Software Maintenance Model through the Development Distinct Stages

[1]Ahmad AbdulQadir Al Rababah, [2]Ahmad A. Alzahrani
[1]Faculty of Computing and Information Technology, King Abdulaziz University, Rabigh 21911, Saudi Arabia.
[2]Faculty of Computing and Information Technology, King Abdulaziz University, Saudi Arabia.

Summary
With the rapid development of software systems and the growing demand to follow the results of these systems, in the face of intense competition to provide the best outputs, it became necessary to pay attention to the maintenance phase of software systems, and work to develop them to serve their users effectively through qualified results, taking into account the saving time, effort and cost. The proposed model is concerned with how to maintain the software projects starting from the pre-testing stages of the software project[1]. The research will provide in its initial phase the maintenance method from the implementation stage, where this stage is one of the most important stages for the formation of software projects, because it occupies a high weight in building[2], developing and evaluating the requirements of the issues in various domains. This manuscript will provide a design model for the application of maintenance for software projects while saving time, effort and cost, in additional to ensure positive results with high performance and efficiency[3].

Key words:
Software System

1. Introduction

High-quality software and the use of modern technologies are considered the guarantee of the uninterrupted operation of the enterprise. This also affects its financial state[4]. Therefore, companies are increasingly demanding quality and testing software. To reduce the risks of introducing a low-quality product, improve the level of software availability and reduce the financial costs of owning software. It is important to check whether this product meets the requirements for business, whether it can work and develop in modern conditions or not[5],[6].
Operations on software testing include: Functional testing is based on an analysis of the functionality of a component or system; Automated testing implies software for performing tests, which allows reducing testing time and simplifying its process; Performance testing - determines the performance of the software product; Load testing - allows you to assess the capabilities of the system or application in accordance with predefined objectives; Stress testing - allows you to assess the capabilities of the system or application in conditions of exceeding the limits of normal operation; Localization testing is done to make sure that the localized product is fully functional and linguistically correct and that there were no problems during the localization; Compatibility testing is the process of testing the system with each of the software and hardware configurations for which support services are provided; Usability testing allows you to determine how much the software product is understandable, easy to learn, easy to use and attractive to users, provided they are used under specified operating conditions; Security testing - allows you to assess the security of the software product; Testing the installation is aimed at checking the successful installation and configuration, as well as updating or removing software and hardware[7],[8],[9].
This service has a modular structure, so you can choose only those elements in which your business really needs. This helps create a testing process that best meets the project requirements and allows you to spend your current budget more efficiently. Independent testing is: Objective assessment of software quality level, as well as compliance with the requirements of your company; the right step, which serves as a guarantee of reducing the number of hidden defects (errors) in software development. The company's specialists are highly qualified professionals who have undergone serious training in software testing[2],[1].

2. Life Cycle Stages of Software Systems

The phases of software systems development determined as a lifecycle period of their creation and use, covering various states, starting from the moment the need arises for such a system and ending with the moment of its complete withdrawal from use by users. The life cycle of information systems includes four stages: pre-design, design, implementation, operation. The quality of the system depends on the quality of the design work; therefore, each stage is divided into a number of stages and provides for the preparation of documentation reflecting the results of the work. At the pre-project stage, some steps can be distinguished, like a collection of materials for design - provides for the development and selection of options for the system concept, the identification of all the characteristics of the object and management activities, the flow of internal and external information communications, the composition of tasks and

Manuscript received February 5, 2019
Manuscript revised February 20, 2019

specialists who will work in new technological conditions, their level of preparation as future users system. And then analysis of materials and documentation - the design assignment, the approval of a feasibility study. For the successful creation of a management information system, the ways of passing information flows are studied comprehensively, both within the enterprise and in the external environment.

Creating an information management system for an organization is a rather complicated and time-consuming process. The most typical and simple form of changing a company is automation. A deeper form of organization change — derived from automation — is the rationalization of procedures. A deeper change in the company is business process reengineering. Its essence lies in the analysis, simplification and modernization of business processes. New information systems can fundamentally change the structure of the entire organization, changing the way the company operates, or even the direction of its activities. Such a more radical form of changing the company's activities is called a paradigm shift. A paradigm shift implies a revision of the nature of the activity, not of individual procedures and processes, but of the company itself.

Depending on the type, scope and requirements of the project, the development procedure is determined[10]. It will be somewhat different for the development of mobile applications, embedded software, automation solutions and databases, but the general sequence of actions for creating software is universal as shown on Figure1:

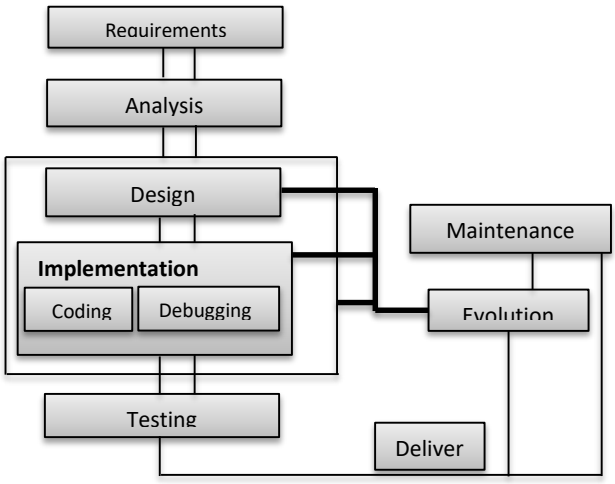

Fig. 1 General Model of maintenance from testing phase

Software evolution. Since the mid-50s. began a new period in the development of computing, associated with the advent of semiconductor elements. During these years, the first algorithmic languages and the first system programs compilers appeared. In 1957, FORTRAN was created, in 1960 - COBOL, Algol and Lisp, in 1964 - Basic, Simula, PL / 1, in 1970 - Pascal and Smalltalk. By the end of the 60s. The number of programming languages exceeded one thousand. Practically all the basic concepts procedural, logical, object-oriented programming was proposed at this time. In subsequent years, progress in programming automation did not go towards the creation of new languages, but, on the contrary, along the path of natural selection. Programming languages were born and died, but only some of them - the most persistent and viable - survived to the beginning of the XXI century, and became standard in the international community of programmers. Another achievement of the 60s. is the appearance of the first batch processing systems. The cost of CPU time has increased in order to increase the efficiency of using a computer, tasks with similar required resources begin to come together to create a task package. Batch processing systems were a prototype of modern operating systems (OS), they became the first system programs designed to control the computing process. Packaged OS greatly facilitated the work, and at the same time increased the efficiency of the use of computers.

In the 70s. in the technical base there was a transition to integrated circuits, which gave great opportunities to the new generation of computers. There are families of software compatible machines. The first family of software compatible machines built on integrated circuits was the IBM / 360 series, which significantly outperformed the second generation machines in terms of price / performance. Soon the idea of software compatible machines became universally accepted.

Computers were still very expensive, but their power and reliability increased dramatically. Then it began to create large information systems for industrial and commercial enterprises, banks, social institutions with displays connected to the central computer located in the computer center of the company appeared on the users' workplaces. To organize the computational process in these conditions, we needed operating systems of a new type, which allow organizing a dialogue between a large numbers of users in the time sharing mode. There were interactive OS.

Currently, the development of software designed for a wide range of users is no longer taking place in the competition of individual programmers, but in the process of fierce competition between software manufacturers. In the United States alone, more than 50 software manufacturers have sales of more than $ 10 million, and ten of them (including Microsoft, Lotus, Novell, Borland, Autodesk, Symantec and Computer Associates) have sales in excess of $ 100 million. Non-commercial software is constantly declining and increasingly limited to programs created in the process of scientific research or for personal pleasure. When developing commercial programs, the main task of development companies is to ensure their

success in the market. For this, it is necessary that the programs have the following qualities:

-program functionality, i.e. Completeness of its satisfaction of user needs;

-visual, convenient, intuitive and familiar to the user interface;

-ease of learning the program, even for novice users, which uses informative prompts, built-in directories and detailed documentation;

- Reliability of the program, i.e. its resistance to user errors, equipment failures, etc., and its reasonable actions in these situations.

Software Design – one of the most important components of the product after the technical characteristics, affecting the efficiency and speed of user interaction with it[11]. The design stage is divided into two sub stages: first is the stage of technical design - design solutions are formed for the supporting and functional parts of the information system, modeling production, business and financial situations, setting the task and flowcharts and solving them.The second stage is the detailed design - the development and refinement of the system, the adjustment of the structure, the creation of various documentation are carried out: for the supply, for the installation of technical equipment, operating instructions, job descriptions.

Requirements analyzed for design are defined as a rule, simplicity, intuition and minimum costs for performing the action (achievement of the result), as well as beauty and conformity to the style of the company and (or) the product are important[12]. The analysis- design model is evaluated by the software group in an effort to define whether it holds faults, contradictions, or tasks among all various transactions ; whether well another possibility exist which shown on Fig.2; and whether the model can be applied within the restrictions plan and cost that have been recognized[13].

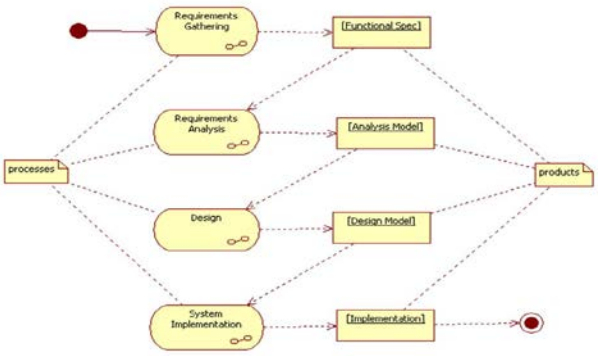

Fig. 2 Analysis-Design transactions Model

The **code** is that part of the work that is usually associated with software development as such. It is important that the code is sufficiently optimized, concise and understandable[14]. We assign the languages of the programmers specializing in their use to the tasks selected for the specific tasks[15, 16].

Software Implementation and Testing is carried out at each stage of software development Fig.3 , includes a lot of tests on the test plan, customized taking into account the specifics of the project at the stage of drafting the technical assignment. The test results are documented and available to the client in real time[4]. Payment for the product is made only after passing all kinds of tests, including client tests[17]. The stage of implementation of the information system involves initially the preparation for commissioning - at this stage installation of technical means, system tuning, personnel training, trial use are performed. And then as conducting pilot tests of all system components before launch. Lastly by putting the instanced results into commercial operation, that is issued by the act of acceptance of work. At the stage of operation of the information system in the operating mode, the adjustment of functions and control parameters is not excluded, also carried out operational service and administration.

Fig. 3 Software development testing

Software Maintenance, the program enters the maintenance phase when transferred to users. The software maintenance goals are to provide financial support for a distributed software product[18]. Just like software development, maintenance is also a frequent program changes, but the goals are significantly reduced: purposes are to maintain software that can be used for cost-effectiveness and in this case no ambition for adding new conditions or functions[1]. Developers changes made at this level are general corrections or adaptations to technological changes or uses Fig.4 Many software problems arise from the fact that the program is being used while maintaining it[4, 10]. **Maintainability:** Is the simplicity with which a program can be precise if a fault arises. Since there is no straight way of measuring this, incidental way has been used to measure this. It measures when a fault is exposed to determine how much time it takes to investigate the change.

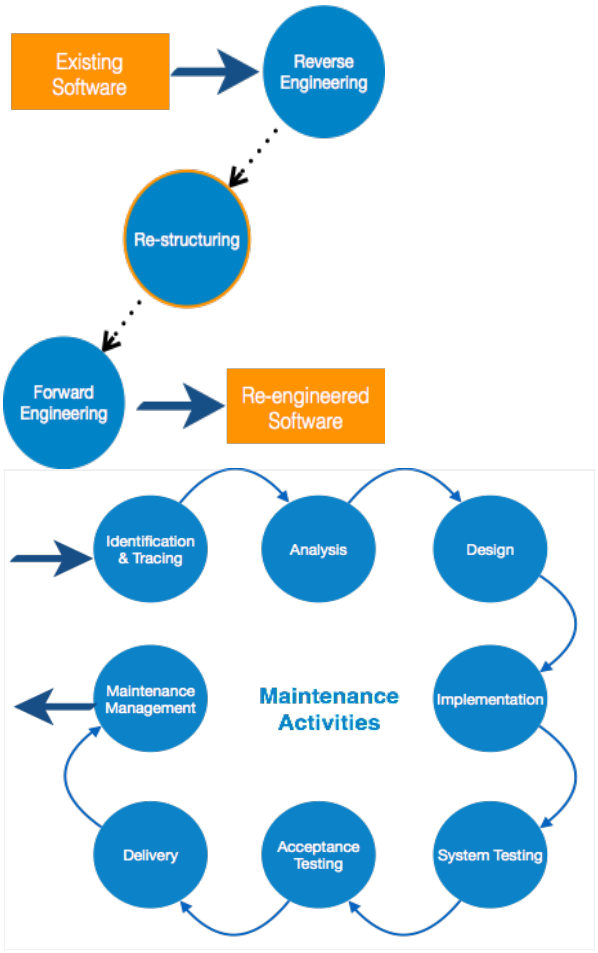

Fig. 4 Software Maintenance overview and activities

The unique evolution reason is considered the knowledge volatility of the parties concerned, in which the parties concerned learn how to manage this technique. Stakeholders have the learning process that found suitable solutions, so no reason for additional development and the software will reach the enough stability level. Therefore, maintenance still may be necessary[2]. The above reasons for the evolution of the program are the volatility of requirements, technology or knowledge of the parties involved. Once the fluctuations are over, the software project reaches stability[19]. The non-original development team can maintain the program. During the maintenance period it is normal that there are at least two versions of the program: a copy in production is maintained (maintained), while the development team develops simultaneously a different version for future releases[20].

System Documentation is a procedure that fixes the plan, process and result of software development. Includes all the initial information, work plans, costs, testing, tasks list of performers at each point in time, work reports and so on[21]. Documentation is necessary for the rapid and accurate identification of errors, transparency of joint work, as a mandatory legal part of the contract[1, 2].

3. Propesed Models

As mentioned in the literatures, the coding-step value of the software projects is around 60%, that means the all other project development stages are taking only 40%, it leads us to conclude that coding and debugging as sub stages of the implementation are weighted 70% at least, while the remaining steps, including the requirements analysis, design, and testing will have just 30% that means 10% for each. So it is very useful and performed to start the maintenance before the testing phase of the software project[5]. In this research, the maintenance model of software projects was divided into three sub models as shown on Fig.5:

- Sub Model 1: Maintenance to (Implementation, Design)
- Sub Model 2: Maintenance to Implementation
- Sub Model 3: Maintenance to Design

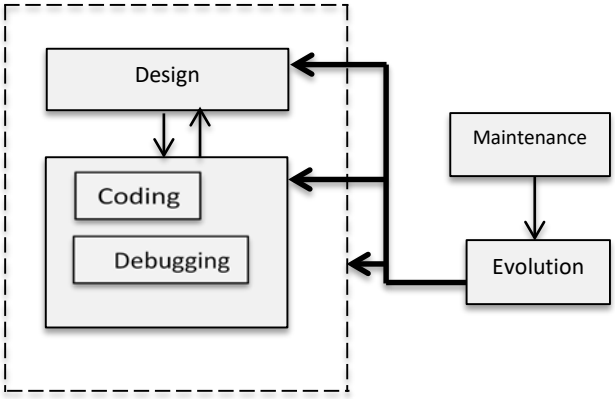

Fig. 5 Proposed Models of maintenance starting before testing Step

First sub model: After the maintenance request, we return to the stage of participation (design - implementation). In this way, we have saved the time, cost and effort we take in the testing phase of the software project that may equal no less than 10% of the overall value. There must be strong coordination between the two phases of design and implementation (coding part)[9] in order to maintain cohesion requirements with a high quality and efficiency to be used as necessary part for the next stages of the projects development[22], in this moment the debugging part is not needed and its value becomes = 0, this method is represented on Fig. 6.

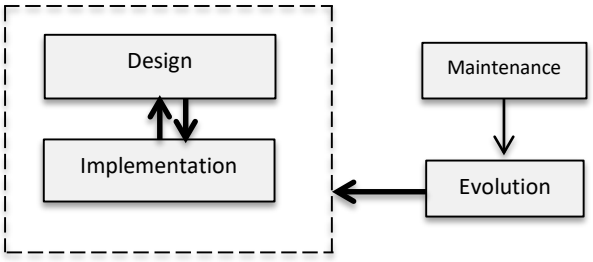

Fig. 6 Sub Model of maintenance: Implementation-Design

Second sub model: As the maintenance request will arise, this model will activate the link between both system stages maintenance and implementation directly. So this model will jump over the testing, taking into consideration the implementation steps coding and debugging that will have values (60% + 10%) with saving of testing weight(time, cost and effort) 10%. The internal relationship between the implementation steps has to be very consistent and accurate for exact succession of the programming side of the project[23], this method is represented on Fig. 7.

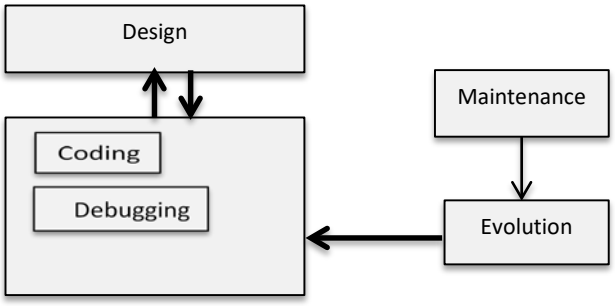

Fig. 7 Sub Model of maintenance starting from the implementation step

Third sub model: In this type of maintenance of the software projects, the maintenance request is answered by direct return to the design stage, so that the maintenance relationship is activated by the design without attention to the implementation and testing stages[24]. Therefore, the total time, cost and effort used for these stages in the previous first and second sub models will be saved and totally it becomes nearly = 0, for this sub model the project managers must focus on managing people and project teams distribution. It is a must to merge the team of implementation with the design team according to the type and the size of the software project in maintenance-design demands[22]. Such created new team will play the role of the hidden implementation and the design as well; especially it is well known that its weight is around 70% in additional to 10% of the testing weight [7, 25]. This sub model is illustrated on Fig. 8.

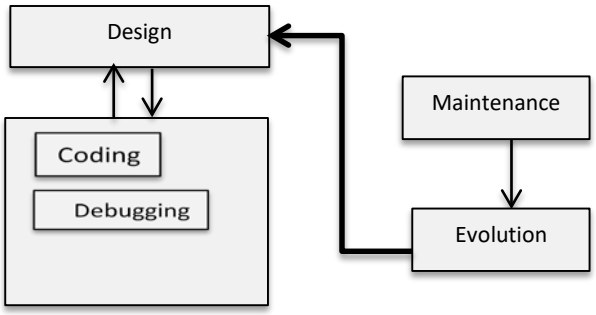

Fig. 8 Sub Model of maintenance starting directly from a design step

In the following table 1. There is a distribution of the importance value that is distributed to all stages of development of the software systems such as analysis, design, implementation and testing.

Table 1. Value distributions among the system development stages in several proposed models

	Analysis %	Design %	Implementation %	Testing %	Total usage %	Total saving value %
Existing Model	10	10	60 + 10	10	100	0
Sub Model 1	10	10	60 + 0	0	80	20
Sub Model 2	10	10	60 + 10	0	90	10
Sub Model 3	10	10	0 + 0	0	20	80

4. Conclusion

By implementing and conducting the necessary examination and analysis on the results of the programming and the compatibility of those results with the requirements of the user based on the issue required to establish the software system. It was clearly found that the efficiency of the software system increased by 20% when applying the first model: design – implementation, also it was increased by 10% when applying the second model: implementation based, and lastly when applying the third model- design based, the efficiency may be increased up to 80%, this means that the time, effort and cost of the software projects will be reduced by those percentages when following any of these three models mentioned above.

5. Acknowledgement

This work was funded by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, under grant No. (D-161-830-1439). The authors, therefore, acknowledge with thanks DSR technical and financial support.

References

[1] Grieskamp, W., et al., Model-based quality assurance of protocol documentation: tools and methodology. Software Testing, Verification and Reliability, 2011. 21(1): p. 55-71.

[2] Al-Rababah, A.A., T. AlTamimi, and N. Shalash, A New Model for Software Engineering Systems Quality Improvement. Research Journal of Applied Sciences, Engineering and Technology, 2014. 7(13): p. 2724-2728.

[3] Asuncion, H.U., A.U. Asuncion, and R.N. Taylor. Software traceability with topic modeling. in Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1. 2010. ACM.

[4] Hoefler, D., et al., Software maintenance management. 2012, Google Patents.

[5] Al-Rababah Ahmad, A., UML–Models Implementations in Software Engineering System Equipments Representations. International Journal of Soft Computing Applications, 2009(4): p. 25-34.

[6] Fitzgerald, B. and K.-J. Stol, Continuous software engineering: A roadmap and agenda. Journal of Systems and Software, 2017. 123: p. 176-189.

[7] Vyatkin, V., Software engineering in industrial automation: State-of-the-art review. IEEE Transactions on Industrial Informatics, 2013. 9(3): p. 1234-1249.

[8] Ciccozzi, F., et al., Model-Driven Engineering for Mission-Critical IoT Systems. IEEE Software, 2017. 34(1): p. 46-53.

[9] AlRABABAH, A.A., IMPLEMENTATION OF SOFTWARE SYSTEMS PACKAGES IN VISUAL INTERNAL STRUCTURES. Journal of Theoretical & Applied Information Technology, 2017. 95(19).

[10] Trivedi, S.H., Software testing techniques. International Journal of Advanced Research in computer science and software Engineering, 2012. 2(10).

[11] Siewiorek, D. and R. Swarz, Reliable Computer Systems: Design and Evaluatuion. 2017: Digital Press.

[12] Al-rababah, A.A. and M.A. Al-rababah, Module Management Tool in Software Development Organizations 1. 2007.

[13] Moustafa, A., et al., A New Dynamic Model for Software Testing Quality. Research Journal of Applied Sciences, Engineering and Technology, 2014. 7(1): p. 191-197.

[14] .Lewis, W.E., Software testing and continuous quality improvement. 2016: CRC press.

[15] Al Ofeishat, H.A. and A.A. Al-Rababah, Real-time programming platforms in the mainstream environments. IJCSNS, 2009. 9(1): p. 197.

[16] Newcomer, K.E., H.P. Hatry, and J.S. Wholey, Handbook of practical program evaluation. 2015: John Wiley & Sons.

[17] Utting, M., A. Pretschner, and B. Legeard, A taxonomy of model-based testing approaches. Software Testing, Verification and Reliability, 2012. 22(5): p. 297-312.

[18] Panichella, S., et al. How can i improve my app? classifying user reviews for software maintenance and evolution. in Software maintenance and evolution (ICSME), 2015 IEEE international conference on. 2015. IEEE.

[19] Rodríguez, P., et al., Continuous deployment of software intensive products and services: A systematic mapping study. Journal of Systems and Software, 2017. 123: p. 263-291.

[20] Silver, A., Collaborative software development made easy. Nature, 2017. 550(7674): p. 143-144.

[21] Rafi, D.M., et al. Benefits and limitations of automated software testing: Systematic literature review and practitioner survey. in Proceedings of the 7th International Workshop on Automation of Software Test. 2012. IEEE Press.

[22] Mir, F.A. and A.H. Pinnington, Exploring the value of project management: linking project management performance and project success. International journal of project management, 2014. 32(2): p. 202-217.

[23] Snyder, C.S. A Guide to the Project Management Body of Knowledge: PMBOK (®) Guide. 2014. Project Management Institute.

[24] Nguyen, C.D., et al., Evolutionary testing of autonomous software agents. Autonomous Agents and Multi-Agent Systems, 2012. 25(2): p. 260-283.

[25] Vogel-Heuser, B., et al., Evolution of software in automated production systems: Challenges and research directions. Journal of Systems and Software, 2015. 110: p. 54-84.

Ahmad AbdulQadir Al Rababah received Phd degree in 1998 in computer Engineering, now he is an associate professor at king Abdulaziz university(KSA), he has around 20 experience years of teaching and research in different fields of computing technology and engineering, his research interest areas are: information systems, software engineering, artificial intelligence and others.

Dr. Ahmad A Alzahrani has received a PhD in computer science from La Trobe University in 2014, he is currently an assistant professor in the faculty of Computing and Information Technology at King Abdulaziz University. His research interests include pervasive and mobile computing, Human-Computer Interaction and data science.