Software Engineering

Software Evolution & Maintenance

Part 4
(Program comprehension & Reverse Engineering)

T.A Mohammed Sultan

Modified from Sommerville's originals

Program comprehension

Program comprehension

- ❑ concerned with studying the way software engineers understand programs.

Objective of studying program comprehension

- ❑ design tools that will facilitate the understanding of large programs.

Program Comprehension Strategies

- **The bottom-up model**
- **The top-down model**
- **The integrated model**

The bottom-up model

- Comprehension starts with the source code and abstracting from it to reach the overall comprehension of the system.
- Steps:
  - ❑ *Read the source code*
  - ❑ *Mentally group together low-level programming details (chunks) to build higher-level abstractions*
  - ❑ *Repeat until a high-level understanding of the program is formed*

The top down model

- Comprehension starts with a general idea, or hypothesis, about how the system works
- Often obtained from a very quick look at what components exist
- Steps
  - ❑ *First formulate hypotheses about the system functionality*
  - ❑ *Verify whether these hypotheses are valid or not*
  - ❑ *Create other hypotheses, forming a hierarchy of hypotheses*
  - ❑ *Continue until the low-level hypotheses are matched to the source code and proven to be valid or not*

The Integrated Model

- Combines the top down and bottom up approaches.

- Empirical results show that maintainers tend to switch among the different comprehension strategies depending on

    - ❑ *The code under investigation*
    - ❑ *Their expertise with the system*

Partial program comprehension

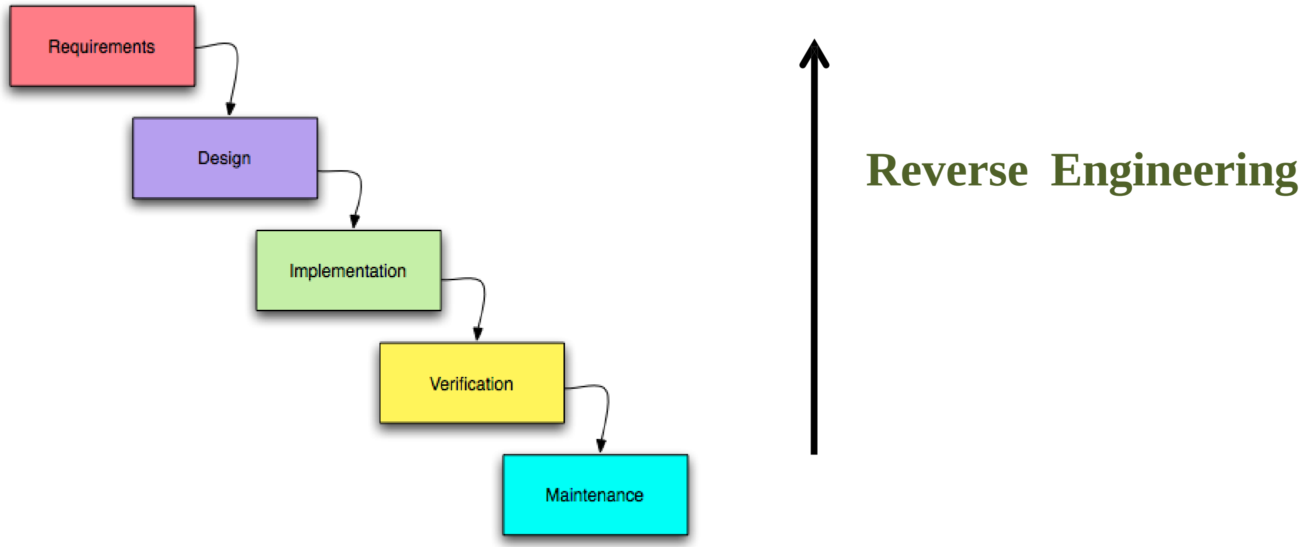
- Usually is not necessarily to understand the whole system if only part of it needs to be maintained. But a high fraction of bugs arise from not understanding enough!

Most software maintenance tasks can be met by answering **seven** basic questions:

- ❑ *How does control flow reach a particular location?*
- ❑ *Where is a particular subroutine or procedure invoked?*
- ❑ *What are the arguments and results of a function?*
- ❑ *Where is a particular variable set, used or queried?*
- ❑ *Where is a particular variable declared?*
- ❑ *What are the input and output of a particular module?*
- ❑ *Where are data objects accessed?*

Reverse Engineering

- *Process of analyzing a subject system to create representations of the system at a higher level of abstraction.*
- *Going backwards through the development cycle.*



Two main levels of reverse engineering

<u>Binary</u> reverse engineering

- *Take a binary executable*
  - *Recover source code you can then modify*
- *Useful for companies that have lost their source code*
- *Used extensively by hackers*
- *Can be used legally, e.g. to enable your system to interface to existing system*
- *Illegal in some contexts*

<u>Source code</u> reverse engineering

- *Take source code*
  - *Recover high level design information*
- *By far the most widely performed type of reverse engineering*

Reverse engineering objectives

Cope with complexity

- *Have a better understanding of voluminous and complex systems*
- *Extract relevant information and leave out low-level details*

Generate alternative views

- *Enable the designers to analyze the system from different angles*

Recover lost information

- *Changes made to the system are often undocumented;*
  - *This enlarges the gap between the design and the implementation*
- *Reverse engineering techniques retrieve the lost information*

Reverse engineering objectives

CDetect side effects

- Detect problems due to the effect a change may have on the system before it results in failure

Facilitate reuse

- *Detect candidate system components that can be reused*

Source code reverse engineering techniques

- Program analysis
- Program slicing
- Design recovery
- Architecture recovery

Source code reverse engineering techniques

Program slicing

- concerned with analyzing all the statements in a program that may
  - Affect the value of a given variable at a certain point during execution
    - Looking backward
  - Be affected by the execution of a certain statement
    - Looking forward
- Can be either static or

Static slicing

- Considers all possible inputs
  - the resulting slices are usually quite large

Dynamic slicing

- Extracting parts of the program that contribute to the computation of the function according to a specific input

Source code reverse engineering techniques

Design recovery

- Create design abstractions in order to understand what a program does, how it does it and why it does it.

- Examine multiple sources of knowledge including:
  - ❑ the system documentation (if available),
  - ❑ the knowledge that the software engineers have of the system

- Difficult to perform on large systems that have undergone ad-hoc maintenance for a long period of time
  - ❑ Documentation of such legacy systems is usually out of date
  - ❑ Original developers are usually no longer working within the organization

Source code reverse engineering techniques

Architecture recovery

- Aims to recover the overall system structure in terms of its high-level components and the way they interact

- There are several techniques
  - ❑ Using human experts
  - ❑ Recognizing known patterns
  - ❑ Static and dynamic analysis
  - ❑ Clustering and data mining

Reverse engineering tools

- The process of reverse engineering is accomplished by making use of some tools that are categorized into:
  - ❖ **Disassemblers** – A disassembler is used to convert binary code into assembly code and also used to extract strings, imported and exported functions, libraries etc. The disassemblers convert the machine language into a user-friendly format. There are different disassemblers that specialize in certain things.

  - ❖ **Debuggers** – This tool expands the functionality of a disassembler by supporting the CPU registers, the hex duping of the program, view of stack etc. Using debuggers, the programmers can set breakpoints and edit the assembly code at run time. Debuggers analyze the binary in a similar way as the disassemblers and allow the reverser to step through the code by running one line at a time to investigate the results.

Reverse engineering tools

- **Hex Editors** – These editors allow the binary to be viewed in the editor and change it as per the requirements of the software. There are different types of hex editors available that are used for different functions.

- **PE and Resource Viewer** – The binary code is designed to run on a windows based machine and has a very specific data which tells how to set up and initialize a program. All the programs that run on windows should have a portable executable that supports the DLLs the program needs to borrow from.