Moving into Design

Analysis versus Design

Logical versus Physical Design

System versus Detailed Design

Characteristics of Good Design

Tradeoffs in Design

Analysis versus Design (1)

- Analysis
  - **What** happens in the current system?
  - What is required in the new system?
  - Seeking understanding, investigating requirements and modelling requirements
  - Focus: the way the business is organised and a possible better organisation
- Design
  - Produce the best possible solution that satisfies the requirements
  - **How** the system will be constructed without actually building it
  - Focus: implementation of the new system

Analysis versus Design (2)

- Design as a stage or an activity
  - Lifecycle models: waterfall versus iterative
  - Unified process
    - Phases: inception, elaboration, construction, transition
    - Each phase requires one or more iterations
    - Each phase requires a number of activities
    - Activity effort lifecycle: first increase, then decline
- Progress metric: over time the less analysis activity and more design activity

Analysis versus Design (3)

- Traditional Lifecycle
  - A clear break between analysis and design
- Advantages
  - Project Management
  - Staff skills and experience
  - Client decisions
  - Choice of development environment

- Iterative Lifecycle
  - Both activities overlap
- Advantages
  - Risk mitigation
  - Change management
  - Team learning
  - Improved quality

Analysis versus Design (3)

- Traditional Lifecycle
  - A clear break between analysis and design
- Advantages
  - Project Management
  - Staff skills and experience
  - Client decisions
  - Choice of development environment

- Iterative Lifecycle
  - Both activities overlap
- Advantages
  - Risk mitigation
  - Change management
  - Team learning
  - Improved quality

Very important: in both cases the requirements are explored fully!

Analysis versus Design (4)

- Why should we consider the separation?
  - Traditional methods enforce separation!
    - Dataflow versus structure diagrams
  - Object oriented methods enforce **seamlessness**!
    - Object models
    - Same diagrams with added detail
    - The analysis – design continuum: from “what to do” to “exactly how to do it”

Logical versus Physical Design (1)

- Implementation platform independent versus implementation platform dependent design
  - Switch: when a decision about the implementation platform (hardware and software) is taken
  - Examples
    - Distributed platform requires middleware
    - Object oriented language and relational database requires object to relation mapping – ODBC
    - Choice of programming language limits the choice in GUI libraries, or particular language features used (e.g. multiple inheritance), etc.

Logical versus Physical Design (2)

- Examples of platform independent design
  - Designing how the interaction that implements a particular use case will take place
  - Designing the layout of data entry screens and the objects that are required to process them
- The main advantage of separating logical and physical design is design reuse in multiple platforms

System versus Detailed Design (1)

- System design is concerned with the overall architecture of the system and the setting of standards
  - Organisation of the system into sub-systems, organisation of the communication between sub-systems
    - Distribution and concurrency?
  - Standards – conventions to be followed throughout
    - Job design
- Detailed design is concerned with the designing individual component that fit the architecture and conform to the standards

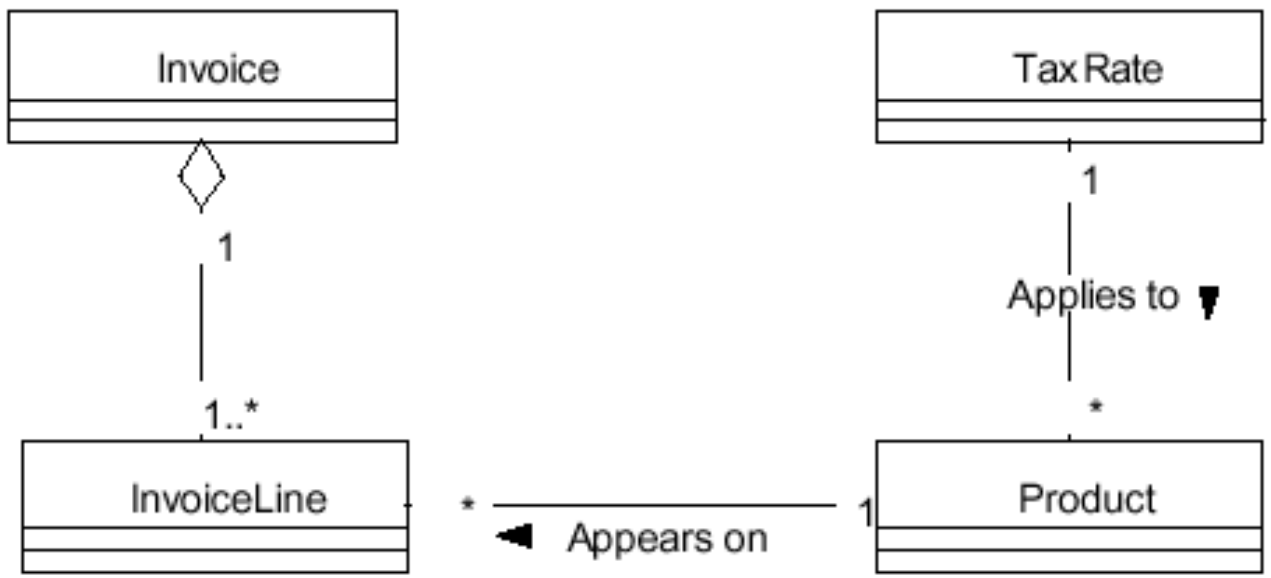
System versus Detailed Design (2)

- Traditional detailed design
  - Designing input, output, processes and files
  - Easy to develop and maintain modules: cohesion and coupling
    - High Cohesion: design modules that carry out a clearly defined process or groups of processes that are functionally related
    - All elements of a module contribute to the performance of a single function
    - Bad cohesion: coincidental, logical, temporal and sequential
    - Low Coupling: design modules that perform their function using only the data that is passed to them and using the minimum necessary amount of data
    - Modules are independent of one another and amendments in one do not have knock on effects on others
    - Poor coupling: use of global variables, large amounts of data passed in parameters, control information passed as parameters

System versus Detailed Design (3)

- Good Systems comprise of encapsulated modules exhibiting high cohesion and low coupling
  - In object-oriented systems the same applies to classes!
- Object-oriented detailed design
  - Entity classes in analysis, boundary and control classes in detailed design
    - Boundary to the database!
  - Human interface, data management and task management components
  - Presentation, application logic and storage layers
- Reuse and assignment of responsibilities
  - Design reuse: design patterns and previously developed business classes (**software components**)
  - Code reuse: encapsulation and inheritance

System versus Detailed Design (4)



Where should we put the responsibility for tax calculation?

Characteristics of Good Design (1)

- Good design requires good analysis
  - Fixing faults later in the lifecycle is more costly
  - Criteria for good analysis
    - Correct scope
      - Not this time component
      - Scope creep in traditional and iterative lifecycles?
    - Completeness
      - Analyst experience, analysis patterns!
      - Non-functional requirements, design requirements
    - Correct content
      - Accuracy should not be confused with precision!
    - Consistency
      - Consistency between diagrams

Characteristics of Good Design (2)

- Errors in scope and completeness ⇒ a system that does not do what it is supposed to
- Errors in correctness and consistency ⇒ a systems that performs incorrectly and problems for the designers

- Good system design objectives
  - Functional, Efficient, Economical (development and running costs), Reliable (software and hardware errors, integrity maintenance, testing and dependencies), Secure (legislation), Flexible (modifiability), General (portability), Buildable (design clarity, language features), Manageable, Maintainable (quality of design documentation), Usable (affordance, productivity), Reusable (economies and development culture)

Characteristics of Good Design (3)

- Design objectives tend to conflict with one another
  - Many conflicts result from the non-functional requirements
  - Constraints increase conflicts (e.g. budget, timescale, integrate existing hardware)
  - Need for compromises or tradeoffs in design
  - Documentation of rational is crucial!
- Measurable design objectives
  - Expected benefits from cost-benefit analysis!

Planning for Design

- When will the architecture and the system standards be set?
- When will the development platform be fixed?
  - How will issues regarding the team expertise on the platform will be addressed?
- What are the objectives against which the design should be tested?
  - What are the testing procedures to be followed?
- How are conflicts resolved?
  - What are the procedures for agreeing and documenting tradeoffs?
- How much time should be spent on each aspect of the system (i.e.user interface, data management, etc)?