ISSN: 2278-3369

International Journal of Advances in Management and Economics
Available online at: **www.managementjournal.info**

RESEARCH ARTICLE

Understanding Traditional Systems Development Methodologies

Tefo Sekgweleo*

Polytechnic of Namibia, Department of Business Computing, Windhoek, Namibia

*Corresponding Author: Email: ***ts330ci@gmail.com***

Abstract

Software development methodology plays a vital role in systems development life cycle. It is a framework that guides the systems development team in achieving what the customer/user has requested. Decision making may impact the systems development team positively or negatively. Hence, understanding strengths, limitations, how, why and who can use the software methodology is imperative. It helps all the stakeholders involved in a systems development team to make informed decision for a particular project. Hence, the software development methodology is not a silver bullet for all the projects.

Keywords: *Systems development methodology (SDLC), Information technology (IT), Systems development.*

Introduction

There is a stage in life where a need arises within the organisation to purchase or develop a system. This need may require the organisation to undergo through a system development life cycle (SDLC) in order for it to increase productivity to remain competitive. To achieve that, most organisations have to make a decision to developing a system or purchase a system. Therefore, there are various systems development methodologies that can be followed after making that crucial decision because there are finances involved.

Nelson and Teng [1] define SDLC as a guideline and logical process used by system developers to develop systems. According to Rob [2], SDLC stipulates the required ways that comprises various stages and activities to successfully develop the system. However, it should be taken into consideration that the methodology is not one size fit all. The software development team has to carefully select the appropriate methodology for a particular project they are undertaking.

These methodologies serve as a framework to be followed by a software development team. It can also be used to ensure that the designed solution meet the user requirements that supports business strategic goals and objectives. The SDLC can be either agile or traditional. However, both methodologies are made up of various stages namely analysis, design, development, implementation and maintenance [3]. However, the main purpose of this paper is to examine the traditional methodologies and to understand why, who and how they are used and also highlight the limitations thereof.

Traditional Systems Development

The traditional systems development existed prior to the agile systems development. These methodologies include waterfall method, V-model, Rational Unified Process and others [4]. According to Matkovic and Tumbas [5], these methodologies are based on the systems development principles that have served as a foundation for the creation of the systems development to date which can be either sequential or iterative. Sequential approach means that the methodology is made up of a series of steps/stages that follow each other sequential. The steps are dependent of each.

With the iterative approach, [6] posits that the methodology divides the intended system into a series of versions. After the implementation of version1 the additional work is done on version 2 and the process continues until the completion of the overall system. The emphasis of the traditional development approach is to create ample documentation which serves as a means for communication and traceability of the design [7]. Documentation also plays a vital role in sharing knowledge and keeping tacit knowledge within the organisation. However, knowledge management is of vigorous importance within organisations [8]. Documentation serves as a means of knowledge for newly recruits and any organisational employee who may want to join the systems development team.

There is a variety of traditional systems development methodologies that can be adopted within the organisation. They include waterfall, Spiral, V-Model, Rational Unified Process (RUP) and Rapid Application Development [9]. The traditional methodology is the most commonly used approach by organisations whereby software development activities are completed sequentially. The traditional methodologies are briefly explained below.

Waterfall Model

The waterfall model (also referred to as systems development life cycle) is the most popular of the traditional models. This model was originally proposed by Winston W. Royce in 1970 to define a potential software engineering practice [10]. It is made up of various stages and has distinct goals for each stage of software development. Hedman and Lind [11] argue that the waterfall model is a process that describes and recommends the stages that have to be completed in the process of developing a system for a particular usage. It consists of six stages including requirement gathering, analysis, design, testing, implementation, and maintenance [12].

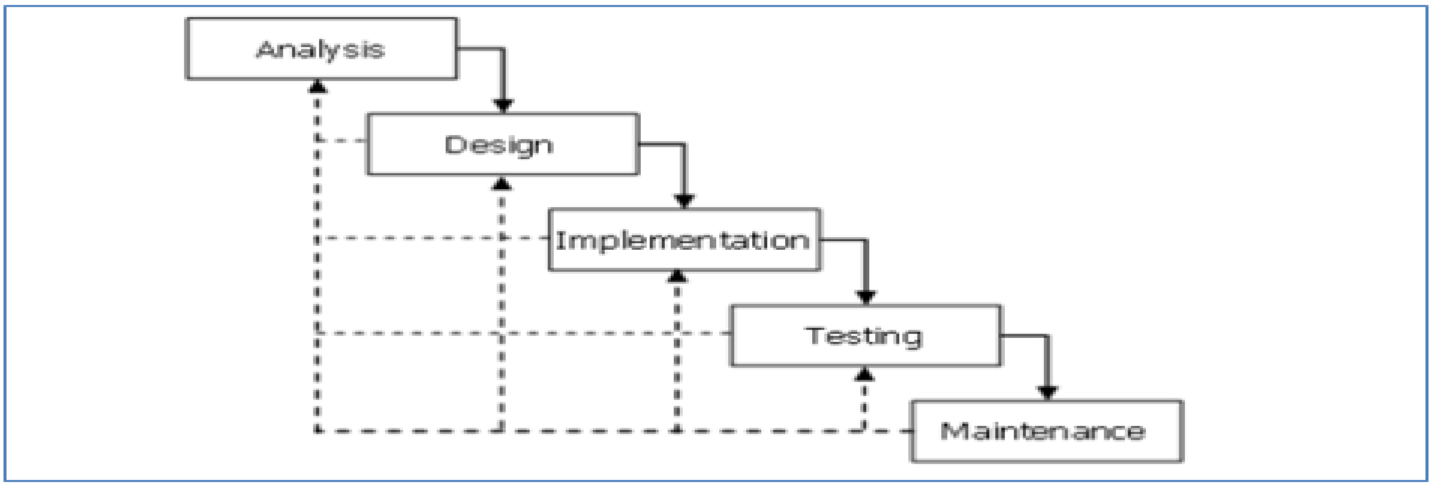

Figure 1: Adopted from [10]

These stages are dependent on each other and they follow each other sequentially. Pefkaros [13] posits that each stage within the waterfall model flows downward into each other. Each stage has to be completed prior to the next stage could commence [14]. One of the strengths of the waterfall model is the extensive documentation of requirements which is good for communication among the systems development team [15].

However, the waterfall model does not allow changes to be made to the previously completed stage [16]. As a result, the system will have to be implemented with missing/faulty requirements or mistakes committed in any stage of system development. Fixing such mistakes is not easy but costly and it also leads to late delivery of the requested system [15]. The user requirements keep on changing throughout the system development because the client does not usually know what they exactly want. To fix mistakes or gaps encountered in the business requirements specification, change requests are often logged but can only be attended to once the system has been implemented.

Spiral Model

The spiral model was introduced by Barry W. Boehm in 1988 [5]. It was introduced to solve the limitations encountered in the waterfall model. Boehm created the spiral model with the intention of introducing iterative software development. This model combines the features of the prototyping and the waterfall model.The spiral model consists of four stages starting with the planning, objectives, risk analysis and development [14].

The model arranges all the activities in the form of a spiral. All the stages are continuously repeated for a certain period of time until the completion of the requested system [18]. The emphasis of the spiral model is to evaluate risks, which are used as a source for decision making to further develop the system [5]. In each cycle, problems that are encountered are resolved. The next iteration occurs until the system completed and meets the user requirements. A prototype is built for every iteration.

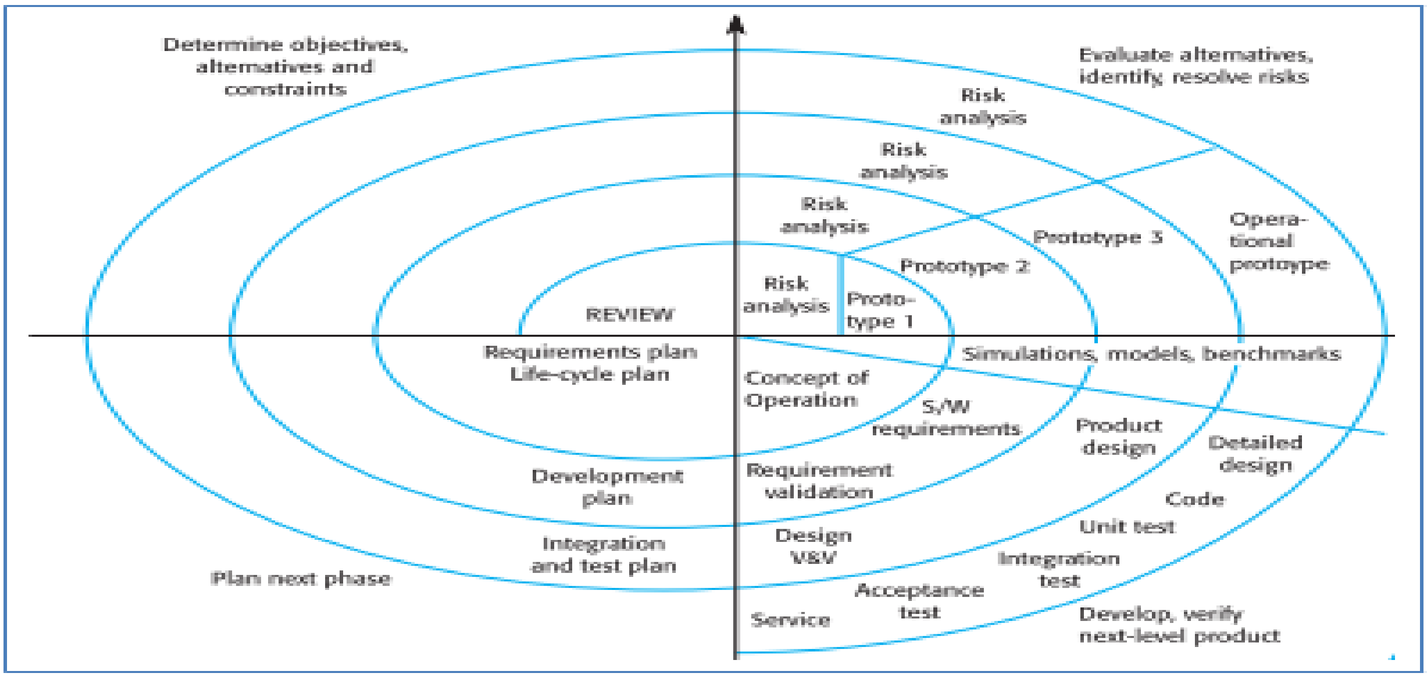

Figure 2: Adopted from [17]

Due to iterative pattern of the spiral development, feedback given on each stage makes it possible to fix errors at early stages, enhance requirements and get rid of risks identified. According to Butt and Hameet [18], problems encountered in every iteration, are resolved quicker and possible risks are removed earlier stages of systems development. This approach makes it possible for the organisations to safe costs since it is cheaper to identify problems and risk in the early stages of the systems development. This model also makes it possible to enhance or make changes to the requirements until the acceptable system is delivered to the users.

The spiral development starts smaller and grows bigger depending on the number of iterations. However, lots of activities occur parallel and make it difficult to manage the systems development and rework is likely to occur since requirements are not fully specified prior to systems development [19]. Due to the fact that requirements are not fully specified when the development starts, additional work may be required. The main reason for not specifying all requirements at once is because users do not normally know exactly what they want until the system is delivered to them. Another setback of the spiral model is that is works well for big projects than small ones [17].

V-Model

The V-Model was first proposed by Paul Rook in the late 1980s and can be thought as the extension of the waterfall model [20]. It was introduced was developed with the intention to address some of the problems encountered in the waterfall model. In the waterfall model defects were found very late in the development life cycle

because testing was not involved as early as the initial stage. The emphasis of the V-Model is more on the testing of each stage of the development life cycle. Balaji and Murugaiyan [21], posits that the V-Model illustrates the link between each stage of the systems development life cycle relating to its software testing stage.

Mushtaha and Tolba, [22] posits that the V-Model is made up of four main stages of the waterfall model with their equivalent testing stages such as requirements analysis -(acceptance testing), requirements specification-(system testing), design specification -(integration testing), program specification and coding-(unit testing).The mentioned testing activities should be carried out in parallel to the development activities so that testers can produce a set of test deliverables. However, the V-Model outlines who is responsible for conducting a particular testing at which stage [23]. Without that kind of information it would be very difficult to execute testing.

It is always a best practice to involve software testers at earlier stages of the product life cycle. The overlap of testing stage with the development stage ensures that problems encountered are addressed as early as possible [4]. Lee and Xia [24] posits that the response from software teams with regards to vital requirement changes in early stages of systems development life cycle is critical as it enables organisations to save time and cost in later stages. However, the V-Model does not indicate a clear path for problems encountered during the testing stage [17].

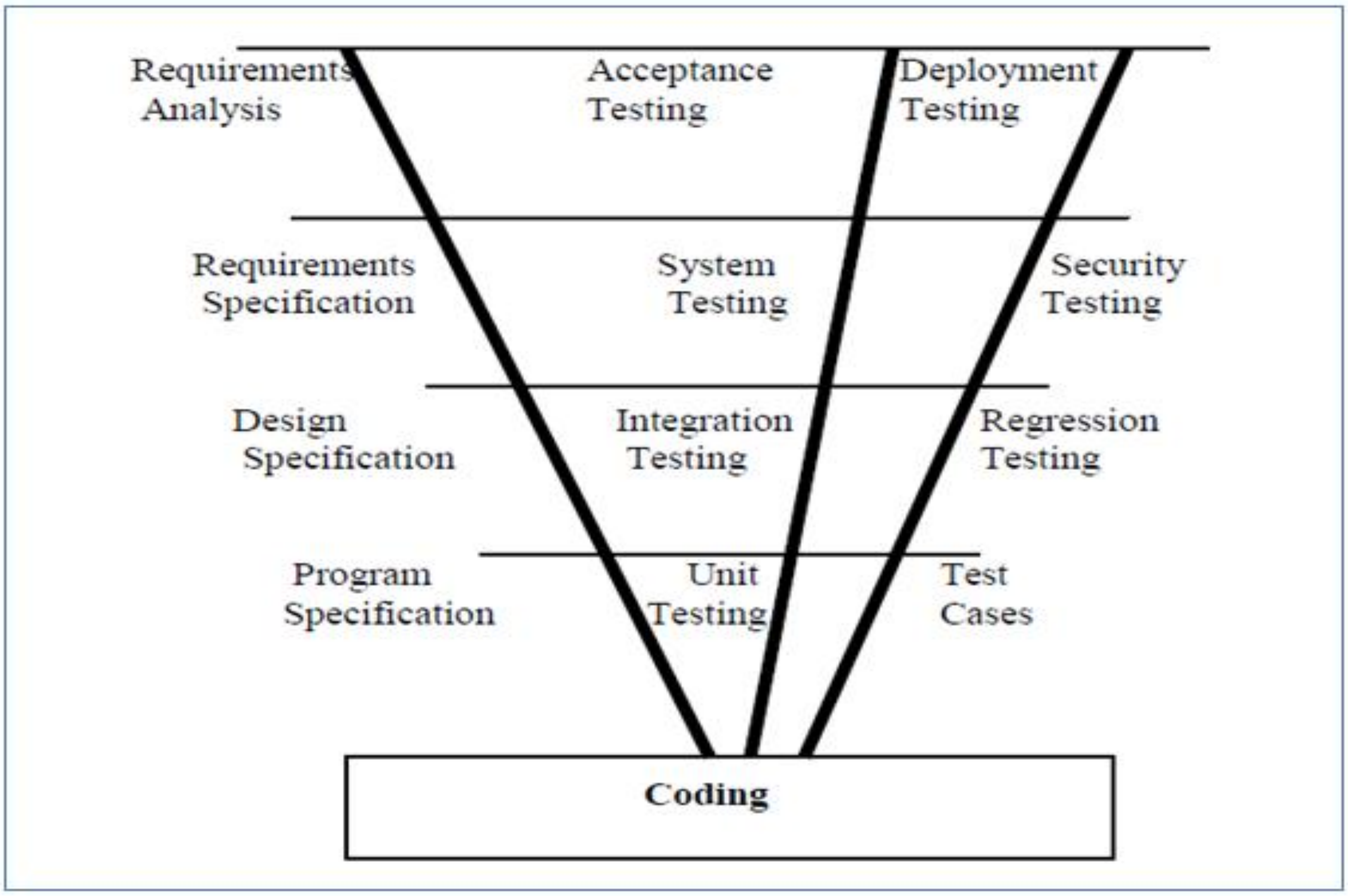

Figure 3: Adopted from

Rapid Application Development

The need arose in the early nineties, to speed up the systems development within organisations. That is when James Martin in 1991, introduced rapid application development (RAD) to rapidly develop systems [9]. The main objective of RAD is to develop systems faster and produce high quality results compared to linear traditional model. As a result, this enables organisations to take leadership in implementing latest technology systems quicker. In order to shorten the systems development schedules it is imperative for the team to identify the systems development methodology, tools, techniques and technologies suitable for the selected methodology [19].

RAD methodology makes use of Computer Aided Software Engineering (CASE) tools in combination with iterative development and rapid prototyping in order to achieve quick systems. Choo and Lee [25] posit that various products are used in RAD including testing tools, groupware for communication, requirements gathering tools, CASE tools, prototyping tools and language development environments such as Java and C++ environments. The advantage of using such tools is that they can be reused within or in other systems development projects. In order to achieve the best results, the systems development team should be dedicated and highly skilled in using the above mentioned tools.

RAD follows the same stages used in the waterfall model but it has added certain features to achieve quick and better results. Some of the features of RAD are outlined below:

Table 2: Adopted from (Avison and Fitzgerald, 2006)

RAD Features	Definition
Incremental Development	Requirements are never complete but evolves
Time Box	Functions developed parallel into time boxed cycle
Prototype	Developed functions are assembled into a working prototype
Pareto Principle	80/20 rule applied to requirements. 80% of the functioning system can be delivered, 20% effort required to complete 100% of the requirements
MoSCoW rules	Must Haves, Should Haves, Could Haves, Won't Haves
JAD Sessions	These meetings are used to beef up the requirements and occur throughout time box cycles
Sponsor and Champion	The success of the system relies on a committed sponsor and the champion of the system
Toolsets	It helps to speed up the process and improve productivity

Within RAD, clients are involved very early in the systems development because they provide feedback which helps to enhance requirements. A prototype is developed and given to clients to use in order to critique it and with that feedback a proper system is developed. The system developed in components/functions which occurs parallel in time boxed cycles which are then integrated into a working prototype. However, if the tools and techniques mentioned above are not properly managed then the systems development may not be a success. The success of the system is tremendously dependent on high technical skilled developers [26].

Rational Unified Process (RUP)

Due to the dynamic nature of technology, new methodologies keeps on being implemented to improve limitations encountered to its predecessors. According to Jain and Chandrasekaran [9] in 2000, Kruchten introduced rational unified process (RUP). It was introduced to consider the need for accommodating change and adaptability during the system development process [26]. As a result RUP becomes extremely flexible as it allows change to occur at any time at any stage of systems development. RUP is made up of four stages namely inception, elaboration, construction and transition [27]. These stages are executed sequentially and iteratively throughout the systems development life cycle. Every stage of RUP is composed of one or more iterations [28]. Any discrepancies, risks and errors encountered in each stage are addressed in each iteration of that particular stage. The final iteration forms part of the final system.

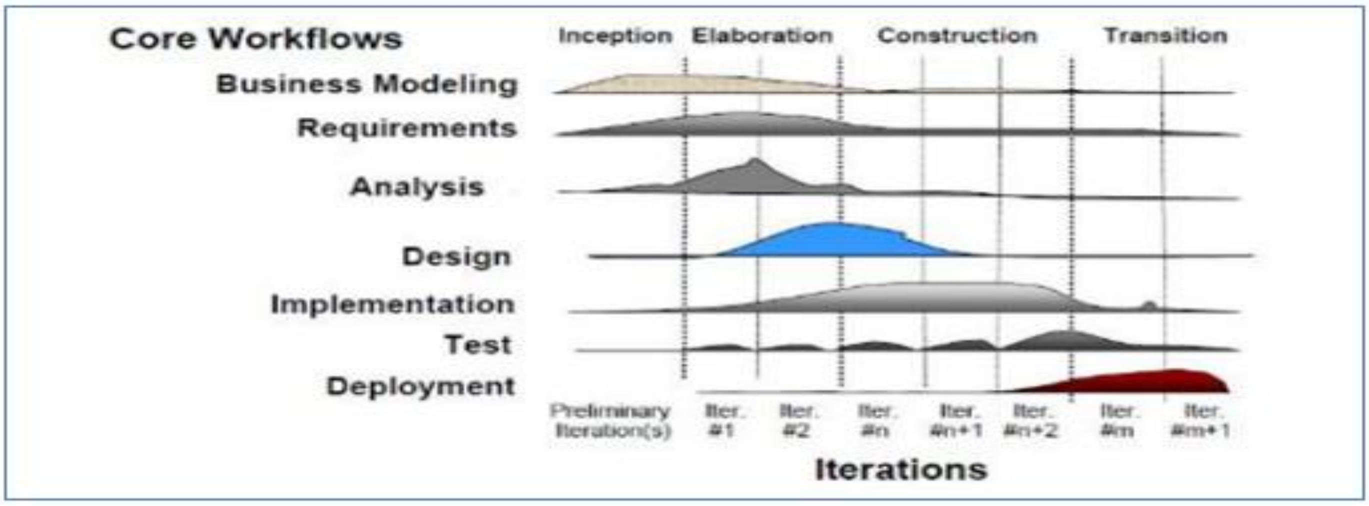

Figure 4: Adopted from [29]

Table 2: Adopted from

Discipline	Description
Business Modeling	Describe the business process and the internal structure of the business in order to capture proper requirements for the system to be built
Requirements Management	Elicit, organise and document the requirements
Analysis and Design	Create the architecture and design of the system
Implementation	Write and debug the source code, conduct unit testing and build management
Test	Conduct functional, integration, system and user acceptance testing
Deployment	Packaging the software, creating in-stallation setups, compiling end user manuals and other tasks needed to make the software available to its end users
Project Management	Project planning and monitoring
Configuration and Change Management	Covers all the tasks concerned about release management and change request management
Environment	Adapt the RUP process according to the needs of the project and selecting the supporting systems development tools.

The above mentioned stages are accompanied by nine principles described in the table.

The aim of introducing new methodologies is to fill up the gaps encountered in the previous methodologies. RUP has been adopted in the systems development industry as a model for the reason that it is well defined and documented [30]. The documentation is accessible electronically. RUP is an adaptable model allowing organisations to select elements of processes that are most relevant to the particular project. RUP make use of unified modelling language (UML) to emphasise object-oriented analysis and the maintenance model [31]. UML is an industry-standard language that allows the systems development team to clearly communicate requirements, architectures and designs visually [12]. Pictures or diagrams enable the entire systems development team to visualise the inner workings the system to be built. It also makes it easier for the team to explain the how the system is going to work.

The rational unified process (RUP) is a process framework that provides a disciplined approach to define activities and responsibilities inside the organised system development [32]. The four development stages are accompanied by the nine basic principles. Hence each stage is iteratively executed. The main goal of RUP is to make changes manageable because problems encountered in testing of each iteration are resolved early in the systems development life cycle [33]. RUP also ensure the production of high-quality software that meets the needs of its end-users, within a predictable schedule [12]. However, this model is too complex, not easy to learn and problematic to apply correctly if project managers or team members are not experts in using it [33]. Another limitation is that RUP is a commercial product and needs to be purchased from IBM before it could be used [33].

Why is Traditional Methods Applied

Any organisation that needs to develop a system may use the traditional systems development. These methodologies consists of a sequence of stages that must be followed and completed by system designers and developers in order to achieve some results and deliver the requested system [10]. One thing that users must know is that it takes time for the program to be complete. It is built in such a way that the programmer cannot move on to the next step if the step prior to that is not yet completed. The traditional systems development methods are dependent on a set of prearranged processes and continuing documentation which is written as the work evolve to guide further development [4].

The traditional development attempts to minimize change during system development through severe upfront requirements gathering, analysis and design with the intent to achieve higher quality results under a controlled schedule [34]. There are various specialists involved in each and every stage of systems development. Traditional methodologies, the development is done by immense and organised teams with specialists for some activities in the stages of development. With traditional methods, systems are fully specified, predicted and built through careful and extensive planning [35].

How are Traditional Methods Used

The whole process of software development, according to the traditional methods, begins with the understanding of the requirements and expectations from the customer or end user. After the requirements are clearly understood by the developers, analysis and design of the software begins. The systems development undergoes through a sequence of some fundamental stages such as planning, analysis, design, and implementation [2]. The activities of one stage must be completed before moving to the next stage. Amid all the stages the documents have to pass a quality check, this approach is referred to as a stage-gate model [36]. After the stage-gate model, it is when the document is signed off and the next stage begins.

The traditional methodologies endorse a strict sequence of the quoted stages and it cannot be violated. Hence, the processes have to be fully defined and documented. Stages such as requirements gathering and systems design, the way they are performed in the traditional methodologies helps the team members to broadly gain knowledge about the entire system [37]. It is vital for organisations to keep knowledge within itself because the systems that are developed require maintenance. Therefore, these documents that are compiled and stored empower the employees who later join the organisation to know what the systems are all about hence they may be the same people to maintain such systems.

Who Uses Traditional Methods

People who use the traditional methods are known as IT professionals. Such professionals include systems analyst, database analysts, database administrators, network administrators, webmasters, programmers, vendors, steering committees and other IT professionals [38]. However, the structure of the company also determines how the systems development team is setup. The other IT professionals include project manager, software development manager, software architect, software developer, test manager, test leader, test designer, software tester, quality manager, quality assurance engineer and quality control engineer [20].

Limitations and Challenges of Traditional Development

Due to the formal/sequential pattern of the traditional methodologies, users are expected to give out requirements at the early stages of the project. As a result users may give out incorrect requirements or leave out critical requirements. According to Mujumdar et al., [39], due to uncertainties at the beginning of the project with regards to the requirements the traditional models are unable to accommodate such uncertainties properly. These traditional methods are not most suitable for the development of projects whereby system requirements change regularly, the development schedules have to be shortened because that has a negative impact on quality [40].

The traditional methodologies (also known as plan-driven methodologies) assume that the correct information can be obtained up front [41].

Humans are prone to committing errors since there are uncertainties in developing systems. Things cannot always be done correctly the first time since we are humans. Another limitation of the traditional methodologies is that customers realises the problems of early stages very late when they have to accept the system they requested [36]. Therefore, any change requested late in the development or after sign off of a particular stage, requires additional cost.

Conclusion

Prior to systems development it is vital for the systems development team to have a good understanding of what is required by the customer/user. It is crucial for the systems development team to also understand the type of project they are faced with before they could start working on it. That enables the team to decide on which system development methodology to follow. The understanding of how each methodology is applied by who, how and why is applied helps the system development team to make informed decision. The intention of most organisations when it comes to systems development is to be successful since IT enables them to be efficient and remain competitive to its counterparts. Failure is not an option to organisations and success is what is expected from the systems development team.

References

1. Nelson AC, Teng JTC (2000) Do systems development methodologies and CASE tools decrease stress among systems analysts? Behaviour & Information Technology, 19(4):307-313.
2. Rob MA (2006) Dilemma between the Structured and Object-Oriented Approaches to System Analysis and Design. Journal of Computer Information Systems.
3. Avison DE, Fitzgerald G (2003) Where Now for Development Methodologies? Communication of the ACM, 46(1):79-82.
4. Leau YB, Loo WK, Than WY, Tan SF (2012) Software Development Life Cycle Agile vs Traditional Approaches. International Conference on Information and Network Technology.
5. Matkovic P, Tumbas P (2010) A Comparative Overview of the Evolution of Software Development Models. International Journal of Industrial Engineering and Management, 1(4):163-172.
6. Ayman Al Ahmar M (2010) Rule Based Expert System for Selecting Software Development Methodology. Journal of Theoretical and Applied Information Technology.
7. Nerur S, Mahapatra R, Mangalaraj G (2005) Challenges of Migrating to Agile Methodologies. Communications of the ACM, 48(5):73 -78.
8. Mishra, S. & Weistroffer, H. R. (2008). Issues with Incorporating Regulatory Compliance into Agile Development.
9. Jain R, Chandrasekaran A (2009) Rapid System Development (RSD) Methodologies: Proposing a Selection Framework. Engineering Management Journal, 21(4):30-35.
10. Bassil Y (2012) A Simulation Model for the Waterfall Software Development Life Cycle. International Journal of Engineering & Technology, 2(5).
11. Hedman J, Lind M (2009) Is There Only One Systems Development Life Cycle? Information Systems Development: Challenges in Practice, Theory, and Education, 1:105-115.
12. Jiang M, Jong CJ, Poppell P, Budhathoky, K, Hull R (2009) System Infrastructure Development Life Cycle for Enterprise Computing Systems.
13. Pefkaros K (2008) Using Object-Oriented analysis and Design over Traditional Structured Analysis and Design. International Journal of Business Research, 8(2):219-227.
14. Avison DE, Fitzgerald G (2006) Information Systems Development Methodologies, Techniques & Tools. 4th Ed. The McGraw-Hill Companies.
15. Nasution MFFA, Weistroffer HR (2009) Documentation in Systems Development: A Significant Criterion for Project Success. Paper presented at the Proceedings of the 42nd Hawaii International Conference on System Sciences.
16. Seilheimer SD (2000) Information management during systems development: a model for improvement in productivity. International Journal of Information Management, 20:287-295.
17. Munassar NMA, Govardhan A (2010) A Comparison Between Five Models Of Software Engineering. IJCSI International Journal of Computer Science Issues, 7(5):94-101.
18. Butt A, Hameed S (2011) Success of Spiral Model along with its Development Techniques. Models and methods applied in sciences.

19. Satzinger JW, Jackson RB, Burd SD (2004) Systems Analysis and Design in a Changing World. 3rd ed. USA: Thomson.
20. Mathur S, Malik S (2010) Advancements in the V-Model. International Journal of Computer Applications, 1(12):29-34.
21. Balaji S, Murugaiyan, MS (2012) Waterfall vs V-Model vs Agile: A Comparative Study on SDLC. International Journal of Information Technology and Business Management.
22. Mushtaha A, Tolba R (2008) Integrating V-Model Into The Web Development Process. International Arab Conference on e-Technology-IACET.
23. Skidmore S (2006) The V-Model developing systems. Student Accountant.
24. Lee G, Xia W (2010) Toward Agile: An Integrated Analysis of Quantitative and Qualitative Field Data on Software Development Agility. MIS Quarterly, 34(1):87-114.
25. Choo CH, Lee SP (2008) Towards Persistence Framework-Based Rapid Application Development Toolkit for Web Application Development. Journal of Computer Science, 4(4):290-297.
26. Khan AI, Qurashi RJ, Khan UA (2011) A Comprehensive Study of Commonly Practiced Heavy and Light Weight Software Methodologies. IJCSI International Journal of Computer Science Issues, 8(4:2):441-450.
27. Ge C (2010) Modifying RUP (Rational Unified Process) to Comply with CMM (Capability Maturity Model) Levels 2&3. IEEE.
28. Manzoni LV, Price RT (2003) Identifying Extensions Required by RUP (Rational Unified Process) to Comply with CMM (Capability Maturity Model) Levels 2 and 3. IEEE Transactions on Software engineering, 29(2):181-192.
29. Guo F, Xia B, Xue F (2011) Analysis on Software Processes and Enhancement for RUP.
30. Bergandy J (2008) Work in Progress - Software Engineering Capstone Project with Rational Unified Process. 38th ASEE/IEEE Frontiers in Education Conference.
31. Wu X, Ge C (2010) The Research on Necessity and Plan for Using Extreme Programming in Rational Unified Process.
32. De Barros Paes CE, Hirata CM (2007) RUP Extension for the Development of Secure Systems. International Conference on Information Technology (ITNG'07).
33. Khan ME, Khan F (2012) A Comparative Study of White Box, Black Box and Grey Box Testing Techniques. (IJACSA) International Journal of Advanced Computer Science and Applications, 3(6):12-15.
34. Vinekar V, Slinkman CW, Nerur S (2006) Can Agile and Traditional Systems Development Approaches Coexist? An Ambidextrous View. Information Systems Management.
35. Dyba T, Dingsoyr T (2008) Empirical studies of agile software development: A systematic review. Information and Software Technology.
36. Petersen K, Wohlin C, Baca D (2009) The Waterfall Model in Large-Scale Development. LNBIP, 32:386-400.
37. Uikey N, Susman U, Ramani AK, A Documented Approach in Agile Software Development. International Journal of Software Engineering (IJSE), 2(2):13-22.
38. Shelly GB, Cashman TJ, Vermaat ME (2004) Discovering Computers 2004 A Gateway to Information Web Enhanced. Thomson Course Technology.
39. Mujumdar A, Masiwal G, Chawan PM (2012) Analysis of various Software Process Models. International Journal of Engineering Research and Applications, 2(3):2015-2021.
40. Carlo KM, Estevez E, Fillottrani P (2010) A Quantitative Framework for the Evaluation of Agile Methodologies. JCS&T, 10(2):68-73.
41. Marchesi M, Mannaro K (2008) Adopting Agile ethodologies in Distributed Software Development.