Information Systems Concepts

System Design

Roman Kontchakov

Birkbeck, University of London

Based on Chapter 12 of Bennett, McRobb and Farmer:

Object Oriented Systems Analysis and Design Using UML, (4th Edition), McGraw Hill, 2010

Outline

- System Design
  - Chapter 12 (pp. 347–360)

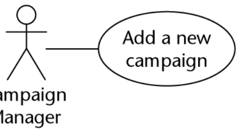
How is Design Different from Analysis?

- Design states `how the system will be constructed **without** actually building it'

  (Rumbaugh, 1997)

- Analysis identifies `what' the system must do
  - The analyst seeks to understand the organization, its requirements and its objectives
- Design specifies `how' it will do it
  - The designer seeks to specify a system that will fit the organization, provide its requirements effectively and assist it to meet its objectives

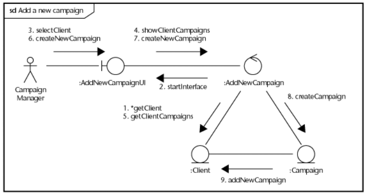
Requirements

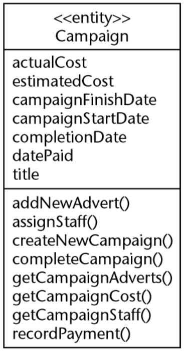


2. To record the details of each campaign for each client. This will include the title of the campaign, planned start and finish dates, estimated costs, budgets, actual costs and dates, and the current state of completion.

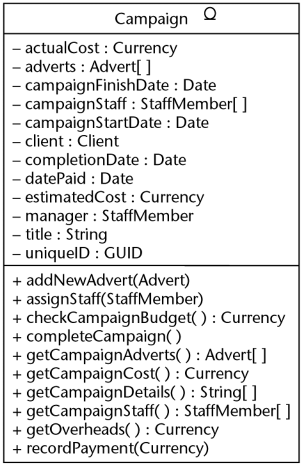
analysis identifies the fact that the `Campaign` class has a `title` attribute

Analysis

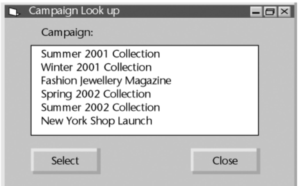




Design


Campaign
– actualCost : Currency
– adverts : Advert[ ]
– campaignFinishDate : Date
– campaignStaff : StaffMember[ ]
– campaignStartDate : Date
– client : Client
– completionDate : Date
– datePaid : Date
– estimatedCost : Currency
– manager : StaffMember
– title : String
– uniqueID : GUID
+ addNewAdvert(Advert)
+ assignStaff(StaffMember)
+ checkCampaignBudget( ) : Currency
+ completeCampaign( )
+ getCampaignAdverts( ) : Advert[ ]
+ getCampaignCost( ) : Currency
+ getCampaignDetails( ) : String[ ]
+ getCampaignStaff( ) : StaffMember[ ]
+ getOverheads( ) : Currency
+ recordPayment(Currency)

```
CREATE TABLE Campaigns
  (VARCHAR(30) uniqueID PRIMARY KEY NOT NULL,
  FLOAT actualCost,
  DATE campaignFinishDate,
  DATE campaignStartDate,
  VARCHAR(30) clientID NOT NULL,
  DATE completionDate,
  DATE datePaid,
  FLOAT estimatedCost,
  VARCHAR(30) managerID,
  VARCHAR(50) title);
CREATE INDEX campaign_idx ON Campaigns
(clientID, managerID, title);
```


Campaign Look up

Campaign:

Summer 2001 Collection
Winter 2001 Collection
Fashion Jewellery Magazine
Spring 2002 Collection
Summer 2002 Collection
New York Shop Launch

design determines how this will be entered into the system, displayed on screen and stored in a database, together with all the other attributes of `Campaign` and other classes

When Does Analysis Stop and Design Start?

- In a *waterfall* life cycle there is a clear transition between the two activities
- In an *iterative* life cycle the analysis of a particular part of the system will precede its design, but analysis and design may be happening in parallel
- It is important to distinguish the two activities and the associated mindset
- We need to know `what' before we decide `how'

Qualities of Analysis

- **Correct scope**: everything in the system is required
- **Completeness**: everything required is in the system and everything is documented in the models
- **Correct content**: accurate description of requirements
- **Consistency**: each element is consistently referred to by the same name

Qualities of Design (1)

- **Functional**: system will perform the functions that it is required to
- **Efficient**: the system performs those functions efficiently in terms of time and resources
- **Usable**: provides users with a satisfying experience (affordance, objects of UI suggest their function, reducing error rates)
- **Reliable**: not prone to hardware or software failure, will deliver the functionality when the users want it
- **Buildable**: design is not too complex for the developers to be able to implement it

choice of the programming language

Qualities of Design (2)

- **Manageable**: easy to estimate work involved and to check of progress
- **Economical**: running costs of system will not be unnecessarily high
- **Secure**: protected against errors, attacks and loss of valuable data
- **Flexible**: capable of being adapted to different configurations, to run in different countries or to be moved to a different platform
- **General**: general-purpose and portable (mainly applies to utility programs)

Qualities of Design (3)

- **Maintainable**: design makes it possible for the maintenance programmer (fixing bugs, migrating to new hardware, etc.) to understand the designer's intention
- **Reusable**: elements of the system can be reused in other systems

Trade-offs in Design (1)

- Design objectives may conflict with constraints imposed by requirements
  - The requirement that the system can be used in different countries by speakers of different languages will mean that designers have to agree a list of all prompts, labels and messages and refer to these by some system of naming or numbering (this increases flexibility and maintainability but increases the cost of design)

Trade-offs in Design (2)

- Design to meet all the qualities may produce conflicts
  - Functionality, reliability and security are likely to conflict with economy
  - Level of reliability, for example, is constrained by the budget available for the development of the system
- Trade-offs have to be applied to resolve these
- It is helpful if guidelines are prepared for prioritizing design objectives
- If design choice is unclear users should be consulted

Design in TLC

- Making a clear transition from analysis to design has advantages:
  - project management—is there the right balance of activities?
  - staff skills—analysis and design may be carried out by different staff
  - client decisions—the client may want a specification of the ‘what’ before approving spending on design
  - choice of development environment—may be delayed until the analysis is complete

Design in the Iterative Life Cycle

- Advantages of the iterative life cycle include
  - risk mitigation—making it possible to identify risks earlier and to take action
  - change management—changes to requirements are expected and properly managed
  - team learning–all the team can be involved from the start of the project
  - improved quality—testing begins early and is not done as a ‘big bang’ with no time

Seamlessness in OO A&D

- The same model—the class model—
  is used *through* the life of the project
- During design, additional detail is added to the analysis classes, and extra classes are added to provide the supporting functionality for the user interface and data management
- Other diagrams are also elaborated
  in design activities

Logical and Physical Design

- *Logical* design is independent of the implementation language and platform
  - Some design of the user interface classes can be done without knowing whether it is to be implemented in Java, C++ or some other language—types of fields, position in windows
- *Physical* design is based on the actual implementation platform and the language that will be used
  - Some design can only be done when the language has been decided upon—the actual classes for the types of fields, the layout managers available to handle window layout

Logical and Physical Design

- It is not necessary to separate these into two separate activities
- It may be useful if the software is to be implemented on different platforms
- Then it will be an advantage to have a platform-independent design that can be tailored to each platform
- Model-Driven Architecture (promoted by OMG)
  - generate platform-specific models (PSMs) from platform-independent models (PIMs)

System Design v Detailed Design

- *System design* deals with
  the high-level architecture of the system
  - structure of sub-systems
  - distribution of sub-systems on processors
  - communication between sub-systems
  - standards for screens, reports, help etc.
  - job design for the people who will use the system

System Design v Detailed Design

- Object-oriented *detailed design* adds detail to the analysis model
  - types of attributes
  - operation signatures
  - assigning responsibilities as operations
  - additional classes to handle user interface
  - additional classes to handle data management
  - design of reusable components
  - assigning classes to packages

Coupling and Cohesion: Criteria for Good Design

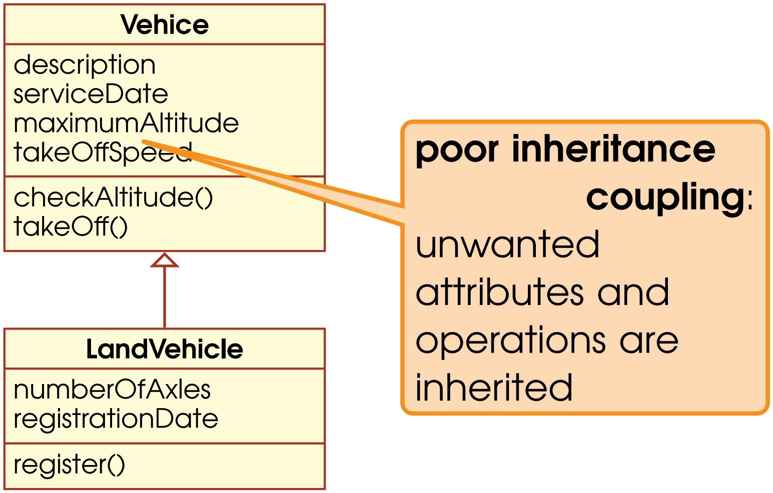
- **Coupling** describes the degree of **interconnectedness** between design components
    - reflected by the number of links an object has and by the degree of interaction the object has with other objects
- **Cohesion** is a measure of the degree to which an element **contributes** to a **single purpose**
- Coupling and Cohesion are not mutually exclusive but actually support each other

Interaction Coupling

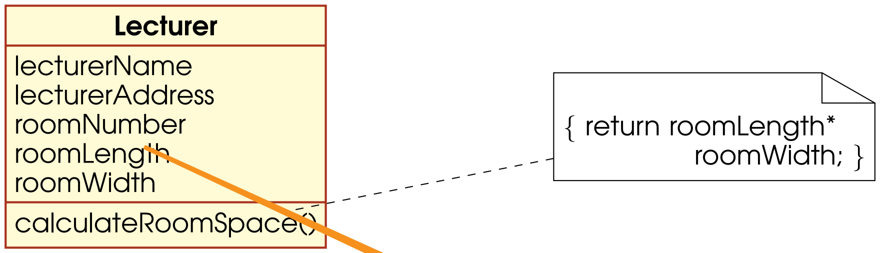
- A measure of the **number of message types** an object sends to other objects and the number of parameters passed with these message types
- Should be kept to a minimum to reduce the possibility of changes rippling through the interfaces and to make reuse easier.

Inheritance Coupling

- A degree to which a subclass actually needs the features it inherits from its base class



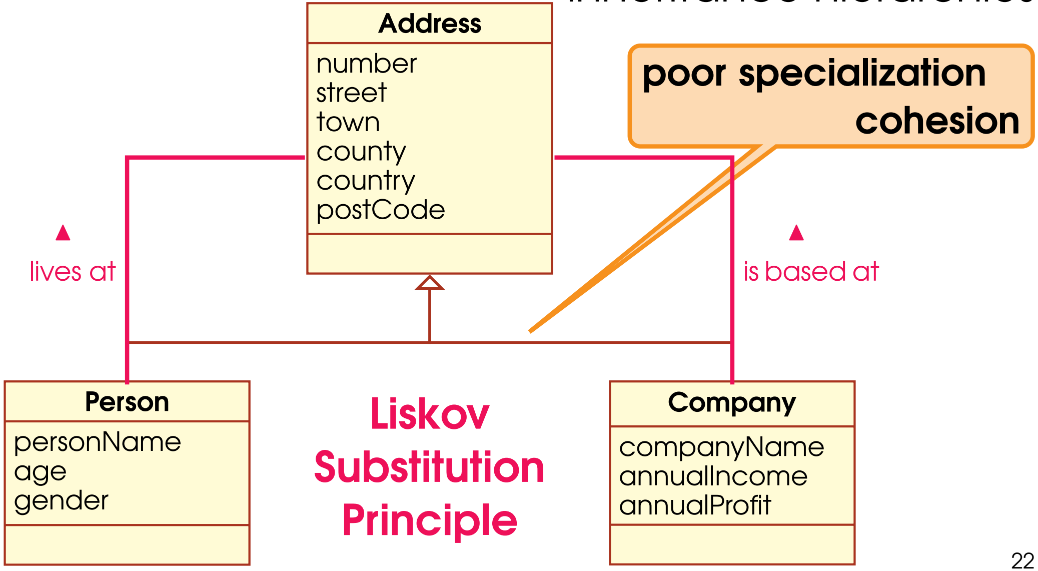
Operation Cohesion



good operation cohesion
but **poor class cohesion**

Specialization Cohesion

- addresses the semantic cohesion of inheritance hierarchies



Take Home Messages

- The difference between analysis and design
- The difference between logical and physical design
- The difference between system and detailed design
- The characteristics of a good design
- The need to make trade-offs in design