Presentations
Dates – to be announced

- Timeslot : 10 Minutes - 15 marks
  - Title
  - Introduction
  - Problem
  - Related research
  - Findings
  - Your opinion (flaws , threats to validity , missing assumptions ,..etc.)
  - (2 marks presentation skills+ 3 marks opponent)

Faculty of Computers and Information
BSc. In Software Engineering Program

Dr. Lamia Abo Zaid

د. لمياء أبوزيد

Based on slides by © Tripathy & Naik

Software Evolution : TOC

1. Introduction to Software Evolution
2. Taxonomy of Software Maintenance and Evolution
3. Evolution and Maintenance Models
4. Reuse and Domain Engineering
5. Program Comprehension
6. Impact Analysis
7. Refactoring
8. Reengineering
9. Legacy Information Systems

Reengineering – What and Why ?

❑What is Reengineering ?

- Reengineering is the examination, analysis, and restructuring of an existing software system to reconstitute it in a new form and the subsequent implementation of the new form.

❑Why do we do reengineering?

1. Understand the existing software system artifacts, namely, specification, design, implementation, and documentation
2. Improve the functionality and quality attributes of the system.

- Some examples of quality attributes are: performance, reliability, correctness, integrity, efficiency, maintainability, usability, testability, interoperability, reusability, and portability

Objectives of Reengineering

Software systems are reengineered to meet one or more of the objectives:

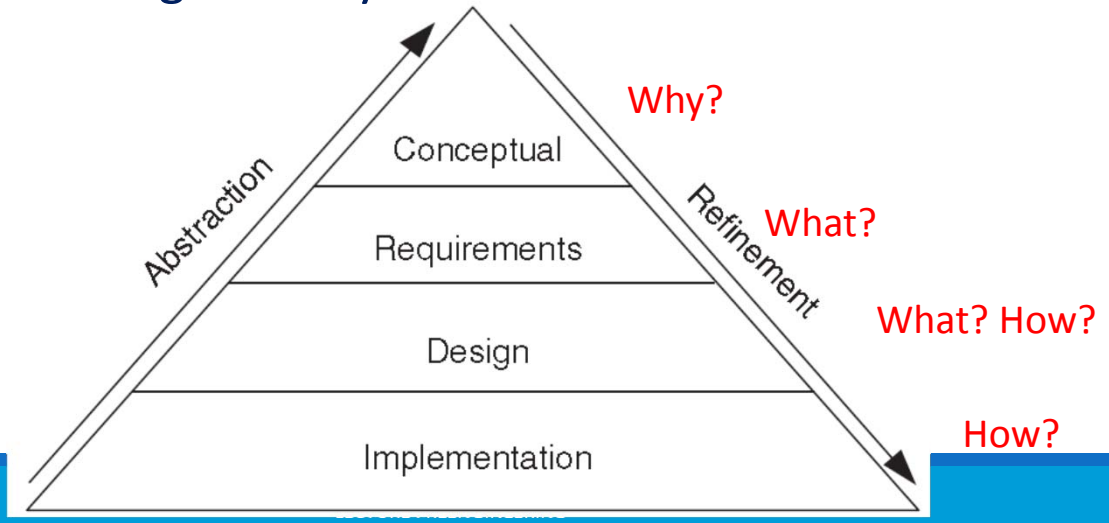
❑ Improve maintainability	Lehman's second law, increasing complexity
❑ Migrate to a new technology	Lehman's first law, continuing change
❑ Improve quality	Lehman's seventh law, declining quality
❑ Prepare for functional enhancement	Lehman's sixth law, continuing growth

Reengineering Concepts

- Abstraction and Refinement are key concepts used in software development, and are equally useful in reengineering.
- Principle of abstraction: By means of abstraction one can produce a view that focuses on selected system characteristics by hiding information about other characteristics. The level of abstraction of the representation of a system can be gradually increased by successively replacing the details with abstract information.
- Principle of refinement: The level of abstraction of the representation of the system is gradually decreased by successively replacing some aspects of the system with more details.
- Refinement is the reverse of abstraction

Reengineering Concepts

- A new software is created by going downward from the top, i.e. highest level of abstraction to the bottom i.e. lowest level.
- This downward movement is known as Forward Engineering
- Upward movement through the layers of abstractions is called Reverse Engineering.



Reengineering Concepts

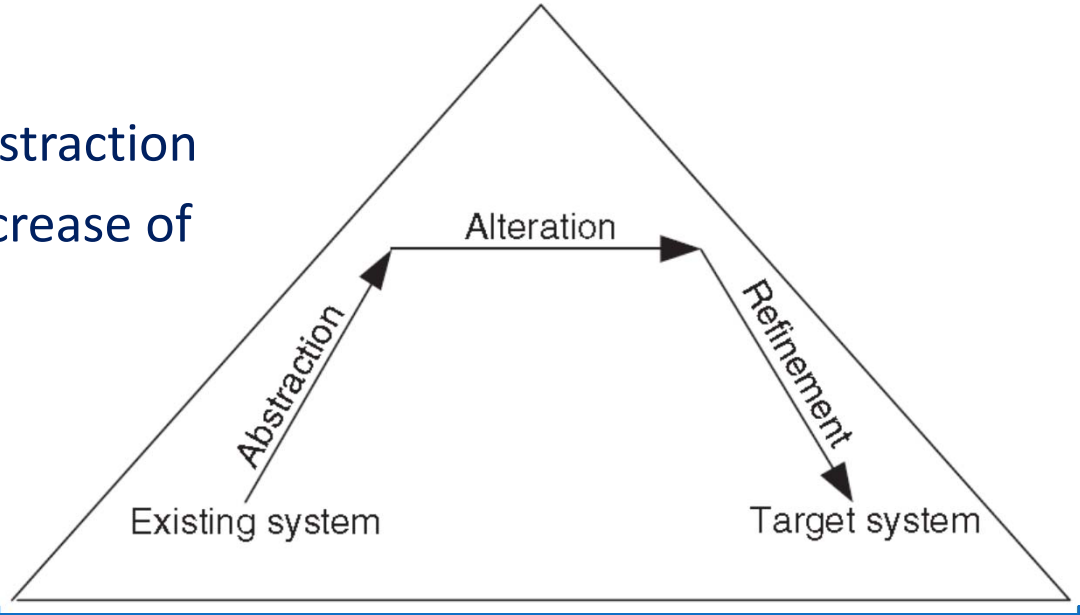
- An optional principle called **alteration** underlies many reengineering methods.
- **Principle of alteration:** The making of some changes to a system representation is known as alteration. Alteration does not involve any change to the degree of abstraction, and it does not involve modification, deletion, and addition of information.

Reengineering Concepts

- **Reengineering principles** are represented by means of arrows. Abstraction is represented by an up-arrow, alteration is represented by a horizontal arrow, and refinement by a down-arrow.

- The arrows depicting refinement and abstraction are sloped indicating the increase and decrease of system information respectively.

- A term closely related to “alteration” is restructuring (refactoring)



General Reengineering Model

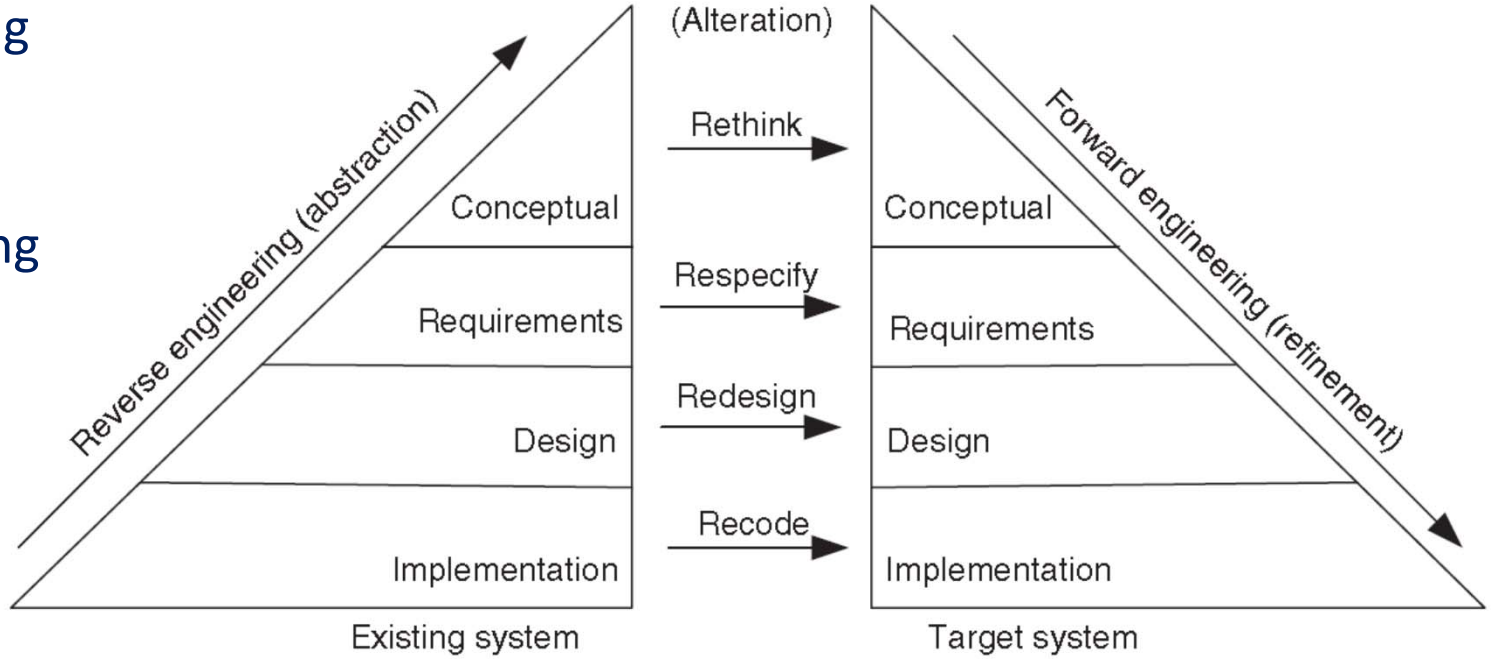
The reengineering process consists of the following stages:

1. Recreate a design from the existing source code.
2. Find the requirements of the system being reengineered.
3. Compare the existing requirements with the new ones.
4. Remove those requirements that are not needed in the renovated system.
5. Make a new design of the desired system.
6. Code the new system.

General Reengineering Model

Eric J. Byrne suggests that reengineering is a sequence of three activities:

1. Reverse engineering
2. Re-design
3. Forward engineering



General Reengineering Model

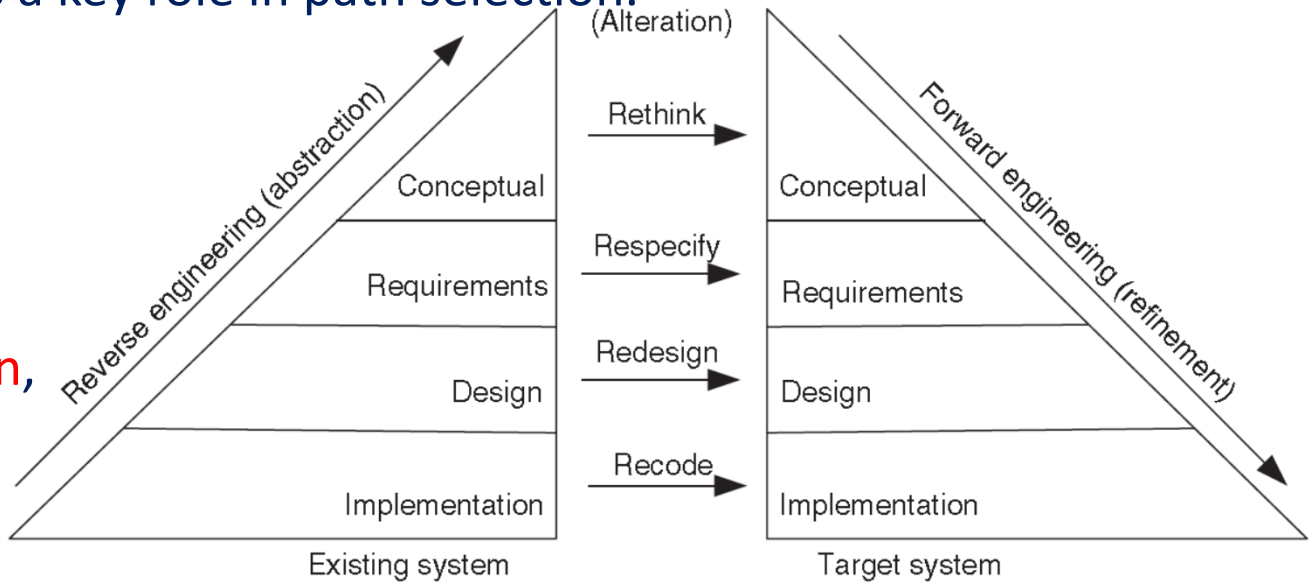
- Jacobson and Lindstorm define reengineering as:

  Reengineering = Reverse engineering + Δ + Forward engineering

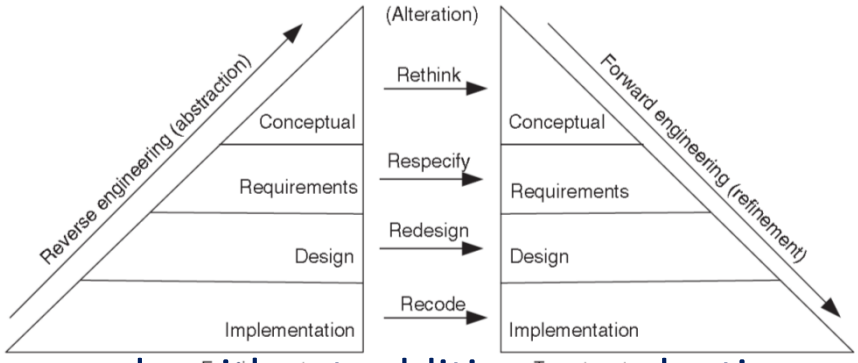
- The element "Δ" captures alterations made to the original system.
- Two major dimensions of alteration are: change in functionality and change in implementation technique.
  - A change in functionality comes from a change in the business rules.
  - Change of implementation technique. The end-user of a system never

Types of Changes

- The selection of a specific path for reengineering depends partly on the characteristics of the system to be modified.
- For a given characteristic to be altered, the abstraction level of the information about that characteristics plays a key role in path selection.
- Based on the type of changes required, reengineering characteristics are divided into groups: rethink, respecify, redesign, and recode

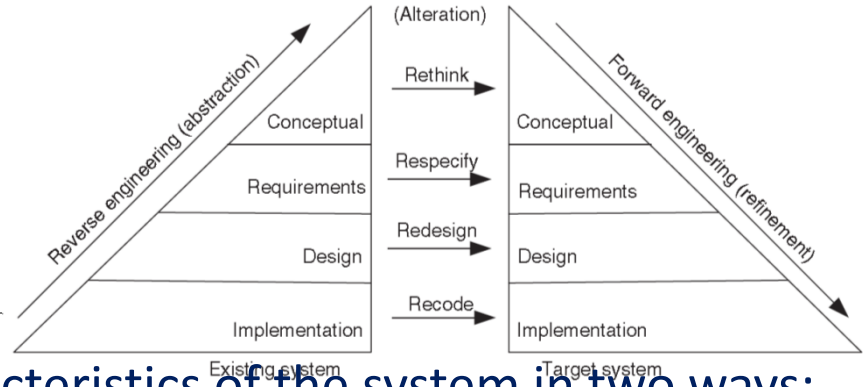


Types of Changes



- Recoding (also Rehosting) means reengineering of source code without addition or reduction of features in the transformed targeted source code.
  - Rehosting is most effective when the user is satisfied with the system's functionality, but looks for better qualities of the system.
  - Rephrasing and program translation are other means of source-code level changes
- Redesign refers to altering the design characteristics of the software by re-designing the system to :
  - Restructure the architecture
  - Modify the data model of the system
  - Replace a procedure or an algorithm with a more efficient one.

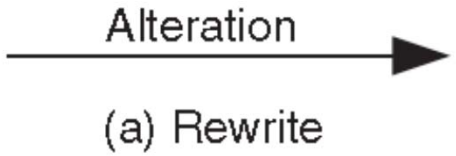
Types of Changes



- Respecify means changing the requirement characteristics of the system in two ways:
  - Change the form of the requirements.
  - Change the scope of the requirements.
- Rethink means manipulating the concepts embodied in an existing system to create a system that operates in a different problem domain.
  - Rethink involves changing the conceptual characteristics of the system, and it can lead to fundamental changes.
  - E.g. moving from the development of an ordinary cellular phone to the development of smartphone.

Software Reengineering Strategies

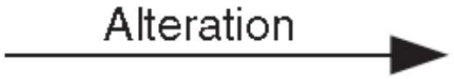
- Three strategies rewrite, rework, and replace specify the basic steps of reengineering
- Rewrite strategy: reflects the principle of alteration. By means of alteration, an operational system is transformed into a new system while preserving the abstraction level of the original system.
  - For example, the Fortran code of a system can be rewritten in the C language



(a) Rewrite

Software Reengineering Strategies

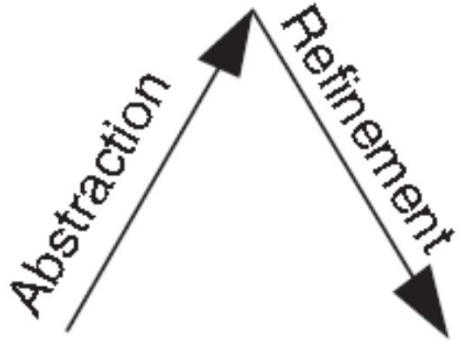
- Three strategies rewrite, rework, and replace specify the basic steps of reengineering
- Rewrite strategy: reflects the principle of alteration.



  - For example, the Fortran code of a system can be rewritten in the C language
- Rework strategy reflects the principles of abstraction , alteration and refinement .
  - For example, one can create an abstraction of source code in the form of a high-level design. Next, the reconstructed system model is transformed into the target system representation, by means of alteration. Finally, by means of refinement, an appropriate new system representation is created at a lower level of abstraction

Software Reengineering Strategies

- Replace strategy: applies abstraction and refinement
  - The system is reconstructed at a higher level of abstraction by hiding the details
  - a suitable representation for the target system is generated at a lower level of abstraction by applying refinement



Reengineering Process Variations

- Possible variations in reengineering processes can obtained from combing the reengineering strategies (rewrite, rework, and replace) and the change types (rethink, respecify, redesign, and recode)

Questions

?