CS5030

Software Design

Learning objectives

- On completing this lecture and associated reading, you should
  - Understand the motivation for software design and its place within the software lifecycle
  - Be aware of UML as a general-purpose graphical modelling language and the different aspects of a system that can be represented in it

Software design

- Follows requirements engineering and architecture design
- Closely linked with implementation
  - Influenced by programming paradigm, implementation concerns, etc
  - Can be used to generate implementation skeletons
- Moving from the ‘what’ towards the ‘how’
- Development process determines how much design is done up front
- Should be documented

Need for design

“Weeks of coding can save you hours of planning”

[unknown]

Software design process

- Define context and external interactions of the target system
- Design system and software architecture
- Identify the principal entities / constructs for implementation paradigm
- Develop and document design models
- Specify dependencies and interactions
- Evaluate design

Types of software design

[Martin Fowler]

- Planned
  - Significant issues are thought out in advance
  - Potentially inflexible and out of sync with technology and changing requirements
- Evolutionary
  - Design of system grows as the system is implemented
  - Can become a collection of ad-hoc decisions

Software design - considerations

- Design for change
- Use appropriate patterns
- Communicate the design
- Listen to development team
- Refactor often

Model

- “A simplified or idealized description or conception of a particular system, situation, or process, often in mathematical terms, that is put forward as a basis for theoretical or empirical understanding, or for calculations, predictions, etc.; a conceptual or mental representation of something” [OED]

- “All models are wrong, but some are useful”

  [George Box]

Models - uses

- Managing complexity
- Communicating with stakeholders
- Detecting errors and omissions early
- Guiding implementation
- Understanding and managing changes
- Planning work and resources

Modelling notations for software design

- Informal diagrams
  - Boxes and lines
- Semi-formal diagrams
  - UML, SysML
- Formal specifications
  - CSP, π-calculus, petri nets, Promela, …

UML

- Unified Modelling Language

- General-purpose graphical modelling language
  - A set of diagrams to represent different aspects of design
  - Particularly aimed at object-oriented design
  - Usable by humans and machines

- Became a standard in 1997
  - Regular revisions

UML diagrams

- Structure
  - Component, composite structure, deployment, package, class, object, profile
- Behaviour
  - Activity, state machine, use case
- Interaction
  - Communication, sequence, interaction overview, timing

UML diagrams and software development artefacts

- Requirements
  - Use case diagram
- Software architecture
  - Component diagram, activity diagram, deployment diagram, …
- Software design
  - Class diagram, sequence diagram, state machine diagrams, …

UML tools

- diagrams.net (formerly draw.io)
- Papyrus
- PlantUML
- Eclipse UML tools
- Lucidchart
- JetUML
- …

Key points

- Software design is a key artefact in software development and is closely linked with system implementation
- A model is an abstract view of a system
- A set of complementary models can be created to show different perspectives of a software system design
- UML is a graphical language used to model various artefacts of a software system, and particularly focuses on object-oriented design