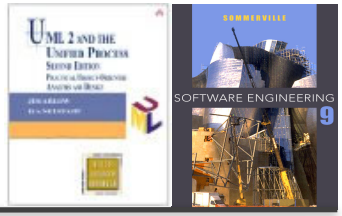
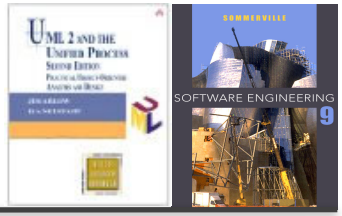


Lecture 11

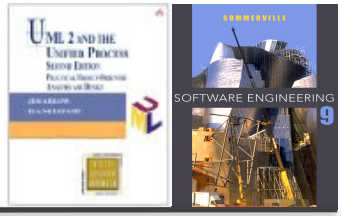
SOFTWARE DEVELOPMENT MANAGEMENT

PB007 Software Engineering I
Faculty of Informatics, Masaryk University
Fall 2017

Topics covered



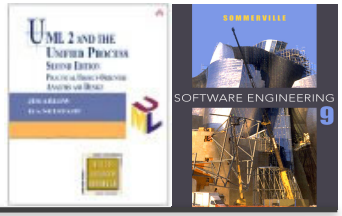
- ✧ Project management
- ✧ Project planning
- ✧ Risk management
- ✧ People management



Project Management

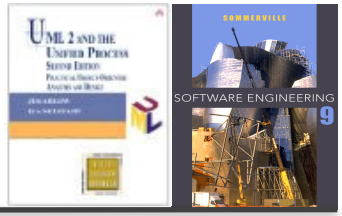
Lecture 11/Part 1

Software project management



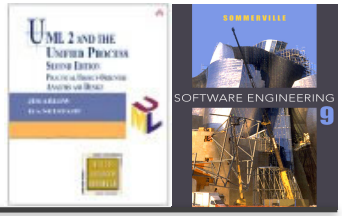
- Concerned with activities involved in ensuring that software is delivered **on time** and **within budget** and in accordance with the organisations developing and procuring the software.
- Success criteria:
  - Deliver the software to the customer at the agreed **time**.
  - Keep overall costs within **budget**.
  - Deliver software that meets the **customer's expectations**.
  - Maintain a **happy** and well-functioning **development team**.

Software management distinctions



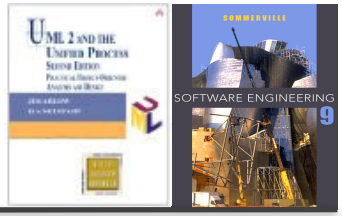
- The product is intangible.
  - Software cannot be seen or touched. Software project managers cannot see progress by simply looking at the artefact that is being constructed.
- Many software projects are 'one-off' projects.
  - Large software projects are usually different in some ways from previous projects. Even managers who have lots of previous experience may find it difficult to anticipate problems.
- Software processes are variable and organization specific.
  - We still cannot reliably predict when a particular software process is likely to lead to development problems.

Management activities

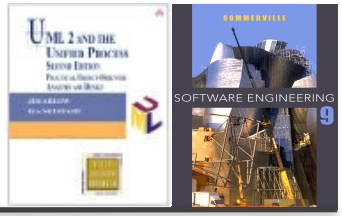


- Project planning
  - Project managers are responsible for **planning, estimating and scheduling** project development and assigning **people to tasks**.
- Risk management
  - Project managers **assess the risks** that may affect a project, **monitor** these risks and **take action** when problems arise.
- People management
  - Project managers have to **choose people** for their team and establish ways of working that leads to **effective team performance.**

Management activities



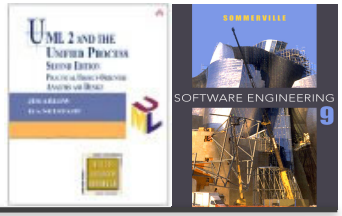
- Reporting
  - Project managers are usually responsible for **reporting on the progress** of a project to customers and to the managers of the company developing the software.
- Contract negotiation
  - The first stage in a software project may involve writing a **proposal to win a contract** to carry out an item of work. The proposal describes the objectives of the project and how it will be carried out.
  - Then the contract is negotiated and later **extended** with requirements changes and changing schedule constraints.



Project Planning

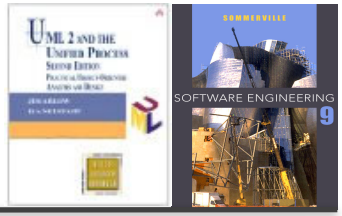
Lecture 11/Part 2

Project planning



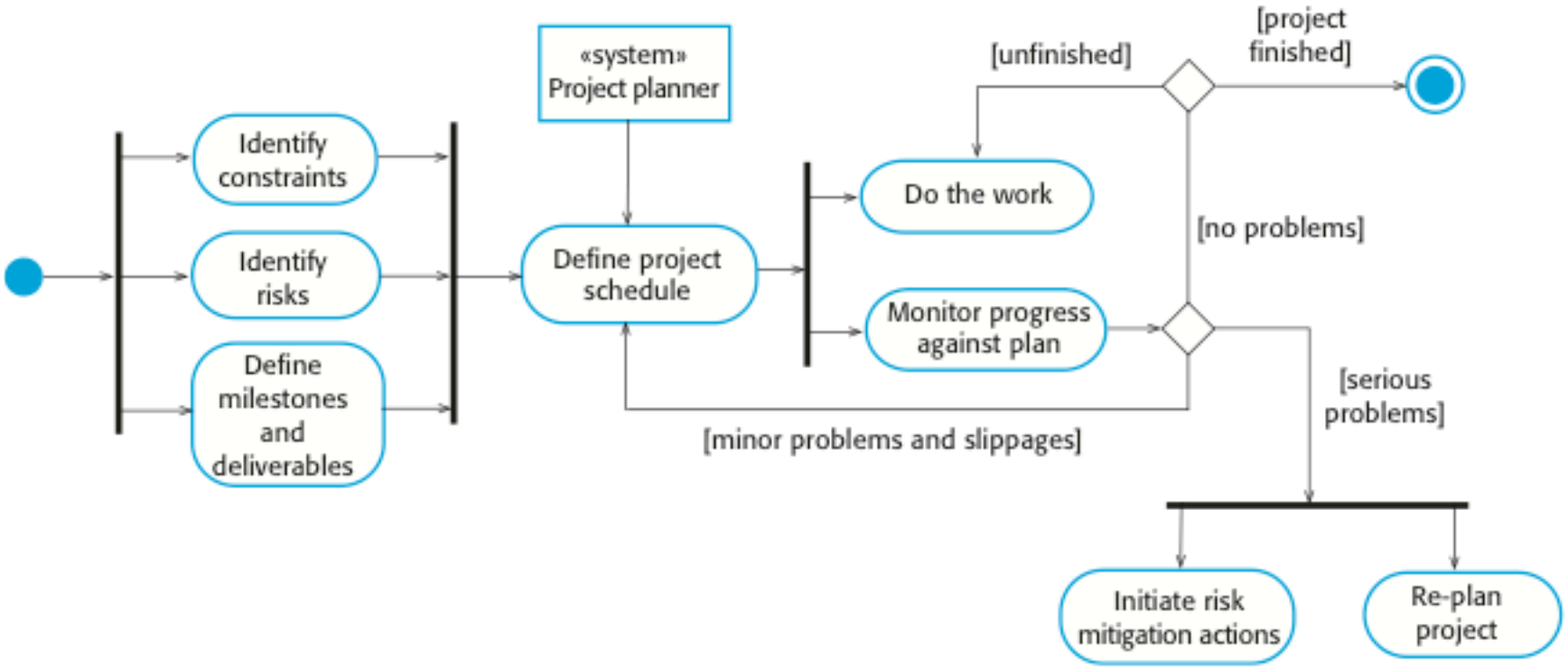
- Project planning involves **breaking down the work** into parts and **assign** these to project team members, anticipate problems that might arise and prepare tentative solutions to those problems.
- The **project plan**, which is created at the start of a project, is used to communicate how the work will be done to the project team and customers, and to help assess progress on the project.

Plan-driven development

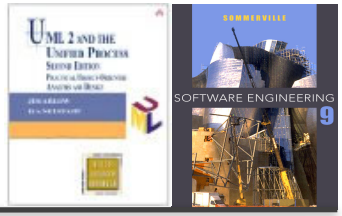


- Plan-driven or plan-based development is an approach to software engineering where the **development process is planned in detail**.
  - Plan-driven development is based on engineering project management techniques and is the 'traditional' way of managing large software development projects.
- A **project plan** is created that records the **work** to be done, **who** will do it, the development **schedule** and the work **products**.
- Managers use the plan to support project decision making and as a way of **measuring progress**.

The project planning process

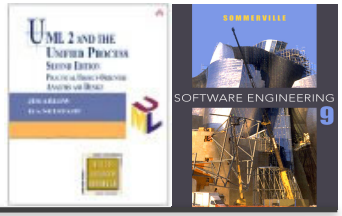


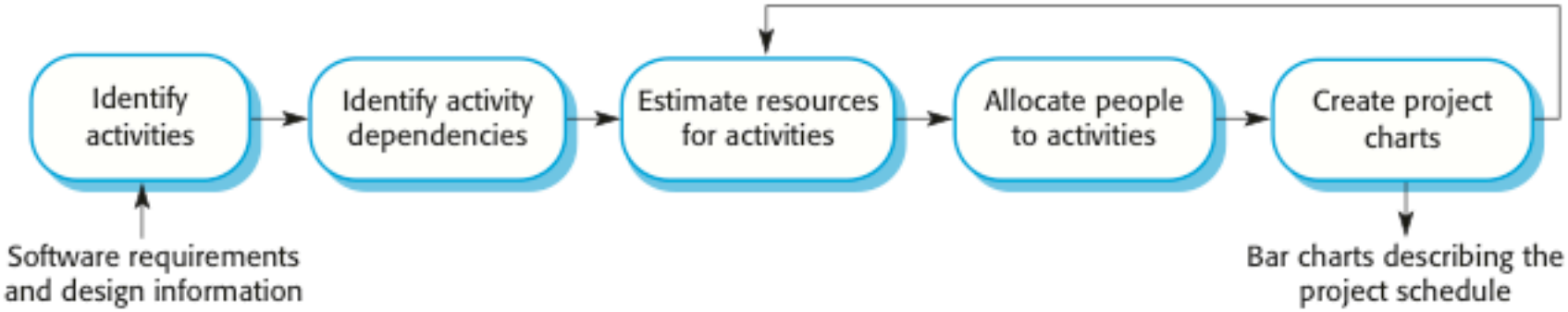
Project scheduling



- Project scheduling is the process of deciding how the work in a project will be **organized as separate tasks**, and when and how these tasks will be executed.
- You estimate the calendar **time** needed to complete each task, the **effort** required and **who** will work on the tasks that have been identified.
- You also have to estimate the **resources** needed to complete each task, such as the disk space required on a server, the time required on specialized hardware, such as a simulator, and what the travel budget will be.

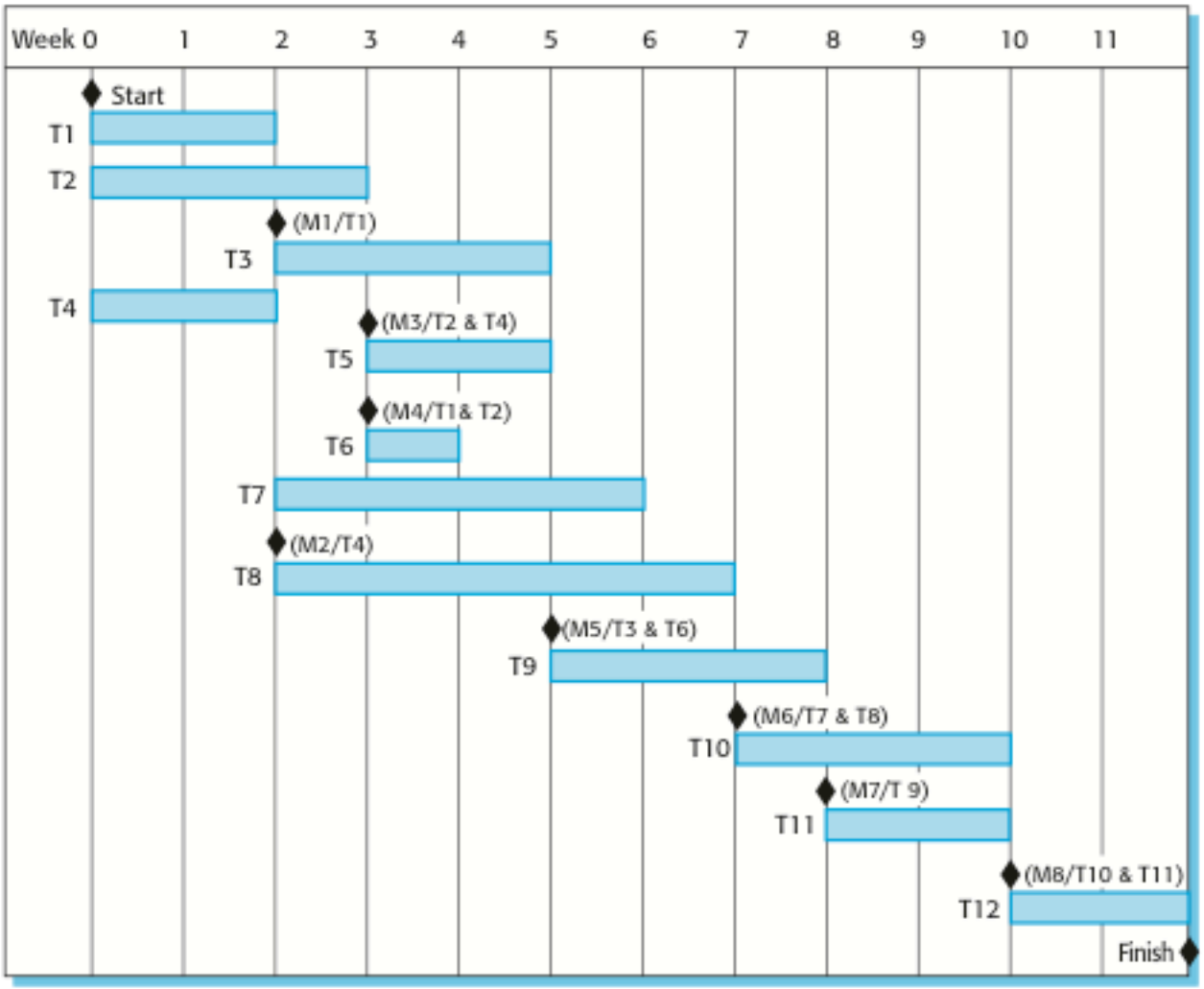
The project scheduling process



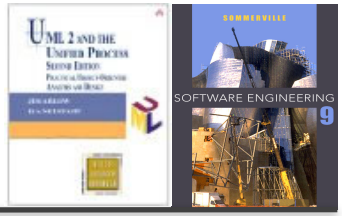


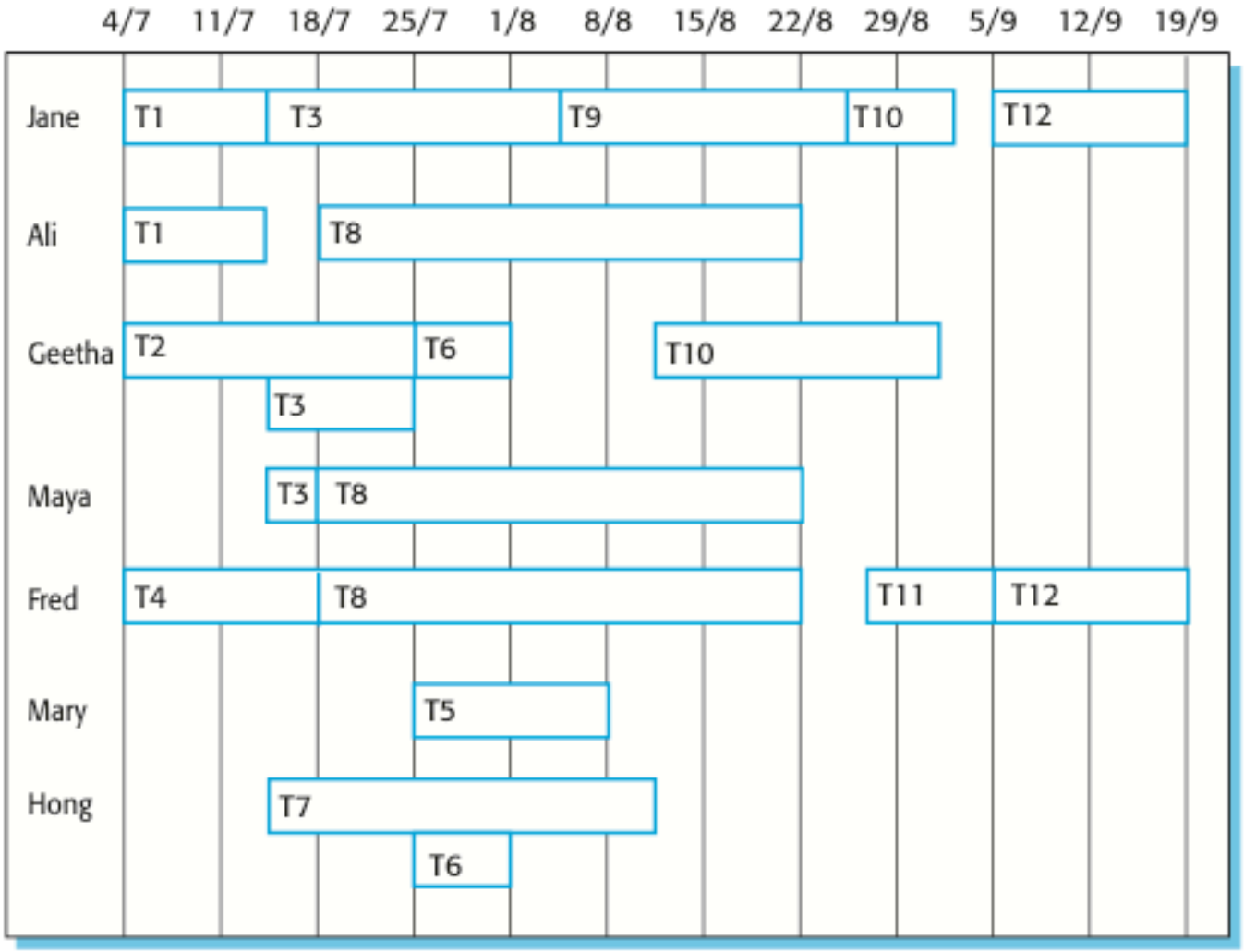
- Split project into tasks, which are not too small (about week or two).
- Organize tasks concurrently to make optimal use of workforce.
- Minimize task dependencies to avoid delays caused by one task waiting for another to complete.
- Use graphical notation to visualise and manage project schedule.

Schedule representation – Gantt chart

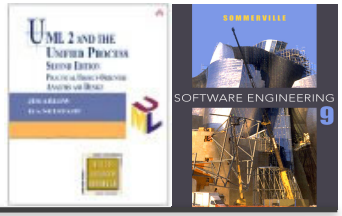


Schedule representation – Staff allocation chart



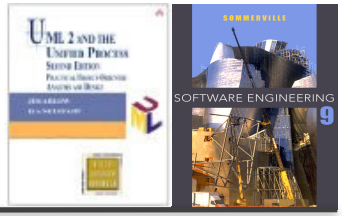


Scheduling problems



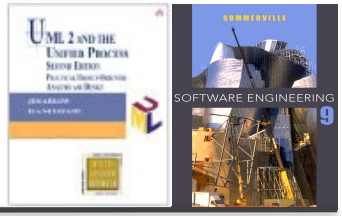
- **Estimating the difficulty** of problems and hence the cost of developing a solution is hard.
- **Productivity is not proportional** to the number of people working on a task.
- **Adding people to a late project** often makes it later because of communication overheads.
- **The unexpected always happens**. Always allow contingency in planning.

Agile planning



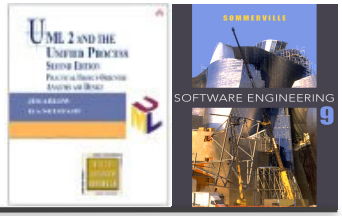
- Agile methods of software development are **iterative approaches** with **incremental delivery**.
- Unlike in plan-driven approaches, the **functionality of these increments is not planned** in advance but is decided during the development.
  - The customer's **priorities and requirements** change so it makes sense to have a flexible plan that can accommodate these changes.
- While plan-driven approaches work with **fixed functionality** and decide on the time plan accordingly, agile approaches have **fixed time plan** (e.g. weekly sprints) and decide on the functionality accordingly.

Software pricing



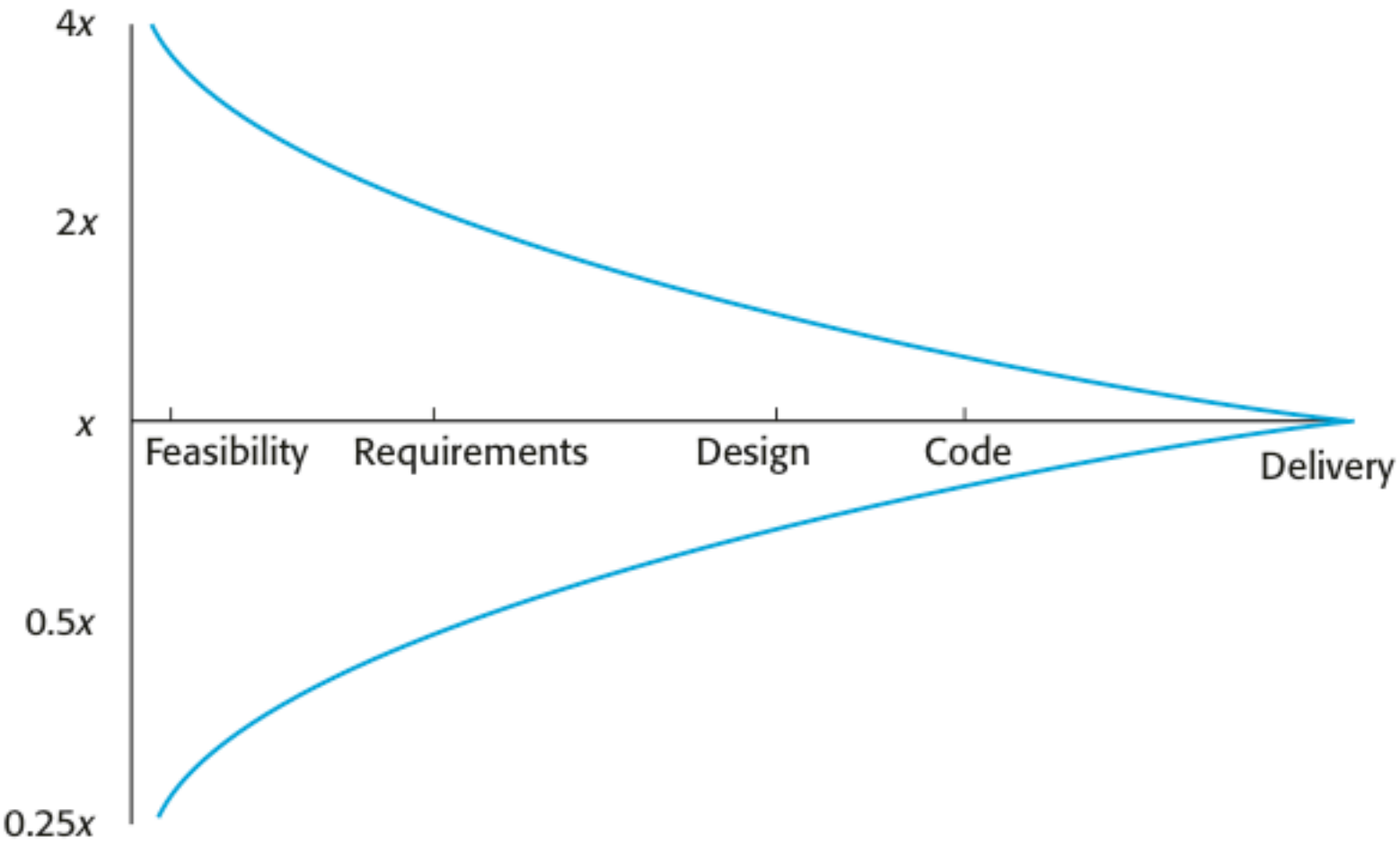
- Estimates are made to discover the cost, to the developer company, of producing a software system.
  - You take into account hardware, software, travel, training, effort and other costs.
  - Both **fixed and variable costs** need to be considered.
- There is not a simple relationship between the development cost and the price charged to the customer.
- Broader **organisational**, **economic**, **political** and **business** considerations influence the price charged.

Factors affecting software pricing

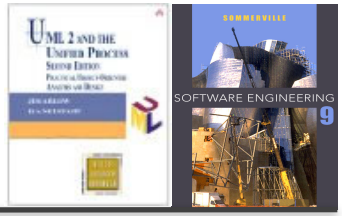


Factor	Description
Market opportunity	A development organization may quote a low price because accepting a low profit on one project may give the organization the **opportunity to make a greater profit later**. The experience gained may also help in future products.
Cost estimate uncertainty	If an organization is unsure of its cost estimate, it may increase its price by a contingency over and above its normal profit.
Contractual terms	A customer may be willing to allow the developer to **retain ownership of the source code** and reuse it in other projects. The price charged may then be less than if the software source code is handed over to the customer.
Financial health	Developers in financial difficulty may lower their price to gain a contract. It is better to make a smaller than normal profit or break even than to go out of business. **Cash flow is more important than profit in difficult economic times.**

Cost estimate uncertainty

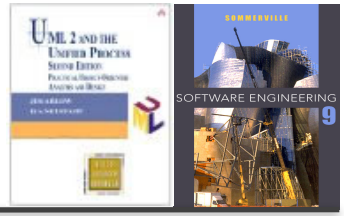


Estimation techniques



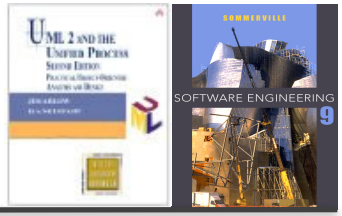
- Organizations need to make software effort and cost estimates. There are two types of technique that can be used to do this:
  - **Experience-based techniques** The estimate of future effort requirements is based on the manager's experience of past projects and the application domain. Essentially, the manager makes an informed judgment of what the effort is likely to be.
  - **Algorithmic cost assessment** In this approach, a formulaic approach is used to compute the project effort based on estimates of product attributes, such as size, and process characteristics, such as experience of staff involved.

Experience-based approaches



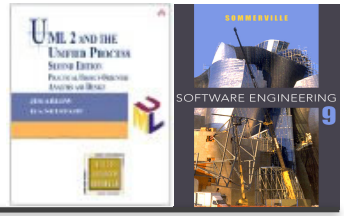
- Experience-based techniques rely on judgments based on **experience of past projects** and the effort expended in these projects on software development activities.
- Typically, you **identify the deliverables** to be produced in a project and the different software components or systems that are to be developed.
- You document these in a spreadsheet, **estimate them individually and compute the total** effort required.
- It usually helps to get **a group of people involved** in the effort estimation and to ask each member of the group to explain their estimate.

Algorithmic cost assessment



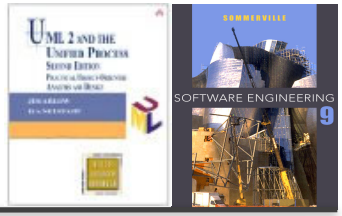
- Cost is estimated as a **mathematical function** of product, project and process attributes whose values are estimated by project managers:
  - Effort = $(A * Size^B) * M$
  - A is an organisation-dependent constant, B reflects the disproportionate effort for large projects and M is a multiplier reflecting product, process and people attributes.
- The most commonly used product attribute for cost estimation is **code size**.
- Most models are similar but they use different values for A, B and M.

Estimation accuracy



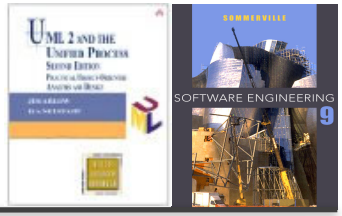
- The **size of a software system** can only be known accurately when it is finished.
- Several factors influence the final size
  - Use of COTS and components;
  - Programming language;
  - Distribution of system.
- As the development process progresses then the size estimate becomes more accurate.
- The estimates of the factors contributing to B and M are subjective and vary according to the judgment of the estimator.

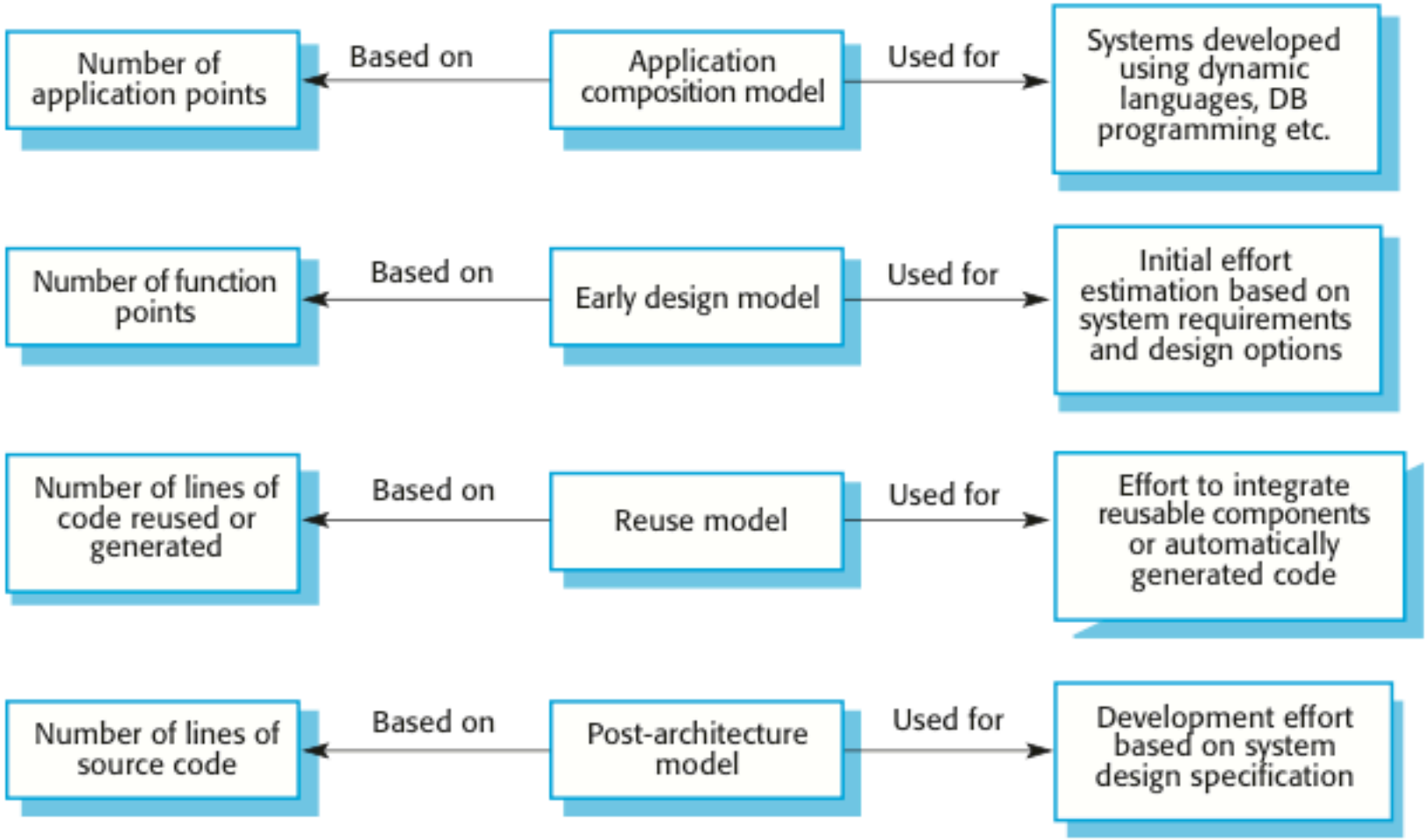
The COCOMO II model



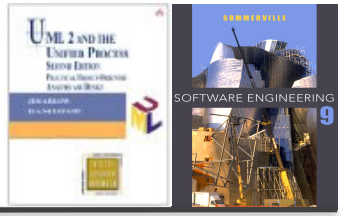
- An empirical model based on project experience.
- Well-documented, 'independent' model which is not tied to a specific software vendor.
- Long history from initial version published in 1981 (COCOMO-81) through various instantiations to COCOMO II (published in 2000).
- COCOMO II takes into account different approaches to software development, reuse, etc.
- COCOMO II incorporates a range of sub-models that produce increasingly detailed software estimates.

COCOMO II estimation models



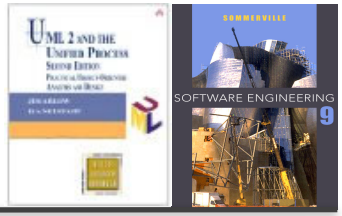


COCOMO II factors

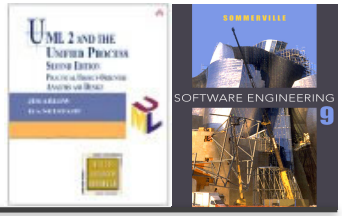


	Ratings					
Cost Drivers	Very Low	Low	Nominal	High	Very High	Extra High
Product attributes						
Required software reliability	0.75	0.88	1.00	1.15	1.40	
Size of application database		0.94	1.00	1.08	1.16	
Complexity of the product	0.70	0.85	1.00	1.15	1.30	1.65
Hardware attributes						
Run-time performance constraints			1.00	1.11	1.30	1.66
Memory constraints			1.00	1.06	1.21	1.56
Volatility of the virtual machine environment		0.87	1.00	1.15	1.30	
Required turnabout time		0.87	1.00	1.07	1.15	
Personnel attributes						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Software engineer capability	1.42	1.17	1.00	0.86	0.70	
Virtual machine experience	1.21	1.10	1.00	0.90		
Programming language experience	1.14	1.07	1.00	0.95		
Project attributes						
Application of software engineering methods	1.24	1.10	1.00	0.91	0.82	
Use of software tools	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

Key points



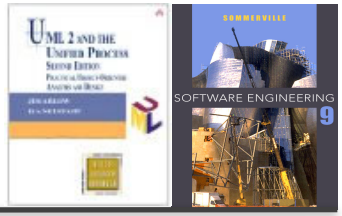
- Plan-driven development is organized around a complete **project plan** that defines the project **activities**, the planned **effort**, the activity **schedule** and **who** is responsible for each activity.
- Project **scheduling** involves the creation of graphical representations the project plan. Bar charts show the activity duration and staffing timelines, are the most commonly used schedule representations.
- The **price** charged for a system does not just depend on its estimated development costs; it may be adjusted depending on the market and organizational priorities.
- The **COCOMO II** costing model is an algorithmic cost model that uses project, product, hardware and personnel attributes as well as product size and complexity attributes to derive a cost estimate.



Risk Management

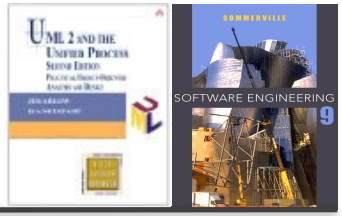
Lecture 11/Part 3

Risk management



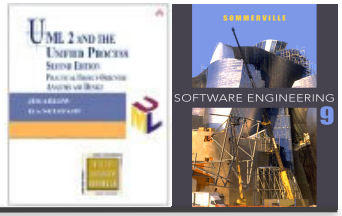
- Risk management is concerned with **identifying risks** and drawing up plans to **minimise their effect** on a project.
- A risk is a probability that some adverse circumstance will occur
  - **Project risks** affect schedule or resources;
  - **Product risks** affect the quality of the software being developed;
  - **Business risks** affect the organisation developing or procuring the software.

Examples of common project, product, and business risks



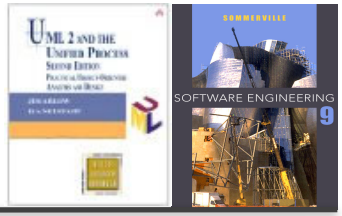
Risk	Affects	Description
Staff turnover	Project	Experienced staff will leave the project before it is finished.
Management change	Project	There will be a change of organizational management with different priorities.
Hardware unavailability	Project	Hardware will not be delivered on schedule.
Requirements change	Project and product	There will be a larger number of changes to the requirements than anticipated.
Specification delays	Project and product	Specifications of essential interfaces are not available on schedule.
Size underestimate	Project and product	The size of the system has been underestimated.
CASE tool underperformance	Product	CASE tools, which support the project, do not perform as anticipated.
Technology change	Business	The underlying technology on which the system is built is superseded by new technology.
Product competition	Business	A competitive product is marketed before the system is completed.

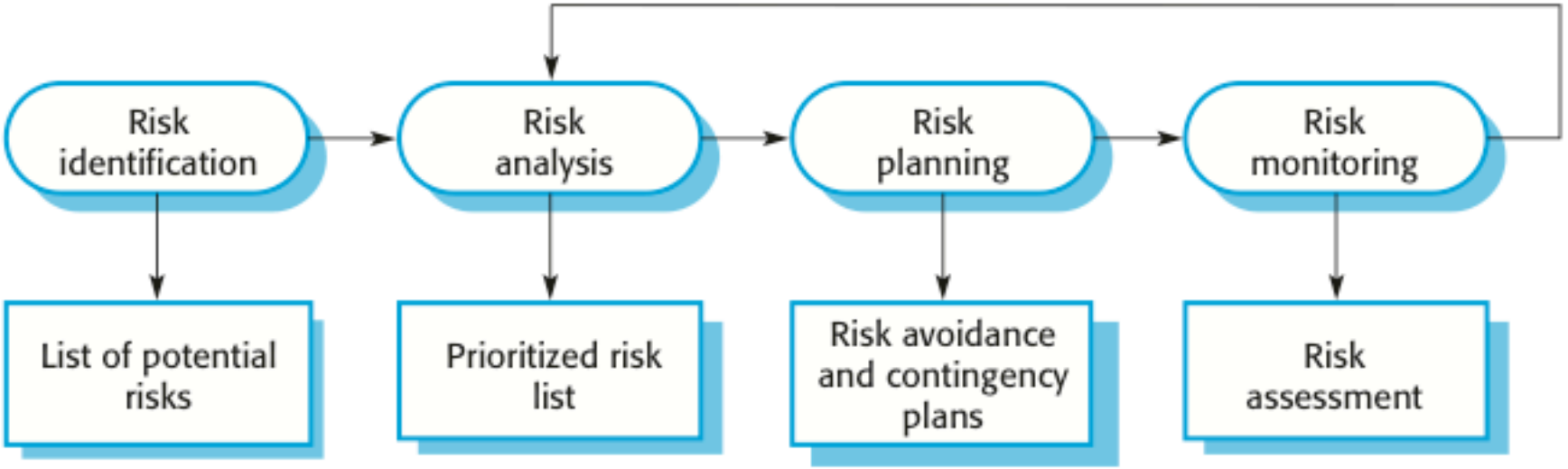
Fine-grained risk types and their examples



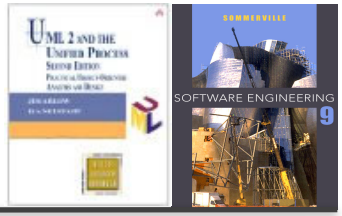
Risk type	Possible risks
Technology	The database used in the system cannot process as many transactions per second as expected. (1) Reusable software components contain defects that mean they cannot be reused as planned. (2)
People	It is impossible to recruit staff with the skills required. (3) Key staff are ill and unavailable at critical times. (4) Required training for staff is not available. (5)
Organizational	The organization is restructured so that different management are responsible for the project. (6) Organizational financial problems force reductions in the project budget. (7)
Tools	The code generated by software code generation tools is inefficient. (8) Software tools cannot work together in an integrated way. (9)
Requirements	Changes to requirements that require major design rework are proposed. (10) Customers fail to understand the impact of requirements changes. (11)
Estimation	The time required to develop the software is underestimated. (12) The rate of defect repair is underestimated. (13) The size of the software is underestimated. (14)

The risk management process





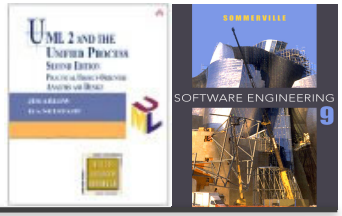
- ✧ Risk analysis
  - Assess the likelihood and consequences of these risks;
- ✧ Risk planning
  - Draw up plans to avoid or minimise the effects of the risk;



People Management

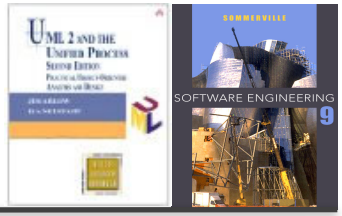
Lecture 11/Part 4

Managing people



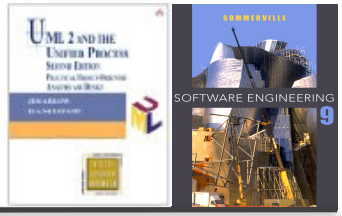
- People are an organisation's **most important assets**.
  - Especially in IT where the developer company does not need to invest into expensive **input material** (like in production companies).
  - Most of the **input investments are into people** either directly (salaries) or indirectly (tools increasing people's productivity, working environment, etc.).
- The tasks of a manager are essentially **people-oriented**. Unless there is some understanding of people, management will be unsuccessful.
- Poor people management is an important contributor to **project failure**.

People management factors



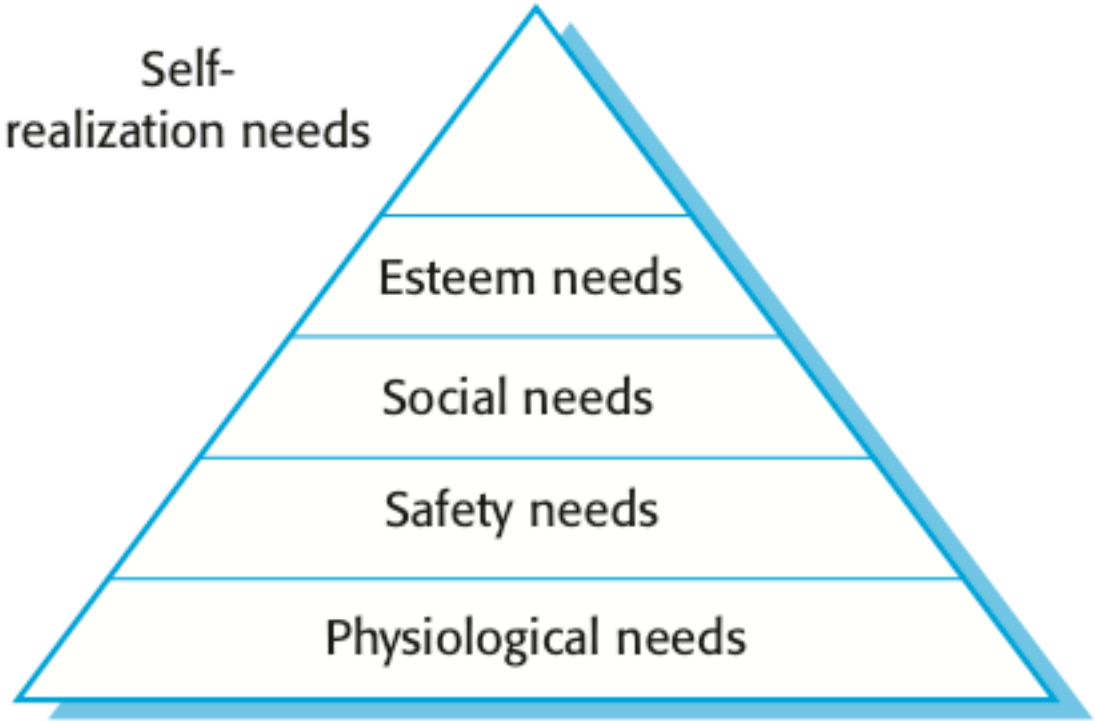
- ✧ Consistency
  - Team members should all be treated in a comparable way without favourites or discrimination.
- ✧ Respect
  - Different team members have different skills and these differences should be respected.
- ✧ Inclusion
  - Involve all team members and make sure that people's views are considered.
- ✧ Honesty
  - You should always be honest about what is going well and what is going badly in a project.

Motivating people

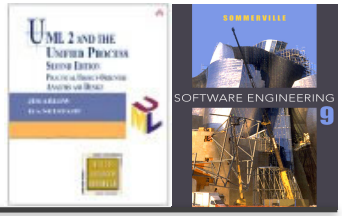


- An important role of a manager is to motivate the people working on a project.
- Motivation means organizing the work and the working environment to **encourage people to work effectively**.
  - If people are not motivated, they will not be interested in the work they are doing. They will work slowly, be more likely to make mistakes and will not contribute to the broader goals of the team or the organization.
- Motivation is a complex issue but it appears that their are different types of motivation based on:
  - Basic needs (e.g. food, sleep, etc.);
  - Personal needs (e.g. respect, self-esteem);
  - Social needs (e.g. to be accepted as part of a group).

Maslow's hierarchy of needs

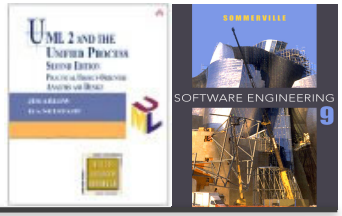


Need satisfaction



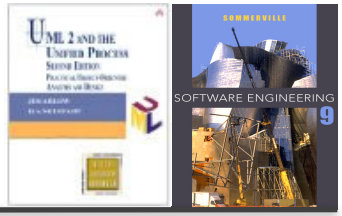
- In software development groups, basic physiological and safety needs are not an issue.
- Social
  - Provide communal facilities
  - Allow informal communications e.g. via social networking
- Esteem
  - Recognition of achievements
  - Appropriate rewards
- Self-realization
  - Training - people want to learn more
  - Responsibility

Personality traits



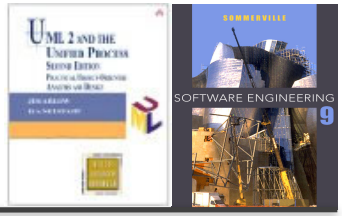
- Each personality is a composition of various traits.
- The Big Five personality traits include
  - extraversion,
  - conscientiousness,
  - agreeableness,
  - openness to experience,
  - neuroticism (also referred to as emotional stability).
- The traits may evolve with age, experience or other life conditions.

Personality orientation



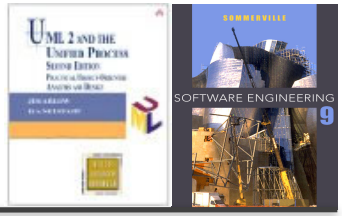
- Task-oriented
  - The motivation for doing the work is the work itself.
- Self-oriented
  - The work is a means to an end which is the achievement of individual goals - e.g. to get rich, to play tennis, to travel etc.
- Interaction-oriented
  - The principal motivation is the presence and actions of co-workers. People go to work because they like to go to work.
- Individual motivations are **made up of elements of each class**, where **teamwork** plays an essential role.

Teamwork



- Most software engineering is a group activity
  - The development schedule for most non-trivial software projects cannot be completed by one person working alone.
- A good group is **cohesive** and has a **team spirit**.
  - In a cohesive group, members consider the **group to be more important** than any individual in it.
- The advantages of a cohesive group are:
  - Team members **learn from each other** and get to know each other's work; Inhibitions caused by ignorance are reduced.
  - **Knowledge is shared**. Continuity can be maintained if a group member leaves.
  - Refactoring and continual **improvement is encouraged**. Group members work collectively to deliver high quality results and fix problems.

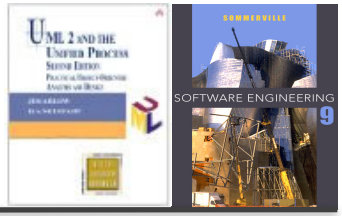
Optimal team size



- Anywhere between 2-20 members depending on:
  - the **purpose** for forming the team,
  - the **expectations** you have of the team and its members,
  - the **roles** that the team members need to play,
  - the amount of **cohesiveness** and interconnectivity necessary for optimal team performance, and
  - the **function, activities, and goals** of the team.
- In software development, the most common optimal size is **4-6 members (Two Pizza Rule)**.

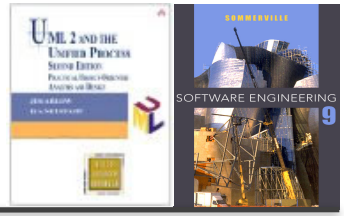
Can you think of a task that makes an optimal team size equal to 2, and other where the optimal size can be 20?

The effectiveness of a team



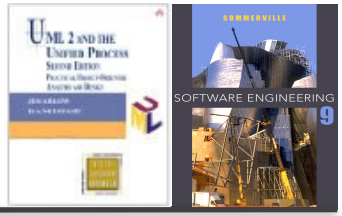
- The people in the group
  - You need a mix of people in a project group as software development involves diverse activities such as **negotiating** with clients, **programming**, **testing** and **documentation**.
- The group organization
  - A group should be organized so that individuals can contribute to the best of their abilities and tasks can be completed as expected.
- Technical and managerial communications
  - Good communications between group members, and between the software engineering team and other project stakeholders, is essential.

Group communications



- Group size
  - The larger the group, the harder it is for people to communicate with other group members.
- Group structure
  - Communication is better in **informally structured** groups than in hierarchically structured groups.
- Group composition
  - Communication is better when there are different **personality types** in a group and when groups are mixed rather than single gender.
- The physical work environment
  - Good workplace organisation can help encourage communications.

Key points



- People are motivated by **interaction with other people**, the **recognition of management and their peers**, and by **being given opportunities for personal development**.
- Software development groups should be fairly **small and cohesive**. The key factors that influence the effectiveness of a group are the people in that group, the way that it is organized and the communication between group members.
- **Communications** within a group are influenced by factors such as the status of group members, the size of the group, the gender composition of the group, personalities and available communication channels.