Moving into Design

Based on Chapter 12

Bennett, McRobb and Farmer

Object Oriented Systems Analysis and Design Using UML

4th Edition, McGraw Hill, 2010

In This Lecture You Will Learn:

- 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

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
- Design specifies ‘how’ it will do it

How is Design Different from Analysis?

- The analyst seeks to understand the organization, its requirements and its objectives
- The designer seeks to specify a system that will fit the organization, provide its requirements effectively and assist it to meet its objectives

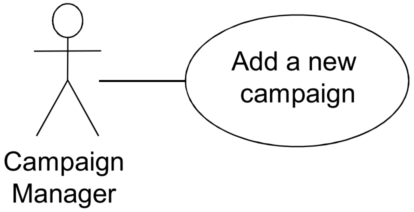
How is Design Different from Analysis?

- As an example, in the Agate case study:
  - analysis identifies the fact that the `Campaign` class has a `title` attribute
  - 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?

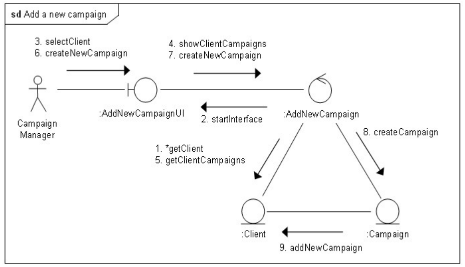
- In a waterfall life cycle there is a clear transition between the two activities
- In an iterative life cycle the analysis of a particular part of the system will precede its design, but analysis and design may be happening in parallel
- It is important to distinguish the two activities and the associated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

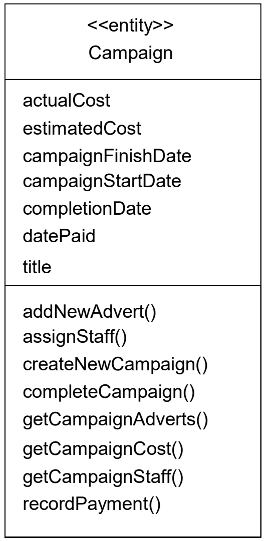
Requirements



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



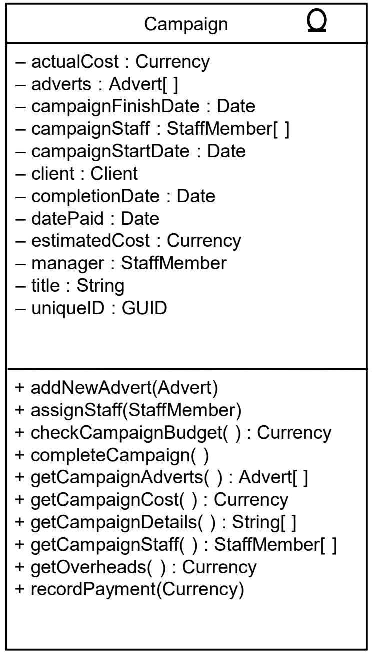


<<entity>>
Campaign

actualCost
estimatedCost
campaignFinishDate
campaignStartDate
completionDate
datePaid
title

addNewAdvert()
assignStaff()
createNewCampaign()
completeCampaign()
getCampaignAdverts()
getCampaignCost()
getCampaignStaff()
recordPayment()

Design

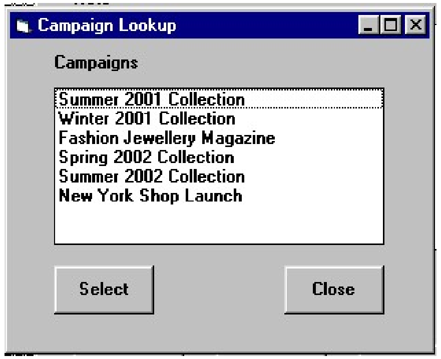


Campaign

– 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);
```



Campaigns

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

Traditional Design

- 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

- 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

- In structured analysis and design a distinction has been made between logical and physical design
- Logical design is independent of the implementation language and platform
- Physical design is based on the actual implementation platform and the language that will be used

Logical and Physical Design Example

- 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
- 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

- Note the MDA Initiative
  - Generate platform-specific models (PSMs) from platform-independent models (PIMs)

This is discussed in more detail in Chapter 13

System Design and 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 and Detailed Design

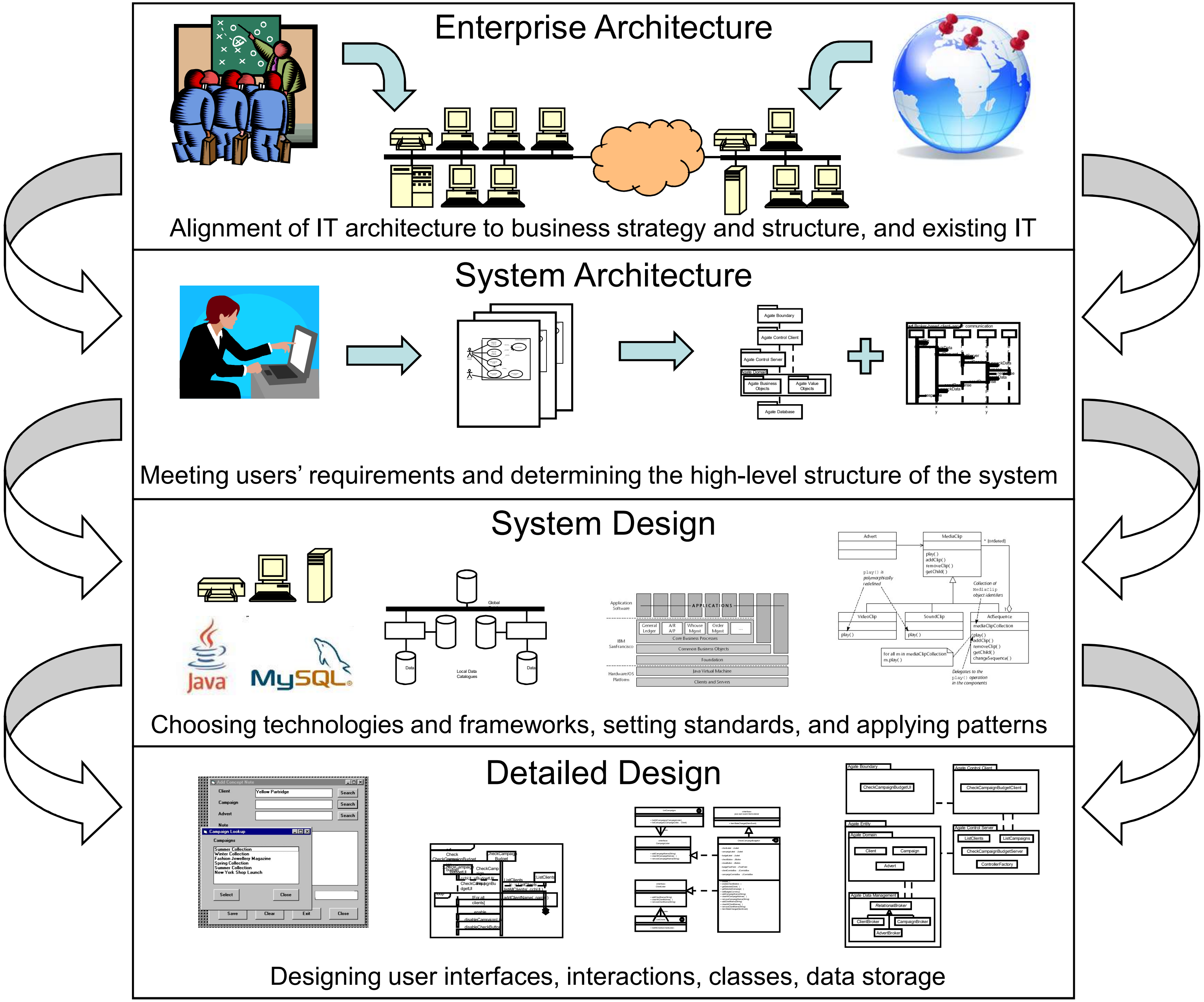
- Traditional detailed design consists of four main activities
  - designing inputs
  - designing outputs
  - designing processes
  - designing files and database structures

System Design and Detailed Design

- Traditional detailed design tried to maximise cohesion
  - elements of module of code all contribute to the achievement of a single function
- Traditional detailed design tried to minimise coupling
  - unnecessary linkages between modules that made them difficult to maintain or use in isolation from other modules

System Design and 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



Enterprise Architecture

Alignment of IT architecture to business strategy and structure, and existing IT

System Architecture

Meeting users’ requirements and determining the high-level structure of the system

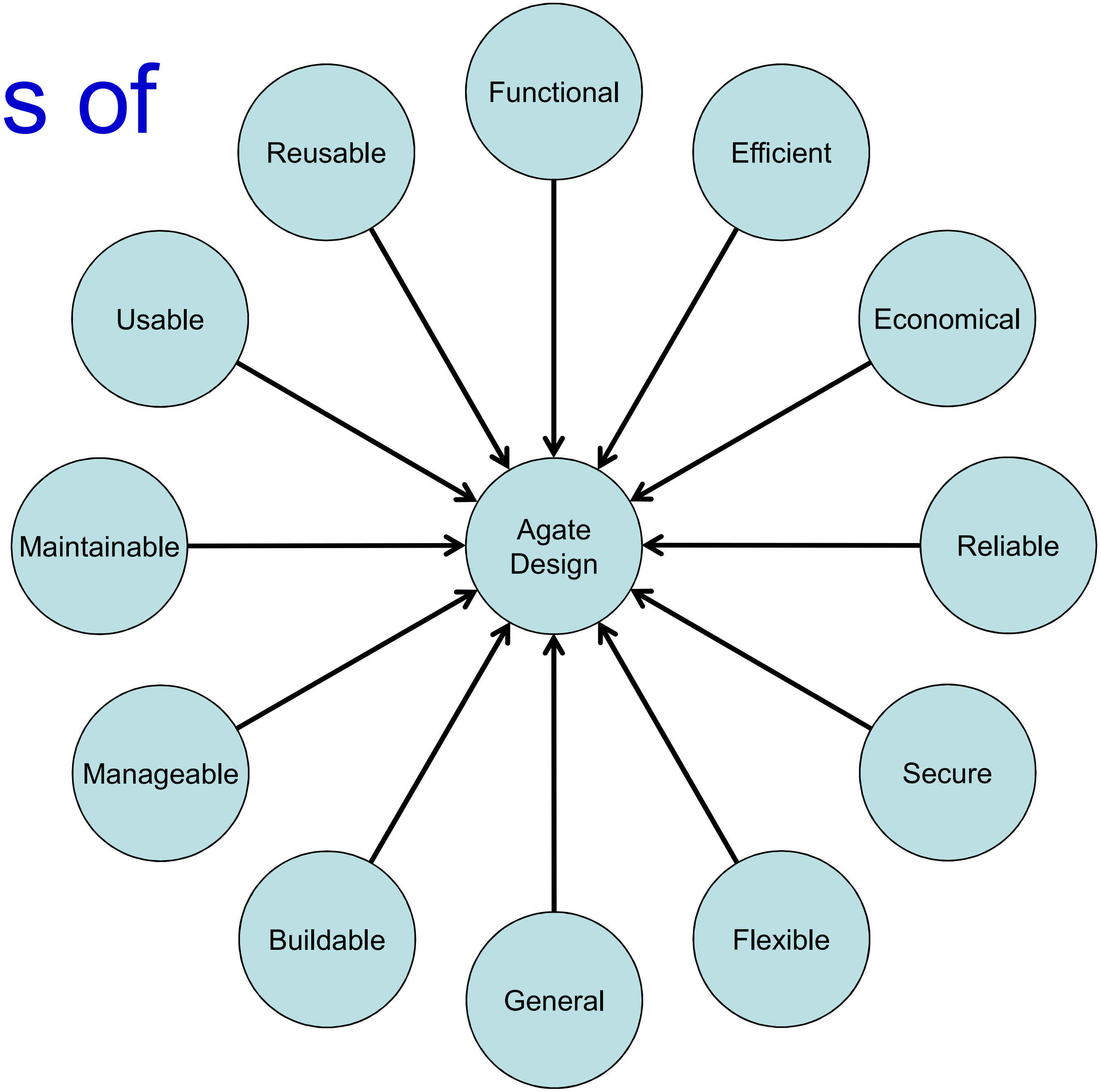
System Design

Choosing technologies and frameworks, setting standards, and applying patterns

Detailed Design

Designing user interfaces, interactions, classes, data storage

Qualities of Design



Qualities of Design

- Functional—system will perform the functions that it is required to
- Efficient—the system performs those functions efficiently in terms of time and resources
- Economical—running costs of system will not be unnecessarily high
- Reliable—not prone to hardware or software failure, will deliver the functionality when the users want it

Qualities of Design

- Secure—protected against errors, attacks and loss of valuable data
- Flexible—capable of being adapted to new uses, to run in different countries or to be moved to a different platform
- General—general-purpose and portable (mainly applies to utility programs)
- Buildable—Design is not too complex for the developers to be able to implement it

Qualities of Design

- Manageable—easy to estimate work involved and to check of progress
- Maintainable—design makes it possible for the maintenance programmer to understand the designer's intention
- Usable—provides users with a satisfying experience (not a source of dissatisfaction)
- Reusable—elements of the system can be reused in other systems

Prioritizing Design Trade-offs

- Designer is often faced with design objectives that are mutually incompatible.
- It is helpful if guidelines are prepared for prioritizing design objectives.
- If design choice is unclear users should be consulted.

Trade-offs in Design

- Design to meet all these qualities may produce conflicts
- Trade-offs have to be applied to resolve these
- 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 in Design

- 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

Measurable Objectives in Design

- In Chapter 6, non-functional requirements were described
- How can we tell whether these have been achieved?
- Measurable objectives set clear targets for designers
- Objectives should be quantified so that they can be tested

Measurable Objectives in Design

- To reduce invoice errors by one-third within a year
- How would you design for this?

Measurable Objectives in Design

- To reduce invoice errors by one-third within a year
- How would you design for this?
  - – sense checks on quantities
  - – comparing invoices with previous ones for the same customer
  - – better feedback to the user about the items ordered

Measurable Objectives in Design

- To process 50% more orders at peak periods
- How would you design for this?

Measurable Objectives in Design

- To process 50% more orders at peak periods
- How would you design for this?
  - design for as many fields as possible to be filled with defaults
  - design for rapid response from database
  - design system to handle larger number of simultaneous users

Summary

In this lecture you have learned about:

- 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

References

- More detail about design is provided in Chapters 13 to 18
- In particular, Chapter 14 covers Class Design

(For full bibliographic details, see Bennett, McRobb and Farmer)

References

- Rumbaugh et al (1991)
- Yourdon (1994)
- Jacobson et al. (1995)
- Meyer (1997)
- Somerville (2007)
- Pressman (2009)

(For full bibliographic details, see Bennett, McRobb and Farmer)