Mathokoza Mntambo

ID: UB76859SCO86054

Bachelors in Computer Science

ASE 065: Advanced Software Engineering

Atlantic International University
Honolulu, Hawaii

Date: 19th August 2022

TABLE OF CONTENTS

Introduction.

Software Engineering is a disciplined, systematic, quantifiable research and tactic to the design, evolution, operation and support of a software system. The sphere of software engineering pertains the structured, disciplined approach to programming that is utilized in engineering to software evolution with the stated objectives of improving the quality, budget and time efficiency.

Software engineering is naturally used for intricate and large software systems instead of single programs or applications. Development, though, is simply one stage of the process. Whereas a software engineer is basically in charge of the design of systems, programmers are repeatedly in charge of coding its execution.

Software engineering includes a number of areas that cover certification and the process of engineering software including: software design, requirements gathering, software construction, maintenance, configuration management, engineering management, software development process creation and management, software engineering methods and models, software quality as well as foundational computing, engineering and mathematical study.

Literature Review.

The word software was not utilized until the late 1950s. Throughout this time, although different kinds of programming software were being built, they were naturally not commercially offered. Therefore, computer users (generally scientists and big enterprises) had to create their personal software. Renowned computer science and mathematician pioneer, Margaret Hamilton, is accredited with having created the term *software engineering* while making the navigation and guidance system for the Apollo spacecraft as leader of the Software Engineering Division of MIT Instrumentation Laboratory.

Software engineering was motivated by the alleged software calamity of the 1960s, 1970s and 1980s, which recognized several of the problems of software development. Countless software projects ran over schedule and budget. Some projects triggered property damage. A few projects triggered loss of life. Some utilized the phrase software crisis to denote their failure to hire trained programmers. The software crisis was initially defined in terms of productivity but changed to stress quality.

Software engineering.

Software engineering has two words: software and engineering. *Software* is a group of codes, records and initiations that does a precise job and fulfills a specific requirement. *Engineering* is the creation of products using best principles, practices and methods. Software engineering is a process of studying user requirements and then designing, developing and testing software which will please that requirements.

Software Engineering Objectives:
Maintainability – It must be practicable for the software to change to meet varying requirements.
Efficiency – The software must not make wasteful utilization of computing devices like, processor cycles, memory etc.
Correctness – A software product is accurate if the different requirements as stated in the SRS document have been appropriately implemented.
Reusability – A software product has fine reusability if the different segments of the product can be easily reused to create new products.
Reliability – It is a trait of software quality.
Portability – The software can be moved from one environment or computer system to another.
Adaptability – The software lets different system constraints and the operator needs to be pleased by making variations to the software.

Main Challenges in Software Engineering
Fast technology advancement - Technology developing at an extraordinary rate leads to an extra pressure for software engineering.
Growing customer demands - Software projects are commonly theoretical and are aimed at developing and designing software products that meet diverse customer demands.

Time limitations - Software engineering is a time-game. Creators work under pressured situations and struggle to finish project requirements within scanty and strict timelines.
Restricted infrastructure/resources - Lack of IT infrastructure or resources to execute projects efficiently.

Extreme programming.

Extreme programming (XP) is one of the utmost vital software development structures of Agile models. It is utilized to enhance software responsiveness and quality to customer requirements. The XP model commends taking the best actions that have worked fine in the earlier in program development missions to utmost levels. Quality practices must be practiced in XP. Some of the respectable practices that have been recognized in the XP model and recommended to maximize their usage are as follows:
Code Review - Code review senses and amends errors efficiently.
Testing - Testing code assists to remove mistakes and enhances its reliability
Incremental development - This is very good since client response is gained and grounded on this development crew comes up with new increases every few days after individual iteration.
Simplicity - This makes it simpler to evolve good quality code over and above to debug and test it.
Design - Good quality design is vital to develop good quality application.
Integration testing - It aids to recognize bugs at the crossing points of different functionalities

Basic principles of Extreme programming
XP is grounded on the regular iteration through which the designers execute User Stories. These are informal and simple statements of the client about the functionalities required. It is a conventional explanation by the customer of an aspect of the required system. Some of the uncomplicated activities that are done during software development are Coding, Testing, Listening, Designing, Feedback and Simplicity.

Formal methods.

Formal methods are system design practices that utilize rigorously stated mathematical models to make hardware and software systems. Contrary to other design systems, formal methods utilize mathematical proof as a supplement to system testing with the aim of ensuring correct behavior. As systems turn out to be more complicated and safety turn out to be a more important matter, the formal approach to system design delivers another level of assurance.

Formal methods differ from other design systems by the use of formal verification methods, the elementary principles of the system have to be proven correct before being accepted. It is very crucial to note that formal verification in no way remove the need for testing. Formal verification cannot correct bad expectations in the design, but it can assist identify errors in reasoning that would then be left unconfirmed.

Formal design can be seen as a three-step process, roughly speaking.

Formal Specification - The engineer rigorously describes a system utilizing a modeling language during the formal specification stage.

Verification - Formal methods differ from other specification systems by their hefty emphasis on correctness and provability. Verification is not as easy process, mainly because even the easiest system has more than a few dozen theorems to be proven.

Implementation - Once the prototype has been verified and stated, it is then implemented by changing the specification into code.

Automated Verification and Provability

Formal methods are differentiated from other specification systems by their stress on proof correctness, which is eventually another measure of system purity. Proof is not a substitute but a complement for testing. Testing is a vital part of assuring any system's suitability, but it is limited. Testing cannot prove that a system functions properly, it can only demonstration that the system functions for the tested cases. Because of this, formal proof is required.

The Lightweight Approach

The faults in formal specifications have been severely focused on in the past years, leading to numerous alternate approaches. The old-style view of formal methods as all-inclusive highly abstracted schemes has made formal methods to be all-encompassing, very expensive and extremely rigorous. Although theoretically attractive, formal methods have commonly been overlooked by engineers in the field.

The lightweight method to formal design makes out that formal methods are not a cure-all: there are fields where formal methods are suitable and fields where a formal specification will achieve nothing. In a lightweight design, formal methods are utilized in precise locations, and different formal methods may be utilized in different subsystems, preferably playing to the strong point of each method.

While formal systems are appealing in theory, their practical executions are to a certain degree wanting. By trying to define all of any system, formal methods have outsmarted, and mostly failed. The lightweight method, which offer a compromise between the goals of formal design and the necessities of engineering, delivers a good compromise and is the utmost productive route for future study.

Soft systems methodology.

Peter Checkland, a British system thinker, developed the Soft Systems Methodology (SSM) on the foundation of 10 years of research. It is a method to model business processes and it can be utilized for general problem cracking and handling changes in the organization.

The main use of SSM turns around the analysis of complex conditions, with divergent views on the description of the problem. SSM can intercede in such circumstances by creation discussion amongst all parties involved possible. This makes it a possibility to reach an agreement, in which can satisfy all parties involved.

Consensus

Soft Systems Methodology (SSM) is a technique to structure compound problems and to develop feasible and desirable changes in a distinguished group of people. Such a mixed group can consist of developers, employees, customers and users whereby everybody sheds a distinct light on a problem.

It is regularly difficult to bring into line all these different thinking methods. That is the reason Peter Checkland has plainly designed the SSM to guide a mixed group through such a process so that each member gets the chance to assembly a complex problem. In this method it is a possibility to implement a feasible change.

7 steps

Checkland defines SSM as a technique that can be followed in 7 steps.

Step 1 – Recognizing a problematic situation

In order to describe the problem, it is important to gather lots of information first.

Step 2 – Define the problematic situation

Based on all the information got from step 1, the problem situation can be properly defined and described.

Step 3 – Formulating elementary definitions

The stress lies on telling the ideal way in which the system must function.

Step 4 – Building conceptual models

Conceptual models are built for each activity from the origin definition from step 3, with a briefly well-defined objective.

Step 5 – Comparison of models and reality

The theoretical models are built on theory, but that can be very different from reality.

Step 6 – Defining changes

It is not always required to execute changes in the solution. If this is the situation, then the changes must be well-defined and checked if they are feasible or not.

Step 7 – Taking action

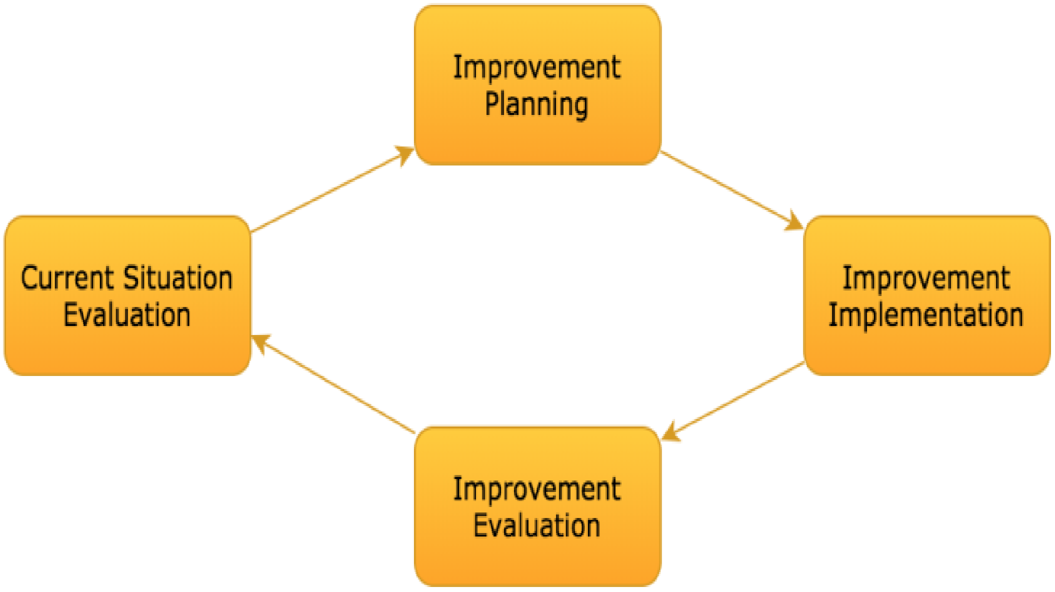
It is essential to take action to enhance the problematic situation. The variations from step 6 will be executed in the organization. If problems happen, the cycle begins again at the first stage. Therefore, SSM has a repetitive cycle.

Software Process Improvement.

Software Process Improvement (SPI) methodology is described as an order of tasks, techniques and tools to plan and execute improvement activities to attain specific goals like achieving higher product quality, increasing development speed or reducing costs. SPI can be thought of as method re-engineering or change management project to notice the software development lifecycle incompetence and fix them to have an improved process.

SPI primarily contains of 4 cyclic steps as revealed in the figure below, though these steps can be fragmented down into additional steps according to the techniques and method used. While in most situations the process will encompass these steps.

Figure 1 shows SPI 4 cyclic steps.



Software quality.

Software quality is explained as an area of practice and study that describes the necessary qualities of software products. There are two key approaches to software quality: quality attributes defect and management.

Software quality defect management approach.

A software defect can be viewed as any failure to address the user's requirements. Common defects include misunderstood or missed requirements and mistakes in design, data relationships, functional logic, validity checking, process timing and coding errors. The software defect management approach is founded on managing and counting defects. Defects are generally characterized by severity and the figures in each category are utilized for planning.

Software quality characteristics approach.

This method to software quality is prime illustrated by fixed quality representations, like ISO/IEC 25010:2011. This standard defines a hierarchy of eight quality features, each made of sub-features:

- Functional suitability
- Operability
- Reliability
- Compatibility
- Performance efficiency
- Security
- Maintainability
- Transferability

Moreover, the standard describes a quality-in-use model made up of five characteristics:

- Effectiveness
- Efficiency
- Satisfaction

- Safety
- Usability

A fixed software quality model is regularly useful in view of an overall comprehending of software quality. Quality function distribution provides a process for making products based on features derived from user requirements.

Software metric.

A software metric is a quantity of software characteristics which are countable or measurable. Software metrics are treasured for numerous reasons, including measuring software performance, measuring productivity, planning work items and several other uses.

In the software development process, various metrics are that are all associated. Software metrics are comparable to the four roles of management: Planning, Control, Organization or Improvement.

Software metrics can be classified into two types as follows:

Product Metrics - They are the measures of various features of the software product. It has two significant software features which are:

- Complexity and size of software.
- Reliability and quality of software.

Process Metrics – They are the measures of various features of the software development process. For instance, the effectiveness of fault detection.

Requirements analysis.

Requirements analysis, also known as requirements engineering, is the process of determining user needs for a modified or new or product. These attributes, called requirements, must be relevant, quantifiable and detailed. In software engineering, these requirements are usually referred to as functional specifications. Requirements analysis is a vital part of project management.

Requirements analysis involves regular communication with system operators to determine precise feature requirements, resolving of conflict or uncertainty in requirements as wanted by the various users, evasion of feature creep and keeping records of all aspects of the project progress process from beginning to finish. Requirements analysis is a team work that demands a combination of software, hardware and human factors engineering skill as well as expertise in dealing with people.

Configuration management

Configuration Management is the procedure for maintaining systems, like computer software and hardware, in a wanted state. Configuration Management (CM) is also a technique of making sure that systems execute in a manner constant with expectations over time.
Initially established in the US military and now broadly used in many different types of systems, CM assists identify systems that require to be updated, patched or reconfigured to adapt to the desired state. CM helps engineering teams creates stable and robust systems through the utilization of tools that automatically monitor and manage updates to configuration data. Complex software systems are designed by components that vary in granularity of complexity and size.
Configuration management is an essential tool for handling complex software systems. Lack of CM can cause serious glitches with uptime, reliability and the ability to measure a system. Numerous existing software development tools have CM features built in.

Conclusion.

Software engineering has changed steadily from its formation days in the 1940s until today in the 2000s. Applications have changed endlessly. The ongoing aim to improve practices and technologies, pursues to enhance the productivity of practitioners and the quality of software to users. While it is not possible to forecast how the traditional and latest forms of systems engineering will advance, a strong future lies in front. Rises in technological complication result in new trials in networks, architecture, software engineering and hardware and human systems incorporation. The engineering measure for systems surpasses points that could have been unreal only a brief time ago.

Evolving trends show that the future of software development will experience significant change. The wide collection of new innovations and technologies has big implications for the application development space. Software development companies can't afford to pay attention to developing trends. Firms that select to invest finances, time and other capitals to adjust to the expectations of the varying market are those that will achieve and keep a maintainable competitive advantage.

Bibliography.

https://www.computer.org/publications/tech-news/events/what-to-know-about-the-scientist-who-invented-the-term-software-engineering. Lori Cameron
https://users.ece.cmu.edu/~koopman/des_s99/formal_methods/
https://www.geeksforgeeks.org/software-engineering-introduction-to-software-engineering. Date: 08 June 2021
https://melsatar.blog/2018/06/26/the-software-process-improvement-spi-reward-or-risk/ Mohamed Sami. Date: 26 June 2018
https://www.techtarget.com/whatis/definition/software-engineering. Date: November 2016
https://www.toolshero.com/problem-solving/soft-systems-methodology-ssm/ Date: 04 March 2022