Waterative Model: an Integration of the Waterfall and Iterative Software Development Paradigms

Mohammad Samadi GHARAJEH
Young Researchers and Elite Club, Tabriz Branch,
Islamic Azad University, Tabriz, Iran
m.samadi@iaut.ac.ir; mhm.samadi@gmail.com

Software development paradigms help a software developer to select appropriate strategies to develop software projects. They include various methods, procedures, and tools to describe and define the software development life cycle (SDLC). The waterfall and iterative models are two useful development paradigms, which have been used by various software developers in the last decades. This paper proposes a new software development methodology, called waterative model, which applies an integration of the waterfall and iterative development paradigms. In this model, the iterative model is embedded into the waterfall model to use the advantages of both models as an integrated one. It, in the most cases, is appropriate for large software products that need a long-term period of time for the development process. Experimental results demonstrate that the customer satisfaction score could be high by using the proposed model in various software projects.
Keywords: *Software Engineering, Software Development, Waterfall Model, Iterative Model, Waterative Model*

1 Introduction

Software is formed by using a collection of executable programming codes, associated libraries, and essential documentations. Software product is developed for a specific requirement, which is composed of various phases such as system analysis, coding, and testing. Software engineering is, in fact, the development process of software products using well-defined methods, tools, and procedures. It includes some of the important challenges such as scalability, cost, and quality management that should be considered by software development teams, precisely. The development process of software products is not possible without considering such challenges [1-6].

A software developer should choose appropriate software development models in the development process. A new software development model, called waterative model, is proposed in this paper. This model uses an integration of the waterfall and iterative software development paradigms. Since the waterfall and iterative models have useful advantages in the software development process, their integration could be very useful compared to some of the existing models. This model, in the most cases, can be used to develop large software products.

The remainder of this paper is organized as the follows. Section 2 represents the elements of software development life cycle. Section 3 presents a literature review on some of the software development paradigms. Section 4 describes various steps of the proposed model. Section 5 includes the evaluation results of some software projects that have been developed by the proposed model. Finally, the paper is concluded by Section 6.

2 Software development life cycle

Software development life cycle (SDLC) [7] is a well-defined and structured sequence of various stages in software engineering to develop various types of software products. SDLC framework includes various steps that are entitled by Fig. 1. The remainder of this section describes a short description about each stage.

Communication is the first step of SDLC, which indicates the user's requirements for a

desired software product. Requirement Gathering leads the software development team to conduct the project's requirements. Requirements consist of user requirements, system requirements, and functional requirements. They can be conducted by studying the desired software, referring to the database, or collecting answers from questionnaires. Feasibility Study investigates whether a software project can be designed to fulfill all of the user's requirements or not. Moreover, it analyzes the financial, practical, and technical feasibilities of the project. System Analysis assists developers to determine a roadmap about their plan and also define an appropriate software model for the project. It considers some of the features such as product limitations, problems identification, and the effect of project on organization. Software Design brings down the user's requirements and any other knowledge about development process to design the software project. It is composed of two designs: logical design and physical design. Moreover, this step can be conducted by using various tools such as data dictionaries, logical diagrams, and use cases. Coding is also known as programming step. It implements the software product via writing the program codes by a suitable programming language and developing an error-free executable program. Testing determines the acceptance rate of a software product, which is done by testing team and, later, by the customer. It can be managed while writing the codes by the developers and at various levels of the coding step such as module testing, programming testing, and product testing. Integration step can be used to integrate the software product with libraries, databases, and other programs. Implementation means installing the software product on the user's machines. It can be used to test the portability and acceptability features of product. Maintenance determines the efficiency and error-free rate of software product. Furthermore, it aids to train the users by using some of the required documentations. Software product can timely be maintained by updating the program code based on the changes taking place in users, environment, and/or technology.

Fig. 1. An overall view of the SDLC framework

3 A literature review on software development paradigms

Waterfall model [8, 9] is the simplest development model from among a list of available software development paradigms. All steps of the SDLC framework are conducted one after another through a linear manner. That is, the second step will start only after the first step is finished and so on. This model considers that every process is perfectly conducted as planned in the previous step without any need to think about the past issues. Therefore, if there are some of the issues that are left from the previous step, this model will not work smoothly. This model is more appropriate for development teams when they have already designed and developed the same software in the past so they are aware of any development conditions. Fig. 2 shows an overall viewpoint of the model.

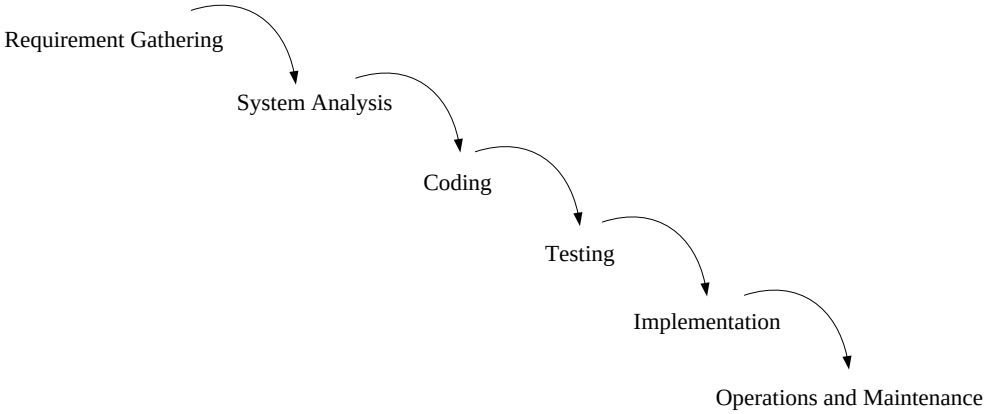

Fig. 2. Elements of the waterfall model

Iterative model [10] works based on iteration in the software development process. It conducts the development process via a cyclic manner so that every step is repeated after another. Firstly, software is developed on a small scale so that all of the steps are followed sequentially. Afterwards, more features and modules are designed, tested, and appended to the project on next iterations. In fact, the software at every iteration has more features and capabilities than the previous iterations. It is worth noting the management team can investigate risk managements of the project after completing each iteration in order to prepare the next iteration. Fig. 3 illustrates elements and processes of this model.

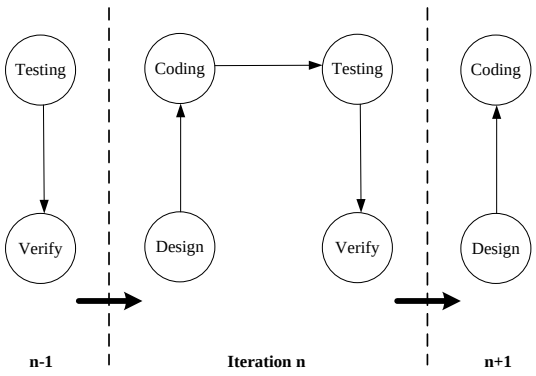

Fig. 3. A schematic of the iterative model

Spiral model [11-13] is a combination of iterative model and one of the SDLC models. In addition to any other process, it considers risk management that is often neglected by the most development models. In the first phase, it determines objectives and constraints of the software product at the beginning of iteration. In the next phase, the model conducts prototyping of the software and risk analysis. Afterwards, one of the SDLC models is applied to develop the software. The plan of next iteration is prepared in the final phase. Fig. 4 depicts a schematic of the spiral model.

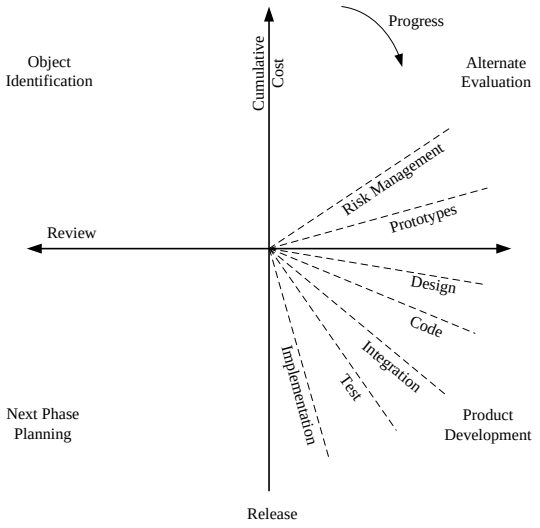

Fig. 4. An overall view of the spiral model

V-model [14, 15] facilitates the testing capabilities of software at each step via a reverse manner. It can solve the major drawback of waterfall model in which every step should be started only after the previous step is finished, without any chance to go

back. In this model, test plans and test cases are created at every step to verify and validate the software product based on the requirements of that step. This process leads both verification and validation to be conducted simultaneously. It is worth noting this model is also known as verification and validation model. Fig. 5 shows all elements and steps of the v-model.

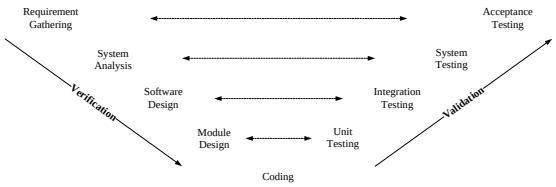

Fig. 5. An overall description of the v-model

Big bang model [16, 17] is the simplest model compared to the other development paradigms. It can be conducted by little planning, lots of programming, and lots of funds. This model is, in fact, similar to the process of universe in which lots of galaxies, planets, stars are created after an event. If the lots of programming and funds are put together, the best software product maybe will be achieved. A small amount of planning is required for this model. There is not any especial process to conduct the model as well as customers are not sure about the current and future requirements of product. In this model, the input requirements and conditions are arbitrary. It is worth noting this model is not appropriate for large software projects, but it is suitable for the learning and experimental purposes. Fig. 6 depicts a brief schematic of the big bang model.

Fig. 6. An overall schematic of the big bang model

4 Waterative model

Waterative model is formed based on an integration of the waterfall and iterative software development paradigms. It can be useful in software projects because it uses the advantages of both waterfall and iterative models. However, some of the new steps are involved in the proposed model as well as some of the steps are merged together. Since large software projects require a long-term development time, this model is suitable for such projects. Fig. 7 shows an overall view of the proposed development model. The most steps of this model have feedback to previous steps in order to solve the problems of software product at any step.

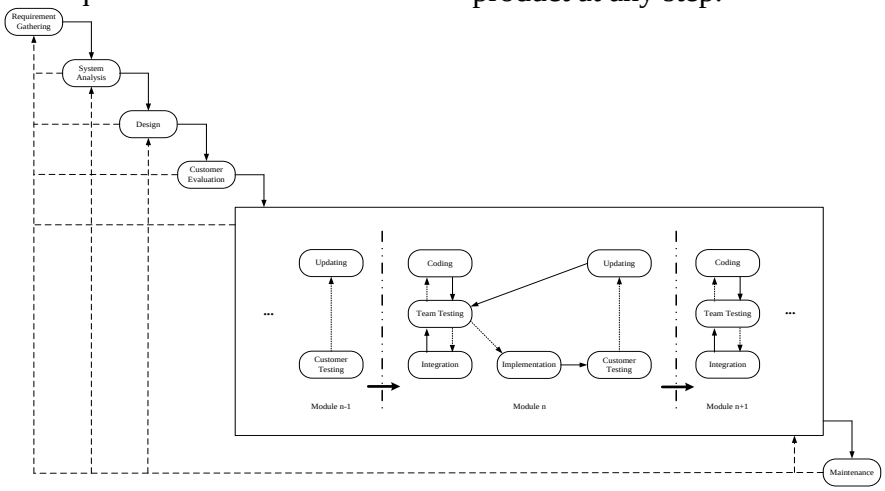

Fig. 7. Overall view of the proposed waterative model

Requirement Gathering includes both communication and requirement gathering of the SDLC framework. The service provider receives the request from the customer about developing a new software product. Afterwards, the development team manages the requirements according to user requirements, system requirements, and functional requirements. This step is a fundamental and essential step in this development model.

System Analysis involves both feasibility study and system analysis of the SDLC framework. The development team analyzes the feasibility and useful features of software product based on the user's requirements. The developers can use various algorithms to evaluate the financial, practical, and technical feasibilities of the product. Afterwards, the development team specifies a suitable roadmap to develop the software product according to requirements and feasibilities. It can consider development constraints, apply some of the learning systems, determine the scope, schedule, and resources of the project, and analyze other specifications of the project.

Design is, in fact, the software design step of SDLC. The software product is designed at this step based on the user's requirements and system analysis. The logical and physical designs of the product are also carried out in this step. They can be conducted by using some of the logical diagrams, data-flow diagrams, and pseudo codes. These designs should be organized carefully because the customer's initial evaluation and the team evaluation will be done by using meta-data and diagrams of this step.

Customer Evaluation is used is the proposed model before going on coding and the other development process. The reason is that the most problems of product at the system analysis and design steps should be solved by the customer's initial evaluation. This process leads further problems at the customer testing step of every module to be reduced considerably and, thereby, the most modules will be created and delivered to the customer during an acceptable period of time.

After the customer confirmed the correctness of development process until the design step, the software product will be divided into several modules to develop every module one after another, sequentially. Every module is composed of Coding, Team Testing, Integration, Implementation, Customer Testing, and Updating. All of the modules can be delivered to the customer sequentially to reduce fault rate of the whole system and decrease the project delivery time. After the code is programmed by programmers, all units of the code will be tested by tester team. Afterwards, the source codes are integrated with each other and the integration testing and system testing will be conducted by testers. Finally, the module will be implemented on the customer's machine to perform an acceptance testing by the customer. If the module has any problem according to the user's requirements then the module will be updated and tested by the team; otherwise, the next module will be created until the whole product will be delivered to the customer.

Maintenance is the final step in the proposed development model. In this step, the code could be updated based on any changes in the customer's requirements, technology, and platforms. It can also solve some of the challenges from the unpredicted problems and hidden bugs.

5 Evaluation results

Table 1 represents the experimental results related to some of my software projects in the last decade. In the most cases, I have attempted to use the proposed development model to accomplish various steps of these projects. However, this model has been progressed during a long-term period of time based on my experiences in developing various software projects. Customer

satisfaction score indicates that the customers have highly been satisfied by the development process.

Table 1. Experimental results

#	Project subject	Number of updates	Customer satisfaction score
1	Financial accounting software	7	85/100
2	Management software for civil engineering	34	95/100
3	Accounting software for shops	5	90/100
4	Thesis management software for universities	4	100/100
5	Educational scheduling software for educational institutes	6	100/100

6 Conclusions

A software developer should use a proper software development paradigm to develop software products from the requirements phase to the maintenance phase. All phases of the software development life cycle (SDLC) are defined and expressed by every development model. The developer can use various methods, tools, and procedures of any model to facilitate the development process of software products.

This paper proposed a new software development methodology, called waterative model. It uses an integration of the waterfall and iterative software development paradigms, which uses all phases of the SDLC framework through an efficient process. The iterative model is embedded into the waterfall model to use advantages of both models. In the most cases, large software projects can be developed by this model via a long-term development process. Evaluation results indicated that the customer satisfaction score was enhanced by using the waterative development model in various software projects.

I have used the proposed model in various software projects in the last 15 years. My experiences have demonstrated that this model could be useful in the most projects, especially large software products. Therefore, I decided to present this model as a new development platform to could be used by other software development teams.

References

[1] R. Mall, Fundamentals of software engineering. Delhi: PHI Learning Pvt. Ltd., 2014.

[2] N. Fenton and J. Bieman, Software metrics: a rigorous and practical ap-proach. Boca Raton, Florida: CRC press, 2014.

[3] F. Tsui, O. Karam, and B. Bernal, Essentials of software engineering. Burlington, Massachusetts: Jones & Bartlett Learning, 2016.

[4] E.J. Braude and M.E. Bernstein, Software engineering: modern ap-proaches. Long Grove, Illinois: Waveland Press, 2016.

[5] B. Fitzgerald and K. -J. Stol, "Continuous software engineering: A road-map and agenda," Journal of Systems and Software, vol. 123, pp. 176-189, 2017.

[6] M. Solari, S. Vegas, and N. Juristo, "Content and structure of laboratory packages for software engineering experiments," Information and Soft-ware Technology, vol. 97, pp. 64-79, 2018.

[7] D.J. Mayhew, "The usability engineering lifecycle," CHI'99 Extended Ab-stracts on Human Factors in Computing Systems, New York City, New York: ACM, pp. 147-148, 1999.

[8] Y. Bassil, "A simulation model for the waterfall software development life cycle," The International Journal of Engineering & Technology, vol. 2, no. 5, pp. 742-749, 2012.

[9] S. Madgunda, U. Suman, G.S. Praneeth, and R. Kasera, "Steps in re-quirement stage of waterfall model," International

journal of computer & mathematical sciences, vol. 4, no. 7, pp. 86-87, 2015.

[10] S.C. Ahluwalia, D.B. Bekelman, A.K. Huynh, T.J. Prendergast, S. Shreve, and K.A. Lorenz, "Barriers and strategies to an iterative model of advance care planning communication," American Journal of Hospice and Pallia-tive Medicine, vol. 32, no. 8, pp. 817-823, 2015.

[11] B.W. Boehm, "A spiral model of software development and enhancement," Computer, vol. 21, no. 5, pp. 61-72, 1988.

[12] B. Boehm, J. Lane, S. Koolmanojwong, and R. Turner, The Incremental Commitment Spiral Model. Boston, Massachusetts: Addison Wesley, 2014.

[13] B. Boehm, J.A. Lane, S. Koolmanojwong, and R. Turner, The incremental commitment spiral model: Principles and practices for successful systems and software. Boston, Massachusetts: Addison-Wesley, 2014.

[14] S. Mathur and S. Malik, "Advancements in the V-Model," International Journal of Computer Applications, vol. 1, no. 12, pp. 29-34, 2010.

[15] G. Whyte and A. Bytheway, "The V-model of service quality: an African case study," European Journal of Marketing, vol. 51, no. 5/6, pp. 923-945, 2017.

[16] A. Finkelstein and J. Kramer, Software engineering: a roadmap. New York City, New York: ACM Press, 2000.

[17] J. Ludewig, "Models in software engineering–an introduction," Software and Systems Modeling, vol. 2, no. 1, pp. 5-14, 2003.

Mohammad Samadi Gharajeh received ASc in Computer Software on 18 February 2005, BSc in Engineering of Computer Software Technology on 18 February 2009, and MSc in Computer Engineering – Computer Systems Architecture on 3 February 2013. He has already developed various software programs, simulations, intelligent systems, and research projects. His research interests include software engineering, artificial intelligence, soft computing, intelligent control, and embedded systems. He was a Technical Program Committee member and a Reviewer in some of the international conference proceedings. Besides, he is an Editorial Board member and a Reviewer in some of the international scientific journals, a Lecturer of university, and an IEEE Member now.