System development lifecycle essay

Table of Contents

References

1. 1. Introduction

According to Allerano and Taverz (2012), the system development lifecycle has dominated the information technology (IT) arena for numerous decades and remains one of the most commonly used methods in software development and acquisition. It is observed that, though the system development lifecycle has evolved over the years and has gone through a multitude of paradigm shifts with respect to building of software, at its core the process is resilient and its central tenets are applicable in business, industry and research (McMurtrey, 2013). Piccoli (2012), in his research,

reflects on the SDLC system as being a dominant system development methodology in the 21st century, along with the method of prototyping. Langer and Langer (2008) define the system development lifecycle as one which promotes the planning, analysis, implementation and maintenance of information systems and which thereby serves as the foundation of the different activities associated with every phase of the SDLC. According to Hoffer et al. (2011), the process of systems development lifecycle can be broadly categorised into planning, analysis, design, development and testing so as to meet the stages of the various iterations and models of SDLC development. This report will focus on the importance of SDLC in different stages of system implementation and describe three methods of SDLC development.

1. 2. Importance of SDCL in System Implementation

According to Piccoli (2012), the SDLC is a guide to the project as it provides the flexible and consistent medium required to accommodate specific changes through the information system development and helps meet the objectives of the client. This section will examine the importance of SDLC in the investigation of a specific system through the different phases (Figure 1).

Figure 1: Stages of SDLC

Source: Adapted from Piccoli (2012)

This report argues that to date most system analysis and design books identify the need to promote SDLC from these five stages so as to help in the comparison of different models (Kendall and Kendall, 2011). Hoffer et al.

(2011), however, criticise the system and argue that the systems analysis and design taking place in a cycle sometimes is pervasive and makes it difficult to develop and model a specific system. This research, however, will focus on the positives of the SDLC by identifying with its role in terms of its objectives to meet organisational requirements and its objectives during different stages of systems development.

1. 2. 1 SDLC Objectives

According to Kendall and Kendall (2011), the use of the SDLC framework as part of systems development helps in ensuring the quality of systems. The authors contend that the SDLC should aim at promoting return on investment by enabling cost savings, product flexibility, improved decision support or promotion of strategic and operational planning. Leau et al. (2012) reiterate this view by indicating that through all the stages of the lifecycle development, SDLC helps relate a high degree of intrinsic quality.

According to Khan et al. (2013), the importance of SDLC implementation for information system adoption is to provide a certain degree of management control. The author indicates that the use of the SDLC process should ensure that there is sufficient degree of information in terms of predictability (i. e. planning estimates for different stages of the project, the associated resources needed and the various stakeholders involved). It is also contended that the SDLC should promote management control by ensuring time for feedback from the management and the stakeholders.

Leau et al. (2012) identify that the role of SDLC is to help in the maximisation of product performance by ensuring that the productivity of

the project is promoted by meeting the ROI expectations. Hoffer et al. (2011) further indicate that the SDLC plays a primary role in delineating between the tasks and deliverables of a given project, thereby ensuring that resources are used in the most effective and efficient way possible.

1. 2. 2. Role of SDLC in Different Stages of the Project

According to Hoffer et al. (2011), the use of SDLC in the planning phase is for three primary reasons: identification and selection of the system for development, assessment of project feasibility, and the development of project plan. Kendall and Kendall (2011) contend that in the planning stage the SDCL system is useful in focusing on the most ideal system which can help support the goals of the organisation. Boehm et al. (2000) further reiterate that the identification and selection of a system can be carried out by using specific strategic tools like value chain analysis (which helps identify the extent to which a new system is important to an organisation) and cost benefit analysis (the process by which the benefits and costs are compared). Nurmuliani et al. (2004) argue that the planning stage of the SDLC also helps identify the feasibility of a proposed system adoption and whether the same can be promoted from a financial, technical and organisational perspective. Therefore it is contended that the use of the SDLC is important in the development of the final project plan, wherein the activities of the system development are identified to ensure that the systems development is on time.

During the analysis stage of systems development, the SDLC plays a vital role in the gathering and building of business requirements. The SDLC helps

define the requirements of a system by identifying the needs of the project during the communication, documentation, management and change processes. Hoffer et al. (2011) contend that the development of a system using the SDLC process also helps in process modelling, wherein the entire system process and the flow of data within the system can be identified. The use of process models and data flow diagrams is to present a visual presentation of the system that is to be integrated with the organisational environment.

The SDLC design stage involves the designing of the desired features into the system, wherein the design of the IT infrastructure and the systems model is important. According to Kendall and Kendall (2011), during the design of the IT infrastructure stage the identification of the type of networks, the type of clients and servers are identified, along with the type of database which is to be adopted. The authors further argue that the SDLC design stage helps in the designing of the system models, wherein the graphical user interface (GUI) and data models are used to help present a representation of the system model.

In the systems development phase, the development of the IT infrastructure as per the design stage, along with the type of databases and programmes, is carried out. The final stage is the testing phase, where the errors, bugs and interoperability of the product are tested to ensure that the business requirements of the analysis are met. According to Devi (2012), the SDLC helps in the testing of project needs, project requirements and project transition based on the identification of defects and bugs in the project.

1. 3. Different Models of Systems Development1. 3. 1. Waterfall Model of Systems Development

According to Munassar and Govardhan (2010), the first model of SDLC is the waterfall model, which is static in nature and approaches the process of system development in a linear manner, wherein one activity is to be completed before the next activity begins, thereby promoting a sequential approach. Fowler (2004) argues that the adoption of the waterfall style helps breaks up the project activities into requirements analysis (planning and analysis), design, coding (development) and testing. Pfleeger and Atlee (2006), on the other hand, identify the waterfall model as one which needs more detailed steps to present the phases of requirements analysis, systems design, program design, coding, unit and integration testing, systems testing and maintenance during operation. The author expands on the original waterfall approach, as he feels that there is a need to focus on the functioning of the systems after testing and the need to expand on the testing process. Fowler (2004) further contend that the waterfall model is most effective for systems implementation, as specific goals for different phases of development are promoted. In this approach, once a single phase is completely developed, it proceeds into the next phase. This supports a structured and process centered approach, wherein every stage is clarified with respect to the objectives of the design (Figure 2).

Figure 2: Waterfall Model of Systems Development

Source: Adapted from Pfleeger and Atlee (2006)

Fowler (2004) argues that though there is a period of handoff between phases and associated backflows, it is important to ensure that this is avoided. Researchers, however, contend that this is a primary drawback of the waterfall model, as there is an inherent inability to revisit a previous stage if there are any errors. For instance Adenowo and Adenowo (2012) contend that if there is an error detected during the implementation (coding) phase, there is limited opportunity to revisit the requirements analysis or the design phase to correct the error. On the other hand, Pfleeger and Atlee (2006), contend that the waterfall approach is promoted in structured systems development, wherein the alteration of the software after coding is prohibited.

Over the years there have been improvements made to the waterfall model. McConnell (2010) identifies these models to be modified waterfalls, wherein phases of the project are allowed to overlap, where every phase is found to influence and is influenced by the next and previous phases of the model. The authors contend that the overlap of phases helps address the inherent limitations of lack of flexibility of the waterfall model.

1. 3. 2. Spiral Model

According to Boehm and Hansen (2000), the spiral model of software development is one which places emphasis on risk analysis. The development of any system goes through the four phases of planning, risk analysis, engineering and evaluation. Boehm (1988) argues that the adoption of this model involves the system going through the phases in iterations. The baseline spiral is found to start with the planning phase,

wherein the requirements of the system are gathered. The subsequent spiral is built on the baseline spiral, wherein the risks are identified, assessed and methods for risk mitigation are developed. At the end of the risk planning phase, a prototype is generated. The third spiral is the engineering spiral, which enables the development of the system, which is then evaluated by testing in the final phase. In this approach, the angular component represents the progress, while the radius of the spiral represents the cost of the model.

Figure 3: Spiral Model

Source: Boehm and Hanson (2000)

An alternative to the spiral model was proposed by Boehm et al. (1998), wherein the theory of ‘ win-win’ was adopted. This model was developed to address the inherent limitations of the spiral model. According to Boehm and Hanson (2000), the problem related to the application of the spiral model is that there is limited guidance in terms of determining the objectives, constraints and alternatives in a given approach. To address this problem, Boehm et al. (1998) adopted the win-win spiral model, which first identifies the stakeholders of the system and their win condition to then determine the associated set of objectives and alternatives. This model involves four steps, as presented in the following Figure 4.

Figure 4: Steps in the Win-Win Spiral Model

Source: Author (2014)

1. 3. 3. Agile Development

According to Cohen et al. (2003), the adoption of an agile process of software development is based on promoting incremental and iterative development, wherein the phases of the development lifecycle are revisited a number of times. Dingsoyor et al. (2012) argue that the adoption of this approach involves the improvement of the software while using customer feedback to ensure that a convergence on the proposed solutions is arrived at. Nerur et al. (2010) identify that in the agile development model, when compared to other traditional models, the process of lifecycle development is not just divided into large phases, but there is use of smaller parts called increments or iterations, which help in promoting the development cycle.

Cohen et al. (2003) summarise that the agile development process involves four primary attributes, including the early customer involvement, iterative development, self organising teams and adaptation to change. Nerur et al. (2010) argue that agile development methods are currently predominantly used, of which six primary approaches are promoted extensively, including crystal methods, dynamics development, feature driven development, lean development, extreme programming and scrum. This report will focus on one approach, extreme programming, and present its features.

According to Dingsoyor et al. (2010), the adoption of the extreme programming approach relies on development and delivery of small increments in functionality. This process is largely dependent on the improvement in code in a constant manner. This process involves incremental planning where the requirements are recorded to be included along with the release. The key feature of extreme programming is the small releases, wherein the minimal useful set of functionalities providing business

value is first developed. In this approach, there are frequent releases with improvement in product functionality.

1. 3. 4. Comparison of Different Methods

The following Figure 5 presents a comparison of the limitations of the above approaches.

Figure 5: Limitations of Systems Development Models

Source: Author (2014)

This report contends that the agile development methods are better than the traditional development methods. This is because the agile development method promotes the successful delivery of results in a quick and inexpensive manner, with an emphasis on teams and customer collaboration. In contrast, the traditional methods of waterfall development and spiral methods focus on contracts, plans and processes with a one step involvement of the client (Cohen et al., 2003). It is also argued that since the development takes place in iterations, it is possible to change the direction of development at any stage with limited change in cost. Such an approach is effective in ensuring that the productivity and return on investment objectives are met.

References

Adenowo A. A &Adenowo B. A (2012). Software engineering methodologies: A review of waterfall model and object oriented approach, *International Journal of Scientific & Engineering Research, 4(* 7), 427- 434.

Arellano, M. M., &Tavarez, J. M. M. (2012, June). A comparative analysis about Software Development Life Cycle Methodologies involving Business Processes and Web Services. In *Information Systems and Technologies (CISTI), 2012 7th Iberian Conference on* (pp. 1-6). IEEE.

Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer* ,(5), 61-72.

Boehm, B., Abts, C., &Chulani, S. (2000). Software development cost estimation approaches—A survey. *Annals of Software Engineering* , (1-4), 177-205.

Boehm, B., & Hansen, W. J. (2000). *Spiral development: Experience, principles, and refinements* (No. CMU/SEI-2000-SR-008). CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST.

Boehm, B., Egyed, A., Kwan, J., Port, D., Shah, A., &Madachy, R. (1998). Using the WinWin spiral model: a case study. *Computer* ,(7), 33-44.

Cohen, D., Lindvall, M., & Costa, P. (2003). Agile software development. *Data & Analysis Center for Software (DACS), New York* .

Devi, R. (2012). Importance of Testing in Software Development Life Cycle. *International Journal of Scientific & Engineering Research, 3* (5), 1-5.

Dingsoyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software* , U(6), 1213-1221.

Fowler, M. (2004). *UML Distilled: A Brief Guide to the Standard Object ModelingLanguange* . Addison-Wesley Professional.

Hoffer, J. A., George, J. F. and Valacich, J. S. (2011) *Modern Systems Analysis and Design.* Prentice Hall, Boston.

Kendall, K. and Kendall, J. E. (2011) *Systems Analysis and Design* , 8/E, Prentice Hall , Englewood Cliffs, NJ.

Khan, M. F., Qazi, K. A., & Shah, K. A. (2013). Performance Evaluation of Software Development Models. *Software Engineering* ,(1), 1-4.

Langer, A. M., & Langer, A. M. (2008). System Development Life Cycle (SDLC). *Analysis and Design of Information Systems: Third Edition* , 10-20.

Leau, Y. B., Loo, W. K., Tham, W. Y., & Tan, S. F. (2012). Software Development Life Cycle AGILE vs Traditional Approaches. In? International Conference on Information and Network Technology (ICINT 2012) IPCSIT(Vol. 37).

McConnell, S. (2010). *Rapid development: taming wild software schedules* . O’Reilly.

McMurtrey, M. (2013). A Case Study of the Application of the Systems Development Life Cycle (SDLC) in 21st Century Health Care: Something Old, Something New?. *Journal of the Southern Association for Information Systems* ,(1).

Munassar, N. M. A &Govardhan, A. (2010). A Comparison of Five Models of Software Engineering.

International Journal of Computer Science, 7(5).

Nerur, S., Cannon, A., Balijepally, V., & Bond, P. (2010). Towards an Understanding of the Conceptual Underpinnings of Agile Development Methodologies. In *Agile Software Development* (pp. 15-29). Springer Berlin Heidelberg.

Pfleeeger, S. L. & Atlee, J. M. (2006). *Software Engineering: Theory and Practice* , 3rd Edition. US: Prentice Hall

Piccoli, G. (2012) *Information Systems for Managers: Text and Cases* , John Wiley & Sons, Inc., Hoboken, NJ.