Project Management

Massimo Felici

Room 1402, JCMB, KB

0131 650 5899

mfelici@inf.ed.ac.uk

Project Management

- Software project management is an essential part of software engineering
- The Good, The Bad and The Ugly
  - **Good** management cannot guarantee project success
  - **Bad** Management usually results in project failures
    - Software is delivered late, costs more than originally estimated and fails to meet its requirements …**The Ugly**
- Software managers are responsible for **planning** and **managing** project development

The Role of the Project Manager

- **Estimation** of the project effort, time and cost
- **Planning**. Scheduling deliverables, review points and allocation of staff to activities
- **Replanning**. Re-estimating and rescheduling in the light of unfolding circumstances, e.g., risks and quality assurance results
- **Organization**. Establishing a division of labour which is able to make the most effective use of available skills and maximizes productivity potential in the context of characteristics (e.g., risk factors) of the particular project
- **Quality assurance**. Planning and carrying out actions to ensure that the software product meets required quality targets

Project Planning

- Effective management of a software project depends on thoroughly **planning the progress** of the project. Managers must **anticipate problems** that might arise and prepare tentative solutions to those problems. A plan, drawn up at the start of a project, should be used as the driver of the project. This initial plan should be the best possible plan given the **available information.** It **evolves** as the project progresses and further information becomes available.
- A good project plan includes the following items: Project **scope**; Project **schedule**; Project team **organization**; Technical description of the proposed **system**; Project **standards**, **procedures**, and proposed **techniques** and **tools**; **Quality** assurance plan; **Configuration** management plan; **Documentation** plan; Data management plan; **Resource** management plan; **Test** plan; **Training** plan; **Security** plan; **Risk** management plan; **Maintenance** plan; etc.

Scoping the Problem

- **Objectives** expressed in general terms and in the language application domain
- **Scope** defines the system boundary, explaining what will be included in the system and what will not be included
- **Identify**: the Customer, the system environment, necessary tools, potential reuse, etc.
  - Ask the Customer: Who is the end user? (often not the customer) Who has the authority to accept the finished product? What problem are we addressing? What documentation will be required? When do they believe they need the product? Where is the work to be dune? Why do they need the product? How will the product be developed/acquired?

Other Management Activities…

- **Measurement Framework** allows the quantitative analysis of project (e.g., productivity, progress, etc.) and product features (e.g., quality, size, etc.)
  - **Software Metrics.** Measurement is the process by which numbers or symbols are assigned to attributes of entities in the real word in such a way as to describe them according to clearly defined rules.
  - **Quality Assurance plan** describes how reviews, inspections, testing, and other techniques will help to evaluate quality and ensure that it meets the customer's needs.
- **Resource management** identifies (and quantifies) the (needed) resources and describes how resources are allocated throughout the project
  - Resources include infrastructure, staff and time.
- **Feasibility study** also explores alternative solutions

Tracking Progress and Control

- **Scheduling** explores possible ways of allocating (limited) resources across tasks
- **Project scheduling** involves separating the total work involved in a project into separate activities and judging the time required to complete these activities.
- Project can be late with respect to the initial plan. It is important to **track the progress** of the project and compare it to the plan. If significant divergences arise it is necessary to re-plan to take account of the changed circumstances.

An Example of Project Workplan

Project Activity	Duration (in Trimesters)							
	1	2	3	4	5	6	7	8
Project Milestone		**M1**		**M2**		**M3**		**M4**
T1								
PA 1.1								
PA 1.2								
T 2								
PA 2.1								
PA 2.2								
PA 2.3								
T3								
PA 3.1								
PA 3.2								
PA 3.3								
PA 3.4								
T 4								
PA 4.1								
PA 4.2								

Risk Management

- Project managers must engage in **risk management** to understand and control the risks on their projects
- A **Risk** is an unwanted event that has negative consequences
  - **Risk impact**: the loss associated with the event
  - **Risk probability**: the likelihood of the risk, measured from 0 (impossible) to 1 (certainty)
  - **Risk control**: the degree to which we can change the outcome
- **Risk Management** involves: Risk **Identification**, Risk **Analysis**, Risk **Planning** and Risk **Monitoring**
  - **Risk Identification** concerns with discovering possible risks to the project
  - **Risk Analysis** considers each identified risk and makes a judgment about the probability and seriousness of it
  - **Risk Planning** considers each identified risk and identifies strategies to manage the risk
  - **Risk Monitoring** involves regularly assessing each identified risk to decide whether that risk is becoming more or less probable and whether the effect of the risk have changed

Boehm's Top Ten Risk Items (1991)

1. Personnel shortfalls
2. Unrealistic schedules and budgets
3. Developing the wrong software functions
4. Developing the wrong user interface
5. Gold plating
6. Continuing stream of requirements changes
7. Shortfalls in externally performed tasks
8. Shortfalls in externally furnished components
9. Real-time performance shortfalls
10. Straining computer science capabilities

Project Personnel

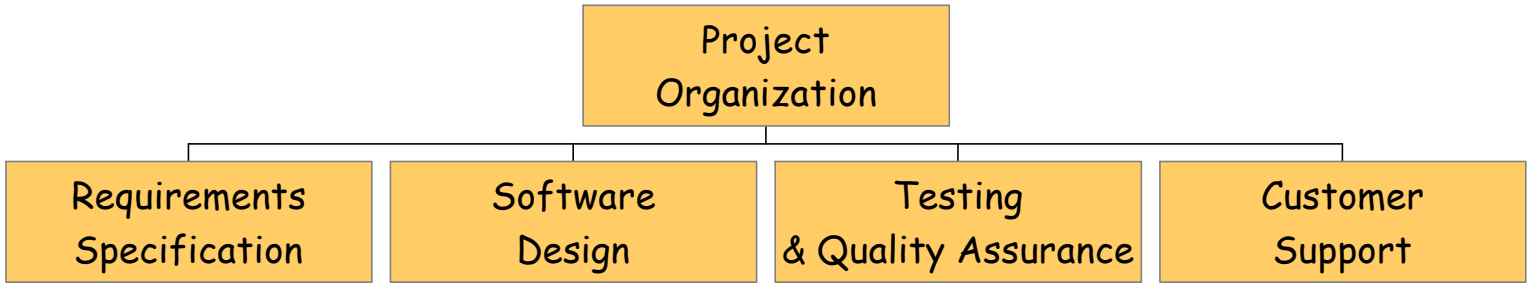
- Determine the project schedule and estimate the associated effort and costs
  - How many people will be involved in the project
  - What tasks they will perform
  - What abilities and experience they must have so that they can do their job effectively
- The assignment of staff to tasks depends on project size, staff expertise and staff experience
- People have different work styles (e.g., preferred styles for interacting with others)

Project Organization

- Team members are organized in ways that enhance the completion of quality products
- The choice of an appropriate structure for your project depends on several things
  - The backgrounds and work styles of the team members
  - The number of people on the team
  - The management styles of the customers and developers
- Comparison of Organizational Structures: Highly or Loosely Structured; High Certainty of Uncertainty; Repetition or New techniques (or Technology); Large or Small Projects
- Examples of Organizations
  - Functional
  - Matrix
  - Integrated Product Development Teams (IPDTs)

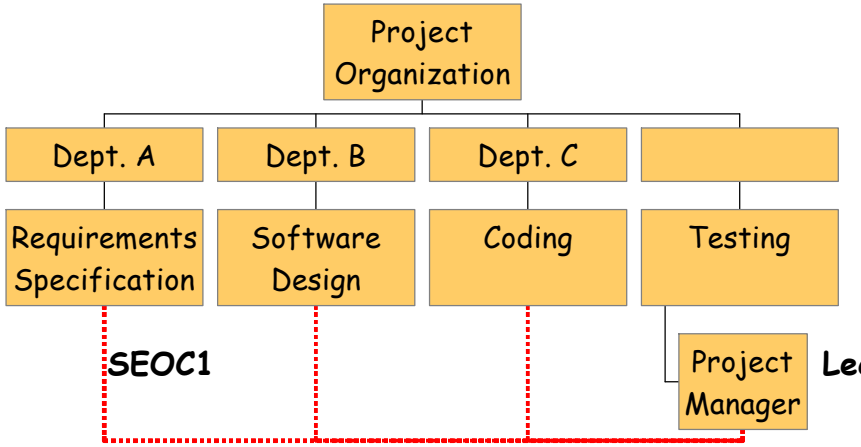
Project Organization: Functional

- Basic hierarchical organization
- Project organized by disciplines and functions
- **Characteristics**: Narrow set of work methods, deep technical expertise, Develops skills and morale; Service-oriented, Communication responsibility on group manager
- **Problems**: Elitism within expertise areas, Communication difficult, no project "ownership"



Project Organization: Matrix

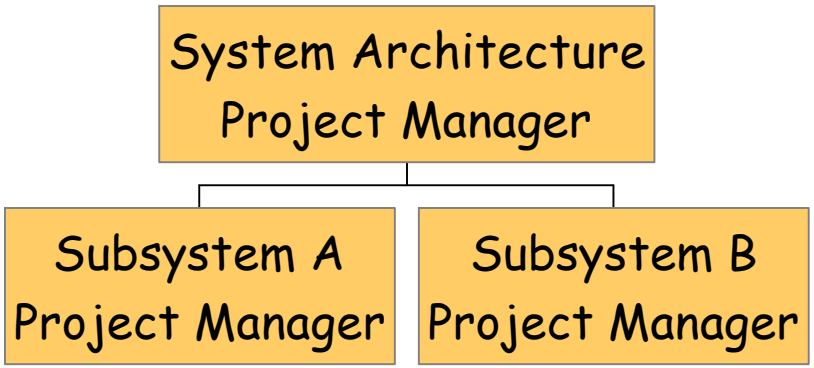
- Based on a specific project; Experts are borrowed, but not removed
- **Strong Matrix:** team leader is the principal authority, Control of schedule and budget, Acquire personnel, Perform reviews
- **Weak Matrix:** team leader is only a coordinator, Spokesperson to higher management, Steering committee has ultimate authority



- **Characteristics**: Specialists work on part-time basis for several projects, Top management selects project manager and staff Good for short-lived projects "Task force" Mentality
- **Problems**: Staff attention fractured Conflicting obligations Large amount of communication Strong top management involvement; Reporting to home "base" is difficult

Project Organization: IPDT

- Single, long-term project; Organized by component
- Combining individuals from different functional groups into an interdisciplinary work unit



- **Characteristics**: Tightly controlled effort, Complex or large project, Independent authority for sub-managers, Direct contact with customer, Reporting is easy
- **Problems**: Loss of project – what to do with staff?, Difficult to enforce standards, Overspecialization

Reading/Activity

- Chapter 8 (Software Engineering Management) and 9 (Software Engineering Process) of the SWEBOK

Secure Coursework Submission System

Massimo Felici

Room 1402, JCMB, KB

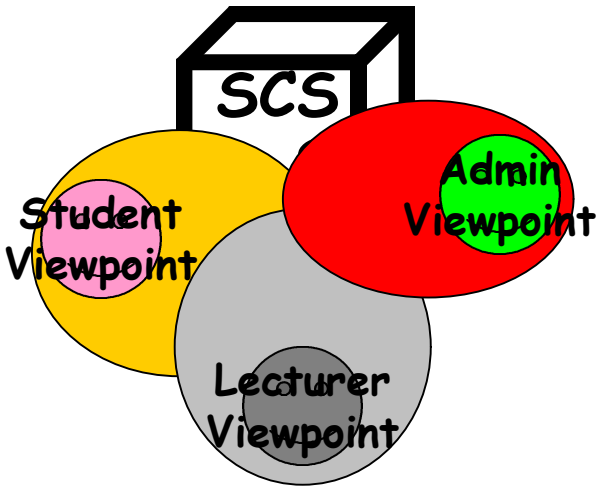
0131 650 5899

mfelici@inf.ed.ac.uk

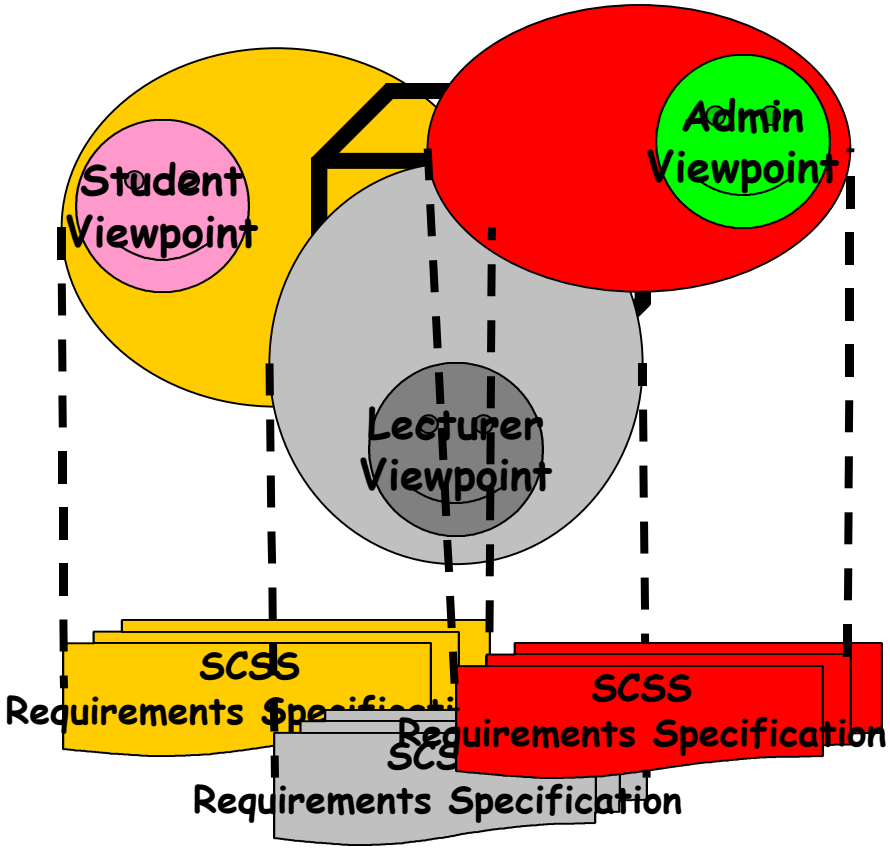
Requirements Viewpoints

- What are your system requirements?
  - You are looking at the same system from different viewpoints
  - Requirements viewpoints highlight system requirements (i.e., your perspective understanding of the system)
- Requirements Viewpoints:
  - Student viewpoint (S)
  - Lecturer viewpoint (P)
  - Admin viewpoint (A)
- Why viewpoints?
  - Requirements completeness
  - Partial specifications
  - Viewpoints highlight requirements association, hence Traceability

- What is a viewpoint?
  - Encapsulation of partial information about system requirements from a particular perspective
- **Main Issues:** Difficult to Identify a stable set of requirements; Contradicting viewpoints



From Requirements Viewpoints to Requirements Specification(s)



- Identify
  - high level requirements
  - the System Stakeholders
  - The scope of the system
- Capture requirements from your viewpoint (i.e., S, P or A)

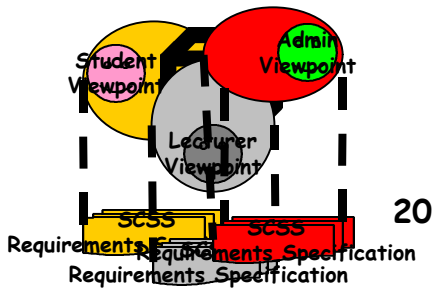
Document your Requirements

- Use a Requirements Specification template

1. **Overall Objectives**

   "The system has to support teaching and administrative activities related to the courses' assessments. The system should support the secure submission of course exams and solutions."

2. **Functional Requirements**
   1. "The system should be integrated with DICE."
   2. "The system should allow the submission of practicals and exams.
   3. "The system supports logins from different locations."
   4. …
3. **Non-Functional Requirements**
   1. "The System should guarantee secure submissions"
   2. …
4. **Others**



UML Modelling

5. Use Cases

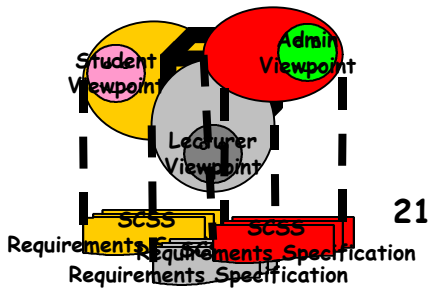
- Capture your "functional" requirements" graphically
- Define the system boundaries
- Generalizing and structuring let you simplify use cases
- Use a Template to write each use case

6. Class Diagrams

- Some class may have missing information, because you don't yet know all the "responsibilities" and the "collaborators"

7. Validation by CRC Cards

- Run few use cases (point 4) to validate your class diagrams (and use cases too)
- Pick up significant use cases
- Pick up also use cases involving different Actors (e.g., S and P, S and A, P and A)



Deliverable 1 Assessment

- **Completeness**

  Deliverable 1 consists of 1-7

- **Coverage (Requirements Completeness)**
  - The "basic" functionalities should be covered
- **Quality (of the design)**
  - UML and OO Design Proficiency
  - Software Engineering practice (e.g., use of templates)
- **Productivity**
  - Taking into account team size