International Journal of Electrical and Computer Engineering (IJECE)
Vol. 12, No. 1, February 2022, pp. 534~540
ISSN: 2088-8708, DOI: 10.11591/ijece.v12i1.pp534-540

Formalization of the prediction and ranking of software development life cycle models

Laiali Almazaydeh[1], Moath Alsafasfeh[2], Reyad Alsalameen[1], Shoroq Alsharari[1]
[1]Faculty of Information Technology, Al-Hussein Bin Talal University, Ma'an, Jordan
[2]Faculty of Engineering, Al-Hussein Bin Talal University, Ma'an, Jordan

Article Info

Article history:

Received Apr 27, 2021
Revised Jun 26, 2021
Accepted Jul 13, 2021

Keywords:

Agile
Incremental
ITerative
Scrum
SDLC
Spiral
Waterfall

ABSTRACT

The study of software engineering professional practices includes the use of the formal methodology in a software development. Identifying the appropriate methodology will not only reduce the failure of software but will also help to deliver the software in accordance with the predetermined budget and schedule. In literature, few works have been developed a tool for prediction of the most appropriate methodology for the specific software project. In this paper, a method for selecting an appropriate software development life cycle (SDLC) model based on a ranking manner from the highest to the lowest scoring is presented. The selection and ranking of appropriate SDLC elaborate the related SDLC's critical factors, these factors are given different weights according to the SDLC, then these weights are used by the proposed mathematical method. The proposed approach has been extensively experimented on a dataset by software practitioners who are working in the software industry. Experimental results show that, the proposed method represents an applicable tool in predicting and ranking suitable SDLC models on various types of projects, such as: life-critical systems, commercial uses systems, and entertainment applications.

Corresponding Author:

Laiali Almazaydeh
Faculty of Information Technology, Al-Hussein Bin Talal University
Ma'an, Jordan
Email: laiali.almazaydeh@ahu.edu.jo

1. INTRODUCTION

Software engineering aims to use the best practices for building quality software systems. Nowadays, software systems can be seen in a variety of applications, the different types of applications you encounter may be in the business domain, in the engineering domain or maybe in scientific applications. In fact, for any software solutions which are of long duration, it is very significant that you control and review its development progress very systematically. The engineering approach basically means that, a well-defined systematic approach must be applied to software development in order to have a very high probability of success [1], [2]. As a well-defined systematic approach is the main benefit of the engineering approach, software development life cycle (SDLC) is an essential process for the development of software, which consists of defining a sequence of different activities and phases. They are requirements analysis, design, coding, testing, and maintenance [3].

In literature, different SDLCs-based software engineering models can be used. Firstly, predictive models, also called traditional or plan driven models are one of the most common classic models. This approach depends on the predictable experience that utilizes many steps organized in a linear order and these steps will be highly controlled. In addition, this approach considered very documentation-heavy, it means many documentations in a standard format and in contractual obligations is being produced as a baseline for

future reference. Therefore, the development team strives to adhere to the approved plan in terms of scope, timeline, and scope by undertaking risk planning and management throughout the project lifecycle [4]-[6]. One of the most common predictive life cycles is the waterfall model, which requires defining and documenting a stable set of requirements completely at the beginning of a project.

Secondly, iterative and incremental models, where requirements can be repeated and changed multiple times leading to different iterations and increments that are developed either at a phase wise or at a cycle wise. Within the iterative life cycle, a throwaway prototype built from currently known customer needs, then the prototype is tuned based on the customer's feedback to incorporate changes and new requirements [7], [8]. Within the incremental life cycle, although the broad concept is normally agreed up front, the software is developed as a series of a mini waterfall cycle, it released increments then combined them in all increments to produce the final software. The spiral model is a risk driven model that combines the features of the prototyping (iterative) model and the waterfall model [8], [9]. Through the different cycles of the spiral model, different risks will be addresses, for user interface risks, a prototype will take a place, and for development risks, a waterfall will be used.

Thirdly, agile models, also known as the change driven life cycle. It considered as a rapid incremental iterative model, where the software project's scope emerges as the project is being executed. Hence, the approach here is to develop minimum viability of the software product called minimal viable product (MVP) in the first iteration, then the subsequent iterations can add further product features and functionality. One of the widely used framework to implement agile is Scrum, within Scrum various process, techniques and methods are used to continuously improve the product, the teams, and the working environment [10], [11].

Apparently, there are many SDLCs in literature that are utilized by software engineers, but there are no such key criteria that define appropriate SDLC selection. However, the practice of SDLC selection is related to the knowledge and skills of software engineers taking into their account different related project factors as application domain and technological needs [12], [13]. In this paper, we develop such key criteria based on the identification of related SDLC's critical factors, these factors are given different weights according to the SDLC, then the selection and ranking of suitable SDLC will be based on the proposed mathematical method.

This paper is organized as follows: Section 2 presents the background of existing SDLC selection methodologies. Section 3 is the development of the mathematical method using the relationship between various software methodologies and related software factors. Section 4 discusses about the tool's assesment and findings. Section 5 summarizes how the research objectives being achieved, their contribution and future works.

2. RELATED WORKS

Over the past few years, a limited number of related works have been developed as a tool for the prediction of the most proper SDLC for the specific software project. One of these some key related studies in [13] where the rule-based technique was applied to determine whether the suitable SDLC falls under agile or non-agile categories. The proposed technique used mainly Project size, requirement stability and complexity as software factors to formulate the relationship between these factors and software methodologies based on a set of rules and predetermined questions. However, additional works may involve enhancing the system in the area of rules management and refinement and its representation.

A computer-aided decision support system for SDLC selection was developed in [14]. In this system, a list of criteria was created concerning two groups: project and product. The criteria concerning project group, include planned schedule, cost, resources, risk, and level of users' skills in IT. The criteria concerning product group, include the type of information system, complexity of the software, modularity, the clarity and stability of user requirements, and system architecture. Then, matching between the rules of model selection which are available in the conditions table and the parameters of project and product entered by user will be employed. However, the result of applying this algorithm cannot be considered completely objective, because it depends highly on the weights assigned to the individual criteria, as well as the questions. All these values and answers are subjective as it is given according to decision maker preferences.

Fuzzy logic (FL) system [15] was developed to select an appropriate SDLC. In this system, a set of criteria including schedule, complexity, risks, size of the project, in addition to clarity of requirements and experience of team members were assigned to FL inputs. A set of applied fuzzy rules are a "conditional statement" of FL inputs connected with "and operator" which led to exactly matched one of SDLCs. However, in this work, the fuzzy rules are just rigid rules that require matching of a combination of a set of assigned criteria to the specific SDLC.

On the other hand, regarding other software project issues, further approaches based on FL have been employed to select the appropriate team among three categories: Low, average, and highly experienced

teams, where each team consists of members who are software analyst, designer, coder, tester, and manager [16]. Other approach used FL to predict usability of software product, as the seven factors of usability and their attributes were used to show the SDLCs ranks [17]. Based on the mentioned related works in this field, our contribution in this work is developing a new approach that is used to predict and rank suitable SDLC models, in addition to identify the critical factors that affect the choice of SDLC models.

3. RESEARCH METHOD

In this proposed methodology, the main focus is to develop an applicable tool to accomplish precise selection of SDLC. The proposed methodology as illustrated as shown in Figure 1 and Algorithm 1 mainly is composed of these main phases. Firstly, we have studied a large number of SDLC models, in order to get the most used models. Secondly, we have identifying a set of criteria that distinguish each model from the other one, the outcome of a list of models together with the ascribed criteria is shown in Table 1. Thirdly, we have assigned weights to the individual criteria, the weights values could be 0.5, 1 or 2 according to the significance of the criteria as shown in Table 1, a dark gray-filled cell represents a weight of 2, a light gray-filled cell represents a weight of 1, and a cell with no color represents a weight of 0.5. Lastly, the decision maker only needs to select the appropriate cell for each individual criterion, then, the weight of the most selected models will be the cumulative weight of each model in the selection divided by total weight.

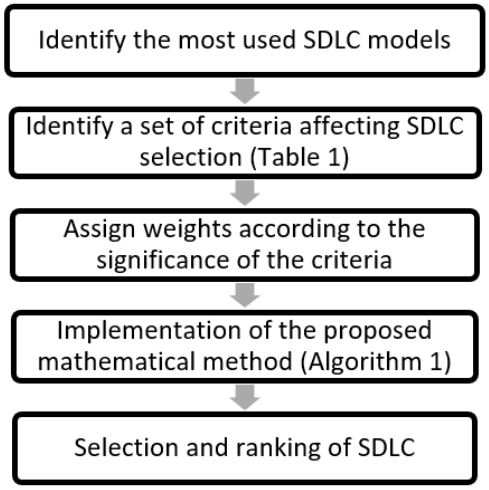

Figure. 1 Proposed framework

To illustrate the methodology by an example, a website development project might consider as a real-world example, however, different scenarios can be discussed here for SDLC selection. When all the requirements are well-defined at the beginning of website development we can follow waterfall to deliver a complete functional web-site at the end, when the requirements are unknown at the beginning we can take the iterative approach, it starts with creating the website with a very basic framework, then refining the website in subsequent iterations to deliver the final product, iterative approach is all about refining the product through iterations until all customer requirements are met. Also, we might deliver a functional website with few important features that the customer can start using, this is called the first increment, then, adding the new features in the form of increments it continues until the final product is delivered, this way the final product is delivered through multiple increments.

On the other hand, the most preferred approach to deal with complexity is the agile approach, here we deliver the smaller increments as well as refine the website through iteration, as the customer's feedback and change in requirements are adapted through multiple iterative and incremental deliveries iterations and increments continue until the final product is delivered, due to this multiple delivery models the project team can adapt to the changing requirements, hence the agile approach is the most preferred approach for the complex adaptive problems. Furthermore, if there is a unique risk pattern on the website development project, the spiral model helps to ensure an efficient development process. Further, if there a need for team's work, the scrum model provides an empirical basis for teams to deliver iterations more frequently with higher possible value and better outcomes.

Therefore, for addressing such complex adaptive problems in the most productive and creative, we need a systematic and lightweight framework. From Table 1, let's assume that the decision maker input the following criteria as a visualization of a certain software project:

a. Iterative workflow (T1)
b. Requirements are not clearly defined at the beginning (T2)
c. Evolving changes (T3)
d. Moderate involvement of user in all stages (I4)
e. No overlapping phases (I5)
f. No risk analysis (T6)
g. Moderate documentation (I7)
h. Above estimated cost (A8)
i. Availability of early prototype (T9)
j. Rapid development objective (A10)

Table. 1 The most used SDLC models and critical criteria

Criteria\Model	**W**aterfall	**I**ncremental	**IT**erative	**A**gile	**S**piral	**Sc**rum
Sequential workflow	Sequential [18]	Multi sequential [18]	Iterative [18]	Incremental + iterative [18]	Incremental + iterative [18]	Incremental + iterative [18]
Well defined requirements	Requirements are defined up-front before development begins [19]	Requirements of the system are clearly understood [20]	Requirements are not clearly defined [20]	Requirements are poorly defined [21], [22]	Requirements are not easily defined [15]	Requirements are not easily defined [23]
Accommodate changes	Requirements are stable or unchanging [23] [24]	Functional requirements may change frequently [19]	Functional requirements may change frequently and significantly [23]	Requirements are changing and largely emergent [20], [21]	Ability to incorporate changes at any stage [19]	Frequently changed [19]
User involvement in all stages	Very less/only at beginning [25]	Moderate [18]	High/After each iteration [23]	High/After each iteration [23]	Involving customers more frequently [20], [18]	High [23]
Sdlc phases overlapping	No Overlapping [20]	No [20]	Yes [20]	Yes [20]	Yes [20]	Yes [20]
Risk analysis	Only at beginning [20]	No risk analysis [20]	No risk analysis [20]	Yes [20]	Yes [20]	Yes [20]
Documentation	Vital [20]	Moderate [9]	Less [20]	Moderate [18]	Less [18]	Required, but Limited [23], Less [18]
Project cost estimation	Almost As estimated cost [23], [25]	Above estimated cost [23], [25]	Above estimated cost [23], [25]	Above estimated cost [23], [25]	Very costly [6] [25]	Almost as Estimated Cost [23], [25]
Availability of working software-early prototype	At the end of the life cycle [25]	At the end of every increment [25]	At the end of every iteration [25]	At the end of every iteration [25]	At the end of every iteration [25]	At the end of every iteration [25]
Primary objective	High assurance [23]	Rapid development [23]	Rapid development [23]	Rapid development [23]	High assurance [23]	Rapid development [23]

Where T1 stands for I**T**erative model represented by T string with "iterative criteria" as number 1 in the set of criteria that shown in Table 1, I4 stands for **I**ncremental model represented by I string with "requirements of the system are clearly understood" as number 4 in the set of criteria, A8 stands for **A**gile model represented by A string with "above estimated cost" as number 8 in the set of criteria, and so on and so forth for many other criteria.

Therefore, the weights of the above selected criteria of certain software project will be computed by adding the weights of all selections as shown in (1).

$$Total\ of\ selected\ models\ weight = 0.5\,T1 + 0.5\,T2 + 2\,T3 + 0.5\,I4 + 0.5\,I5 + 0.5\,T6 + 2\,I7 + 1\,A8 + 2 + T9 + 0.5\,A10 \quad (1)$$

Undoubtedly, the probability of an event tells how likely the event will happen. Each model weight mentioned in (1) can be computed by finding the probability using (2), the total of available model weights out of all the selected models' weights.

$$\begin{aligned} \text{T weight} &= (0.5\,\text{T1} + 0.5\,\text{T2} + 2\,\text{T3} + 0.5\,\text{T6} + 2\,\text{T9}) / 10 = 0.55 \\ I\,weight &= (0.5\,I4 + 0.5\,I5 + 2\,I7) / 10 = 0.3 \\ A\,weight &= (1\,A8 + 0.5\,A10) / 10 = 0.15 \end{aligned} \quad (2)$$

From the above given specific selected criteria, we conclude that the best SDLC ranking are Iterative model, Incremental model, then Agile model according to their ascribed criteria values.

Algorithm 1

Set the selection matrix by $M \times N$
a. M: number of project specification
b. N: number of SDLC models
c. Cell of $M \times N$ is the level of M selection

For each M
Set which cell in each row has the highest, medium or lowest weight
While the user selects a cell from each M
Selection $weight = cell\,weight \times model\,string$
Break if number of $selection = M$
Get the total weight by adding the weights of all selections
Calculate the W for each model by:
Weight of each model=sum of this model in the selection/total weight

4. RESULTS

In order to validate the proposed methodology, a dataset of different software project domains is assumed and employed such as: life-critical systems, commercial uses systems, and entertainment applications. The evaluation process involves thirty participants who are working in the software industry either as software developers or system analysts with at least three years of working experience in software development. The participants were asked to validate the proposed approach themselves using the dataset. In order to get dependable feedback, the same dataset was distributed across the participants and the given results were compared. Accordingly, the results of SDLC selection and prediction among them are nearly identical and the difference is negligible. The result of implementing this methodology can be considered generic and objective, for it is highly reliant on the weights ascribed to the individual SDLC criteria from literature, and the SDLC selection should be error-prone only because of erroneous settings from decision-maker. Hence, the proposed methodology generated the more accurate selection and ranking of SDLC as compared to other rarely existing SDLC selection models.

5. CONCLUSION

The proposed approach represents a beneficial tool in predicting and ranking suitable SDLC models. In addition, the approach identifies the critical factors affecting the choice of SDLC models. The SDLC models and software factors have been rigorously analyzed in order to meet the needs of developing a tool that it is suitable to all level of software professionals. Further works which outline distinct phases of SDLC from requirement analysis to maintenance may involve in the future to develop predictable tools. Such in progress valuable work is a visualization of user requirements through automatically generated diagrams to create a suitable software design as per customer expectations.

REFERENCES

[1] A. Ahmed, and B. Prasad, *Foundations of Software Engineering*, 1st Ed., Auerbach Publications, 2016.
[2] I. Sommerville, *Software Engineering*, 10th Ed., Pearson, 2015.
[3] R. Ibrahim, and S. Y. Yen, "Formalization of the data flow diagram rules for consistency check," *International Journal of Software Engineering and Applications*, vol. 1, no. 4, pp. 95-111, Oct. 2010, doi: 10.5121/ijsea.2010.1406.
[4] A. I. Khan, M. R. J. Qureshi and U. A. Khan, "A comprehensive study of commonly practiced heavy and light weight software methodologies," *International Journal of Computer Science Issues (IJCSI)*, vol. 8, no. 2, pp. 441-450, 2011, *arXiv:1202.2514.*
[5] Y. Yang, X. Xia, D. Lo, T. Bi, J. Grundy, and X. Yang, "Predictive models in software engineering: challenges and opportunities," *Software Engineering*, vol. 1, no. 1, pp. 1-35, 2016, *arXiv:2008.03656.*
[6] T. Menzies and G. Koru, "Predictive models in software engineering," *Empirical Software Engineering*, vol. 18, no. 3, pp. 433-434, 2013, doi: 10.1007/s10664-013-9252-1.

[7] J. S. Shirabad, "Predictive techniques in software engineering," *Encyclopedia of Machine Learning*, 2010, doi: 10.1007/978-0-387-30164-8_661.
[8] A. Alshamrani and A. Bahattab, "A comparison between three SDLC models waterfall model, spiral model, and Incremental/Iterative model," *International Journal of Computer Science Issues (IJCSI)*, vol. 12, no. 1, pp. 106-111, 2015.
[9] B. Boehm and W. J. Hansen, "Spiral development: experience, principles, and refinements," *Special Report, CMU/SEI-2000-SR-008*, pp. 1-49, 2000.
[10] Y. Khmelevsky, X. Li, and S. Madnick, "Software development using agile and scrum in distributed teams," *2017 Annual IEEE International Systems Conference (SysCon)*, 2017, pp. 1-4, doi: 10.1109/SYSCON.2017.7934766.
[11] F. Hayat, A. U. Rehman, K. S. Arif, K. Wahab, and M. Abbas, "The influence of agile methodology (scrum) on software project management," *2019 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, 2019, pp. 145-149, doi: 10.1109/SNPD.2019.8935813.
[12] A. B. Nassif and L. F. Capretz, "Towards an early software estimation using log-linear regression and a multilayer perceptron model," *Journal of Systems and Software*, vol. 86, no. 1, pp. 144-160, 2013, doi: 10.1016/j.jss.2012.07.050.
[13] M. Hanafiah and Z. Kasirun, "Using rule-based technique in developing the tool for finding suitable software methodology," *Malaysian Journal of Computer Science*, vol. 20, no. 2, pp. 209-224, 2007, doi: 10.22452/mjcs.vol20no2.8.
[14] G. H. Janczura and I. Golinska, "Decision support system for choosing a model for a software development life cycle," *Operations Research and Decisions*, vol. 1, pp. 61-77, 2010.
[15] V. Ozturk, "Selection of appropriate software development life cycle using fuzzy logic," *Journal of Intelligent and Fuzzy Systems*, vol. 25, pp. 797-810, 2013, doi: 10.3233/IFS-120686.
[16] R. Raj, R. Mittal, S. C. Gupta and T. Choudhury, "Proposal of software development model using fuzzy logic," *2018 Second International Conference on Green Computing and Internet of Things (ICGCIoT)*, 2018, pp. 654-658, doi: 10.1109/ICGCIoT.2018.8753049.
[17] D. Gupta, A. Ahlawat, and K. Sagar, "Usability prediction & ranking of SDLC models using fuzzy hierarchical usability model," *De Gruyter*, vol. 7, no. 1, pp. 161-168, 2017, doi: 10.1515/eng-2017-0021.
[18] U. Shah, D. Jinwala, and S. J. Patel, "An excursion to software development life cycle models: an old to ever-growing models," *ACM Transactions on Software Engineering and Methodology*, vol. 41, no. 1, pp. 1-6, 2016, doi: 10.1145/2853073.2853080.
[19] S. Ateeq and M. Shuaib, "Comparison of various SDLC models," *Global Journal of Multidisciplinary Studies*, vol. 3, no. 11, pp. 176-181, 2014.
[20] R. G. Sabale and A. R. Dani, "Comparative study of prototype model for software engineering with system development life cycle," *IOSR Journal of Engineering (IOSRJEN)*, vol. 2, no. 7, pp. 21-24, 2012, doi: 10.9790/3021-02722124.
[21] M. L. Todd, "Selecting a development approach for competitive advantage," *Bellevue University*, 2010.
[22] B. Boehm, "Get ready for agile methods, with care," in *Computer*, vol. 35, no. 1, pp. 64-69, Jan. 2002, doi: 10.1109/2.976920.
[23] A. Mansoori and M. Tayab, "Standardized software development model for SME software houses in Pakistan," *Journal of Independent Studies and Research - Computing*, vol. 12, no. 1, pp. 11-17, 2014.
[24] A. Deeb, "Software development methodologies in industry," M.S. thesis, Departement Software Engineering, Australian National University, 2010.
[25] D. Radhika, "Analysis and comparative study of various software development process models," Ph.D. dissertation, Department of Computer Science, Saurashtra University, 2013.

BIOGRAPHIES OF AUTHORS

Laiali Almazaydeh received her doctorate in Computer Science and Engineering from University of Bridgeport in USA in 2013, specializing in human computer interaction. She is currently an associate professor and the dean of Faculty of Information Technology, Al-Hussein Bin Talal University, Jordan. Laiali has published more than 45 research papers in various international journals and conferences proceedings, her research interests include human computer interaction, pattern recognition, and computer security. She received best paper awards in 3 conferences, ASEE2012, ASEE2013 and ICUMT 2016. Recently she has been awarded two postdoc scholarships from European Union Commission and Jordanian-American Fulbright Commission. She can be contacted at email: laiali.almazaydeh@ahu.edu.jo.

Moath Alsafasfeh earned his Ph.D. in 2017 at Western Michigan Univesrsity (WMU) with excellent evaluation. He is currently an assistant professor of Electrical and Computer Engineering at Al-Hussein Bin Talal University (AHU), Jordan. The current research of Alsafasfeh is going in using computer vision and machine learning to operate, monitor, maintain, and control the different systems in an efficient processing time. Dr. Alsafasfeh awarded several national prizes, scholarships to get Ph.D., and bachelor's degrees. He has strong relationships with different local, regional, and international researchers and he is involved in many different international research and capacity building projects. He can be contacted at email: moath.al-safasfeh@ahu.edu.jo.

Reyad Alsalameen obtained his Ph.D. in Software Engineering from University of Salford, UK in 2016. He is currently an assistant professor and the head of software engineering department at Al-Hussein Bin Talal University (AHU), Jordan. His current research interest include fault-tolerant systems and software operation and maintenance. He can be contacted at email: Reyad.m.salameen@ahu.edu.jo.

Shoroq Alsharari obtained her master degree in computer science from Middle East University, Jordan in 2013. She is currently a lecturer of software engineering at Al-Hussein Bin Talal University (AHU), Jordan. Her current research interest include image processing and software design and implementation. She can be contacted at email: shoroq.al-sharari@ahu.edu.jo