SOFTWARE MAINTENANCE

ANIKET MISHRA

WHAT IS MAINTENANCE

- Once the product has its acceptance test, it is handed over to the client.
- The product is installed and used for the purpose for which it was constructed.
- Any useful product, however, is almost certain to undergo maintenance during the maintenance phase, either to fix faults (corrective maintenance) or extend the functionality of the product (enhancement).

TYPES OF MAINTENANCE

- *Corrective maintenance*
- *Perfective maintenance*
- *Adaptive maintenance*
- *Preventive maintenance*

- *i. Corrective maintenance*:
- This involves correcting faults, whether specification faults, design faults, coding faults, documentation faults, or any other types of faults.

- *ii. Perfective maintenance*:
- Here, changes are made to the code to improve the effectiveness of the product.
- For instance, the client may wish additional functionality or request that the product be modified so that it runs faster.
- Improving the maintainability of a product is another example of perfective maintenance. The study showed that 60.5 percent time was spent on this type of maintenance.

- *iii. Adaptive maintenance*:
- Here, changes are made to the product to react to changes in the environment in which the product operates.
- For example, a product almost certainly has to be modified if it is ported to a new compiler, operating system, or hardware.
- Adaptive maintenance is not requested by a client; instead, it is externally imposed on the client.
- The study showed that 18 percent of software maintenance was adaptive in nature.

- *Preventive maintenance* is other type which concerns activities aimed at increasing the system's maintainability, such as updating documentation, adding comments, and improving the modular structure of the system.

MAINTENANCE LOG AND DEFECT REPORTS

- Maintenance Log keeps track of the cost of operations performed, scheduled maintenance, and unscheduled repairs. In short, it is the storage of activities and daily updates involved in the maintenance phase

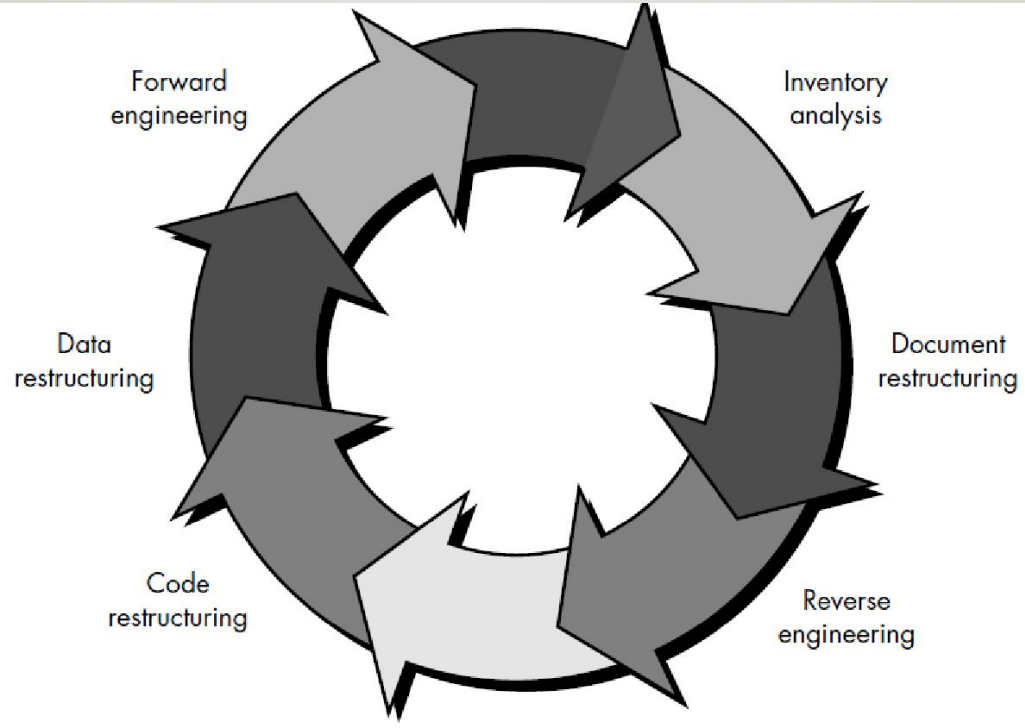
RE-ENGINEERING

- *Re-engineering is the reorganizing and modifying existing software systems to make them more maintainable.*

REENGINEERING

- Reengineering takes time; it costs significant amounts of money; and it absorbs resources
- For all of these reasons, reengineering is not accomplished in a few months or even a few years.
- Reengineering of information systems is an activity that will absorb information technology resources for many years.
- That's why every organization needs a pragmatic strategy for software reengineering.
- A software reengineering model that defines six activities is shown by the following figure:

REENGINEERING



INVENTORY ANALYSIS

- sorting the information according to business criticality, longevity, current maintainability, and other locally important criteria, candidates for reengineering appear. Resources can then be allocated to candidate applications for reengineering work.

DOCUMENT RESTRUCTURING

- Weak documentation is the trademark of many legacy systems. The following options can be adopted:
- Creating documentation is far too time consuming.
- Documentation must be updated, but we have limited resources.
- The system is business critical and must be fully redocumented.
- Each of these options is viable. A software organization must choose the one that is most appropriate for each case.

REVERSE ENGINEERING

- reverse engineering for software is the process of analyzing a program in an effort to create a representation of the program at a higher level of abstraction than source code.
- Reverse engineering is a process of design recovery. Reverse engineering tools extract data, architectural, and procedural design information from an existing program.

CODE RESTRUCTURING

- The most common type of reengineering is code restructuring. Some legacy systems have the individual modules coded in a way that makes them difficult to understand, test, and maintain. In such cases, the code within the suspect modules can be restructured.
- To accomplish this activity, the source code is analyzed using a restructuring tool. Violations of structured programming constructs are noted and code is then restructured. The resultant restructured code is reviewed and tested. Internal code documentation is updated.

DATA RESTRUCTURING

- A program with weak data architecture will be difficult to adapt and enhance. In fact, for many applications, data architecture has more to do with the long-term viability of a program that the source code itself.
- Unlike code restructuring, which occurs at a relatively low level of abstraction, data structuring is a full-scale reengineering activity. In most cases, data restructuring begins with a reverse engineering activity. Current data architecture is dissected and necessary data models are defined. Data objects and attributes are identified, and existing data structures are reviewed for quality.
- When data structure is weak, the data are reengineered. Because data architecture has a strong influence on program architecture and the algorithms that populate it, changes to the data will invariably result in either architectural or code-level changes.

FORWARD ENGINEERING

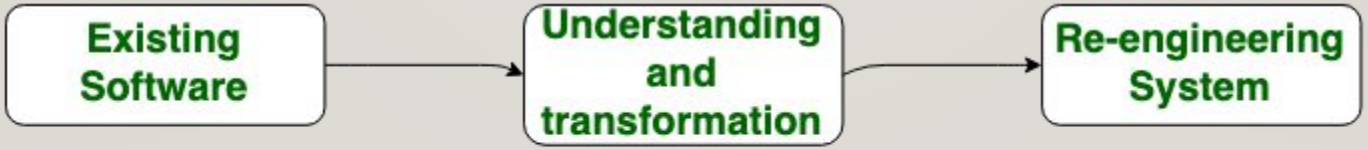
- Forward engineering, also called *renovation* or *reclamation*, not only recovers design information from existing software, but uses this information to alter or reconstitute the existing system in an effort to improve its overall quality. In most cases, reengineered software re implements the function of the existing system and also adds new functions and/or improves overall performance.

DIFFERENCE BETWEEN FORWARD ENGINEERING AND REVERSE ENGINEERING

- **Forward Engineering:**

Forward Engineering

- **Reverse Engineering:**



Reverse Engineering

DIFFERENCE BETWEEN FORWARD ENGINEERING AND REVERSE

S.NO	Forward Engineering	Reverse Engineering
1.	In forward engineering, the application are developed with the given requirements.	In reverse engineering or backward engineering, the information are collected from the given application.
2.	Forward Engineering is a high proficiency skill.	Reverse Engineering or backward engineering is a low proficiency skill.
3.	Forward Engineering takes more time to develop an application.	While Reverse Engineering or backward engineering takes less time to develop an application.
4.	The nature of forward engineering is Prescriptive.	The nature of reverse engineering or backward engineering is Adaptive.
5.	In forward engineering, production is started with given requirements.	In reverse engineering, production is started by taking the products existing products.
6.	The example of forward engineering is the construction of electronic kit, construction of DC MOTOR , etc.	An example of backward engineering is research on Instruments etc.