International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 04 | April - 2023 Impact Factor: 8.176 ISSN: 2582-3930

# CHANGING SECURITY WITH EVERY SPIRAL

**Sakshi Modi[1], Dhamini R Nijgal[2]**

*1Student, CHRIST (Deemed to be University), Pune, Lavasa*

*2 Assistant Professor, CHRIST (Deemed to be University), Pune, Lavasa*

---

**ABSTRACT**: The Microsoft SDL integrates security and privacy considerations at every stage of development, assisting programmers in creating highly secure software while also addressing security compliance needs and lowering development costs ("[1]"). The software model: Spiral is crucial to this lifecycle's perspective. Microsoft has proposed the Microsoft Security Development Lifecycle (Microsoft SDL), a software development method based on the spiral model, to assist developers in producing software or applications while lowering security concerns, addressing security flaws, and even lowering development and maintenance costs. Training, requirements gathering, design, implementation, testing, deployment, and reviews are the seven phases that make up the process which is further explained in twelve different methodologies of the lifecycle model created by them.

***KEYWORDS:*** Microsoft, Software Engineering, Spiral Mode.

## 1. INTRODUCTION

One of the earliest and most versatile forms of the software development lifecycle (SDLC), the spiral model allows for well-planned, swift, and iterative development. The software development lifecycle (SDLC) creates high-quality, low-cost software. It combines the prototyping method and the linear development approach. More prominence is given to risk analysis under this model. This concept mostly applies to complex, large-scale projects where risk is significant. Every iteration begins with planning and concludes with a client evaluation of the final result. Microsoft employs an amalgamation of the waterfall model with an iterative nature. Microsoft is aware that the process of creating and launching the product would be tough and fraught with risk. When the current version of the product is available, the team is equipped to release the new version. The company prefers using the spiral model to create the product iteratively. They may make one version of the product available to the public before beginning to work on a new version that incorporates updates and improvements from the first (taking into consideration the problems faced by the end-users). For instance, after releasing Windows 8, Microsoft modified it based on users' feedback before releasing the next version (Windows 8.1). Windows Vista and Gantt charts are other leading examples of the model.

**Figure 1: Iterative Waterfall SDLC model to ensure security and analyze risk**

## 2. ITERATIVE WATERFALL MODEL

The linear development approach is shown using the waterfall model where the team finishes one stage before the next stage starts. During the requirement stage, the scope of the project is pre-defined in a strict and organized manner. Even after the pre-requisite has been specified the model lacks clarity. This gets eliminated with the concept of an iterative model. The team starts specifying and developing only the part of the software product which is required. Gradually, the product is brought to life with iterations carried out in segments that have set deadlines and tasks. As a result, dealing with risk is simplified at any stage, which reduces the overall likelihood of dealing with risk in the future.

## 3. PHASES OF SPIRAL MODEL

The architecture of the spiral model is designed to undergo four phases. These phases are:

1. Planning
2. Risk Analysis

3. Designing and Development
4. Evaluation

Each spiral speaks about the four phases as depicted in Figure 1.

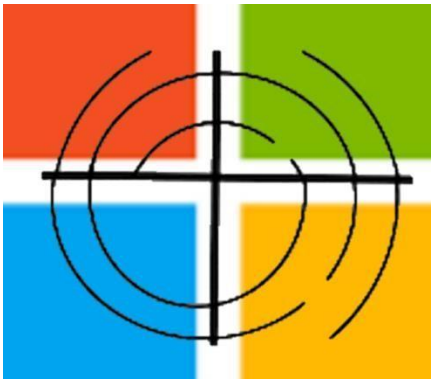

***Figure 2: Microsoft's Spiral***

## *I. PLANNING*

The group evaluates the requirement at this stage to determine their viability. The planning process, tasks, resource definition, deadlines, and the gathering of additional project-related data are all included in this phase. Estimating costs and creating an iteration timeline are also a part of the planning step. As soon as the planning is complete, the team moves on to step two, risk analysis.

## *II. RISK ANALYSIS*

In this phase, the team considers the strengths and potential shortcomings of the project. They settle on various approaches to tackle the problem and for the solution, the project prototype is displayed. Technical and managerial risks are also taken into account during the risk analysis process.

## *III. DESIGNING AND DEVELOPMENT*

This phase's execution is handled by the engineers and the developers. After the final planning and analyzing the risk, software coding, internal testing, and final deployment of the project are done during this stage.

## *IV. EVALUATION*

This is the final stage of the project presentation to the client. The team goes through each of these stages-development, prototyping, or testing to get feedback from the client. The outcome is repeatedly examined during the engineering phase of the project in light of the clients revert.

## 4. MICROSOFT SECURITY DEVELOPMENT LIFECYCLE (MICROSOFT'S SDL)

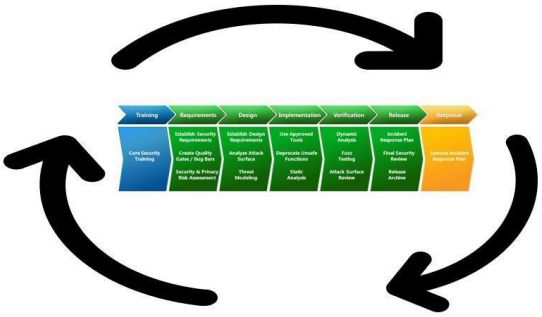


***Figure 3: The Iterative Model***

The four phases of the SDLC model are followed by seven components, five core and two supporting as referred from Microsoft. The five core phases are requirements, design, implementation, verification, and release ("[3]"). To establish all security and privacy requirements are properly addressed, each of these phases includes compulsory checks and approvals. To make sure the core phases are implemented correctly and software is kept secure after deployment, the two supporting security activities, training and response, are carried out before and after the core phases, respectively. These components are further structured into twelve methodologies in SDL:

a) Provide Training
b) Define Security Requirements
c) Define Metrics and Compliance
d) Reporting
e) Perform Threat Modelling
f) Establish Design Requirements
g) Define and Use Cryptography
h) Standards
i) Manage the Security Risk of Using
j) Third- Party Components
k) Use Approved Tools
l) Perform Static Analysis Security
m) Testing (SAST)
n) Perform Dynamic Analysis Security
o) Testing (DAST)
p) Perform Penetration Testing
q) Establish a Standard Incident Response Process [("[2]")

The training phase is crucial because SDL implementation is believed to require practice. Secure design, threat modeling, secure code, security testing, and privacy policies are among the aspects covered in this phase. On the other hand, the construction of security and privacy that end-users require is part of the requirements phase. The second phase includes making sturdy gates/

bug bars and conducting security and privacy risk analyses.

In the third step, design, security, and privacy factors are taken into account to help minimize the risk of public backlash. The application of a structured system to deal with threat scenarios throughout the design process will be made possible by the use of threat modelling, attack surface analysis, or both.

The design should be implemented using authorized tools, and an analysis of dynamic run-time performance should be included to verify an application's functional capabilities. Each security measure that will contribute to assure the software's security capability is given the final review during the release phase. The response phase, which involves placing the incident response plan that was created during the release phase into action, follows after the release phase. This is important because it protects users from potential software flaws that could harm them or the software itself ("[4]").

## 5. CONCLUSION

The Spiral Model of SDLC is an advanced yet incredibly successful software development approach that may assist huge teams in managing complicated projects and reducing possible risks. Development teams may confidently manage budget, scheduling, security, performance, and other risks by following a process of thorough risk analysis, risk planning, and ongoing verification.

The Spiral Model's capacity to forge deep relationships with stakeholders makes it simpler to satisfy client demands, requirements, and expectations. This is one of its key advantages. The model also allows for the flexibility to add new features as the project progresses, guaranteeing that the finished output is current and relevant.

While the Spiral Model may not be as adaptable as Agile, it nevertheless offers a solid strategy. Despite not being as versatile as Agile, the Spiral Model still offers a stable and profitable approach to creating software. Clients can provide comments after each stage and convey the team periodically to ensure the project is progressing forward.

While it might not be a good fit for every project, it is still a great option for large, complicated projects where stakeholder communication and risk management are essential for success.

## 6. REFERENCES

"[1]." Microsoft, https://www.microsoft.com/en-us/securityengineering/sdl/. Accessed 30 April 2023.

"[2]." https://files.eric.ed.gov/fulltext/EJ1338318.pdf.

"[3]." 20 May 2018, https://www.radiojitter.com/sdlc-secure-development-life-cycle/. Accessed 30 April 2023.

"[4]." Techopedia, 14 August 2013, https://www.techopedia.com/definition/23582/microsoft-security-development-lifecycle-microsoft-sdl. Accessed 30 April 2023.