Reverse Engineering

Reverse engineering can extract design information from source code, but the abstraction level, the completeness of the documentation, the degree to which tools and a human analyst work together, and the directionality of the process are highly variable.

Objective of Reverse Engineering:

1. **Reducing Costs:** Reverse engineering can help cut costs in product development by finding replacements or cost-effective alternatives for systems or components.

2. **Analysis of Security:** Reverse engineering is used in cybersecurity to examine exploits, vulnerabilities, and malware. This helps in understanding of threat mechanisms and the development of practical defenses by security experts.

3. **Integration and Customization:** Through the process of reverse engineering, developers can incorporate or modify hardware or software components into pre-existing systems to improve their operation or tailor them to meet particular needs.

4. **Recovering Lost Source Code:** Reverse engineering can be used to recover the source code of a software application that has been lost or is inaccessible or at the very least, to produce a higher-level representation of it.

5. **Fixing bugs and maintenance:** Reverse engineering can help find and repair flaws or provide updates for systems for which the original source code is either unavailable or inadequately documented.

Reverse Engineering Goals:

1. **Cope with Complexity:** Reverse engineering is a common tool used to understand and control system complexity. It gives engineers the ability to analyze complex systems and reveal details about their architecture, relationships and design patterns.

2. **Recover lost information:** Reverse engineering seeks to retrieve as much information as possible in situations where source code or documentation are lost or unavailable. Rebuilding source code, analyzing data structures and retrieving design details are a few examples of this.

3. **Detect side effects:** Understanding a system or component's behavior requires analyzing its side effects. Unintended implications, dependencies, and interactions that might not be obvious from the system's documentation or original source code can be found with the use of reverse engineering.

4. **Synthesis higher abstraction:** Abstracting low-level features in order to build higher-level representations is a common practice in reverse engineering. This abstraction makes communication and analysis easier by facilitating a greater understanding of the system's functionality.

5. **Facilitate Reuse:** Reverse engineering can be used to find reusable parts or modules in systems that already exist. By understanding the functionality and architecture of a system, developers can extract and repurpose components for use in other projects, improving efficiency and decreasing development time.

Reverse Engineering

Reverse Engineering to Understand Data:

Reverse engineering of data occurs at different levels of abstraction .It is often the first reengineering task.

1. At the **program level**, internal program data structures must often be reverse engineered as part of an overall reengineering effort.

2. At the **system level**, global data structures (e.g., files, databases) are often reengineered to accommodate new database management paradigms (e.g., the move from flat file to relational or object-oriented database systems).

Internal Data Structures

Reverse engineering techniques for internal program data focus on the definition of classes of objects.

1. This is accomplished by examining the program code with the intent of grouping related program variables.

2. In many cases, the data organization within the code identifies abstract data types.

3. For example, record structures, files, lists, and other data structures often provide an initial indicator of classes.

Database Structures

A database allows the definition of data objects and supports some method for establishing relationships among the objects. Therefore, reengineering one database schema into another requires an understanding of existing objects and their relationships.
The following steps define the existing data model as a precursor to reengineering a new database model:

1. Build an initial object model.

2. Determine candidate keys (the attributes are examined to determine whether they are used to point to another record or table; those that serve as pointers become candidate keys).

3. Refine the tentative classes.

4. Define generalizations.

Reverse Engineering to Understand Processing:

To understand processing begins with an attempt to understand and then extract procedural abstractions represented by the source code. To understand procedural abstractions, the code is analyzed at varying levels of abstraction :system, program, component, pattern, and statement.

1. Each of the programs that make up the application system represents a functional abstraction at a high level of detail. A block diagram, representing the interaction between these functional abstractions, is created.

2. Each component performs some sub function and represents a defined procedural abstraction. A processing narrative for each component is developed.

For large systems, reverse engineering is generally accomplished using a semi automated(partial automation) approach. Automated tools can be used to help you understand the semantics of existing code. The output of this process is then passed to restructuring and forward engineering tools to complete the reengineering process.

Steps of Software Reverse Engineering:

1. **Collection Information:** This step focuses on collecting all possible information (i.e., source design documents, etc.) about the software.

2. **Examining the Information**: The information collected in step-1 is studied so as to get familiar with the system.

3. **Extracting the Structure**: This step concerns identifying program structure in the form of a structure chart where each node corresponds to some routine.

4. **Recording the Functionality:** During this step processing details of each module of the structure, charts are recorded using structured language like decision table, etc.

5. **Recording Data Flow**: From the information extracted in step-3 and step-4, a set of data flow diagrams is derived to show the flow of data among the processes.

6. **Recording Control Flow:** The high-level control structure of the software is recorded.

7. **Review Extracted Design**: The design document extracted is reviewed several times to ensure consistency and correctness. It also ensures that the design represents the program.

8. **Generate Documentation**: Finally, in this step, the complete documentation including SRS, design document, history, overview, etc. is recorded for future use.

Reverse Engineering Tools:

Reverse engineering tools accept source code as input and produce a variety of structural, procedural, data, and behavioral design. Reverse engineering if done manually would consume a lot of time and human labor and hence must be supported by automated tools. Some of the tools are given below:

1. **CIAO and CIA:** A graphical navigator for software and web repositories and a collection of Reverse Engineering tools.

2. **Rigi:** A visual software understanding tool.

3. **Bunch:** A software clustering/modularization tool.

4. **GEN++:** An application generator to support the development of analysis tools for the C++ language.

5. **PBS:** Software Bookshelf tools for extracting and visualizing the architecture of programs.

Forward engineering and reverse engineering

Forward engineering and reverse engineering are two approaches to software development, with different goals and processes. Forward engineering involves creating new software systems from scratch using given requirements and design specifications. It focuses on building new applications through a structured process of analysis, design, implementation, and testing. Reverse engineering, on the other hand, involves analyzing and understanding existing software to extract its design, structure, and functionality. This approach is used to modify, improve, or document existing applications by working backward from the existing code.

What is Forward Engineering?

Forward Engineering is a method of creating or making an application with the help of the given requirements. Forward engineering is also known as Renovation and Reclamation. Forward engineering requires high proficiency skills. It takes more time to construct or develop an application. Forward engineering is a technique of creating high-level models or designs to make complexities and low-level information. Therefore this kind of engineering has completely different principles in numerous packages and information processes. Forward Engineering applies all the software engineering process that contains SDLC to recreate associated existing applications. It is near to full fill new needs of the users into re-engineering.

Characteristics of forward engineering

1. Forward engineering is a variety of engineering that has different principles in numerous package and information processes.

2. Forward engineering is vital in IT as a result of it represents the 'normal' development process.

3. Forward engineering deals with the conversion of business processes, services, and functions into applications.

4. In this method, the business model is developed first. Then, a top-down approach is followed to urge the package from the model developed.

5. Forward engineering tools are accustomed to moving from implementation styles and logic to the event of supply code.

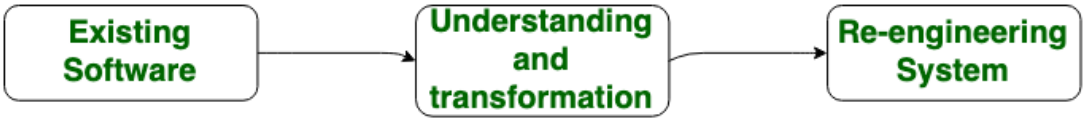
6. It essentially permits the user to develop a business model which may then be translated into data system components.

7. These tools follow the top-down approach. System creator and visual Analyst is a forward engineering CASE tool.

Forward Engineering

What is Reverse Engineering?

Reverse Engineering is also known as backward engineering, is the process of forward engineering in reverse. In this, the information is collected from the given or existing application. It takes less time than forward engineering to develop an application. In reverse engineering, the application is broken to extract knowledge or its architecture.



Reverse Engineering

Key Elements of Forward engineering and reverse engineering

Here are some key differences between the two:

1. **Goal:** The goal of forward engineering is to develop new software from scratch, while the goal of reverse engineering is to analyze and understand an existing software system.

2. **Process:** Forward engineering involves designing and implementing a new software system based on requirements and specifications. Reverse engineering involves analyzing an existing software system to understand its design, structure, and behavior.

3. **Tools and Techniques:** Forward engineering often involves the use of software development tools, such as IDEs, code generators, and testing frameworks. Reverse engineering often involves the use of reverse engineering tools, such as decompilers, disassemblers, and code analyzers.

4. **Focus:** Forward engineering focuses on the creation of new code and functionality, while reverse engineering focuses on understanding and documenting existing code and functionality.

5. **Output:** The output of forward engineering is a new software system, while the output of reverse engineering is documentation of an existing software system, such as a UML diagram, flowchart, or software specification.

In summary, forward engineering is focused on the creation of new software systems, while reverse engineering is focused on understanding and documenting existing software systems. Both approaches use different tools and techniques to achieve their goals and produce different outputs.