Global Journal of Engineering and Technology Advances
eISSN: 2582-5003
Cross Ref DOI: 10.30574/gjeta
Journal homepage: https://gjeta.com/

(RESEARCH ARTICLE)

Check for updates

Analytical study of software development process model variants

Monday Eze [1,*] and Charles Okunbor [2]

[1] *Department of Computer Science, Babcock University, Nigeria.*
[2] *Department of Computer Science, Admiralty University, Delta State, Nigeria.*

Global Journal of Engineering and Technology Advances, 2021, 08(02), 023–031

Publication history: Received on 23 June 2021; revised on 02 August 2021; accepted on 05 August 2021

Article DOI: https://doi.org/10.30574/gjeta.2021.8.2.0111

Abstract

Software Engineering is a branch of Computer Science that evolved as a result of urgent need to deal with decades of software crisis, characterized by low theoretical knowledge and practice of the construction of error-free and efficient software. The introduction of well-organized scientific, engineering and management strategies in the process of software development no doubt led to major breakthroughs, and solutions to software failures. One of the obvious game-changer in this regard is the evolution of Software Development Life Cycle, also known as Software Process Model for driving the different phases of software construction. A sound understanding of the process model is therefore inevitable, not just for software developers, but also to users and researchers. Such a theoretical cum practical understanding will enhance decisions on which process model is best for a particular job or perspective. This invariably, contributes immensely to the probability of success or failure of the project in question. Thus, the necessity for this research. This work presents an unambiguous expository of selected software development model variants. A total of four process model variants were studied, in a theoretical, visual and analytical manner. The variants were analyzed using strength versus weakness (SVW) tabular scenario. This work was concluded by presenting guides towards choice of these models. This research is expected to be a useful reference to software practitioners and researchers.

Keywords: Software Development; Process Models; Waterfall; V-Model; Incremental; Spiral Model

1. Introduction

The field of Software Engineering is a branch of Computer Science, saddled with the analysis, design, testing, implementation, and maintenance of efficient software system [1]. While the successful construction and deployment of an efficient software product is no doubt a milestone [2] in the software development journey, it is however not the climax, rather it is the beginning of the herculean task of software maintenance [3]. The importance of understanding and operationalizing the chronology of stages of software development phases in real life cannot be overemphasized, thus the reason for this research. The evolution of Software Engineering as engineering discipline stemmed from what appears to be a rescue mission and a response to widespread software crises [4] of the past decades. That critical moment was characterized by uncountable number of software project failures [5], software overbudgeting [6], lack of software maintenance know-how [7], low end-user satisfaction, among others. The evolution of Software Engineering however established scientific, quantitative, engineering and management perspectives to the evolution of software products, through the introduction and practice of Software Development Life Cycles (SDLC). The SDLC [8] also known as software development process models is a chronology of steps used by software practitioners to build software from conception to completion. The aim is to successfully design, develop and test high quality and cost-effective software that meets or exceeds customer expectations and completed within deadline.

* Corresponding author: Monday Eze
Department of Computer Science, Babcock University, Nigeria.

Software Development Life Cycle (SDLC) is a very organized process for building quality and error-free software. If follows that the choice of software development model in use will no doubt affect what becomes of the resultant product. This research presents a study of four major software development process models – water fall, incremental, spiral and V-models respectively. Effort is made in this study to present requisite theoretical foundations [9] of the subject matter. This was achieved using diagrammatic and analytical contents. This work presents a strength versus weakness (SVW) tabular appraisal of each of the models studied, as well as determinant factors to the choice of process models. First and foremost, a brief exploration of the process model variants will be presented here.

2. Process model enumerations

The focus of this expository study is on four process models as shown in Fig. 1. Effort will be made to perform a critical and analytical study of each of these models, so as to ensure that practitioners [10] are able to make informed decision on which choice to model to use in order to tackle software projects in real life.

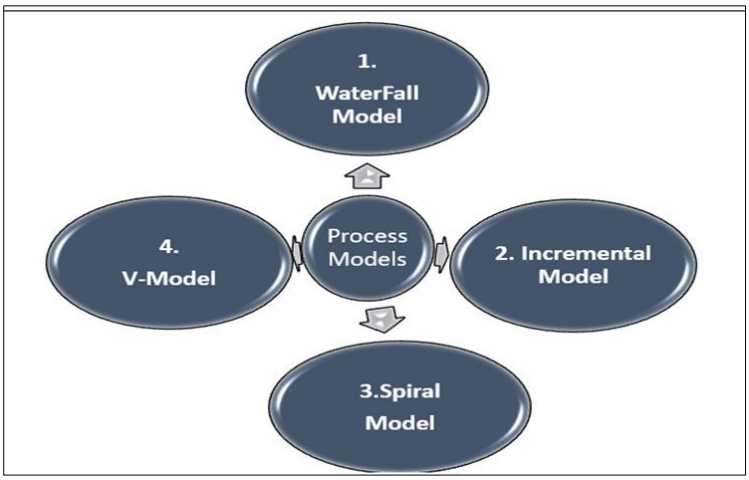

Figure 1 Enumeration of Software Process Models

3. Water fall model

The waterfall software process model is adjudged one of the oldest of all the development models. Thus, a number of researches usually refer to it as the traditional model of software development [11].

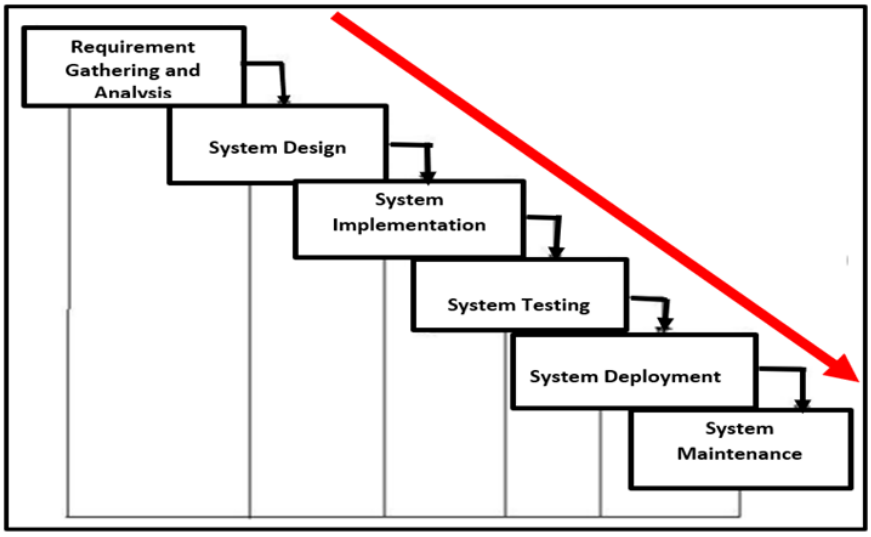

Figure 2 Diagram of Waterfall Model

The name is derived from the shape of a natural water fall in a typical electricity generating dam [12], where water flows in a step-by-step fashion down from the topmost water table. The Waterfall Model follows a serial execution of the SDLC phases from inception to completion. One of the key attributes of waterfall model that distinguishes it from the rest of the others is that every component phase must be given adequate attention, and fully completed before proceeding to the next phase [13]. The visual representation of the Waterfall model is shown in Fig. 2.

As shown in the diagram, a typical waterfall model consists of a number of developmental phases or stages such as requirement gathering and analysis, system design, system implementation, system testing, system deployment and system maintenance. Thus, in waterfall model, a particular phase must be fully competed, before the next one.

4. Incremental model

The easiest way to understand the incremental model is to look at it from the angle of a mathematical representation [14]. A simple quantitative representation of the incremental model is shown in equation 1.

$$IncMOD = WaterFall_1 + WaterFall_2 + \cdots WaterFall_N \ (1)$$

where:

IncMOD = the Incremental Process Model, Water Fall$_x$ is a series of waterfall models, where x is an integer from 1 to N, and N is the total number of times, a waterfall is repeated in the course of system development.

It follows that the incremental model is a batch or multiple of N number of waterfalls, where N is an integer. The earlier equation can thus be rewritten as equation 2.

$$IncMOD = N\ (WaterFall) \ (2)$$

Fig. 3 shows an incremental model of multiple value 3. The implication is that there is usually an improvement in the value of the software after each of the increments.

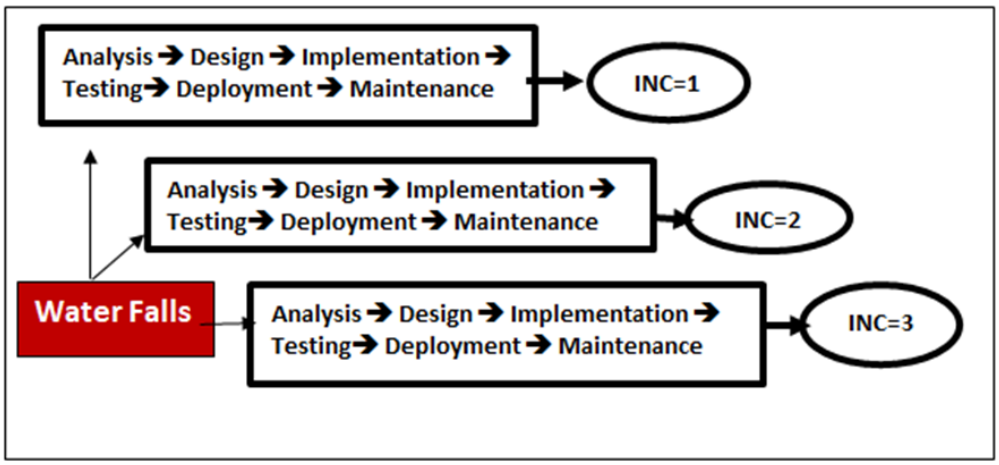

Figure 3 Incremental Model Diagram

As shown in the diagram, there are a series of increments (INCs) emanating from the execution of series of water falls. In this case, there were three increments or improvements.

5. Spiral model

The spiral model [15] is a risk-focused software development strategy. In other words, the strong point of this model is that ample time, energy and resources are invested to ensure that software related risks [16] are trapped, and handled within the developmental process. Apart from the fact that spiral model lays a very strong emphasis on risk handling, it consists of series of spirals with multiple loops. A diagrammatical illustration of the model is shown in Fig. 4. In spiral model, each phase consists of four unique quadrants [17], designated as QUAD1, QUAD2, QUAD3 and QUAD4, each of

which have definite functionalities. For instance, QUAD1 determine objectives, QUAD2 identifies and resolves risks, QUAD3 is dedicated to development and test, while QUAD 4 involves planning the next iteration.

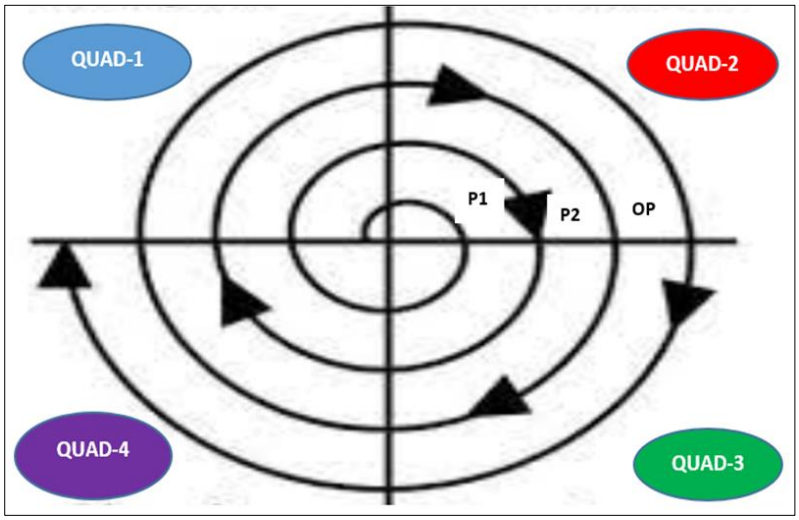

Figure 4 Spiral Model Diagram

There are a number of important theoretical and conceptual facts of key importance in the spiral model, some of which are as follows. First, the exact number of loops of the spiral is unknown, and varies from project to project [18]. Secondly, each loop of the spiral is called a Phase of the software development process [19]. Thirdly, a series of spirals give rise to prototypes, for instance P1, P2, as shown in the diagram, while the final one OP represents the operational prototype [20]. In spiral model, the exact number of phases needed to develop a product is varied by the project manager, and depends on the project risks. Furthermore, the Radius of the spiral [21] at any point represents the expenses (cost) of the project so far, while the angular dimension [22] represents the progress made so far in the current phase.

6. V-model model

The V-model is a SDLC model, in which the component processes are executed in a sequential V-shape manner as shown in Fig. 5. The V-Model is commonly known as the Verification and Validation model [23]. As can be clearly deduced from the diagram, V-Model contains Verification related phases on the left-hand side (LHS) and Validation related phases on the right-hand side (RHS), both of which are joined at the coding phase in an apparent V-shape. This is why it is known as V-Model.

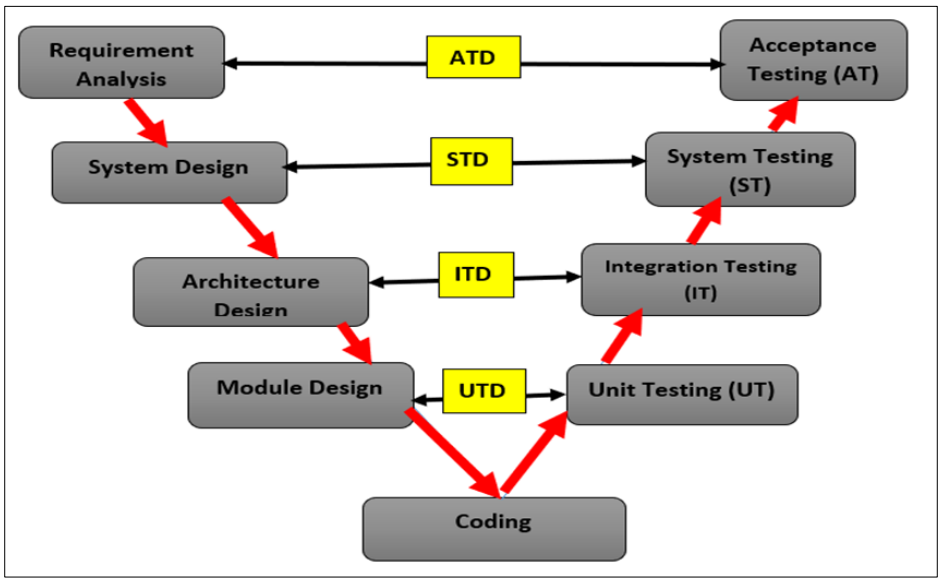

Figure 5 V-Model Diagram

Verification related activities encompass the static analysis or reviews done without code execution. In other words, they are numerous evaluations to ensure that system requirements are met. Validation on the other hands, encompasses dynamic analysis, code execution and testing. One major objective here is to evaluate the resulting software emanating from development phase to ensure it meets customer expectations and requirements. Obviously, the major strength of V-Model is that it involves thorough testing at all its numerous stages of system development. There are four major tests in V-Model, and each of these are designed before they take place. In the diagram, the tags ATD, STD, ITD and UTD represent acceptance test design, system test design, integration test design and unit test design respectively. Unit testing [24] is done during module design phase, and is executed to eliminate bugs at code or unit level.

The Integration testing [25] is performed at the architecture design phase, and the major aim is to ensure correct modules integration and inter-communication. System testing [26] focuses on testing the entire system functionality, inter dependency, and communication. This is where the functional and non-functional requirements of the system are verified. The User Acceptance testing [27] is used to verify that completed application meets user's requirement in real world.

7. Comparative studies

It is necessary at this stage to perform critical analysis [28] of the four models. This will be done using the Strength Vs Weakness (SVW) comparative studies as depicted in Table 1.

Table 1 SVW Comparative Studies

Process Model	S/N	Strength	Weakness
Waterfall Model [29]	1	The Waterfall model is simple to understand and implement.	Unfortunately, it is always difficult to get all requirements at project start.
	2	It is highly systematic in nature	It is difficult to go back to a previous phase.
	3	It allows for proper documentation.	Working software may be available late, since all the phases must be followed chronologically.
	4	It is easy to implement and manage.	It is risky, since risk management is not considered.
Incremental Model [30]	1	The first version of software is produced from the first module	Decision on the number of increments is always difficult.
	2	High success rate	Additional increments may introduce software bugs.
	3	It is easy to manage and monitor progress.	Documentation is more difficult to handle due to need for continuous updates.
Spiral Model [31]	1	Spiral model favours large software products.	The process of spiralling may go on longer than necessary.
	2	It also lays emphasis on risk-based development.	It could be expensive and unsuitable for small projects.
	3	Working software is produced as early as possible.	It requires expert skills for risk analysis.
	4	It accommodates future end-user requirements	Documentation is more difficult to handle due to need for continuous updates.
V-Model [32]	1	V-Model is highly disciplined, and phases are completed one at a time.	There is high risk and uncertainty
	2	It is simple and easy to implement.	It may not be appropriate for complex and object-oriented projects.
	3	This model focuses on project verification and validation early enough, thus ensuring an error-free and quality product.	It is not suitable for software projects whose requirements are unclear at inception.

8. Choice determinant factors

There are a number of factors that determine the choice of software development model. Eight of such common factors shown in Fig. 6 will be discussed.

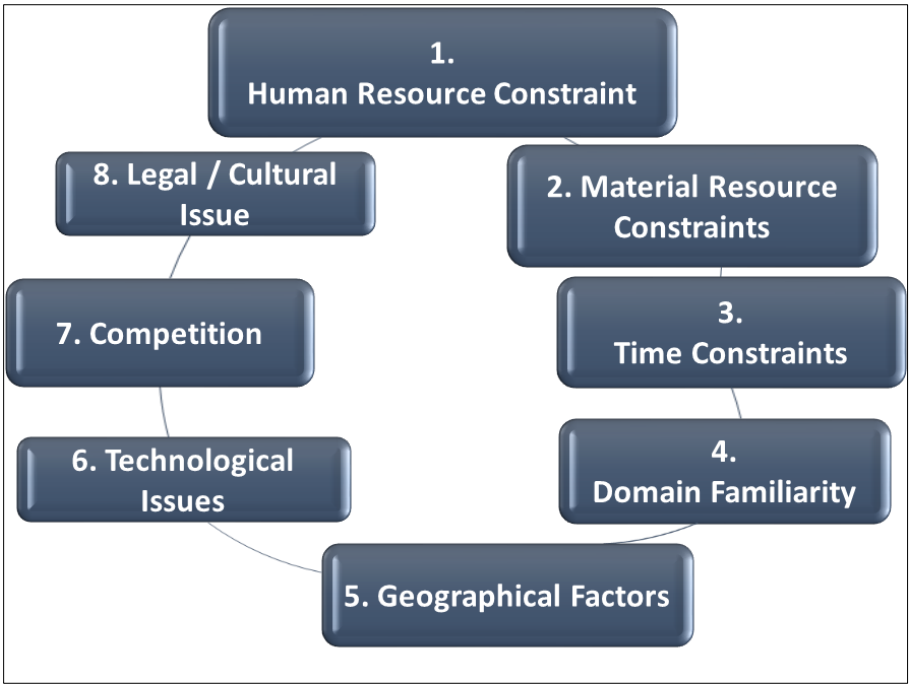

Figure 6 Determinant Factors Listing

Human resource constraints [33] could affect the decision on which software development model to choose. For instance, if a particular software requires expertise in particular niche area of computing, but a company lacks personnel in that area, the project manager may be required to make a decision on how best to tackle the challenge. One of the decisions may be to go for a particular developmental model, or to drop it. Human resources requirement can also come from the angle of end-user [34] expertise. In general, the existence or lack of particular software stockholders can affect decision on which software model to adopt.

Material resources constraint [35] can also affect the choices of software development model. For instance, financial constraint can lead to material constraints. In other words, decisions on which software model to use will take careful consideration on the financial and material resources available and budgets must be tailored within what is affordable.

Time constraint [36] is very key. Given two software products, one urgently needed while the second one rather needed in 12 months. It is obvious that the two may require different models. Domain familiarity [37] is also a decision factor. If software is needed to be developed for usage in specialized domains such as aviation, agriculture, military, mining, and so on, it may require the analysts, designers and developers to elicit specialized knowledge from domain experts. This will definitely affect the model of usage. Geographical factors [38] is another important decisional factor. For instance, if the software being developed is used in a different geographical setting, it will affect decision on software development model. The software developed on the planet earth, for usage in the moon, should put into cognizance the fact that the gravitational force on the earth is quite different from what is obtainable in the moon and vice versa. The sixth one is technological factor [39]. Changes in technology can affect decision on which software model to use. This is because, new techniques and technologies are evolved on a daily basis, and software practitioners are required to be as innovative and current as possible. Competition [40] can also affect choice of software development model. This stems from the fact that competing establishments will most likely choose the method that places them on a competitive edge over their peers in the same industry. Finally, legal and cultural issues [41] can also affect choice of software development models. This is because operational agreements governing an establishment, as well as the law of the land could be important in determining the software model to use.

9. Conclusion

This research has presented a very unambiguous study of four software development process models. The work was concluded by presenting a SVW comparative table showing the strength and weaknesses [42] of each of the models. This research has also presented eight factors that affect the choice of software development model. It is believed that this research will be of relevance to software researchers, and other practitioners.

Compliance with ethical standards

Acknowledgments

The authors acknowledge themselves for a successful collaboration.

Disclosure of conflict of interest

The Authors of this paper, Monday Eze and Charles Okunbor certify that they have no conflict of interest.

References

[1] Ozgur E. Design and Implementation of a Software Development Process Measurement System: An MSc Thesis Submitted to Department of Electrical and Electronics Engineering, The Middle - East Technical University, Turkey. 2004.

[2] Maria C, Isidro R. The Baseline: The Milestone of Software Product Lines for Expert Systems Automatic Development. In Proceedings of Ninth Mexican International Conference on Computer Science, Mexico. 2008; 6-10.

[3] Massimiliano P, Jonathan M. Guest editorial: special section on Software maintenance and evolution. Empirical Software Engineering. 2015; 20(1): 410–412.

[4] Nicolas C. Why Software Fails. IEEE Spectrum. 2005; 42 (9): 42-49.

[5] Chang L, Peng H, Scott S. Understanding, Detecting and Localizing Partial Failures in Large System Software. In the Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation. 27 Feb 2020; 559-574.

[6] Sergey V. IT Crisisology: The New Discipline for Managing Software Development in Crisis. Procedia Computer Science. 2019; 159(1): 1777-1786.

[7] Ogheneovo E. Software Maintenance and Evolution: The Implication for Software Development. West African Journal of Industrial and Academic Research. 2013; 7(1): 81-92.

[8] Bindia T. A Review on Models of Software Development Life Cycle. Data Research. 2019; 3(1): 61-68.

[9] Mike Y. The theoretical foundation(s) for Systems Engineering?. Systems Research and Behavioural Science. 37(1): 184-187.

[10] Edna C, Angelica C, Eloisa M, Pedro C, Fernanda L. Perceptions of ICT Practitioners Regarding Software Privacy. Entropy. 2020; 22(4): 1-23.

[11] Krishpersad M, Anthony A. Hydro power energy resources in Nigeria. Journal of Engineering and Applied Sciences. 2009; 4(1): 68-73.

[12] Lamiaa A, Tarek E. Reducing Carbon Dioxide Emissions from Electricity Sector Using Smart Electric Grid Applications. Journal of Engineering. 2013; 13(845051): 1-8.

[13] Samar A, Samer S, Hiba A. Agile Software Development: Methodologies and Trends. International Journal of Interactive Mobile Technologies. 2020; 14 (11): 246-270.

[14] Yingxu W. Software Science: On the General Mathematical Models and Formal Properties of Software. Journal of Advanced Mathematics and Applications. 3(2): 130–147.

[15] Mamdouh A, Sadiq A. Security Risks in the Software Development Lifecycle. International Journal of Recent Technology and Engineering. 2019; 8(8): 7048 – 7055.

[16] Vinita M, Sukhdip S. Risk Driven Testing, Managing Risks and Quality Assurance: An Introspection and Benchmarking Implementation. International Journal of Innovative Technology and Exploring Engineering. 2019; 8(8S2): 630-638.

[17] Sunil R, Vinay K. A Review of Conventional SDLC Process Models. International Journal of Scientific Research and Reviews. 2019; 8(2): 4186-4191.

[18] Neelu L, Kavitha D. Software Engineering for Smart Healthcare Applications. International Journal of Innovative Technology and Exploring Engineering. 2019; 8(6S4): 325-331.

[19] Mohanad A, Tariq T, Abdul-Aziz R. Multimedia Software Engineering Methodology: A Systematic Discipline for Developing Integrated Multimedia and Software Products. Software Engineering. 2019; 8(10): 1-10.

[20] Azhar S. System Development Method with The Prototype Method. International Journal of Scientific and Technology Research. 2019; 8(87): 141-144.

[21] Eze N. Development Process for controllable Software. International Journal of Education and Research. 2017; 5(7): 37-52.

[22] Ehmer K, Shadab M, Farmeena K. Empirical Study of Software Development Life Cycle and its Various Models. International Journal of Software Engineering. 2020; 8(2): 16-26.

[23] Li-Ren Y, Kun-Shan W, Chun-Far H. Validation of a model measuring the effect of a Project manager's leadership style on project performance. KSCE Journal of Civil Engineering. 2013; 17(2): 271-280.

[24] Michael O. Unit testing: Test early, Test often. Journal of Computing Sciences in Colleges. 2003; 19(2): 319-328.

[25] Roopa S, Imran K. An approach for Integration testing in online retail applications. International Journal of Computer Science and Information Technology. 2012; 4(3): 141-158.

[26] Sukhdev G. Software Testing Techniques. International journal of computer science and mobile computing. 2014; 3(10): 988-993.

[27] Shreyas P. Acceptance testing technique: A survey along with its operating frameworks. International journal on recent and innovation trends in computing and communication. 2016; 4(4): 772-775.

[28] Javed K, Khan AH, Tubbassum L. Critical Analysis of Software Development Methodologies based on Project Risk Management. International Journal of Academic Research in Business and Social Sciences. 2019; 9(12): 253–263.

[29] Lucas K, Carl M. Waterfall and Agile information system project success rates—A South African perspective. South African Computer Journal. 2020; 32(1): 43–73.

[30] Nandini R, Abdul R. Incremental Models of Policy Formulation and Non-incremental Changes: Critical Review and Synthesis. British Journal of Management. 2005; 6(4): 289-302.

[31] Soobia Saeed, N, Mehmood N, Mamoona H. Analysis of Software Development Methodologies. International Journal of Computing and Digital Systems. 2019; 8(5): 445-460.

[32] Mustafa D, Ilker U, Roman T, Josef B. Enhanced V-Model. Informatica. 2018; 42(1): 577–585.

[33] Bouzidi-Hassini S, Benbouzid-Si F, François M, Rabahi M. Considering human resource constraints for real joint production and maintenance schedules. Computers & Industrial Engineering. 2015; 90: 197-211.

[34] Barbara B, Fabio C, Daniela F, Antonio P. End-user development, end-user programming and end-user software engineering: A systematic mapping study. Journal of Systems and Software. 2019; 149(1): 101-137.

[35] Zhenfeng L, Jian F, Jinfeng W. Resource-Constrained Innovation Method for Sustainability: Application of Morphological Analysis and TRIZ Inventive Principles. Sustainability. 2020; 12(917): 1-23.

[36] Rui H, Yingbo L, Lijie W, Jianmin W. Dynamically Analyzing Time Constraints in Workflow Systems with Fixed-Date Constraint. In Proceedings of the 12th Asia-Pacific Web Conference, APWeb 2010, Busan, Korea. April 2010; 6-8.

[37] Alberto E. Familiarity, complexity, and team performance in geographically distributed software development, Organization Science. 2007; 18(1): 613-630.

[38] Aleksandra M, Monika Z. The characteristics of geographical information systems in terms of their current use. Journal of Water and Land Development. 2018; 39(1): 101-108.

[39] Dmitriy C. On the role of Switching Costs and Decision Reversibility in Information Technology Adoption and Investment. Journal of Information Systems and Technology Management. 2017; 14(3): 309–321.

[40] Zeljko S, Ilija H, Pece M, Vladimir B. The Role of Effective Software Maintenance in Increasing Competitiveness of Very Small Software Companies. In Proceedings of SMEs Development and Innovation: Building Competitive Future of South-Eastern Europe, held at Ohrid, Macedonia. 2014; 835-845.

[41] Abbe B. Access to essential technologies: The role of the interface between intellectual property, competition and human rights. International Review of Law, Comp. & Tech. 2010; 24(1): 51-61.

[42] Christine N, Shastri L, Torsten R. Design of a SWOT Analysis Model and its Evaluation in Diverse Digital Business Ecosystem Contexts. Procedia Computer Science. 2019; 159(1): 1145–1154.