

UML Crashkurs

v0.1

UML für Fachinformatiker

von
Hanjo Müller

3. Mai 2005

Inhaltsverzeichnis

1	UML - Unified Modeling Language	3
2	UML im Software Entwurf	4
2.1	Ablauf der Softwareentwicklung	4
2.2	UML Diagrammtypen	4
2.3	Analyse der Anwendungsfälle	5
2.4	Spezifikation der Abläufe zwischen den Anwendungsfällen	6
2.5	Detaillierter Entwurf der Klassen und Datenstrukturen	7
2.6	Zustände eines Objektes	8
2.7	Kurze Erläuterungen zu den verbleibenden Diagrammen	8
3	UML zur Softwaredokumentation	13
4	Kurze Einführung in die Verwendung von CASE Tools	14

1 UML - Unified Modeling Language

Was ist UML

UML steht für Unified Modelin Language und ist eine Entwurfsprache für die Softwareentwicklung. UML ist speziell auf die belange von Objektorientierten Sprachen ausgelegt lässt sich aber in großen Teilen auch für prozedurale Sprachen anwenden.

UML wird seit 1998 entwickelt und ist seit 1997 von der Object Modeling Group zertifiziert. Seit Oktober 2000 ist UML weltweit ISO zertifiziert und wurde offiziell im selben Monat in Version 1.4 veröffentlicht. Aktuell liegt UML in Version 2.0 vor.

Warum dieser Crashkurs

Grundlegend wird UML für die recht theoretische Modelierung von großen Softwareprojekten verwendet, stellt aber zugleich ein komfortables Werkzeug zur Dokumentation von kleinen und mittleren Projekten dar. Mit Hilfe von UML lassen sich die Überlegungen vor der eigentlichen Implementation wesentlich effizienter visualisieren als mit verbalen Mitteln.

Dieser Crashkurs soll es Erleichtern ein Softwareprojekt mit Hilfe von UML zu planen, durchzuführen und zu dokumentieren. Dabei ist keine vollständige Erklärung aller UML Diagramme und Möglichkeiten weder geplant noch möglich.

2 UML im Software Entwurf

2.1 Ablauf der Softwareentwicklung

Nicht nur beim Entwurf von Software mit UML werden im groben die folgenden Schritte eingehalten. Die Ausführung der Schritte muss keinesfalls erschöpfend sein und kann wenn notwendig an einzelnen Punkten (Meilensteine) des Implementationsprozesses wiederholt werden.

1. Analyse der Anwendungsfälle
2. Spezifikation der Abläufe zwischen den Anwendungsfällen
3. Detaillierter Entwurf der Klassen und Datenstrukturen

2.2 UML Diagrammtypen

Allgemein dient UML als Modellierungswerkzeug für Strukturen und Abläufe innerhalb eines Programms. UML setzt dabei da an wo Struktogramme zu groß werden und Programmablaufpläne zu unhandlich. Insgesamt stellt der UML Standard 13 verschiedene Diagramme zur Visualisierung zur Verfügung.

Diese können grob in zwei Kategorien eingeteilt werden. Zum einen die Strukturdiagramme welche sich mehr an der tatsächlichen Implementation und der Struktur der Software orientieren und den Verhaltensdiagrammen, welche sich an den Abläufen innerhalb der Software beschäftigen.

Strukturdiagramme

- Klassendiagramm
- Objektdiagramm
- Komponentendiagramm
- Kompositionsstrukturdiagramm
- Verteilungsdiagramm
- Paketdiagramm

Verhaltensdiagramme

- Anwendungsfalldiagramm
- Zustandsdiagramm
- Aktivitätsdiagramm
- Sequenzdiagramm
- Interaktionsübersichtsdiagramm
- Kommunikationsdiagramm
- Zeitverlaufsdiagramm

2.3 Analyse der Anwendungsfälle

Die Analyse der Anwendungsfälle ist im eigentlichen nichts weiter als die Überlegung welche Funktionen eine Software dem Benutzer oder auch anderen Programmen zur Verfügung stellen soll. Dazu werden alle möglichen Akteure, welches Benutzer oder auch andere Programmkomponenten sein können und alle möglichen Anwendungsfälle aufgeführt.

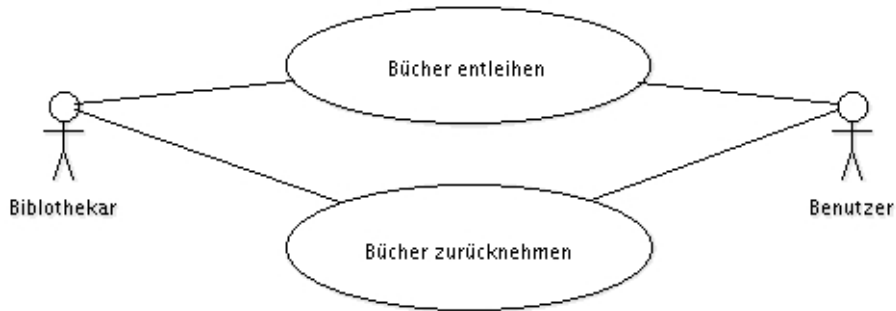


Abbildung 1: Beispiel eines Anwendungsfalldiagramms

Im Beispiel wird eine Bibliothek dargestellt welche die Anwendungsfälle des Entleihens und Zurückgebens von Büchern enthält. Diese Anwendungsfälle werden von zwei möglichen Akteuren ausgelöst und genutzt, zum einen der Bibliothekar welcher das Buch verleiht, bzw. zurücknimmt und zum zweiten der Benutzer welcher das Buch entleiht, bzw zurücknimmt.

Übung

Das Anwendungsfalldiagramm ist um die folgenden Funktionen: An- und Abmelden von Benutzern, Buch im Bestand finden, Buch aus Bestand entfernen und Buch zum Bestand hinzufügen zu erweitern, dabei ist auf die korrekten Assoziationen zwischen Anwendungsfällen und Akteuren zu achten.

2.4 Spezifikation der Abläufe zwischen den Anwendungsfällen

Hat man alle Anwendungsfälle und Akteure erfasst kann man daran gehen die einzelnen Abläufe näher zu untersuchen und zu skizzieren. Hierbei hilft insbesondere das Aktivitätsdiagramm.

Das Aktivitätsdiagramm ist von seiner Struktur her dem Programmablaufplan sehr ähnlich. Es gibt immer einen Start und wenigstens einen Endpunkt, dazwischen liegen verschiedene Einzelaktivitäten, Verzweigungen, Gabelungen und Vereinigungen. Bis auf die letzten beiden wird dies auf durch Programmablaufpläne abgedeckt. Gabelungen und Vereinigungen werden immer dann benötigt wenn innerhalb eines Ablaufs mehrere Prozesse gleichzeitig ablaufen.

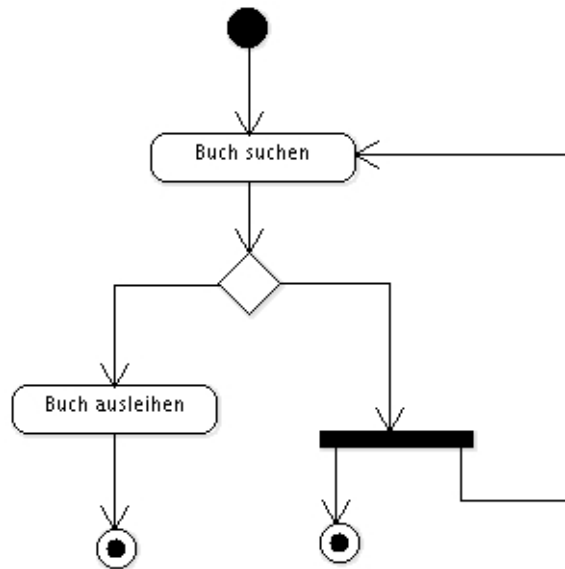


Abbildung 2: Beispiel eines Aktivitätsdiagramms

Im Beispiel wird der Anwendungsfall der Entleihe näher untersucht. Ausgehend von einem Startpunkt muss ein Buch zuerst gesucht und gefunden werden um es zu entleihen. Ist dies nicht der Fall ist auch keine Entleihe möglich, alternativ ist es auch möglich nach einem anderen Buch zu suchen.

Übung

Äquivalent zum oben gezeigten Aktivitätsdiagramm soll ein zweites für den Anwendungsfall der Buchrückgabe erstellt werden. Dabei soll eine Prüfung stattfinden ob das Buch rechtzeitig zurückgegeben wurde oder ob eine Mahngebühr fällig wird.

2.5 Detaillierter Entwurf der Klassen und Datenstrukturen

Kurz zum Thema Objektorientierung

Unter Objektorientierter Programmierung versteht man allgemein das Konzept zusammengehörige Daten und dazugehörige Funktionen in Objekten zusammenzufassen. Grundlegend spricht man von der Abbildung eines allgemeinen Modelles, der Klasse auf eine individuelle Instanz dem Objekt.

Ein Buch der Bibliothek beispielsweise hat eine Inventarnummer, einen Namen, einen Autor und einen Verleihstatus, diese können als Attribute der Klasse Buch bezeichnet werden. Der Zugriff auf diese Attribute ist üblicherweise eingeschränkt, d.h. der Autor und der Name eines Buches ist für niemanden veränderbar. Die Inventarnummer und der Verleihstatus ist seitens der Bibliothek jedoch nicht seitens des Benutzers veränderlich.

Diese Einschränkungen bezüglich des Zugriffsschutzes bezeichnen das Konzept der Datenkapselung. Ein direkter Zugriff auf die Attribute wird unterbunden und jeder notwendige Zugriff durch Zugriffsmethoden geregelt und kontrolliert.

Klassen werden in UML mit Hilfe von Klassendiagrammen beschrieben. Diese enthalten den Namen der Klasse, alle Attribute und alle Methoden. Ein besonderes Konzept der Objektorientierung ist die Vererbung. Dabei kann von einer Klasse eine weitere Klasse abgeleitet werden, diese Klasse verfügt automatisch über alle Attribute und Methoden der übergeordneten Klasse, kann diese neu definieren und weitere hinzufügen.

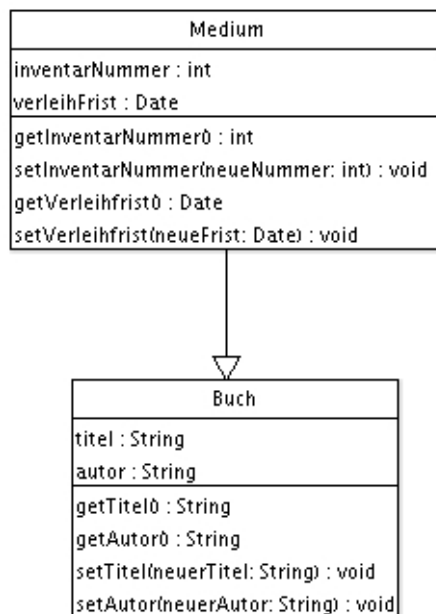


Abbildung 3: Beispiel eines Klassendiagramms

Übung

Das vorhandene Klassendiagramm soll um die Klassen Zeitschrift, Video, DVD, CD und Musikkassette erweitert werden. Dazu sind die notwendigen Klassen mit Attributen und Methoden aufzuzeichnen und auf sinnvolle Vererbung zu achten.

2.6 Zustände eines Objektes

Hat man erstmal die Klassen definiert kann man sich Gedanken darum machen welche Zustände eine solche Klasse denn haben kann. Ein Medium in der Bibliothek beispielsweise kann sich im Bestand befinden oder verliehen sein. Zur Darstellung dieser Zustände nutzt man optimalerweise ein Zustandsdiagramm. Ein Zustandsdiagramm hat immer einen Start und einen Endzustand sowie mehrere Zwischenzustände.

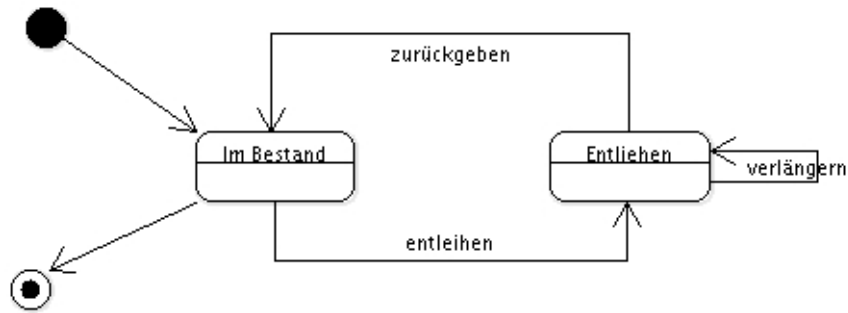


Abbildung 4: Beispiel eines Zustandsdiagramms

Zustandsdiagramme eignen sich auch zur Skizzierung von Benutzerdialogen und Sitemaps. Unter Verwendung der MFCs beispielsweise wird tatsächlich jeder Dialog als Klasse implementiert und hat demzufolge wenigstens zwei Zustände, aktiv und inaktiv.

Übung

Das vorhandene Zustandsdiagramm soll um die Zustände, Beschädigt, Reparatur und Kaputt ergänzt werden.

2.7 Kurze Erläuterungen zu den verbleibenden Diagrammen

Objektdiagramm

Objektdiagramme werden immer dann verwendet wenn bestimmte Einzel- oder Sonderfälle einer Programmstruktur dargestellt werden sollen. In einem Objektdiagramm werden einzelne Objekte und deren Beziehungen dargestellt.

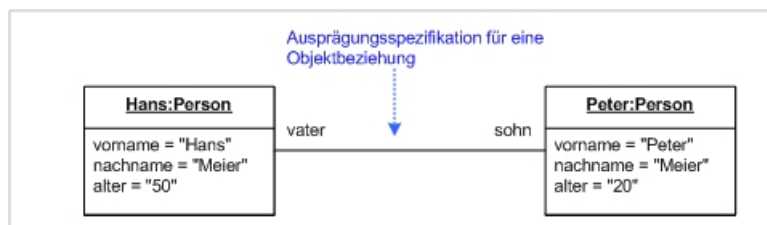


Abbildung 5: Beispiel eines Objektdiagramms

Komponentendiagramm

Komponentendiagramme werden verwendet um die Schnittstellen zwischen einzelnen Programmkomponenten darzustellen. Beispielsweise der Zugriff auf eine Datenbank mittels Webinterface und GUI über die selben Schnittstellen.

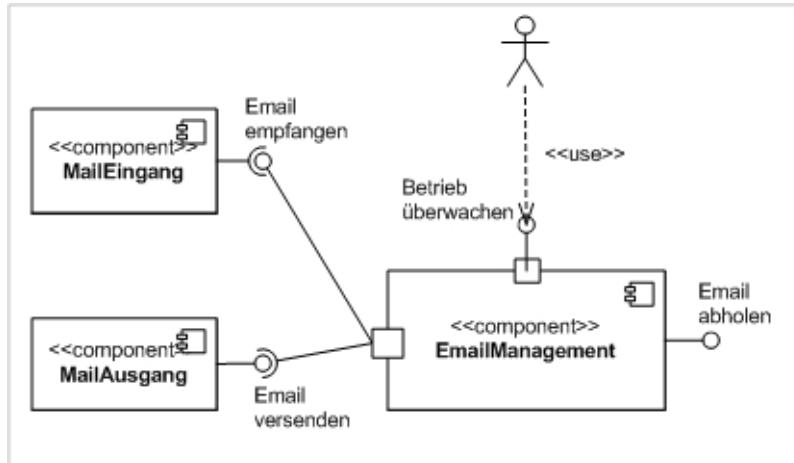


Abbildung 6: Beispiel eines Komponentendiagramms

Kompositionsstrukturdiagramm

Eine Kompositionsstrukturdiagramm stellt die innere Struktur eines Softwaresystems dar. Beispielsweise besteht eine Klasse Auto aus vier Rädern von denen jeweils zwei mit einer Achse verbunden sind.

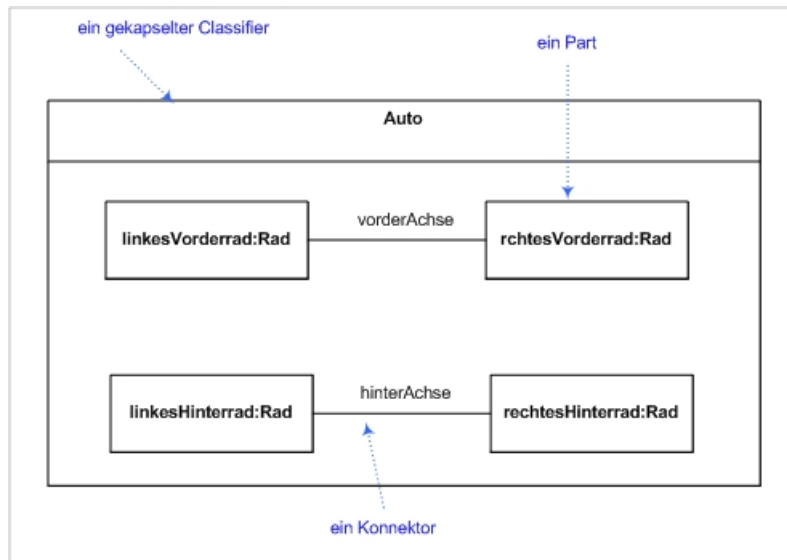


Abbildung 7: Beispiel eines Kompositionsstrukturdiagramms

Verteilungsdiagramm

Mit einem Verteilungsdiagramm lässt sich darstellen wie komplexe Softwaresysteme, welche auf mehr als einem Rechner arbeiten zusammenwirken, beispielsweise über welche Protokolle die Komponenten kommunizieren.

Paketdiagramm

Mit einem Paketdiagramm lassen sich die Gegenseitigen Abhängigkeiten von Komponenten innerhalb eines komplexen Softwaresystems darstellen. Beispielsweise die auch Abhängigkeit von externen Bibliotheken.

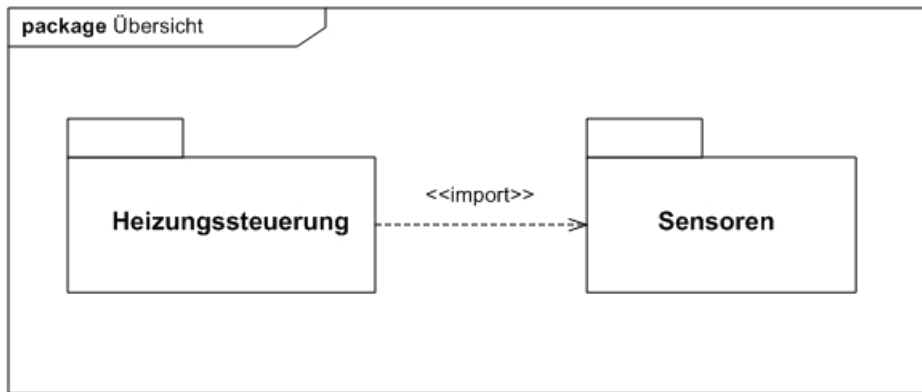


Abbildung 8: Beispiel eines Paketdiagramms

Sequenzdiagramm

Das Sequenzdiagramm wird ebenfalls verwendet um Abläufe darzustellen. Im Gegensatz zu den Aktivitätsdiagrammen wird hier jedoch insbesondere die chronologische Reihenfolge beachtet und dargestellt.

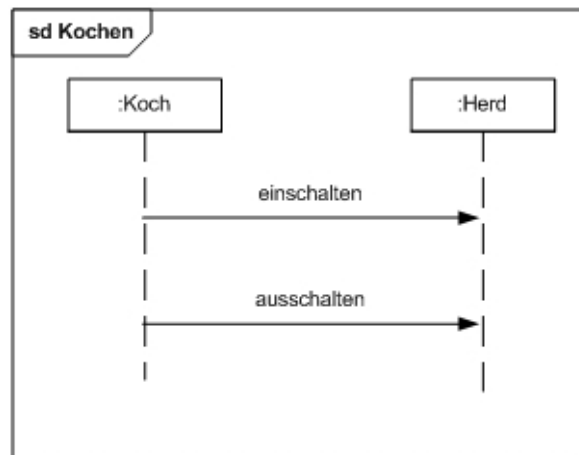


Abbildung 9: Beispiel eines Paketdiagramms

Interaktionsübersichtsdiagramm

Die Interaktionsübersichtsdiagramme werden ebenfalls genutzt um Abläufe darzustellen, jedoch werden hier Sequenzdiagramme mit Interaktionen gekoppelt dargestellt.

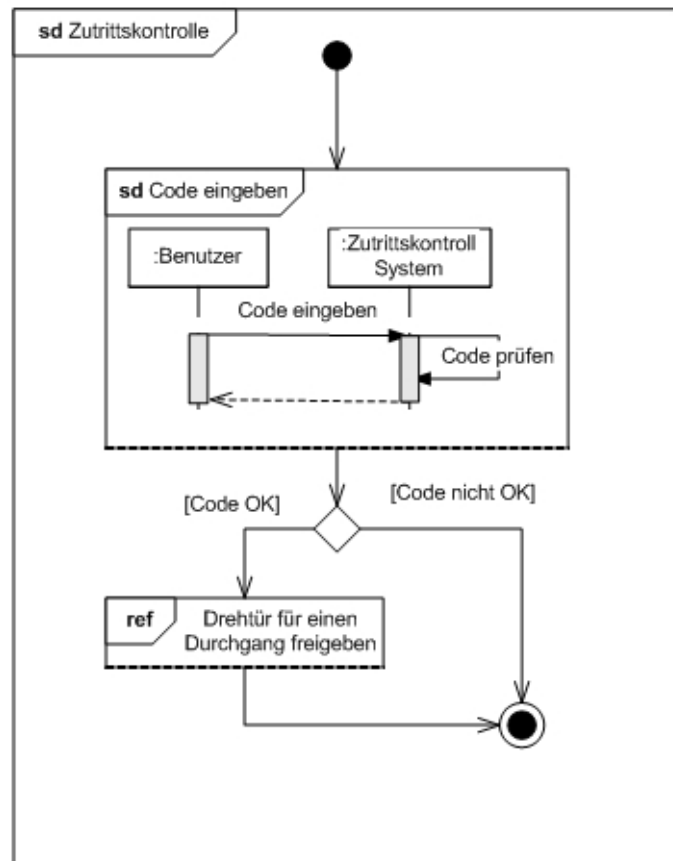


Abbildung 10: Beispiel eines Sequenzdiagramms

Kommunikationsdiagramm

Kommunikationsdiagramme werden verwendet um die Kommunikationswege und -möglichkeiten zwischen Komponenten zu beschreiben.

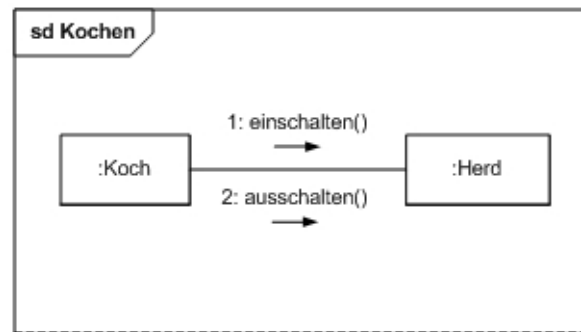


Abbildung 11: Beispiel eines Kommunikationsdiagramms

Zeitverlaufdiagramm

Zeitverlaufdiagramme sind erst kürzlich dem UML Standard hinzugefügt worden und werden genutzt um Zustandsänderungen von Objekten mit zeitlichen Bezügen darzustellen.

3 UML zur Softwaredokumentation

UML eignet sich auch zur Dokumentation von Software. Kommentare im Quellcode und umfangreiche Schriften sind eine Sache. Um die niedergeschriebenen Texte zu unterstützen ist es vor allem bei komplexen Softwaresystemen sinnvoll zumindest einzelne Teile grafisch darzustellen.

Hierzu eignen sich insbesondere, Klassendiagramme zur Darstellung von Datenstrukturen (u.A. auch als Tabellenstrukturdiagramme für Tabellen), Anwendungsfall-, Sequenz- und Aktivitätsdiagramme zur Darstellung von Programmabläufen als auch Komponenten-, Paket- und Kommunikationsdiagramme zur Darstellung von Zusammenhängen innerhalb des Systems.

Statt mit selbsterkreierten Zeichnungen und Pseudocode zu hantieren lohnt es sich allemal gleich auf einen inzwischen etablierten und ausgereiften Standard zu setzen.

4 Kurze Einführung in die Verwendung von CASE Tools

to be written