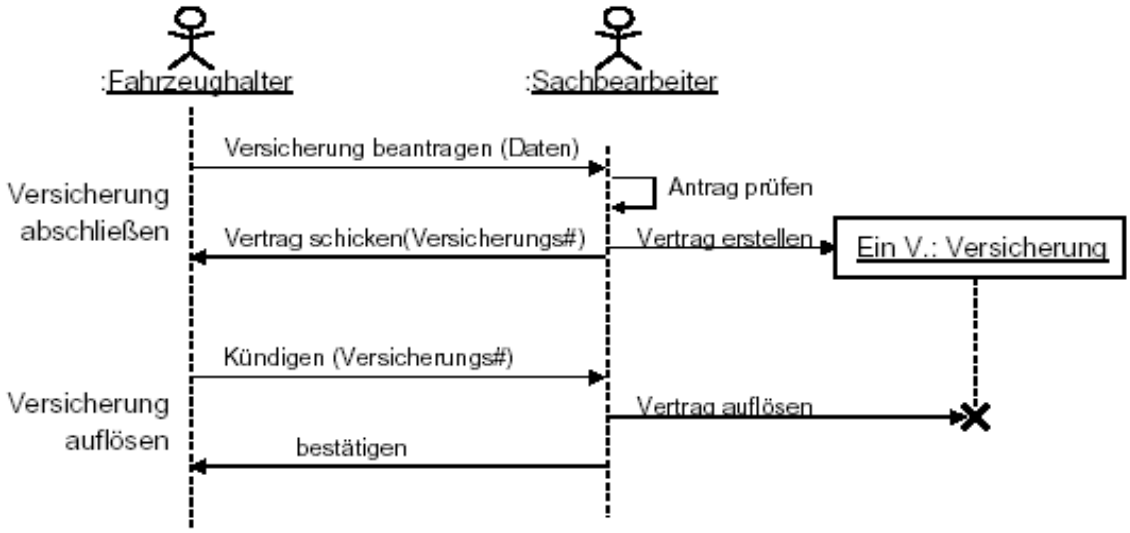
Sequenzdiagramme sind neben den Kollaborationsdiagrammen der zweite Fall von Interaktionsdiagrammen. Beide beschreiben Interaktionen zwischen Objekten, jedoch steht beim Sequenzdiagramm der zeitliche Verlauf des Nachrichtenaustausches im Vordergrund, während beim Kollaborationsdiagramm die für den Programmablauf wichtigen kommunikativen Aspekte zwischen den einzelnen Objekten dargestellt werden.

Interaktionsdiagramme (wie eben Sequenzdiagramme) werden ausgehend von Fallbeispielen - den *Szenarios* - erstellt. Deshalb werden alle Interaktionsdiagramme auch *Szenariodiagramme* genannt. Beim Erstellen der Diagramme konzentriert man sich auf die "wichtigsten" Fälle (primäre Szenarios). Anschließend werden die Sonderfälle mit einbezogen und eventuell weitere Diagramme erstellt (sekundäre Szenarios).

Das Sequenzdiagramm beschreibt die zeitliche Abfolge von Interaktionen zwischen einer Menge von Objekten innerhalb eines zeitlich begrenzten Kontextes. Die Zeitlinie verläuft senkrecht von oben nach unten, die Objekte werden durch senkrechte Lebenslinien beschrieben und die gesendeten Nachrichten waagrecht entsprechend ihres zeitlichen Auftretens eingetragen.

Die Objekte werden durch Rechtecke visualisiert. Von Ihnen aus gehen die senkrechten Lebenslinien, dargestellt durch gestrichelte Linien, ab. Die Nachrichten werden durch waagerechte Pfeile zwischen den Objektlebenslinien beschrieben. Auf diesen Pfeilen werden die Nachrichtennamen in der Form: *nachricht(argumente)* notiert. Objekte, die gerade aktiv an Interaktionen beteiligt sind, werden durch einen Balken auf ihrer Lebenslinie gekennzeichnet.

Beispiel: KFZ-Versicherung
Szenarien “Versicherung abschließen“ und "Versicherung auflösen"



Quelle: Dr. Michael Hahsler, Abteilung fuer Informationswirtschaft, WU Wien

6. Aktivitätsdiagramm

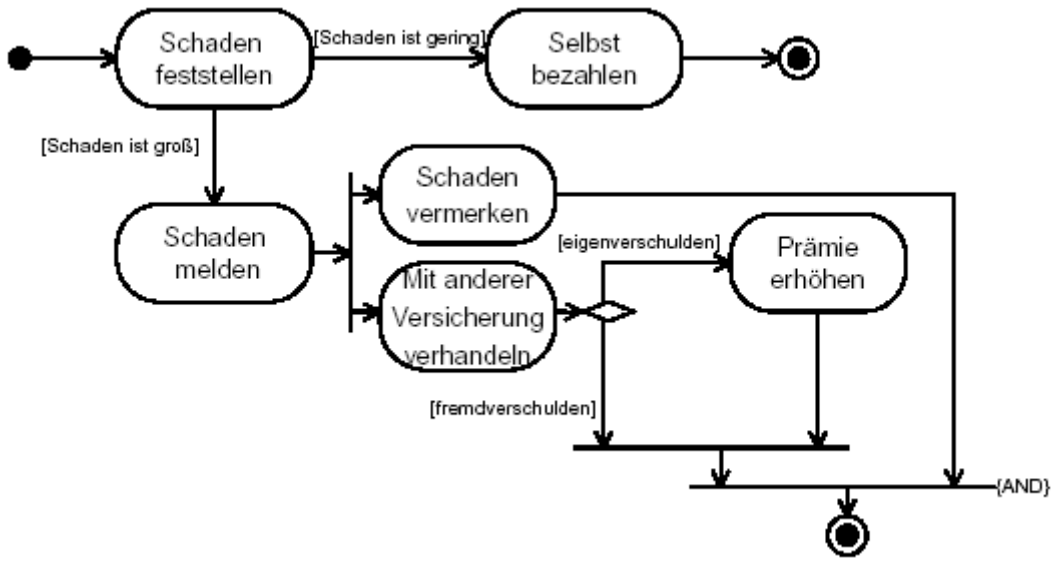
⇨ Aktivitätsdiagramme beschreiben die Objekte eines Programms mittels der Aktivitäten, die sie während des Programmablaufes vollführen.

Eine Aktivität ist ein einzelner Schritt innerhalb eines Programmablaufes, der eine oder mehrere von ihm ausgehende Transitionen enthält. Gehen mehrere Transitionen von der Aktivität aus, so müssen diese mittels Bedingungen voneinander zu entscheiden sein. Somit gilt ein Aktivitätsdiagramm als Sonderform eines Zustanddiagramms, dessen Zustände der Modellelemente in der Mehrzahl als Aktivitäten definiert sind.

In einem Programmablauf durchläuft ein Modellelement eine Vielzahl von Aktivitäten, d.h. Zuständen, die eine interne Aktion und mindestens eine daraus resultierende Transition enthalten. Die ausgehende Transition impliziert den Abschluss der Aktion und den Übergang des Modellelementes in einen neuen Zustand bzw. eine neue Aktivität. Diese Aktivitäten können in ein Zustandsdiagramm integriert werden oder besser aber in einem eigenen Aktivitätsdiagramm visualisiert werden.

Eine Aktivität wird durch ein "Rechteck" mit konvex abgerundeten Seiten visualisiert. Sie enthält eine Beschreibung der internen Aktion. Von der Aktivität aus gehen die Transitionen, die den Abschluss der internen Aktion und den Übergang zur nächsten Aktivität darstellen.

Beispiel: Kfz-Versicherung
Szenario "Schadensfall": Ein Schadensfall tritt ein. Der Sachbearbeiter wird informiert und bearbeitet den Fall. Der Vorfall wird in den Versicherungsunterlagen vermerkt. Bei Verschulden durch den Versicherungsnehmer wird die monatliche Prämie erhöht.



Quelle: Dr. Michael Hahsler, Abteilung fuer Informationswirtschaft, WU Wien

7. Zustandsdiagramm

⇨ Zustandsdiagramme visualisieren die unterschiedlichen Zustände, die Objekte in ihrem Leben annehmen können.

Zustandsdiagramme werden auch Zustandsübergangsdiagramme (engl. State Transition Diagram) genannt, da sie auch die Funktionen beschreiben, die zu Änderungen des Zustands eines Objekts führen.

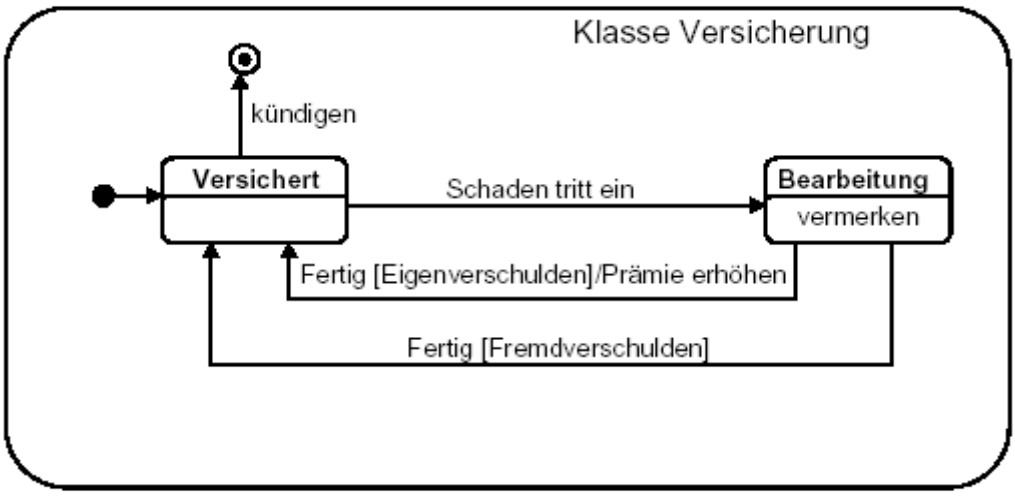
Ein Zustandsdiagramm beschreibt eine hypothetische Maschine, die sich zu jedem Zeitpunkt in einer Menge endlicher Zustände befindet. Sie besteht aus:

- einem Anfangszustand
- einer endlichen Menge von Zuständen
- einer endlichen Menge von Ereignissen
- einer endlichen Anzahl von Transitionen, die den Übergang des Objektes von einem zum nächsten Zustand beschreiben
- einem oder mehreren Endzuständen

Im Zustandsdiagramm werden die Zustände als abgerundete Rechtecke visualisiert, die mit Pfeilen verbunden sind. Auf den Pfeilen stehen die Transitionen. Startzustand ist ein gefüllter Kreis, die Endzustände sind leere Kreise mit einem kleineren gefüllten Kreis in der Mitte.

Theoretisch kann das Verhalten jeder Klasse durch ein Zustandsdiagramm beschrieben werden. In der Praxis werden aber nur für Klassen mit interessantem Verhalten Zustandsdiagramme angefertigt. Zustandsdiagramme sind nicht erforderlich für Klassen, die nur als Schnittstelle dienen und das Verhalten weiter delegieren. Das gleiche gilt für Klassen, die nur Werte zwischenspeichern.

Beispiel: Kfz-Versicherung
Szenario "Schadensfall": Ein Schadensfall tritt ein. Der Sachbearbeiter wird informiert und bearbeitet den Fall. Der Vorfall wird in den Versicherungsunterlagen vermerkt. Bei Verschulden durch den Versicherungsnehmer wird die monatliche Prämie erhöht.



Quelle: Dr. Michael Hahsler, Abteilung fuer Informationswirtschaft, WU Wien

8. Komponentendiagramm

⇨ Komponentendiagramme stellen die Zusammenhänge der einzelnen Komponenten einer zu erstellenden Softwarelösung dar.

Mit Komponentendiagrammen wird sichergestellt, dass bei späterer Implementierung der Softwarelösung Compiler- und Laufzeitabhängigkeiten klar sind.

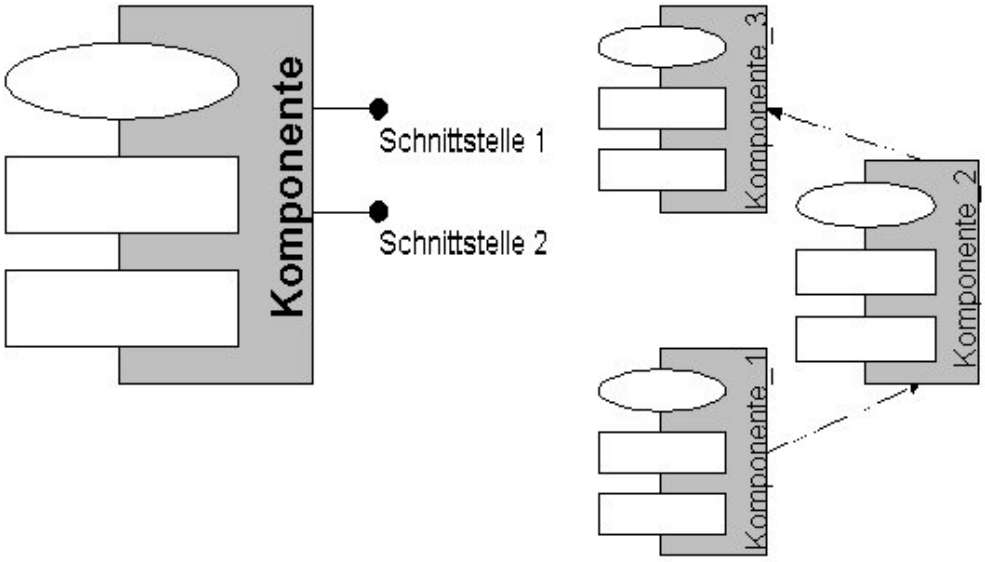
Eine Komponente stellt ein physisches Stück Programmcode dar, welches entweder ein Stück Quellcode, Binärcode oder ein ausführbares Teilprogramm der gesamten Softwarelösung sein kann.

Dargestellt werden die einzelnen Komponenten als Rechtecke, die den Namen und den Typ der jeweiligen Komponente enthalten. Am linken Rand tragen sie eine Ellipse sowie 2 Rechtecke. Eine Komponente kann wiederum weitere Elemente, wie Objekte, Komponenten oder Knoten enthalten und Schnittstellen besitzen.

Die Abhängigkeiten zwischen den einzelnen Komponenten werden durch gestrichelte Pfeile symbolisiert. Die in dieser Art dargestellten Abhängigkeiten zeigen die spätere Kompilierreihenfolge auf.

Der Unterschied zu Paketdiagrammen besteht darin, dass diese ein allgemeines Mittel für die Strukturierung darstellen, während die Komponentendiagramme Implementierungsaspekte zeigen. In der Praxis entspricht eine Komponente einem Package, aber nicht jedes Package einer Komponente.

Hier ein Beispiel dafür, wie Komponentendiagramme aussehen:



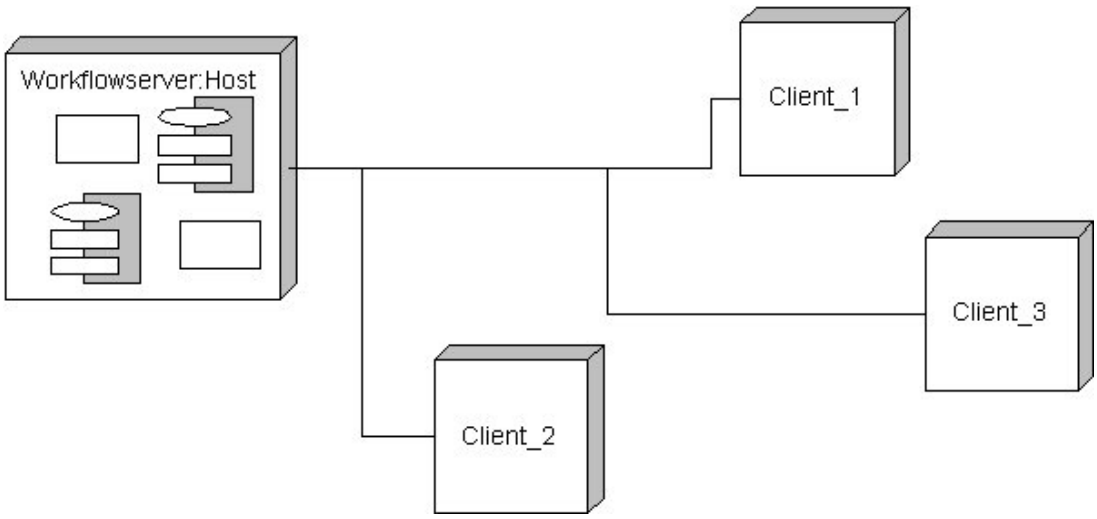
9. Verteilungsdiagramm

⇨ Verteilungsdiagramme beschreiben, welche Komponenten (Objekte) auf welchen Knoten ablaufen.

Verteilungsdiagramme (auch Deploymentdiagramme oder Einsatzdiagramme genannt) zeigen daher, wie Komponenten (Objekte) konfiguriert sind und welche Abhängigkeiten bestehen.

Knoten werden in Verteilungsdiagrammen als Quader visualisiert. In den Knoten können die dort ablaufenden Komponenten (Objekte) dargestellt werden, wobei auch Schnittstellen und Abhängigkeitsbeziehungen zwischen den Elementen erlaubt sind. Knoten, die miteinander kommunizieren, werden durch Linien verbunden.

Hier ein Beispiel dafür, wie Verteilungsdiagramme aussehen:



Werkzeuge

Die Unified Modeling Language (UML) selbst ist keine Methode. Sie ist vielmehr ein Satz von Notationen zur Formung einer allgemeinen Sprache zur Softwareentwicklung. Im Unterschied zu Methoden verfügt UML über keine Notation und keine Beschreibungen für Entwicklungsprozesse. Um UML erfolgreich zu nutzen, ist es notwendig, eine passende Methode zu verwenden, welche die UML unterstützt.

Es gibt eine Reihe von Werkzeugen auf dem Markt, die bei der Erstellung von UML-Modellen helfen. Tatsächlich unterstützt heutzutage praktisch jedes einschlägige Tool UML, zumindest die wichtigsten Diagramme.

Es wird hier keine Liste von Tools gegeben, zumal sich die verfügbaren Werkzeuge ständig ändern. Nur die wichtigsten Werkzeuge seien hier erwähnt:

- *Rose* von Rational Software Corp. Rational ist die "Heimat" der drei Amigos (Booch, Jacobson und Rumbaugh). Allein aus diesem Grund sollte Rose in die engere Wahl genommen werden, außerdem hat es sich zu einer Art Standardtool entwickelt. www.rational.com/rose

- *Together* von Togethersoft (jetzt Borland). Der unabhängige amerikanische Test-Service "Java Applet Rating Service" (JARS) bewertet das Tool als Top 1%-Spitzenprodukt. www.togethersoft.com/products

- *Visio* von Microsoft. Wenn man die UML vorwiegend für Analyse und Design einsetzen will und auf Features wie Codegeneration und Reverse Engineering verzichten kann, wird man auch mit einem Werkzeug wie *Visio*, das vorwiegend ein Diagramm-Zeichen-Werkzeug ist, das Auslangen finden.

Ergänzend zu diesem Text steht Ihnen auf der InfraSoft Website ein Glossar zur Verfügung, in dem Sie die meisten der hier verwendeten Begriffe finden. Über das aktuelle Angebot an weiteren, kostenlosen Fachbeiträgen zur Softwareentwicklung informieren Sie sich bitte unter www.infrasoft.at/service.

Harald Pichler
Wien, im Februar 2003

Der Autor ist Mitarbeiter der InfraSoft, einem Unternehmen, das auf komplexe Softwareentwicklungen spezialisiert ist. Die Experten der InfraSoft haben langjährige Erfahrungen in der Entwicklung und verfügen über fundierte Kenntnisse in Design, Analyse, Realisierung, Test und Projektmanagement.
Für **individuelle Beratungen** zur Entwicklung von Softwarelösungen und die Bereitstellung von **Realisierungsteams** wenden Sie sich bitte an info@infrasoft.at.