Review of Business & Finance Studies
Vol. 9, No. 1, 2018, pp. 77-84
ISSN: 2150-3338 (print)
ISSN: 2156-8081 (online)

www.theIBFR.com

BEST PRACTICES IN SYSTEMS DEVELOPMENT LIFECYCLE: AN ANALYSES BASED ON THE WATERFALL MODEL

Mitch Kramer

ABSTRACT

This paper discusses best practices for the Systems Development Lifecycle (SDLC) based on the Waterfall Model. Topics covered include an introduction to the Systems Development Lifecycle and the Waterfall Model including advantages and disadvantages. It also discusses the six stages of SDLC 1) Requirements Gathering and Analysis, 2) Systems Development, 3) Systems Implementation and Coding, 4) Testing, 5) Deployment, 6) Systems Operations and Maintenance.

JEL: L86, M15

KEYWORDS: Systems Development Lifecycle, SDLC, Waterfall Model, Requirements Gathering and Analysis, Systems Development, Systems Implementation and Coding, Testing, Deployment, Systems Operation and Maintenance

INTRODUCTION

In this paper, the discussion will be based on best practices and deliverables for the Systems Development Lifecycle; analyses based on the Waterfall Model. I have completed numerous development projects and used different methodologies in the past. From small businesses to Fortune 1000 companies, many use some sort of Systems Development Lifecycle for their projects, whether it was for an Enterprise Resource Planning (ERP) system, Customer Relationship Management (CRM) system, or a custom application. The Global ERP Software Market is expected to garner $41.69 billion by 2020, registering a Compound Annual Growth Rate (CAGR) of 7.2% during the forecast period 2014 to 2020 (Chaudhari & Ghone, 2015). Saying this, there will be many opportunities for using the Systems Development Lifecycle for these projects. The Systems Development Lifecycle is not exclusive to the ERP software market. The current literature discusses the theory behind the Systems Development Lifecycle. While this paper adds to the current body of literature through practical knowledge and experience it also demonstrates best practices and the deliverables for each stage. Herein, I discuss what the Waterfall Model is and the advantages and disadvantages.

As well as data and methodology based on an analyses of the six different stages of the Systems Development Lifecycle 1) Requirements Gathering and Analysis, 2) Systems Development, 3) Systems Implementation and Coding, 4) Testing, 5) Deployment, 6) Systems Operations and Maintenance. Creating, implementing or upgrading a system can be stressful for all Stakeholders. In many cases, these projects can be extremely difficult, but the process can be made easier. All Stakeholders want a smooth project, whether it was for an Enterprise Resource Planning (ERP) system, Customer Relationship Management (CRM) system, or a custom application. There is a process known as Systems Development Lifecycle (SDLC), which can make the process both simpler and less stressful. All six stages must be completed to achieve a positive outcome. The six stages of the Waterfall model are shown in Figure 1 in the Appendix Section. There are different methodologies for the Systems Development Lifecycle, the way systems are developed and placed into production. The most commonly used methods include the following; Waterfall Model; V-Shaped Model, Interactive Model, Big Bang Model, and Agile Model. There are additional models, however; these five are the most commonly used. This paper briefly discusses the Waterfall Model while describing best practices for each stage of the Systems Development Lifecycle.

LITERATURE REVIEW

Meir "Manny" Lehman discussed one of the first approaches to the Systems Development Lifecycle in 1969. "The basic approach recognizes the futility of separating design, evaluation, and documentation processes in software-system design. An expanding model seeded by a formal definition of the system, which provides a first, executable, functional model, structures the design process. It is tested and further expanded through a sequence of models that develop an increasing amount of function and an increasing amount of detail as to how that function is to be executed. Ultimately, the model becomes the system."
In addition, Winston W. Royce first described the Systems Development Lifecycle based on the Waterfall Model in 1970. He did not actually state that it was called the Waterfall Model; however, he implies it by showing each step as preceding and succeeding stages. He discusses the stages and shows the importance of doing it systematically. "The Systems Development Lifecycle (SDLC) is perceived as the timeframe that spans from the development of a new system to its eventual retirement. It is a process that starts with the emergence of an idea, goes through its implementation, and ends with its termination, moving across all the intermediate stages in which its viability and usability are prioritized." (Pedro Isaias and Tomayess Issa 2015) Projects need to be properly planned and executed. There have been cases when the Systems Development Lifecycle was not planned and executed properly.

A few instances of this include the implementation of Enterprise Resource Planning (ERP) systems, which essentially all Fortune 1000 companies rely on for their routine transaction processing. Contemporary ERP systems such as SAP take years to implement and cost several millions of dollars when properly executed. Mistakes in one or more stages of the SDLC process can be devastating for companies. Hershey's was in the midst of its ERP implementation in the summer of 1999. Unfortunately, due to problems in the configuration of Hershey's SAP system; it was unable to deliver Hershey's Kisses to stores during Halloween that year. Hershey has had a net result of $100 million in lost sales; their stock fell by 8 percent consequently. Similar difficulties were faced by Nike, which spent $400 million to upgrade its supply chain and ERP systems. Nike also lost about $100 million in sales and the stock price fell 20 percent. Even college students have been negatively affected by problems in systems implementation. Students at the University of Massachusetts, Indiana University, and Stanford University experienced problems with their course schedules and even their financial aid checks due to glitches in the newly implemented ERP systems at these schools. These accounts highlight the meaning of employing a planned approach to systems development, paying close attention to detail at every step. (Uday S. Murthy 2000) "The Waterfall Model is the oldest and most straightforward of the structured methodologies. The waterfall process originated during the 1960s in firms building large-scale software systems for the U.S. defense and space industry as well as for commercial applications. These companies worked on multi-year projects and designed software for large computer (mainframe and minicomputer) systems that evolved relatively slowly.

They modeled the waterfall process after hardware design projects, where engineers could more easily (though not always completely) predict how pieces of a system would interact" (Michael A. Cusumano and Stanley Smith 1995) This method consists of completing a stage then moving on to the next stage. Generally, projects do not go back a stage after completing the current one. Projects continue to move forward. Each stage depends on information from the previous stage. This model allows the opportunity to return to an earlier stage, however, going back to a previous stage involves costly rewrites for both documentation and developing the application. If delays happen, then the entire project and timeline will change. The Waterfall Model is simple to understand and manage throughout the Systems Development Lifecycle. Waterfall Model – Advantages The Waterfall Model is simple, methodical, and easy to understand and to implement. It works well and yields the correct results. It is extensively used and recognized in the industry. Since the stages are rigid and precise, each stage is executed one at a time, making it easy to maintain. The start and finish criteria are well defined, alleviating stress. It ensures quality and identifies deliverables and milestones. The focus is on documentation.

“The major advantage of the Waterfall Model for the SDLC is that it provides a structure for organizing and controlling a software development project. The single most important methodological need in this approach, however, is to identify user requirements accurately.” (Andrew P. Sage and James D. Palmer 1990) Waterfall Model – Disadvantages A major defect with this method is that you cannot adopt major changes in requirements in the middle of the project. If the application has moved on to the Testing stage and a requirement has been changed (at the direction of the project sponsor), it becomes difficult to go back and change it. Another issue can be the lengthy delivery of the final product. One cause might be there was not a prototype previously available for comparison to the application being developed. Without a prototype, there is a higher risk of error. Since the testing is performed at a later stage, it does not allow for identifying the challenges and risks in an earlier stage. This makes the risk mitigation strategy difficult to predict. Making changes to documents can be costly, as all stages need to be changed. This happens with major and minor issues. If it were known before beginning the project that there might be a number of changes, my recommendation would be to use a different model, as the Waterfall Model does not handle change well or cost-effectively.

“Disadvantages associated with the Waterfall Model include problems that occur if there is no iteration and feedback among stages, beginning with the need to define the system requirements specifications accurately. Unless there is iteration and feedback, there may be no way to improve initial imperfections in one of the later stages. Realistic life-cycle development processes are, therefore, iterative and interactive, and a company's software development process needs to accommodate this fact.” (Andrew P. Sage and James D. Palmer 1990) With the Waterfall Model, it is important to get the approved sign-offs for all deliverables of each stage. The bigger and more complex projects-today are moving towards Agile and Sprint Models. I do notice that the Waterfall Model still holds up well for smaller projects like system process improvements or if requirements are straightforward and testable. The Waterfall Model yields the best results for these scenarios. This paper will discuss the necessary steps based on a process improvement for a custom application.

DATA AND METHODOLOGY

After the thought process has occurred and the process improvements have been identified, the project sponsor defines the objectives, scope, and purpose. A solution needs a scope document to get started. There are additional documents that can be utilized if the solution, scope and size call for it. These include the following: Systems Boundary, Cost-Benefit Analysis, Risk Management Plan and Feasibility Study. For larger projects, some companies get Request for Proposals (RFP) if they do not have either the knowledge or workforce to implement a particular solution. Once the process has been officially approved, the Project Manager develops a Project Plan that includes Project Charter, Project Scope, Project Schedule and Project Timeline. This plan helps define the basis for acquiring the necessary resources to achieve the solution. The Project Manager needs to have all six stages included in the Project Plan. Below is a look at all six stages of the Systems Development Lifecycle: analyses based on the Waterfall Model. Each stage shows what should happen during each stage and best practices.

Requirements Gathering and Analysis

The first stage: Requirements Gathering and Analysis. The Project Manager begins planning the project. The requirements of the organization are gathered, and analysis begins to determine the approach of the project, deliverables, and anticipated outcomes; these are the functional and non-functional requirements.

The Business Analyst ensures the business and functional requirements meet the Stakeholder's expectations. Business requirements are gathered to determine who will use the software and how they will use it. The Business Analyst then performs a needs analysis. Part of the needs analysis will include collaboration with end users of the system. They are key at this stage of the process. They know which results they expect and how to identify them in the system. They also have ideas about which results and information they would like to get from the system. In this stage, the users are providing best practices, their needs and wants. Capture all the requirements during this stage, either with the use of flowcharts, user or test cases, and current documentation.

"The evolution of software that satisfies its user expectations is a necessary goal of a successful software development organization. To achieve this goal, software engineering practices must be applied throughout the evolution of the software product. Most of these software engineering practices attempt to create and modify software in a manner that maximizes the probability of satisfying its user expectations. Other practices, addressed in this module, actually attempt to insure that the product will meet these user expectations." (James S. Collofello, 1998) Some best practices of needs analysis are accomplished with the following: meeting with Stakeholders as a group or one-on-one to find out their requirements, discussing their current challenges and retrieving any documents containing current processes. If there are no documents, process mapping should occur. Work with the group or individual to create a document outlining each process. Record a complete understanding of their needs, for example, "must haves" and "wish list" items. When gathering information for the documents, use visuals like screenshots, use cases, flowcharts, and diagrams. The documents should include a purpose, current processes, details of expected accomplishments and expectations.

Another best practice is to brainstorm and perform walkthroughs to understand the requirements. Do a Feasibility Test to ensure whether the requirements are testable or not. Remember that the process improvements help the business increase quality and efficiency. Process improvements can include the following: a better process to complete a task, the ability to retrieve data to improve efficiency by creating an automated report, the creation of a new application, or the creation of an interface to upload transactions from another system into the current system. Functional requirements should describe how the business would like a software system to work, or the steps they currently take to perform a manual process. With the analysis stage, the requirements are to collect and analyze the information provided. Projects are not successful when requirements are missing or Developers miss an important piece of information. This happens many times when there is a lack of communication between Stakeholders. The expected outcome will not happen when the Architects or Developers do not truly understand the issue. This is the most important step in the Systems Development Lifecycle for the Waterfall Model. If the project fails, it is usually because of this step not being fully utilized.

Deliverables: RUD (Requirements Understanding Document), Functional and Business Requirements
Systems Development

The second stage: Systems Development. In this stage, the detailed requirements are converted into completed detailed systems design. This design describes the key components and the interface of those components. It focuses on how to deliver the required functionality (must-haves and wish-list items) to the system. The Business Architects use the requirements gathered from the first stage to produce several designs of the product. The Business Architects know if a Developer can deliver the designs. Various internal Stakeholders review each design in a design document. Stakeholders approve the best design. Once selected the Architect creates the design, captures the hardware/software requirements, and documents the designs. In other words, he or she creates the blueprint for the design with the necessary specifications for the hardware, software, people and data resources. He or she also add additional information for the coding and debugging of the application.

Best practices include having high-level design and low-level design documents to describe the characteristics of the system and operations in detail. Design elements should be detailed and describe the desired characteristics of the software. These include functional hierarchy diagrams, screen format maps, tables, business rules; business processes diagrams, pseudo-code and an entity relationship diagram of the data dictionary. The design elements assist the Developers with the development of the software with limited additional input required. The result of this stage is to describe how the application needs to be created. Be sure to include the sign-off of all documents from the Stakeholders; this is another best practice.

Deliverables: HLD (High-Level Design Document) & LLD (Low-Level Design Document)

Systems Implementation and Coding

The third stage: Systems Implementation and Coding. Convert the Requirements Gathering and Analysis to the System Development into the Systems Implementation and Coding. Thorough and detailed documentation and mapping are necessary for the Developers to implement the design and develop the application. This stage can be the longest part of the Systems Development Lifecycle process. This is where the Developers create the code from the two other stages and from the approved design. Developers can use different tools such as compilers, interpreters, and debuggers to generate code. Different programming languages can be used to code the application. Some examples of these include C++, Java, JavaScript, SQL, Pascal, ProvideX, PHP, VBA, and XML. The best programming language, for each project, depends on the Developer and on the application that is being developed. This paper is specific to systems implementation and coding of a custom application based on Waterfall Model in the Systems Development Lifecycle. A future paper will explore the Software Development Lifecycle based on the implementation of an Enterprise Resource Planning (ERP) system. A best practice that a Developer should employ is to verify that the application works as per the design specifications.

Deliverables: Application, Unit Test cases and Results from the Developer

Testing

The fourth stage: Testing. After the application has been developed, testing can begin. The application will be tested based on the requirements to ensure that the code works according to the specifications. Integration testing is needed to confirm that the application works as expected. This is another best practice that should be followed. The Business Architect or Analyst who provided the design specs should be the one to verify the changes to the application. The Business Architect or Analyst is looking to confirm that the application gives the expected and intended results. This verification process confirms all applications are working as designed. If expectations have not been met, then it will need to go back to the Developer so any issues can be corrected.

There are many methods of testing including prototype, workflow diagram, use test cases, a story map, a stress test and module unit testing. When testing, track progress, report testing activities and report all issues that arise before going to the next stage. If issues do arise, make notes and take screenshots so the Developer and Business Architect or Analyst can review these aspects in detail. State what the issue is and restate what the expectations are. Another important step in this phase is to create an issues log. The details in the log should include Creation Date, Issue Number, Priority, Description and any additional details that provide further information. Once an issue has been resolved, log the issues as closed and the date completed. Confirm no critical and high-level issues exist before going to deployment. The issues log needs to be shared with the Project Manager and all Stakeholders throughout the process.

Deliverables: Test Cases, Test Reports, Defect Reports, Issues Log

Deployment

The fifth stage: Deployment. After the testing stage, when the application has been fully tested and verified, the application is now ready for deployment. A best practice utilizes a test system to load the application before placing it into production. With this, an exit strategy is needed in case something goes wrong with the data conversion or the application. This may affect other systems and cause issues. Have a backup or be able to back out the application. The end user needs to ensure that the product actually meets their needs and that the specifications were correct in the first place. In other words, demonstrate that the product fulfills its intended use when placed in its intended environment. This would be the validation part of the application. This is called User Acceptance Testing (UAT). If the users cannot validate, then the back out strategy needs to take place. This will alleviate some of the pressure when deploying the application to the production environment. Confirm the environment setup matches the testing stage. This will help determine if it is an issue in the environment or if there is something wrong with the application itself.

Part of the deployment includes training the users on the new application. User manuals will be useful to assist with training. The user can utilize the manual for further clarification and future reference on how to use the application. An issues log is also needed at this stage. It is acceptable to have outstanding issues; however, none of the issues should be critical or high-level defects. If there are minor issues, confirm they will be completed in the sixth stage, which is the Systems Operations and Maintenance Stage. Once the application has been approved, in the Stakeholder's test environment, place the application in the production environment. Perform a stability check in the production environment after the application is deployed to ensure the application does not break. Once again, have the Stakeholders validate that the system works. It is entirely possible that a product passes when verified but fails when validated.

Deliverables: User Manual, Environment Definition/Specification, Issues Log

Systems Operation and Maintenance

The final stage: System Operations and Maintenance. This is when the process improvement has come to together and the application is ready for use. The Stakeholder may experience technical issues, and maintenance of the software is conducted at this time. There will be questions from the end users; make sure that there are technical and procedural experts available to answer any additional questions. Keep an issues log open for any outstanding issues. If there were any non-critical defects remaining, this is the time to correct them. Ensure these issues are not enhancements that need the approval of the Project Owner. User manuals will be very beneficial at this stage as a reference guide for the users. Resolving all issues on the open issues log is another best practice. If an issue has been corrected in the production environment, ensure the code has been updated in both the development environment and coded environment. The application will evolve over time, so it is important to incorporate these changes to all environments.

Deliverables: User Manual, List of Production Tickets (Resolved), List of New Features Implemented

RESULTS

With following the six stages of the System Development Lifecycle, the results will be a smooth transition into the new custom application that has developed. All deliverables from the six stages include the following for each stage, 1) RUD (Requirements Understanding Document), Functional and Business Requirements. 2) HLD (High-Level Design Document) & LLD (Low-Level Design Document). 3) Application, Unit Test cases and Results from the Developer. 4) Test Cases, Test Reports, Defect Reports, Issues Log. 5) User Manual, Environment Definition/Specification, Issues Log. 6) User Manual, List of Production Tickets (Resolved), List of New Features Implemented.

A Path Forward

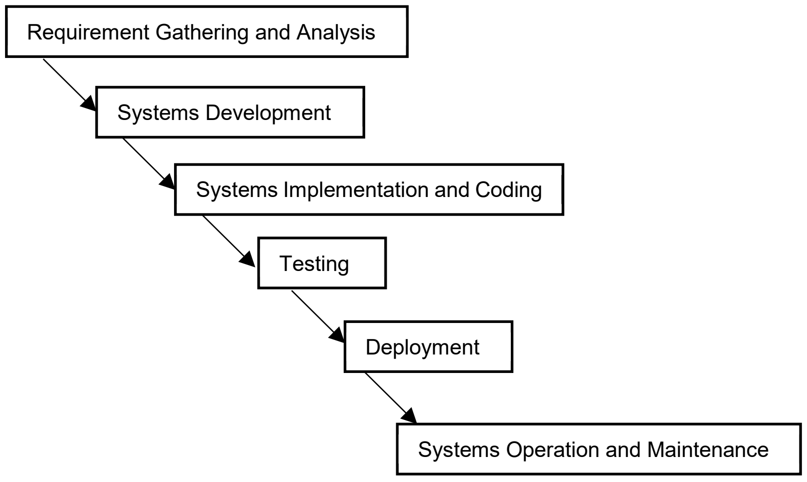
The Systems Development Lifecycle will continually be part of the implementation of any type of system. The methodologies that assist with implementations of all projects have evolved over time. In the past, there have been additional methodologies; however, I do think the Waterfall Model can withstand time. As discussed the Waterfall Model is simple, methodical, and easy to understand and to implement. It works well and yields the correct results. As new methodologies develop, academics and professionals need to work together through teaching the theory and using real-world experience to help educate students and young professionals. This will aid the learning process and help facilitate the next methodology.

CONCLUDING COMMENTS

This paper analyzes the history, theory and best practices for the Systems Development Lifecycle based on the Waterfall Model. There are many actions that must be executed and best practices that all parties should follow. All the requirements in one stage must be completed before moving on to the next stage, and ensure all sign-offs are received from all Stakeholders. This will show that the milestones for each stage happened. Remember to understand the true issue or business need (business requirement). This will ensure that you are delivering the highest value to your Stakeholders. Once you understand the business needs, start determining the best way to approach building a solution. Have a precise implementation strategy with thorough critical success factors: develop a comprehensive project design and plan. Collaborate with all users: create small working groups of key users to outline goals and follow progress. Communicate promptly: organize effective communication at all stages. Develop and conduct user training: Add additional training as required, this includes discussions on processes and systems. Confirm users know the best way to communicate during and after the implementation. It will be interesting to see the result of academics and professionals working together to develop a new methodology through the discussion of theory and practical experience. This concludes the best practices for the System Development Lifecycle based on the Waterfall Model.

APPENDIX

Figure 1: Systems Development Lifecycle


Requirement Gathering and Analysis
Systems Development
Systems Implementation and Coding
Testing
Deployment
Systems Operation and Maintenance

REFERENCES

Chaudhari, Shrikant, and Akshay Ghone. (2015). "ERP software market by deployment (on-premise deployment and cloud deployment) and function (finance, human resource, supply chain and others) global opportunity analysis and industry forecast, 2013 - 2020. ERP Software Market," *Allied Market Research* March 2015.

Meir "Manny" Lehman (1969) "The Programming Process" *Program Evolution—Processes of Software Change, Academic Press* reprinted in 1985

Dr. Winston W. Royce (1970) "Managing the Development of Large Software Systems" *Proceeding, IEEE WESCON The Institute of Electrical and Electronics Engineers,* August 1970

Pedro Isaias and Tomayess Issa (2015) "High Level Models and Methodologies for Information Systems" *ISBN 978-1-4614-9253-5 Springer*

Uday S. Murthy (2000) "Advanced Systems Analysis & Design" *CyberText Publishing, Inc.*
Michael A. Cusumano and Stanley Smith, (1995) "Beyond the Waterfall: Software Development at Microsoft" *MIT Sloan School of Management, International Business Machines,* August 16, 1995

Andrew P. Sage and James D. Palmer (1990) "Software Systems Engineering" *ISBN 978-0471617587 John Wiley & Sons*

James S. Collofello, (1998) "Introduction to Software Verification and Validation" *SEI Curriculum Module SEI-CM-13-1.1, Carnegie Mellon University Software Engineering Institute*

ACKNOWLEDGEMENT

I would like to thank the journal editors, Terrance Jalbert and Mercedes Jalbert, two anonymous reviewers, for their insightful comments.

BIOGRAPHY

Mitch Kramer's experience includes six years as a Senior Financial Systems Analyst. Additionally, he has worked for ten years as a Business Analyst/Project Manager with different ERP Systems. His research paper appears in Review of Business and Finance Studies. His educational background includes a Bachelor of Commerce in Accountancy and a Certificate in Management. He lives, with his wife, Michelle, in Florida. My email address is investwithmitch@gmail.com