Software Design

Stuart Anderson

Room 1610, JCMB, KB

0131 650 5191, soa@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 – what are the components and how do they relate? In particular, try to deal with issues that pervade the system.
  - Detailed design – the function and characteristics of components and how they relate to the overall architecture.
- No "magic bullet" in general

Reading/Activity

- Please read pages 35-41 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.
- Read the outline of the practical activity in preparation for Wednesday's tutorials (issued today!).

The Link to Requirements

- Main activity in design here: decomposing 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 are harder to decompose, e.g. usability (e.g. It takes 30 minutes to become competent in using the system).
- We might require traceability back from the design

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 quality and evaluation
- Design notations
- Design strategies

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.

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 Issues in S/W Design

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

Key Issues (ctd)

- 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?

Quality Analysis and Evaluation

- Quality attributes:
  - Performance, security, availability, …
  - Modifiability, portability, reusability, testability, maintainability, …
- 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.

Architecture and Structure

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

Design Notations

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

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

Summary

- Design is a complex matter
- Generally two stages – architecture, detailed design
- Many notations and procedures to support design.
- More domain-specificity => easier design task
- Design links requirements to construction, essential to ensure tracability.

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: ie the orginal address is correct, and only the volunteer can effect a change in address.