SOFT2201 introduction to software engineering and modelling

Week 1

**Agenda**

– **Software Engineering**

– **Why, What**

– **Software Modeling**

– **What, Why and How**

– **The Unified Process**

**Software engineering**

**Why**:

* Need to build **high-quality software** systems under **resource constraints**
* Social: user need, convenience
* Economical: cost
* Time: deliver of product

**What**:

* Accidents: difficulties related to the production of software
* Essences: difficulties inherent (or intrinsic) in the nature of software

• **Complexity**

• Many diverse software entities - interactions increase exponentially

• Intrinsic complexity cannot be abstracted - aircraft software, air traffic control

• **Conformity**

• Arbitrary changes from environment (people, systems) - no unifying principle

• **Changeability**

• Changing a building model vs. a software

• Stakeholders understanding of software changes

• **Invisibility**

• Software is intangible (invisible)

• Building model vs software models (UML - 13 diagram types)

* Engineers use Theories, methods, tools, techniques and approaches to manage the project, make good product
* Engineers process for managing and developing, account Dependability, Performance and costs.

Software Design/Modelling & Construction

**What**: Software Modelling

• The process of **developing conceptual models of a system at different levels of abstraction**

• Fulfil the defined **system requirements**

• Focus on important system **details to deal with complexity**

• **Object-oriented** design approach

• Concepts in the domain problem are modelled as interacting objects

• Using **graphical notations** (e.g., UML) to capture different views or perspectives

**WHY**: **Abstraction**

• Conceptual process to **derive general rules and concepts from concrete information in a certain context**

• Analysis and understanding of domain-specific problems

• **Decompose large problems into smaller** understandable piece

• software models with **different views/perspective**s

• Time, structural, requirements, etc.

**HOW: UML Principles**

• Graphical notations to visually specify and document design artifacts software systems using OO concepts

• Combines techniques from data modeling (ER diagrams), business modeling (workflows), object and component modeling

• Capture system activities (jobs), system components and its interaction, software entities and its interaction, run-time behavior, external interfaces

* + UML require design knowledge and thinking in objects

• Three ways to apply UML

• Sketching to explore part of the problem or solution space

• Blueprint: detailed design diagrams for:

• Reverse engineering to visualize and understand existing code

• Forward engineering (code generation)

• Programming language (Model Driven Engineering): executable specification of a system

**The Rational Unified Process(UP)**

* UP: an **iterative and Evolutionary** Development
* Agile: a practice of UP

Software Construction / Implementation

• Realization of design to produce a working software system

• Meet **customer requirements**

• Design and implementation activities often interleaved

• **Agile** development to **accommodate for changes**

• **Object-Oriented** design and Implementation model

• **Encapsulation**

• **Abstraction**

• **Reuse**

• **Maintenance**