Program Comprehension Report for Chemistry Calculator

Course Title: Software Maintenance Lab
Course Code: SE4204

Team Members

Abdullah An-Noor (ASH1825001M)
Fazle Rabbi (ASH1825004M)
Rahat Uddin Azad (ASH1825022M)
Saifur Rahman (ASH1825031M)
Anwar Kabir Sajib (ASH1825038M)

Submitted To

Dipok Chandra Das
Assistant Professor
IIT, NSTU

Date of Submission
02-March-2023

Table of Contents

List of Figures

1. Program Understanding of Chemistry Calculator

To Understanding of Chemistry Calculator , We follow the following step :

1.1. Read about the program

At this stage of the process, the 'understander' browses, peruses different sources of information such as the system documentation - specification and design documents - to develop an overview or overall understanding of the system. Documentation aids such as structure charts and data and control flow diagrams can be used.

In this project we have only source code, there are no system and specification documentation, design documents etc. that we can read about the program. So, this phase is omitted because we don’t have any document for Chemistry calculator.

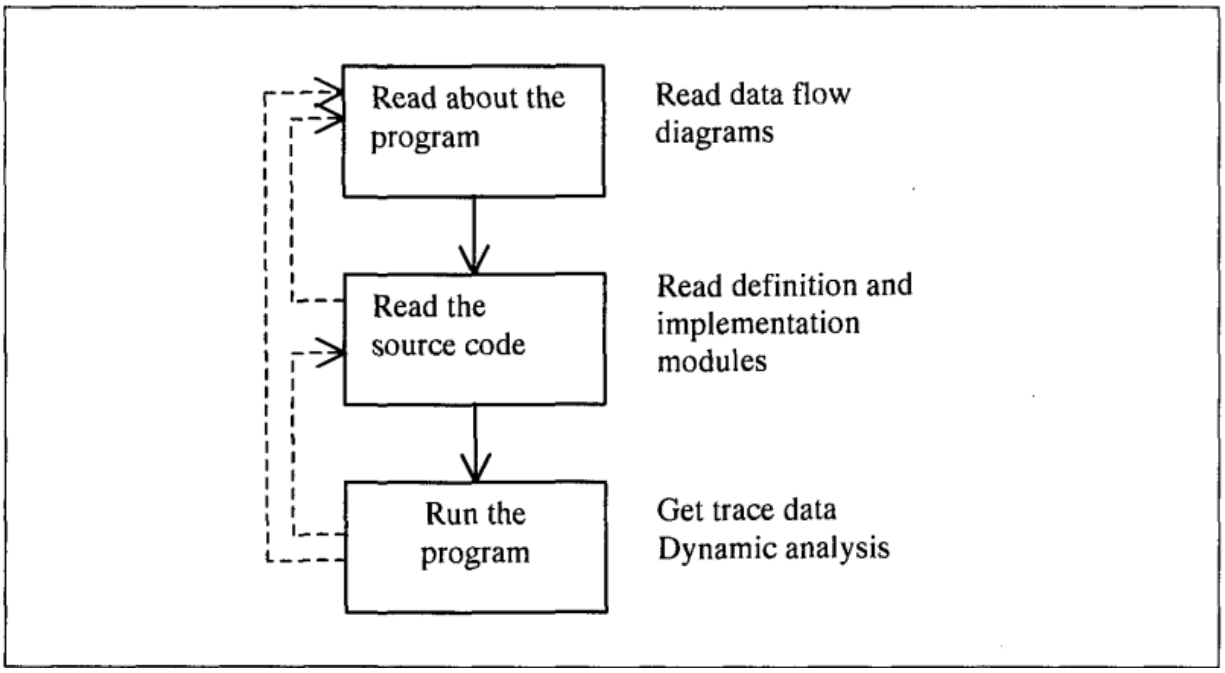


Figure 1: Program Comprehension

1.2. Read about the source code:

Information about the system's structure, data types and algorithmic patterns is obtained. They produce cross-reference lists, which indicate where different identifiers - functions, procedures, variables and constants - have been used (or called) in the program.

For Chemistry calculator , We read the source code, to find out which pattern are used in Code. Find out the tools, framework, programming language, database etc. are used to implement Chemistry Calculator. Find out the list of functions, procedures, variables and constants - have been used in the program.

1.3. Run the program:

The aim of this step is to study the dynamic behavior of the program in action, including for example, executing the program and obtaining trace data. The benefit of running the program is that it can reveal some characteristics of the system which are difficult to obtain by just reading the source code.

After reading the source code of Chemistry calculator, next step is running the program. In this step ,we know how the chemistry calculator work , behaviour of the chemistry calculator, trace the data for documentation.

2 Program Comprehension Model:

For chemistry calculator , We follow Mental Model to understand the project.

2.1. Mental Model :

A mental model describes a programmer’s mental representation of the program to be comprehended. Consequently, a mental model of a program is not unique. Rather, different programmers may have different mental models depending upon their own knowledge and interpretation about the same program.

After running the Chemistry calculator source code , we know how to chemistry calculator work, the behaviour of the chemistry calculator and gain knowledge about the chemistry Calculator.

2.2. Program Comprehension Strategies :

A program comprehension strategy is a technique used to form a mental model of the target program. The mental model is constructed by combining information contained in the source code

and documentation with the assistance of the expertise and domain knowledge that the programmer brings to the task. A number of descriptive models of how programmers go about understanding programs have been proposed based on the results of empirical studies of programmers. Examples of Program Understanding these models include:

1. Top-down ,
2. Bottom-up and
3. Opportunistic models

For Chemistry Calculator , We used Bottom Up comprehension strategies.

Here are we follow some steps in the bottom-up program comprehension Strategies:

Identify the smallest units of code: Firstly, Start by identifying the smallest units of code in the software, such as individual functions, methods, or code blocks. Analyze these units to understand their purpose and how they work.

Understand the relationships between the code units: Once when we have identified the individual units of code, analyze how they are related to each other. Then we Identify the data and control flow between the units to understand how they work together to achieve the overall functionality of the software.

Identify the larger components: After that we have a good understanding of the smaller units of code and how they work together, then we can identify the larger components of the software, such as classes, modules, or subsystems.

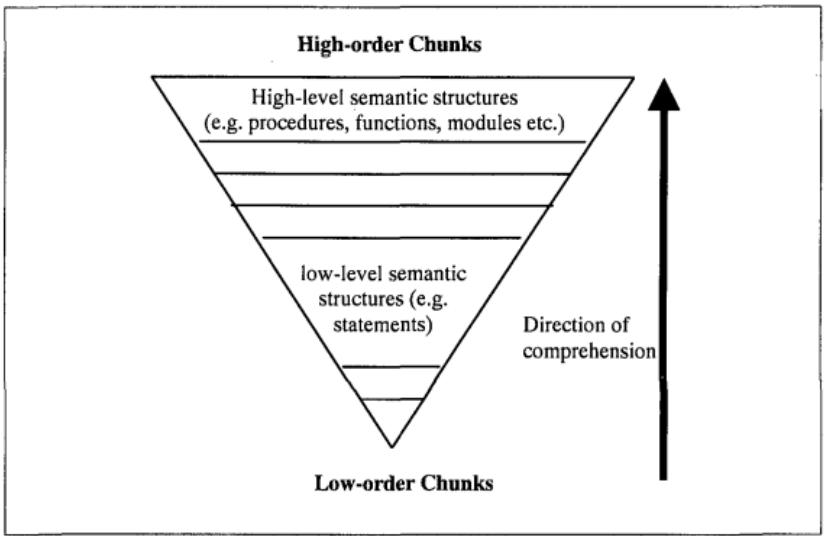
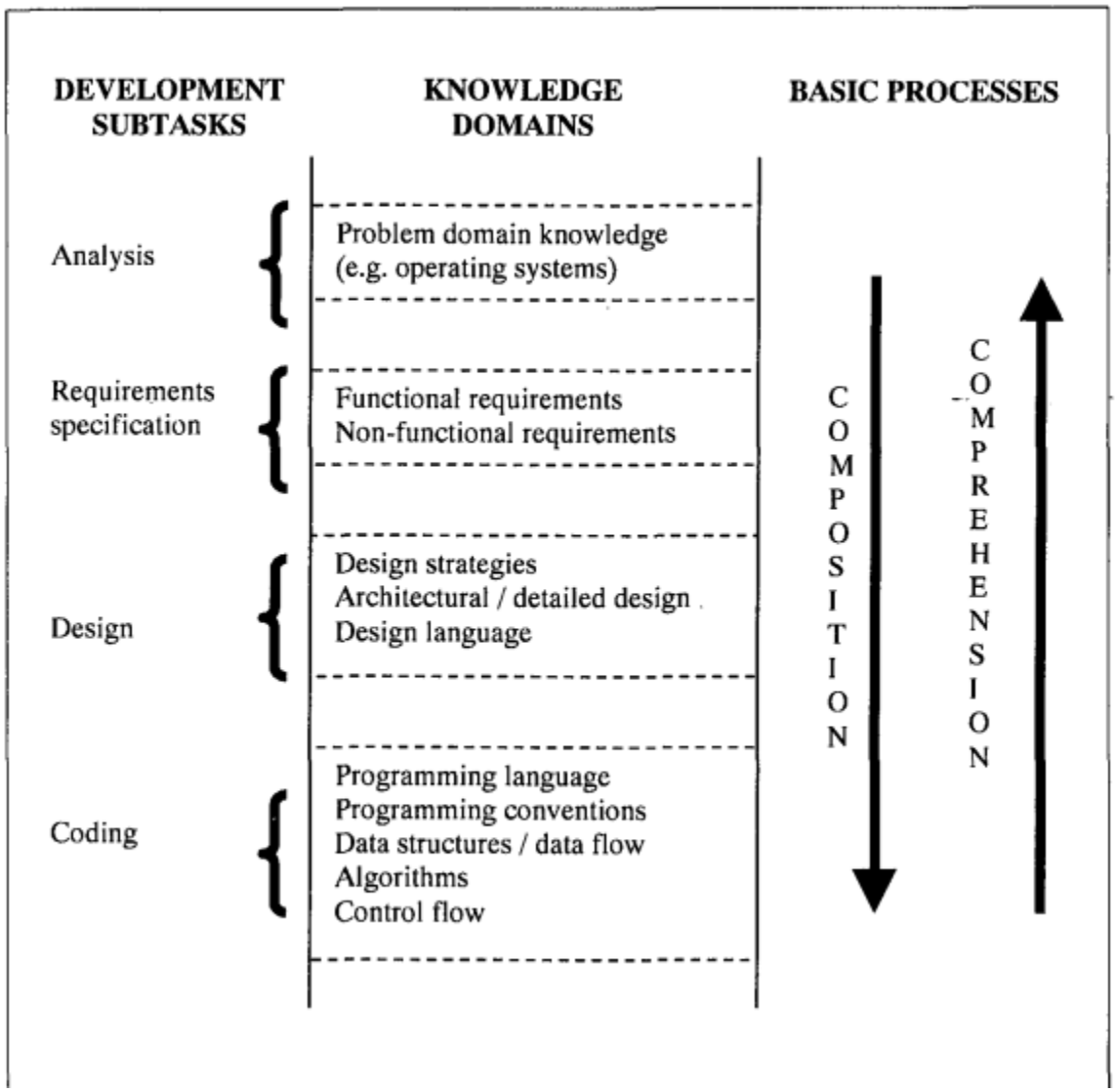


Figure 2: Bottom-up Approach

Build up the understanding of the system: With a good understanding of the smaller units and larger components, we build up the understanding of the system as a whole. Identify how the components work together to achieve the overall functionality of the software.

3. Reverse Engineering:

Reverse engineering is a process used in software maintenance to understand and analyze existing software systems. In software maintenance, reverse engineering is typically used to extract information from legacy systems that may be poorly documented, difficult to understand, or no longer supported by the original developers.



The process of reverse engineering involves analyzing the existing software system to understand its components, structure, and behavior. This can involve a variety of techniques, including code analysis, data analysis, and system architecture analysis.

The goal of reverse engineering in software maintenance is to gain a better understanding of the software system, so that it can be maintained, modified, or extended more easily. Reverse engineering can help software engineers to identify bugs, understand how the system works, and identify areas where the system could be improved or modernized. Reverse engineering can also be used to migrate legacy systems to new platforms or technologies.

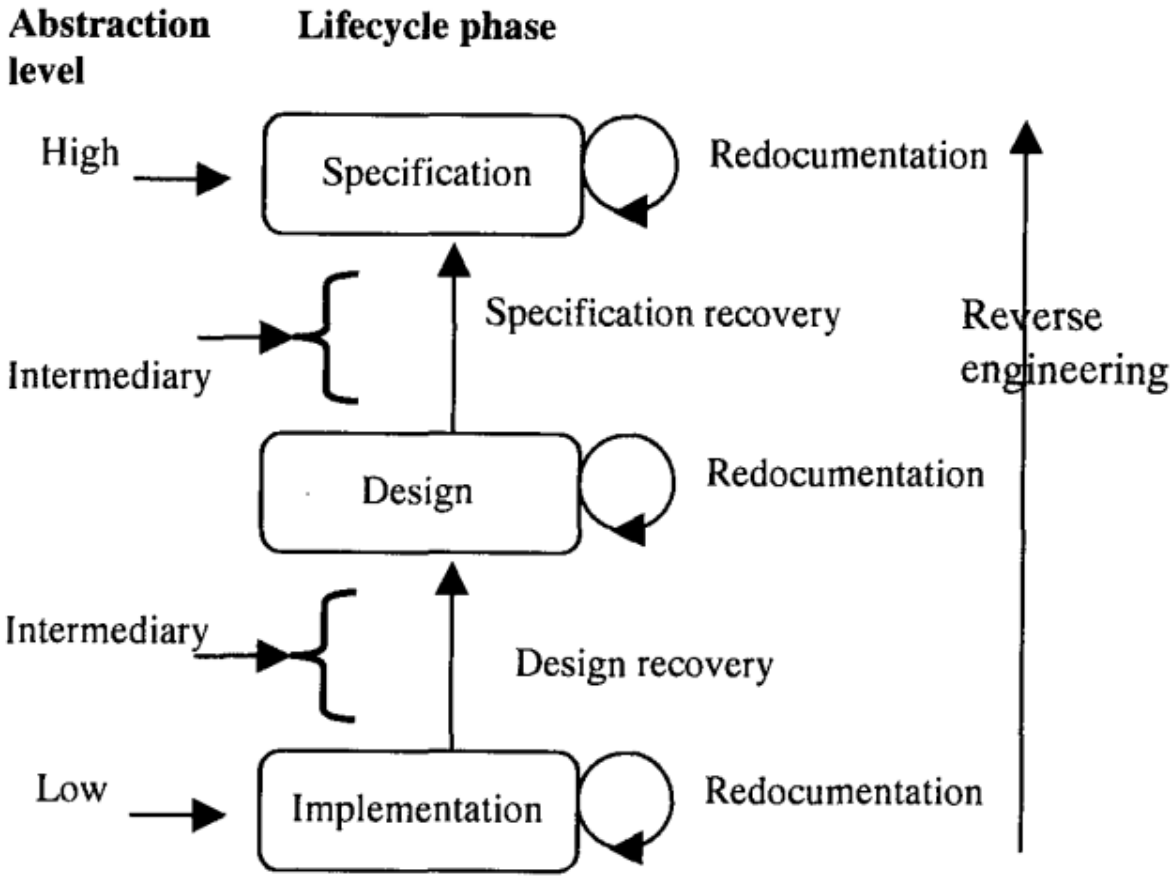


Figure 3: Level of Abstraction

3.1. Level Of Abstraction of Chemistry Calculator

Abstraction level	**Product**
Low Level	Source code Test Cases User interface Database User Manual
Mid Level	Details Design Specifications Use case Diagram Activity Diagram Database Schema UML diagram
High Level	Project objectives and characteristics Project stakeholders Functional Requirements Non-Functional Requirements Tools and Technology Configuration management plan Requirement Traceability Matrix(RTM)

Table – 01:Level of Abstraction