Die Business Object Notation

Die BON hat eine Reihe von expliziten Zielsetzungen:

- Wiederverwendbarkeit
- Durchgängkeit

Slide 1

- Umkehrbarkeit
- Software Contracting

Aufgabe und Inhalt der BON soll hier grob wiedergegeben werden.

Wiederverwendbarkeit

Wiederverwendbarkeit kann mehrere Vorteile haben, u.a. Kostenreduktion und erhöhte Sicherheit. Allerdings stehen dabei einige Probleme im Weg, z.B. Organisation, Werkzeuge, Kosten, . . .

Jedoch kann Wiederverwendung auf mehreren Ebenen erfolgen:

Slide 2

- Softwarekomponenten (Funktionen, Module, Klassen, Frameworks, . . .)
- Konzepte (Patterns)
- Ideen und Wissen
- Organisation und Werkzeuge

Wiederverwendbarkeit

Der Vorteil objektorientierter Techniken soll sich dabei jedoch auf ähnliche Weise ergeben wie bei der strukturierte Programmierung:

Slide 3

- einfache Konstrukte wie z.B. Schleifen oder Unterfunktionen mußten von Hand programmiert werden
- strukturierte Programmiersprachen fassen solche Konstrukte zu vordefinierten Sprachelementen zusammen
- auch objektorientierte Sprachen fassen gewisse Konzepte zusammen, insbesondere Abstraktion und Kapselung

Durchgängigkeit

Der Vorteil objektorientierter Techniken soll laut den Autoren der BON auch in der durchgängigen Verwendbarkeit der Paradigmen liegen:

Slide 4

- in Analyse, Design und Implementierung kann immer das gleiche Konzept von Klassen und Objekten benutzt werden
- auch Systeme der Realwelt lassen sich einfacher mittels Objekten und Klassen als Funktionen und Daten beschreiben

Umkehrbarkeit

Eine Erweiterung des Begriffs der Durchgängigkeit führt zur *Umkehrbarkeit*. Dabei wird die Möglichkeit der Überführung von Endprodukten wie Quelltext zurück zu abstrakten Beschreibungen wie Klassendiagrammen gefordert. Gründe für diese Forderung liegen auf der Hand: ein System wird meistens in mehreren Iterationen entwickelt, Dokumentation oft nicht nachgeführt oder überhaupt nicht

Slide 5

erstellt, Entwickler vertrauen eher dem Quelltext usw.

Zwangsläufig bedeutet dies, daß die abstrakte Beschreibungsnotation nicht mehr Konzepte enthalten kann als die zugrundeliegende Programmiersprache, da ansonsten Informationen verloren gehen würden (Natürlich bedeutet dies auch das erschwerte Erstellen und Warten von hybriden Systemen, die diese Forderung natürlich nicht erfüllen können).

Software Contracting

Um Komponenten wiederverwenden zu können, muß verstärkt Wert auf Korrektheit gelegt werden. In der BON werden hierfür drei Konstrukte benutzt:

Slide 6

- Vorbedingungen
- Zusicherungen
- Invarianten

Sie formen die Basis für einen *Vertrag* zwischen Kunden- und Anbieterklassen in einem Softwaresystem, welche für alle Beteiligten bindend ist. Ein solcher Vertrag erleichtert zusätzlich auch den Einsatz von Komponenten, ohne deren Implementierung zu kennen.

Eigenschaften der Notation

Bei der eigentlichen Notation werden neben den oben genannten Charakteristika weitere Eigenschaften unterstützt:

Skalierbarkeit: Einsatz sowohl für kleine als auch sehr große Systeme

Slide 7

Typisierte Interfaces: alle Objekte werden statisch über eine entsprechende Klasse beschrieben

Einfachheit: Reduktion der Anzahl der Konzepte

Kompaktheit: Kompakte graphische Darstellung

Diese Eigenschaften stehen in einem **dynamischen** und einem **statischen** Modell zur Verfügung

Systemebenen

Die BON erlaubt die Beschreibung eines Systems auf drei verschiedenen Ebenen:

Systemebene: zeigt das Gesamtsystem, bestehend aus einer Menge von Clustern

Cluster-Ebene: zeigt eine Menge von Klassen und evtl. untergeordneten Clustern

Slide 8

Klassenebene: zeigt eine Menge von Klassen und ihre Beziehungen untereinander

Statisches Modell

Bestandteile des statischen Modells sind als Elemente

- Klassen und
- Cluster

Slide 9

sowie die Beziehungen

- Vererbung und
- Anbieter - Kunde

Darstellungsformen

Für die Elemente sind zwei Darstellungsformen vorgesehen:

Charts: Beschreibung von Elementen auf informeller (untypisierter) Ebene, werden zu Anfang der Analysephase verwendet, um Informationen auch mit Nicht-Entwicklern zu kommunizieren

Slide 10

graphische / textuelle BON: die eigentliche Notation mit formalen (typisierten) graphischen und textuellen Beschreibungsmitteln

Charts

Charts lehnen sich an die auch aus anderen Ansätzen bekannten *Class Responsability Cards (CRC)* an, die für ein Element den Namen, den Einsatzzweck und die wichtigsten Bestandteile auf einer Art Karteikarte aufführen. Der Vorteil solcher Karten ist auf jeden Fall die einfache Handhabung auch ohne Computerunterunterstützung.

Slide 11

In der BON existieren Charts für:

- das Gesamtsystem,
- einzelne Cluster
- und einzelne Klassen.

System Chart

SYSTEM	Systemname	**Teil:** 1/1
Aufgabe: ...	**Index:** ...	

Cluster	**Beschreibung**
...	...
...	...
...	...

Slide 12

Cluster Chart

Slide 13

CLUSTER	Clustername	**Teil:** 1/1
Aufgabe: ...	**Index:** ...	

Klasse / Cluster	**Beschreibung**
...	...
...	...
...	...

Klassen-Chart

Slide 14

KLASSE	Klassenname	**Teil:** 1/1
Art des Objekts: ...	**Index:** ...	

Erbt von	...
Anfragen	...
Operationen	...
Nebenbedingungen	...

Diagrammarten

Slide 15

Die strukturierte (und auch typisierte) Notation wird entweder textuell oder graphisch in

- Klassen-,
- Cluster- und
- Systemdiagrammen

verwendet.

Klassendiagramme

Slide 16

Hauptelement der Klassendiagramme sind die Klassen, welche folgende Attribute besitzen:

Name: Name der Klasse mit zusätzlicher Information über den Status der Klasse

Indexdaten: Metadaten der Klasse (Datum der Erstellung, Autor, . . .)

Oberklassen: Klassen, von denen diese Klasse erbt

Features: Operationen mit ohne Rückgabewert, eingeteilt nach Sichtbarkeitsbereich

Invarianten: immer gültige Nebenbedingungen

Features vs. Attribute

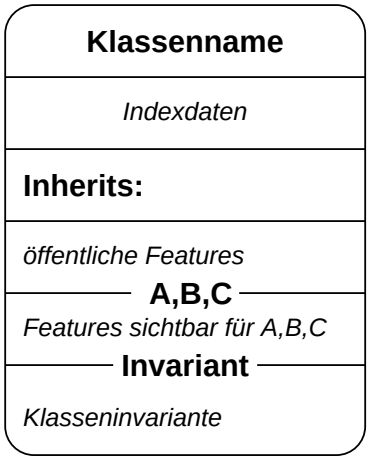
Zu beachten ist, daß keine Attribute (d.h. exportierte Daten) beschrieben werden, da eine Klasse keine internen Zustandsvariable exportieren soll. Dies entspricht einem *abstrakten Datentyp*, der auch keine Daten veröffentlicht (trotz seines Namens!), sondern nur abstrakte Aussagen trifft.

Slide 17

Attribute werden deswegen als Features (Operationen) mit Rückgabewert behandelt, wobei nicht intessiert, ob diese Operation den Rückgabewert errechnet oder er in einer Variable gespeichert ist.

Klassensymbol

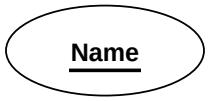
Slide 18



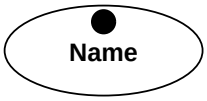
Klassenkopf

Der Klassenkopf besteht normalerweise nur aus dem Klassennamen. Zusätzlich sind jedoch Annotationen möglich:

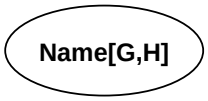
Slide 19



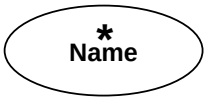
wiederverwendete Klasse



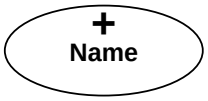
Klasse mit persistenten Objekten



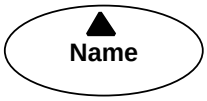
generische Klasse



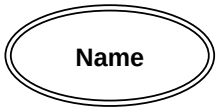
virtuelle Klasse



Klasse, die eine Oberklasse reimplementiert



Interface-Klasse für Systemschnittstellen



Wurzelklasse

Feature-Deklaration

Features werden nach Sichtbarkeitsbereich und Implementierungsstatus eingeteilt:

Slide 20

- Features mit gleichem Sichtbarkeitsbereich werden Sektionen im Klassensymbol zusammengefaßt
- der Implementierungsstatus wird in
  - virtuelle (*name**),
  - implementierte (*name*+) und
  - reimplementierte Features (*name*++) unterteilt

Feature-Signaturen

Die Signatur eines Features besteht aus den Parametern und dem Rückgabewert:

Parameter: → p1: KLASSE, → p2: KLASSE, ...

Rückgabewert: Featurename: KLASSE

Slide 21

Vorbedingungen, Zusicherungen und Invarianten

Klasseninvarianten werden in einer eigenen Sektion im Klassensymbol eingetragen, Vorbedingungen und Zusicherungen stehen beim entsprechenden Feature:

? **Vorbedingung**

Slide 22

! **Zusicherung**

Cluster-Diagramme

Cluster-Diagramme dienen der Darstellung von zusammengehörigen Klassen, wobei ein Cluster auch untergeordnete Cluster enthalten kann. Kriterien für die Zusammengehörigkeit von Klassen sind z.B.

Slide 23

- Funktionalität in einem Subsystem
- bestehende und neue Komponenten
- Hardware und Software
- Abstraktionsgrad

Systemsichten

Die Art und Weise wie Klassen in Clustern zusammengefaßt werden, hängt von der aktuellen *Systemsicht* ab.

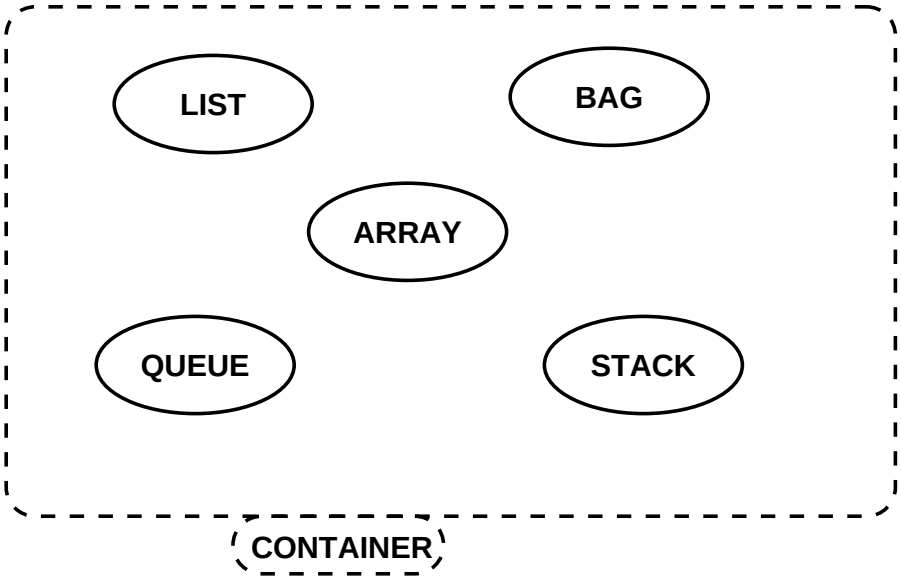
Ein System kann unter mehreren verschiedenen Gesichtspunkten betrachtet werden, die jeweils in einer unterschiedlichen Systemsicht resultieren (in der Praxis wird meistens eine einzige Systemsicht ausreichen).

Slide 24

Cluster sollen nun so gebildet werden, daß in jeder Systemsicht eine Klasse immer nur zu einem Cluster gehört.

Graphische Darstellung

Slide 25



Statische Beziehungen

Die BON beinhaltet nur zwei Beziehungstypen zwischen Klassen und Clustern:

- die Vererbungsbeziehung und
- die Kunde-Anbieter-Beziehung

Slide 26

Dies steht in starkem Kontrast zu anderen Modellierungsmethoden, die sehr viele verschiedene Beziehungstypen verwenden (oft resultierend aus einer Verwandheit oder auch Herkunft der Methoden mit ER-Modellierung).

Vererbung

Slide 27

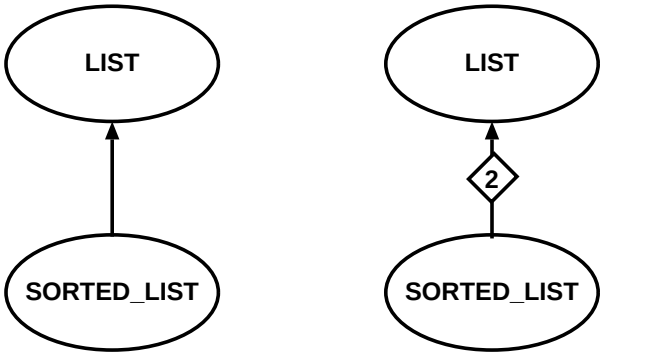
Vererbung kann auf mehrere Arten erfolgen:

- eine Klasse erbt von einer anderen Klasse
- eine Klasse erbt von mehreren anderen Klassen
- eine Klasse erbt mehrfach von derselben Klasse

Der Normalfall ist dabei die Vererbung des Interfaces einer Klasse, d.h. Features können nur reimplementiert werden. Erst die letzte Vererbungsart kombiniert Interface- und Implementierungsvererbung.

Graphische Repräsentation von Vererbung

Slide 28



einfache Vererbung wiederholte Vererbung

Vererbung und Cluster

Auch zwischen Klassen und Clustern und zwischen Clustern selbst kann vererbt werden:

Slide 29

- falls eine Klasse von *allen* Klassen eines anderen Clusters erbt, so besteht eine Vererbungsbeziehung zwischen der Klasse und dem gesamten Cluster
- falls *alle* Klassen eines Clusters von einer anderen Klasse erben, so erbt der gesamte Cluster von der Klasse

Kunde-Anbieter-Beziehung

In der BON werden nur statische Beziehungen zwischen Klassen beschrieben, da davon ausgegangen wird, daß sich daraus die entsprechenden Beziehungen zwischen Objekten ableiten lassen. Dies entspricht dem Prinzip der Umkehrbarkeit, da auch aus einem Quelltext nur die statischen Klassenbeziehungen ablesbar sind.

Slide 30

Als zweiter Beziehungstyp wird deshalb nur die Kunde-Anbieter- Beziehung (kurz *Client-Beziehung*) eingeführt. Diese kann in drei verschiendenen Formen auftreten.

Assoziationen

Eine Assoziation zwischen zwei Klassen besteht dann, wenn zur Ausführungszeit eine Instanz der einen Klasse mit einer Instanz der zweiten Klasse verbunden ist. Dies in folgenden Situationen der Fall sein:

Slide 31

1. ein Feature einer Klasse liefert eine Instanz einer anderen Klasse zurück
2. ein Feature-Parameter einer Klasse ist eine Instanz einer anderen Klasse
3. ein generischer Parameter einer Klasse wird durch eine andere Klasse bestimmt
4. eine Klasse wird mit Hilfe einer anderen Klasse implementiert (durch lokale Variable, private Features und Zustandsvariable)

Geteilte Assoziationen

Geteilte (*shared*) Assoziationen bestehen dann, wenn eine Insanz einer Klasse immer nur mit derselben Instanz einer anderen Klasse verbunden ist.

Eine Erweiterung dieses Assoziationstyps ist die Verbindung zu einer begrenzten Menge von Instanzen einer anderen Klasse.

Slide 32

Aggregation

Slide 33

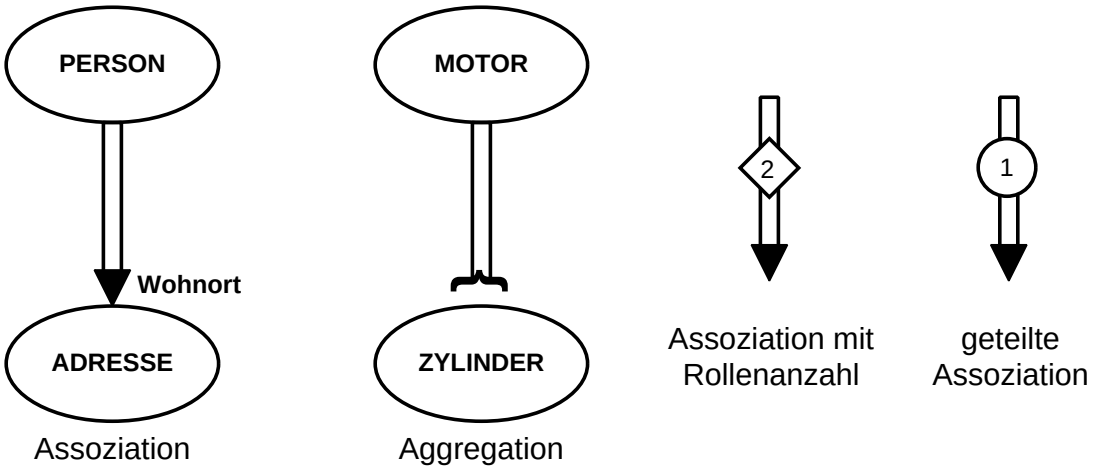
Die Aggregation ist der letzte der drei Client-Beziehungen. Dabei werden *essentielle Bestandteile* des Clients durch andere Klassen gebildet.

Das Konzept der Aggregation kann jedoch verschiedenste Bedeutungen haben, so können z.B. Operationen auf dem übergeordneten Teil entweder auch an die untergeordneten Teile propagiert werden, oder sie betreffen nur das übergeordnete Teil.

Als feste Regel gilt jedoch, daß eine Instanz nur an einer Aggregationsbeziehung teilnehmen kann.

Graphische Darstellung von Assoziationen

Slide 34



Assoziationen und Cluster

Ähnlich wie bei der Vererbungsbeziehung sind auch Assoziationen zwischen Klassen und Clustern bzw. Clustern selbst möglich:

Slide 35

- falls *eines oder mehrere* Elemente in einem Cluster Clients eines anderen Elements sind, so ist der gesamte Cluster Client des Elements
- falls ein Element Client *eines oder mehrerer* Elemente in einem Cluster sind, so ist das Element Client des gesamten Clusters

Elemente können dabei Klassen oder Cluster sein. Der Unterschied zur Vererbungsbeziehung liegt darin, daß bei der Vererbung die erbende Klasse *alle* Eigenschaften der Oberklasse erbt, während es dem Client bei einer Assoziation freigestellt ist, welche Features er nutzt.

Semantische Beziehungen

Andere Beziehungen als die oben genannten werden als *semantische Beziehungen* bezeichnet. Beispiele sind z.B. „ A *benutzt* B “, „ A *implementiert* B“, usw.

Diese Bezeichnungen werden als Kommentare zu den Vererbungs- bzw. Assoziationsbeziehungen angefügt.

Slide 36

Das dynamische Modell

Ergänzt wird das statische Modell um ein dynamisches Modell, dessen Hauptbestandteile *Szenario-Charts* und *Objektszenarien* sind.

Die dynamische Modelle sind jedoch häufiger Änderungen unterworfen, außerdem können sie aus Komplexitätsgründen immer nur Aussschnitte aus dem Gesamtsystem zeigen. Deshalb werden sie teilweise nur temporär oder nur zur Unterstützung des Entwurfsprozessen eingesetzt.

Slide 37

Auf eine ausführliche Darstellung wird an dieser Stelle verzichtet.

Die Methode

Die Methode, die angewendet wird, um das System zu modellieren, ist in drei Phasen aufgeteilt:

1. Sammeln von Analyse-Informationen
2. Beschreibung der zusammengetragenen Struktur
3. Entwurf des Systems

Slide 38

In jeder Phase soll dabei *risikoorientiert* vorgegangen werden, d.h. es sollen die problematischsten und riskantesten Elemente zuerst behandelt werden.

Sammeln von Analyse-Informationen

Die Aufgaben in der ersten Phase sind:

Finden der Systemgrenze: Identifikation von wichtigen Subsystemen, Metaphern, Use-Cases, Szenarien

Slide 39

Finden von potentiellen Klassen: Erstellen eines Glossars wichtiger technischer Ausdrücke

Auswahl und Gruppierung von Klassen: Klassifizierung, Identifikation von zusammengehörigen Klassen

Als Produkte werden System- und Cluster-Charts und eine erste statische Klassenstruktur erstellt.

Beschreibung der Struktur

In der nächsten Phase wird die erste gefundene Struktur weiter beschrieben:

Definition der Klassen: Bestimmen der Operationen, Anfragen und Invarianten

Beschreibung des Systemverhaltens: Identifikation von Ereignissen, Instanziierungen und relevanten Objektszenarien

Slide 40

Definition exportierter Features: Spezifikation typisierter Signaturen und formaler Beziehungen

In diser Phase werden Event-Charts, Objektszenarien und Klassendiagramme erzeugt.

Entwurf des Systems

Während des Systementwurfs fallen folgende Aufgaben an:

Verfeinerung des Systems: Identifikation neuer Designklassen und neuer Features

Slide 41

Generalisierung: Ausfaktorierung gemeinsamen Verhaltens

Komplettierung und Review: Entwurf der endgültigen statischen Struktur

Analyse- und Designklassen

Wie oben gesehen werden in späteren Phasen sog. *Designklassen* hinzugefügt. Die Einteilung in Analyse- und Designklassen erfolgt nach folgenden Kriterien:

Analyseklassen: Klassen, die vornehmlich aus dem Anwendungsgebiet stammen und deren Schwerpunkt auf dem Interface liegt

Slide 42

Designklassen: Klassen, welche nicht direkt mit dem Anwendungsgebiet in Verbindung stehen oder neue Features aufweisen

Eine eindeutigte Trennung ist natürlich nicht immer sinnvoll, u.a. auch deshalb weil der Entwurfsprozeß selbst nicht streng nach Analyse und Design getrennt ist.

Standardaktivitäten

In der BON werden neben den Entwicklungsphasen auch Standardaktivitäten beschrieben, wie sie in fast allen Phasen eines Projekts anfallen. Die wichtigsten sind:

Slide 43

- Identifikation von Klassen
- Klassifikation
- Gruppierung in Clustern
- Definition von Features