CHAPTER 3

INTRODUCTION TO SOFTWARE ENGINEERING

The term ***software engineering*** is composed of two words, software and engineering. **Software** is more than just a program code. A program is an executable code, which serves some computational purpose. Software is considered to be a collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called **software product.**

Engineering on the other hand, is all about developing products, using well-defined, scientific principles and methods.

So, ***software engineering*** can be defined as an engineering branch associated with the development of software product using well-defined scientific principles, methods and procedures. The outcome of software engineering is an efficient and reliable software product. IEEE defines software engineering as:

- The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software.

In short, Software engineering is a branch of computer science, which uses well-defined engineering concepts required to produce efficient, durable, scalable, in-budget and on-time software products.

Without using software engineering principles it would be difficult to develop large programs. In industry it is usually needed to develop large programs to accommodate multiple functions. A problem with developing such large commercial programs is that the complexity and difficulty levels of the programs increase exponentially with their sizes. Software engineering helps to reduce this programming complexity.

Software engineering principles use two important techniques to reduce problem complexity: *abstraction* and *decomposition*. The principle of abstraction implies that a problem can be simplified by omitting irrelevant details. In other words, the main purpose of

abstraction is to consider only those aspects of the problem that are relevant for certain purpose and suppress other aspects that are not relevant for the given purpose. Decomposition is technique, a complex problem is divided into several smaller problems and then the smaller problems are solved one by one. The problem has to be decomposed so that each component of the decomposed problem can be solved independently and then the solution of the different components can be combined to get the full solution.

Why is Software Engineering required?

Software Engineering is required due to the following reasons:

- To manage large software
- For more scalability
- Cost Management
- To manage the dynamic nature of software
- For better quality Management

The necessity of software engineering appears because of a higher rate of progress in user requirements and the environment on which the program is working.

- **Huge Programming**: It is simpler to manufacture a wall than to a house or building, similarly, as the measure of programming become extensive engineering has to step to give it a scientific process.
- **Adaptability:** If the software procedure were not based on scientific and engineering ideas, it would be simpler to re-create new software than to scale an existing one.
- **Cost:** As the hardware industry has demonstrated its skills and huge manufacturing has let down the cost of computer and electronic hardware. But the cost of programming remains high if the proper process is not adapted.

- **Dynamic Nature:** The continually growing and adapting nature of programming hugely depends upon the environment in which the client works. If the quality of the software is continually changing, new upgrades need to be done in the existing one.
- **Quality Management:** Better procedure of software development provides a better and quality software product.

Importance of Software Engineering

(1) Reduces complexity:

- Large software systems are always complicated and challenging to progress.
- Software engineering divides big problems into various small issues. And then start solving each small issue one by one.
- All these small problems are solved independently to each other and then integrated together to produce the software product.

(2) To minimize software cost:

- A lot of resources are required to develop large-scale software systems, such as: manpower, software licenses, hardware…etc.
- As companies seek to build cutting-edge software to drive growth, determining the overall budget becomes very tricky.
- Software engineering provides systematic means for having regular interaction and obtaining a budget estimates.

(3) To decrease time:

- One of the main criteria to measuring project success and yet the most challenging is delivering software projects on time.
- Schedule issues are the main reason for conflicts on projects, especially during the second half of projects where the actual implementation takes place to produce the final working product.

- Software engineering involves the processes required to ensure timely completion of a project.

(4) Handling Big projects:

- Big projects are not done in a couple of days, and they need lots of work, planning, and management. And to invest six and seven months of any company, it requires heaps of planning, direction, testing, and maintenance.
- Companies provide many resources to the plan to be completed.
- So to handle a big project without any problem, the company has to go for software engineering methods.

(5) Reliable software:

- Software reliability is the "probability that the software will execute for a particular period of time without failure".
- Software engineering provides models for software quality measurements and evaluations. Examples include models for estimating defects inserted and removed throughout the software lifecycle.

(6) Effectiveness:

- Effectiveness means getting the desired results/ doing the right thing!
- From example: achieving missions and goals; generating satisfied customers; producing work of high quality.
- This includes proper use of: communication, technology, organizational and individual knowledge, and resources.

Software Engineering as an Engineering discipline

Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use. In this definition, there are two key phrases:

2. Engineering discipline Engineers make things work. They apply theories, methods, and tools where these are appropriate. However, they use them selectively and always try to discover solutions to problems even when there are no applicable theories

and methods. Engineers also recognize that they must work to organizational and financial constraints so they look for solutions within these constraints.

3. All aspects of software production: Software engineering is not just concerned with the technical processes of software development. It also includes activities such as software project management and the development of tools, methods, and theories to support software production.

Software engineering is related to both computer science and systems engineering:

1. Computer science is concerned with the theories and methods that underlie computers and software systems, whereas software engineering is concerned with the practical problems of producing software. Some knowledge of computer science is essential for software engineers in the same way that some knowledge of physics is essential for electrical engineers. Computer science theory, however, is often most applicable to relatively small programs.
2. System engineering is concerned with all aspects of the development and evolution of complex systems where software plays a major role. System engineering is therefore concerned with hardware development, policy and process design and system deployment, as well as software engineering. System engineers are involved in specifying the system, defining its overall architecture, and then integrating the different parts to create the finished system. They are less concerned with the engineering of the system components (hardware, software, etc.).

Software Engineering vs. Computer Science

- **Computer science** is concerned with theory and fundamentals; this field involves the understanding and application of both abstract and concrete knowledge.
- **Software engineering** is a field largely concerned with the application of engineering processes to the creation, maintenance, and design of software for a variety of different purposes.

Software Engineering Professional Practice

Professionalism, Accreditation, Certification, and Licensing

Professionalism

Software professionalism is one of the most important aspects of software engineering. Software touches every aspect of our life on a daily basis, from personal interactions all the way up through enterprise transactions. It is the movement to make software engineering a profession, with various aspects like degree and certification programs, government licensing, and professional ethics. A software engineering displays professionalism notably through adherence to codes of ethics, professional conduct and to standard practices that are established by the engineer's professional community.

Software engineer is a person who applies the principles of software engineering to the design, development, maintenance, testing and evaluation of computer software.

Six (6) Key Qualities of Software Professionalism

- Demand Quality
- Be Stable
- Continually Produce
- Be Fearless
- Estimate Honestly
- Keep Learning

Software Accreditation

Accreditation is a third party attestation related to a conformity assessment body such as certification body showing formal demonstration of its competence to carry out tasks. It is granted by a management official and provides an important quality control. By accrediting a system or application, a manager accepts the associated risk. Accreditation software helps organizations gather the

documentation that is necessary to earn and maintain accreditation status. The right accreditation software will fit seamlessly into your workflow, rather than force you to adjust your operational functions to match its specifications.

The **Accrediting Board for Engineering and Technology (ABET**) is the accrediting body for US software engineering programs. The organization has a track record for success in setting professional engineering standards. Although accreditation of software engineering programs is relatively new, ABET has been the accreditor for other engineering disciplines for many decades.

Importance of Software Accreditation

Professional could be able to

- Identify, formulate, and solve complex engineering problems by applying principles of engineering, science, and mathematics.
- Apply engineering design to produce solutions that meet specified needs with consideration of public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors.
- Recognize ethical and professional responsibilities in engineering situations and make informed judgments, which must consider the impact of engineering solutions in global, economic, environmental, and societal contexts.
- Function effectively on a team whose members together provide leadership, create a collaborative and inclusive environment, establish goals, plan tasks, and meet objectives.
- Give customers the confidence that your tool can be used to produce an accurate social value analysis.
- Develop and conduct appropriate experimentation, analyze and interpret data, and use engineering judgment to draw conclusions.
- Acquire and apply new knowledge as needed, using appropriate learning strategies

Software Certification

Certification is formal recognition of a level of proficiency in the information technology (IT) quality assurance industry. The recipient is acknowledged as having an overall comprehension of the disciplines and skills represented in a comprehensive body of knowledge for a respective software discipline. Software certification demonstrates the reliability and safety of software systems in such a way that it can be checked by an independent authority with minimal trust in the techniques and tools used in the certification process itself.

Software certification comprises a wide range of formal, semi-formal, and informal assurance techniques, including formal verification of compliance with explicit safety policies, system simulation, testing, code reviews. It represents a written assurance by a third party of the conformity of a product, process or service to specified requirements. The certification process ensures that security weaknesses are identified and plans for mitigation strategies are in place.

The Need to be certified

As the IT industry becomes more competitive, the ability for management to distinguish professional and skilled individuals in the field becomes mandatory. Certification demonstrates a level of understanding in carrying out quality assurance principles and practices. The certifications supported by the International Software Certification Board (ISCB) are as follows:

Certified Associate in Software Quality (CASQ)

Certified Associate in Software Testing (CAST)

Certified Software Quality Analyst (CSQA)

Certified Software Tester (CSTE)

Certified Manager of Software Quality (CMSQ)

Certified Manager of Software Testing (CMST)

Certified Associate Business Analyst (CABA)

Certified Software Business Analyst (CSBA)

Acquiring the designation for any of the above listed certifications indicates a professional level of competence in the principles and practices of quality assurance in the IT profession as well as gain recognition as software quality profession, achieve potentially more rapid career advancement, and gain greater acceptance in the role as advisor to management.

Software Licensing

A software license is a document that provides legally binding guidelines for the use and distribution of software. It is a contract between the entity that created and supplied an application and its user. The license is a text document designed to protect the intellectual property of the software developer and to limit any claims against them that may arise from its use.

Most software are licensed, not sold, meaning there are terms the end-user must follow. The terms and conditions are often described in the Software License Agreement and usually include rules and restrictions on using the software. Software licensing exists to protect a copyright of software and can restrict the way that the user can use it.
For example, how long you can use the software for (e.g., 12 months), how many computers you can install it on, what types of uses are permitted (e.g., educational vs. commercial use) and restrictions on reverse engineering, selling, or transferring the software are usually included in the Software License Agreement.

Different Types of License

Different types of software licenses require to meet certain obligations if you want to reuse the code. The following are the common software license types:

Public domain: This is the most permissive type of software license. When software is in the public domain, anyone can modify and use the software without any restrictions. But you should always make sure it's secure before adding it to your own codebase. Warning:

Code that doesn't have an explicit license is NOT automatically in the public domain. This includes code snippets you find on the internet.

Permissive: Permissive licenses are also known as "Apache style" or "BSD style." They contain minimal requirements about how the software can be modified or redistributed. This type of software license is perhaps the most popular license used with free and open-source software. Aside from the Apache License and the BSD License, another common variant is the MIT License.

LGPL: The GNU Lesser General Public License allows you to link to open-source libraries in your software. If you simply compile or link an LGPL-licensed library with your own code, you can release your application under any license you want, even a proprietary license. But if you modify the library or copy parts of it into your code, you'll have to release your application under similar terms as the LGPL.

Copyleft: Copyleft licenses are also known as reciprocal licenses or restrictive licenses. The most well-known example of a copyleft or reciprocal license is the GPL. These licenses allow you to modify the licensed code and distribute new works based on it, as long as you distribute any new works or adaptations under the same software license. For example, a component's license might say the work is free to use and distribute for personal use only. So, any derivative you create would also be limited to personal use only. (A derivative is any new software you develop that contains the component.)
The catch here is that the users of your software would also have the right to modify the code. Therefore, you'd have to make your own source code available. But of course, exposing your source code may not be in your best interests.

Proprietary: Of all types of software licenses, this is the most restrictive. The idea behind it is that all rights are reserved. It's generally used for proprietary software where the work may not be modified or redistributed.

Why is Software Licensing Important?

Software licensing protects the software company's intellectual property and the end-user. For instance, someone could purchase the software, reverse-engineer it, and sell a knock-off version. The software company misses out on revenue, and the end-user receives an illegitimate copy of the software, which could lead to performance issues and cybersecurity threats.

If an end-user violates the terms of the software agreement, they could lose the right to use the software or be forced to pay a fine. Individuals and organizations should read and fully understand the terms of the software license and ensure compliance.

Code of Ethics and Professional Conduct

A code of ethics and professional conduct deals with the ethical principles that govern decisions and behavior at a company or organization. They give general outlines of how employees (i.e. software engineers) behave, as well as specific guidance for handling issues like harassment, safety, and conflicts of interest.

A ***code of ethics*** is broad, giving members of employees or members a general idea of what types of behaviors and decisions are acceptable and encouraged in a work environment or at a business. Whereas ***code of conduct*** is more focused. It defines how employees or members should act in a specific situation.

A ***professional code of conduct*** is an official document that clearly defines how a company's employees should behave in the workplace on a day-to-day basis.

A code of ethics and professional conduct consists of four key sections. These helps employees to be clear on how to handle many common situations. These are:

i. **The work environment**: employees should act with integrity, comply with laws, maintain a professional work environment and comply with company policies. Work environment code of conduct topics include:
    - Violence policy

- Private policy
- Equal opportunity and so on.

ii. **Conflicts of interest:** it is essential that they avoid relationship and activities that hurt, or appears to hurt, their ability to make objective and fair decisions. Conflict of interest code of conduct topics include:

- Corporate asset contribution
- Running for public office
- Employee political interests
- Security transactions and so on.

iii. **Protecting company assets:** employees should always act to protect company assets, including physical, intellectual, and electronic or digital properties. Company assets code of conduct topics include:

- Preparing, maintaining, and disclosing accurate records.
- Information security
- Use of company property.
- Facility security and so on.

iv. **Anti-bribery and corruption:** a company's integrity is essential for maintain trustworthiness and reputation. Employees should always do their work fairly, honestly, and legally. Anti-bribery and corruption topics include:

- Choosing and maintain service providers.
- Obligations for departing and former employees.
- Relationships with affiliates, international entities, and customers and so on.

As a profession software engineering contains eight principles related to the behavior and decisions made by professional software engineers. These principles govern the behavior and decisions made by software engineers, identifies the ethically responsible relationship in which individuals, groups and organizations participate and the obligations within these relationships. These codes can't

be used in isolation to justify errors; in other words, they can't be read separately by selecting the acceptable from the unacceptable one since these codes are not ethical algorithms that generate ethical decisions. Therefore, a software engineer should use ethical judgements to act in a manner which follows the code of ethics and professional practice. The following are the eight principles that are included in the code of conducts.

1. Public

Software engineers shall act consistently with the public interest, particularly they should:

i. Accept full responsibility for their own work.

ii. Moderate the interests of software engineers, the employer, the client and the users with the public good.

iii. Approve software only if they have a well-founded belief that this is safe, meets specifications, passes appropriate tests, and doesn't diminish the quality of life, quality of privacy or harm the environment. The ultimate effect of the work should be to the public good.

iv. Disclose to appropriate person of authorities any potential harm or danger to the user, public, or the environment, that they reasonably believe to be associated with the software or related document.

v. Cooperate in efforts to the address matters of grave public concern caused by software, its installation, maintenance, support or documentation.

vi. Be fair and avoid deception in all statements, particularly public ones, concerning software or related documents, methods and tools.

vii. Consider issues of physical disabilities, allocation of resources, economic disadvantages and other factors that can diminish access to the benefits of software.

viii. Be encouraged to volunteer professional skills to good causes and contribute to public education concerning the education.

2. Client and Employer

Software engineers shall act in a manner that is the best interest of their client and employer, consistent with the public interest. In particular, they shall:

i. Provide service in their area of competence, being honest and forthright about any limitation of their experience and education.

ii. Not knowingly use software that is obtained or retained either illegally or unethically.

iii. Use the property of a client or employer only in ways properly authorized, and with the client's employer's knowledge and consent.

iv. Ensure that any document upon which they rely has been approved, when required, by someone authorized to approve it.

v. Keep private any confidential information gained in their professional work, where such confidentiality is consistent with the public interest and consistent with the law.

vi. Identify, document, collect evidence, and reported to the client or the employer promptly if a project is likely to fail, to prove too expensive, to violate intellectual property law, or otherwise to be problematic.

vii. Identify, document, report significant issues of social concern, of which they are aware, in software or related documents, to the employer or he client.

viii. Accept no outside work detrimental to the work they perform for their primary employer.

ix. Promote no interest averse to their employer or client, unless a higher ethical concern is being compromised; in that case, inform the employer or other authority of the ethical concern.

3. Product

Software engineers shall ensure that their products and related modification meets the highest professional standards possible. In particular, software engineers shall, as appropriate:

i. Strive for high quality, acceptance cost, and reasonable schedule, ensuring significant tradeoffs are clear to and accepted by the employer and the client, are available for consideration by the user and the public.

ii. Ensure proper and achievable goals and objectives for any projects] on which they work or purpose.

iii. Identify, define, and address ethical, economic, cultural, legal environmental issues related to work projects.

iv. Ensure that they are qualified for any project on which they work or purpose to work, by an appropriate combination of education, training and experience.

v. Ensure that an appropriate method is used for any project on which they work or purpose to work.

vi. Work to follow professional standard, when available, that are most appropriate for the task at hand, departing from thee only when ethically or technically justified.

vii. Strive to fully understand the specification for software on which they work have been well documented, satisfy user's requirements, and have the appropriate approvals.

viii. Ensure that specifications for software on which they work have been well documented, satisfy the user's requirements, and have the appropriate approvals.

4. Judgment

Software engineers shall maintain integrity and independence in their professional judgment. Particularly:

i. Temper all technical judgements by the need to supports and maintain human values.
ii. Only endorse documents either prepared under their supervision or within their area of competence and with which they are in agreement.
iii. Maintain professional objectivity with respect to any software or related documents they are asked to evaluate.
iv. Not engage in deceptive financial practices such as bribery, double billing or other improper financial practices.
v. Disclose to all concerned parties those conflicts of interest that can't reasonably be avoided.
vi. Refuse to participate, as a member or advisors in a private, governmental or professional body of concerned with software related issues in which they, employers or client have undisclosed potential conflicts of interest.

5. Management

Software engineering managers and leaders shall promote an ethical approach to the management of the software development and maintenance. The managing software engineers shall:

i. Ensure good management for any good project on which they work including effective procedures for promotion of quality and reduction risk.
ii. Ensure that the software engineers are informed of standards before being held to them.

iii. Ensure that software engineers know the employer's policies and procedures for protecting passwords, files, and information that is confidential to the employer or others.

iv. Ensure realistic quantitative estimates of cost, scheduling, personnel, quality, and outcomes on any project on which they work and provide an uncertainty assessment of these estimates.

v. Offer fair and just remuneration.

vi. Ensure that there is fair agreement concerning ownership of any software, process, and research, writing or other intellectual property to which a software engineer has contributed.

vii. Provide for due process in hearing charges of violation of an employer's policy or this code.

6. Profession

Software engineers shall advance the integrity and reputation of the profession consistent with the public interest. In particular, software engineers shall, as appropriate:

i. Help develop an organizational environment favorable to acting ethically.

ii. Promote public knowledge of software engineering.

iii. Extend software engineering knowledge by appropriate participation in professional organizations, meetings and publications.

iv. Support, as members of a profession, other software engineers striving to follow this code.

v. Not promote their own interest at the expense of the profession, client or employer.

vi. Obey all laws governing their work, unless, in exceptional circumstances, such compliance is inconsistent with the public interest.

vii. Take responsibility for detecting, correcting, and reporting errors in software and associated documents on which they work.

viii. Recognize that violations of this code are inconsistent with being aa professional software engineer.

7. Colleagues

Software engineers shall be fair to and supportive of their colleagues. In particular, software engineers shall, as appropriate:

i. Encourages colleagues to adhere to this code.
ii. Assist colleagues in professional development.
iii. Review the work of others in an objective, candid, and properly-documented way.
iv. Assist colleagues in being fully aware of current standard work practices including policies and procedures for protecting passwords, files and other confidential information, and security measures in general.
v. In situations outside of their own areas of competence, call upon the opinions of other professionals who have competence in that area.

8. Self

Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession. In particular, software engineers shall continually endeavor to:

i. Further their knowledge of developments in the analysis, specification, design, development, maintenance and testing of software and related documents, together with the management of the development process.
ii. Improve their ability to create safe, reliable, and useful quality software at reasonable cost and within a reasonable time.
iii. Improve their ability to produce accurate, informative, and well-written documentation.
iv. Improve their understanding of the software and related documents on which they work and of the environment in which they will be used.
v. Improve their knowledge of code, its interpretation and its application to their work.
vi. Recognize that personal violations of this code are inconsistent with being a professional software engineer.

The Importance of a Code of Ethics

A code of ethics is important because it helps employees or organization members make decisions that are in line with company values in the absence of a clear rule or direct supervision. A code of ethics can improve decision making at a business, and make it easier for employees to be autonomous.

The five codes of ethics include:

- ✓ Integrity.
- ✓ Objectivity.
- ✓ Professional competence.
- ✓ Confidentiality.
- ✓ Professional behavior.

The Nature and Role of Professional Societies and Software Engineering Standards

Software Engineering Nature

Some natures of software engineering are:

- Characterized by its primary product, which is ***Software***.
- The professionals who work in this community are deeply knowledgeable in science and computer science which covers various topics like programming languages, algorithms, data structures and other hardware and system software knowledges
- In software engineering, the hardest part is designing. It is relatively easy to reproduce or build a software once it is designed.
- In its nature, software engineering has long working hours in front of a computer, being focused and diligent on the work at hand.

- Another nature of a software engineer is its versatility. A software engineer can specialize in different fields like health care, automotive, green energy, remote sensing, aeronautics, finance etc.
- And also, it is a team job where lots of people come together and work on a project by dividing the work load.

The software engineering society understands the major importance of networking and cooperation between like-minded software engineers. And that is why these societies sponsor different conferences, publish journals and some serve as reviewers or editors of software.

Some major societies in the software engineering communities are:

- **IEEE Computer Society**: are leading organizations for information, inspiration and collaboration in computer science and engineering. The members of this society are found worldwide, as this society empowers the people who advance technology by delivering tools and information for development.
- **AWC (Association of Women in Computing):** is a professional organization for women who are in the field of computing.
- **ACM (Association for Computing Machinery):** a US-based society that promotes academics and discipline for computing.

Software Engineering Roles

A software engineer must know his roles in the job so that he can execute his duties perfectly. The roles of a software engineer are interchangeable depending on the project taken. Some roles of a software engineer are:

- Being able to work with constantly evolving environment because of the ever-changing nature of technologies.
- Create a reliable and trustworthy systems economically
- Research & design new software programs
- Work closely with other staff members
- Improve systems quality by identifying issues
- Testing products before deployment

- Giving support to users

Software Engineering Standards

Just like other fields, software engineering field has standards or guidelines that should be followed in order for the work to be harmonized and same from country to country. These standards are published documents created to ensure the reliability of the materials, products, methods, and/or services. It includes establish requirements, specifications, guidelines, characteristics, and/or procedures designed and developed through a consensus process and approved by various national and international agencies, professional societies, or industry organizations. These are technical specifications or other precise criteria designed to be used consistently as a rule, guideline or definition.

The **IEEE Software & Systems Engineering Standards Committee (S2ESC)** is chartered by the IEEE Computer Society Standards Activities Board to codify the norms of professional software engineering practices into standards, including the standardization of processes, products, resources, notations, methods, nomenclatures, techniques, and solutions for the engineering of software and systems dependent on software. They promotes the use of software engineering standards among clients, practitioners, educators in coordination with other IEEE initiatives. They also harmonize national and international software engineering standards development, and promotes the discipline and professionalization of software engineering.

Some commonly used Software Engineering Standards

ISO/IEC 12207: International software engineering standard that defines the software engineering process, activity, and tasks that are associated with a software life cycle process. It supplies a common structure so that the buyers, suppliers, developers, maintainers, operators, managers and technicians involved with the software development use a common language.

ISO/IEC 9126: This standard deals with the following aspects to determine the quality of a software application −

- Quality model

- External metrics
- Internal metrics
- Quality in use metrics

IEEE Standards Association: it develops global standards in broad industries like power, energy, AI systems, IoT, electronics, robotics, telecommunications and also including programming norms and standards.

ISO/IEC 12119: This standard deals with software packages delivered to the client. It does not focus or deal with the clients' production process. It focuses on the set of requirements for software package and instructions for testing a delivered software package against the specified requirements.

ISO/IEC 9241-11: This standard deals with the extent to which a product can be used by specified users to achieve specified goals with Effectiveness, Efficiency and Satisfaction in a specified context of use. This standard proposed a framework that describes the usability components and the relationship between them.

ISO/IEC 25000:2005: Commonly known as the standard that provides the guidelines for Software Quality Requirements and Evaluation (SQuaRE). It helps in organizing and enhancing the process related to software quality requirements and their evaluations.

Software Quality Requirements and Evaluation (SQuaRE) is divided into sub-parts such as −

- ISO 2500n − Quality Management Division
- ISO 2501n − Quality Model Division
- ISO 2502n − Quality Measurement Division
- ISO 2503n − Quality Requirements Division
- ISO 2504n − Quality Evaluation Division