# **System Development Life Cycle Research Paper**

## University of Maryland University College

## IFSM 301 6380 – 2178

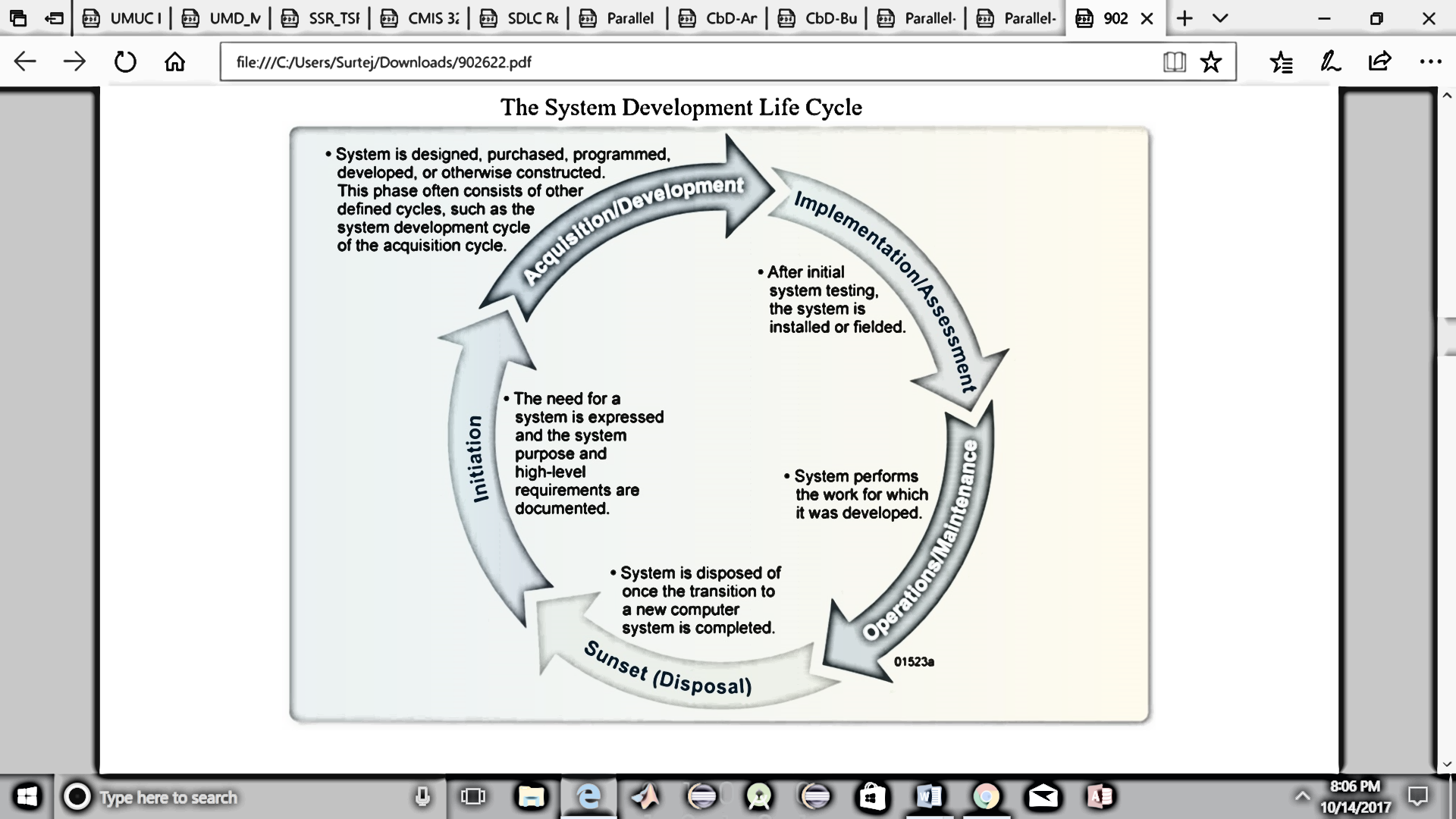
## October 15, 2017

## Surtej Sarin

## Introduction

The System Development Lifecycle (SDLC) is a prescriptive framework for planning, developing, implementing, enhancing, and testing software and information systems. According to the National Institute of Standards and Technology (NIST), the SDLC comprises of five phases: Initiation, Development/Acquisition, Implementation, Operations/Maintenance, and Disposal, as shown in figure 1 (Radack, 2009). Software development teams utilize SDLC descriptive models, such as: waterfall, spiral, iterative and incremental, agile, and evolutionary prototyping to support their development process and deliver projects. It is necessary to consider which model to apply, in relation to the project; “for example, the waterfall model may be the best model to use when developing an enterprise relational database but it may not be the optimum model when developing a web-based application” (Ruparelia, 2010). The following paper delivers an in-depth description and comparison of two SDLC models i.e. Waterfall and Spiral.

# **Figure 1: SDLC**

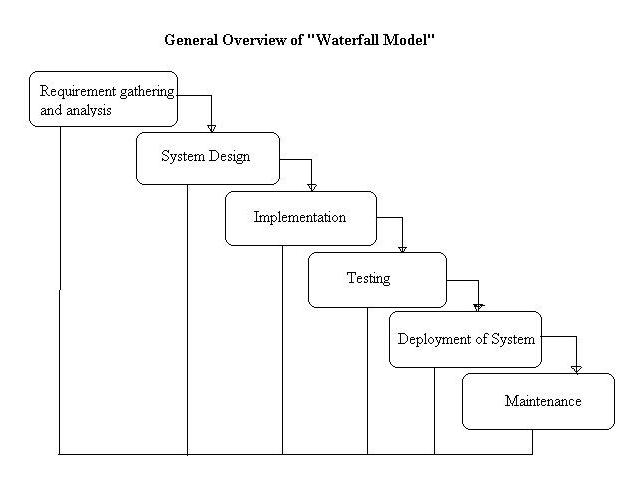
**

*Figure 1 shows the stages of the SDLC lifecycle (Radack, 2009).*

## Waterfall Model

The waterfall or cascade model is a linear-sequential process and top-down approach, in which each phase is completed before subsequent phases. The original model was first documented by H.D. Benington, who recommended that software be developed in stages: operational analysis, operational specification, design and coding specifications, development, testing, deployment, and evaluation (Benington, 1956). This model was modified by Winston W. Royce, who improved the process flow by allowing stages to overlap and revisit previous stages; however, Benington’s model was re-launched in 1983 and is widely acknowledged today. The waterfall model is the oldest and simplest SDLC approach is traditionally applied to SDLC projects that have well-defined requirements, short delivery time, and sufficient resources. In the waterfall model each step or phase is thoroughly documented and completed within a specified time, without any overlapping. The model comprises of the following phases: requirements, system design, implementation, testing, deployment, and maintenance, as shown in figure 2. In the *requirements phase*, general requirements and purpose of the application are documented. In the system design phase, business logic and system components are analyzed and technical requirements are defined for the application. In the *implementation phase*, the application is developed based on the previous phase requirements. Once the code has been developed, in the *testing phase*, the testing team, including quality assurance testers and security testers, search for unresolved problems in the application and report findings back to the development team for reevaluation. In the *deployment phase*, the application is released to the client. In the *maintenance phase*, the IT application team will support, update, and maintain the application in the environment.

# **Figure 2: Waterfall Model**



*Figure 2 shows the phases of the waterfall model (What is Waterfall model- advantages, disadvantages and when to use it?, n.d.)*

### Advantages

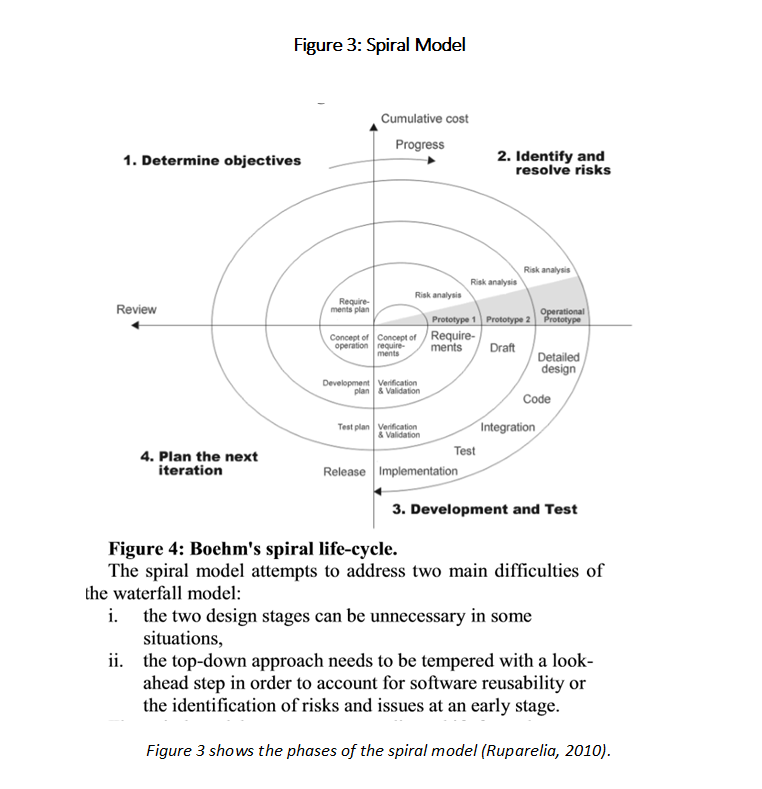
The advantages of the waterfall model are derived from its controlled structure and development process. Requirements are defined before starting development, which makes it simple to manage and allows for early design alterations. Moreover, the model has a linear structure, which allows teams to follow milestones, and works well for small projects. Finally, each phase is completed within a deadline and documented, which improves quality of the end-product.

### Disadvantages

However, the waterfall model also has certain disadvantages, attributed to its ridged structure. This model can represent a badly structured system, as it does not allow for revisions, and is especially problematic when one phase is not completed and defects are found late in the development cycle. Consequently, the client does not have any working software until all phases of the SDLC are complete and this creates high risk and costs. The waterfall model also creates low visibility for clients and does not allow them to change requirements, due to an ineffective change management process. Additionally, the waterfall model is not suitable for complex and long-term projects.

## Spiral Model

The spiral model was first introduced, by Barry Boehm, as an adaption of the systematic waterfall model and iterative development model (Boehm, 1986). This model represents is a risk-driven approach that can be applied to long-term and high-risk projects, with complex requirements. The spiral model includes four phases: (1) Determine objectives, (2) Evaluate alternatives, and identify and resolve risks, (3) Develop and test, (4) Plan the next iteration, as shown in figure 3, and iteratively moves through each phase in a *spiral* (Ruparelia, 2010). The spiral model begins in *the determine objectives phase*, in which the team defines business and system requirements. It passes this phase in subsequent cycles covering additional system and client requirements, which involves communication and customer feedback. In the *identify and resolve risks phase*, the development team evaluates risks that can affect the project. In subsequent cycles, it makes key policies to mitigate these risks and, finally, creates an operational prototype. “Risk management is used to determine the amount of time and effort to be expended for all activities during the cycle, such as planning, project management, configuration management, quality assurance, formal verification and testing. Hence, risk management is used as a tool to control the costs of each cycle” (Ruparelia, 2010). In the *develop and test phase*, a concept of requirements, also referred to as a Proof of Concept (POF), is created by the software team to define conceptual requirements for developers. Additional phases address specific software requirements, product design, developing, and testing i.e. unit, integration, and acceptance. Eventually, a functional program is developed and sent to clients for feedback. In the *plan the next iteration phase*, verification and validation takes place after the completion of every cycle to ensure stakeholders agree with the plan and progress and resources are allocated for subsequent cycles. “An oft-quoted idiom describing the philosophy underlying the spiral method is start small, think big” (Ruparelia, 2010).



### Advantages

The advantages of the spiral model are derived from its ability to constantly capture and adapt to new requirements, irrespective of previous cycles. This model allows for client feedback throughout the SDLC process and updates and accommodations can be made to the project to match any new requirements. Additionally, it creates prototypes throughout the process so clients can view builds much earlier. In this model, it is “easier to manage risk because risky pieces are identified and handled during its iteration. Spiral model is good for large and mission critical projects where high amount of risk analysis is required” (Mishra and Dubey, 2013).

### Disadvantages

According to the Association for Computing Machinery's Special Interest Group on Software Engineering, “The main difficulty of the spiral is that it requires very adaptive project management, quite flexible contract mechanisms between the stakeholders and between each cycle of the spiral, and it relies heavily on the systems designers’ ability to identify risks correctly for the forthcoming cycle” (Ruparelia, 2010). Moreover, this model is costly, as the spiral can continue cycling without an actual project release. Additionally, the spiral model not suitable for simple and small projects because of its complexity.

## Comparison of the Two Models

As described in previous sections, the SDLC models: waterfall and spiral, have unique advantages and disadvantages in the development life-cycle. The waterfall model is a linear-sequential process that has defined project requirements and documentation. The spiral model is an iterative framework that allows projects to adapt to changing requirements and mitigate risks. “The spiral model represents a paradigm shift from the waterfall’s specification driven approach to a risk-driven one” (Ruparelia, 2010). This section will compare these two models based on ease of use, applicability for complex projects, applicability to projects of all sizes, customer involvement, and system interfaces.

1. **Ease of use (Waterfall)**

The waterfall model, as compared to the spiral model, is easier for software teams to use due to two main advantages: documentation and deadlines. Documentation is heavily emphasized in this model to guide team-members through each phase in the process. Comprehensive details allow easy management i.e. software teams and new team members can adopt this model without prior knowledge of a control process, such as agile scrum. Establishing deadline and milestones helps the team determine project progress. Moreover, in the waterfall model, requirements are defined ahead of time and phases are followed in a strict order i.e. the next phase is dependent on the completion of the previous phase, which creates a predictable workflow. “If the requirements are known before hand and well understood and we want full control over the project at all time, then we can use waterfall model” (Mishra and Dubey, 2013). In contrast, the spiral method requires a greater commitment on the software teams part for monitoring a complex SDLC design process and risk assessment for each and every cycle.

1. **Applicability for complex projects (Spiral)**

The spiral model, as compared to the waterfall model, applies to complex projects. The spiral model is adaptive and each iteration allows for design changes: “development [teams] adds functionality for additional requirements in ever-increasing “spirals” until the application is ready for the Installation and Maintenance phase” (Mishra and Dubey, 2013). In contrast, the waterfall model is slow, rigid, and “not very much useful when the project requirements are dynamic in nature” (Mishra and Dubey, 2013).

1. **Applicability to projects of all sizes (Waterfall)**

The waterfall model, as compared to the spiral model, directly applies to projects of all sizes. The waterfall model is a traditional SDLC model and has generally been applied to small projects; however, “Large enterprise projects generally require large number of project teams to work on clearly defined deliverables. The scale of the deliverables is proportional to the size of the project team assigned to do it. Thus, larger project teams are assigned larger set of deliverables which need to be clearly defined. With this kind of scenario, long iterations or waterfall would be more ideal” (Which Life Cycle Is Best for Your Project, n.d.). Thus, it can be applied to any sized project. In contrast, the spiral model is primarily used for larger sized projects due to the complexity and cost of the SDLC model.

1. **Encourages/allows customer involvement (Spiral)**

The spiral model, as compared to the waterfall model, encourages and allows customer involvement in the SDLC life-cycle process. Clients have better visibility of the project all throughout the process and are made aware of the progress of the software project during each cycle. As mentioned earlier, in the *determine objectives phase* of the spiral model involves continuous communication and feedback with clients, who are allowed to update requirements. Moreover, this model involves prototyping, in which clients can see the software in the development process. In contrast, the waterfall model does not encourage client involvement during the process and the project may not reflect the needs of the customer.

1. **Interfaces with other systems or development projects (Spiral)**

The spiral model, as compared to the waterfall model, interfaces with other systems and development projects. This model is flexible and can be applied to many systems, as the design requirements can change through each iteration. The waterfall model is ridged i.e. hardware and software systems cannot be changed once the requirements are defined.

|  |  |  |
| --- | --- | --- |
| **Criterion** | **Waterfall Model** | **Spiral Model** |
| Ease of use | **✓** |  |
| Applicability for complex projects |  | **✓** |
| Applicability to projects of all sizes | **✓** |  |
| Encourages/allows customer involvement |  | **✓** |
| Interfaces with other systems or development projects |  | **✓** |

## Conclusion

The following paper provided a description and comparison of two SDLC models i.e. Waterfall and Spiral. From detailed research and analysis, it is clear that the waterfall and spiral SDLC models are distinct in their processes and uses. The waterfall model is inexpensive, simple, ridged, while, in stark contrast, the spiral model is expensive, complex, and flexible. The waterfall model is a linear-sequential process that has defined project requirements and documentation. It is easy to use and applies to projects of all sizes. The spiral model is an iterative framework that allows projects to adapt to changing requirements and mitigate risks. It is applicable to complex projects, encourages customer involvement, and interfaces with other systems and development projects. In conclusion, “Selecting the correct life cycle model is extremely important in a software industry as the software has to be delivered within the time deadline & should also have the desired quality” (Mishra and Dubey, 2013).

## References

Benington, H.D. (1956): Production of large computer programs. In Proceedings, ONR Symposium on Advanced Programming Methods for Digital Computers, June 1956, pp 15-27.

Boehm, Barry W. (1986): A spiral model of software development and enhancement. In ACM SigSoft Software Engineering Notes, Vol. II, No. 4, 1986, pp 22-42.

Mishra, A., and Dubey, D. (2013). A comparative study of different software development life cycle models in different scenarios. International Journal of Advance research in computer science and management studies.

Radack, S. M. (2009, April 29). *The System Development Life Cycle (SDLC)*. Doc. No. NIST SP 800-64. Retrieved October 14, 2017, from <http://ws680.nist.gov/publication/get_pdf.cfm?pub_id=902622>

Royce, Winston W. (1970): Managing the development of large software systems. In Proceedings, IEEE Wescon, August 1970, pp 1-9.

Ruparelia, N. B. (2010) *Software Development Lifecycle Models*. Hewlett-Packard Enterprise Services. In ACM SigSoft Software Engineering Notes, Vol. XXXV, No. 3, 2010. DOI: 10.1145/1764810.1764814. Retrieved October 14, 2017, from <http://doi.acm.org/10.1145/1764810.1764814>

What is Waterfall model- advantages, disadvantages and when to use it? (n.d.). Retrieved October 15, 2017, from <http://istqbexamcertification.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/>

Which Life Cycle Is Best for Your Project? (n.d.). Retrieved October 15, 2017, from <https://www.projectsmart.co.uk/which-life-cycle-is-best-for-your-project.php>