Analysis Models, Analysis Class Diagram II

PB007 Software Engineering I

Bruno Rossi

04. 11. 2015

Analysis Models

The **Analysis Model** is a conceptual model that captures the abstraction of a problem in modeling the business domain.

The use of design patterns can increase the flexibility and reusability of the model.

In modelling, there are often a set of recurring problems. For example, the model for the organizational structure of a company can be used also with other similar types of hierarchies. Therefore, there are already some prepared patterns that can be used when modelling a software system. We just see here some of them.

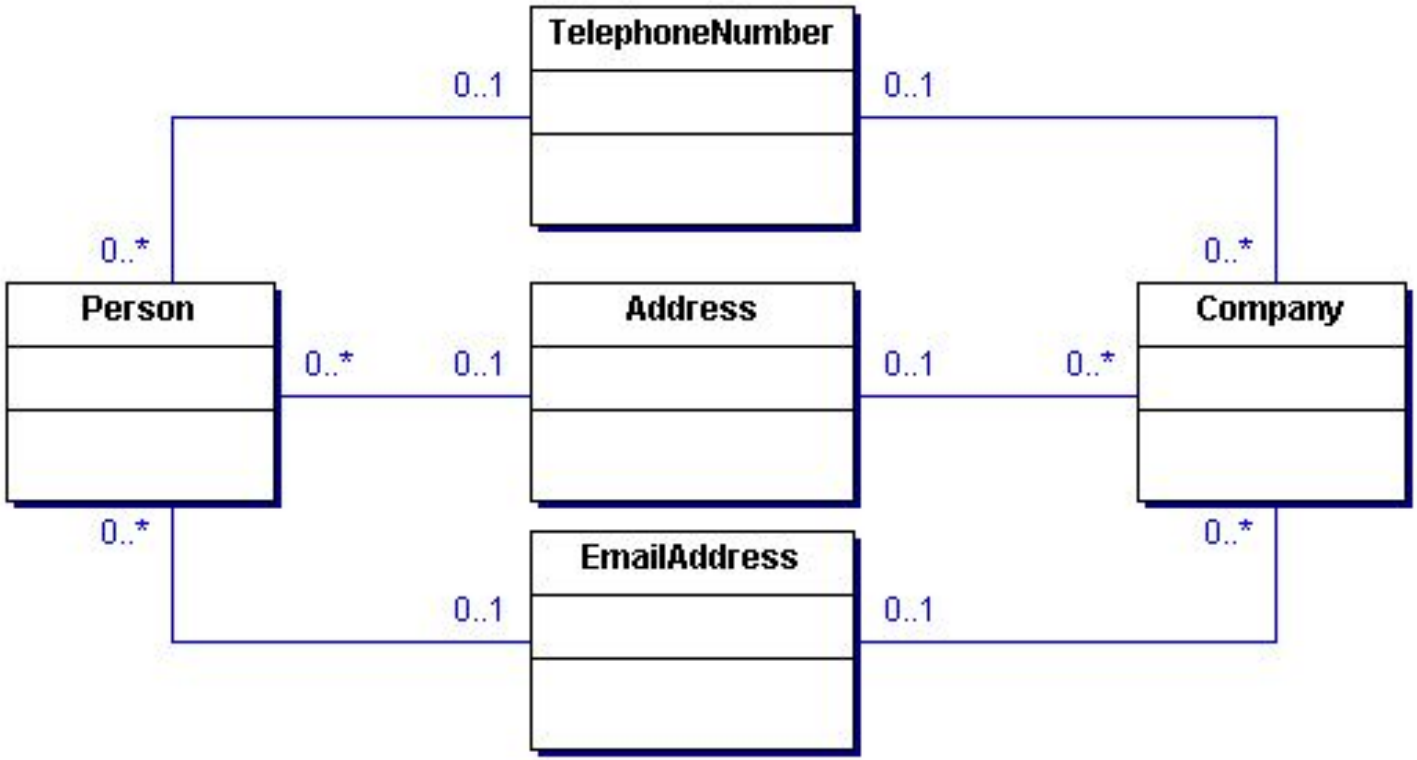
Analysis Patterns - Martin Fowler

Martin Fowler: Analysis Patterns - Reusable Object Models

- http://books.google.cz/books?id=4V8pZmpwmBYC&pg
- available at the library FI, MZK
- different patterns according to the different domains
  Examples: *Accountability, Observations and Measurements, Refering to Objects, Inventory and Accounting, Planning, Trading*
- Each chapter contains several patterns that are then shown in use in some simple example and then also into more complex ones.
- Total of 65 analysis patterns and 21 auxiliary patterns

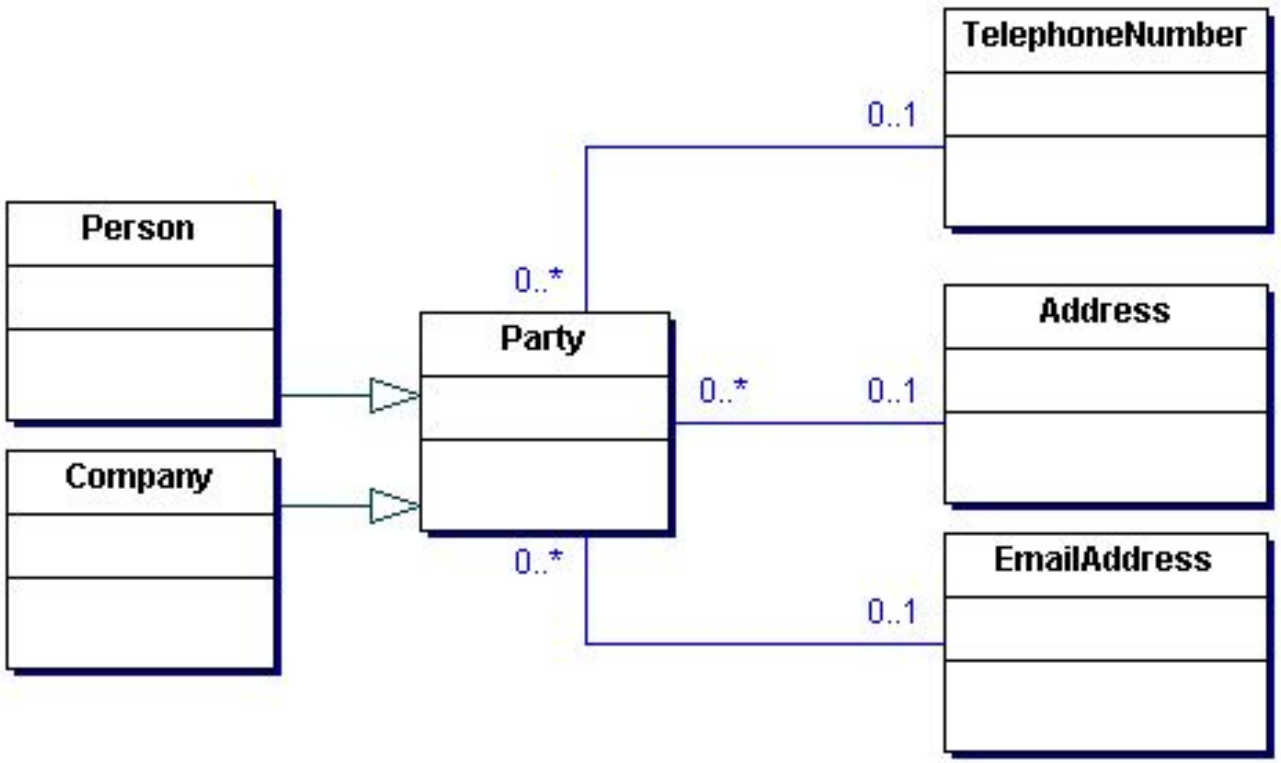
Analysis Patterns - Accountability - Party

Telephone directory:

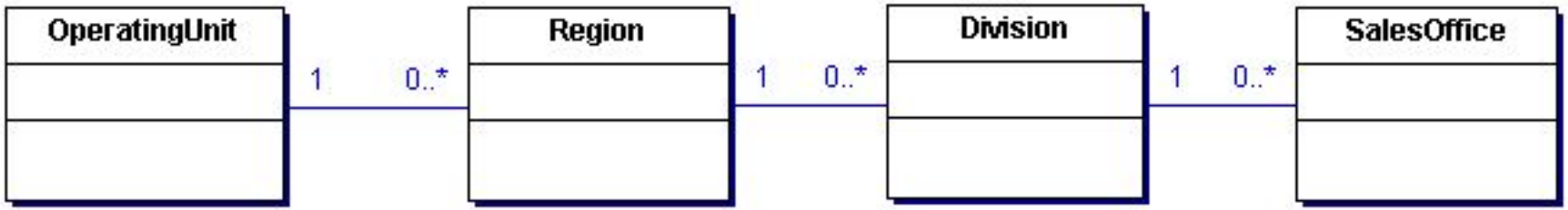


Analytické vzory - Accountability - Party II

Many of the roles that people play can also be derived from organizational units.
Party/participant is an uniform name for these roles

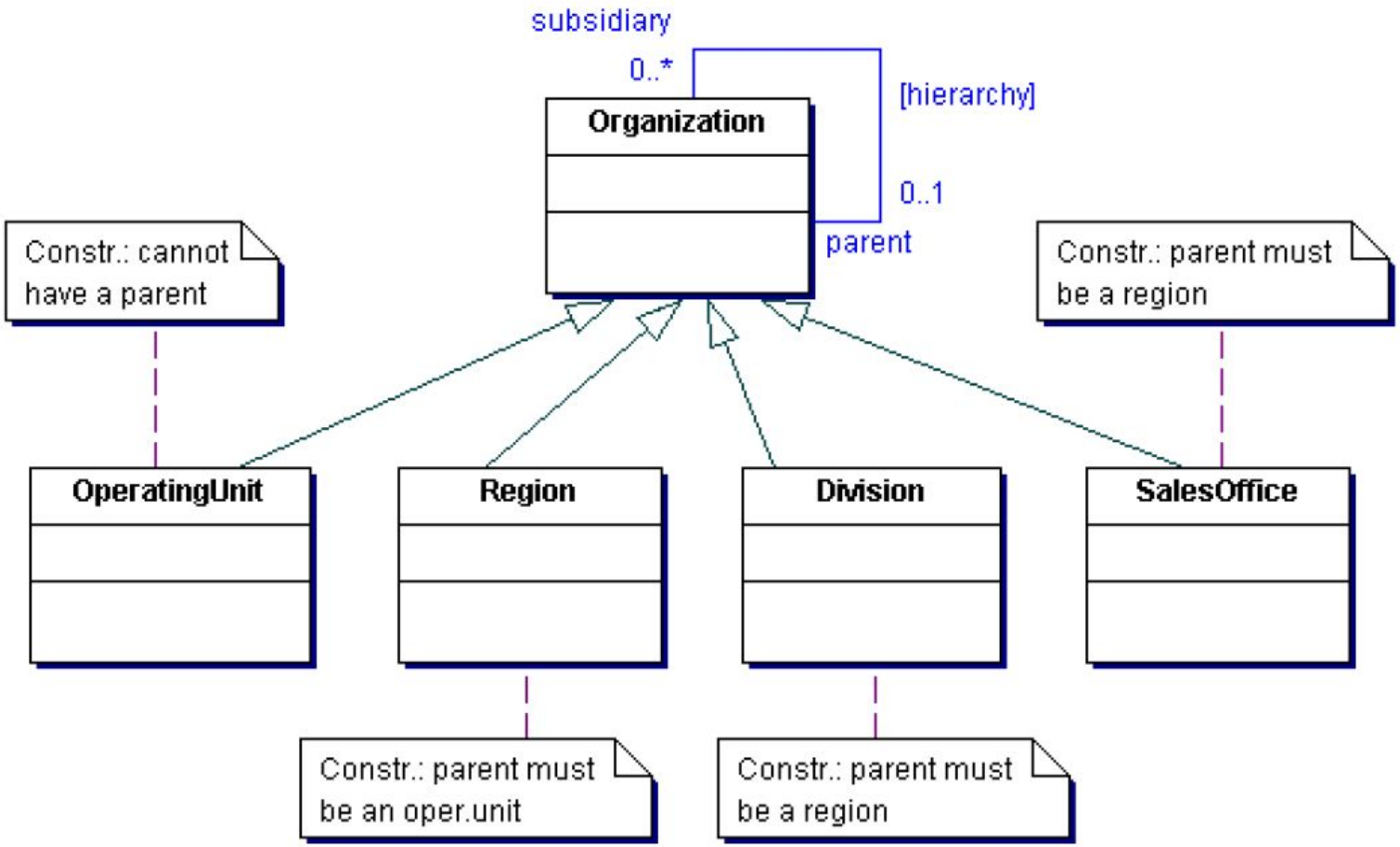


Analysis Models - Organization Hierarchies



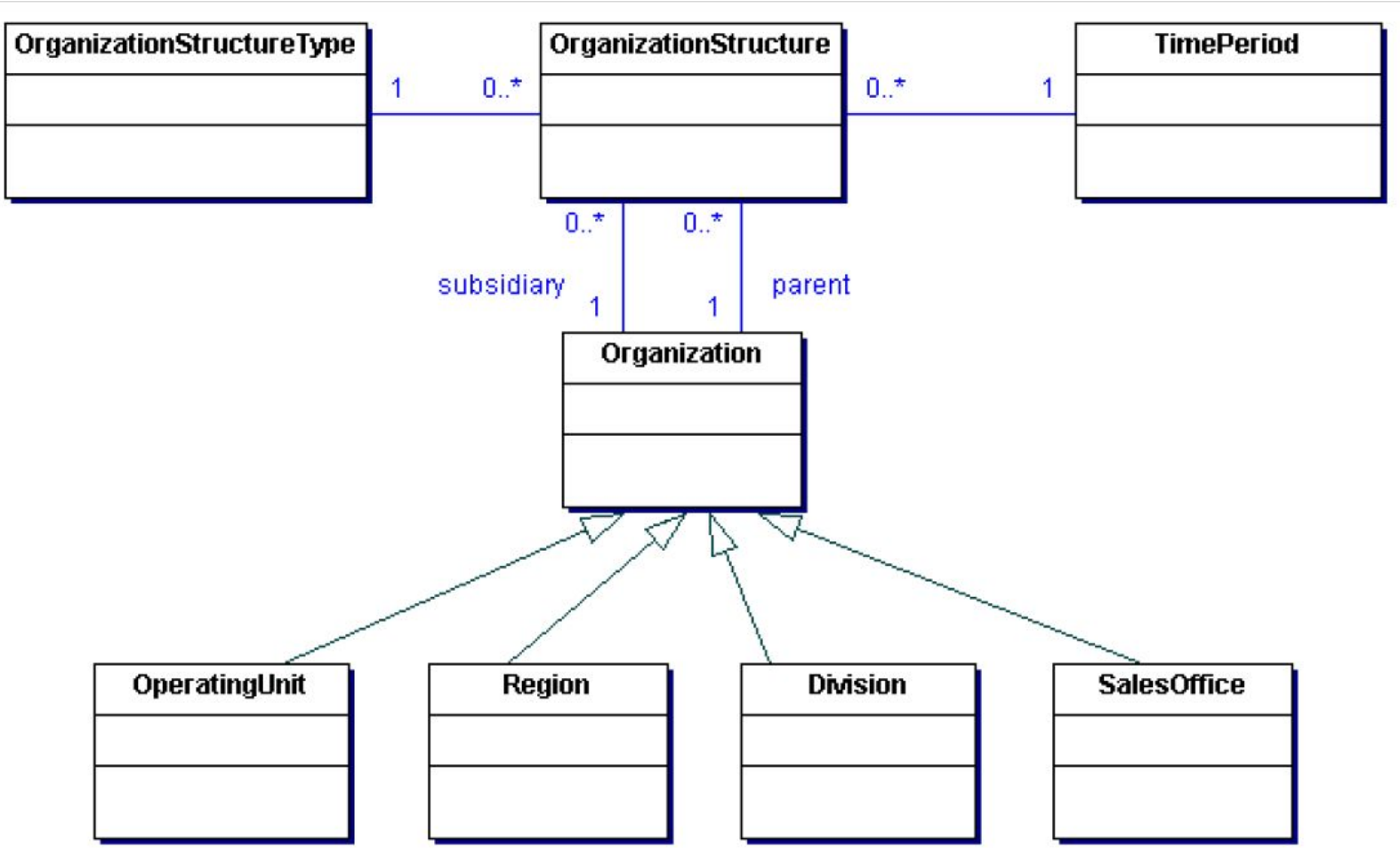
- inflexible - for addition/change of organizational units, it is necessary to change big part of the model
- small reusability - different companies can vary in organizational structure

Analysis Models - Organization Hierarchies II



- And what if the company has a different type of organizational structures (e.g. Sales and Production) or when they change over time?

Analysis Models - Organization Hierarchies III



Analysis Models - other resources

- http://martinfowler.com/articles.html
- Gamma, Helm, Johnson, Vlissides ("Gang of Four" - GoF): *Design Patterns: Elements of Reusable Object-Oriented Software*, 1991-1994
- Buschmann, Meunier, Rohnert, Sommerland, Stal: *Pattern – Oriented Software Architecture: A System of Patterns*
- Ľubor Šešera, Aleš Mičovský, Juraj Červeň: *Datové modelování v příkladech*

Relations between classes

The basic relationships include:

- Generalization
- Association
- Dependency

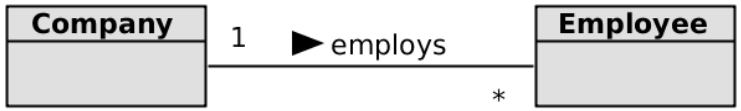
Association

The **Association** is a relationship between classes.

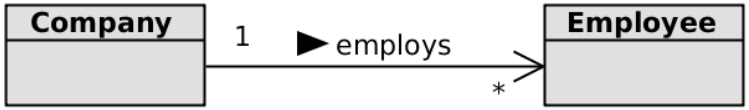
- represents long-term relationships
- this type of reference is most often reflected in the presence of an attribute of the type of the other class
- relations 1:1 correspond to one reference attribute
- relations 1:N correspond generally to an array or collection
- the direction of the association (navigability) indicates the class that will contain the attribute

Association - Navigability

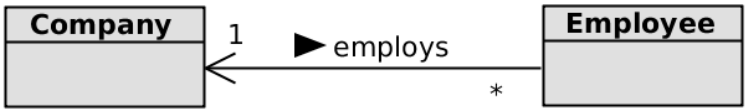
The company has a relationship with more employees



The company contains a reference to a collection of employees, employed by. Employees do not have relation with company.



The company does not have a direct link to the staff. The employee has an attribute of type company as a reference.

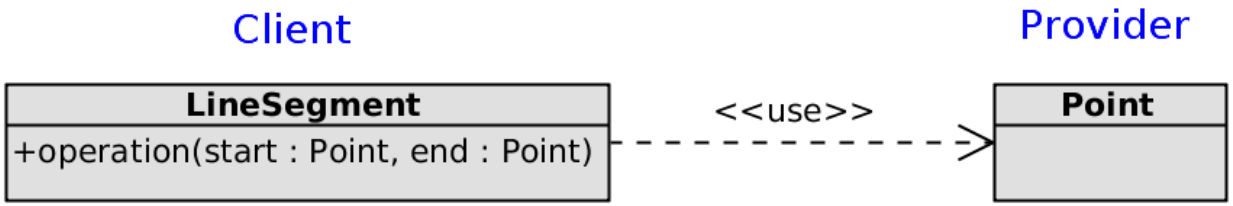


Dependency

A **dependency** is a relationship between two classes (client and provider), where a change in provider may force a change in the client. In other words, the client somehow depends on the provider.

The importance of this dependence may be specified by different stereotypes.

The most often used stereotype is `use`. It indicates that some of the operations of the client object use the provider as an input argument or output.



Tasks

- Finalize the analysis class diagram
  - The diagram should contain the **analysis classes, attributes, operations** and the different relationships (**inheritance, association**, **dependencies**)
  - The associations should specify **names, multiplicity** and **navigability**
  - Add manager classes that will take care of the lifecycle of other classes (maintaining a list of objects, creation, search, ...). Move the appropriate responsibilities/operation
- Update the Use Case diagram. For each class there should be a use case that creates instances, uses, and deletes instances of the class
- Think about the interactions between the different class instances. Based on your class diagram, can you represent all the use cases?
- OPTIONAL: If you have time, think about using analytical models patterns (eg. Party or Organization Hierarchies);
- Upload the **pdf report** on the folder (**Week 06**). **Deadline:** Mon, 09.11. 23:59 (Groups 2,3)

Configuration of PDF Reports