Benghazi University
Faculty of Information Technology software Engineering Department

SE341

Software Evolution & Maintenance

Part .6

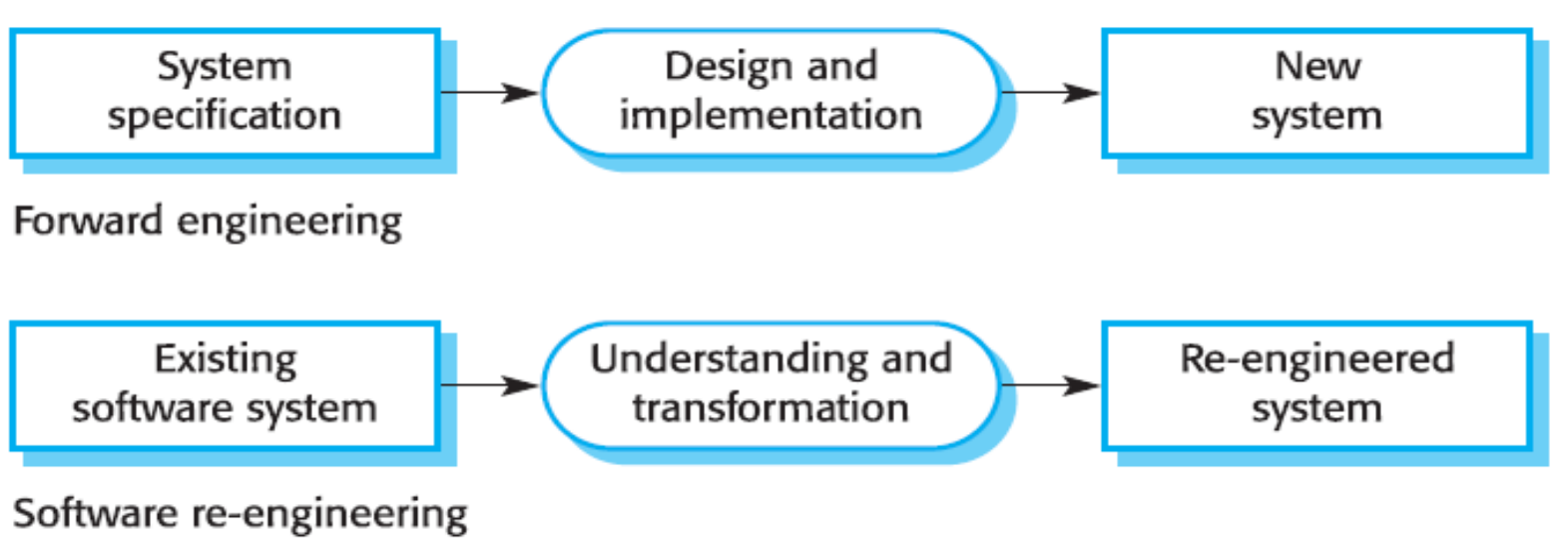
Instructor: ***Mohammed Sultan***

Full (2025-2026)

Forward Engineering and Reengineering

"*Forward Engineering* is the traditional process of moving from high-level abstractions and logical, implementation-independent designs to the physical implementation of a system."

"*Reengineering* ... is the examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form."



System Re-Engineering

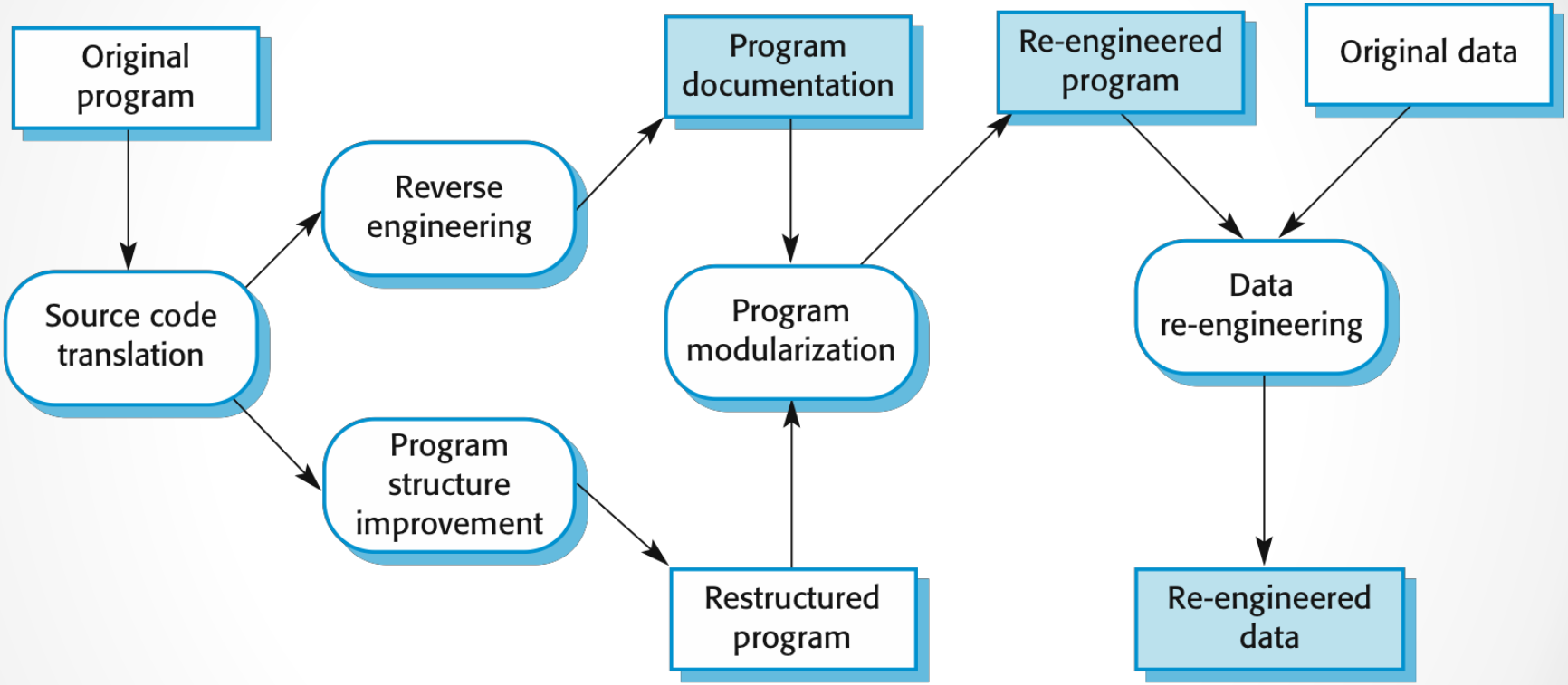
- Re-structuring or re-writing part or all of a legacy system without changing its functionality
- Applicable where some but not all sub-systems of a larger system require frequent maintenance
- Re-engineering involves adding effort to make them easier to maintain. The system may be re-structured and re-documented.

Advantages of reengineering

- Reduced risk
  - There is a high risk in new software development. There may be development problems, staffing problems and specification problems.

- Reduced cost
  - The cost of re-engineering is often significantly less than the costs of developing new software.

The reengineering process



Reengineering process activities

- Source code translation
  - Convert code to a new language
- Reverse engineering
  - Analyze the program to understand it
- Program structure improvement
  - Restructure automatically for understandability
- Program modularization
  - Reorganize the program structure
- Data reengineering
  - Clean-up and restructure system data

Reengineering cost factors

- The quality of the software to be reengineered
- The tool support available for reengineering
- The extent of the data conversion which is required
- The availability of expert staff for reengineering
  - This can be a problem with old systems based on technology that is no longer widely used

Preventive maintenance by refactoring

- Refactoring is the process of making improvements to a program to slow down degradation through change
- You can think of refactoring as 'preventive maintenance' that reduces the problems of future change
- Refactoring involves modifying a program to improve its structure, reduce its complexity or make it easier to understand
- When you refactor a program, you should not add functionality but rather concentrate on program improvement

Refactoring and reengineering

- Re-engineering takes place after a system has been maintained for some time and maintenance costs are increasing. You use automated tools to process and re- engineer a legacy system to create a new system that is more maintainable.

- Refactoring is a continuous process of improvement throughout the development and evolution process. It is intended to avoid the structure and code degradation that increases the costs and difficulties of maintaining a system.

"Bad smells" in program code

- Duplicate code
  - The same or very similar code may be included at different places in a program. This can be removed and implemented as a single method or function that is called as required.
- Long methods
  - If a method is too long, it should be redesigned as a number of shorter methods
- Switch (case) statements
  - These often involve duplication, where the switch depends on the type of a value. The switch statements may be scattered around a program. In object-oriented languages, you can often use polymorphism to achieve the same thing.

"Bad smells" in program code

- Data clumping
  - Data clumps occur when the same group of data items (fields in classes, parameters in methods) re-occur in several places in a program. These can often be replaced with an object that encapsulates all of the data.

- Speculative generality
  - This occurs when developers include generality in a program in case it is required in the future. This can often simply be removed.