E-ISSN: 2664-8784
P-ISSN: 2664-8776
IJRE 2025; 7(1): 37-42

www.engineeringpaper.net
Received: 05-01-2025
Accepted: 10-02-2025

Shafagat Mahmudova
Department of Software Engineering and Systems Analysis, Institute of Information Technology, Baku, Azerbaijan

Correspondence
Shafagat Mahmudova
Department of Software Engineering and Systems Analysis, Institute of Information Technology, Baku, Azerbaijan

Development of a conceptual model for software system maintenance

Shafagat Mahmudova

DOI: https://www.doi.org/10.33545/26648776.2025.v7.i1a.73

Abstract
One of the eight main indicators of the software system effectiveness is its maintenance. The main goal of software system maintenance is to preserve the functionality of the software system over time. Software system maintenance is a broad activity and includes the correction of errors that have arisen periodically after the software system is delivered to the customer, increasing its functionality, removing obsolete functions and introducing new ones, etc. The software maintenance and its costs constitute more than 20-25% of the total expenses of the software system. There are various models of the software system life cycle, and the choice of the model depends on the given project. Each software engineer must have sufficient knowledge in this area to choose the appropriate model of the software system life cycle based on the content and requirements of the project. In these models, the last and most important stage of the software life cycle is software system maintenance. The main purpose of this work is to study the process of software system maintenance and its main issues, types, categories, stages, and to develop a conceptual model based on them. The goal is to achieve an efficient software system based on the conceptual model.

Keywords: Conceptual model, software system, types, models, categories

Introduction

Software system (SS) maintenance is an integral part of the software system life cycle.

Software efficiency (ISO/IEC standard 25010: 2011 (R ISO/MEK 25010-2015 state standard) defines the quality model of the product as it includes eight top-level features (Figure 1) [12]:

SS maintenance is an integral part of the software system life cycle. It includes the process of debugging, optimizing, and correcting defects of software after its commissioning. It is possible to add new functions to improve the application of SS by making changes to the program to correct the defects discovered during the operation [1].

In software engineering, the life cycle of a software system is a sequence of steps for designing and developing software. During these stages, program maintenance plays an important role. After the delivery of the software system, certain changes are required on it for some reasons. These changes may be related to the presence of bugs in the program, expansion of the software functionality, etc. The main purpose of the software system maintenance is to eliminate the problems arisen. This paper touches upon these issues. It examines the models and stages of the software life cycle and studies the process of its maintenance. The problems appeared during the software system maintenance are identified.

The main goal of the software system life cycle is to develop a high-quality, effective and efficient software system (SS) that meets user requirements. To organize SS quickly and efficiently, its life cycle is grouped into different stages. The number of these stages varies depending on the purpose for which the software system is designed and used. This number often varies between five and seven [1].

There are different models of SS life cycle and choosing the right model depends on the given project. Every software engineer should have enough knowledge in this area to choose the appropriate software life cycle model based on the project content and requirements. In these models, the last and most important stage of the life cycle of a software system is its maintenance.

The main goal of the SS maintenance is to preserve the performance of the SS over time. The SS maintenance is one of the broad issues, which includes the detection and correction of errors that have arisen after the delivery of the software system to the customer, increase of

work capacity, removal of obsolete functions, optimization, etc. The cost of the SS maintenance is more than 20-25% of the total cost of the software [2].

Materials and Methods

Errors encountered during SS maintenance

Errors that cause a program or SS to behave unexpectedly and result in incorrect results. Most errors encountered during SS maintenance refer to the are errors made by program developers in the source code or design. Some errors also occur due to the malfunction of compiler tools, for example, the compiler producing incorrect code [14].

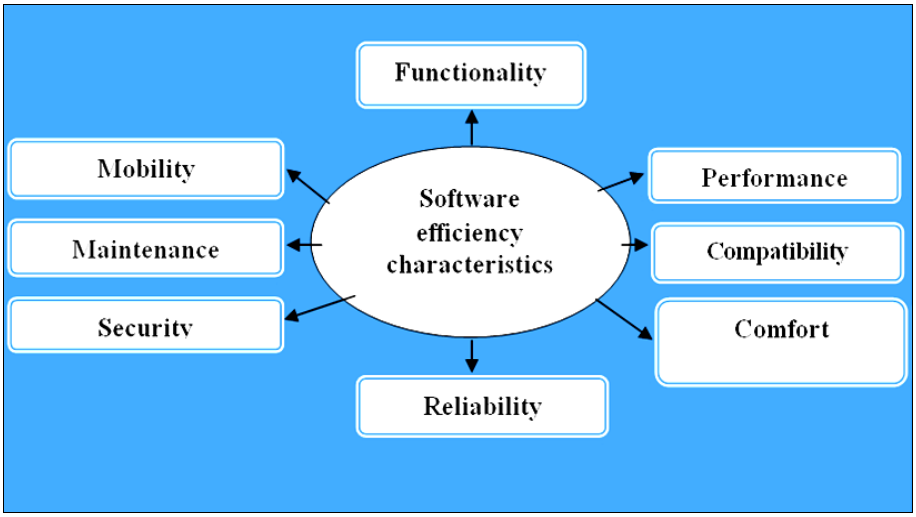

Fig 1: Software efficiency characteristics

The term "SS error" is often used for the errors occurred during program execution, as opposed to, for example, design errors or syntax errors.

Errors encountered during SS maintenance are classified by their severity:

- Blocking: makes the program execution impossible;
- Critical: deprives the SS of all benefits;
- Serious;
- Minor;
- Cosmetic.

When they appear

- Permanent, at every start of the SS execution;
- Occasional;
- Only on the executor's computer (depending on the settings).

By location and direction

- User interface errors;
- Computing errors;
- Data processing;
- SS overload;
- Test errors.

Depending on the nature of the error, the program and the implementation, errors can be immediately visible or, conversely, remain unnoticed for a long time.

Errors can also appear as vulnerabilities that allow unauthorized access to the system or a DoS attack.

Choosing the right model is of great importance for the successful development and completion of the project. The choice appropriate model mainly depends on the requirements, system complexity, project size, cost, etc. Each model has its own stages.

Collection of requirements set by the user: This stage collects information from the user about software to be created and for what purpose it will be used. Requirements are grouped as user requirements, system requirements, and functional requirements. Figure 1 presents these requirements [3].

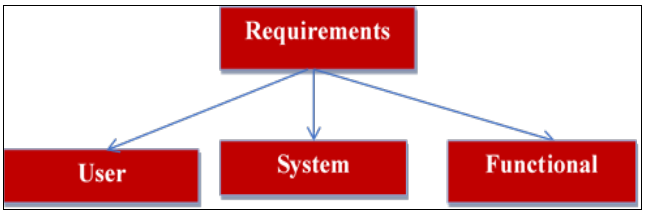

Fig 1: Software requirements

User requirements

One of the most important and difficult stages of creating software is determining what the user wants. The reason for this, in some cases, may be that the user cannot fully coordinate their needs and desires, the information provided is incomplete, inaccurate and contradictory. The user defines own needs and tells what he/she wants to do with the system. User requirements are considered the main input for creating system requirements.

System requirements

System requirements are formed based on user requirements. They may include accuracy, availability, compatibility, effectiveness, privacy, security, quality, etc.

Functional requirements

Functional requirements are the functions that a system or its components are intended to perform. Calculations, technical data, data manipulation, data processing, and other special functions refer to the requirements.

Software projecting

This stage consists of designing SS according to all requirements.

Coding: This phase includes programming. At this stage, the program code is written using the selected programming language and it is ensured that it works efficiently without errors.

Testing phase
The testing phase is conducted by testing experts. This phase begins after the coding phase is completed. Early detection of errors and their correction is one of the main conditions for the development of a reliable SS.

Implementation phase
In this phase, SS is installed on the user's computer and tested in a real environment.

Maintenance
This covers the next step after SS is delivered.

Related works
The ability to monitor the progress of students' academic achievement is an important issue for the academic community of higher education. A system for analyzing students' results based on cluster analysis is described. It uses standard statistical algorithms to organize their score data according to performance level. This paper also applies the k-means clustering algorithm to analyze student achievement data. The model is combined with a deterministic model to analyze the results of students of a private institution in Germany, which is a good benchmark for tracking the progress of students' academic performance in Tertiary Institutions for effective decision making by academic planners [16].
Programming is a course that is estimated quite difficult for most students. Students are required to be proficient in all processes. Computer programming skills require a lot of practice through lab work assignments. Managing and evaluating the results of a student's laboratory assignment is a complex and time-consuming task. Availability of automatic programming evaluation tools to receive and automatically correct and evaluate the results of laboratory work can facilitate the teacher's work. Grouping students by performance level makes it easier for teachers to track student performance levels and provide learning tailored to students' abilities. Clustering is implemented using the K-Means clustering method. The data is obtained from the Automatic Programming Assessment Tool lab assignment. Based on the clustering results, there were 3 groups according to their abilities, that is, 16 people in the middle ability group, 11 people with high ability and 14 people who still lack programming ability [5].

Software maintenance and its stages
Maintenance, which is the last phase of the SS life cycle, is one of the main issues. Meir. M. Lehman spoke for the first time about the maintenance of SS and its evolution in 1969. His two decades of research resulted in the validation of Lehman's laws. Based on his research and key findings, tracking helps software development and understanding what happens to a software system over time. Lehman demonstrated that systems evolve over time. The main purpose of SS maintenance is related to solving the problems shown in Figure 2 [6].

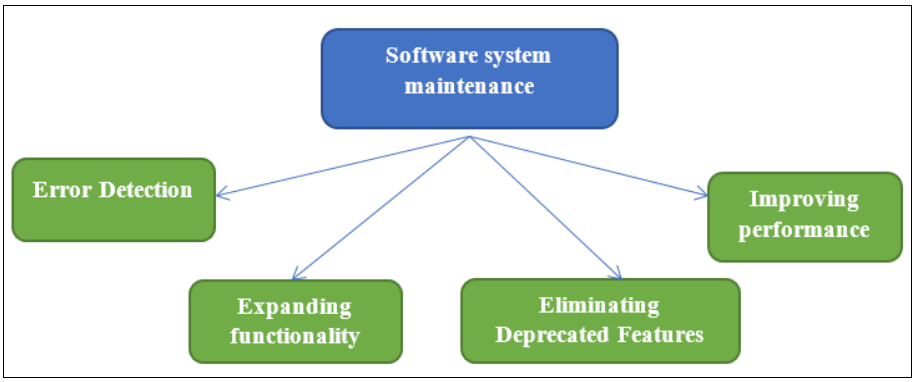

Fig 2: Software maintenance tasks

Error Detection
In SS, error detection and correction are one of the top priorities for the smooth operation of the program. Here, errors in the code are searched, found and corrected [7]. Errors can be in hardware, operating system or any part of the program. Correction of these errors should be implemented not to damage existing software functions.

Expanding functionality
This enables the enhancement of features and functionality to ensure that applications are relevant to the changing market environment.

Eliminating Deprecated Features
Deprecated features are harmful and affect the efficiency of the problem-solving process by taking an important place. During SS maintenance, such codes are removed from the program and replaced by newly developed tools and technologies.

Improving performance: To improve the system, programmers identify problems through tests and solve them.
The SS maintenance is grouped into four categories [5]. Figure 3 illustrates these categories [8].

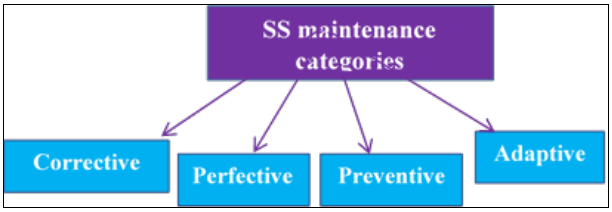

Fig 3: Software maintenance categories

Corrective: correcting errors in the program. Errors in SS can be of different types. These are coding errors, design errors, and requirements errors. Coding errors are corrected by programmers. Errors related to requirements are more important. Because SS is built according to certain requirements, and if these requirements are wrong, the system needs to be revised.

Perfective: Adding or changing functionality to the system. Making changes to get a more efficient and functional system according to changing requirements refers to this category.

Preventive
Involves increasing the durability or reliability of the SS to prevent problems that may arise in the future.

Adaptive
Adapting the SS to a different operating environment. Adapting the SS life cycle to different environments is one of the vital factors. The main purpose of this category is to make the SS adaptable to different environments.
During the SS maintenance, the above-mentioned categories are represented in percentages as illustrated in Figure 4.

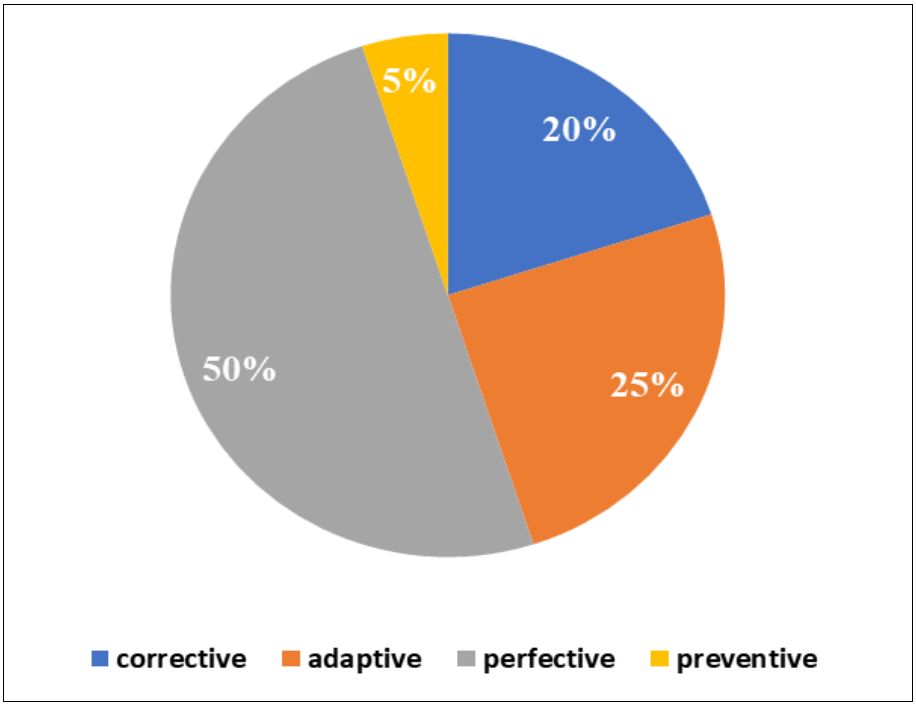

Fig 4: Percentage of software maintenance categories

The SS maintenance process consists of seven different stages
Change management: In this stage, the user defines the changes. The category of the change to be made is also defined.

Analysis
The scope of each approved change is determined and a plan is developed to make the changes to the SS.

Project: This stage actually plans the changes to the system. This existing system contains information about project documentation, databases and existing software, analysis phase. It aims to compile changes in all categories.

Application: This stage includes coding, testing, assimilation (adaptation), integration, analysis of special code, etc.

Regression testing: Regression testing is performed to ensure that there are no errors in the system after changes are made, and after this test is performed, it is confirmed that there are no errors in the software [9].
Regression testing is a specific type of software testing that determines whether recent changes to the software or software parameters have negatively affected existing functionality.
A complete or partial test set of previously executed test cases is re-run during regression testing. In this way, it can be determined whether previously validated or existing functionality still works properly after changes.

Acceptance testing
The main purpose of this test is to check whether all the features of the software meet the requirements of the change.

Delivery
After acceptance testing is done, the application is delivered to the user. The final test of the system is performed by the customer after the system is presented. The stages of the SS maintenance are shown in figure 5.

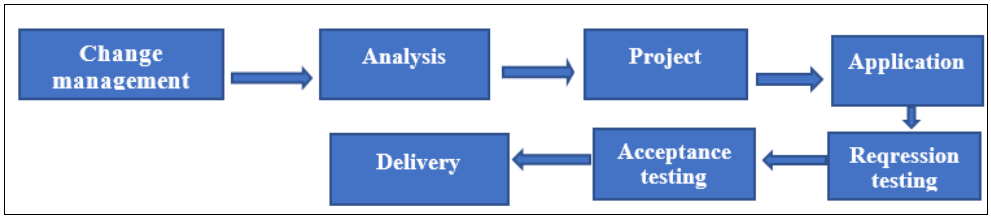

Fig 5: Stages of the software system maintenance

Different models are used to solve problems in SS. These models offer different approaches and methods to facilitate the maintenance process. The most commonly used model is the following [10, 13].

Quick-Fix Model: The main goal of this model is to identify the problem and solve it as quickly as possible. The advantage of this model is the quick problem solution with less cost [11, 12].

Development of a conceptual model for the software system maintenance

Taking into account the above, a conceptual model is developed for the SS maintenance (Figure 6).

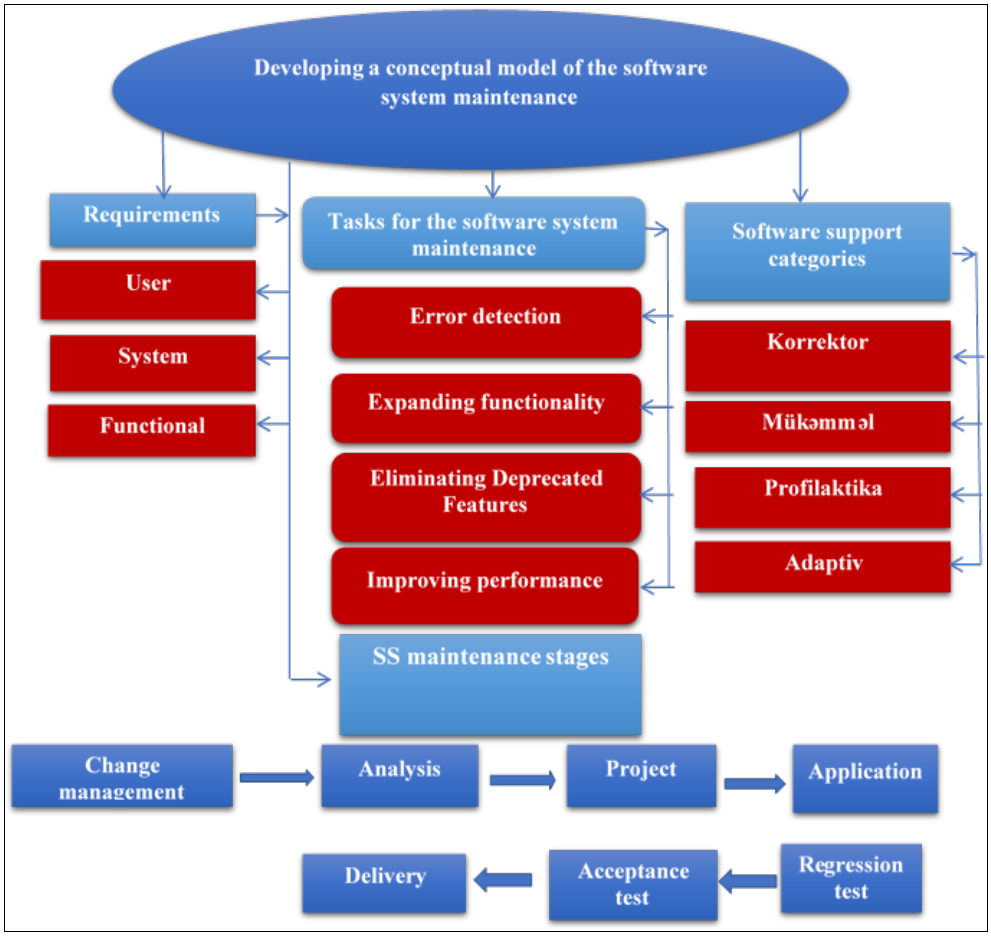

Fig 6: Conceptual model of SS maintenance

Discussion: From the work done, it is clear that the k-means clustering algorithm was used to analyze the student data. In Germany, the algorithm was combined with a deterministic model to improve student productivity in higher education institutions, resulting in a hybrid model. This makes it possible to achieve high efficiency. It is not easy for students to learn programming. It requires special abilities from students and takes a lot of time. Obtaining laboratory research results and the availability of programming assessment tools make the work of teachers easier. Grouping students according to their productivity level can help teachers to facilitate the process of learning programming. For example, clustering can be done using the K-means clustering method, etc.

Conclusion: The SS life cycle is a sequence of stages to create and develop it. The SS maintenance is of great importance in these stages. After the software is delivered, some changes are required on the SS for some reasons. These changes may be related to the presence of bugs in the software, expansion of SS's capabilities, etc. The main goal of the SS maintenance is to eliminate these problems. This paper touched upon these issues. It explored the models and stages of the SS maintenance and the process of its maintenance.

The k-means cluster algorithm was applied to the errors appeared during the maintenance of the software system and an experiment was conducted. The obtained results were satisfactory.

References

1. Aakriti G, Shreta S. Software maintenance: challenges and issues. International Journal of Computer Science Engineering. 2015;4(0):23-25. Available from: http://www.ijcse.net/docs/IJCSE15-04-01-037.pdf
2. Software Development Life Cycle. 2025. Available from: https://www.tutorialspoint.com/software_engineering/software_development_life_cycle.html
3. Software Development Life Cycle (SDLC) Phases, Models, Process and Methodologies. 2025. Available from: https://www.softwaretestinghelp.com/software-development-life-cycle-sdlc/
4. Anita Q, Rina H, Iwan NA, Asmunin A. Application of K-Means algorithm for clustering student's computer programming performance in automatic programming assessment tool. In: Proceedings of the International Joint Conference on Science and Engineering (IJCSE). 2020;196:421-425. doi:10.2991/aer.k.201124.075
5. Why Software Maintenance Is Necessary. 2025. Available from: https://simplified-itoutsourcing.com/blog/why-software-maintenance-is-necessary
6. Mattapullut G, Rughoobur P, Ramdoo Vimla D. Review of software maintenance problems and proposed solutions in IT consulting firms in Mauritius. International Journal of Computer Applications. 2016;156(4):1-5. doi:10.5120/ijca2016912414
7. Usman A, Usman MY. Impact of software comprehension in software maintenance and evolution [Master's thesis]. Computer Science. 2010. p. 1-59.
8. Othman MY, Suhaimi I. Evaluating software maintenance testing approaches to support test case evolution. International Journal on New Computer Architectures and Their Applications. 2011;1(1):74-83.
9. Abdullahi YE, Ogwueleka FN. Software system development life cycle model for improved students communication and collaboration. International Journal of Computer Science & Engineering Survey. 2017;1(4):1-10.
10. Software Maintenance Models. 2018. Available from: http://www.professionalqa.com/software-maintenance-models
11. Mahmudova Sh. Development of a method for processing log files using clustering. Soft Computing. 2023;27(3):1617-1628.
12. MacQueen J. Some methods for classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability. 1967;1:281-297.
13. Kriegel HP, Schubert E, Zimek A. The (black) art of runtime evaluation: Are we comparing algorithms or implementations?. Knowledge and Information Systems. 2017;52(2):341-378. doi:10.1007/s10115-016-1004-2
14. Software error. 2025. Available from: https://ru.wikipedia.org/wiki/Software_error