INTERNATIONAL RESEARCH JOURNAL OF ENGINEERING AND TECHNOLOGY (IRJET) E-ISSN: 2395-0056
VOLUME: 06 ISSUE: 04 | APR 2019 WWW.IRJET.NET P-ISSN: 2395-0072

Scientific-Theoretical Basics of Software Engineering

Rasim Alguliyev[1], Tamilla Bayramova[2]

[1,2]*Department of Software Engineering, Institute of Information Technology of ANAS, Baku, Azerbaijan*

Abstract - *The article provides a systematic analysis of the scientific, engineering and experimental basics of software engineering. The role of SWEBOK knowledge base, standards, infrastructure and management in the development of life cycle processes of software engineering is highlighted. The engineering and practical aspects of the development of software products with the use of ready software components are studied.*

Keywords: software engineering, programming methods, recycled components, base processes, infrastructure, standards

1. INTRODUCTION

Software Engineering (SE) the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software [1].

The life cycle of complex systems starts with the idea of creating a system and ends with the determination of requirements, design, programming, trials, implementation, system accompaniment and decommissioning. The system lifecycle model is divided into successive stages. At each stage, the processes, issues and works to be performed, and the standards and the methods to be used are determined.

High-quality software cannot be achieved if this development process is non-systematic and unregulated. One of the key issues is the application of strictly regulated processes for the development of high-quality software.

The main purpose of SE is to increase the efficiency of software production process and to develop tools, methods and theories to achieve maximum quality [4].

Continuous development of hardware platforms, as well as their increasing application in various industries and productions, further enhances the demand for software. At the same time, programming technologies are constantly developing. At present, the world surrounding us is based on the Internet of Things, Big Data, Cloud technologies, Mobile Internet, Drones, E-medicine, Distant education, etc., which creates new challenges for software engineers.

At present, a variety of curriculums have been developed to improve the quality of disciplines in IT all over the world. There have been several recent uses of case studies in SE education. A range of methods have been proposed in various articles to improve the quality of education. The authors present challenges in software engineering and provide recommendations for their elimination. Education in software engineering must provide the students with knowledge and practice of software development processes.

The purpose of this study is to explore the scientific and theoretical basics of SE and to present the necessary knowledge for the training of literate software engineers in Azerbaijan.

2. PLACE OF SOFTWARE ENGINEERING IN SCIENCE SYSTEM

Before clarifying this issue, the following question arises, which creates disagreements among experts:

Which area is SE closer to in the knowledge system: art, science or engineering?

- *art* – the expression or application of human creative skill and imagination, typically in a visual form such as painting or sculpture, producing works to be appreciated primarily for their beauty or emotional power;
- *science* – a form of human activity aimed at the acquisition, specification and dissemination of objective, systematic and justified knowledge. This activity is based on the collection of scientific facts, their constant updating and systematization, critical analysis and the collection of scientific knowledge based on this;
- *engineering* – is the application of knowledge in the form of science, mathematics, and empirical evidence, to the innovation, design, construction, operation and maintenanceof structures, machines, materials, devices, systems, processes, and organizations [5].

SE requires specific skills and creative thinking, since its key conditions include easy interface of software product and being visually peculiar and eye-catching. In the early years of programming, it was considered an art, since it was an artistic work of a human.

SE is a technology, since the processes, tools, standards, metric, testing and certification are applied. At present, the knowledge about various aspects of software evolution is required for successful SE: technologies (equipment, operating systems, programming languages), development of requirements, architecture designing, software code quality assurance, and its integration and verification

issues, modification and improvement of software systems, organization of work, knowledge on specific disciplines, social and economic aspects of software, ethics of software engineers, marketing and entrepreneurship.

Currently, software developers do not only write software in a specific algorithmic language, they need more knowledge. New methods have to be developed to address the abovementioned issues and it is impossible without knowing the scientific and theoretical basis of SE. In this regard, SE is considered a science. Thus, software engineer has to acquire the knowledge on the networks, databases, hardware, business environment in which software is implemented, technologies, algorithms, accuracy, security, analytical thinking, as well as the experimental knowledge of engineers, scientific knowledge, management, etc.

There is also disagreement between professionals in this area. Some state that programming is an art, pointing out the aesthetic aspects of software development and claiming that science cannot provide the freedom of creativity. However, those who say software is a science relate the existence of numerous errors in software to the creativity basing on the use of scientific knowledge to increase the reliability of software. Thus, we can generally conclude that: The development of software is an engineering which combines an art and science basing on previous experiments. Software developers should be an architect when designing, an artist when building the user interface, and a master when installing and testing the software.

SE applies algorithms, functional design methods, quality assurance methods, and other methods for software development. After designing, the construction engineers determine the materials depending on the purpose of the construction. For example, engineer should not apply expensive materials for the construction of the pedestrian bridge over the river. On the contrary, the use of poor quality materials in the construction of the railway bridge can lead to major disasters. However, the quality of materials in the construction can be verified with physical laws or processes. These principles should be taken into consideration when developing software. Software engineer should choose the right methodology, testing tools and standards, which will be applied during the development process, basing on the appointment of the software. The implementation of more stringent and standardized processes is very important in critical systems such as medical equipment, space industry, finance, and so on [6].

Software engineers use both scientific (mathematical, economic and social) and practical knowledge when designing and managing software systems. In recent years, the software has been assembled from ready modules (reuse, application, asset provision, etc.) on a conveyor basis. Instrumental and technological systems and environments are used as software automation tools [7].

Thus, SE is considered a discipline in the crossroads of science, practice and art. The success of software depends on the right selection of each combination (Figure 1). For example, scientific and technical knowledge in the field of security is more important, while creative potential plays a dominant role in game programming [8].

Obviously, any science is interrelated with other sciences. As a science, SE mutually integrates with engineering, informatics and mathematics. It bases on the theory of algorithms, mathematical logic, control theory, theory of multiplicities, theory of proof, and others. Fundamental sciences and informatics integrated into SE make up its scientific-theoretical fundamentals (Figure 1) [9].

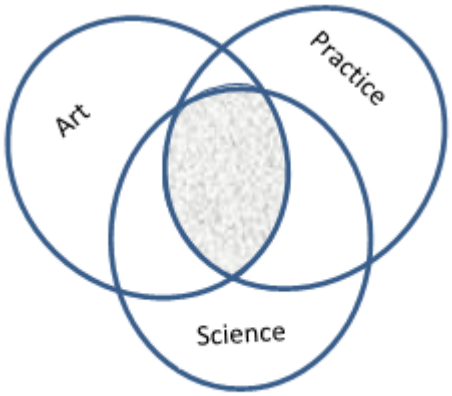

Figure 1. Software engineering

3. FUNDAMENTAL BASIS OF SOFTWARE ENGINEERING

Discrete mathematics is generally considered as the basis of computing. A software engineer should be aware of the fundamentals of discrete mathematics, its application in the field of informatics and the theory of probability in the first years of university. The software engineer should also acquire knowledge in the field of functions, relations, multiplicities, Boolean algebra, numerical logic, theory of numbers, basis of calculation, graphs, trees, matrices, finite machines and so on. Mathematical analysis is not the main subject for the SE, although it develops the abstract thinking, and the software engineer should know this subject as well.

One of the innovative aspects of modern information technology is the development and application of new approaches to the software quality assurance. In recent years, Machine Learning and Soft computing methods have been introduced to eliminate the real problems in modeling and predicting the quality of software [10, 11].

The Computing Curricula 2013: Computer Science describes the differences between the knowledge provided by universities and colleges on software engineering and illustrates common principles of teaching [12]. The curriculum of CS2013 shows 18 basic knowledge fields of SE:

- AL – Algorithms and Complexity;
- AR – Architecture and Organization;
- CN – Computational Science
- DS – Discrete Structures
- GV – Graphics and Visualization
- HCI – Human-Computer Interaction
- IAS – Information Assurance and Security
- IM – Information Management
- IS – Intelligent Systems
- NC – Networking and Communication
- OC – Operating Systems
- PBD – Platform-Based Development
- PD – Parallel and Distributed Computing
- PL – Programming Languages
- SDF – Software Development Fundamentals
- SE – Software Engineering
- SF – Systems Fundamentals
- SIPP – Social Issues and Professional Practice

During the SE training, the students should master such an idea that SE is not only a programming, but also a complex of knowledge with a wide range of applications.

Modern computer technology ranges from micro-sensors to highly productive clusters and distributed clouds. Computer programs affect all aspects of modern life. All aspects of PM education should be taken into account. Students need to comprehend all the capabilities of using computer technology.

4. INFORMATION SCIENCE AND PROGRAMMING IN SOFTWARE ENGINEERING

SE incorporates all theoretical and practical achievements of Computer science. Scientific aspect of SE consists of theoretical and formal methods and tools for building complex software systems. In addition, programmers have developed new methods for managing collective labor, measuring and evaluating their performance.

The aim of other fundamental sciences is to acquire new knowledge to address relevant issues. Whereas in SE, knowledge refers to the theory of computer programs. The discipline of programming includes the followings:

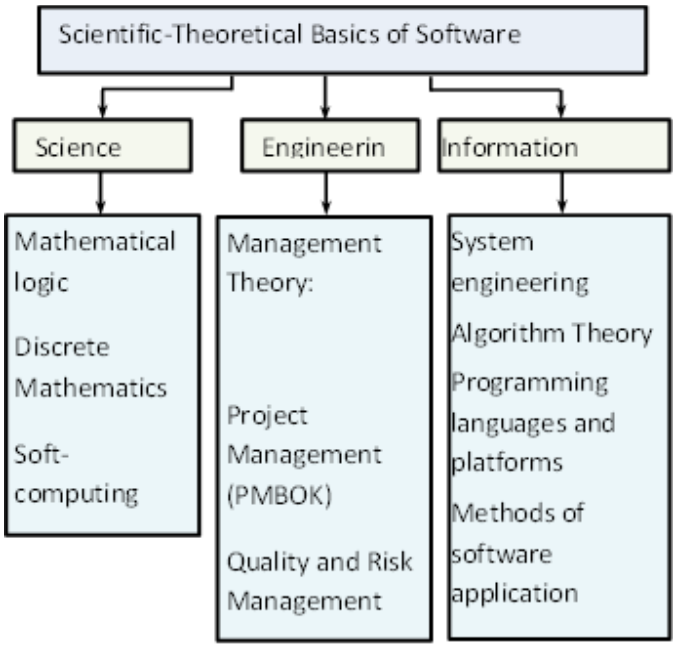

Figure 2. Scientific-Theoretical Basics of Software Engineering

1. ***Basic concepts.*** The core concepts of SE include data and their structure, functions, base objects (module, component, frame, reusable component and targeted objects (software, software system, software design, complex software systems).

2. ***Algorithms theory.*** Includes typical algorithms, computing functions, Turing machine, Post machine, algorithmic calculations, flow graph, models of algorithms and programs, complexity of algorithms, etc;

3. ***Programming theory.*** The basic concepts of programming theory are crucial for the comprehension of the modern methodology and technology of software development. Programming theory includes the following methods (Figure 2) [13]:

   - theoretical methods of programming (mathematical and algorithmic);
   - applied programming methods (object-oriented, component-oriented, aspect-oriented programming, etc.);
   - development of different computing models, comparative analysis based on modeling;
   - program accuracy testing through formal procedures (proof theory, idea, decision, verification);
   - Methods for software quality (reliability, security, accuracy, performance, functionality) assessment and so on.

4. ***Modern programming technologies.*** The Fourth Industrial Revolution requires the application of new programming technologies. Advanced languages and

tools that meet modern requirements are crucial for the development of high-quality software in Cloud Technology, Big Data, mobile Internet, the Internet of Things and other industries. Decomposition, distributed computing, parallel computing, parallel programming, forecasting and adaptive software development are realized with the help of modern programming languages and tools (language compilers, translators, and the tools supporting the development process). Such tools may include templates, software frames, various types of diagrams, and so forth [14].

At present, SE has expanded in all areas as a science with the programming as an element of part of it.

5. ENGINEERING SCIENCES IN SOFTWARE ENGINEERING

In addition to using the achievements of different fields of science, PM uses common engineering methods in managing software products (planning, distributing work among project executives, assessing labor quality and quality).

Engineering activity at the PM covers areas related to software development. Here, as with other engineering products, the application of certain standards, management processes and so on. keeps its relevance. The basis for the engineering activity at PM is the following elements:

1. Theory of control:

- ***project management*** may include standard cases and processes, scientific principles, planning and control methods. In this case, Project Management Body of Knowledge (PMBOK) is used. This standard was developed by the US Project Management Institute [15];
- ***quality and risk management*** is a project quality and risk management system guided by the lifecycle, quality and management standards of software.
- ***collective management*** (scheduling, control of software life cycle processes, measurement and evaluation of intermediary and end product quality, their development period, and price regulation and certification) [16, 17].

2. ***SWEBOK*** (Software Engineering Body of Knowledge) is a set of theoretical concepts and formal rules to be applied in SE; and includes 15 sections (5 core and 10 complementary) [18]. The book illustrates the key concepts related to the software development in a systematic and structured manner. SWEBOK is separately neither standard, nor technique and model. Each section find out what method, tools and ways are used for software development.
3. ***Standards*** include a set of rules that regulate all processes of software life cycle; The core software standards are of ISO/IEC 12207 ("Software life cycle processes") and ISO/IEC 9000 series. Generally, the standards do not obligate to use a particular life cycle model or specific development technique, and do not specify requirements on the format and content of the generated documents.

SWEBOK, PMBOK and SE standards are interconnected and are used in the preparation of methods, recommendations and restrictions important for product development.

From the engineering point of view, SE is responsible for the technological processes for requirements formulation, software design and implementation, and for the completion of the project on the specified date. The project may include the engineers of different categories; they realize the specified theoretical methods and tools, standards and processes recommended by the SE SWEBOK knowledge base.

6. PRACTICAL PRINCIPLES OF SOFTWARE ENGINEERING

The main purpose of SE is to create computer programs, systems and tools using theoretical and engineering methods.

As a result of many years of practices of program developers, a huge experience and collective knowledge on the development of programs, software, and computer programs have been achieved. This knowledge is represented in the massively applied software. Moreover, with the participation of programmers and engineers (project designers, software product and process supervisors, testing engineers, quality engineers, etc.), theoretical and practical methods, tools, principles and guidelines have been developed. As a result of collaborative activity of theoreticians and specialists in this area, formal methods for software testing and verification, mathematical models and software quality assessment methods have been developed for reliability assessment.

In practice, the use of pre-designed programs, automated tools and the Internet resources eases the work of a programmer. These resources are accessible to any user, and can be paid or free of charge. There are several approaches to the use of ready-made resources:

- ***Application of automated means (CASE-tools).*** As the software complexity increases, its processing becomes increasingly challenging and costly. In this regard, manufacturers prefer Computer Aided Software Engineering (CASE), which is used to develop and accompany various stages of software life cycle. CASE tools include a set of automated methods and tools for software engineering.

- ***Application of reusable components*** results in resource savings when creating software systems [19, 20]. When designing the project, the overall structure, that is, the frame of the project is developed first. Then the components that are already applied in other projects or to be applied in further projects are selected and integrated into the system.
- ***Use of applications.*** In addition to components, applications can also be used in the development of the software system. Increasing complexity of the process of software products development enhances the demand for professional and highly-qualified software engineers in software industry. Scientists, teachers, and software industry engineers often argue on the disciplines which computer professionals should study. However, the use of special software products in education is unanimously adopted by the experts in software engineering. These applications may include IBM's Rational Application Developer, Rational Rose, Rational Asset Manager, Rational Software Architect, Rational Software Modeler, Rational Rhapsody, etc. These applications provide analysis, modeling and design tools, through which sustainable infrastructure, project documentation, business management and other operations are performed.
- ***Use of domain engineering.*** Practical tools of program system production, such as SWEBOK, PMBOK and other standards also include a new process of software lifecycle, namely domain engineering processes. These processes include domain analysis (finding relationships, constant and changing demands, concepts and models) and domain design (design of a carcass for reusable components, selection of technique, etc.). As a result of the application of domain engineering technology, an architectural base of software production is built, which stores reused components, applications and domains (products) documentation in repositories [21]. These repositories can be used as the Internet resources (Figure 6). Since 2014, the registration at domain zone .ENGINEERING specifically designed for designers, engineers, architects and programmers has been started. In general, it is designed for all professionals who develop, test and accompany complex industrial, technical & software complexes.

As in other areas of engineering, SE applies re-engineering and reverse engineering:

- ***Re-engineering.*** Over time, software is outdated, and this process is often called software aging. Numerous methods can be used to prevent the software aging. One of them is the application of reengineering in the PT. Reengineering is the replacement and redesign of existing software to improve its quality characteristics, reduce the cost of accompanying, minimize the probability of customer risk. Software engineering is a complex process and aimed at improving software code. It is used to increase performance, safety, scalability, without considerably changing the software functionality. The aim of re-engineering is to optimize the structure of the system and make the software more comprehensible, which in turn reduces the cost of its implementation.
- ***Reverse engineering.*** In some cases, prewritten software codes are unavailable for certain reasons. (For example, code deletion due to theft, decoding etc.) In this case, the program code cannot be written by the programmers. However, in the engineering process, a group of programmers analyzes a machine code and restores the algorithm and pseudo-code of the program [22]. An unauthorized use of Reverse Engineering is forbidden due to the copyright and patent protection of some applications [23]. Reverse engineering can also be implemented during Re-engineering.

Developing software products from ready-made components and applications resembles a conveyor method for material products manufacturing. The application of tested components results in improved software quality.

Taking into account all above mentioned, we can conclude that science and engineering, the standards in this area, SWEBOK, PMBOK, and experimental knowledge are integral parts of software engineering and are interconnected with one another through software life cycle processes, design and management of software systems (Figure 3).

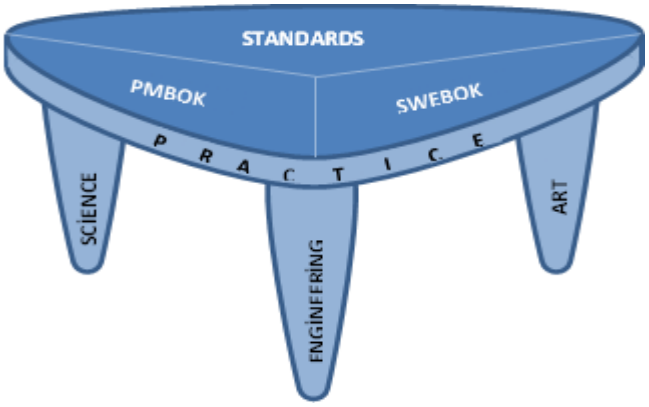

Figure 3. Fundamentals of software engineering

7. CONCLUSIONS

As an alternative to the oil sector in the economy of Azerbaijan, ICT and knowledge-based economy is widely development here. Therefore, the development of modern software and the training of highly-qualified software engineers are characterized as one of the topical problems in this area.

Since the late 1980s, the specialists have focused on the training of the students in the field of SE for the programming industry, and on the organization and management of software life cycle. Many articles focus not only on the importance of student training in the fields of programming and software engineering, but also on the need for the training of teachers for high-quality education process.

An innovative software product that combines the latest achievements of hardware and software, regardless of the area (automation, space, telecommunications, electronics, etc.), where it will be applied, should be developed. Development of software products is still a strategically important business process, and production of competitive software products in Azerbaijan is one of the chief issues [24].

Over the past 40 years, software engineering has evolved very fast. Nevertheless, comprehensive and well-grounded theoretical basics of SE for large-scale software projects have not been identified yet. Over time, theoretical basics of SE will become an integral part of the implementation of functional and non-functional requirements for security, reliability, ease of use, and accompanying software systems.

Software engineers use their knowledge in system engineering in the course of software development. They use different methods, tools, and processes to address emerging problems. Whereas the scholars in this area work on new theories and methods. Software engineering involves both theoretical and practical aspects.

This article examines the fundamentals of SE for teaching software engineering in Azerbaijan based on world practice. It provides recommendations on the preparation of text books and on the training og professional software engineers in Azerbaijan.

REFERENCES

[1] «ISO/IEC/IEEE 24765:2010, Systems and software engineering — Vocabulary », www.smaele.nl/documents/iso/ISO-24765-2010.pdf, pp.331.

[2] Naur P., Brian R. (7–11 October 1968). Software Engineering: Report of a conference sponsored by the NATO Science Committee(PDF). Garmisch, Germany: Scientific Affairs Division, NATO. http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF.

[3] T.H. Kazimov, T.A. Bayramova, Software engineering. Baku: "Information Technologies", 2013, 188 pp.

[4] С.А.Орлов. Программная инженерия: технология разработки программного обеспечения, 5-е изд. «Питер», 2017, 634 с.

[5] "Engineering" https://en.wikipedia.org/wiki/Engineering

[6] Искусство, наука и техника разработки программного обеспечения http://stevemcconnell.com/articles/the-art-science-and-engineering-of-software-development/

[7] В.С. Мараев, П.В. Пересунько. Современный Инструментарий Программной Инженерии, Издательство: "Агентство Международных Исследований", Уфа, № 9-1, 2016, с.39-41.

[8] Software craftsmanship-Between Science, Engineering and Art. https://www.pgs-soft.com/blog/software-craftsmanship-between-science-engineering-and-art/

[9] А.И. Орлов. Теория принятия решений. М:, Изд-во «Март», 2004

[10] K.Khatatneh, T. Mustafa. Software Reliability Modeling Using Soft Computing Technique. European Journal of Scientific Research, 2009, pp.147-152.

[11] N. Raj Kiran, V.Ravi. Software Reliability Prediction by Soft Computing Techniques, J. Syst. Software, 2007.

[12] http://www.acm.org/education/CS2013-final-report.pdf

[13] Е.М. Лаврищева, Методы программирования. Теория, инженерия, практика. Киев: Наук. Думка, 2006, 623 с.

[14] Е.М. Лаврищева, В.А. Петрухин, Методы и средства инженерии программного обеспечения. М.:МФТИ, 2007, 415 с.

[15] «PMBOK® Guide and Standards», https://www.pmi.org/pmbok-guide-standards

[16] http://www.intuit.ru/studies/courses/2190/237/lecture/6128

[17] Е.М. Лаврищева, В.А. Петрухин. Методы и средства инженерии программного обеспечения. Учебное пособие, Москва 2006, 302 с.

[18] P. Bourque and R.E. Fairley, eds., Guide to the Software Engineering Body of Knowledge, Version 3.0, IEEE Computer Society, 2014

[19] "Инженерия повторного использования компонентов", http://www.intuit.ru/studies/courses/2190/237/lecture/6134

[20] T.A. Bayramova The importance of self-management mechanisms to ensure software safety //International Research Journal of Engineering and Technology. Vol.: 02 Issue: 03 | June-2015.

[21] "Основные аспекты инженерии приложений и предметной области", http://www.intuit.ru/studies/courses/2190/237/lecture/6134?page=4

[22] R. Robbes, R. Oliveto, M.Di Penta, Guest editorial: special section on software reverse engineering // Empir Software Eng, 2016, pp.749–752.

[23] Q. Wang, N. Meng, Z. Zhou, J. Li and H. Mei, "Towards SOA-based Code Defect Analysis", IEEE International Symposium on Service-Oriented System Engineering, (SOSE'08), 2008.

[24] T.H. Kazimov, T.A. Bayramova, Problems of teaching software engineering in Azerbaijan // Problems of information society, 2017, №1, pp.92–96.