MODELING SOFTWARE SYSTEMS USING UML

EXERCISE

COMP 3111
SOFTWARE ENGINEERING

LECTURE 4
SOFTWARE DEVELOPMENT

LEARNING OBJECTIVES

1. Understand how software is planned, managed and developed in practice.

2. Understand and critique different software development processes.

3. Understand the importance of using an iterative and incremental software development process.

4. Know what are the components of a good software development process.

SOFTWARE DEVELOPMENT OUTLINE

Overview of Software Development

- Nature and Types of Software
- Types of Software Development Projects
- Software Development Life Cycle (SDLC)
- The Four P's in Software Development

Software Development Processes

- Monolithic
  - Waterfall
- Iterative and Incremental
  - Code-and-Fix
  - Prototyping
  - Spiral
  - Phased-release
  - Agile
  - Unified Process (UP)

THE NATURE OF SOFTWARE

- **Largely intangible**
  - Hard to: visualize; asses quality; appreciate development effort.
- **Easy and cheap to mass-produce**
  - Cost is mainly in its development, not its manufacture.
- **Development is labour intensive**
  - Hard to automate the design and programming process.
- **Easy to physically create and modify by anyone**
  - But may not be well designed; may be hard to find defects or modify correctly!
- **Does not wear out with use**
  - Code and design deteriorates due to defects added when modified.

TYPES OF SOFTWARE

A.

	Copies in use	Development effort	Requirements source
generic	*medium*	*medium*	*market research*
custom	*low*	*high*	*client needs*
embedded	*high*	*low*	*client/hardware needs*

B. **data processing** → organizes and stores business data

real-time processing → controls devices/processes in real time

C. **technical systems** → do not include knowledge of work procedures and processes.

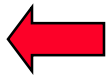
socio-technical systems → include knowledge of work procedures and processes.

Software engineering focuses mainly on the development of technical systems, BUT …

TYPES OF SOFTWARE DEVELOPMENT PROJECTS

1. Green field projects → new development

2. Evolutionary projects → maintenance

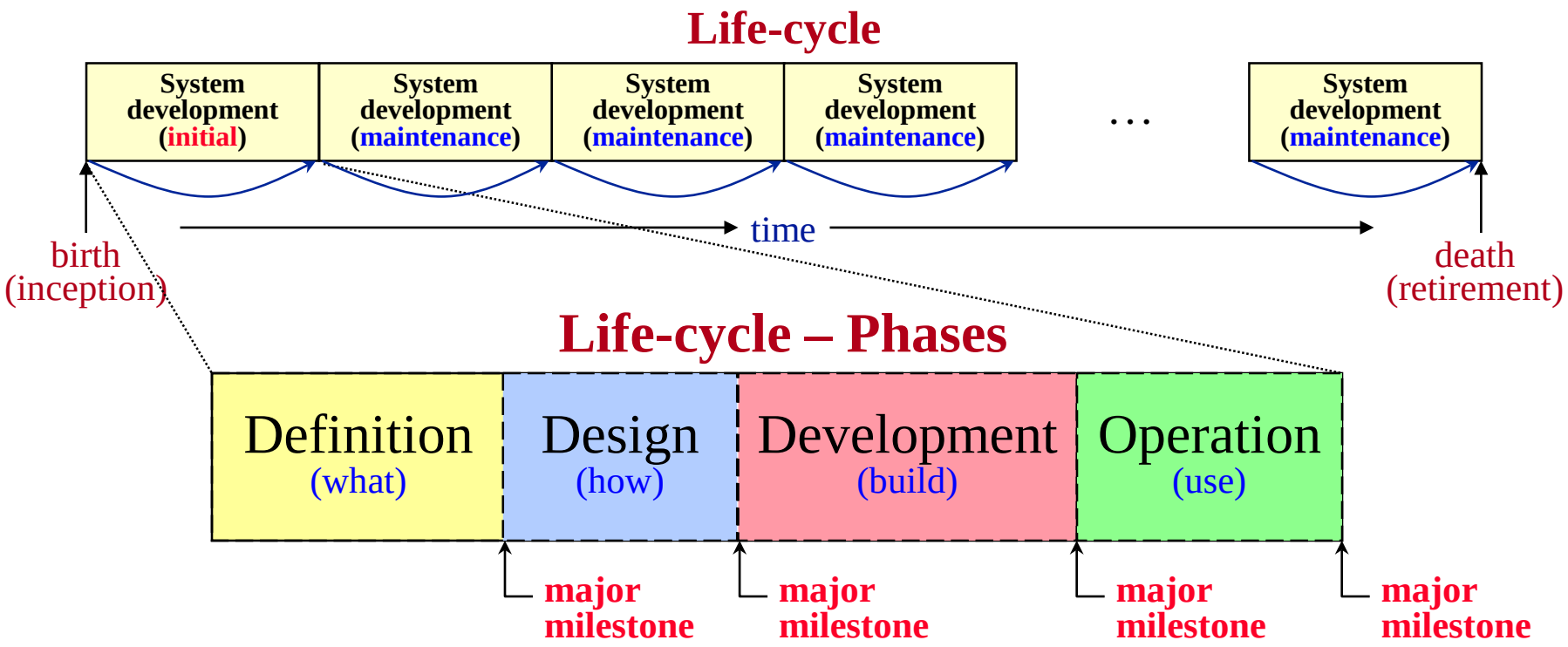
   Most common type.

   – corrective – fix defects
   – adaptive – adapt to new technology, new laws, etc.
   – enhancement – add new features
   – perfective (re-engineering) – make more maintainable

3. Framework/component projects → reuse
   – Use an existing framework or plug together several existing components.

     A framework is a software system designed specifically to be reused in different projects or a product line, but needing to be adapted to handle specific requirements.

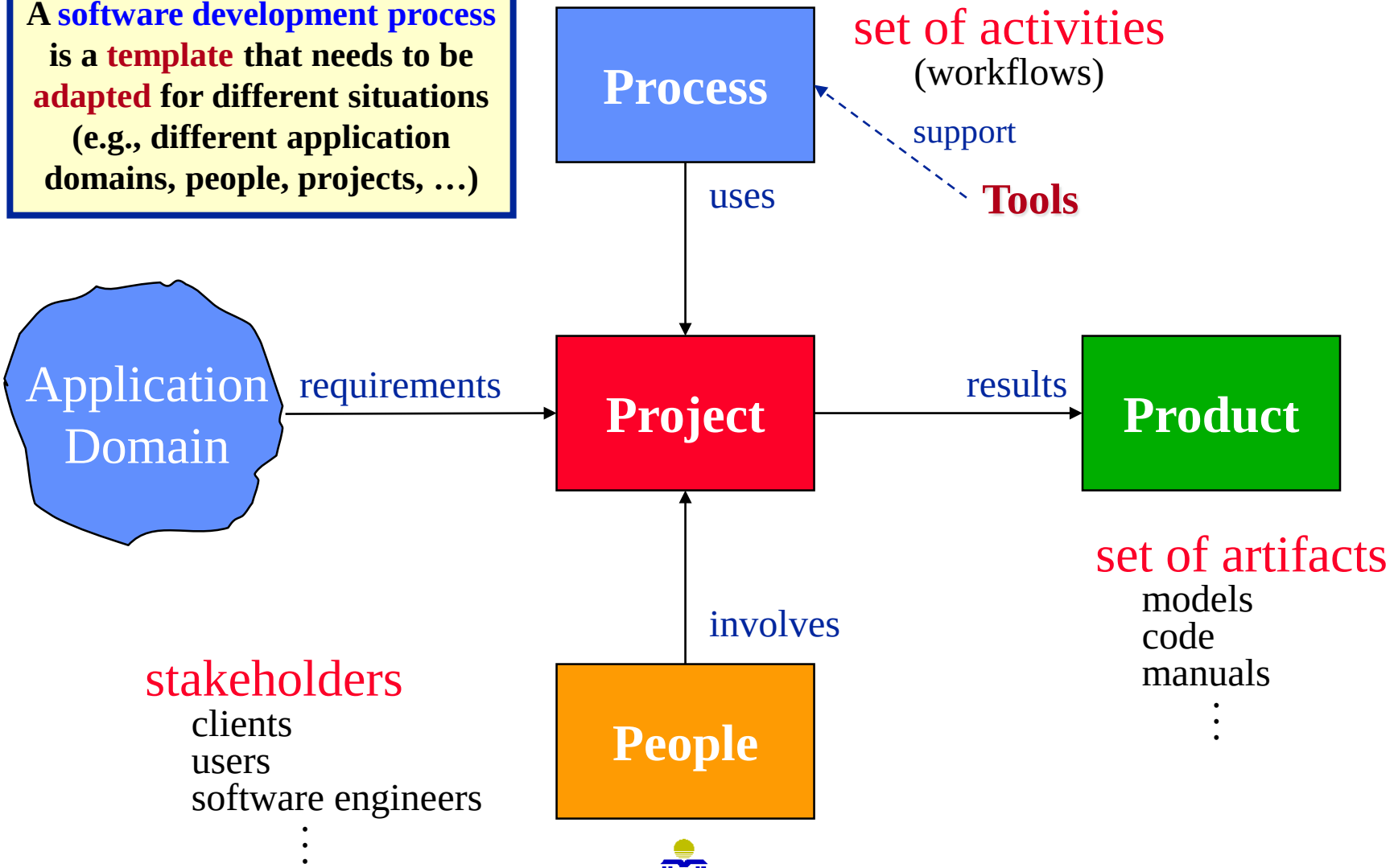
SOFTWARE DEVELOPMENT LIFE-CYCLE (SDLC)



- The life-cycle concept *structures* software development.
- The phases provide a basis for *management and control*, e.g., milestones, deliverables, etc.

THE FOUR P'S IN SOFTWARE DEVELOPMENT

A **software development process** is a **template** that needs to be **adapted** for different situations (e.g., different application domains, people, projects, …)



THE FOUR P'S: PROJECT

An activity carried out by people using a specific process that results in the release of a product and that requires planning, management and control.

- The **project plan** specifies the project's:
  - scope, objectives and constraints (e.g., budget, time, etc.).
  - risks, their likelihood and coping strategies.
  - organization (i.e., people and roles; assignment of people to roles).
  - tasks and activities and their associated milestones and deliverables.
  - schedule (i.e., dependencies between tasks, estimated time required to reach each milestone, allocation of people to tasks).
  - hardware and software resources required.
  - monitoring and reporting mechanisms.

☞ **Planning is a continuous activity!**

PROJECT SCOPE, OBJECTIVES AND CONSTRAINTS

The first task that needs to be performed is to determine the project's scope, objectives and constraints.

- Define the problem and set design goals
  - ☞ **To understand what the client wants and what is important.**
- Analyze the requirements
  - ☞ **To understand what needs to be implemented and to make system size estimates.**
- Prepare a top-level system diagram
  - ☞ **To get an overall view of the system.**
- Estimate the time and effort needed to deliver the product.
  - ☞ **To develop the project organization, schedule and budget.**

PROJECT RISKS

RISK

Anything that can go wrong (endanger project success).

Types of risks

- technology
- people
- organizational
- tools
- requirements
- estimation

Dealing with risks

- avoid (re-plan or change requirements)
- confine (restrict the scope of its effect)
- mitigate (devise tests to see if it occurs)
- monitor (constantly be on the lookout for it)

If you do not actively attack risks, they will actively attack you!

T. Gilb

RISK ANALYSIS

For each identified risk estimate:

1. The **likelihood (l_i)** that the **risk (r_i)** will occur.
   - Establish a scale → Boolean (yes, no), subjective (improbable, likely, etc.), probabilities (.1, .3, .5, etc.).

2. The **consequences** and **impact (x_i)** of the risk.
   - nature → What is likely to happen.
   - scope → What is likely to be affected and to what degree.
   - timing → When is it likely to happen and what will be its duration.

3. The **accuracy/confidence** of the projection.
   - What is the basis for the likelihood and impact of the risk?

Prioritize risks by perceived *impact* on the project.

RISK PLANNING

For each risk, develop strategies to monitor and manage it.

Example: risk r_i = high staff turnover likelihood l_i = 0.7
impact x_i = increase duration 15% and overall cost 12%

☞ **What steps can be taken to mitigate this risk?**

- We need to perform a cost/benefit analysis of countermeasures.

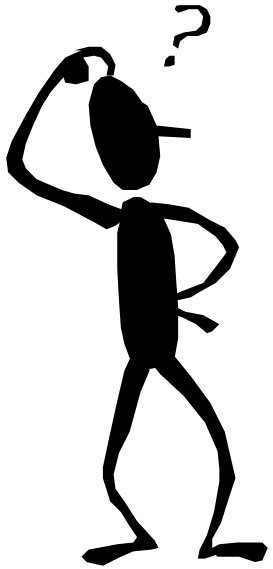
☞ **Do they cost more than the consequences of the risk itself?**

Apply 80:20 rule: 80% of all project risk is accounted for by 20% of identified risks.

☞ ***Closely monitor these 20% of risks!***

QUESTION?

The **greatest risk** you have to deal with in the course project is:



1) falling asleep during the lecture.
2) not knowing how to use the development tools.
3) not knowing the exact requirements for the system.
4) not knowing how to work together as a team.
5) not having enough time to finish the project.

"If you know the enemy and you know yourself, you need not fear the result of a hundred battles."

***The Art of War*, Sun Tzu**

PROJECT ORGANIZATION

The project organization specifies the

- roles and responsibilities; number of staff in each role; teams

☞ **To increase the likelihood of project success, one project member should have experience with a similar system!**

The team organization should:

- be modular to limit communication and complexity of interaction.
- assign clear responsibilities to each team member.
- form teams to have responsibility for the design and implementation of one or more parts of the system
- identify a PIC for each part and the system as a whole.

A *key to success* is achieving the right level of communication!

PROJECT TASKS AND ACTIVITIES

The tasks and activities specify the work that needs to be done.

task a well-defined work assignment for a role

activity a group of related tasks

- **Plan-driven**
  - Tasks, activities and their schedule are planned in detail at the start of the project.

- **Agile-driven**
  - Tasks, activities and their schedule are incrementally planned with most detail provided as the project progresses.

☞ **Which approach to use is highly project dependent.**
(Some combination is often most appropriate.)

PROJECT SCHEDULE

The project schedule specifies:

- **task ordering** → dependencies (sequential, parallel).
- **time estimates for each task** → start time, likely duration.
- **resource assignment** → people, hardware, software.
- **milestones** → important management decision points.
- **deliverables** → specifications, documents, code, etc.

- Often three levels of schedule are maintained:
  - **master schedule**: for management, client communication → **rigid**.
  - **macro schedule**: for day-to-day project management → **semi-rigid**.
  - **micro schedule**: for team management → **highly flexible**.

☞ **Charts or graphs are commonly used to manage schedules.**

ESTIMATES

***Estimating* is trying to quantify something before it occurs.**

The **project plan** may provide estimates of:

- size (lines of code (LOC), number of subsystems, number of classes, etc.)
- effort (persons X duration)
- productivity (size / effort)
- duration (months until delivery)
- development cost (labour)

- Estimating is based on:
  - experience
  - historical data
  - models
  - **courage**!

Estimating carries inherent risk.
(Which we try to minimize as much as possible.)

- The risk associated with estimating is reduced if we:
  - establish project scope in advance.
  - use historical data from past projects.
  - divide and conquer → break into small parts, estimate and sum.

THE FOUR P'S: PROCESS

The complete set of activities (also called workflows) and their sequencing that transforms user requirements into a software product.

- A process prescribes all of the major activities.
- A process has a set of guiding principles that explain the goals of each activity.
- A process uses resources, subject to a set of constraints (such as a schedule) and produces intermediate and final products.
- A process may be composed of subprocesses that are linked in some way (e.g., a hierarchy of processes)
- Each process activity has an entry and an exit criteria, so that we know when the activity begins and ends.
- The process activities are organized in a sequence, so that it is clear when one activity is performed relative to the other activities.
- Constraints or controls may apply to a process activity, resource or product (e.g., the budget may constrain the time spent in an activity or a tool may limit the way in which a resource may be used).

WHY PROCESS IS IMPORTANT

- Eases project management
- Allows division of labour
- Promotes teamwork / individual work / communication
- Allows expertise reuse / reassignment
- Eases training
- Promotes productivity / better development

A *process* imposes consistency and structure on the software development activities.

SOFTWARE DEVELOPMENT OUTLINE

- ✓ Overview of Software Development
  - Nature and Types of Software
  - Types of Software Development Projects
  - Software Development Life Cycle (SDLC)
  - The Four P's in Software Development

- ➔ **Software Development Processes**
  - Monolithic
    - Waterfall
  - Iterative and Incremental
    - Code-and-Fix
    - Prototyping
    - Spiral
    - Phased-release
    - Agile
    - Unified Process (UP)