Transition to Design

Chapter 12

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 has been described by Rumbaugh (1997) as stating ‘How the system will be constructed without actually building it’
  - 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’

Traditional Design

- Making a clear transition from analysis to design and has the following advantages
  - **Project Management** – Being able to plan for and budget for the two stages separately.
  - **Staff Skills and Experience** – Having separate business analysts who are more familiar with business practices and designers who know the development environment well.
  - **Client Decisions** – Having a clear decision point at which the client can read the specification of requirements and agree to it before design begins.
  - **Choice of Development** – Being able to delay the decision about the development environment to take advantage of new developments.

Design in the Iterative Life Cycle

- Advantages of the iterative life cycle include
  - **Risk Mitigation –** Problems can be identified early in a project, which helps to mitigate risks.
  - **Change Management –** Treating requirements change as an expected process and putting in place procedures to handle it makes it easier to manage.
  - **Team Learning –** Team members can be involved in and learning about requirements and the solution from early in the project.
  - **Improved Quality –** Testing of deliverables begins early and quality is improved.

Seamlessness

- **Seamlessness** means that the same model (class diagram) is used and successively refined throughout 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

- Design is sometimes divided into two stage.
  - The first is implementation-independent or logical design
  - The second is implementation-dependent or 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

- 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
  - E.g. types of fields, position in windows
- Some design can only be done when the language has been decided upon
  - E.g. 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

System 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

Traditional Detailed Design

- Traditional detailed design consists of four main activities
  - Designing Inputs
  - Designing Processes
  - Designing Outputs
  - Designing Files and Database Structures

Traditional 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

Class Exercise

- Explain the difference between cohesion and coupling.

Object-oriented Detailed Design

- We elaborate user interface and application control classes, we add mechanisms to support data management.
- The class diagram is also updated with the types and visibility of attributes and operations and to show how associations are designed.

Object-oriented 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

System Design vs. Detailed Design

- **System Design** is concerned with the overall architecture of the system and the setting of standards, for example for the design of the Human Computer Interface (HCI).
- **Detailed Design** is concerned with designing individual components to fit this architecture and to conform to the standards.
- In an object-oriented system, the detailed design is mainly concerned with the design of objects.

Class Exercise

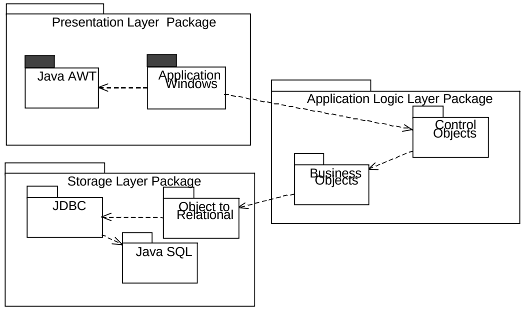
- Users at Agate require a report of unpaid campaigns. Which of the following aspects of the report represents analysis, logical design and physical design?
  - The size of the paper and the position of each field in the report.
  - The fact that the user wants a report of completed campaigns that have not yet been paid far by the client.
  - The selection of the business objects and their attributes used by the report

Class Exercise

- Which of the following sentences describing an element of the FoodCo system represents analysis, logical design and physical design?
  - The reason for stopping a run will be selected from one of the values displayed in a listbox (Java Choice) in the Record Line Stop dialogue window.
  - When a production line stops during a run, the reason for stopping will be recorded.
  - The reason for stopping a run will be entered into the system by selecting from a list of valid reasons.

Elaborating Classes in Packages

- Larmen (1998) proposes an architecture based on three layers:
  - Presentation Layer
  - Application Logic Layer
  - Storage Layer

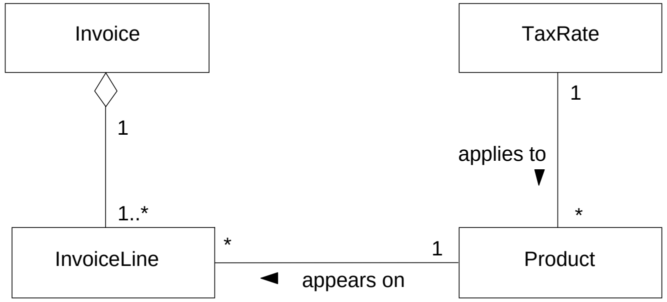


Assigning Responsibilities

- The assignment of responsibilities to classes is an issue that is related to reuse.
- Larman (1998) highlights this activity as the main task in design.
- In an object-oriented system, it is important to assign responsibility for operations to the right classes, and there is often a choice.
- In the FoodCo system, there will be a need to produce invoices for customers that include the calculation of Value Added Tax (VAT).
- The calculation of VAT could be carried out by one of a number of classes in the model.

Assigning Responsibilities Example

- Invoice – which organizes the total information for the whole sale.
- InvoiceLine – which contains the detail of each item sold and to which the tax applies.
- Product – to which different VAT rates may apply.
- TaxRate – which carries the details of the percentage that applies for each valid rate



Assigning Responsibilities Example

- If the designer makes the wrong decision, the resulting class will be less reusable and may constrain the design of other classes.
- If the responsibility for tax calculation is allocated to Invoice or InvoiceLine, then this has implications for CreditNote and CreditNoteLine, which may also need to calculate tax.
- If it is assigned to Product, then it cannot be reused in the Agate project where VAT applies to services as well as products.
- Clearly it needs to be assigned to TaxRate in order to maximize the reuse that can be made of the classes in this design.

Four Quality Criteria for Good 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

Twelve Quality Criteria for Good Design

Functional	General
Efficient	Buildable
Economical	Manageable
Reliable	Maintainable
Secure	Usable
Flexible	Reusable

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 (cont')

- **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 (cont')

- **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

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

- 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

Example of Measurable Objectives

- 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

Example of Measurable Objectives

- 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

Planning for Design

- Planning for when platform is known
- Setting standards
- Allowing time for training
- Agreeing objectives and planning tests
- Agree procedures to decide on trade-offs that significantly affect the system
- Planning time for different aspects of design