Leipzig University of Applied Sciences
Dept. of Sciences
04251 Leipzig
Germany

Oberseminar Softwareentwicklung

'UML 2.0 - Ein Überblick'

Sommersemester 2004 - HTWK Leipzig

Gebhardt, Andreas 00IND
20. April 2004

Zusammenfassung

Es wird in dieser Ausarbeitung ein kurzer Überblick über die Diagramme und Notationselemente der UML 2.0 gegeben. Änderungen zu vorherigen Versionen wurden entsprechend gekennzeichnet.

Beispiele und Textpassagen wurden dem Buch *UML 2 glasklar* aus dem *Hanser Verlag* entnommen.

1 Geschichte

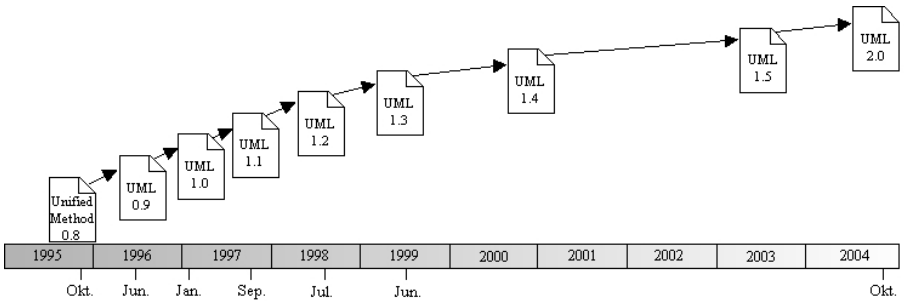

Abbildung 1: UML Historie

- *1994, Oktober*: Grady Booch, Jim Rumbaugh (Rational Software Corporation) zusammenschluß, um erfoglreiche Methoden als Industriestandard (**Unified Method 0.8**)
- *1995, Herbst*: Ivar Jacobsen beteiligt sich an Entwicklung
- *1996, Juni*: **UML 0.9** veröffentlicht
  mehrere Partnerfirmen wirken an Definition mit
- *1997, Januar*: **UML 1.0** verabschiedet; zur Object Managament Group (OMG) zur Standardisierung vorgelegt
  weitere Partnerfirmen
- *1997, November*: OMG verabschiedet **UML 1.1** als Standard
  Weiterentwicklung wurde an die OMG revision Task Force (RTF) übertragen
- *1998, Juni*: **UML 1.2** intern freigegeben, nur redaktionelle Änderrungen
- *1999, Juli*: **UML 1.3** Beseitigung von Inkonsistenzen und kleine Änderrungen
- *2001*: **UML 1.4**
- *2003*: **UML 1.5**
- *2003, Oktober*: **UML 2.0** durch das OMG Board of Directors bestätigt (offizieller Standard bis zur gültigen Spezifikation, vorauss. Oktober 2004, änderbar)

2 Strukturdiagramme

2.1 Klassendiagramm

In einem Klassendiagramm (engl. Class Diagramm) werden die statischen Zusammenhänge eines Systemes dargestellt. Die Klassendiagramme gehören zu den ältesten Bestandteilen der UML. Die Ausprägung, welche dynamisch ist, wird durch Objekte in dem Objektdiagramm dargestellt.

Eine Klasse modelliert mit ihren Operationen und Attributen ein aus dem realen System gefundenen Objekt. Es werden, allen Objekten des realen System gemeinsamen, Attribute und Operationen in der Klasse modelliert. Die Instanz der Klasse, ein Objekt, repräsentiert das Objekt aus dem realen System mit seinen spezifischen Ausprägungen in den Attributen und dem Verhalten der Operationen.

wesentliche Elemente:

- Klassen
  - Attribute
  - Operationen
- Schnittstellen
- Assoziationen
- Generalisierungsbeziehungen
- Abhängigkeitsbeziehungen

2.1.1 Klasse

Notation

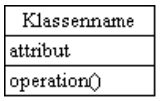

Abbildung 2: Klasse

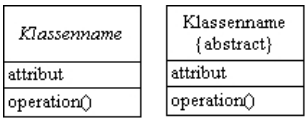

Abbildung 3: abstrakte Klassen

Beschreibung

⚠

Das erste Rechteck enthält den Klassennamen und die weiteren Einteilungen sind nicht vorgeschrieben. Üblich sind die weiteren Einteilungen in *Attribut(e)* und *Operation(en)*. Das Auftreten derer ist geordnet.

Eine Klasse beschreibt eine Menge von Objekten mit gemeinsamer Semantik, gemeinsamen Eigenschaften und gemeinsamen Verhalten. Eine abstrakte Klasse kann niemals Objekte instanziieren, sie dient nur der Modellierung in Form von Generalisierung.

2.1.2 Attribut

Notation

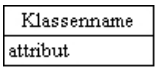

Abbildung 4: Attribut

Beschreibung

```
attribut ::= [Sichtbarkeit] [/] name [:Typ] [Multiplizitaet]
             [= Vorgabewert] [{Eigenschaftswert}]

sichtbarkeit ::= public | private | protected | package
```

Sichtbarkeit

- `+` *public* uneingeschränkter Zugriff
- `-` *private* Zugriff nur für die Klasse
- `#` *protected* Zugriff nur für die Klasse und deren Spezialisierungen
- `~` *package* Zugriff für alle Klassen des selben Paketes

abgeleitetes Attribut: '/' Das angegebene Attribut lässt sich aus den Werten anderer Attribute zur Laufzeit berechnen.

Multiplizität Die Multiplizität legt die Anzahlmäßige Unter- und Obergrenze der ausgeprägten Attributes fest.

- `0..1`
- `1..1`
- `0..*`
- `1..*`
- `n..m`

Die Multiplizität umfasst genau ein Intervall.

Eigenschaftswert Mit den Eigenschaftswerten können bestimmte Attributcharakteristika modelliert werden. Folgende Eigenschaften sind vordefiniert:

- *readOnly* Wert ist unveränderlich
- *union* zugelassene Belegung setzt sich aus *subsets* Eigenschaft zusammen
- *subsets* definiert zugelassene Attributbelegung
- *redifines* überschreiben eines geerbten Attributes
- *ordered* Werte sind duplikatfrei und geordnet

- *bag* Werte sind nicht duplikatfrei und nicht geordnet
- *sequence seq* Werte sind nicht duplikatfrei aber geordnet
- *composite* Attribut übernimmt die Wertzerstörung

Klassenoperation Das Attribut wird bei allen Instanzen der Klasse genutzt.

2.1.3 Operation

Notation

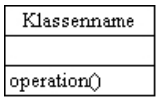

Abbildung 5: Operation

Beschreibung

```
operation ::= [Sichtbarkeit] Name (Parameterliste*) : Rueckgabetyp
              [{Eigenschaftswert}]
Parameterliste ::= [Uebergaberichtung] Name: Typ [Multiplizitaet]
                   [= Vorgabewert] [{Eigenschaftswert}]
Uebergaberichtung ::= in | out | inout | return
```

Übergaberichtung

- *in* Operation liest nur
- *out* Operation schreibt nur
- *inout* Operation liest und schreibt
- *return* ausgewiesener Rückgabeparameter 

2.1.4 Schnittstelle

Notation

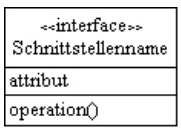

Abbildung 6: Schnittstelle

Beschreibung

Eine Klasse die eine Schnittstelle implementiert und somit die Operationen enthält, wird mit einem Ball (*Ball*) dargestellt. Eine Klasse, welche die Schnittstelle benötigt (d.h. die Klasse ist auf die Existenz der Operationen aus der Schnittstelle angewiesen), wird mit einem Halbkreis dargestellt (*Socket*). Das Zusammenspiel aus Ball/Socket deutet damit auch schon im Modell, auf deren Ineinandergreifen hin.

Eine durch den Stereotyp <<realize>> versehene Abnhängigkeitsbeziehung zwischen Schnittstelle und implementierender Klasse ist durch Ball/Socket ersetzt wurden.

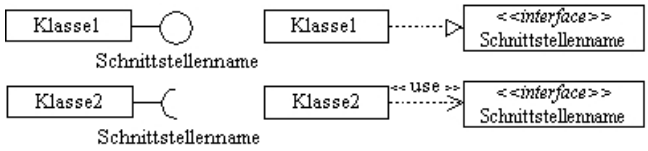

Abbildung 7: Schnittstellennutzung

2.1.5 Parametrisierte Klassen

Notation

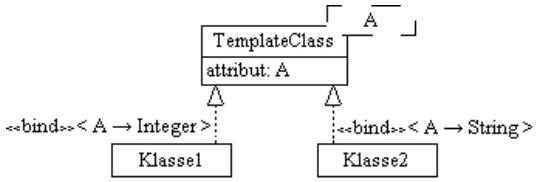

Abbildung 8: parametrisierte Klasse

Beschreibung

Eine parametrisierte Klasse (engl. template class) ist wie eine Schablone mit noch zu bindenden Werten bzw. Typen zu sehen. Von einer parametrisierten Klasse kann keine Instanz gebildet werden, da die Parameter noch nicht gebunden sind. Mit dem Stereotyp *bind* werden die Parameter bzw. Werte gebunden, es ensteht eine Klasse mit gebundnen Paramtern. Die Zuordnung wird als `Parameter -> Typ/Wert` dargestellt. Diese Klasse kann instanziiert werden.

2.1.6 Generalisierung

Notation

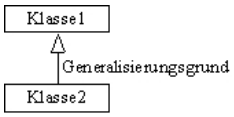

Abbildung 9: Generalisierung

Beschreibung

Die Richtung der Pfeilspitze zeigt in Richtung der Generalisierung. Um den Abstraktionsschritt zu dokumentieren, kann der Generalisierungsgrund neben dem Generalisierungspfeil herangeschrieben werden. Die Generalisierung wird dazu genutzt, um gleiche Eigenschaften und gleiche Semantik von Klassen zusammenzufassen und die Spezialisierungen (welche die Klassen im Detail unterscheidet) in den abgeleiteten neu einzuführen.

Es läßt sich mit der Taxonomie in der Biologie vergleichen, die Einteilung in Gruppe/Rasse/...

2.2 Paketdiagramm

Im Paketdiagramm werden einzelne logisch/semantisch zusammengörige Classifier zu einem Paket zusammengefasst. Dieses wird benötigt, um die Übersicht des Softwaresystems zu behalten und auch die Wiederverwendung von einzelnen Modulen zu unterstützen.

Die UML erlaubt dabei einen beliebigen Grad der Abstraktionsebene (d.h. die Pakete können ineinander verschachtelt werden).

Zu den Versionen UML 1.4 sowie 1.5 wurden keine Änderrungen vorgenommen.

2.2.1 Paket

Notation

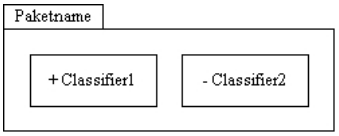

Abbildung 10: Paket

Beschreibung

In einem Paket werden mehrere Classifier zu einer größeren Einheit zusammengefasst, dabei darf jeder classifier nur in genau einem oder in gar keinem Paket auftreten. Die Classifier können innerhalb des Paketes mit den Sichtbarkeiten *private* - oder *public* + gekennzeichnet werden.

Durch das Paket wird ein Namensraum (namespace) aufgespannt, der qualifizierte Name eines Classifiers setzt sich aus dem `Namensraum::Classifiername` zusammen.

In einem Paket werden die vorhanden Pakete von jedem anderen Paket innherhalb diesen gesehen, auserhalb des Paketes regeln die Sichtbarkeiten dieses.

2.2.2 Paket - Import

Notation

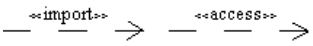

Abbildung 11: Paket - Import

Beschreibung

Wenn ein Paket (Quellpaket) den Namensraum eines anderen Paketes (Zielpaket) erhalten möchte, so muss das Quellpaket den Namensraum des Zielpaketes importieren.

Das Schlüsselwort *public* beschreibt dabei einen öffentliche Paket Import, d.h. der Namesraum ist für andere Pakete, die dieses Importieren auch zugänglich. Das Gegenteil bildet das Schlüsselwort *private*.

2.2.3 Paket - Merge

Notation

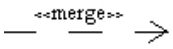

Abbildung 12: Paket - Merge

Beschreibung

Das Paket - Merge stellt eine Erweiterung des Paket - Import dar. Bei einem Paket - Import werden jedeglich die Classifier erreichbar, durch den erweiterten Namesraum, bei einem Paket - Merge hingegen werden neue, spezialisierte, Classifier angelegt, die dann verändert werden können.

Beispiel

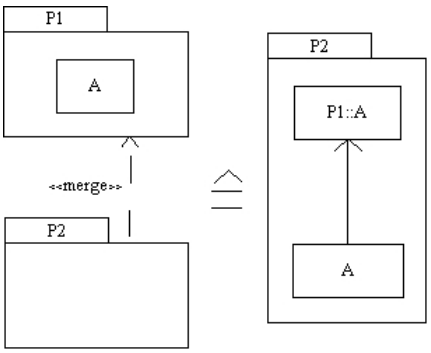

Abbildung 13: Bsp. Paket - Merge

2.3 Objektdiagramm

Im Objektdiagramm (im Original engl. Instances Diagram) wird ein momentanes Bild des Systems dargestellt. Es zeigt die Instanzen der Klassen mit ihren aktuellen Attributen und anderen Ausprägungen.

Wenn komplexe Systemkonfigurationen dargestellt werden sollen, so ist das Objektdiagramm dazu geeignet diese adäquat darzustellen.

2.3.1 Objekt

Notation

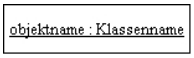

Abbildung 14: Objekt

Beschreibung

Das Objekt beschreibt in der UML die Ausprägungen einer oder mehrere Classifier. Die Ausprägung kann dabei nur teilweise oder folständig spezifiziert sein. Wenn das Objekt eine Instanz einer abgeleiteten Klasse ist, so ist das Objekt eine Ausprägung mehrerer Klassen.

2.3.2 Link

Notation

Abbildung 15: Link

Beschreibung

Der Link stellt die Ausprägung einer Assoziation zur Laufzeit dar. Es können zu einer Assoziation mehrere Links existieren, dies ist von der Multiplizität der Assoziation abhängig.

2.3.3 Wert

Notation

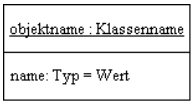

Abbildung 16: Wert

Beschreibung

Der Wert ist eine aktuelle Ausprägung eines Attributes, er muss aber konform zum Typ und der Multiplizität sein. Um eine aktuelles Systembild zur Laufzeit zu geben, ist es sinnvoll nur die Werte darzustellen, die im aktuellen Zeitpunkt relevant sind.

2.3.4 Abhängigkeit

Notation

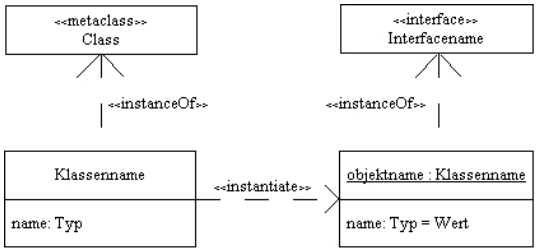

Abbildung 17: Abhängigkeitsnotation im Objektdiagramm

Beschreibung

Um die Abhängigkeit von Klasse und Schnittstelle im Objektdiagramm zu modellieren, kann die Abhängigkeit mit den Stereotypen *instanceOf* oder *initiate* genutzt werden. Die Beziehung ist ist vom Typ zum instanziierten Objekt gerichtet. In umgekehrter Richtung kann die Abstammung mit dem Stereotyp *instanceOf* dargestellt werden.

2.4 Kompositionsstrukturdiagramm

Mit der UML 2 wurde das Kompositionsstrukturdiagramm (engl. Composite Structure Diagram) neu eingeführt. Es soll dabei die Frage *Wie sind die einzelnen Architekturkomponenten strukturiert und wie spielen sie zusammen?* klären. In dem Diagramm wird die interne Struktur eines Classifiers dargestellt. Es wird dabei deutlich die Interaktion derjenigen Stellen gezeigt, die mit anderen Teilen des Systems interagieren. In den frühen Phasen des Entwurfes kann das Diagramm eingesetzt werden, es dient zu abstrakten Dokumentation des zusammenwirkens der einzelnen Architekturkomponenten. In einem Top-Down Prozess können die Systembestandteile Schritt für Schritt verfeinert werden.

2.4.1 Part

Notation

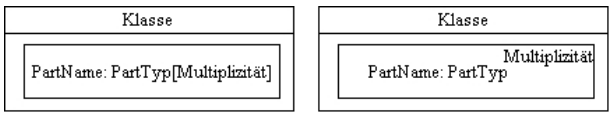

Abbildung 18: Partnotation

Beschreibung

Ein Part repräsentiert eine Ausprägung oder eine Rolle des ihm umgebenden Classifiers (es wird die innere Struktur des Classifiers dargestellt).

Die Ausprägungen können nur im Kontext des Classifiers erzeugt werden und existieren nicht unabhängig von ihm.

In folgenden UML - Diagrammtypen können Parts eingesetzt werden:

- Klassendiagramm
- Objektdiagramm
- Komponentendiagramm
- Verteilungsdiagramm
- Paketdiagramm

Beispiele

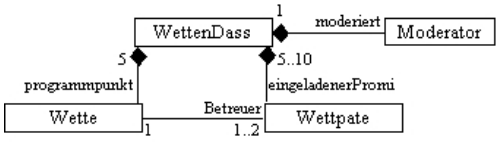

Abbildung 19: Bsp. Klassendiagramm

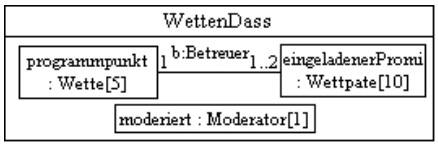

Abbildung 20: Bsp. Partnotation

2.4.2 Port

Notation

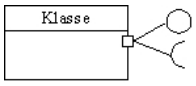

Abbildung 21: Portnotation

Beschreibung

Ein Port repräsentiert eine Stelle, an der ein Classifier mit seiner Umgebung beziehungsweise mit seinen Bestandteilen interagiert. Er spezifiziert die Dienste, die ein Classifier seiner Umgebung an dieser Stelle zur Verfügung stellt oder die ein Classifier von seiner Umgebung benötigt und an dieser Stelle in Anspruch nehmen kann. Dafür wird der Port mit Schnittstellen verbunden, welche die Interaktion mit dem Classifier vollständig spezifizieren.

Beispiel

Wird ein Classifier nur über Ports spezifiziert, so kann dieser ausgetauscht werden. Es genügt die Übereinstimmung der Spezifikation.

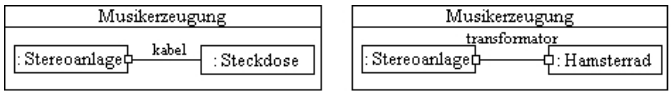

Abbildung 22: Bsp. Portverbindungsmöglichkeiten

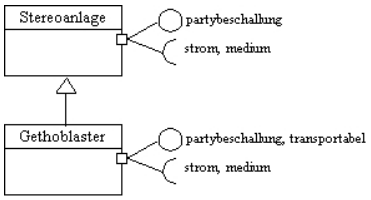

Abbildung 23: Bsp. Portredifinition

2.4.3 Kollaborationstyp

Notation

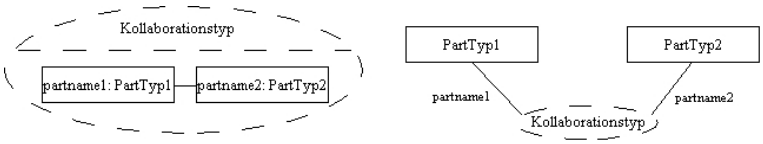

Abbildung 24: Kollaborationstypnotation

Beschreibung

Der Kollaborationstyp gibt eine Sicht auf kooperierende Modellelemente. Er ist nicht instanziierbar aber generalisierbar, da er ein Classifier ist.

Der Kollaborationstyp kann dazu verwendet werden, um die gemeinsame Bewältigung einer Aufgabe durch verschiedene Teile darzustellen. So ist er sehr gut geeignet, um Enwurfsmuster darzustellen.

Beispiel

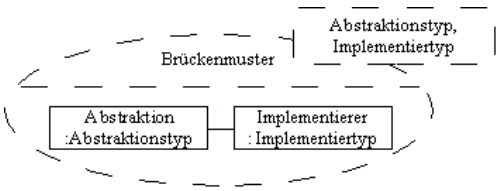

Abbildung 25: Bsp. Entwurfsmuster

2.4.4 Kollaboration

Notation

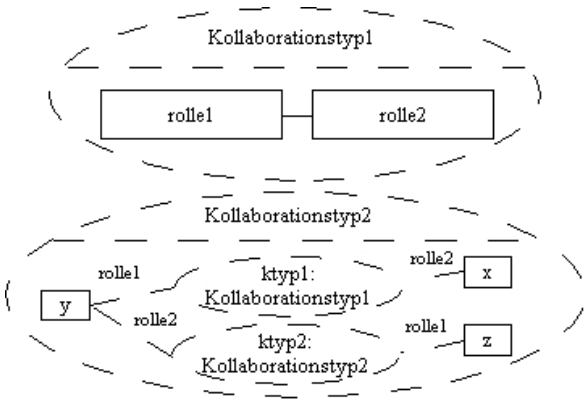

Abbildung 26: Kollaborationsnotation

Beschreibung

Das Verbindungsglied zwischen einem Classifier/Operation und einem Kollaborationstypen ist die Kollaboration. Mit der Kollaboration wird eine ganz bestimmte Situation des Kollaborationstypen dargestellt, somit ist eine Kollaboration immer mit einem Kollaborationstypen verbunden.

2.5 Komponentendiagramm

Das Komponentendiagramm (engl. Component Diagram) stellt verschiedene Bestandteile eines Systems als Komponenten dar. Es stellt die Komponenten so dar, wie sie zur Laufzeit organisiert sin, und zeigt sich daraus ergebende Abhängigkeiten.

Komponenten stellen ein abgegrenztes und über klar definierte Schnittstellen zugreifbares Verhalten bereit. Die konkrete Realisierung einer Komponente kann dabei gegen andere Komponenten, die über dieselben Schnittstellen verfügen, ausgetauscht werden, ohne Änderung am System vorzunehmen.

wesentliche Elemente:

- Komponente
- Artefakt

2.5.1 Komponente

Notation

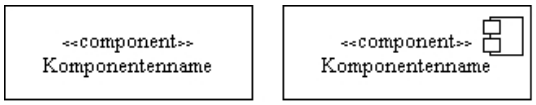

Abbildung 27: Komponente

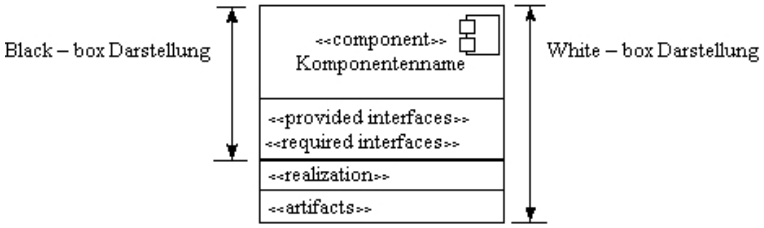

Abbildung 28: White und Black - Box Darstellung der Komponente

Beschreibung

Eine Komponente stellt einen modularen Systemteil dar, der nur über seine Schnittstellen beschrieben ist. Die Schnittstellen können angebotene (provided) und auch benötigte (requiered) sein. Durch diese Beschreibung ist es möglich eine Komponente gegen eine andere auszutauschen, da diese nur über die gleichen Schnittstelen verfügen muss und ansonsten als transparent angesehen werden kann, ohne das System umändern zu müssen.

Mit dem Stereotyp *subsystem* kann eine Komponente gekennzeichnet werden, welch nicht direkt durch den Anwender benutzt werden kann (z.B. E/A System).

2.5.2 Artefakt

Notation

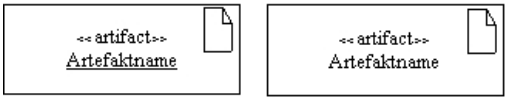

Abbildung 29: Artefakt

Beschreibung

Das Artefakt stellt eine physikalische Informationseinheit dar (z.B. eine Textdatei, Binärdatei etc.).

Innerhalb des Komponentendiagrammes wird das Artefakt verwendet, um konkrete Instanzen einer Komponente darzustellen. Aus der Notation (der Unterstrichene Artefaktname) wird die Ähnlichkeit zu einer Objekt - Klassenbeziehung deutlich.

Die UML sieht folgende Stereotypen vor:

- *file*
- *executable* Sonderform von *file*
- *source* Sonderform von *file*
- *library* Sonderform von *file*

2.5.3 Abhängigkeiten

Notation

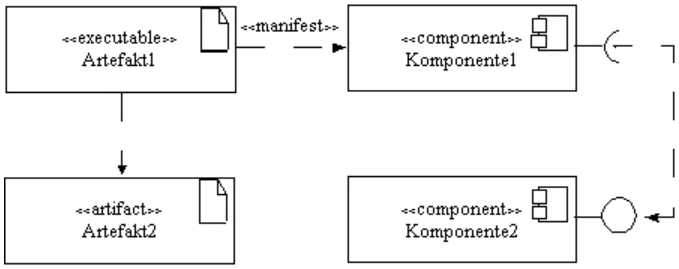

Abbildung 30: Abhängigkeitsnotation im Komponentendiagramm

Beschreibung

Die Semantik der Abhängigkeitsbeziehungen ist dieselbe wie im Klassendiagramm. Schnittstellen mit unterschiedlichen Name können auch dann kombiniert werden, wenn sie die gleichen Spezifikationen haben. Somit kann die nutzende mit der anbietenden Schnittstelle mit einer Abhängigkeitsbeziehung gekennzeichnet werden. Die UML sieht im Komponentendiagramm die weiteren speziellen Stereotypen vor.

- *manifest* verbindet mindestens ein Artefakt mit einer Komponente, dabei kann eine Komponente mehrere Artefakte manifestieren

  - *custom code* beschreibt den Zusammenhang zweier Artefakte, durch manuelle Operation wird das eine in das andere überführt
  - *tool generated* beschreibt den Zusammenhang zweier Artefakte, das eine wird in das andere durch Automatisierung überführt

- *use* (aus dem Klassendiagramm entnommen) verbindet eine Komponente mit der benötigten Schnittstelle

2.6 Verteilungsdiagramm

Im Verteilungsdiagramm (engl. Deployment Diagram) wird zum einen die Hardware Topologie und zum anderen die Verteilung der Softwarekomponenten des Systems auf die Hardware. Es soll ddabei die frage geklärt werden, wie die einzelnen Systemkomponenten zur Laufzeit verteilt werden. Die Verteilung kann dabei mehrere Arten umfassen, wie Installation, Konfiguration, Bereitstellung oder Ausführung von Informationseinheiten (Artefakten).

Die Kommunikationsbeziehungen zwischen den Knoten (Ressourcen) lassen sich ebenfalls im Verteilungsdiagramm darstelen, es wird dabei die Systemsicht um einen weiteren wichtigen Punkt, die der Hardwaresicht ergänzt. Die zur Ausführung des Systems notwendig ist.

2.6.1 Knoten

Notation

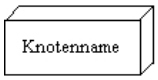

Abbildung 31: Knoten

Beschreibung

Es wird mit dem Knoten eine Ressource des Systems dargestellt. Diese Ressource kann dabei ein zur Installation, Konfiguration, Bereitstellung und Ausführung von Artefakten genutzt werden. Ein Knoten kann dabei auch eine physische Ressource darstellen, ein Gerät .

⚠

Da ein Knoten ein strukturierter Classifier ist, bietet die UML die Möglichkeit, die Knoten zu strukturieren (z.B. schachteln). Dabei wird der Stereotyp *device* verwendet.

Seit der UML 2 kann der Knoten seiner Rolle zu Ausführungszeit feiner spezifiziert werden .

⚠

Beispiel

Abbildung 32: Beispiel Knoten

2.6.2 Kommunikationspfad

Notation

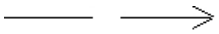

Abbildung 33: Kommunikationspfad (ungerichtet, gerichtet)

Beschreibung

Ein Kommunikationspfad verbindet die verschiedenen Knoten des Systems miteinander, um einen Austausch von Nachrichten zu ermöglichen. Durch die Verwendung von Stereotypen kann die Art der Kommunikation näher spezifiziert werden.

Beispiel

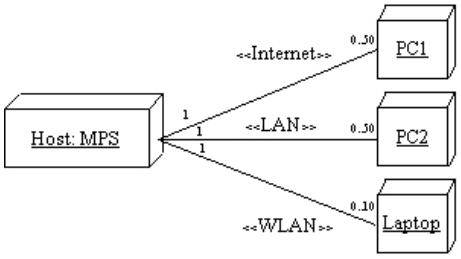

Abbildung 34: Beispiel Kommunikationspfad

2.6.3 Verteilungsbeziehung

Notation

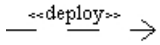

Abbildung 35: Verteilungsbeziehung

Beschreibung

Die Verteilungsbeziehung beschreibt die Beziehung zwischen einem Artefakt und deren dazugehörigen Knoten auf denen das Artefakt verteilt ist.

Beispiel

Abbildung 36: Beispiel Verteilungsbeziehung

2.6.4 Einsatzspezifikation

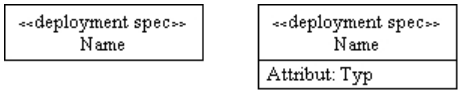

Abbildung 37: Einsatzspezifikation

Notation

Beschreibung

Die Einsatzspezifikation ist eine Spezialform des Artefaktes, sie beschreibt mit derer Menge von Parametern die Verteilung eines Artefaktes auf einen Knoten

Beispiel

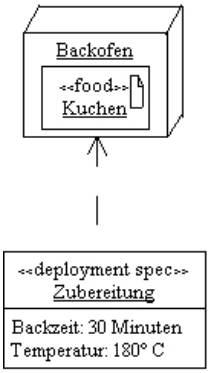

Abbildung 38: Beispiel Einsatzspezifikation

3 Verhaltensdiagramme

3.1 Use - Case Diagramm

Das Use - Case Diagramm (dt. Anwendungsfall zeigt das Verhalten des Systemes nach außen. Die einzelnen Use - Cases stellen dabei für sich eine gekapselte Sammlung von Aktionen dar (als Subsystem).

Aus der Analyse des zu entstehenden Systemes (Dokumente, Anwendungsfälle, Vor - Ort Analyse, etc.) entstehen die einzelnen Uses - Cases, die das Ergebnis der Analyse sind.

Es ermöglicht eine Black - Box Sicht auf das System, mit dem ohne interne Spezifika das Verhalten dargestellt wird. Somit kann es auch genutzt um den Kunden und Anwendern die Funktionalität des Systemes darzustellen.

Ein Use - Case Diagramm enthält die graphische Darstellung

- des Systems,
- der Use - Cases,
- der Akteure außerhalb des Systems und
- der Beziehungen zwischen Akteur und Use - Case, der Akteure, oder Use - Cases untereinander.

Die zentrale Frage *Was soll mein geplantes System am Ende leisten?* wird mit diesem Diagramm geklärt.

3.1.1 Use - Case

Notation

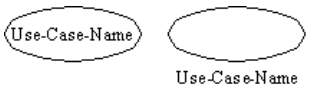

Abbildung 39: Use - Case

Beschreibung

Der Use - Case beschreibt ein Menge von Aktionen, die schrittweise nacheinander ablaufen, um ein spezielles Verhalten darzustelln.

Die UML gewährt bei der Notation viel Freiraum, es muss jedeglich der Use - Case benannt werden.

3.1.2 System

Notation

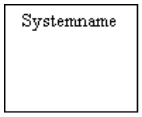

Abbildung 40: System

Beschreibung

Das System ist diejenige einheit, die das Verhalten, welches durch die Use - Cases beschrieben wird, realisiert und anbietet.

Jeder Classifier kann ein Use - Case realisieren.

Beispiel

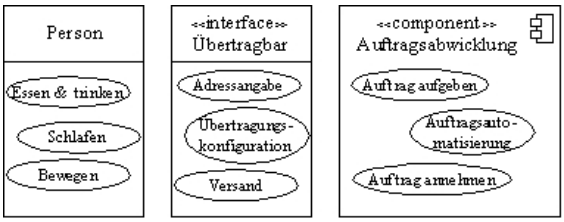

Abbildung 41: Bsp. Use - Case Realisierung durch Classifier

3.1.3 Akteur

Notation

Abbildung 42: Akteur (gebräulichste Form)

Beschreibung

Der Akteur interagiert mit dem System, er stößt dabei Ausführungen an oder ist an der Ausführung aktiv bzw. passiv beteiligt. Seit UML 2.0 muss der Akteur einen Namen tragen.

3.1.4 "include" - Beziehung

Notation

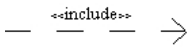

Abbildung 43: include Beziehung

Beschreibung

Die *include* Beziehung visualisiert den Verhaltensimport eines Use - Case, derer am Pfeilende. Das Verhalten des inkludierten Use - Case wird immer ausgeführt und ist *nicht* optional.

3.1.5 "extend" - Beziehung

Notation

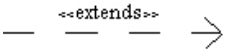

Abbildung 44: extend Beziehung

Beschreibung

Die *extend* Beziehung visualisiert, dass ein Verhalten eines Use - Case erweitert werden kann durch einen anderen (derer an der Pfeilspitze), aber nicht muss.
Die Vorbedingung und die entsprechenden extension points werden als Notiz an die Erweiterung angehängt.

Beispiel

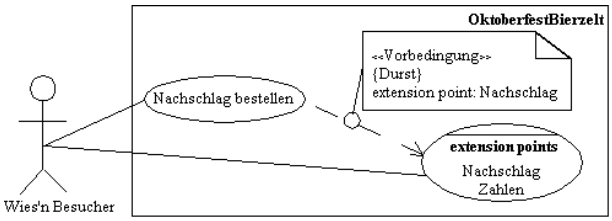

Abbildung 45: Bsp. Vorbedingungen

3.2 Aktivitätsdiagramm

Wenn die Ablaüfe des Systemes modelliert werden sollen, so ist das Aktivitätsdiagramm dafür zuständig. Es soll damit die Frage geklärt werden *"wie realisiert mein system ein bestimmtes Verhalten?"*.

3.2.1 Aktion

Notation

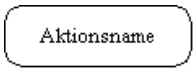

Abbildung 46: Aktion

Beschreibung

Die Aktion steht für den Aufruf eines Verhaltens oder einer Bearbeitung von Daten, die in einer Aktivität nicht weiter zerlegt wurde. Es wird ein Einzelschritt beschrieben, der zur Realisierung einer durch die Aktivität beschrieben Gesamtverhalten beiträgt.

Eine Aktion ist im Aktivitätsdiagram das zentrale Element, alle anderen Elemente dienen nur der Steuerung/Kontrolle der einzelnen ablaufenden Aktionen.

Vor- und Nachbedingung Wenn zu bestimmten Aktionen Vorbedingungen bezüglich einer Konfiguration (z.B. Variablenwert) vorliegen muss, so kann dass im Aktivitätsdiagramm durch einen mit dem Stereotyp *local Precondition* versehenem Notizzettel an der betreffenden Aktion angehäftet werden. Wenn nach einer Aktion eine bestimmte Konfiguration gilt, so kann diese mit einem Notizzettel mit dem Stereotyp *localPostcondition* versehen werden.

Signale und Ereignisse Signale und Ereignisse sind Sonderformen von Aktionen, sie werden als *SendSignalAction* und *AcceptEventAction* bezeichnet.

3.2.2 Aktivität

Notation

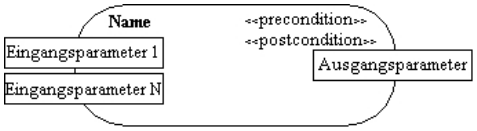

Abbildung 47: Aktivität

Beschreibung

Der Begriff der Aktivitäte ist in UML 2 übernommen, aber mit neuer Bedeutung versehen wurden. Es wird nun die gesamte Einheit, die in einem Aktivitätsmodell modelliert wird gemeint.

Die Aktivität ist schachtelbar, da eine Aktivität wieder eine Aktivität auslösen kann. Wie in der Aktion, besteht auch bei der Aktivität die Möglich Vor- und Nachbedingungen an den Start und an Endpunkt zu notieren.

3.2.3 Objektknoten

Notation

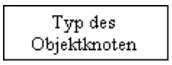

Abbildung 48: Objektknoten

Beschreibung

Ein Objektknoten innerhalb einer Aktivität repraßentiert Ausprägungen eines bestimmten Typs. Objektknoten bilden das logische Gerüst, um Daten und Werte innerhalb einer Aktivität während eines Ablaufs zu transportieren. Ein Objektknoten oder dessen Wert/Inhalt ist das Ergebnis der unmittelbar vorangegangenen Aktion bzw. Eingabe der für die direkt nachfolgende Aktion.

Um den Zusammenhang zwischen Aktivität und Objektknoten zu verdeutlichen, kann die Pin - Notation verwendet werden.

Abbildung 49: Pin - Notation

3.2.4 Kanten

Notation

Abbildung 50: Kanten

Beschreibung

Die Kante (engl. Activity Edge) ist neu in der UML 2, da das Aktivitätsdiagramm nun ein eigenständiges Diagramm ist und nicht mehr eine Sonderform des Zustandsdiagrammes.

Bedingungen, bezüglich der Abarbeitung können an den Kanten in eingeschlossenen eckigen Klammern notiert werden. Neben Bedingungen können Kanten aber auch gewichtet sein, dabei wird die Anzahl der Token bestimmt, die Transportiert werden. Es muss darauf geachtet werden, dass diese nicht die angegebenen Grenzen eines empfangenen Objektes berschreitet.
Die Kanten werden in zwei Arten unterteilt

- *Kontrollfluss* Es findet zwischen zwei Aktionen, oder einer Aktion und einem Kontrollelement, ein (Token-)Fluss statt (d.h. die Abarbeitung wird mit der Kante vortgeführt).
- *Objektfluss* An dieser Kante ist mindestens ein Objektknoten beteiligt, Dabei überträgt die Kante Token, die die Daten oder Werte zum oder vom Objektknoten transportieren.

Um die Übersicht in dem Aktivitätsdiagramm zu behalten, können die Kanten auch unterbrochen werden und an anderer Stelle fortgesetzt.

3.2.5 Kontrollelemente

Notation

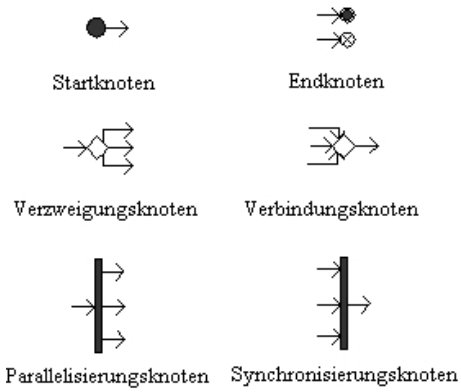

Abbildung 51: Kontrollelemente

Beschreibung

Startknoten Der Startknoten markiert den Startpunkt eines Ablaufes bei Aktivierung einer Aktivität. Beim Start einer Aktivität wird ein *Token* erzeugt, dass allen anliegenden Kanten zur Verfügung steht (vgl. Petrinetze).

Endknoten Es gibt zwei verschiedene Arten von Endknoten, für Aktivitäten und für Kontrollflüsse. Ein Endknoten für Aktivitäten beendet die gesamte Aktivität, wogegen der Endknoten für Kontrollflüsse nur das Ende eines ablaufes markiert.

Die vom Startknoten erstellte(n) Token werden im Endknoten für aktivitäten alle Token und im Endpunkt für Kontrollflüsse nur diejenigen des Ablaufstranges vernichtet.

Verzweigungsknoten Der Verzweigungsknoten spaltet die Kante in mehrere Alternativen auf, die Aufteilung ist von mindestens einer Bedingung abhängig.

Verbindungsknoten Der Verbindungsknoten ist das Gegenstück zum Verzweigungsknoten, er vereint mehrere Kanten zu einer. Es findet dabei keine Synchronistation statt, die anliegenden Token werden auch nicht miteinander verschmolzen, sondern serialisiert der ausgehenden Kante übergeben.

Parallelisierungsknoten Der eingehende Ablauf wird in mehrere parallele Abläufe aufgeteilt, jedes eingegangene Token wird dabei dupliziert und an jeder ausgehenden Kante angeboten. Jegliche Kanten können mit Bedingungen versehen werden.

Synchroniostationsknoten Alle eingehenden Abläufe werden zu einem Ablauf zusammengeführt, dazu müssen alle eingehenden Kanten Token anbieten. Wenn nicht alle Kanten ein Token besitzen, wird solange gewartet, bis alle Kanten ein Token besitzen (Synchronisation).

3.2.6 Parametersatz

Notation

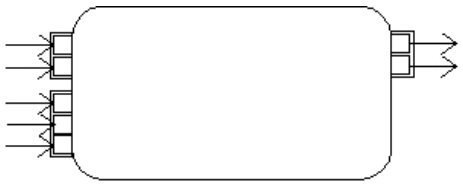

Abbildung 52: Parametersatz

Beschreibung

Ein- un Ausgabeparameter von Objektknoten lassen sich mit Hilfe des Parametersatzes zusammenfassen. Es wird dabei eine implizite Und - Verknüpfung der Ein- bzw. Ausgabepins verstanden, die verküpften Parameter müssen zu einer Sorte gehören.

3.2.7 Unterbrechungsbereich

Notation

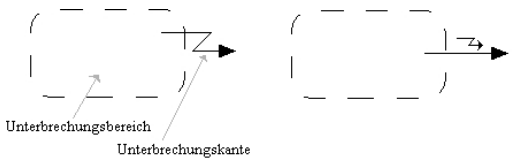

Abbildung 53: Unterbrechungsbereich

Beschreibung

Der Unterbrechungsbereich umschließt eine oder mehrere Aktionen. Wird dieser Bereich mit der Unterbrechungskante verlassen, so werden alle in ihr befindlichen Aktionen abgebrochen und die Token verworfen

3.2.8 Exception - Handler

Notation

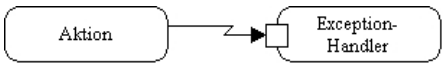

Abbildung 54: Exception - Handler

Beschreibung

Der Excpetion - Handler bearbeitet eine vordefinierte Ausnahme, wenn bei der Abarbeitung der Aktion eine Auftritt. Es wird dabei geprüft, ob zu der Ausnahme ein passender Exception - Handler existiert, dieses wird über die Token geprüft. Sollte kein passender Exception - Handler gefunden werden, so werden alle Token gelöscht und die Ausnahme an die nächst hohere Schicht weitergegeben.

3.2.9 Aktivitätsbereich

Notation

Abbildung 55: Aktivitätsbereich

Beschreibung

Es lassen sich mit Hilfe des Aktivitätsbereiches Aktivitäten mit gemeinsamen Eigenschaften gruppieren. Die UML 2 bietet dabei die Möglichkeit an, diese Bereiche mehrdimensional aufzubauen bzw. hierarchisch anzuordnen.

Beschreibung

3.2.10 Mengenverarbeitungsbereich

Notation

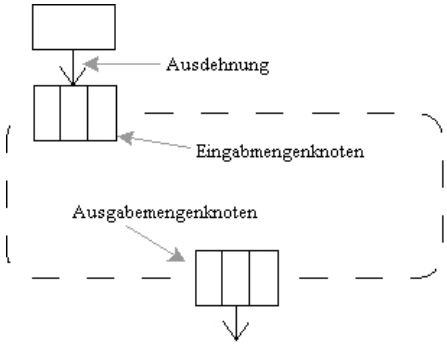

Abbildung 56: Mengenverarbeitungsbereich

Beschreibung

In einigen Situationen treten Mengen von Daten auf, welche verarbeitet werden so z.B. Vektoren, Mengen, etc. . Da die Verarbeitung nur selten im Konkreten gezeigt wird, wird die Verarbeitung als eine Aktion dargestellt. Es ist aber nötig, die Verarbeitung dieser Datenmengen modellieren zu können. Hierzu ist der Mengenverarbeitungsbereich vorgesehen, die Eingehende(n) Datenmengen werden in diesem Bereich einzeln verarbeitet und werden nach Beendigung wieder als Einheit ausgegeben.
Innerhalb des Mengenverarbeitungsbereich können folgende Verarbeitung der Mengen stattfinden

- *Iterative*,
- *Parallel* oder
- *Stream*.

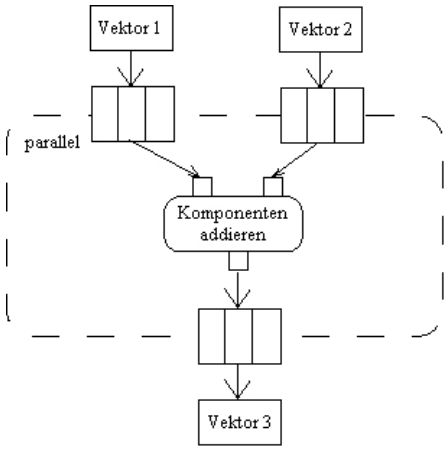

Abbildung 57: Bsp. paralleler Mengenverarbeitungsbereich

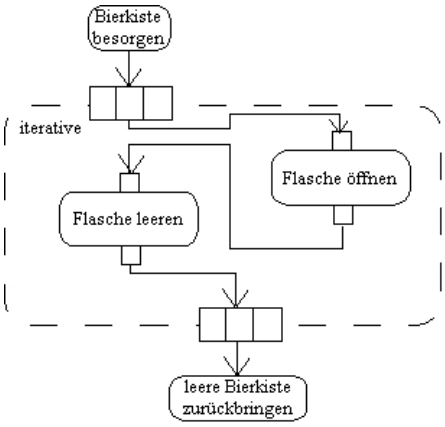

Abbildung 58: Bsp. iterativer Mengenverarbeitungsbereich

3.2.11 Schleifenknoten

Notation

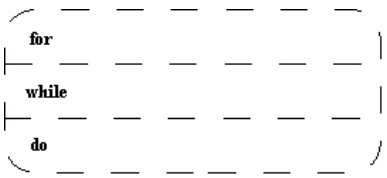

Abbildung 59: Schleifenknoten

3.2.12 Entscheidungsknoten

Notation

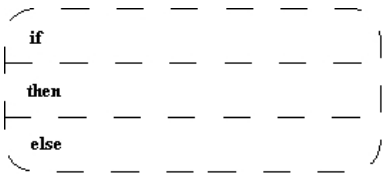

Abbildung 60: Entscheidungsknoten

3.3 Zustandsautomat

Mit dem Zustandsautomaten besteht die Möglichkeit, dass Verhalten eines Classifiers mittels Zustände und Zustandswechsel zu beschreiben. Es wrid dabei die Frage *"Wie verhält sich mein System in einem bestimmten Zustand bei gewissen Ereignissen?"* geklärt.

3.3.1 Einfacher Zustand

Notation

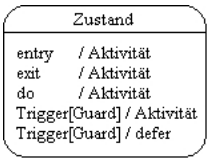

Abbildung 61: Einfacher Zustand

Beschreibung

Mit dem einfachen Zustand wird eine Situation beschrieben, in derem Verlauf eine definierte Bedingung gilt.

- Eintrittsaktivität: *entry* / Aktivität
- Austrittsaktivität: *exit* / Aktivität
- andauernde Aktivität: *do* / Aktivität

3.3.2 Transition

Notation

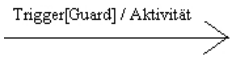

Abbildung 62: Transition

Beschreibung

Die Transition ist der Übergang von einem Zustand zu einem anderen.

- *Trigger*: der Auslöser füt die Transition, werden durch Komma voneinander getrennt
- *Guard*: eine Bedingung, die erfüllt sein muss, dam die Transition bei Triggererhalt durchlaufen wird
- *Aktivität*: die Aktivität, delche beim Durchlaufen der Transition ausgeführt wird

Es gibt in der UML verschiedene Arten von Triggern

- *SignalTrigger*
- *CallTrigger*
- *TimeTrigger*
- *ChangeTrigger*

3.3.3 Startzustand

Notation

Abbildung 63: Startzustand

Beschreibung

Der Startzustand ist der Startpunkt des Zustandsautomaten. Von ihm aus wird der erste Zustand (Startzustand) erreicht, es ist daher nicht erlaubt mehrere abgehende Transition dem Startzustand zuzuordnen.

3.3.4 Endzustand

Notation

Abbildung 64: Endzustand

Beschreibung

Bei erreichen des Endzustandes werden keine weiteren Aktivitäten ausgeführt und die Abarbeitung des umfassenden Zustandsautomaten. Es ist deshalb nicht erlaubt, dem Endzustand Transitionen zuzuweisen.

3.3.5 Zustandsautomat

Notation

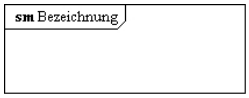

Abbildung 65: Zustandsautomat

Beschreibung

Durch einen Zustandsautomaten wird die Beschreibung über das Verhalten eines Classifiers ermöglicht. Die wichtigsten Elemente sind dabei der *Einfache Zustand*, der *Startzustand* und der *Endzustand*.

3.3.6 Pseudozustände

Notation

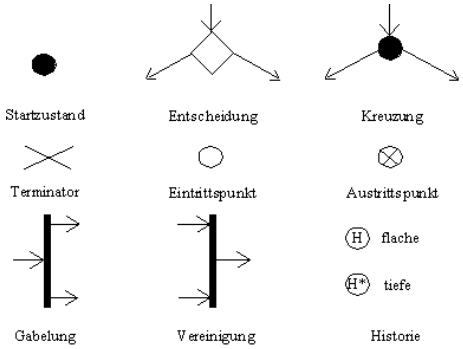

Abbildung 66: Pseudozustände

Beschreibung

Die Pseudozustände wurden in der UML definiert, um komplexe Beziehungen zwischen Zuständen einfacher darstellen zu können.

Startzustand Der Startzustand wird immer da verwendet, wenn der erste aktive Zustand eines Zusandstautomaten gekennzeichnet werden soll.

Entscheidung Um eingehende Transitionen zu einer ausgehenden Transition zusammenzufügen oder in mehrere ausgehende Transition aufzuteilen, wobei die Guards die ausgehende Transition entscheiden, kann mit der Entscheidung dargestellt werden. Diese Form ist dynamisch.

Kreuzung Die Kreuzung wird verwendet, um verschiedene Transitionen ohne verbindende Zustände hintereinander zu schalten. Es muss dabei aber gelten, dass an jeder Kreuzung mindestens je eine eingehende und ausgehende Transistion vorhanden ist. Die Transitionen mussen in ihrer gesamten Spezifikation übereinstimmen, ansonsten können sie nicht die Kreuzung verwenden. Diese ist im Gegensatz zur Entscheidung statisch.

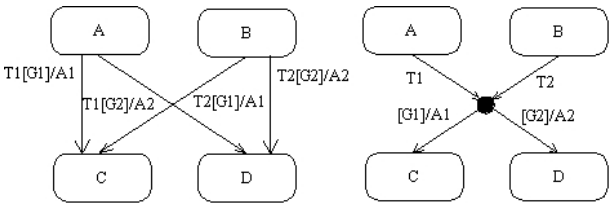

Abbildung 67: Bsp. Kreuzung

Terminator Der Terminator wurde in UML 2 neu eingeführt, um den nicht erfolgreichen Durchlauf, einen Abbruch, eines Zustandandautomaten zu kennzeichnen.

Ein- und Austrittspunkt Es gibt in der UMl die Möglichkeit Zustandsautomaten und Zustände hierarchisch anzuordnen. Um meherer Möglichkeiten des Betretens bzw. Verlassens einer solche Hierarchie zu geben, wurde in der UML 2 die Ein- und Austrittspunkte neu hinzugefügt.

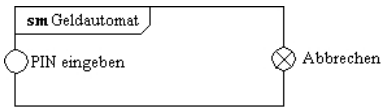

Abbildung 68: Bsp. Ein- und Austrittspunkt

Gabelung und Vereinigung Um eingehende Transitionen auf mehrere, parallel ablaufende, Verhaltensbeschreibungen aufzuteilen, wird die Gabelung verwendet. Wenn mehrere Verhaltensbeschreibungen wieder zu einer Transition zusammengeführt werden sollen, wird ide Vereinigung verwendet.

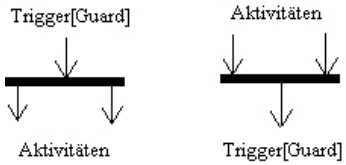

Abbildung 69: Bsp. Gabelung und Vereinigung

flache und tiefe Historie Um sich den letzten aktiven Zustand eines Zustandsautomaten zu merken, dies ist nötig bei der Verwendung von Hierarchie - Stufen, wird die flache bzw. tiefe Historie als Notationsmittel verwendet.

4 Interaktionsdiagramme

4.1 Sequenzdiagramm

Im Sequenzdiagramm werden die Kommunikationspartner und ihre Interaktionen dargstellt. Die zeitlichen Restriktionen sind in diesem Diagramm nicht Vordergründig, dazu wird das neu Eingeführte Timing - Digramm benutzt. Das Sequenzdiagramm gibt die Antwort auf die Frage “Wie läuft die Kommunikation in meinem System ab?”.

4.1.1 Interaktion/Interaktionsrahmen

Notation

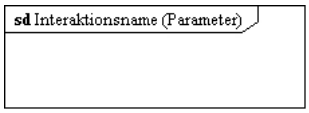

Abbildung 70: Interaktion/Interaktionsrahmen

Beschreibung

Die Interaktion der Kommunikationspartner, allgemein Classifier, werden in diesem Rahmen notiert. **Ein- bzw. Ausgabeparameter** einer Interaktion können genutzt werden, sie werden neben den Interaktionsnahmen in runden Klammer notiert. Auch die **Attribute** eines realisierenden Classifiers können genutzt werden, ohne sie explizit als Parameter zu deklarieren. Sollen **lokale Attribute** genutzt werden, so werden diese im Rahmenbereich der Interaktion deklariert.

4.1.2 Lebenslinie

Notation

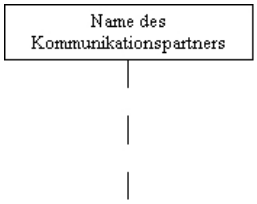

Abbildung 71: Lebenslinie

Beschreibung

Die Lebenslinie repräsentiert genau einen Kommunikationspartner. Die UML versteht unter Teilnehmer dabei alle Teile und Einheiten des umschließenden Classifiers, die zur Kommunikation fähig sind. Insbesondere zählen dazu

- der Classifier selbst (dieser wird mit **self** benannt),
- die Attribute des Classifiers,
- Teile des Classifiers,
- die Operationen des Classifiers,
- Parameter von Operationen und
- Variablen einer Operation.

Der benutzte Unterstrich eines Objektes, bekannt aus den vorherigen UML Versionen, wird nicht mehr genutzt.

Aktionssequenz

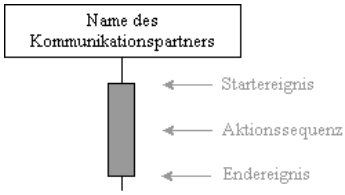

Abbildung 72: Aktionssequenz

Neben dem Empfangen und Senden von Nachrichten führt ein Kommunikationspartner noch andere Tätigkeiten aus. Es zählen dazu die Ausführung von Methoden oder das Ändern von Variablenwerten. Die unterschiedlichen Aktionen können geschachtelt werden.

Destruktion des Kommunikationspartners

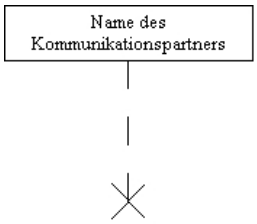

Abbildung 73: Destruktion des Kommunikationspartners

Wird eine Ausprägung eines Kommunikationspartners gelöscht, so wird auch seine Lebenslinie nicht mehr fortgeführt.

4.1.3 Nachricht

Notation

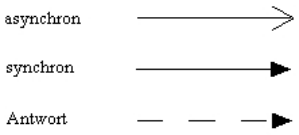

Abbildung 74: Nachricht

Beschreibung

Die Nachricht repräsentiert den Informationsionsfluss zwischen den Kommunikationspartnern. Diese Nachricht muss benannt sein. Es gibt zwei verschieden arten der Kommunikation, die *asynchrone* und die *synchrone* Nachricht. Die Richtung der Kommunikation wird durch die Pfeilspitze angegeben.

Beispiel

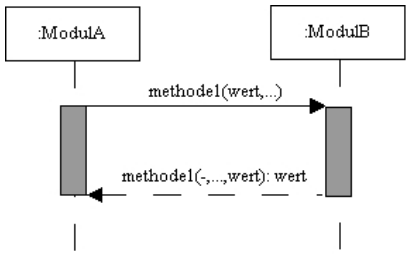

Abbildung 75: Bsp. Nachricht

4.1.4 spezielle Nachrichten

Notation

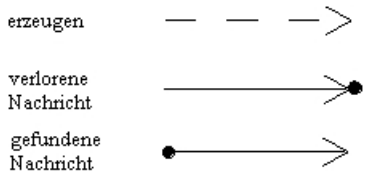

Abbildung 76: spezielle Nachrichten

Beschreibung

Die *erzeugende Nachricht* erstellt einen neuen Kommuniaktionspartner. Bei der *verlorenen Nachricht* ist der Empfänger nicht von Interesse, er steht noch nicht fest oder ist außerhalb des Interaktionsbereiches. Für die *gefundene Nachricht* gilt dasselbe in umgekehrter Richtung.

4.1.5 Zustandsinvariante

Notation

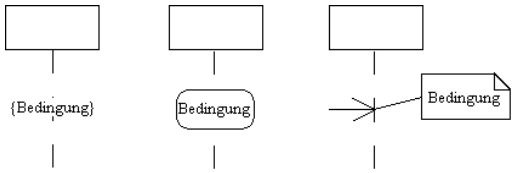

Abbildung 77: Zustandsinvariante

Beschreibung

Die Zustandsinvariante wird unmittelbar vor dem nächsten Ereignis ausgewertet. Diese Bedingung muss zur laufzeit gelten gelten, ansonsten wurde die Interaktion fehlerhaft implementiert.

4.1.6 kombiniertes Fragment

Notation

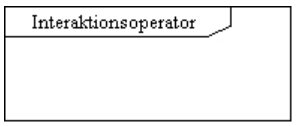

Abbildung 78: kombiniertes Fragment

Beschreibung

Mit Hilfe des *kombinierten Fragmentes* können Interaktionsteile mit bestimmten Regeln versehen werden. Diese Regeln beeinflussen die auswahl und die Reihenfolge der Nachrichten.

Beispiel

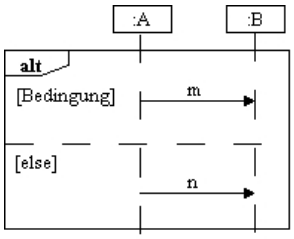

Abbildung 79: Bsp. kombiniertes Fragment

Bezeichnung	Kürzel	Modellieren von ...
alternatives Fragment	**alt**	alternativen Ablaufmöglichkeiten
optionales Fragment	**opt**	optionalen Interaktionsteile
Abbruchfragment	**break**	Interaktionen in Ausnahmesituationen
Negation	**neg**	ungültige Interaktionen
Schleife	**loop**	iterative Interaktionen
Parallele Fragmente	**par**	nebenläufige Interaktionen
lose Ordnung	**seq**	unabhängige chronologische Abläufe
strenge Ordnung	**strict**	abhängige chronologische Abläufe
kritischer Bereich	**critical**	atomare Interaktionen
irrelevante Nachrichten	**ignore**	Filter für unwichtiger Nachrichten
relevante Nachrichten	**consider**	Filter für wichtige Nachrichten
Sicherstellung	**assert**	unabdingbare Interaktion

Tabelle 1: Interaktionsoperatoren

4.1.7 Ordnungsbeziehung

Notation

Abbildung 80: Ordnungsbeziehung

Beschreibung

Eine Ordnungsbeziehung bringt zwei Ereignisse in eine bestimmte Reihenfolge. Der Pfeil verdeutlicht die Reihenolge der Ereignisse.

4.1.8 Interaktionsreferenz

⚠ **Notation**

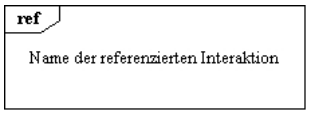

Abbildung 81: Interaktionsreferenz

Beschreibung

Die Interaktionsreferenz zeig innerhalb einer Interaktion auf eine ausgelagerte Interaktion. Mit dieser Auslagerung wird die Übersichtlichkeit der gesamten Interaktion gefördert.

4.1.9 Verknüpfungspunkt

⚠ **Notation**

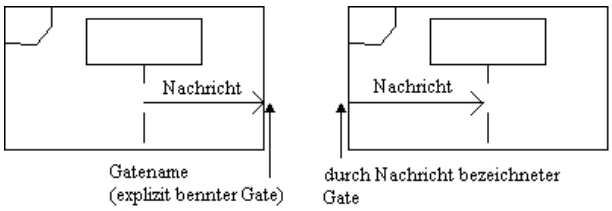

Abbildung 82: Verknüpfungspunkt

Beschreibung

Verknüpfungspunkte sind Kommunikationspunkte, die den Nachrichtenfluss über Interaktionen hinweg erlauben.

4.2 Kommunikationsdiagramm

⚠

Das Kommunikationsdiagramm löst das Kollaborationsdiagramm der vorhergehenden UML Versionen ab. Es ist ein Subset des Sequenzdiagrammes, somit sind z.B. kein Interaktionsverweise, kombinierte Fragmente und keine Ereignisreihenfolgen vorhanden. Es zeigt das Wechselspiel und den Nachrichtenaustausch von Teilen einer komplexen Struktur.

4.2.1 Interaktion/Interaktionsrahmen

Notation

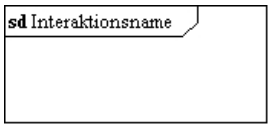

Abbildung 83: Interaktion/Interaktionsrahmen

Beschreibung

In diesem Rahmen werden die Kommunikationspartner und ihre Nachrichten dargestellt. Die Notationsmöglichkeiten sind dem Sequenzdiagramm zu entnehmen.

4.2.2 Lebenslinie

Notation

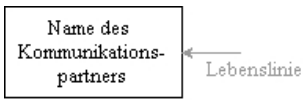

Abbildung 84: Lebenslinie

Beschreibung

Die Lebenslinie aus dem Sequenzdiagramm wird durch die Existenz des Kommunikationspartners ersetzt, die eigentliche Lebenslinie entfällt somit. Sie repräsentiert somit das dasein des Kommunikationspartners. Folgende Konstrukte sind **nicht** einsetzbar

- Attribute von Lebenslinien,
- Aktion und Aktionssequenz,
- Destruktion,
- Zustandsinvariante,
- kombiniertes Fragment,
- Ordnungsbeziehung,
- Interaktionsreferenz,
- Verknüpfungspunkt (Gate),
- und die Zerlegung von Lebenslinien.

4.2.3 Nachricht

Notation

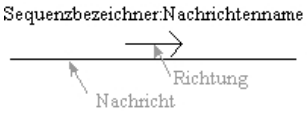

Abbildung 85: Nachricht

Beschreibung

Die Nachricht repräsentiert den Aufruf von Operationen und die Übertragung von Signalen. Es gibt die Möglichkeit den Kontrollfluss der Nachrichten zu steuern.

Sequenzielle und geschatelte Nachrichten Die Reihenfolge der Nachrichten werden durch eine numerische Gliederungshierarchie dargestellt (z.B. 1, 1.1, 1.2, 2, ...). Finden Nachrichten auf einer Ebene statt, so wird eine neue Hierarchieebene eröffnet.

Nebenläufige Nachrichten Um Nebenläufige Nachrichten, die Reihenfolge ist beliebig, im Kommunikationsdiagramm darzustellen, werden bei den betreffenden Nachrichten neben der Sequenznummer noch ein Kleinbuchstabe angehangen (z.B. 1.1a, 1.1b, ...).

Bedingte Nachrichten Wenn Nachrichten nur bei bestimmten Bedingungen versendet werden sollen, so wird diese Bedingung in eckigen Klammern nach der Sequenznummer angehangen (z.B. $1[x > 5]$, $1[x \leq 5]$, ...).

Iterative Nachrichten Wenn eine Nachricht mehrmals versendet werden soll, so wird dies mit Hilfe des Iteratorsterns $*$ dargestellt. Dieser wird nach der Sequenznummer notiert. Die Bedingung wird unmittelbar nach dem Iteratorstern als Pseudocode oder in einer Programmiersprache notiert. Sollen die Nachrichten nebenläufig versendet werden (Broadcast), so wird nach dem Iteratorstern ein Doppelstrich ergänzt.

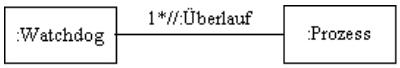

Abbildung 86: Bsp. wiederholte, parallele Nachricht

4.3 Timing - Diagramm

Das Timing - Diagramm zeigt die Zustandsänderungen eines Classifiers, unnter Angabe der *exakten* zeitlichen Bedingungen. Dieses Diagramm ist vor allem in der Digitaltechnik sehr weit verbreitet.

In der *vertikalen* Dimension sind die einzelnen Kommunikationspartner angeordnet. In der *horizontalen* Dimension befindet sich die Zeitachse, auf dieser

wird angezeigt, wann sich ein Classifier in welchem Zustand befindet. Senkrechte Lebenslinien zeigen einen Zustandswechsel und waagerechte einen bestimmten Zustand.

4.3.1 Interaktion/Interaktionsrahmen

Notation

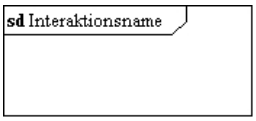

Abbildung 87: Interaktion/Interaktionsrahmen

Beschreibung

In diesem Rahmen werden die Kommunikationspartner und ihre Zustände und Bedingungen dargestellt. Die Notationsmöglichkeiten sind dem Sequenzdiagramm zu entnehmen.

4.3.2 Lebenslinie

Notation

Abbildung 88: Lebenslinie

Beschreibung

Die Lebenslinie in einem Timing - Diagramm ist im Gegensatz zu der im Sequenzediagramm ein ganzer Bereich. Es wird genau ein Teilnehmer, pro Lebenslinie, einer Interaktion dargestellt. Die Reihenfolge der Ereignisse laufen dabei von links nach rechts ab.

4.3.3 Zeitverlaufslinie

Notation

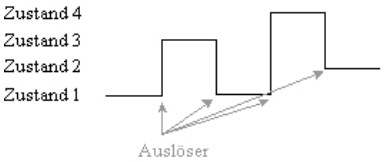

Abbildung 89: Zeitverlaufslinie

Beschreibung

Die Zeitverlaufslinie zeigt die Veränderungen eines Classifiers in seinen Zuständen in Abhängigkeit der Zeit. Mithilfe der, am unteren Interaktionsrahmen befindlichen, Zeitskala ist die Angabe der Zeit und der Zustandsverweildauern *exakt* notierbar.

4.3.4 Nachricht

Notation

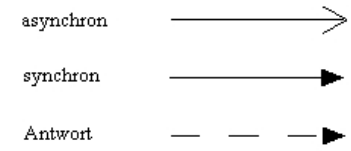

Abbildung 90: Nachricht

Beschreibung

Die Nachrichten im Timing - Diagramm bewirken Zustands- bzw. Attributänderungen bei den interaktionsteilnehmern. Die Zeit, die eine Nachricht zum versenden benötigt, kann mithilfe der Zeitskale auch wieder exakt dargestellt werden.

4.3.5 Sprungmarke

Notation

Abbildung 91: Sprungmarke

Beschreibung

Die Sprungmarke hat keine Auswirkung auf die Semantik der Interaktion, sie unterbricht jedeglich die Nachrichtenlinie, um sie an einer anderen Stelle vortzusetzen. Sie trägt zu einer besseren Übersichtlichkeit des Diagrammes bei.

4.3.6 Wertverlaufslinie

Notation

Abbildung 92: Wertverlaufslinie

Beschreibung

Die Wertverlaufslinie ist eine Alternative zur Zeitverlaufslinie. Sie sollte genutzt werden, wenn

- sehr viele Lenbenslinien,
- vile unterschiedliche Zustände, Attribute oder Bedingungen oder
- folgen von Wertänderungen

dargestellt werden sollen.

Beispiel

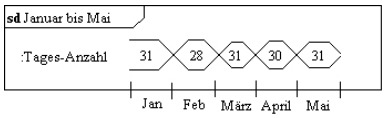

Abbildung 93: Bsp. Wertverlaufslinie

4.3.7 Ordnungsbeziehung

Notation

Abbildung 94: Ordnungsbeziehung

Beschreibung

Eine Ordnungsbeziehung bringt zwei Ereignisse in eine bestimmte Reihenfolge. Der Pfeil verdeutlicht die Reihenolge der Ereignisse.

Beispiel

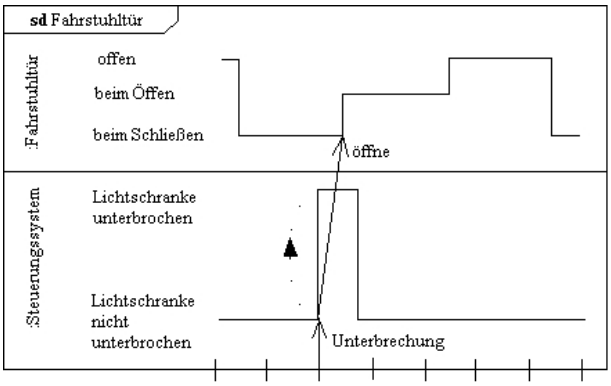

Abbildung 95: Bsp. Ordnungsbeziehung

4.4 Interaktionsübersichtsdiagramm

In dem Interaktionsübersichtsdiagramm werden eine Menge von Interaktionen in einer höheren Abstraktionsebene dargestellt. Dieses Vorgehen ist aus dem immer steigenden Komplexitätsgrad der Systeme zu verzeichnen. Um den Überblick nicht zu verlieren, und das System transparent zu halten. In dem Interaktionsübersichtsdiagramm werden die Kontrollmechanismen des Aktivitätsdiagrammes genutzt.

Dieses neu eingeführte Diagramm leitet sich aus den High Level Message Charts ab.

4.4.1 Interaktion/Interaktionsrahmen

Notation

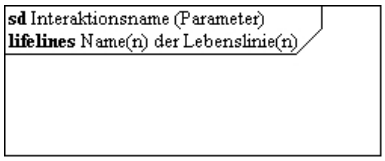

Abbildung 96: Interaktion/Interaktionsrahmen

Beschreibung

In diesem Rahmen werden alle Interaktionen festgehalten. Da diese meißt nur aus Interaktionen bestehen, besteht die Möglichkeit, die beteiligten Kommunikationspartner (Lebenslinien) anzugeben.

Beispiel

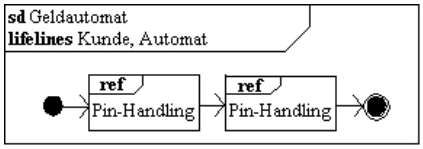

Abbildung 97: Bsp. Interaktion/Interaktionsrahmen

4.4.2 Interaktion/Interaktionsreferenz

Notation

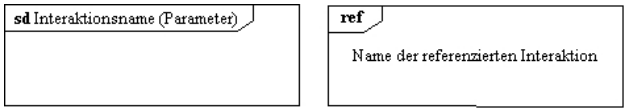

Abbildung 98: Interaktion/Interaktionsreferenz

Beschreibung

Die Interaktionen bzw. Interaktionsreferenzen, die “Orte” der Kommunikation, sind die wesentlichen Bestandteile des Interaktionsübersichtsdiagrammes.

4.4.3 Kontrollelemente

Notation

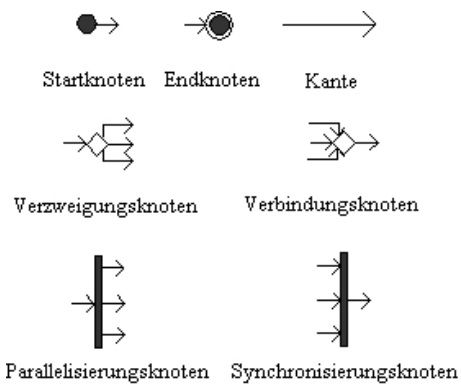

Abbildung 99: Kontrollelemente

Beschreibung

Die Kontrollelemente steuern den Ablauf der Interaktion.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis