L4 - Design during inception

CS3028 - Principles of Software Engineering

Ernesto Compatangelo

Department of Computing Science

4.1 Reminding past issues and mapping them to current topics

Where are we now?

Software development paradigms

⇒ The Unified Process (UP) paradigm

⇒ UP phases and UP disciplines (activities) within each phase

⇒ Inception (first UP phase)

⇒ Business modelling during inception

⇒ Requirements during inception

⇒ Design during inception

⇒ Transition from requirements analysis to design

⇒ Software architectures, subsystems, and packages

⇒ Architectural patterns

⇒ . . . . . .

4.2 From requirements to design

Why software design?

Analysis and business modelling organise the following information:

- A **domain model** that describes relevant concepts in the application area in terms of classes, class attributes, associations
- A **workflow model** that describes relevant activities in the application area in terms of sequences, iterations, and selections of elementary actions
- A list of **functional requirements**, describing system functionalities from a user viewpoint
- A list of **non-functional requirements** describing property ranges for these functionalities

Q: What is missing?

*A: **How** the software system should be devised!*

Design vs. analysis

- Software development is based on three different models at different abstraction levels, namely
  - *problem* model
  - *solution* model
  - *implementation* model
- The purpose of analysis is to figure out what the business needs are (the **what**, specified as a **problem model**).
- The purpose of design is to outline the components of a *system* that satisfies these needs (the **how**, specified as a **solution model**).
- The steps in both analysis and design phases are highly interrelated and may require much 'going back and forth'.

4.3 On software design

What is software design?

In the context of software, design is:

- **The transformation of an analysis model**
  (what, the problem as defined by the software requirements)
  into a synthesis model
  (how, the solution as defined by the software structure)

- **A problem-solving process** whose objective is to find and describe a way to implement the software requirements in terms of
  - Functionalities
  - Platform
  - Quality
  - Budgeting
  - Schedule

Design during inception

How to devise a software system through design

A software system is devised by designing:

- Its **architecture**, which focuses on:
  - the system decomposition in terms of subsystems
  - the identification of layers and partitions
  - adoption of standards
  - mutual dependencies
  - mappings to hardware
  - major policy decisions (control flow, access control, data storage)
  - responsibilities
- Its **detailed structure**, which focuses on:
  - *individual components* which conform to best practice (design patterns)
  - objects manipulated by the software
  - interactions among objects

4.4 Inception focus on architectural design

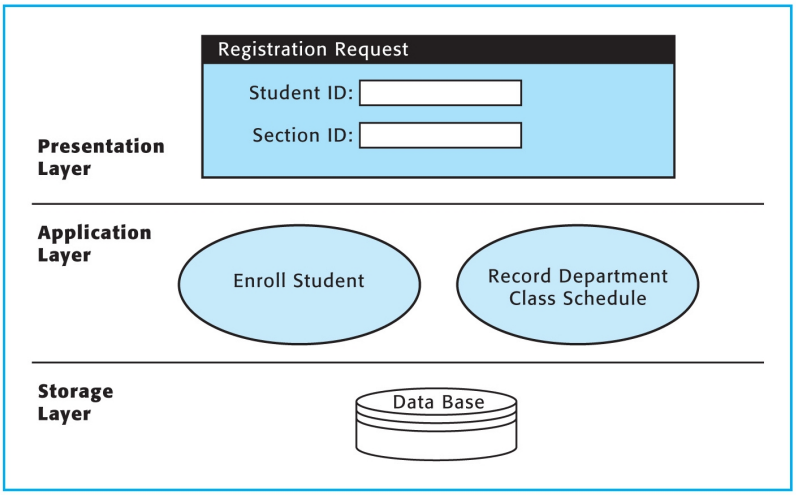
Architectural design

Architectural design encompasses significant decisions about:

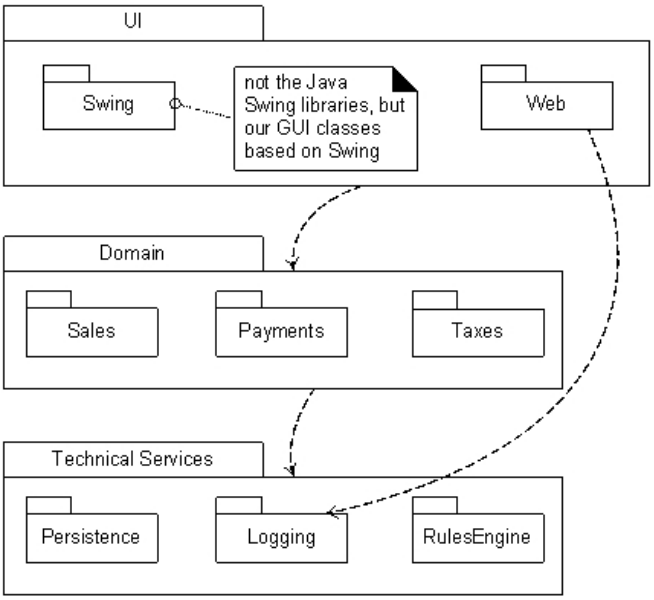
- **The organisation of a software system** into a synthesis model
- **The selection of structural elements and their interfaces** by which the system is composed, together with their behaviour as specified in the collaboration among these elements
- **The composition of these elements** into progressively larger subsystems
- **The architectural patterns** (aka styles) that guide this organisation and element interfaces, their collaborations, and their composition

The three-tier layered architectural pattern

Three-tier model of system architecture



Layering and partitioning with packages



Software architecture, subsystems, and packages

A software architecture is a description of the **subsystems** that compose a software system and of their **relationships**.

Subsystems (often represented as **architectural packages**)

- Group together system elements that share **common properties** (*e.g.*, user interface, data management. . .)
- Result in **smaller development units**, helping developers to cope with complexity
- **Maximise reuse** at component level, improving **maintainability** and **portability**
- Have clearly specified **boundaries** and fully defined **interfaces** with other sub-systems
- Communicate following a **client-server** or **peer-to-peer** architecture

A word on architectural patterns

- Architectural patterns (aka architectural *styles*) outline reusable design solutions to specific problems, abstracting from their concrete form which keeps recurring in different contexts
- They are general models used as a starting point for system design
- Architectural patterns define system decomposition, global control flow, handling of boundary conditions, inter-subsystem communication
- Architectural patterns are based on the two fundamental notions of **layering** (where sub-systems represent different levels of abstraction) and **partitioning** (where sub-systems represent different functional aspects of the system)

4.5 Preparing for the topics ahead

Next week...

Other relevant inception disciplines

More specifically, we will focus on:

- Software project management (who does what when?)
- Software estimating and planning (task duration and cost)
- Agile and XP approaches as a lightweight UP paradigm