College of Electrical and Mechanical
Department of Software Engineering

Title: The Nature and Role of Professional Societies and Software Engineering Standards

Course code: SWEG2103

Group member name	ID
1. Nebiyou Elias	1046/13
2. Nesredin Getahun	1057/13
3. Nebiyou Zewge	1051/13
4. Natnael Worku	1033/13
5. Natnael Endalkachew	1009/13
6. Natnael Yeshiwas	1035/13
7. Nebyat Bekele	1052/13
8. RebumaTadele	1086/13
9. Senay Bihon	1167/13

Submitted to: Natnael T.

Date: April 10, 2022

Table of Contents

The Nature and Role of Professional Societies and Software Engineering Standards

Introduction

On this paper we are going to discuss about the nature and role of professional societies. We are also going to discuss further about software engineering standards. Software engineering standards are published documents created to ensure the reliability of the materials, products, methods, and/or services. They establish requirements, specifications, guidelines, characteristics, and/or procedures designed. Typically, they are developed through a consensus process and approved by various national and international agencies, professional societies, or industry organizations. Standards are the minimally accepted professional practice and/or quality that must be observed. Additionally, the nature of software engineering and several roles that could be found in software engineering are discussed in detail. The Software Engineering Professional Practice is concerned with the knowledge, skills, and attitudes that software engineers must possess to practice software engineering in a professional, responsible, and ethical manner. Because of the widespread applications of software products in social and personal life, the quality of software products can have profound impact on our personal well-being and societal harmony. Software engineers must handle unique engineering problems, producing software with known characteristics and reliability. This requirement calls for software engineers who possess a proper set of knowledge, skills, training, and experience in professional practice. We will further discuss about professionalism in software engineering and the role of group dynamics & communication skills.

Software Engineering Nature

Software engineer is a science that deals with the design, implementation, and maintenance of complex computer programs [1].Or it can be said that it is the process of application of technological knowledge in order to design, implement, and document a given software.

The major qualities which identify software as it is can be seen as 6 major components.

- Functionality: - refers to degree of performance of a software against its needs. This include suitability, accuracy, interoperability, compliance, and security.
- Reliability: - ability/capability of software in maintaining a given performance under a certain condition. For is to be reliable it needs to be recoverable, and fault tolerance.
- Usability: - refers to how easily the software can be understand, operated, and learned.
- Efficiency: - ability of the software to apply all system resources available in an efficient manner.

- Maintainability: - mainly refers to the ease of modification that can be made.
- Portability: - its main attributes include adoptability, ability to be installed, and replace ability.

Software Engineering Roles

Software engineering has a wide scope with different roles based on its complexity level. Accordingly some types of roles of software engineering are listed and discussed below.

- Front end engineer: - are software engineers who specialize in user interface. User interface is the point at which human users interact with a computer, application or website [2]. What they deal with mostly is working on cross browser compatibility, fixing bugs for excellent visual presentation.
- Back end engineer: - are the once who deal with the internal logic and performance of application.
- Full end engineer: - are the individuals who can handle job of both front and back end engineers.
- Software engineer in test: - is a software engineer who is responsible for writing software to validate the quality of the application. Sometimes they are called QA engineers. By creating automated test tool, they make sure that a given program is working as expected [3].
- Devops engineer: - is an engineer who introduces processes, tools, and methodologies to balance needs throughout software development life cycle [4]. They mostly manage application infrastructure like database system, servers, etc…
- Security engineer: - is an engineer who specialize in creating methods to check the security of software's by exploiting and fixing the flaws in it. They are what we call a white hat hackers (ethical hackers) who try to penetrate the system to discover its vulnerabilities.

Software Engineering Standards

A. What are standards?

Standards are published documents created to ensure the reliability of the materials, products, methods, and/or services. They establish requirements, specifications, guidelines, characteristics, and/or procedures designed. Typically, they are developed through a consensus process and approved by various national and international agencies, professional societies, or industry organizations. Standards are the minimally accepted professional practice and/or quality that must be observed.

B. How are standards developed?

The following are a few examples of how standards organizations develop their standards.

ISO - "Like a symphony, it takes a lot of people working together to develop a standard. ISO's role is similar to that of a conductor, while the orchestra is made up of independent technical experts nominated

by our members. The experts form a technical committee that is responsible for a specific subject area. They begin the process with the development of a draft that meets a specific market need. This is then shared for commenting and further discussion. The voting process is the key to consensus. If that's achieved then the draft is on its way to becoming an ISO standard. If agreement isn't reached then the draft will be modified further, and voted on again. From first proposal to final publication, developing a standard usually takes about 3 years."

ASTM - "A full consensus standard is developed by a cross-section of stakeholders with an interest in its use. When there is a need for new standards, requests can come from trade associations, government agencies, and professional societies that do not create their own standards; or manufacturers, consumer groups, and even individuals. The request is presented to an ASTM technical committee and the process of standards development begins." -

C. How long are standards active?

The time frame in which a standard remains active varies based on the standards organization's review and revision processes.

D. Who uses standards?

Small businesses, national and international businesses, governments, engineers, scientists, architects, designers, students, etc.

Why are standards important?

- Enhance health, safety and quality of life
- Improve performance
- Reduce risk
- Become more sustainable
- Facilitate global trade and market access
- Help produce efficient and effective products
- Reduce costs, improve supplier relations

What are some of the risks of not identifying and not complying with relevant current standards?

- Inability to sell completed product
- Damage relationship with suppliers
- Interoperability issues
- Product not approved by regulators
- Possible governmental sanctions, fine, and/or reprimands

- Possible civil lawsuits

E. How are standards named?

Standards are named using the acronym of the standard granting organization, the number of the standard, and the year the standard was issued.

- ASCE/SEI 7-16
- ASTM F963-17
- ISO 13485:2016

Group Dynamics and Psychology

A software engineer must be able to interact cooperatively and constructively with others to first determine and then meet both needs and expectations. Knowledge of group dynamics and psychology is an asset when interacting with customers, coworkers, suppliers, and subordinates to solve software engineering problems [5].

i. Dynamics of Working in Teams/Group

Software engineers must work with others. On one hand, they work internally in engineering teams; on the other hand, they work with customers, members of the public, regulators, and other stakeholders. Performing teams are cohesive and possess a cooperative, honest, and focused atmosphere. Individual and team goals are aligned so that the members naturally commit to and feel ownership of shared outcomes.

Team members facilitate this atmosphere by being intellectually honest, making use of group thinking, admitting ignorance, and acknowledging mistakes. They share responsibility, rewards, and workload fairly. They take care to communicate clearly, directly to each other and in documents, as well as in source code, so that information is accessible to everyone. Peer reviews about work products are framed in a constructive and non-personal way. This allows all the members to pursue a cycle of continuous improvement and growth without personal risk. In general, members of cohesive teams demonstrate respect for each other and their leader.

One point to emphasize is that software engineers must be able to work in multidisciplinary environments and in varied application domains. Since today software is everywhere, from a phone to a car, software is impacting people's lives far beyond the more traditional concept of software made for information management in a business environment.

ii. Individual Cognition

Engineers desire to solve problems. The ability to solve problems effectively and efficiently is what every engineer strives for. However, the limits and processes of individual cognition affect problem solving. In software engineering, notably due to the highly abstract nature of software itself, individual cognition plays a very prominent role in problem solving.

In general, an individual software engineer's ability to decompose a problem and creatively develop a solution can be inhibited by

- need for more knowledge,
- subconscious assumptions,
- volume of data,
- fear of failure or consequence of failure,
- culture, either application domain or organizational,
- lack of ability to express the problem,
- perceived working atmosphere, and
- Emotional status of the individual.

iii. Dealing with Problem Complexity

Many, if not most, software engineering problems are too complex and difficult to address as a whole or to be tackled by individual software engineers. When such circumstances arise, the usual means to adopt is teamwork and problem decomposition.

Teams work together to deal with complex and large problems by sharing burdens and drawing upon each other's knowledge and creativity. When software engineers work in teams, different views and abilities of the individual engineers complement each other and help build a solution that is otherwise difficult to come by. Some specific teamwork examples to software engineering are pair programming and code review.

iv. Interacting with Stakeholders

Success of a software engineering endeavor depends upon positive interactions with stakeholders. They should provide support, information, and feedback at all stages of the software life cycle process.

During development, stakeholders may provide feedback on specifications and/or early versions of the software, change of priority, as well as clarification of detailed or new software requirements. Last, during software maintenance and until the end of product life, stakeholders provide feedback on evolving or new requirements as well problem reports so that the software may be extended and improved.

Therefore, it is vital to maintain open and productive communication with stakeholders for the duration of the software product's lifetime.

v. Dealing with Uncertainty and Ambiguity

As with engineers of other fields, software engineers must often deal with and resolve uncertainty and ambiguities while providing services and developing products. The software engineer must attack and reduce or eliminate any lack of clarity that is an obstacle to performing work. Often, uncertainty is simply a reflection of lack of knowledge. In this case, investigation through recourse to formal sources such as textbooks and professional journals, interviews with stakeholders, or consultation with teammates and peers can overcome it.

vi. Dealing with Multicultural Environments

Multicultural environments can have an impact on the dynamics of a group. This is especially true when the group is geographically separated or communication is infrequent, since such separation elevates the importance of each contact. Intercultural communication is even more difficult if the difference in time zones make oral communication less frequent.

Multicultural environments are quite prevalent in software engineering, perhaps more than in other fields of engineering, due to the strong trend of international outsourcing and the easy shipment of software components instantaneously across the globe.

For a software project to be a success, team members must achieve a level of tolerance, acknowledging that some rules depend on societal norms and that not all societies derive the same solutions and expectations.

This tolerance and accompanying understanding can be facilitated by the support of leadership and management. More frequent communication, including face-to-face meetings, can help to mitigate geographical and cultural divisions, promote cohesiveness, and raise productivity. Also, being able to communicate with teammates in their native language could be very beneficial.

Communication Skills

It is vital that a software engineer communicate well, both orally and in reading and writing. Successful attainment of software requirements and deadlines depends on developing clear understanding between the software engineer and customers, supervisors, coworkers, and suppliers. Optimal problem solving is made possible through the ability to investigate, comprehend, and summarize information. Customer product acceptance and safe product usage depend on the provision of relevant training and documentation. It follows that the software engineer's own career success is affected by the ability to consistently provide oral and written communication effectively and on time [6].

I. Reading, Understanding, and Summarizing

Software engineers are able to read and understand technical material. Technical material includes reference books, manuals, research papers, and program source code.

Reading is not only a primary way of improving skills, but also a way of gathering information necessary for the completion of engineering goals. A software engineer sifts through accumulated information, filtering out the pieces that will be most helpful. Customers may request that a software engineer summarize the results of such information gathering for them, simplifying or explaining it so that they may make the final choice between competing solutions.

Reading and comprehending source code is also a component of information gathering and problem solving. When modifying, extending, or rewriting software, it is critical to understand both its implementation directly derived from the presented code and its design, which must often be inferred.

II. Writing

Software engineers are able to produce written products as required by customer requests or generally accepted practice. These written products may include source code, software project plans, software requirement documents, risk analyses, software design documents, software test plans, user manuals, technical reports and evaluations, justifications, diagrams and charts, and so forth.

Writing clearly and concisely is very important because often it is the primary method of communication among relevant parties. In all cases, written software engineering products must be written so that they are accessible, understandable and relevant for their intended audience(s).

III. Team and Group Communication

It is absolutely essential that group members communicate effectively and efficiently with each other and with other project stakeholders. Group members must exchange information on the status of their work, the design decisions that have been made and changes to previous design decisions. Good communication helps to resolve problems, to strengthen group cohesiveness and to understand the motivation, strengths, and weaknesses of other people in the group.

The effectiveness and efficiency are influenced by:

1) Group size: - As a group gets bigger, it gets harder for members to communicate effectively. The number of one-way communication links is n*(n-1), where n is the group size. This shows that it is quite possible that some people will rarely communicate with each other.

2) Group structure:- People in informally structured groups communicate more effectively than people in groups with a formal, hierarchical structure where communication tend to flow up and down the hierarchy and where people at the same level may not talk to each other.
3) Group composition: - Communication is also usually better in a mixed-use groups than in single-sex groups. Women are often more interaction-oriented than men.
4) The physical work environment: - The organization of the workplace is a major factor in facilitating or inhibiting communications.
5) The available communication channels: - There are many different forms of communication. As project teams become increasingly distributed, with team members working remotely, you need to make use of interaction technologies to facilitate group communication.

Effective communication is achieved when communication are two-way and the people involved can discuss issues and information and establish a common understanding of proposals and problems.

IV. Presentation Skills

Software engineers rely on their presentation skills during software life cycle processes. During and after software design, software construction, and software maintenance, software engineers lead reviews, product walkthroughs and training. All of these require the ability to present technical information to groups and solicit ideas or feedback.

Therefore the software engineer's presentation skill influences product acceptance, management, customer support and influences the ability of stakeholders to comprehend and assist in the product effort. This knowledge needs to be archived in the form of slides, knowledge write-up, technical whitepapers, and any other material utilized for knowledge creation.

Conclusion

During recent years we have seen dramatic changes in software engineering education, accreditation, licensing, and certification. Undergraduate software engineering degree programs are now being offered. ABET, as a result of the CSAB/ABET merger, is planning to accredit undergraduate software engineering programs. Some countries are licensing software engineers. Certification, which has existed in some form for years, is taking on increased importance as an alternative to licensing. All of these changes, and many associated activities, have taken place in the context of making software engineering a profession. In the future we expect that there will be continued debate and continued evolution of software engineering as a profession. The exact role that licensing and certification will play is undetermined. The role of accreditation seems clear and is likely to be the same for software engineering as it is for other engineering fields. Accreditation of software engineering degree programs are one of the more mature processes of those under discussion in this paper

References

[1] mks075, "software engineering | software characteristics," 18 August 2021. [Online]. Available: https://www.geeksforgeeks.org/software-engineering-software-characteristics/. [Accessed 7 April 2022].

[2] e. o. indeed, "what is user interface," 17 September 2021. [Online]. Available: https://www.indeed.com/career-advice/career-development/user-interface. [Accessed 8 April 2022].

[3] i. e. team, "what is user interface," 17 September 17. [Online]. Available: https://www.indeed.com/career-advice/career-development/user-interface. [Accessed 8 April 2022].

[4] redhat, "who is a devops engineer," 8 January 2019. [Online]. Available: https://www.redhat.com/en/topics/devops/devops-engineer#:~:text=A%20DevOps%20engineer%20introduces%20processes,drive%20adoption%20within%20your%20company. [Accessed 8 April 2022].

[5] I. Sommerville, "Software Engineering, 10th edition," in *Software Engineering, 10th edition* , Pearson EdEducation..

[6] swebokwiki, "swebokwiki.," [Online]. Available: http://swebokwiki.org/Chapter_11:_Software_Engineering_Professional_Practice.