Software Design

Massimo Felici

Room 1402, JCMB, KB

0131 650 5899

mfelici@inf.ed.ac.uk

Software Design

- IEEE standard glossary: "the process of defining the architecture, components, interfaces and other characteristics of a system or component."
- Usually a two stage process:
  - **Architectural Design** (or High-level Design)
    - What are the components and how do they relate?
    - How does the system architecture deal with issues that pervade the system?
  - **Detailed Design** deals with the function and characteristics of components and how they relate to the overall architecture.
- No "magic bullet" in general

The Link to Requirements

- Main activity in design:
  - decomposing system (components) into smaller more manageable components.
- Ideally we retain the link from requirements to components (traceability):
  - By allocating a particular requirement to a particular component as we decompose, e.g., in VolBank, we might require a log.
  - By decomposing requirements into more refined requirements on particular components, e.g., a particular function in VolBank might be realised across several components.
  - Some requirements (e.g., usability) are harder to decompose, e.g., it takes 30 minutes to become competent in using the system.
- We might require traceability back from the design

Traceability

- There are four basic types of traceability:
  - **Pre-traceability** (e.g., requirements-sources, requirements-rationale, etc.)
    1. **Forward-to** requirements traceability links other documents preceding requirements (e.g., users document)
    2. **Backward-from** requirements traceability links requirements to their sources (e.g., rationale)
  - **Post-traceability** (e.g., requirements-architecture, requirements-design, requirements-interface, etc.)
    3. **Forward-from** requirements traceability links requirements to design and implementation
    4. **Backward-to** requirements traceability links design and implementation back to requirements.
- To manage requirements, you need to maintain traceability information (e.g., Traceability Tables)
- Requirements Management Tools support traceability practice (e.g., IBM Rational RequisitePro or Telelogic DOORS)

Main Topics in Software Design

- Basic design concepts
- Key issues
  - the main elements of software that need to be managed
- Structure and architecture
  - design in the large and design in the small
- Design notations
- Design quality and evaluation
- Design strategies

Basic Design Concepts

- Design is a pervasive activity
  - often there is no definitive solution
  - solutions are highly context dependent
- Design links requirements to "implementable specifications"
  - definitions of components that are easily codable
- Distinction between architectural design and detailed design

Key Design Techniques

- **Abstraction**
  - ignoring detail to get the high level structure right
- **Decomposition** and **Modularisation**
  - big systems are composed from small components
- **Encapsulation/information hiding**
  - the ability to hide detail (linked to abstraction)
- Defined **interfaces**, seperable from implementation
- Evaluation of structure:
  - **Coupling**: How interlinked a component is.
  - **Cohesion**: How coherent a component is.

Key Issues in S/W Design

- **Concurrency**
  - what are the main concurrent activities?
  - how do we manage their interaction?
  - Often there is significant interaction that needs management
  - For instance, in VolBank, matching and specifying skills and needs goes on concurrently
- **Workflow** and **event handling**
  - What are the activities inside a workflow?
  - How do we handle events?
- **Distribution**
  - how is the system distributed over physical (and virtual) systems?

Key Issues in S/W Design continued

- **Error handling** and **recovery**
  - action when a physical component fails (e.g., the database server).
  - how to handle exceptional circumstances in the world (e.g., a volunteer fails to appear).
- Persistence of **data**:
  - does data need to persist across uses of the system, how complex?
  - How much of the state of the process?
- Can you think through some of these issues for VolBank?

Architecture and Structure

- **Architectural structures** and **viewpoints**
  - attempt to deal with facets separately, e.g., physical view, functional (or logical) view, security view, etc.
- **Architectural styles**, for example:
  - Three-tier architecture for a distributed system (interface, middleware, back-end database)
  - Blackboard
  - Layered architectures
  - Model-View-Controller
  - Time-triggered
- **Design patterns**
  - small-scale patterns to guide the designer
- **Families** and **frameworks**
  - component set and ways of plugging together
  - software product lines

What are the Architect's Duties?

- Get it **Defined**, **documented** and **communicated**
- Make sure everyone is **using** it (correctly)
- Identify architecture timely **stages** that support the overall organization progress
- Make sure the **software** and system **architectures** are in synchronization
- Act as the emissary of the architecture
- Make sure management understands it
- Make sure the right **modeling** is being done, to know that **quality attributes** are going to be met
- Identify suitable **tools** and **design** environments
- Identify and interact with **stakeholders**
- Make sure that the architecture is not only the right one for operations, but also for deployment and sustainment
- Resolve disputes and make tradeoffs
- Resolve technical problems
- Maintain morale
- understand and plan for **evolution**
- Manage **risk** identification and risk **mitigation strategies** associated with the architecture

Architectural Design

- **Advantages**:
  - Stakeholder Communication
  - System Analysis
  - Large-scale reuse
- **Main Activities**:
  - System structuring
  - Control modelling
  - Modular decomposition
- There is no clear distinction between Sub-systems and modules. Intuitively,
  - Sub-systems are independent and composed of modules, have defined interfaces for communication with other sub-systems
  - Modules are system components and provide/make use of service(s) to/provided by other modules.

Architecture Models

- Architecture Models that may be developed may include:
  1. A **static structural model** that shows the sub-systems or components that are to be developed as separate units.
  2. A **dynamic process model** that shows how the system is organised into processes at run-time. This may be different from the static model.
  3. An **interface model** that defines the services offered by each sub-system through their public interface.
  4. **Relationship models** that show relationships such as data flow between the sub-systems.

UML Design Notations

- **Static Notations**:
  - Component diagrams
  - Class and object diagrams
  - Deployment diagrams
  - CRC Cards
- **Dynamic Notations**:
  - Activity diagrams
  - Collaboration diagrams
  - Statecharts
  - Sequence diagrams

Comparing Architecture Design Notations

- **Modelling Components**:
  - Interface, Types, Semantics, Constraints, Evolution, Non-functional Properties
- **Modelling Connectors**:
  - Interface, Types, Semantics, Constraints, Evolution, Non-functional Properties
- **Modelling Configurations**:
  - Understandable Specifications, Compositionality (and Conposability), Refinement and Traceability, Heterogeneity, Scalability, Evolvability, Dynamism, Constraints, Non-functional Properties

Quality Analysis and Evaluation

- The system architecture affects the quality attributes of a system
- **Quality attributes**:
  - Performance, security, availability,… modifiability, portability, reusability, testability, maintainability, etc.
- **Quality analysis**:
  - reviewing techniques, static analysis, simulation, performance analysis, prototyping
- **Measures (metrics)**:
  - Defined measure on the design
  - Predictive, but usually very dependent on the process in use

Design Strategies

- Depends on the type of system:
  - **Function** oriented: sees the design of the functions as primary
  - **Data** oriented: sees the data as the primary structured element and drives design from there.
  - **Object** oriented: sees objects as the primary element of design

VolBank: Example

- Suppose we consider two requirements:
  - That a request for a volunteer should produce a list of volunteers with appropriate skills.
  - The system shall ensure the safety of both volunteers and the people and organisations who host volunteers.
    - This may decompose into many more specific requirements:
      - That the organisation has made reasonable efforts to ensure a volunteer is bona fide.
        - » That we have a confirmed address for the individual: i.e., the orginal address is correct, and only the volunteer can effect a change in address.

Reading/Activity

- Please read Chapter 3 – Software Design - of the SWEBOK for an overview of the work on design.
- Please read chapters 4 and 5 of the Schaum's outline on UML for an introduction to class diagrams.
- Please look at the additional material in the course webpage.

Summary

- Design is a complex matter
- Generally two stages:
  - Architecture Design (or High-level Design)
  - Detailed Design
- Many notations and procedures to support design
- More domain-specificity for easier design task
- Design links requirements to construction, essential to ensure traceability