

Chapter Five

Business Process Engineering

Outline

- **Definitions**
- **Phases of Software Development Life Cycle**
- **SDLC Models (Software Development Process Models)**

Definitions

- A **process** is a specific ordering of work activities across time and space, with a beginning, an end, and clearly identified inputs and outputs: **a structure for action**.
- A **business process** is a group of logically related tasks that use the firm's resources to provide customer-oriented results in support of organization's objectives.

A business process is the DNA of a company.

Definitions Cont....

- **Business process Engineering** focuses (in our case) on automating business processes with software processes and on assisting the analysis, design, implementation, control, maintenance, and optimization of software development process **to ensure success**.
- Also called
 - Business process Re-engineering
 - Business process Re-design (to reduce cost and add more value)

Definitions Cont....

- **Software engineering** is an important and critical discipline concerned with **cost effective software development**.
- It is based on a **systematic approach** that uses appropriate **tools and techniques**, operates under specific **constraints** and most importantly follows a **process**.
- **Read about triple constraint** (project management, an umbrella activity)

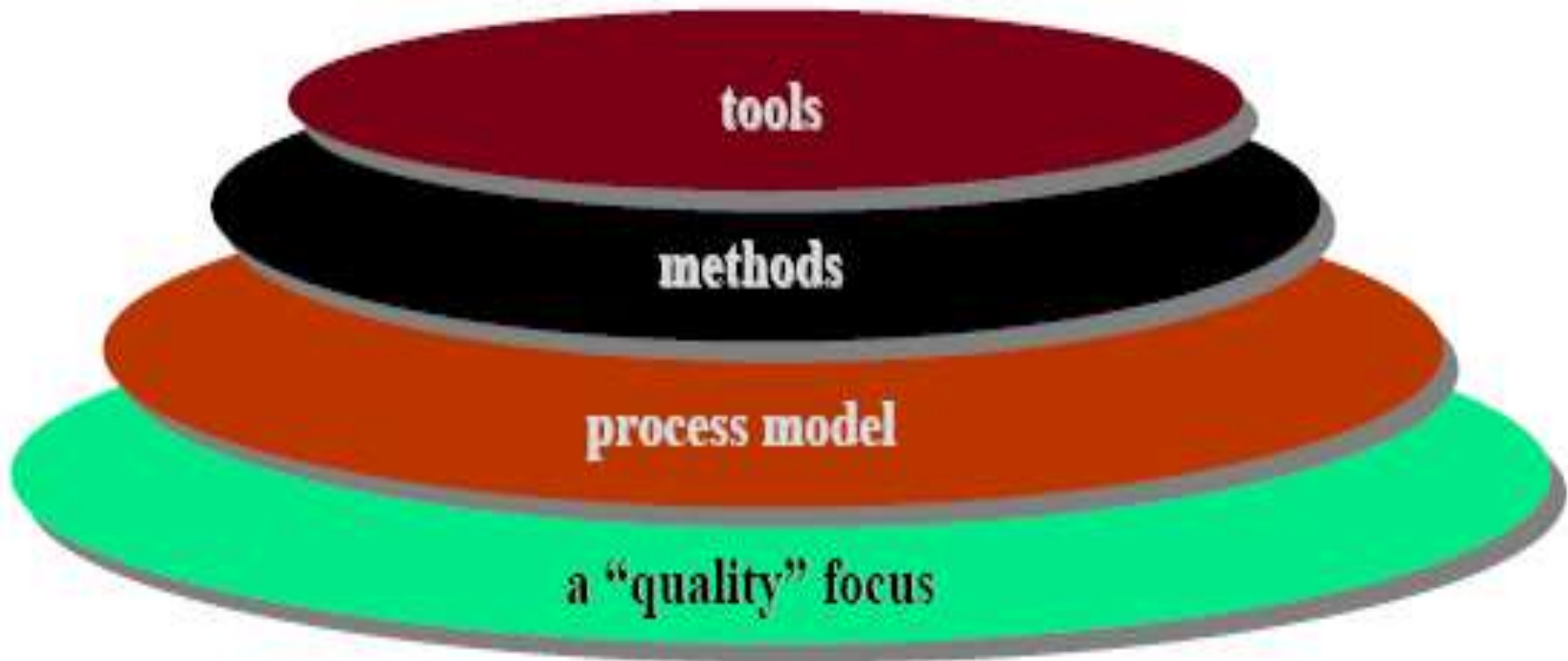
What is software engineering?

- Software engineering is an engineering discipline which is concerned with all aspects of software production
- Software engineers should adopt a systematic and organised approach to their work and use appropriate tools and techniques depending on the problem to be solved, the development constraints and the resources available

The IEEE definition Software Engineering:

- 1) The application of a **systematic, disciplined, quantifiable approach** to the **development, operation, and maintenance** of software; that is, the application of engineering to software.
- 2) The study of approaches as in 1)

A Layered Technology



Software Engineering

Software (System) development process

- The **steps, tools, processes** and **methods** used to develop information systems are collectively called **methodology**.
- The software system development process is also called ***system development life cycle (SDLC)***.

System Development Life Cycle (SDLC)

- System Development Life Cycle: is about developing a software-driven solution to a business problem
- It is about a process which takes roughly from months to years
- SDLC might more correctly be called a (Business) Solution Development Life Cycle

What is CASE (Computer-Aided Software Engineering)

- Software systems which are intended to provide automated support for software process activities
- Upper-CASE
 - Tools to support the early process activities of requirements and design
- Lower-CASE
 - Tools to support later activities such as programming, debugging and testing

Software (System) development process

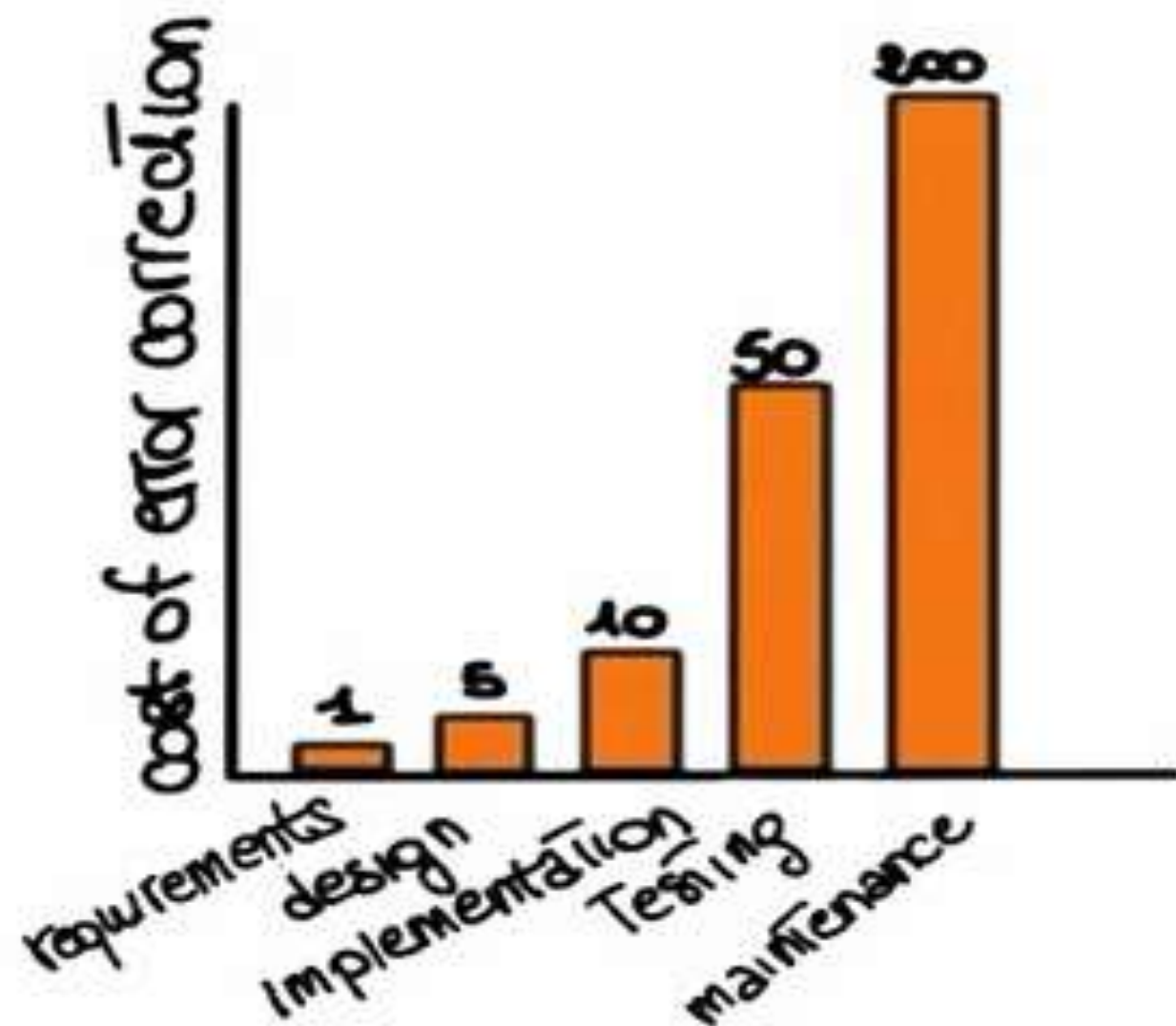
- Software System development process contains **fundamental** activities or phases.
- These phases are: Requirements engineering (As-Is), followed by System Design (To-Be), Implementation, Verification and Validation, and finally Maintenance.

Software (System) development process phases: SDLC phases

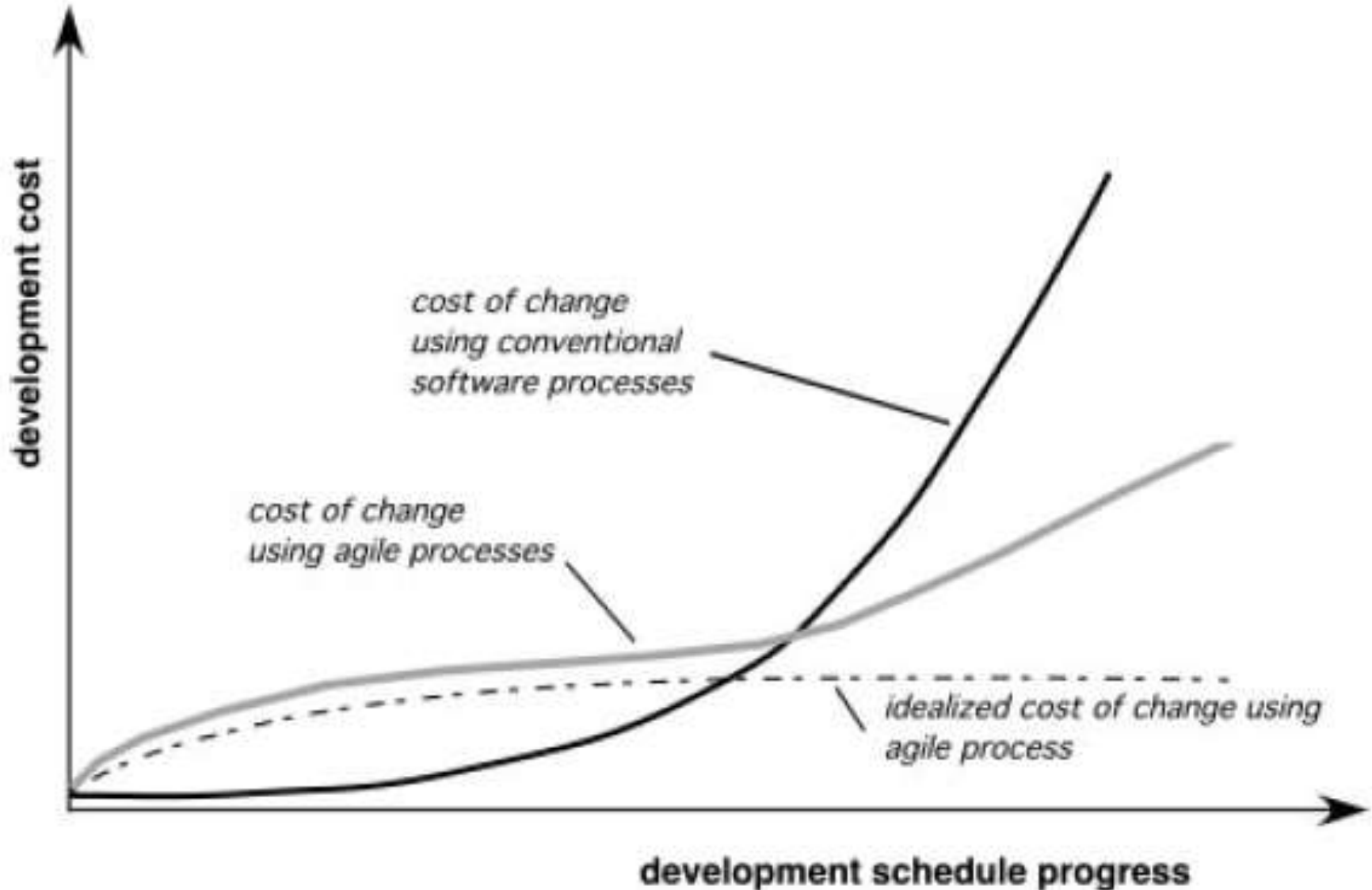
- 1) **Requirements Engineering**: is the process of establishing the **needs of stakeholders** that are **to be solved by software**.

Why is this phase so important? In general the **cost of correcting an error** depends on the number of subsequent decisions that are based on it. Therefore, **error made in understanding requirements** have the potential for greater cost because many **design** and other following phase decisions depend on **RE** results.

Cost of late correction



Agility and the Cost of Change



Requirements Engineering

- Also called **requirements analysis and specification**
- **How can we collect the right requirements?**
Traditional requirements engineering does so through a set of steps: The first step is **Elicitation** which is the collection of requirements from stakeholders and other sources and can be done in a variety of ways (**methods**) [through **questionnaire survey**, **interview**, **brainstorming**, **prototype**, **document analysis**, **physical work observation**, and so on]

Requirements Engineering

Cont.

- The **second** step is **requirements analysis** which involves the study and deeper understanding of the collected requirements.
- Structuring (**modeling**) of system requirements is also part of this activity.
- The **third** one is the **specification of requirements** in which the collected requirements are suitably represented organized and saved so that they can be **shared (SRS)**.
- They can be **validated** and **delivered (SRS)**.
- **Requirements management** may account to changes to requirements during the life time of the project.

System Design

- It is the phase where the description of the **internal structure** and **organization of the system** are produced and this description will serve as the basis for the **construction** of the actual system.
- Design activities normally consist of **architectural design**, **abstract specification**, **interface design**, **component** or **subsystem design**, and so on.
- These activities result in a set of **design products** which describe various characteristics of the system.

Implementation

- Here what we do is basically realizing the design of the system that we just created and **create an actual software system**.
- There are **four fundamental principles** or pillars that can affect the way software is constructed.
- The first is the **reduction of complexity (Usability)**.
- The second is the **anticipation of diversity (Agility)**.
- The third pillar is the **structuring for validation (Testability)**.
- Finally it is important that the SW **conforms to standards**.

Verification and Validation

- After we have built the system, here is the phase where we check that the software system **meet its specification** and fulfills **its intended purpose**.
- **Validation** is the activity that answers the question, **did we build the right system?**
- **Verification** answers a different question, **did we build the system right?** Did we build the system that actually implement the specification that we defined?
- **Verification** can be performed at the **unit level**, **integration level**, and finally the **system as a whole**.

Verification and Validation Cont.

- Thus, in **verification**, we want to make sure that the different modules talk to each other in the right way.
- During **system testing**, we test the system as a **whole** and want to make sure all the system, all the different pieces of the system work together in the right way. This is also the level at which we apply **validation** and some other testing techniques like **stress testing** or **robustness testing** and so on.

Maintenance

- Software development effort normally result in the delivery of a software product **that satisfies the user requirements**. However, when the software is in operation many things can happen.
- The **environment might change** (there might be new libraries, new operating systems) in which our software system has to operate.
- There may be **feature request** that the users might want to do something different with the product that we gave them.
- Or users might have **problems with the software** and may file **bug report**.

Maintenance Cont.

- Software maintenance is the activity that sustains the software product as it **evolves throughout its lifecycle**.
- Development organizations should perform three kinds of maintenance activities:
 - **Corrective maintenance** to eliminate problems with the code (bug).
 - **Perfective maintenance** to accommodate feature requests (and in some cases just to improve the software for example to make it more efficient).
 - **Adaptive maintenance** to take care of environment changes.

Maintenance Cont.

- And after these activities have been performed, the software developer will produce a **new version of the application** and will release it and the cycle will continue throughout the lifetime of the software.
- That is why **maintenance is a fundamental activity and very expensive one.**
- One of the reasons, in addition to the reasons in the previous slide, why maintenance is expensive is **regression testing**. Regression testing is the activity of retesting software after it has been modified.

Participants

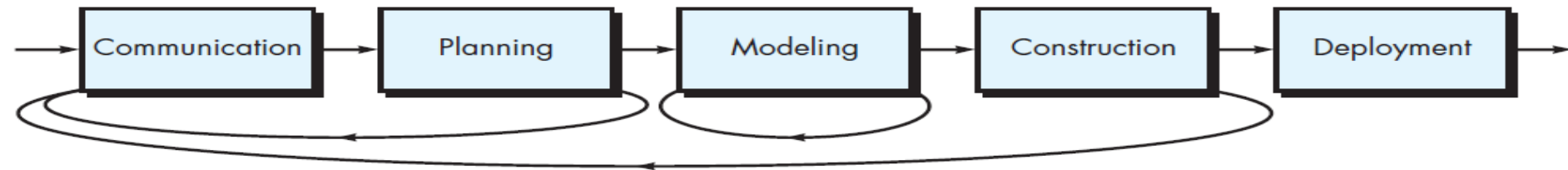
- As to the participants of a given Software System development process, the following are major ones:
 - system owners/clients or financers,
 - system users/stakeholders,
 - system analysts, software architects,
 - system designers/software engineers,
 - system builders/programmers,
 - project manager, software testers
 - etc.

What is a software process model?

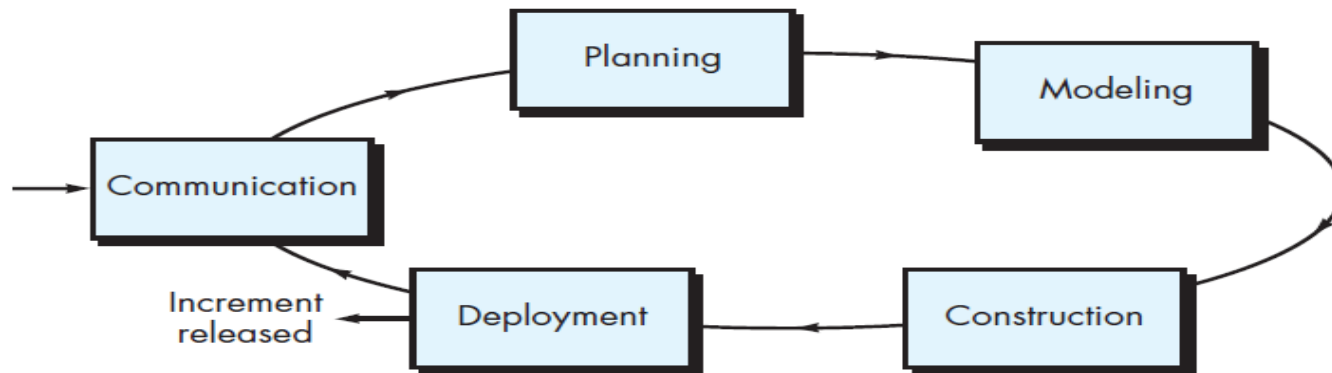
- A simplified representation of a software process, presented from a specific perspective
- Examples of process perspectives are
 - Workflow perspective - sequence of activities
 - Data-flow perspective - information flow
 - Role/action perspective - who does what
- Generic process models
 - Waterfall
 - Evolutionary development
 - Formal transformation
 - Integration from reusable components



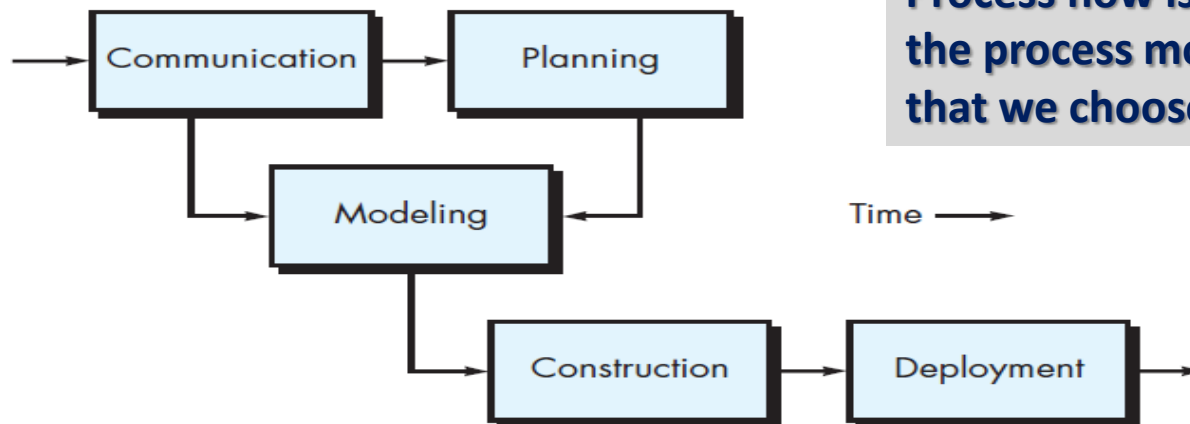
(a) Linear process flow



(b) Iterative process flow



(c) Evolutionary process flow

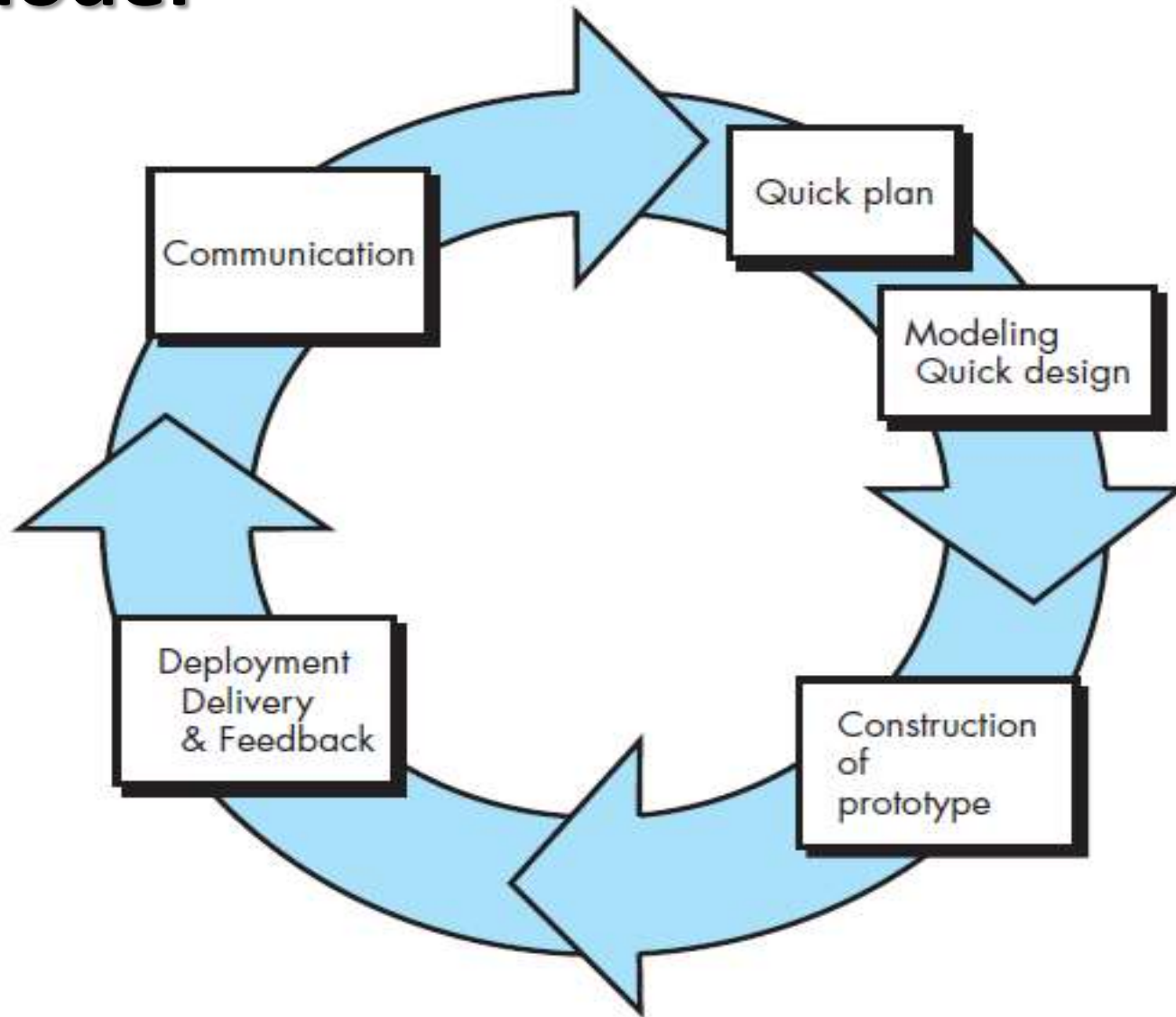


(d) Parallel process flow

Process flow is determined by the process model or framework that we choose for development

Time →

The Prototyping Paradigm or model



Assignment:

1. Make a research about SDLC process models and write your findings. Specify the advantages and disadvantages of each model. Include the open source model in your discussion.
 2. Make a thorough investigation about software CASE tools and write your findings.
- Work in groups of two students.