Georgische Technische Universität
Humboldt Universität zu Berlin

Klaus Bothe, Gia Surguladze

SOFTWARE ENGINEERING MIT UML

(Lehrmaterialien-2)

unterstützt durch DAAD
(Deutschland)

Berlin - Tbilissi - 2006

bothe@informatik.hu-berlin.de **Prof.Dr. Klaus Bothe**

- 1979 Promotion (A) zur Thematik "Spezifikation und Verifikation abstrakter Datentypen";
- 1986 Promotion (B) zur Thematik "Ein algorithmisches Interface fuer Pascal-Compiler: Compiler-Portabilität durch Modularisierung";
- 09.1986-07.1987 Forschungsaufenthalt am ungarischen Forschungszentrum SZKI in Budapest;
- 09.1991 - 02.1992 Sonderforschungsstipendiat der Humboldt- Stiftung in Erlangen bei Prof. Stoyan;
- Seit 12.1993 Professur fuer Softwaretechnik und Theorie der Programmierung an der Humboldt-Universität zu Berlin.
- Bisherige Arbeitsgebiete: Theorie der Programmierung (Algebraische Spezifikation abstrakter Datentypen); Implementation prozeduraler Programmiersprachen (Compilertechnik); Logische Programmierung und Implementation von Prolog Wissensverarbeitung und Expertensysteme.

Prof.Dr. Gia Surguladze **gsurg@gmx.net**

- 1980 Promotion (A), Elektrotechnisches Institut, St-Peterburg (Leningrad);
- 1991 Sonderforschungsstipendiat DAAD in Erlangen bei Prof. Stoyan (Deutschland);
- 1993 Habilitation (B) an der GTU, Tbilissi;
- 1995,96, 99, 2003 Sonderforschungsstipendiat DAAD in Erlangen bei den Professoren: H. Wedekind,
F. Hofmann, G.. Bolch;
- 2000, 03 Sonderforschungsstipendiat DAAD an der Humboldt Universität zu Berlin, bei Prof. K.Bothe, Prof. W.Reisig;
- 1994 - bis h/Z Professor an dem Lehrstuhl „Management Information Systems“, GTU.
- 2001 – bis h/Z Direktor des Deutsch-Georgischen wissenschaftlichen Gemeinschaftslehrzentrum GTU.
- Forschungsinteressen: UML, OO-Programmierung; Relationale verteilte Datenbanken; Petrinetzen, PNML; .NET Technologie, C#, ADO, ASP.

ISBN 99940-56-01-8

Inhaltsverzeichnis

2. EINFÜHRUNG IN UML

UML – die Unified Modeling Language ist eine Sprache und Notation zur Spezifikation, Konstruktion, Visualisierung und Dokumentation von Modellen für Softwaresysteme.

Die UML ist keine Methode, aber sie kann die Basis für verschiedene Methoden sein, denn sie stellt eine definierte Menge von Modellierungskonstrukten mit einheitlicher Notation und Semantik bereit. Die UML enth]ält eine Vielzahl von Modellelementen und Details. Für den Software-Entwickler ist die UML eine Beschreibung mehrerer Diagrammtypen, die zur Spezifikation der statischen Struktur und des dynamischen Verhaltens der Objekte sowie zur Dokumentation von Implementierungsdetails benutzt werden können (Abb.2.1).

Folgende Diagrammtypen existieren:

- Anwendungsfalldiagramm *- zeigt Akteure, Anwendungsfälle und ihre Beziehungen.*

- Klassendiagramm *- zeigt Klassen und ihre Beziehungen untereinander.*

- Verhaltensdiagramme:

- Aktivitätsdiagramm *- zeigt Aktivitäten, Objektzustände, Zustände, Zustandsübergänge und Ereignisse.*

- Kollaborationsdiagramm *- zeigt Objekte und ihre Beziehungen inklusive ihres räumlich geordneten Nachrichtenaustausches.*

- Sequenzdiagramm *- zeigt Objekte und ihre Beziehungen inklusive ihres zeitlich geordneten Nachrichtenaustausches.*

- Zustandsdiagramm *- zeigt Zustände, Zustandsübergänge und Ereignisse.*

- Implementierungsdiagramme:

- Komponentendiagramm *- zeigt Komponenten und ihre Beziehungen.*

- Verteilungsdiagramm *- zeigt Komponenten, Knoten und ihre Beziehungen.*

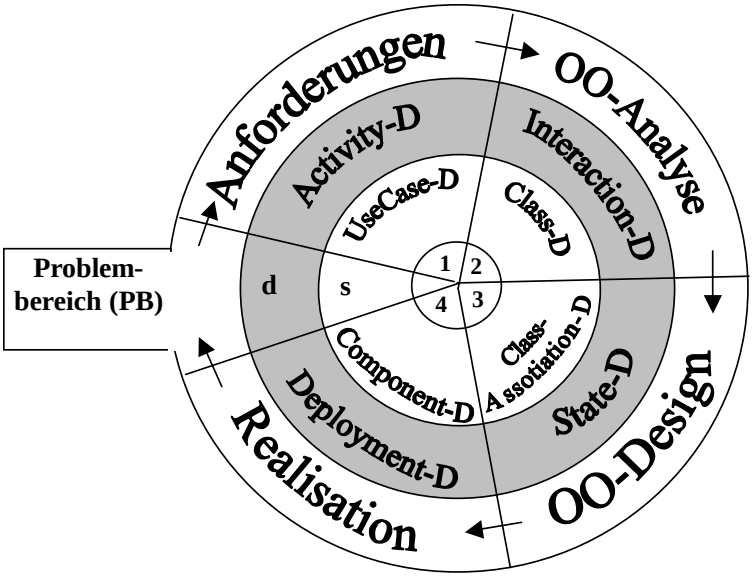

Abb.2.1. Problembereich und 4-Phasen fuer UML Programmierung
d - dynamische Modelle,
s - statische Modelle,
D - Diagramms

Anmerkungen:

- *Software Engineering* ist die Entwicklung, Pflege und Einsatz qualitativ hochwertiger Software unter Einsatz von wissenschaftlichen Methoden, wirtschaftlichen Prinzipien, geplante Vorgehensmodellen, Werkzeugen und quantifizierbaren Zielen.
- Die *Modellierungssprache UML* vielfдltige Mцglichkeiten zur objektorientierten Beschreibung von Software-Systemen zur Verfьgung stellt, die in den meisten Fдllen nur teilweise genutzt werden mьssen.
- *CASE-Werkzeuge* (Computer Aided Software Engineering) unterstьtzen grapische Darstellung der entworfenen Modelle, z.B. Rational Rose, Paradigm-Plus, ObjectivF, Together u.s.w.

Die Anforderungen des Systems können durch eine Anzahl verschiedener ***Anwendungsfälle*** *(Use Cases) ermittelt werden.*

Anwendungsfalldiagramme halten diese Anwendungs-Szenarien fest und beschreiben so den Funktionsumfang der Software aus der Sicht des späteren Benutzers. Anwendungsfälle bilden die Kommunikationsgrundlage mit dem Auftraggeber und sollten zum Entwurf von Testfällen herangezogen werden.

Das zentrale Element eines UML-Modells ist das ***Klassendiagramm*** *(Class Diagram). Darin werden die Klassen und ihre unterschiedlichen Beziehungen dargestellt, wobei die Klassen als Rechtecke und die Beziehungen als beschriftete Verbindungslinien gezeichnet werden. Im Verlauf des Entwicklungs-prozesses nimmt der Detaillierungsgrad dieser Darstellung zu, d.h. die Modelle werden immer präziser, bis sie schließ lich implementiert werden. Zu den Klassen können die Attribute und Operationen angegeben werden.*

Die UML betont die detaillierte Modellierung von ***Assoziationen zwischen Klassen*** *(Assotiation Class Diagram), zu denen Kardinalitäten und Rollennamen der beteiligten Objekte angegeben werden. Eine Aggregation ist eine spezielle Assoziation zwischen zwei Klassen, in der eine Klasse ein Bestandteil der anderen ist. Eine Komposition ist eine Aggregation, bei der eine Klasse ein Attribut einer anderen ist. Vererbungsbeziehungen werden durch Pfeile symbolisiert.*

Mit Hilfe von ***Aktivitätsdiagrammen*** *(Activity Diagram) läß t sich ein Vorgang (Workflow) veranschaulichen. Die verschiedenen Phasen oder Zustände eines z.B. Geschäftsprozesses werden dabei durch Transitionen miteinander verbunden. Dabei wird der Kontrollfluß in Form von Bedingungen und Verzweigungen, und der Datenflulß mit der Weitergabe von Objekten veranschaulicht. Die Verantwortung der einzelnen Objckte wird durch Aufteilen des Diagramms in mehrere Spalten,*

den sogenannten Schwimmbahnen, verdeutlicht.

Das dynamische Verhalten (Interaction Diagrams) von Komponenten kann übersichtlich in ***Sequenzdiagrammen*** *veranschaulicht werden, die den MSC-Diagrammen (Message Sequence Charts) sehr ähnlich sind. Der Nachrichtenaustausch der Objekte wird durch Pfeile angezeigt. Bedingungen, z.B. für Echtzeitsysteme, können angegeben werden.*

Die ***Kooperationsdiagramme*** *(Collaboration Diagrams) berücksichtigen zusätzlich noch die strukturellen Beziehungen. So kann z.B. der Einsatz von Entwurfsmustern dokumentiert werden.*

Mit ***Zustandsdiagrammen*** *(State Diagrams) wird das dynamische Verhalten von Objekten in der Form eines erweiterten Zustandsautomaten beschrieben. So wird das dynamische Verhalten einzelner Objekte präzise spezifiziert. Aus dieser Beschreibung lassen sich leicht Testfälle für die entsprechende Klasse ableiten. Auch Aktivitätsdiagramme können die Funktionsweise von Methoden veranschaulichen.*

Durch ***Komponenten- und Verteilungsdiagramme*** *(Component and Deployment Diagrams) wird die Architektur des Software-Systems und dessen Implementation in einer verteilten Rechnerumgebung (z.B. Client/ Server-Systeme) beschrieben.*

Die UML sieht die Modularisierung von Systemen in Paketen vor. Ein Paket faßt Modellelemente zusammen und regelt die Sichtbarkeit auf die Bestandteile. Die Abhängigkeit der einzelnen Pakete untereinander wird durch Paketdiagramme veranschaulicht. Also, Modellierungssprache UML stellt vielfältige Möglichkeiten zur objektorientierten Beschreibung von Software-Systemen zur Verfügung, die in den meisten Fallen nur teilweise genutzt werden müssen. Für die Hersteller von graphischen CASE-Werkzeugen hat die Standardisierung der UML einen einheitlichen Marktgeschaffen und nicht zuletzt

2.1. ANWENDUNGSFALL- UND AKTIVITÄTSDIAGRAMME

Ein Anwendungsfalldiagramm (Use Case) beschreibt die Zusammenhänge zwischen einer Menge von Anwendungsfällen und den daran beteiligten Akteuren. Es bildet somit den Kontext und eine Gliederung für die Beschreibung, wie mit einem Geschäftsvorfall umgegangen wird.

Ein Geschäftsvorfall ist beispielsweise die schriftliche Schadensmeldung eines Hausrat-Versicherten. Der Geschäftsprozeß (z.B. „Schadensmeldung Hausrat“) beschreibt den gesamten Ablauf, um ein solches Ereignis zu verarbeiten. Der Geschäftsprozeß enthalt dabei unter Umständen auch Aktivitäten die nicht direkt durch Software bzw. die zu entwickelnde Anwendung unterstützt werden (z.B. „Besichtigung des Schadenortes durch einen Sachverstän- digen“).

Anwendungsfälle beschreiben gewöhnlich nur die Aktivitäten, die durch die zu entwickelnde Software unterstiitzt werden sollen, und deren Berührungs punkte zum Umfeld dieser Software. Alle Anwendungsfälle zusammen bilden ein Modell, das die Anforderungen an das externe Verhalten des Gesamtsystems beschreibt. Was genau einen Anwendungsfall ausmacht, wird im nachsten Abschnitt detailliert beschrieben.

Zu beachten ist, daß Anwendungsfälle primer keinen Designansatz darstellen und nicht das interne Verhalten des zukunftigen Systems beschreiben, sondern ein Hilfsmittel zur Anforderungsermittlung sind. Anwendungsfälle sollten nicht zur funktionalen Dekomposition verwendet werden, sie sind kein

Ablaufdiagramme, Datenfluß diagramme oder Funktionenmodelle.

Anwendungsfälle sind vor allem dazu da, die Kommunikation mit den zukünftigen Anwendern, dem Auftraggeber, der Fachabteilung o.ä. zu unterstützen. Anwendungsfälle beschreiben das externe Systemverhalten, d.h. was das System leisten soll. Wie dieses entsteht, d.h. welches Systemdesign und welche Realisierung zu diesem äüß eren Systemverhalten beiträgt, darüber treffen die Anwendungsfälle keine Aussage.

Ein Anwendungsfalldiagramm enthält eine Menge von Anwendungsfällen, die durch einzelne Ellipsen dargestellt werden und eine Menge von Akteuren und Ereignissen, die daran beteiligt sind (Akteure). Die Anwendungsfälle sind durch Linien mit den beteiligten Klassen verbunden. Ein Rahmen um die Anwendungsfälle symbolisiert die Systemgrenzen (Abb 2.2).

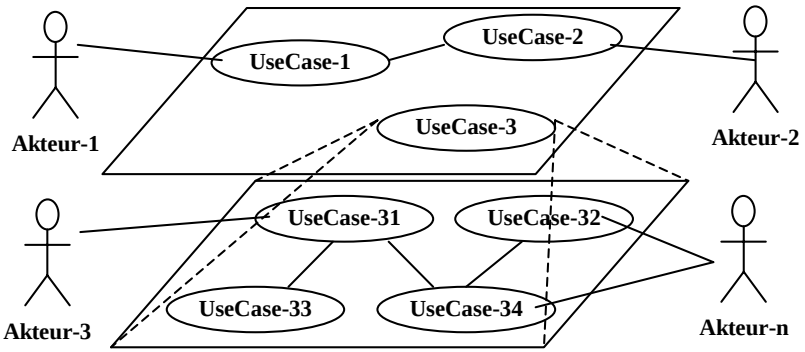

Abb.2.2. Hierarchisches AF-Diagramm

Anwendungsfalldiagramme können hierarchisch verschachtelt werden, d.h. ein Anwendungsfall in einem Anwendungsfalldiagramm kann durch ein weiteres Diagramm in detailliertere Anwendungsfälle untergliedert werden.

Innerhalb eines Diagramms können die Anwendungsfälle durch Beziehungen verbunden werden. Ein Vererbungspfeil von einem Anwendungsfall zum anderen soll zeigen, daß das Verhalten des Anwendungs-falles, auf den der Pfeil zeigt, Teil des Anwendungsfalles ist, von dem der Pfeil losgeht. Der Pfeil kann mit den Stereotypen «uses» oder «extends» beschriftet werden, je nachdem ob eine Benutzt- oder eine Erweiterungsbeziehung ausgedrückt werden sol l .

«uses» oder «benutzt» wird verwendet, wenn das gleiche Stück Anwendungsfallbeschreibung in verschiedenen Anwendungsfällen vorkommt. Um dies zu vermeiden, wird der entsprechende Teil separiert und mit einer «uses»-Beziehung in die andere Anwendungsfallbeschreibung wieder eingebunden. Der Pfeil zeigt in Richtung auf den mitbenutzten Anwendungsfall.

«extends» oder «erweitert» wird verwendet, um Variationen eines Anwendungsfalles zu zeigen, beispielsweise Fehler- und Ausnahmesituationen, spezielle Abweichungen oder Erweiterungen des Standardfalles. Anstelle von «extends»-Beziehungen können Varianten auch direkt als Text im Anwendungsfall beschrieben werden. Der Pfeil zeigt von der Variante zum Standard-Anwendungsfall.

Die Stereotypen «uses» und «extends» sind nützliche, aber entbehrliche Modellkonstrukte, manche Modelliererlnnen verzichten darauf Siehe hierzu.

Das folgende Beispiel zeigt die Verwendung von «uses» und «extends» und leutet gleichzeitig auf die damit verbundenen Schwierigkeiten hin. Das Beispiel zeigt den Anwendungsfall Vertrag schließ en, der unter anderem den Anwendungsfall Kunde identifizieren mitbenutzt (Abb.2.3).

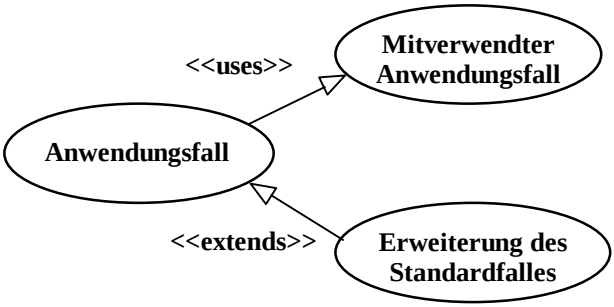

Abb.2.3

Kunde identifizieren vird auch in verschiedenen anderen Kontexten verwendet und existiert deslalb als eigener Anwendungsfall. Innerhalb des Anwendungsfalles Kunde identifizieren wird in bestimmten Fällen auß erdem der Anwendungsfall Kunde neuanlegen verwendet, nämlich immer dann, wenn der Kunde im Kundenbestand noch nicht enthalten ist und neuangelegt werden muß .

Das Anwendungsfalldiagramm zeigt zwar sehr allgemein die Verbindungen zwischen den drei Anwendungsfällen, beschreibt aber keinerlei Details. Man sieht beispielsweise, daß der Anwendungsfall Kunde identifizieren irgendwie durch Kunde neuanlegen erweitert wird, aber nicht wie.

*Das nebenstehende **Aktivitaetsdiagramm** beschreibt diesen Zusammenhang sehr viel konkreter (Abb.2.4).*

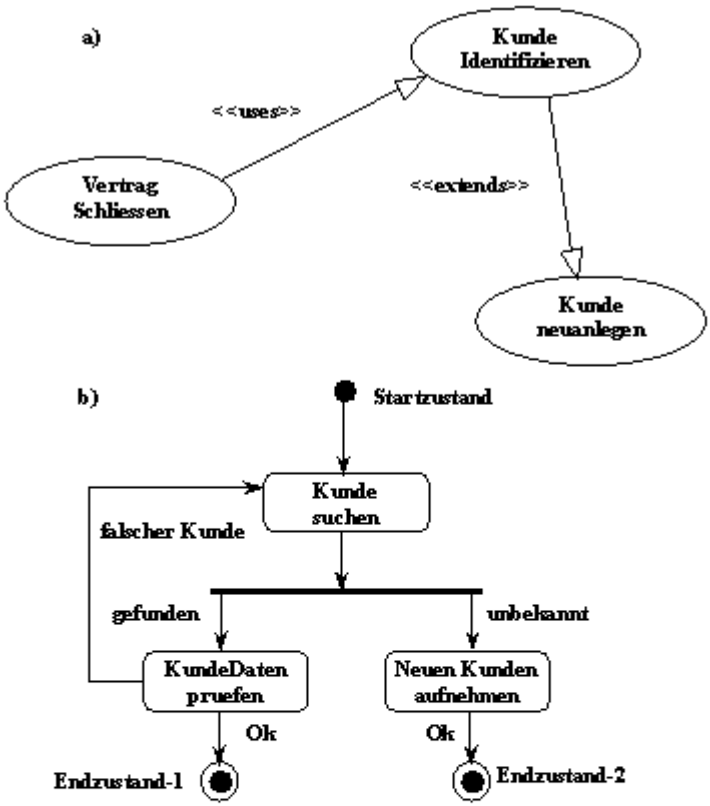

Abb.2.4. Use Case-Diagramm (a) und Aktivitaetsdiagramm (b)

Wie die einzelnen Anwendungsfälle zusammenhängen, d.h. die anwendungsfallübergreifende Beschreibung von Ablaufen, ließe sich zwar textuell innerhalb der Anwendungsfallbeschreibungen notieren, es wäre aber wenig anschaulich. Aktivitätsdiagramme vermitteln solche Zusammenhange visuell und damit einfacher.

2.2. KLASSENDIAGRAMM

Klassen werden durch Rechtecke dargestellt, die entweder nur den Namen der Klasse tragen oder zusätzlich auch Attribute und Operationen. Sie sind durch eine horiyontale Linie getrent (Abb.2.5). Dazu verwendbar sind alle Regeln OO-Modellierung von 1.Kapitel dieses Buch.

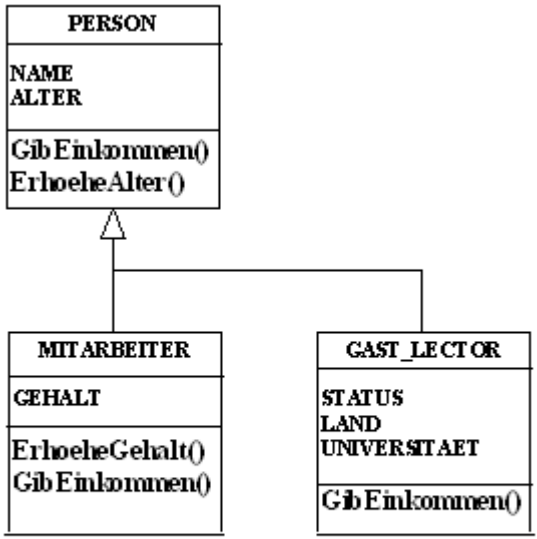

Abb.2.5. Beispiel fuer Klassendiagram

Ein Sequenzdiagramm zeigt eine Reihe von Nachrichten, die eine ausgewählte Menge von Objekten in einer zeitlich begrenzten Situation austauscht, wobei der zeitliche Ablauf betont wird (Abb.2.6).

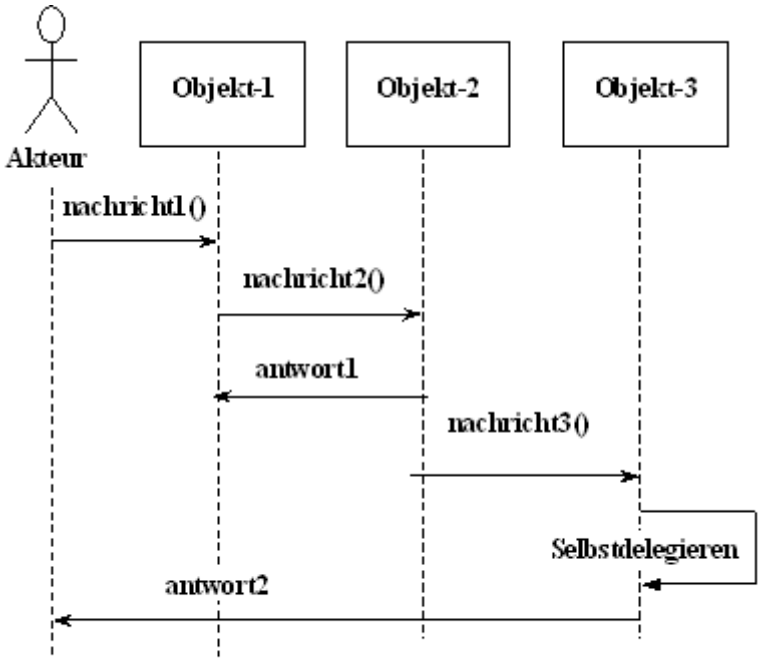

Abb.2.6. Sequenzdiagramm

Ein Kollaborationsdiagramm zeigt eine Menge von Interaktionen zwischen ausgewählten Objekten in einer bestimmten begrenzten Situation (Abb.2.7).

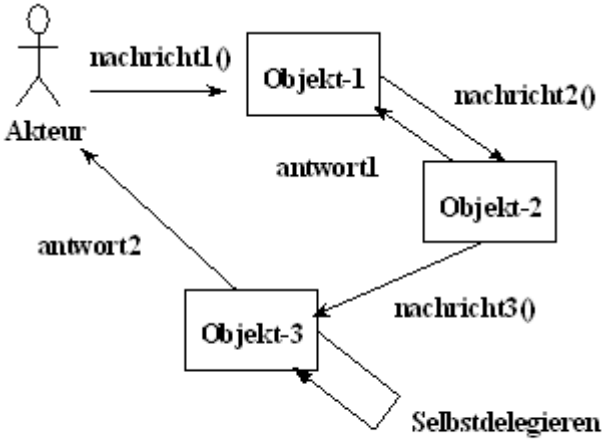

Abb.2.7. Kollaborationsdiagramm

Ein Kollaborationsdiagramm zeigt im Grunde die gleichen Sachverhalte wie ein Sequenzdiagramm, jedoch aus einer anderen Perspektive. Der zeitliche Verlauf der Kommunikation zwischen den Objekten, der beim Sequenzdiagramm im Vordergrund steht, wird beim Kollaborationsdiagramm durch Numerierung der Nachrichten verdeutlicht.

Die Nacrichten werden als Pfeile zwischen den Objekt-Linien gezeichnet. Die vertikale gestrichelte Lebenslinien symbolisieren den Steuerungsfokus (welches Objekt gerade aktiv ist).

2.4. KOMPONENTENDIAGRAMM

Eine Komponente stellt ein physisches Stück Programmkode dar, entweder als Quellkode, Binärkode, DLL oder ausführbares Programm. Komponentendiagrame zeigen die Beziehungen der Komponenten unter einander (Abb.2.8).

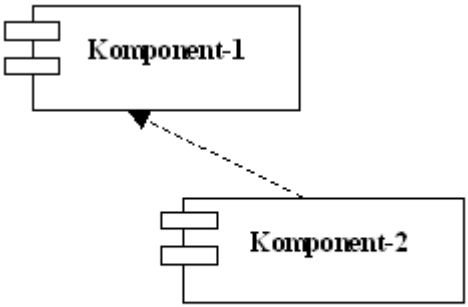

Abb.2.8. Komponentendiagramm

2.5. VERTEILUNGSDIAGRAMM

Verteilungsdiagramme zeigen, welche Komponenten und Objekte auf welchen Knoten (Prozessen, Computern) laufen, also wie diese konfiguriert sind und welche kommunikations-beziehungen dort bestehen (Abb.2.9).

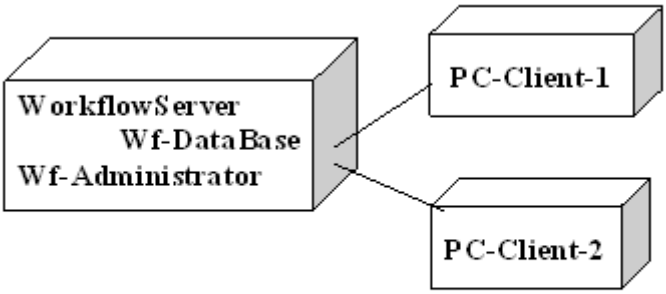

Abb.2.9. Verteilungsdiagramm

2.6. GLOSSAR

***Abhäengigkeit** (dependency) is eine Beziehung zwischen zwei Modellelementen, die zeigt, daß eine Änderung in dem einen (unabhängigen) Element eine Änderung in dem anderen (abhängigen) Element notwendig macht.*

***Akteur** (actor) ist eine außerhalb des Systems liegende Klasse, die an der in einem Anwendungsfall beschriebenen Interaktion mit dem System beteiligt ist. Akteure nehmen in der Interaktion gewöhnlich eine definierte Rolle ein.*

***Aktivität** (action state) ist ein Zustand mit einer internen Aktion und einer oder mehreren ausgehenden Transitionen, die automatisch dem Abschluß der internen Aktion folgen. Eine Aktivität ist ein einzelner Schritt in einem Ablauf. Sie kann mehrere ausgehende Transitionen haben, wenn diese durch Bedingungen unterschieden werden können.*

***Aktivitätsdiagramm** (activity diagram) ist eine spezielle Form des Zustandsdiagramms, das überwiegend oder ausschlieslich Aktivitäten entält.*

***Anwendungsfall** (use case) beschreibt eine Menge von Aktivitäten eines Systems aus der Sicht seiner Akteure, die für die Akteure zu einem wahrnehmbaren Ergebniss führen. Ein Anwendungsfall wird stets durch einen Akteur initiert. Er ist ansonsten eine komplette, unteilbare Beschreibung.*

***Anwendungsfalldiagramm** (use case diagram) zeigtdie Beschreibung ywischen Akteuren und Anwendungsfällen.*

***Assoziation** (assotiation) beschreibt eine Relation zwischen Klassen.*

Assoziationsklasse *(assotiation class) ist ein Modellelement, das sowohl über die Eigenschaften einer Klasse, als auch über die einer Assoziation verfügt (Attributierte Assoziation).*

Assoziationsrolle *(assotiation role) is die Rolle, die ein Typ oder eine Klasse in einer Assoziation spielt. Eine Rolle repräsentiert eine Klasse in einer Assoziation.*

Beziehung *(relationship) ist eine Verbindung zwischen Modellelementen mit semantischem Gehalt. Sie ist Oberbegriff für Assoziation, Aggregation, Komposition, Generalisierung und Spezialisierung.*

Interaktionsdiagramm *(interaction diagram) ist Sammelbegriff für Sequenzdiagramm, Kollaborationsdiagramm und Aktivitätsdiagramm.*

Klassendiagramm *(class diagram) zeigt eine Menge statischer Modellelemente, allem Klassen und ihre Beziehungen.*

Knoten *(node) ist ein physisches Laufzeit-Objekt, das über Rechnerleistung (Prozessor, Speicher) verfügt. Laufzeitobjekte und Komponenten können auf Knoten residieren.*

Komponente *(component) ist ein ausführbares Softwaremodul mit eigener identität und wohldefinierten Schnittstellen (Sourcecode, Binärcode, DLL oder ausführbares Programm).*

Komponentendiagramm *(component diagram) zeigt die Organisation und Abhängigkeiten von Komponrnten.*

Komposition *(composite) ist eine strenge Form der Agregation, bei der die Teile vom Ganzen existenzabhängig sind.*

Objektdiagramm *(object diagram) zeigt Obiekte und ihre Beziehungen untereinander zu einem bestimmten Zeitpunkt.*

***Pakete** (package) sind Ansammlungen von Modellelementen beliebigen Typs. Pakete und Komponenten können zu sehr ähnlichen Zwecken verwendet werden. Während Pakete eine mehr logische Sicht darstellen, betonnen Komponneten die Physische Sicht.*

***Problembereich** (Domäne) bzw. Anwendungsgebiet, innerhalb dessen die fachliche Modellierung stattfindet.*

***Sequenzdiagramm** (sequence diagram) zeigt eine Menge von Interaktionen zwischen einer Menge ausgewählter Objekte in einer bestimmten begrenzten Situation unter Betonung der zeitlicher Abfolge.*

***Szenario** (scenario) ist eine spezifische Folge von Aktionen.*

***Transition** (transition) ist eine Zustandsübergang*

***Verantwortlichkeitsbereiche** (swimlane) sind durch Linien getrennte Bereiche in Aktivitätsdiagrammen, die Verantwortlichkeit der im Diagramm enthaltenen Elemente beschreiben.*

***Verteilungsdiagramme** (deployment diagram) zeigen, welche Komponenten und Objekte auf welchen Knoten (Prozessen, Computern) laufen.*

***Zustand** (state) ist eine Abstraktion der möglichen Attributwerte eines Objektes.*

***Zustandsdiagramm** (state diagram) zeigt eine Folge von Zuständen, die ein Objekt im Laufe seines Lebens einnehmen kann.*

2.7. FRAGEN UND ÜBUNGEN

1. Was versteht man unter UML und CASE ?

2. *Welche Diagrammtypen sind für Sie bekannt ?*
3. *Welche Phasen für Systementwicklung Sie kennen ?*
4. *Was für ein Unterschied gibt es zwischen statischen und dinsmischem Modellen ?*
5. *Was bedeutet der Begriff Use-Case ?*
6. *Was repräsentiert Activity Diagram ?*
7. *Was versteht man unter Interaction Diagram ?*
8. *Wozu braucht man Sequence Diagram ?*
9. *Wozu braucht man Collaboration Diagram ?*
10. *Was für ein Unterschied gibt es zwischen den Sequence und Collaboration Diagrams ?*
11. *Was repräsentiert State Diagram ?*
12. *Was repräsentiert Activity Diagram ?*
13. *Was bedeutet Klassen Diagramm und wie repräsentiert man Kommunikation zwischen Klassen ?*
14. *Was versteht man unter Component Diagram ?*
15. *Was versteht man unter Deployment Diagram ?*
16. *Man konstruiere ein Anwendungsfalldiagramm für den Lehrprozeß an der Universitätslehrstull.*
17. *Worin unterscheiden sich die Begriefe <<uses>> und <<extends>> ?*
18. *Man konstruiere ein Aktivitätsdiagramm für den Lehrprozeß an der Universitätslehrstull.*
19. *Man konstruiere ein Sequenzdiagramm für den Lehrprozeß an der Universitätslehrstull.*
20. *Man konstruiere ein Kollaborationsdiagramm für den*

Lehrprozeß an der Universitätslehrstull.

21. Man konstruiere ein Klassendiagramm für den Lehrprozeß an der Universitätslehrstull.

22. Was ist ein Komponentendiagram ?

23. Was ist ein Verteilungsdiagramm ?

LITERATUR

1. Bothe K., Surguladze G. Modern Platforms and Languages of Programming. Tbilisi, 2003.

2. Booch G., Jacobson I., Rumbaugh J. Unified Modeling Language for Object-Oriented Development. Rational Software Corp., Santa Clara, 1996.

3. Bothe K., Surguladze G.. Inheritance in the Programming of MIS: from the Databases to the UML Technology. GTU, 4(437),2001

4. Seemann J., Von Gudenberg J.W. UML – Unified Modeling Language. „Informatik Spektrum“, N21,1998, Springer, S.89-90.

5. Kahlbrandt B. Software-Engineering. OO Software-Entwicklung mit der UML. Springer, Berlin, 1998.