Understanding Advanced UML Concepts

Gerd Wagner
http://GerdWagner.rezearch.info

This Tutorial is …

- about conceptual modeling issues.
- addressing the domain analysis view point.
- restricted to state modeling (class models).

This Tutorial is **not** …

- about design or implementation modeling.
- discussing platform technologies.
- discussing behavior modeling (interaction diagrams and activity graphs).

General Overview

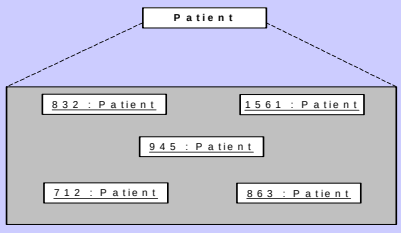
- Part I: Warm Up
- Part II: Association Classes
- Part III: Customizing UML Standard Model Elements
- Part IV: Powertypes
- Part V: Aggregation/Composition
- Part VI: Expressing Integrity Constraints with OCL

Overview Part I: Warm Up

- Objects and Classes
- Links and Associations
- Role Names
- Generalization
- (Integrity) Constraints

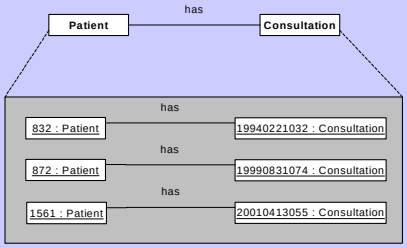
Classifying Objects

- Objects of the same type can be classified by means of an object class which is visualized as a rectangle labelled with the name of this class.

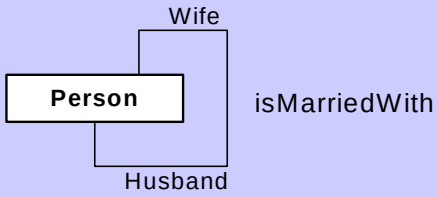


The Association **has** between **Patient** and **Consultation**

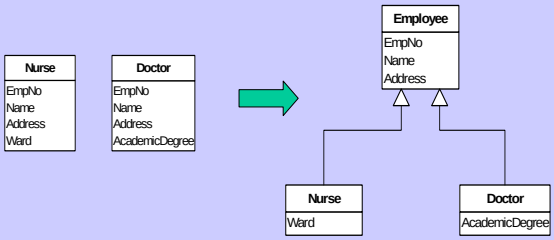
"An instance of an Association is a Link, which is a tuple of Instances drawn from the corresponding Classifiers."



Associations with Role Names



Generalization



Integrity **Constraints**

- In order to exclude system states that are not consistent with the intended information model, various types of integrity constraints can be expressed in a class model.
- Important attribute constraints are:
  - **mandatory** (default) versus **optional**, and
  - **single-valued** (default) versus **multi-valued**.
- Multiplicity constraints define for an association how many objects of one participating class can be associated with how many objects of the other one.

Integrity **Constraints** (cont.)

- An association may be
  - functional (many-to-one),
  - inverse functional (one-to-many),
  - total
  - inverse total
- Many-to-many associations are neither functional nor inverse functional.
- Uniqueness constraints for defining keys are not well supported by UML 1.

Part II

Association Classes

Consider the association **`hasAppointmentWith`**:

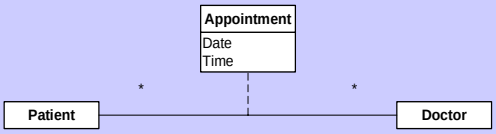
Problem: How to include the **date** and **time** of an appointment?

Consider the association **`hasAppointmentWith`**:

Problem: How to include the **date** and **time** of an appointment?

Is this a solution?

The Association Class `Appointment`



Question: Can a patient have more than one appointment with the same doctor?

No!

PatientNo	*DoctorNo*	*Date*	*Time*
0832	17	21-Dec-01	15:10
0832	24	19-Dec-01	16:45
0269	24	03-Jan-02	8:30
0588	15	20-Dec-01	9:15
0832	**17**	28-Dec-01	9:30
0699	13	28-Dec-01	9:30
…	…	…	…

Turning an Association into an Object Class

When we "objectify" an association (class), we can have more than one link between the same objects.

Note that such a class still represents a relationship type, and not an entity type!

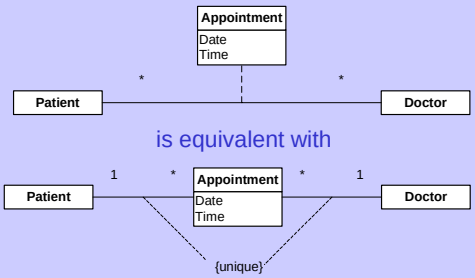
Appointments as an Object Class

DoctorNo	*Date*	*Time*	*PatientNo*
17	**21-Dec-01**	**15:10**	**0832**
24	19-Dec-01	16:45	0832
24	03-Jan-02	8:30	0269
15	20-Dec-01	9:15	0588
17	**28-Dec-01**	**9:30**	**0832**
13	28-Dec-01	9:30	0699
…	…	…	…

An association class

- … is an association that is also a class. It not only connects a set of classes but also defines a set of features that belong to the relationship itself and not any of the classes.

Association Class versus Object Class



Questions

Are the following concepts association classes or object classes?

- a ***job***
- a ***book reservation*** (in a library)
- a ***purchase*** (of a product individual)

Make a justifiable choice between the two possibilities, and draw a corresponding class diagram, including all natural role names.

Part III

Customizing UML Model Elements

Tags, Constraints and Stereotypes

- are used in the definition of UML itself for defining standard model elements that are not considered complex enough to be defined directly as UML metaclasses
- A coherent set of Tags, Constraints and Stereotypes, defined for specific purposes, constitutes a UML profile.

Tag Definitions

- allow to specify user-defined meta-attributes (tagged values) for a model element
- should be defined in conjunction with a stereotype since that allows them to be used in a more disciplined manner
- Example of a predefined tag: derived (refers to ModelElement)

Examples of Tag Definitions

Tag	*Base Class or Stereotype*	*Type*	*Multi-plicity*
aliasNames	ModelElement	UML::Datatypes::String	*
derived	ModelElement	UML::Datatypes::Boolean	1

Constraints

- allow to specify semantics/usage for a model element
- may be expressed by means of (predefined or user-defined) keywords, or in a
  - designated constraint language (such as OCL)
  - a programming language
  - mathematical notation, or
  - natural language
- can be enforced by a tool, if it "understands" the syntax and semantics of the constraint language.
- Example of a predefined constraint keyword: disjoint (refers to Generalization)

Stereotypes (1)

- provide a way of branding model elements
- allow adding user-defined categories of UML model elements by referring to a base class, which is a class in the UML metamodel such as Class, Association, etc.
- may specify additional constraints and tag definitions

Stereotypes (2)

- may be used to indicate a difference in meaning or usage
- may be defined as a subclass of one or more existing stereotypes inheriting their constraints and tag definitions
- Example of a predefined stereotype: «type» (refers to Class)

Example: UML Profile for Business Modeling

Stereotype	*Base Class*	*Parent*	*Constraints*
Worker «worker»	Class	NA	None
CaseWorker «caseWorker»	Class	Worker	None

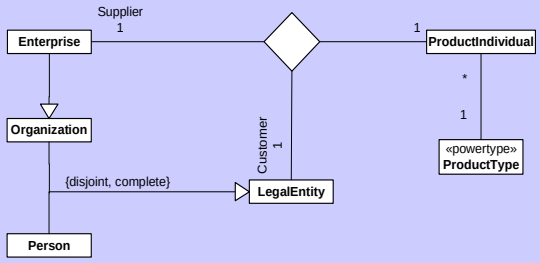
Part IV

Powertypes

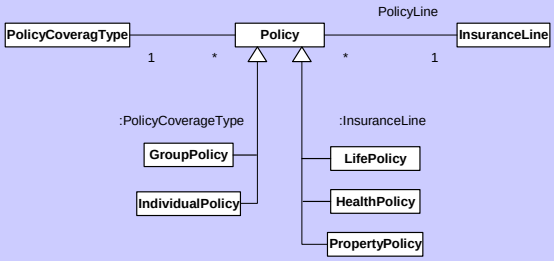
Powertypes

- are special classes, designated with the stereotype «powertype»
- In UML 1: "a user-defined metaelement whose instances are classes in the model."
- In UML 2: "***a class whose instances are subclasses of another class***".

Example: ProductType



Subclass Partitions and Powertypes



Nested Powertype Constructions

- The instances of a powertype of order 1 are subclasses of a base type (not a powertype).
  Examples: EngineType, CarType
- The instances of a powertype of order *2* are powertypes of order *1*.
  Example: ProductType
- The instances of a powertype of order *n* are powertypes of order *n-1*.

Questions

- Make a diagram for AccountType, Account, CheckingAccount, SavingsAccount.
- Find another example of a powertype.

Part V

Aggregation and Composition

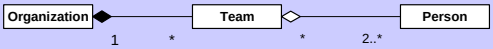
Aggregation

- An aggregation is a part-whole relationship, that is, a special binary association where the instances on one side are aggregates (or wholes) and the instances on the other side are their parts.
- An aggregation relationship may be called isPartOf or consistsOf.

Composition

- If the parts in the part-whole relationship are non-shareable, it is called a composition (visualized as a black diamond).
- *"A shareable aggregation denotes weak ownership; that is, the part may be included in several aggregates and its owner may also change over time."*

Ex: An organization is composed of teams that consist of persons.



Notice that Organization is a ***mandatory*** aggregate for Team, whereas Team is an optional aggregate for Person.

Mandatory Aggregates and Mandatory Parts

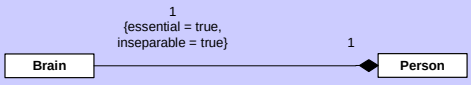
Engine is a mandatory part of Car, and Car is an optional aggregate for Engine.

Heart is a mandatory part of HumanBody, and HumanBody is a mandatory aggregate for Heart.

Lieftime Dependency

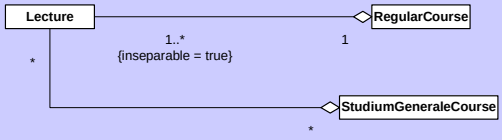
- UML 1: "the composite object is responsible for the creation and destruction of its parts" "if the composite is destroyed, it must destroy all its parts"
- However: lifetime dependency is a characteristic of part-whole relationships with inseparable parts.
- We can define our own tags for expressing this.

Inseparable and Essential Parts



Brain is an inseparable and essential part of Person.

Inseparable Parts may be Shareable



Although Lecture is an inseparable part of RegularCourse, it is shareable.

An aggregation relationship is

- anti-symmetric: when an object o1 is part of an aggregate object o2, then o2 cannot be part of o1
- transitive: when an object o1 is part of an aggregate object o2, and o2 is part of another aggregate object o3, then o1 is also part of o3.

Part VI

Expressing Integrity Constraints with OCL

Expressing Constraints with the Object Constraint Language (OCL)

- There are many types of constraints that cannot be expressed visually but have to be included as a piece of text, or a symbolic expression, in a class diagram.
- OCL can be used to write unambiguous constraints that can't be misinterpreted.
- OCL expressions do not have any side effects, which means that evaluating an OCL expression will not affect the object to which the expression is applied.

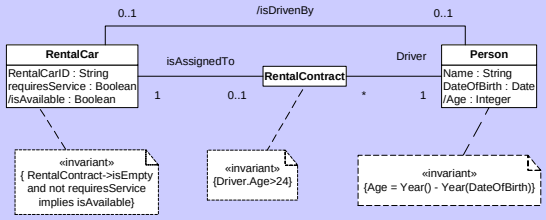
OCL (continued)

- All expressions in OCL have a type and evaluate to a value from that type when applied to a specific context.
- In OCL, a constraint is called an invariant.
- In addition to invariants, OCL allows to define preconditions and postconditions for operations.

Business Rules

- A rental car is available, if it is not assigned to a rental contract, and it does not require service.
- A driver of a rental car must be at least 25 years old.
- The age of a person is computed as the difference between the current year and the person's birth year.

Expressing Business Rules with OCL Invariants



Bibliography

- **James Martin and James Odell*. Object-Oriented Methods: A Foundation (UML Edition),* Prentice-Hall, 1998.** This book presents the fundamental concepts underlying the object-oriented approach in a clear, concise manner using the Unified Modeling Language (UML). It also introduces more advanced structures—including constraints, business rules, meta-modeling, power types, and dynamic and multiple classification.