WIA2002: Software Modelling

Semester 1, Session 2016/17

Lecture 11: Design Modelling

Learning Objectives

- Understand the difference between analysis and design.
- Understand the difference between logical and physical design.
- Understand the difference between system and detailed design.
- Know the characteristics of a good design.
- Understand the need to make trade-offs in design.
- Know how to design a boundary class.

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

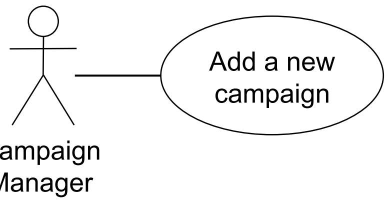
How is Design Different from Analysis?

- 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.

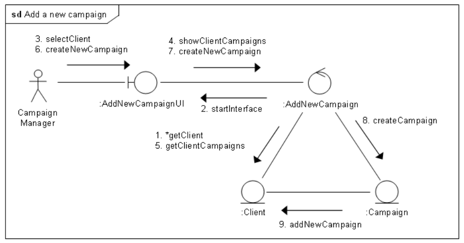
- As an example, in the Agate case study:
  - **analysis** *identifies* the fact that the Campaign class has a title attribute.
  - **design** *determines* <u>how this will be entered</u> into the system, <u>displayed</u> on screen and <u>stored</u> in a database, together with all the other attributes of Campaign and other classes.

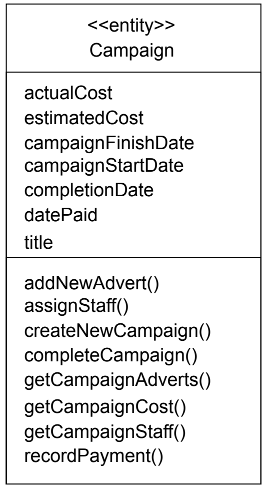
Requirements



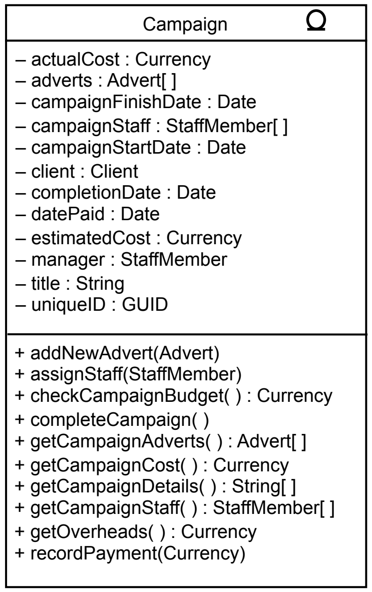
2. To record the details of each campaign for each client. This will include the title of the campaign, planned start and finish dates, estimated costs, budgets, actual costs and dates, and the current state of completion.

Analysis

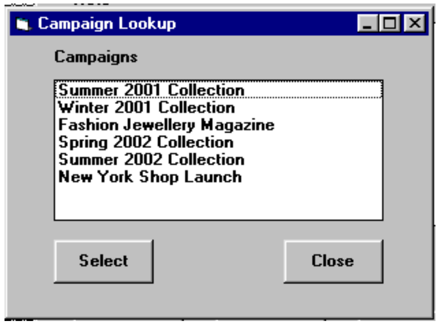




Design


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);
```



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

When Does Analysis Stop and Design Start?

- In a waterfall life cycle:
  - 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’.

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
  - Interaction diagrams, state machine diagrams, deployment diagrams.

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
  - 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.

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

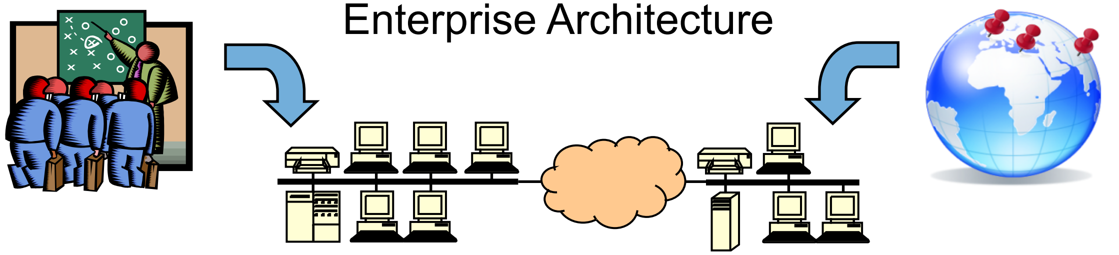
- Traditional detailed design consists of designing:
  - (i) inputs
  - (ii) outputs
  - (iii) processes
  - (iv) files and database structures

- 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 and Detailed Design

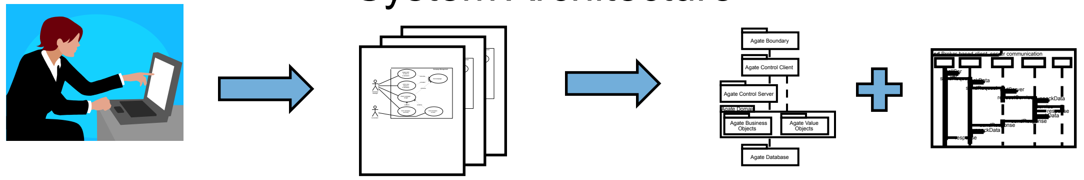
- Detailed design tried to **maximise cohesion**
  - elements of module of code all contribute to the achievement of a single function.
- Detailed design tried to **minimise coupling**
  - reduce unnecessary linkages between modules that made them difficult to maintain or use in isolation from other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

Enterprise Architecture



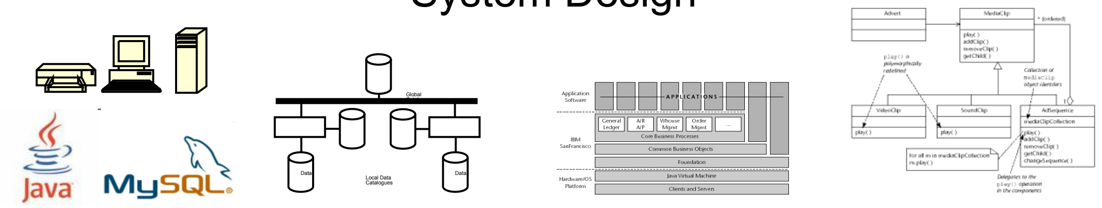
Alignment of IT architecture to business strategy and structure, and existing IT

System Architecture



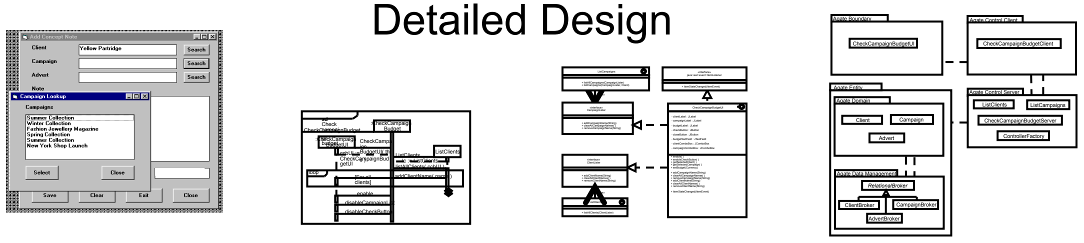
Meeting users' requirements and determining the high-level structure of the system

System Design



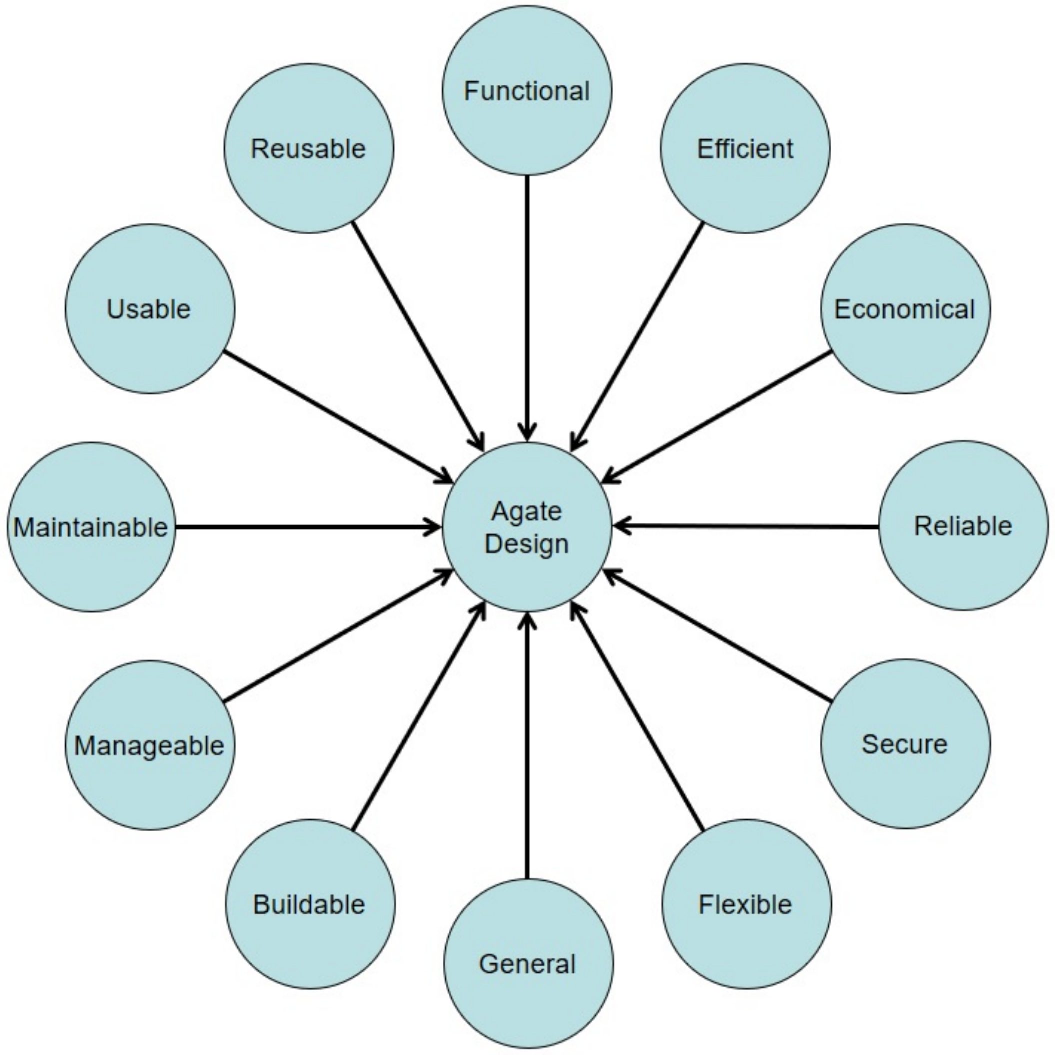
Choosing technologies and frameworks, setting standards, and applying patterns

Detailed Design



Designing user interfaces, interactions, classes, data storage

Qualities of Design



Trade-offs in Design

- Design to meet all these qualities may produce conflicts.
- **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.
- 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.

Measurable Objectives in Design

- Measurable objectives **set clear targets** for designers
    - Represent the requirements that we referred to as *non-functional requirements*.
    - Reflect the fact that system are not built for their own sake, but are developed to *meet the business needs* of some organization.
- Objectives should be **quantified** so that they can be **tested.**

Measurable Objectives in Design

- **<u>Example 1:</u>** 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

- **Example 2:** 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.

DESIGNING A BOUNDARY CLASS

Three-Tier Architecture

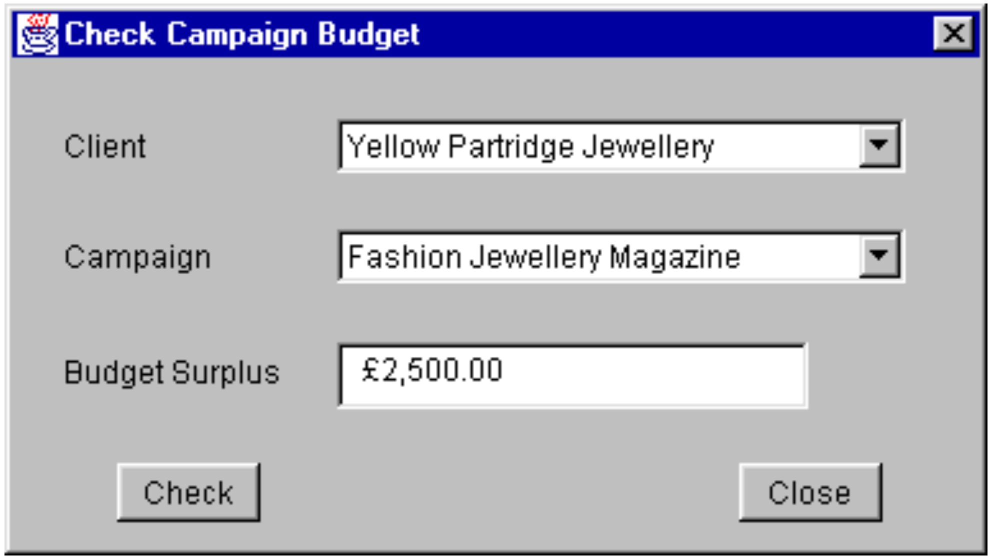
- Separate the responsibilities of classes that
  - responsible for the interface with the user, or with other systems (*boundary classes*).
  - responsible for the business classes (*entity classes*).
  - handle the application logic (*control classes*).
- Different authors have used different terms
  - Entity, Boundary, Control.
  - Model, View, Controller (MVC).
  - Problem Domain Component, Human Interaction Component, Task Management Component.

Presentation Layer

- Handles **interface** with users and other systems.

- Use the techniques and diagrams of UML to <u>model the user interface</u> by adding details to the **boundary classes** in the class model
    - Does not contain business classes—Clients, Campaigns, Adverts, Invoices, Staff etc.
    - Does not contain the business logic—rules like ‘A Campaign must have one and only one Campaign Manager’.
    - Does not handle validation, beyond perhaps simple checks on formatting.

1. Prototype the user interface: Check Campaign Budget Use Case

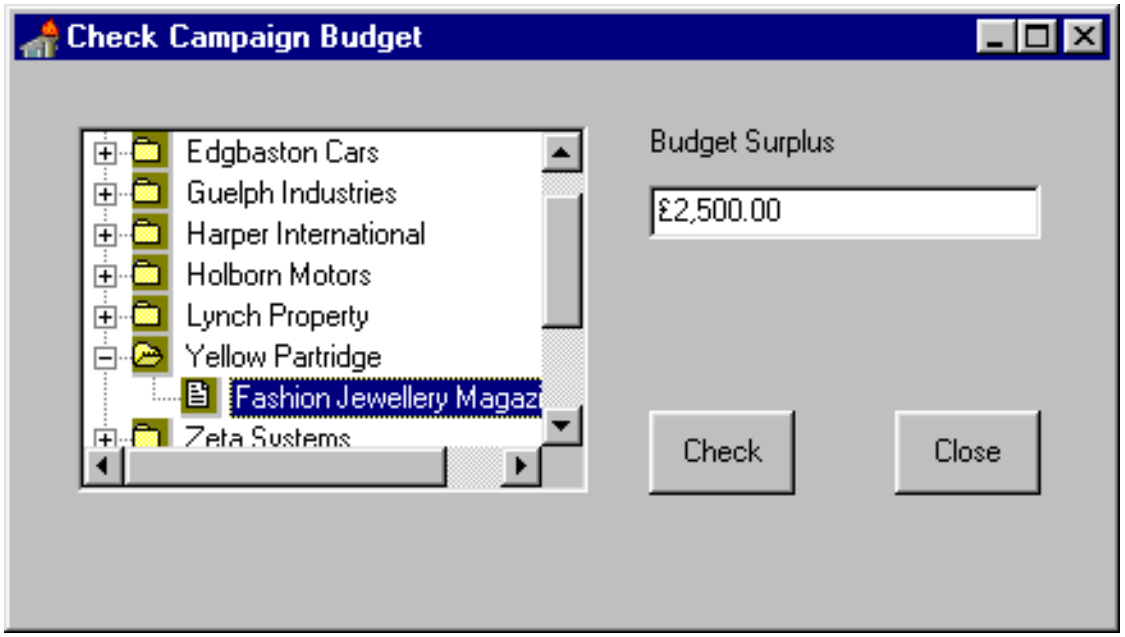

Check Campaign Budget

Client	Yellow Partridge Jewellery
Campaign	Fashion Jewellery Magazine
Budget Surplus	£2,500.00

- In this prototype, Clients and Campaigns are selected in drop-down lists.

- There are other ways…

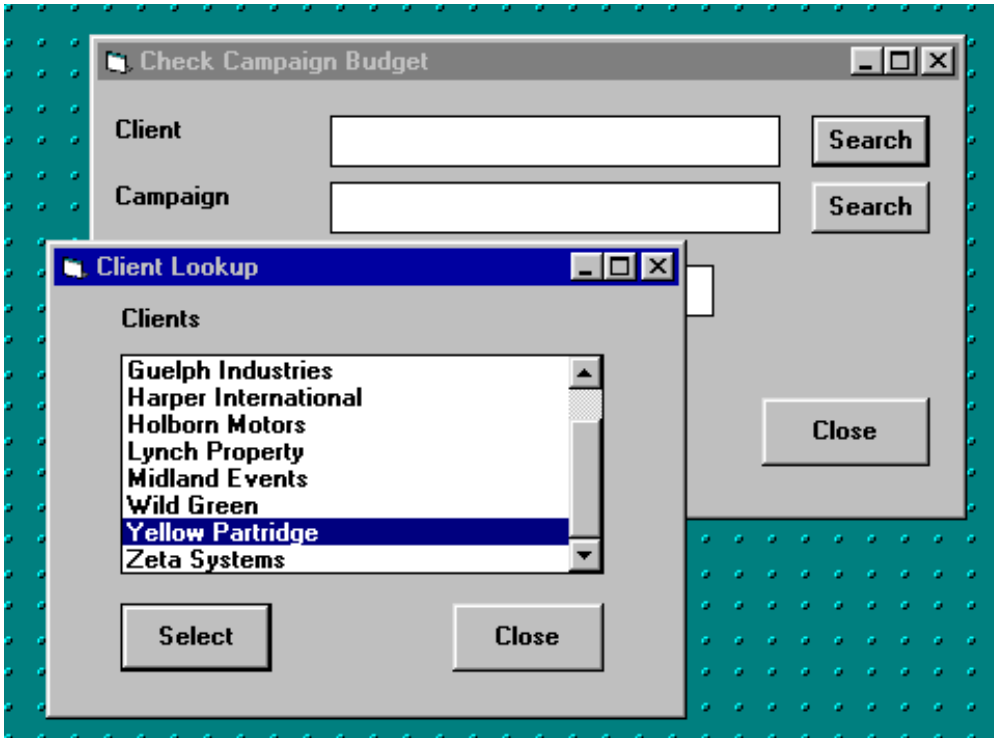
1. Prototype the user interface: Check Campaign Budget Use Case



£2,500.00

- Using a treeview control…

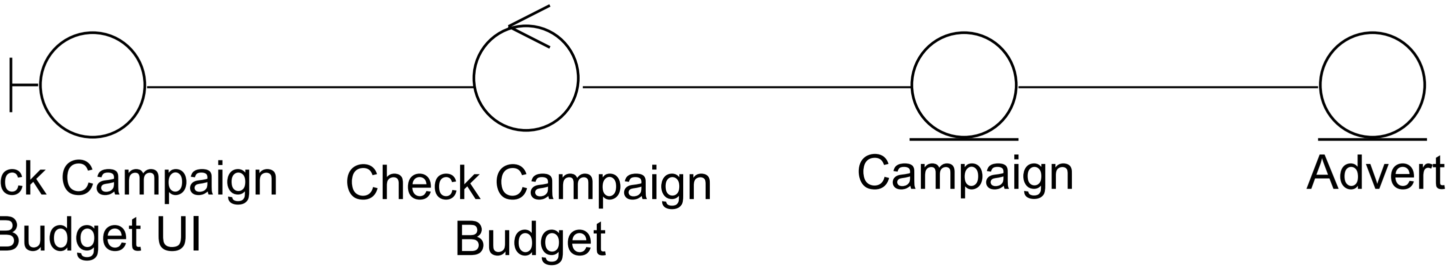
1. Prototype the user interface: Check Campaign Budget Use Case



Guelph Industries
Harper International
Holborn Motors
Lynch Property
Midland Events
Wild Green
Yellow Partridge
Zeta Systems

- Using a separate look-up window.

2. Designing classes

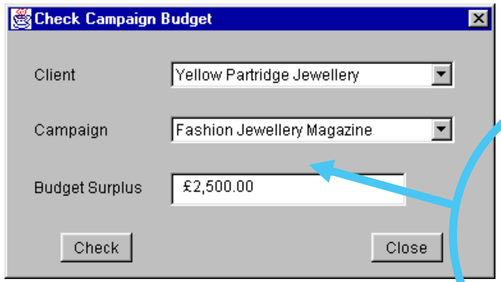


- Collaboration for the use case Check Campaign Budget.

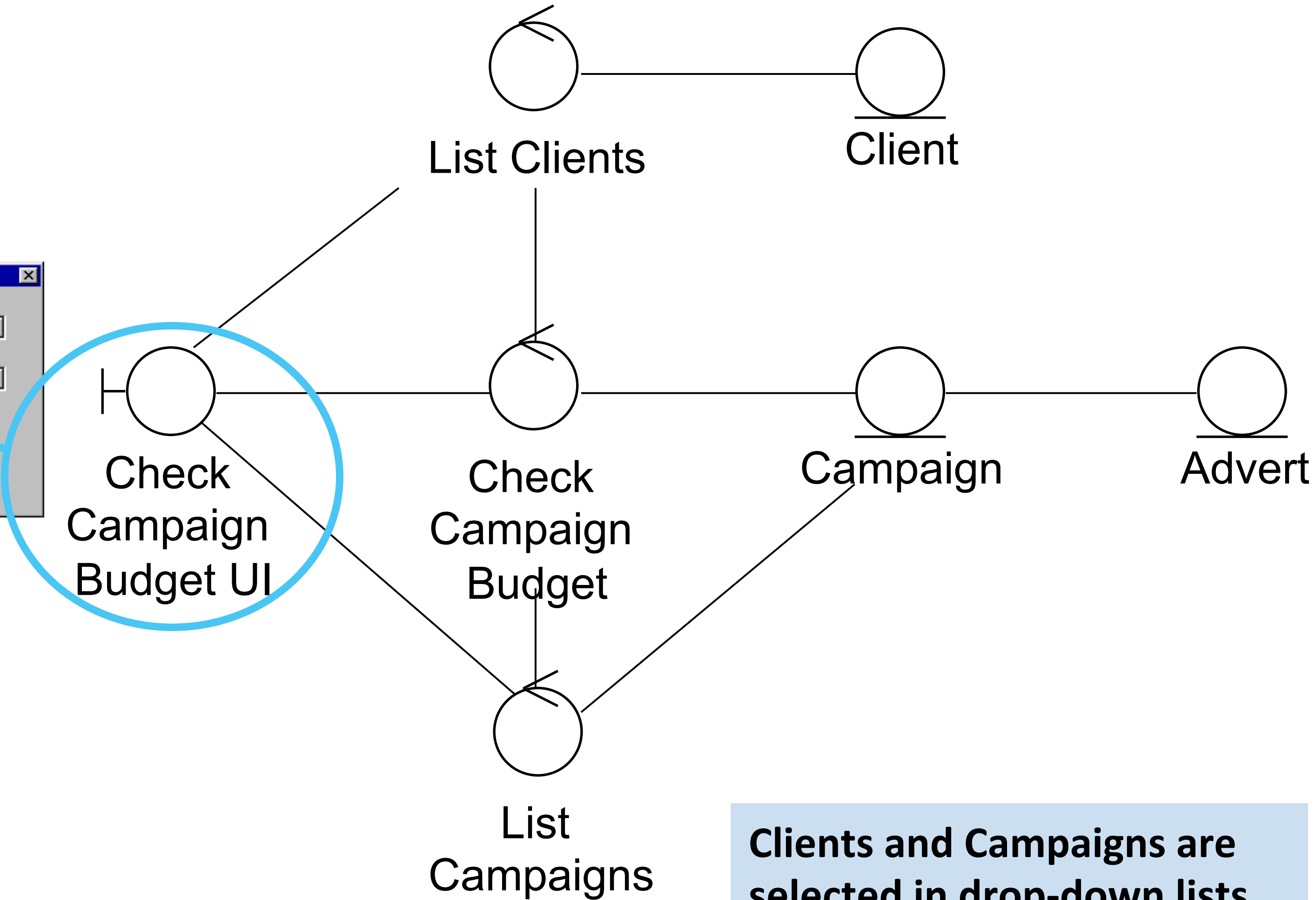
- In order to find the right Campaign, we also *need to use the Client class*, even though it doesn't participate in the real process of checking the budget.

- We also *add control classes* for the responsibilities of listing clients and campaigns.

2. Designing classes: Collaboration for Check campaign budget

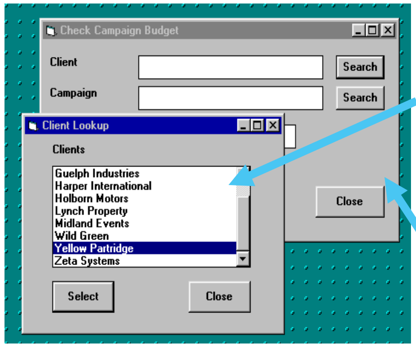

Check Campaign Budget

Client	Yellow Partridge Jewellery
Campaign	Fashion Jewellery Magazine
Budget Surplus	£2,500.00



Clients and Campaigns are selected in drop-down lists

2. Designing classes: Alternative Collaboration for Check campaign budget


Check Campaign Budget

Client [Search]

Campaign [Search]

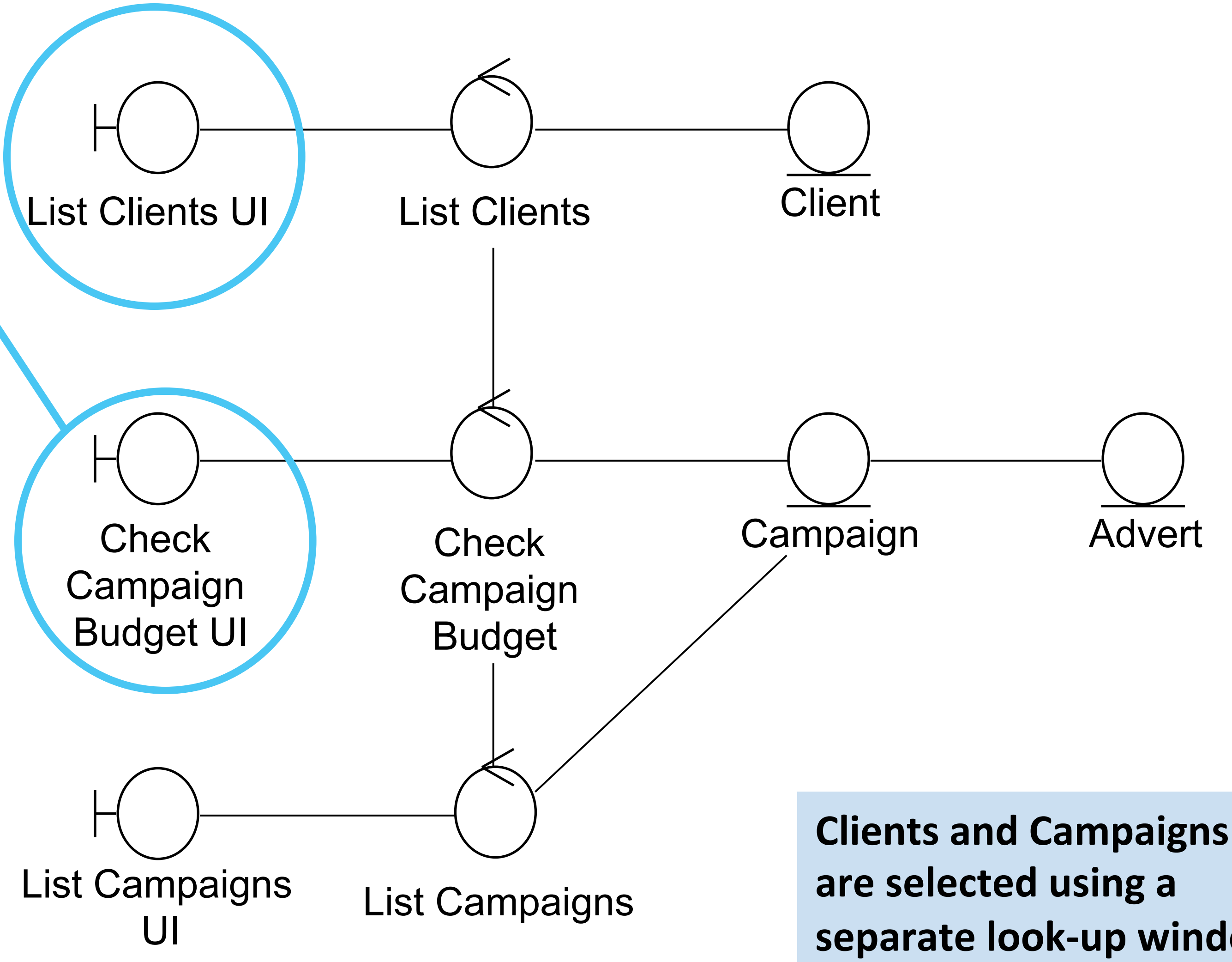
[Close]

Client Lookup

Clients

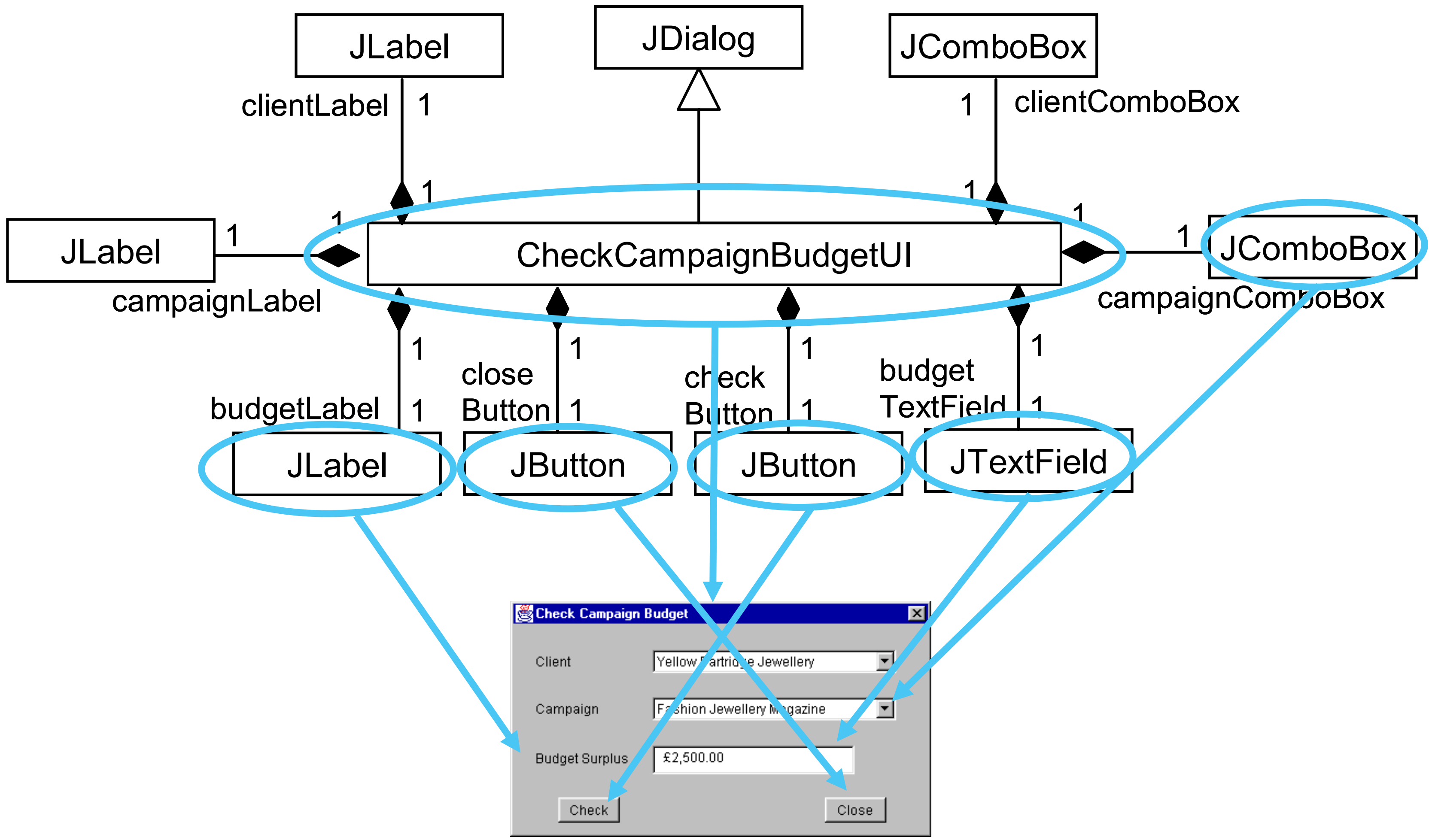
Guelph Industries
Harper International
Holborn Motors
Lynch Property
Midland Events
Wild Green
Yellow Partridge
Zeta Systems

[Select] [Close]



Clients and Campaigns are selected using a separate look-up window

2. Designing Classes: Class Diagram for CheckCampaignBudgetUI

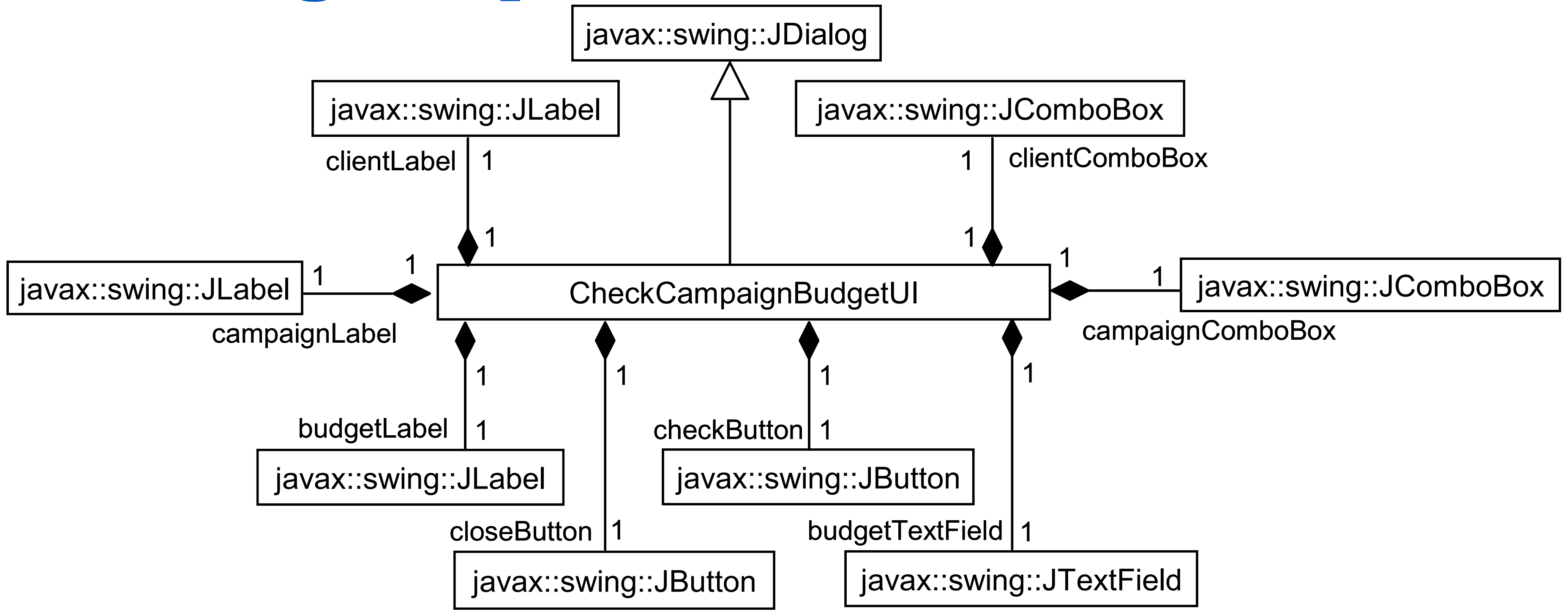


Client: Yellow Partridge Jewellery

Campaign: Fashion Jewellery Magazine

Budget Surplus: £2,500.00

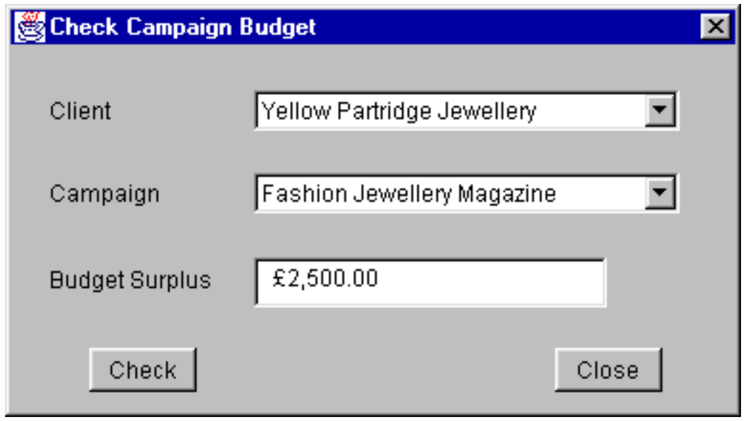
2. Designing Classes: Package Dependencies



The `CheckCampaignBudgetUI` class is dependent on the classes in Java **Swing**, and this can be shown using packages in a class diagram.

2. Designing Classes: Single Class for CheckCampaignBudgetUI

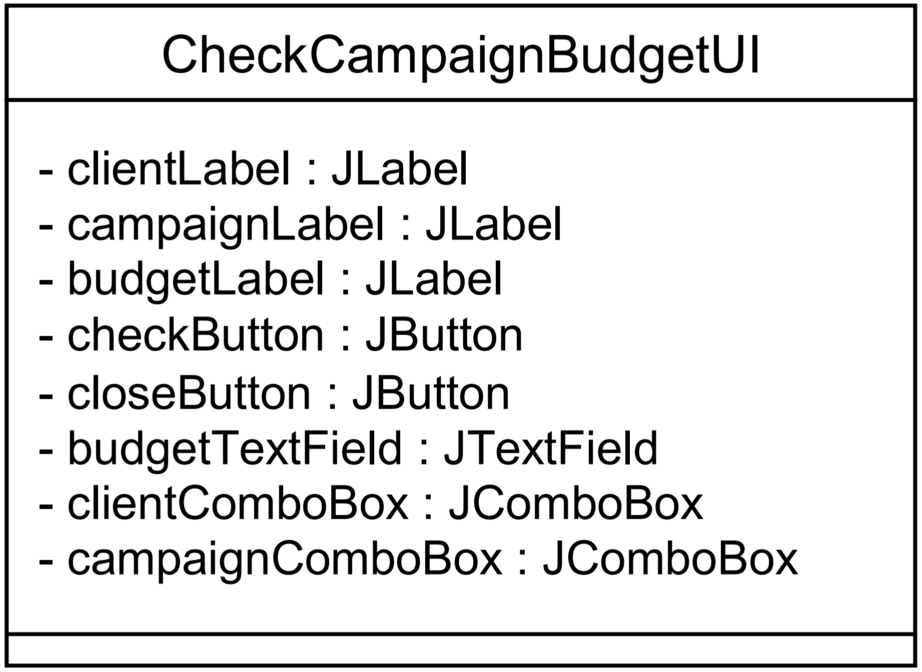
- Draw in your own lines to show which attribute is which element of the interface.



Yellow Partridge Jewellery

Fashion Jewellery Magazine

£2,500.00



CheckCampaignBudgetUI

- clientLabel : JLabel
- campaignLabel : JLabel
- budgetLabel : JLabel
- checkButton : JButton
- closeButton : JButton
- budgetTextField : JTextField
- clientComboBox : JComboBox
- campaignComboBox : JComboBox

Key points

- The difference between analysis and design is that:
  - 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.

- The difference between logical and physical design is that
  - 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

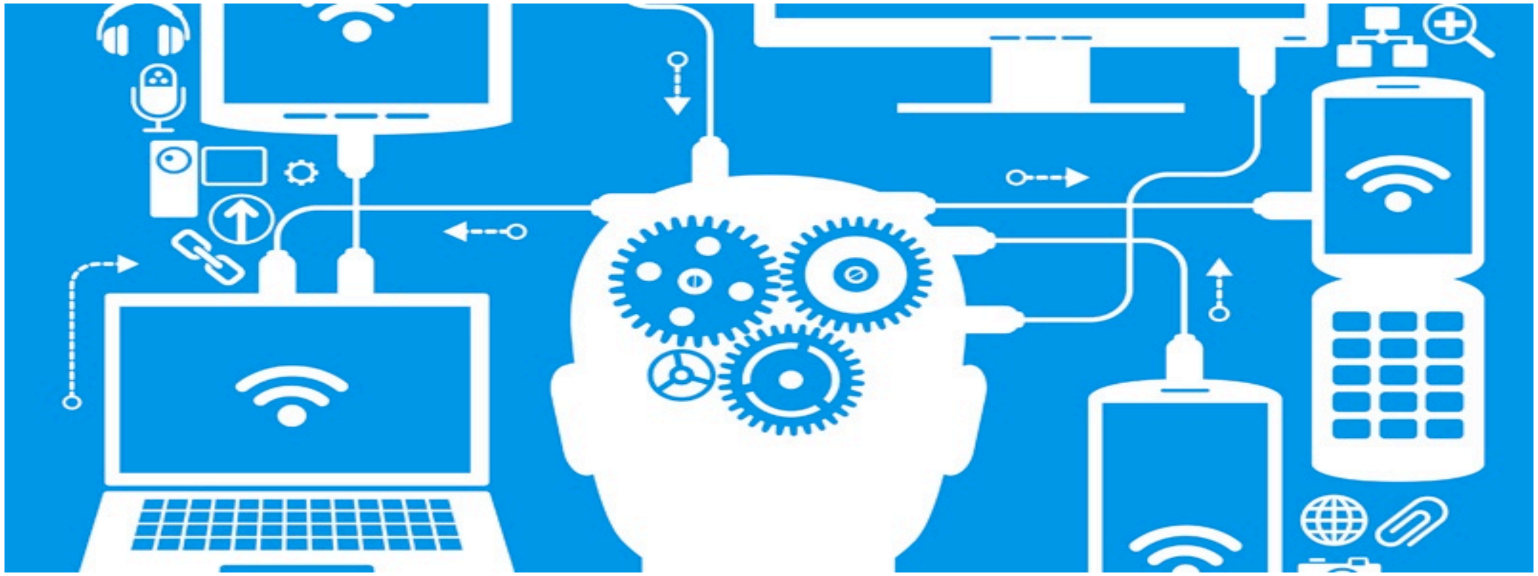
Key points

- The difference between system and detailed design is that
  - System Design deals with the high level architecture of the system
  - Traditional detailed design consists of designing:
    - (i) inputs
    - (ii) outputs
    - (iii) processes
    - (iv) files and database structures
- The characteristics of a good design needs to be determined early in the development and they need to have trade-offs in design.
- 2 steps to design a boundary class
  - Prototype the user interface
  - Designing classes

References

- Alan Dennis, Barbara Haley Wixom & David Tegarden. 2015. Systems Analysis and Design with UML, 5th edition, Wiley.
- Simon Bennett, Steve McRobb & Ray Farmer. 2010. Object Oriented Systems Analysis and Design using UML 4th Edition, McGraw-Hill. (Chapters 14 & 17)
- UML Reference Manual (OMG, 2009)
- Bennett, Skelton and Lunn (2005)

In the next lecture..



Lecture 12: Software Testing