CS4125
SYSTEMS ANALYSIS
SPRING SEMESTER 2010-2011

J.J. Collins
Dept of CSIS
University of Limerick

Lecture O - W9L2

Introduction to Design

1. Design Overview: Moving into Design

- Process of design separate from analysis.
- Analysis - what should the system do?
- Design - how should the system as specified in analysis, be implemented?
- Implementation independent or logical design is distinguished from implementation dependent or physical design.
- Design takes place at two levels:
  - System design – architecture.
  - Detailed design: classes.
- In producing a design, will be working within guidelines such as quality criteria and measurable objectives.

1. Design Overview: Analysis v Design

- Analysis: what happens in the current system & what is required in the new one.

- Design is stating 'how the system will be built without actually building it' - Rumbaugh (1997).
- Design: producing a solution that meets the requirements as specified in the analysis phase.
- Jacobson et al. (1992) conceptualise design as part of the construction process, together with implementation.
- System designer focused on the implementation of the new system, while the systems analyst is focused on the way the business is organised and a possible better organisation.

- E.g. during analysis, the class *staffMember* has been identified as having a *staffid* attribute. During design, it is determined how this will be entered into the system, displayed and stored in a database.

1. Design Overview: Design & Lifecycles

- Traditional: phase
  - Project management.
  - Staff skills and experience.
  - Client decisions: end of analysis often key point in project.
  - Choice of development environment.
- Iterative: activity
  - Risk mitigation – immediate focus on architecture, incremental development implies that technology problems are identified early.
  - Change management – just as users requirements change during a project, so will technology.
  - Team learning – all members involved in the project from inception. Those involved in testing and deployment will better understand the system and the probability of bugs getting through is reduced.
  - Improved quality – continuous testing.

1. Design Overview: Transitions in Methodologies

- Deliverables in the structured approach:
  - Analysis: data flow diagrams.
  - Design: structured charts or structured diagrams.
- Object oriented approach: same model used in analysis and design. Classes identified in analysis and refined in design.
- Seamless transition blurs distinction between analysis and design, when we move into design, different information is added to the class diagram and other diagrams used to support the class diagram.

1. Design Overview: Logical & Physical Design

Physical Design

- Deals with those aspects of the system that are platform dependent.
- May be limited by existing configurations.
- Use of client server architectures and open system standards allow for different hardware and software to operate together.
- Some choices dependent on system platform which effect system architecture, the design of objects, and interfaces with various components of the system.

1. Design Overview: Logical & Physical Design

Examples:

- Creating a distributed system will require the use of middleware such as .Net or Websphere to allow objects to communicate with each other across a network.
- Developing a SOA will require Web Services (SOAP, XML, WSDL, etc.).
- The decisions to write programs in Java and to use a relational database that supports ODBC (Object Data Base Connectivity) will require the use of JDBC (Java data Base Connectivity), and the creation of classes to map between the objects and the relational database.
- If using Java, choice of Java AWT (Abstract Windowing Toolkit) or the Java Swing classes for designing interfaces.
- Java does not support multiple inheritance, other languages such as C++ do. Will have to be implemented using Java's interface mechanism.
- If the system has to interface to special hardware such as bar-code scanners, it may be necessary to design the interface so that it can be written in C as a native method and encapsulated in a Java class. Java cannot directly access low-level features of hardware.

1. Design Overview: Logical & Physical Design

Logical Design

- Concerned with those aspects of the system that can be designed without knowledge of the implementation platform.
- It is also the case that many design decisions can be made independently of the hardware and software platform:
  - The interaction between objects to provide functionality of a particular use case can be designed using interaction sequence or communication diagrams.
  - The layout of data entry screens can be designed in terms of the fields that will be required to source data for objects. However, the exact nature of a textbox and whether it is a Borland C++ TEdit, a Delphi Edit, a Java textField etc., can be left until much later.
  - The nature of data to be sent to and received from special hardware or other systems can be determined without needing to design the exact format of the messages.
- Most projects, platform issues resolved at beginning of design.
- If not, logical design precedes physical design.

1. Design Overview: System and Detailed Design

- System design is concerned with the overall architecture of the system and setting of standards such as human computer interfaces (HCI).
- Detailed design is concerned with designing individual components to fit this architecture and to conform to standards i.e. design of objects, using design patterns, etc.

1. Design Overview: System Design

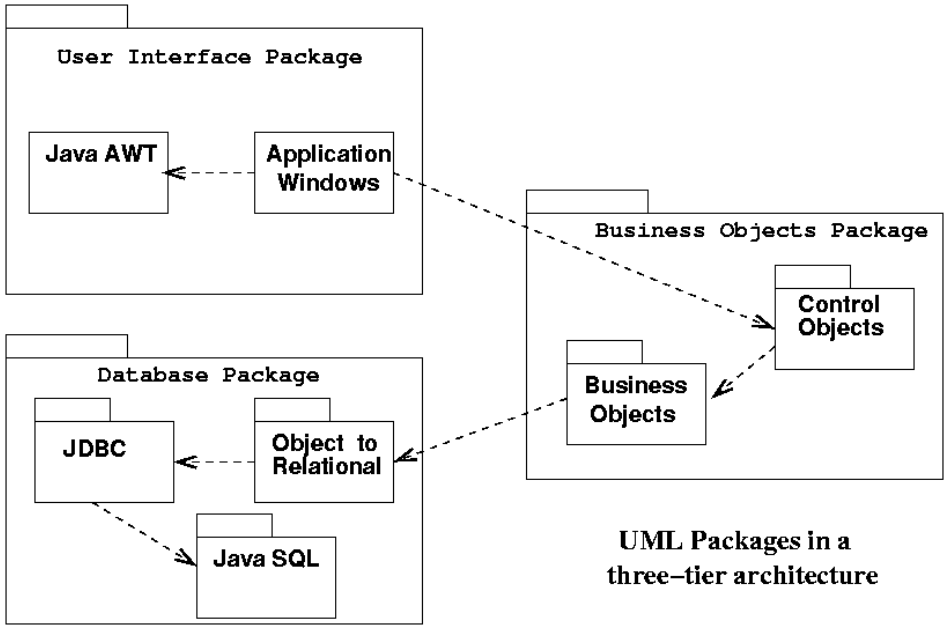
- Architecture - if client server, how are processes and objects distributed on different machines.
- Subsystems identified, will need to determine communications protocols between processors.
- Concurrency needs to be explicitly designed into the system.
- Job design often included in system design.
- Decisions about design standards will have to be made i.e. screen layouts, help, errors, etc.
- How people use particular use cases will be included in the detailed design of human computer interfaces.

1. Design Overview: Traditional Detailed Design

- In the 60's and 70s, detailed design consisted of:
  - Designing inputs - layout of menus and data entry screens, file organisations, etc..
  - Designing outputs - layout of enquiry screens, reports and printed documents.
  - Designing processes - choice of algorithm
  - Designing files - file organisation and the access methods used to update and retrieve data from files.
- The work of Jackson (1975) provided diagramming notation and a methodology.
- The introduction of two criteria - cohesion and coupling, by Yourdon and Constantine (1979).
- Cohesion: production of modules, where all the elements of a module contribute to the performance of a single function/task/etc.

1. Design Overview: What to Add in OO Detailed Design

- During analysis, the classes in the class diagram will represent the business model, but will not include classes to handle
  - interfaces to users and other systems,
  - the storage of data,
  - or the overall coordination of the other classes into programs.
- Coad and Yourdon (1991), Coad et al. (1997) call the business classes the problem domain component and develop three further components in the design phase:
  - Human interface component.
  - Data management component.
  - Task management / system interaction component.



UML Packages in a three–tier architecture

1. Design Overview: Reusability

Concerns Requiring Attention in OO Detailed Design

- Object Oriented Languages promote reuse through encapsulation of functionality and data together in classes and through inheritance.
- Design reuse takes place at two levels:
  - Design patterns.
  - Business classes may be provided by reusing classes that already have been designed within the organisation, or brought in from the outside.
- Move towards the use of components purchased from vendors.
- CASE tool suppliers such as Select Software tools have added component management software to their range of products.
- IBM's Websphere (formerly San Francisco) makes general purpose business classes available.
- The assignment of responsibilities to classes is related to reuse.
- Larman (1998) identifies this activity as the main task in design.
- If the designer makes the wrong decision, the class may be less reusable, and constrain the design of other classes.

1. Design Overview: What makes for Good Analysis

- Correct scope: everything in the analysis model should fall within the scope of the system. Coad et al. (1997) include a *not this time* component with their other four components (problem domain, human interface, data management and system interaction).
- Completeness: everything that is within the scope should be documented in the analysis model. Analysis patterns and strategies as proposed by Coad et al. (1997) and Fowler (1997) can help less experienced analysts to identify likely issues.
  - Rumbaugh (1997) suggests that some of the requirements found during analysis are not analysis requirements but design requirements.
- Correct content.
- Consistency.

- Errors of scope or completeness will result in a finished product that does not do what the user requires.
- Errors of correctness and consistency will typically result in bugs.

1. Design Overview: What Makes for Good Design

- Yourdon and Constantine (1979) cite efficiency, flexibility, generality, maintainability and reliability.
- DeMarco (1979) proposes efficiency, maintainability, buildability.
- Page-Jones (1988) suggests that a good design is efficient, flexible, maintainable, manageable, satisfying and productive. The latter two points highlight issues concerned with HCI and the need for a design to produce a usable system.

1. Design Overview: Planning for Design

- If the hardware and software platform has not been decided upon, the project manager must ensure that design is logical and must plan for the time when physical design can begin.
- The system architecture must be agreed and system standards set that will affect the design of individual sub-systems.
- Allow for designers to familiarize themselves with new platforms through training, and hire in external expertise through contractors.
- Design objectives must be set and procedures put in place for testing those that can be tested by simulation during the design process.
- Procedures must be put in place for resolving conflicts between constraints and requirements.
- The amount of time to be spent on different aspects of the system must be agreed - system object, HCI, interface objects.

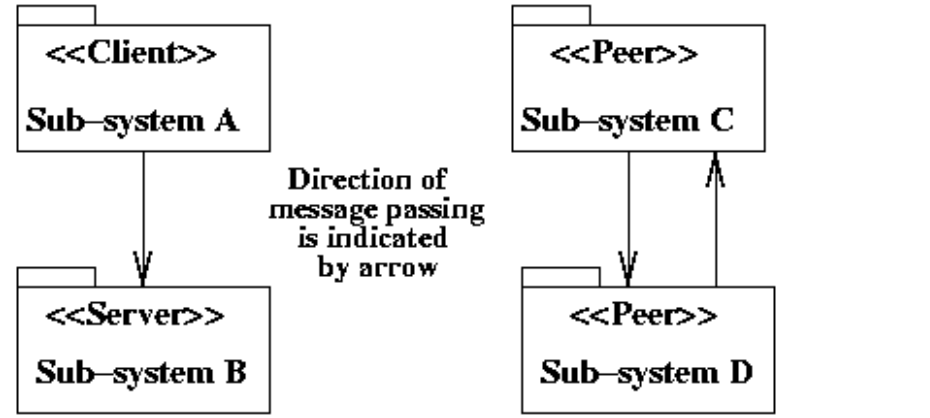
2. System Design: Activities

- Sub-systems and major components identified.
- Inherent concurrency identified.
- Sub-systems allocated to processors.
- Data management strategy selected.
- HCI standards specified.
- Code development standards specified.
- Control aspects of the application planned.
- Test plans specified.
- Priorities are set for design trade-offs.
- Implementation requirements are identified.

A similar list is identified by Rumbaugh et al. (1991).

2. System Design: Sybsystems

- An OO sub-system encapsulates a coherent set of responsibilities in order to ensure that it has integrity and can be maintained.
- E.g. HCI sub-system, data management sub-system, business logic sub-system.



Styles of communication between sub–systems

2. System Design: Subsystems

- Advantages of division of system into sub-systems:
  - Produces smaller units of development.
  - Maximises reuse at the component level.
  - Helps developers to cope with complexity.
  - Improves maintainability.
  - Improves portability.
- Each sub-system should have a clearly specified boundary and fully defined interfaces.
- Each sub-system provides services for other sub-systems, and two styles of communication make this possible: client server and peer to peer.
- Client server sub-systems better - less tightly coupled.
- Could also use component and deployment diagrams.

3. Data Management Issues

- Relational: large data volumes and extensive query support.
- Object-oriented: extensive transaction support.

- What is the support typically offered by a DBMS
  - Different views of the data by different users.
  - Access control.
  - Distributed deployment.
  - Security.
  - Enforcement of integrity constraints.
  - Access to data by various other applications.
  - Data recovery.
  - Portability across platforms.
  - Query languages and query optimisation.

4. Other Considerations

- Development Standards i.e. HCI, I/O, error handling, and construction guidelines.
- Prioritizing design trade-offs.
- Design for implementation. When system is first introduced, it must be initialised and populated with existing data volumes.

5. Reading

- Chapter 12 and 13 in Bennett et al. (Fourth Edition).