Chapter 1

Introduction

1.1 The Role for Software Engineering

The role of software engineering is to introduce the notion of software as a product designed and built by software engineers using effective methods. Software is important because it is used by a great many people in companies and institutions. Software engineers as programmers should have a moral and ethical responsibility to ensure that the software and design they provide do not cause any serious problems. Software engineers tend to be concerned with the technical elegance of their software products and tools, whereas customers tend to be concerned only with whether a software product meets their needs and is easy and ready to use.

For two years, I was a software programmer for Hercules Missile Defense, where my responsibility was to make sure that the software that is available for implementation and use was programmed and performed according to requirements.

In order to develop, operate, and maintain software engineering capabilities, a major discipline in supporting software programming, design, builds, and delivery to customers should be completely understood. A critical understanding of and a right approach for the methods used for software engineering is necessary to enhance the process and achieve effective programming results.

The right disciplines are identified in Figure 1.1.

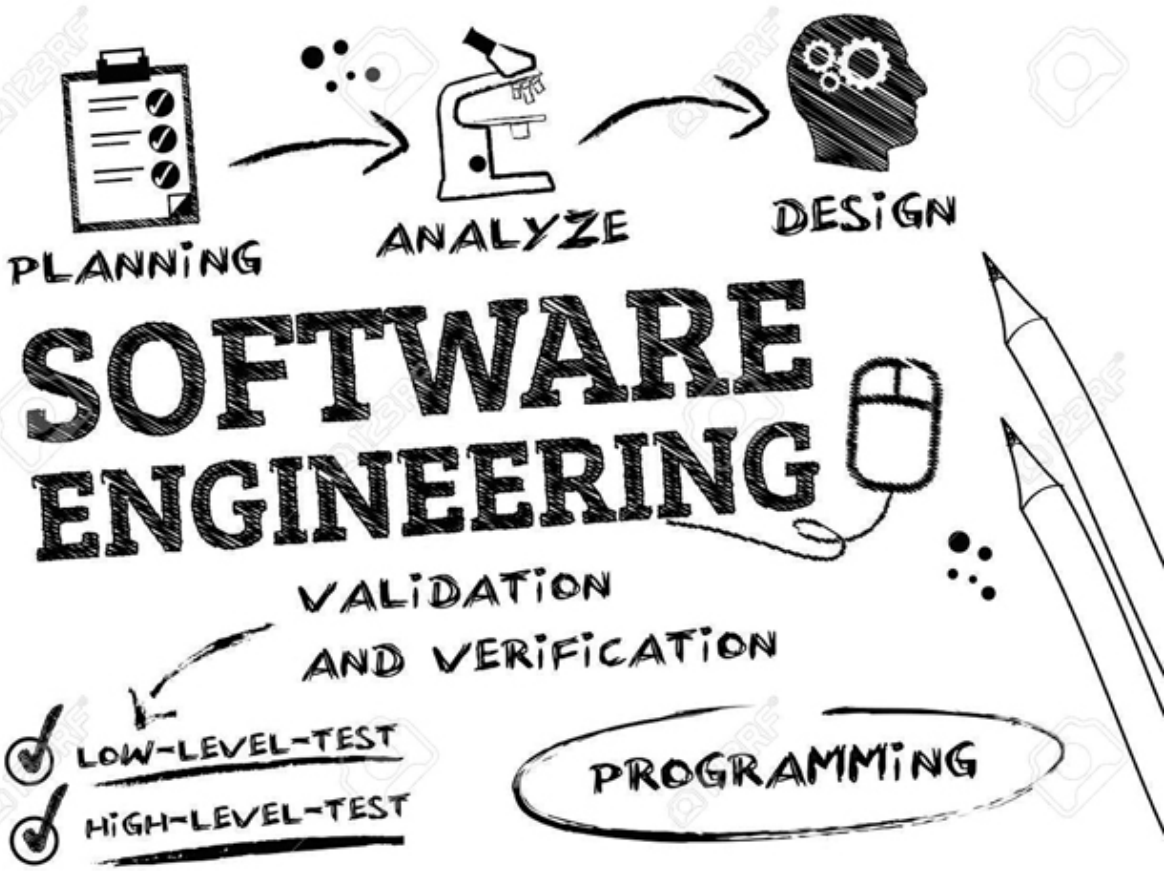

Figure 1.1 Right approach for software engineering.

It is more than a discussion and a way of life for software engineering to be important related to the standards and requirements for programming and design. Software engineering must comply with these standards and requirements during several stages of the development process. The standards and requirements adopt and deploy the following:

- Understand the basic concepts, standards, and requirements of software engineering.
- Select appropriate techniques for programming and designing.
- Effectively use software engineering tools and applications.
- Create specifications to comply with the software standards and requirements.
- Utilize various standards and requirement techniques to identify defects.
- Manage changes to standards and requirements.

1.2 Software Engineering Process

This section discusses the concept of a software process framework and provides a brief overview of the software engineering Capability Maturity Model Integration (CMMI). It is important to emphasize that CMMI does specify which process model needs to be used for a given project or organization. Rather, it provides a means of assessing how well company and institutional processes allow them to complete and manage and program new software projects.

Software engineering divides the process of software development into distinct phases to improve programming and designing, which is known as “software development life cycle”.

Most modern software engineering processes can be described as Agile. Other methodologies include waterfall, prototyping, iterative and incremental development, spiral development, rapid application development, and extreme programming.

Software engineering considers “life cycle model” as a more generic term for a category of methodologies and “software engineering development process” as a more specific term to refer to a process chosen by a company or an institution. There are many specific software engineering processes that fit the life cycle model.

Software engineering is the process of envisioning and defining software solutions to one or more sets of problems. One of the main components of software engineering requirement (SER) is part of the software development process that lists specifications used in software engineering. If the software is semi-automated for software engineering, it can help determine those specifications. If the software is completely automated, software engineering may be as simple as describing a planned sequence of events. In either case, a software plan is usually the product of the software engineering and design. Furthermore, software engineering may either be platform-independent or be platform-specific depending on the availability of the technology used for programming processes.

1.3 Software Engineering Planning

It is very important for software engineering programmers to have some basic knowledge of planning and performing software projects. Software engineering planning involves estimating how much time, effort, and resources are required to build a software product. There is enough information provided in the text that allows programmers to estimate their own projects and write their own planning documents. Software engineering programmers should be assigned the task of writing planning documents using appropriate templates as part of their coursework roles.

First, management and software engineering programmers must prepare for planning by identifying contracts and requirements evaluated during a specific period for companies, institutions, military programs, or successful businesses. Then, the identified contracts and requirements require the right criteria derived from associated software plans, documents, procedures, and work instructions.

1.4 Software Applications

Software applications are different from the artifacts produced in most other engineering disciplines. Opportunities for replication without customization are not very common. Software may deteriorate, but it does not wear out. The chief reason for software engineering is that many changes are made to software products over their lifetime. When changes are made, defects may be inadvertently introduced to portions of the software that interact with the portion that was changed. The most popular tools used by the military and defense programs are Rational ClearCase and ClearQuest. I implemented these tools to many Boeing Defense and Space programs to be used by software engineers, software configuration managers, and testers, and I will explain the use of these two tools in Chapter 6.

1.5 Software Engineering Programming

Software engineering programming encompasses preprogramming compilers that effect the translation of languages from a variety of editors to software codes to be used by software engineers in their tools. Software engineers that do programming can be important for software design and delivery. Programming implements design characteristics for the programming language used. There are two design topics used:

- Data design.
- Procedural design.

The strategy for software engineering programming is illustrated in Figure 1.2.

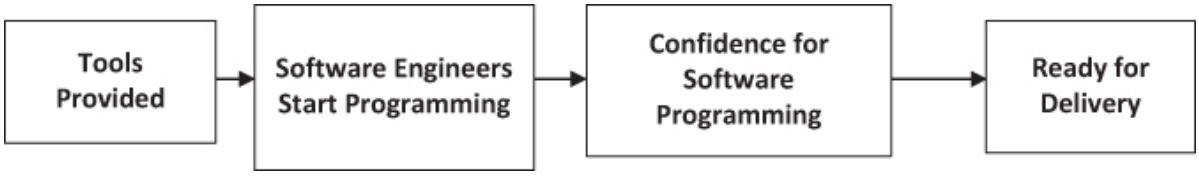

Figure 1.2 Strategy for software engineering programming.

Economies of the developed software are dependent on software engineering. In recent days, more and more systems are software-controlled and concerned with the methods and tools used for professional software engineering programming and development.

The elements of software engineering are as follows:

- Software requirements and software design.
- Software testing and maintenance.
- Software configuration management (SCM) builds, plans, and documents.
- Software engineering management.
- Software engineering plans, processes, and procedures.
- Software engineering tools and methods.
- Software quality engineering reviews and buy-offs.

When poor Software Engineering methods and designs are not effective; that can lead to major problems with management, team members, employees and customers.

Boyd L. Summers

Summary

Software engineering is a detailed study of designing, programming, developing, and maintaining software products. It is introduced to address the issues that arise due to low quality of software products. Problems arise when development of a software product exceeds its timeline and/or it has reduced levels of quality. Software engineering ensures that the application is built consistently, correctly, on time, and according to requirements. The demand of software engineering also emerges to cater to the immense rate of change in user requirements and environment on which applications are to be working (Figure 1.3).

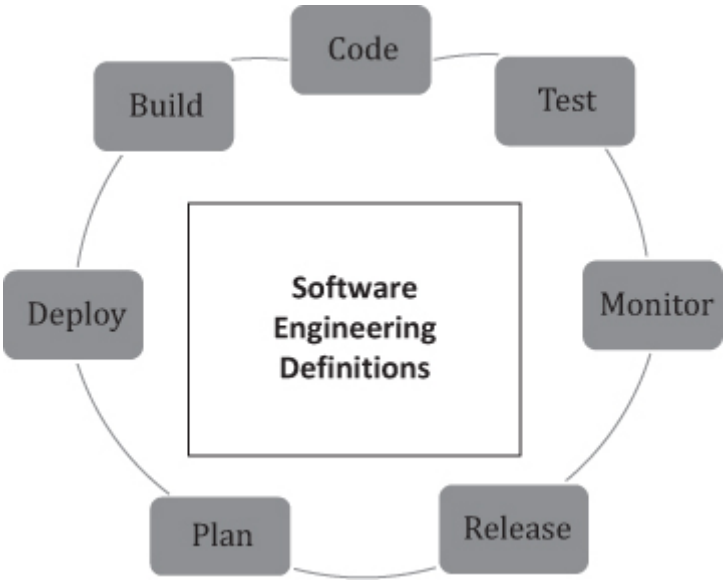

Figure 1.3 Software engineering definitions.

Further Reading

Long, L. N., lnl@psu.edu 2005. www.personal.psu.edu/lnl.

Pressmen, R. S., *Software Engineering Book: A Beginner Guide*. 1988. ISBN 0-07-050790-1. www.ebay.com/itm/143051317925.