Unit 3 – Chapter 9

Development

Prerequisites:

Software development life cycle, software requirements, software design

Unit Outcomes:

The objective of this chapter is to introduce the concepts of software implementation. At the end of this chapter, students will be able to:

- Discuss the factors in coding.
- Understand the importance of reuse
- List the various coding guidelines
- Summarize the steps in the code documentation

9.1 Introduction

Software implementation is an important step in SDLC, that results in an executable version of the software. Software coding or implementation includes developing a program using a high- or low-level programming language. The important aspects of software implementation are as given below:

9.1.1 Selection of a language:

The programming language used in software development is a critical part. The choice of language depends on several factors such as:

a) For what application the software is used? For example, android mobile app, or a system software in an embedded system, web-based software, database-oriented tool, graphical software, etc. On what platform the program should run is an important decision. If the application is developed in C, it needs two executable forms on Windows and Linux operating systems, whereas if the program is in Java is platform-independent, and it can run using the installation of Java virtual machine (JVM).

b) Use of new technology that is popular in the latest market. For example, the use of the current version of the operating system, internet, and other tools.

c) What is the time necessary for software launch? The number of lines of code, man-power, and duration of development depend on the programming language.

d) Is the programming language used in software support scalability? If there is a need for expansion, then how much time and space it is going to occupy also defines the goodness of the programming language.

e) What security measures are required for the protection of the developed software from the outside world? This refers to licenses, copyrights, and other security measures.

f) Efficiency of the program is decided by the development time and the amount of memory space. The lesser the time and space more is the efficiency of the software.

e) Error checking and debugging diagnostic tools must be present in the programming languages which makes it is easier to check the errors and make the software more robust.

9.1.2 Reuse of a software

The programs are written must fulfill all the requirements that are listed in requirement specifications and utilize or adapt the existing available software components commonly known as commercial-off-the-shelf (COT). The COT software components are readily available, and they can be bought and modified for the specific software. For example, a student record maintenance software related to admissions, results, library records, hostel detail, etc. can be purchased and updated with a few modifications to suit the specific requirements of a college or university This requires lesser time than developing the same from the scratch using any programming language.

The reuse can be at different levels as given below:

a) The abstraction level: The different abstractions of software in architecture and design patterns are reused to build software.

b) The component level: The available software components such as front ends, GUI, tables, database structure, etc are reused.

c) The object level: The existing library files are used to export objects and files rather than develop repeatedly.

d) The system-level: An entire existing system is modified and refined to build new software in this approach.

The main advantage of reuse is lower costs for software development. Reuse also reduces time and that is why most software developers look for the existing available software components at the start of the coding phase. However, sometimes the cost of mapping the new software to an old one may be more in some applications. Therefore, the reuse of software depends on the type of application.

9.1.3 Configuration Management

There will be multiple versions of the software during the development process. Keeping track of the latest version of the software is critical. In software development, when a team of people is working, there must be coordination for accommodating the changes in the software. If there is a problem or error in the new system, they should be able to go back to the older version. Configuration management is a process of managing the changes during software development. There are three main activities in configuration management.

a) Version management: Keeping track of different versions of software.
b) System integration: This is support necessary for combining different version components to form a single system.
c) Problem tracking: Tools necessary for error detection and correction.

9.1.4 Host-target Development

The computer system used during the development process may not be the same as the host computer. This needs to take care to resolve the differences between the development system and the actual customer's computer system environment. The environment may refer to hardware, operating system, database management system, web, cloud support, etc. Sometimes the development platform and the customer's host environment may be the same. This makes the launch of a new software very easy. But most of the time, they are different, and the developers must move the software to the host platform for testing or make a toy model for testing. For example, simulators are commonly used for developing embedded systems. Also, if the target software is some part of already developed software, then it is not possible to install the software on the new system, it must be transferred to the customer's system and tested.

The different tools that form a software development platform include compiles, testing tools, editors, supporting tools, etc. These tools can be combined under one unit popularly known as an integrated development environment (IDE). The purpose of IDE is to coordinate all the activities in coding and testing.

9.2 Coding

The coding phase of software development refers to converting the design of a system into a high-level language program. There are standard coding practices that make the code uniform though it is managed by a team of different programmers. Each programmer must have a good knowledge of the programming language, databases, web, cloud platform, mobile app, and any other tools related to the problem. Some of the standard requirements in coding are as given below:

a) A code with a modular approach is always preferred as it gives ease of understanding and debugging. An integrated collection of compiled modules gives better performance than a single large program.
b) The notations used in the coding must be common and familiar so that many developers can understand.
c) Portability and generality are two critical features that make the standard coding style. The developed software should be portable and generic without very rigid constraints.
d) Error checking and debugging must be handled efficiently as there can be many exception handling situations while programming.
e) The size of the code should be concise that making the execution faster and consuming less memory.
f) The cost of the code depends on the functionalities and efficiency. It also depends on various technical processes such as web development, front and back end, user interface, etc.
g) The programming language must be available on all the machine platforms. If there is any modification or obsolete feature, then there should be translators to such cases.

Any good software development firm expects its developers to practice a standard coding style. The coding standards are decided by the organization by considering the type of software they are developing. If the coding standard is not maintained, then the code might get rejected in the process of code review. Coding guidelines dictate the standard coding styles and good programming, thus resulting in a good quality software product.

9.3 Coding Guidelines:

The coding transforms the system design to code in a high-level language. Coding also includes a part of testing the code. The most common coding practice is to follow the well-defined and standard coding style. The standard coding technique result in a uniform, reusable, and good software product. Coding guidelines provide common knowledge information for using coding style, but the actual implementation of the code is up to the individual. Some of the coding guidelines are discussed in brief.

a) The size of the code should be small, and the code should be easy to understand. Too clever code may be difficult to debug and understand and thus system testing and maintenance becomes difficult. If the size of the functions used in code is more than 10-15 lines, it defeats the purpose of dividing a program into several functions and makes error detection very challenging.
b) Nature of variables such as global and local with block and file scopes must be determined. This will help in limiting access to different users in the system.
c) It is important to mention the name of the module, date of creation, name of the author, a brief synopsis, modification details in the past, etc. Guidelines documents include information regarding different jobs of each module, input and output parameters, conditions, assumptions, and effect of operations on global and local variables.
d) Naming conventions are yet another important guideline in coding. The most used technique is naming variables with alphabets, digits, and underscore, where alphabets may be a combination of uppercase and lowercase letters. Constants are generally expressed in capital letters. E.g., *Student_Name* (Variable name), MAX(Constant)
e) While handling errors in the code the standard guideline is to return 1 or 0 based on the condition. This will give a uniform coding style, thus can be reused by different programmers.
f) Many unconditional statements such as go-to statements should be avoided. The use of the go-to statement makes the program difficult to understand and debug.

Many programmers use the same identifier to store temporary results, this can confuse and lead to wrong results. It is always good practice to use variable names that are more relevant to the usage than random names. E.g., *temp_loan_value* is preferred over just *temp*.

g) Side effects of the function call are very critical. A side effect of a function code is a modification of the parameters that are outside its definition. Some I/O operations or parameters passed by reference may create an ambiguous alteration of the values and generate wrong results.

h) Code documentation: The code must be with sufficient comments explaining its purpose. The rule of thumb is at least one comment must be there for an average of three lines of code.

9.4 Coding Documentation:

Coding documentation is an effective means of connecting humans and machines. The purpose of documenting code is it is helpful to the developer as he/she will be using the code for months, sometimes for years in case of lengthy projects. In such cases, the documentation helps as a guide. Documentation also helps other people to continue the work further. It shows the contribution of the person who is developing. Other people can modify or build upon the existing software using the documentation. It helps in promoting open-source software and makes the software transparent. The major steps in software documentation are:

a) Give a brief introduction of the project that consists of the installation procedure, operating system, computer hardware requirements, a brief sample, etc. This is commonly known as the README file.
b) Explain the functionality of each module, the parameters, their type, return value.
c) In documenting the code, discuss naming conventions for files, variables, constants, which programming practice is used, various comments on a database or web-based apps used.
d) Name the contributors in a team and their part in coding by giving proper citations.
e) Give details of the license and version of the software developed.
f) Include the roles of the developers, their name, and email address for future contact.

g) Mention the version of the developed software and the editing performed.

Example of Coding Documentation: A sample of steps in code documentation for the development of the project: *Hospital Management System* is shown in Figures 9.1 and 9.2.

README FILE

HOSPITAL MANAGEMENT SYSTEM

Description

Prerequisites

Installation Guide

License and contact us

Figure 9.1. README File for project titled *Hospital Management System*

Function to insert a new patient record

int Insert_Patient(int Patient_waiting_no, char Patient_Name, char Doctor_Name)

Description

Inserts a new patient name and number for a doctor

Parameters

Patient_Waiting_no is an integer, Patient_Name and Doctor_Name are of string data type

What Function returns

Adds the record and returns the patient number after inserting a record.
This number is unique and can be used in all the other modules

Figure 9.2. Insert function for project titled *Hospital Management System*

9.5 Code Review

Code review is done after the compilation of a coding module. Review can identify the algorithmic, logical, and poor programming code. It does not check the syntax errors. It is different than testing as the testing is applied on executable code. Review is a process where different team members working on software check for any problems or defects. The software review helps in improving the quality and working of the software. The review may be used in design and implementation and various other steps in software development. The code review may be done by software developers, testing engineers, or quality control managers. Use of code review is discovering any errors and rectifying them at the early stage, If the errors are identified at the beginning stage, then the work of debugging them in later stages is less time-consuming. Code review at this stage can identify the performance of software related to the following.

- Are the requirements covered in various test cases?
- Is the software design consistent?
- Is the developed code following the standard coding style?
- Are there enough test cases to verify the developed software?
- Are there any security threats to the software?
- Is the code documentation complete?
- Are there any major drawbacks or flaws in the software? For example, Insufficient memory, insufficient data, memory overflow/underflow, etc. Whether all the syntax and logical errors are handled or not? Are there any performance defects in the developed software?

9.6 Types of code reviews

There are two types of code reviews carried out on the module:

Code Walkthrough: The purpose of a code walkthrough is to find out if there are any algorithmic and logical errors. In this process, each team member is given a sample test case. Each member must go through the code and find out whether the code is working and producing desired results or not. This is done in the presence of the coder who has developed that code. Every module in the developed software is traced by the members by hand for different test cases. The minutes of the code walkthrough is recorded and used as coding guidelines. It is an informal method but perron's experience, knowledge about the domain, common sense, etc. is used in the code walkthrough.

Code Inspection: Unlike code walkthrough, in this process, some common programming errors are examined. The choice of algorithm, techniques used in programs, library files usage, coding standard, etc. are checked in this step. In code walkthrough, code execution is performed to check the results, whereas, in this step, the way of programming, common mistakes such as infinite loops, memory allocation, type mismatch of actual and formal parameters, incorrect use of priority of operators, out of bound arrays, cases of dangling pointers, etc. are identified.

Thus, after the completion of software design and its review, coding is undertaken. Every module in the design is programmed and tested. This testing is also known as unit testing. That is each function or module is tested independently.

9.7 Multiple Choice Questions (MCQs)

1. Transforming a software design using a programming language is a phase known as:
    a. Software requirement Analysis
    b. Software Development
    c. Software Design
    d. Object-oriented Design
2. Modification of parameters that are outside of a function in coding is known as:
    a. Version Mismanagement
    b. A side effect
    c. Type mismatch
    d. Run time error
3. The ease of debussing and understanding in coding can be managed by:
    a. Modularity
    b. Cohesion
    c. Coupling
    d. Functional Design
4. Algorithmic and logical errors are detected in the phase of:
    a. Coding
    b. Data Structures
    c. Code Walkthrough
    d. Code Inspection
5. Including library files in the code to be developed is done with:
    a. Commercial Off the Shelf (COTS)
    b. Functions
    c. Integration

d. Testing

9.8 Review Questions

1) What is software configuration management?
2) What essential problems in coding are identified by code review?
3) Explain the different errors that may get detected in code inspection

1.9 MCQ solution keys

1: b	**2: b**	**3: a**	**4: a**	**5: a**