National University of Computer & Emerging Sciences, Karachi

Quiz 1 – SREE – BSE – 8B (Spring-2024)

20th February 2024 (Tuesday), 03:00 PM – 03:55 PM

Course Code: SE4001	**Course Name: Software Re-Engineering (SREE)**
Instructor Name: Dr. Syed Muazzam Ali Shah	
Student Roll No:	**Section No:**

Time: 40 minutes. **Max Marks**: 10 Points

Q1: Differentiate between these two types of maintenance: corrective and adaptive.

Answer:

Corrective Maintenance:

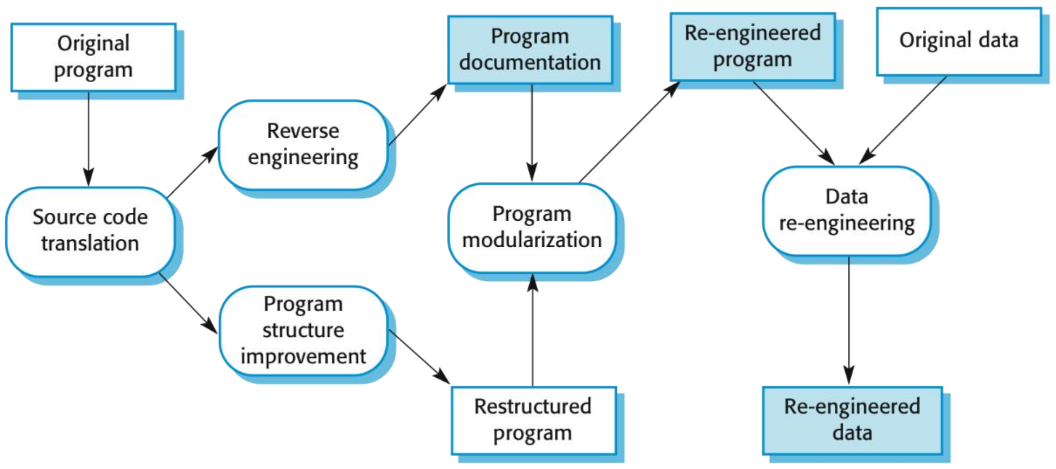
Fault repairs to fix bugs and vulnerabilities. Coding errors are usually relatively cheap to correct; design errors are more expensive because they may involve rewriting several program components. Requirements errors are the most expensive to repair because extensive system redesign may be necessary.

Adaptive Maintenance:

Environmental adaptation to adapt the software to new platforms and environments. This type of maintenance is required when some aspect of a system's environment, such as the hardware, the platform operating system, or other support software, changes. Application systems may have to be modified to cope with these environmental changes.

Q2: Draw the visual representation of the entire re-engineering process.

Answer:



Q3: Refactoring is concern about making improvements in to a program to slow down degradation through change. Here what does mean by "slow down degradation through change".

Answer:

The software system goes through a series of modifications and upgradations, due to which the architecture/structure of the software system becomes corrupt and degrade. Refactoring is the process of upgrading and make modifications into the internal structure of the software system without affecting the functionality (external behavior), particularly the degraded/corrupted parts or component in order to improve the quality of the software system.

Q4: Why it is expensive/costly to add new functionality or made modifications to existing functionality while maintaining an existing (legacy) software system)?

Answer:

Incorporating new functionality or new requirements to the existing system may require extensive analysis and validation during requirements specification, design and implementation rework. Therefore, it is often expensive to add new functionality to the existing system.

Q5: Explain the Cold Turkey strategy for software re-engineering.

Answer:

In the cold turkey strategy, you are developing the system from starch by ignoring the existing system. Now the question is when to ignore the existing system?
Following are the few situations where we can ignore the existing system

- The business has changed and new methods of doing business have been developed – need to build systems to implement these new methods.
- It is too costly to try and understand what you currently have.
- It is too costly to try to reengineer the current systems.
- Time and resources are available to start from scratch.

Q6: In which circumstances do we need to adopt spiral model of development and evaluation.

Answer:

Spiral model of development and evaluation is appropriate for the software systems where there is a need to release a series of version after the system has been delivered because of the continuous change requests.

Q7: Differentiate between reverse engineering and forward engineering, and also provide an example of reverse engineering.

Answer:

- A new software is created by going downward from the top, highest level of abstraction to the bottom, lowest level. This downward movement is known as forward engineering.
- Forward engineering follows a sequence of activities: formulating concepts about the system to identifying requirements to designing the system to implementing the design.

- On the other hand, the upward movement through the layers of abstractions is called reverse engineering.
- Reverse engineering of software systems is a process comprising the following steps:
- Analyze the software to determine its components and the relationships among the components.
- Represent the system at a higher level of abstraction or in another form.
- Decompilation is an example of Reverse Engineering, in which object code is translated into a high-level program.

Q8: Discuss some of the factors associated with software re-engineering that are affected by the cost.

Answer:

- Costs are affected by the following factors:
  - The quality of the software to be re-engineered.
  - The tool support available for the re-engineering.
  - The extent of the required data conversion.
  - The availability of expert staff for re-engineering.

Q9: How does management making predetermined technical decisions lead to the failure of a software re-engineering project?

Answer:

Mandates or edicts issued by upper management that predetermine the technical approach or schedule, cost, and performance considerations without sufficient project team input or concurrence are frequently seen to cause reengineering failure. More often than we would like to admit, project schedules, costs, and deliverables are dictated by top management decisions. Software is a difficult business, and especially where one is dealing with legacy systems that may have poorly developed components and poor documentation. While top management does need to make decisions on the allocation of scarce resources, it is tempting for them to also determine specific deliverables and timetables. However, detailed planning of schedules and milestones can only be accurately determined through careful study of the technical parameters of a system, based on an understanding of the system, historical data, and knowledge of the specific skills of the staff. When top management prescribes these details with little data other than hunches, the results are usually disastrous.

Q10: Discuss two bed smells that can be improved using refactoring.

Answer:

- 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.