Babylonian Journal of Networking
Vol.2023, **pp**. 77–81
DOI: https://doi.org/10.58496/BJN/2023/009; ISSN: 3006-5372
https://mesopotamian.press/journals/index.php/BJN

Research Article

Network-Centric Approaches in Systems Development Life Cycle (SDLC): A Comprehensive Survey

Roula Abduljabbar [1,*], Mustafa A Jalil [1]

[1] *Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Cordoba, Campus Universitario de Rabanales, Cordoba, 14071, Spain.*

ARTICLE INFO

Article History
Received 12 Jun 2023
Accepted 14 Aug 2023
Published 15 Sep 2023

Keywords
Systems Development Life Cycle (SDLC)
Network Integration
Information Systems
Network Security
Scalability
Agile Development
Cloud Computing

ABSTRACT

In the rapidly evolving technological landscape, the integration of network considerations into the Systems Development Life Cycle (SDLC) is critical for developing robust and efficient information systems. This survey examines traditional SDLC models and explores how network requirements can be incorporated into each phase to enhance system performance, scalability, and security. By analyzing recent advancements and case studies, this paper provides insights into best practices for network integration and highlights the challenges and opportunities that arise in network-centric system development. The findings emphasize the need for adaptive SDLC frameworks that accommodate the complexities of modern network environments, offering valuable guidance for future research and practice.

1. INTRODUCTION

The System Development Life Cycle (SDLC) is a structured process that guides the development of high-quality information systems. Traditionally, SDLC models have focused on the technical and functional aspects of system development, often neglecting the critical role of networks. As organizations increasingly depend on complex, interconnected systems, integrating network considerations into the SDLC becomes essential. Networks impact every phase of the SDLC, from initial planning and design to implementation and maintenance, influencing system performance, security, and scalability [1].

This paper surveys the existing SDLC models, examining how network integration can be achieved throughout the life cycle. By exploring both theoretical frameworks and practical applications, we aim to provide a comprehensive understanding of how network requirements can be embedded within SDLC processes. This survey also identifies the challenges and best practices associated with network-centric system development, offering insights into future trends and research directions.

2. LITERATURE REVIEW

2.1 Traditional SDLC Models and Network Limitations

The Systems Development Life Cycle (SDLC) has been a cornerstone of software engineering, with models such as Waterfall, Agile, and V-Model providing structured frameworks for development [2]. The Waterfall model, characterized by its linear, sequential approach, has been effective for projects with well-defined requirements. However, it often struggles to accommodate the dynamic nature of network environments, where requirements can evolve rapidly. Similarly, Agile methodologies, with their emphasis on iterative development and user feedback, offer greater flexibility but may not fully address network-specific challenges such as security and scalability [3].

*Corresponding author. Email: z02kadkr@uco.es

The V-Model, another traditional SDLC approach, emphasizes verification and validation at each stage but often lacks the flexibility to incorporate network changes that arise during the development process. As network technologies evolve, these traditional models must be adapted to ensure that network considerations are integrated throughout the lifecycle [4,5].

3. NETWORK INTEGRATION IN SDL PHASES

a. Planning: Incorporating network requirements during the planning phase involves identifying network infrastructure needs, bandwidth, latency, and security protocols. This proactive approach ensures that the system's network capabilities align with organizational goals. For example, in cloud-based environments, planning must consider data storage locations, network redundancy, and failover strategies to ensure seamless service delivery [6].

b. Analysis: The analysis phase should assess network dependencies and constraints, considering how network architecture impacts data flow and system performance. This stage is crucial for identifying potential bottlenecks and security vulnerabilities. For instance, in Internet of Things (IoT) applications, the analysis must account for device connectivity, data transmission rates, and energy consumption [7].

c. Design: Network integration in the design phase focuses on developing architectures that support scalability and resilience. This includes selecting appropriate network technologies and designing robust communication protocols. In distributed systems, the design phase must address issues related to load balancing, data replication, and fault tolerance to ensure efficient network operations [8].

d. Development: During development, network engineers and software developers collaborate to implement network features and optimize system performance. This phase requires a deep understanding of network protocols and technologies. For instance, in microservices architecture, developers must design services that can communicate efficiently over the network while ensuring security and data integrity [9].

e. **Testing:** Testing must include network-specific scenarios, such as load testing and security assessments, to ensure that the system can withstand real-world network conditions. Performance testing should evaluate the system's ability to handle peak loads and maintain acceptable response times. Additionally, security testing should identify vulnerabilities in network communications and data exchanges [10].

f. **Implementation and Maintenance:** Implementation involves deploying the system within its network environment, while maintenance ensures that network updates and security patches are regularly applied to maintain system integrity. Continuous monitoring and proactive maintenance are essential to address emerging network threats and ensure optimal performance [11].

4. CHALLENGES AND PRACTICES [5]

Network integration within SDLC presents several challenges, including managing complexity, ensuring security, and achieving scalability. Best practices involve adopting adaptive frameworks that accommodate changing network requirements, leveraging automation tools for network management, and fostering collaboration between network and software teams.

1. Planning: In the first phase, the team determines whether or not there’s a need for a new system to reach the strategic objectives of a business. This is a feasibility study or preliminary plan for the company to acquire any resources necessary to improve a service or build on specific infrastructure. The main purpose of this step is to identify the scope of the problem and come up with different solutions. Some of the things to consider here include costs, benefits, time, resources, and so on. This is the most crucial step because it sets the tone for the project’s overall success. Thorough research is required before moving forward to the next stage.

2. Analysis: The second SDLC phase is where teams will work on the root of their problem or need for a change. In case there’s a problem to solve, possible solutions are submitted and analyzed to figure out the best fit for the project’s ultimate goal or goals. It’s where teams consider the functional requirements of the solution. Systems analysis is key in figuring out what a business's needs are. It also helps point out how those needs can be met, who will be responsible for certain parts of the project, and the timeline that should be expected.

3. Design: A Detailed List of the System Development Life Cycle Phases1-1Phase 3 defines the necessary specifications, operations, and features that will satisfy all functional requirements of the proposed system. It’s where end users can discuss and identify their specific business information needs for the application. During this phase, users will consider the important components, networking capabilities, and procedures to accomplish the project’s primary objectives.

4. Development: Real work officially begins in the fourth phase. This is the part when a network engineer, software developer, and/or programmer are brought on to conduct major work on the system. This includes ensuring the system process is organized properly through a flow chart. Many consider this the most robust SDLC stage as all the labor-intensive tasks are accomplished here. Phase 4 represents the real beginning of software production and hardware installation (if necessary).
5. Testing & Integration: In the fifth phase, systems integration and testing are carried out by Quality Assurance (QA) professionals. They will be responsible for determining if the proposed design reaches the initial business goals set by the company. It's possible for testing to be repeated, specifically to check for bugs, interoperability, and errors.
6. Implementation: Phase 6 begins when a huge part of the program code is completed. This phase also involves the actual installation of the newly-developed application. The project is put into production by moving all components and data from the old system and putting them in a new one through a direct cutover. This move is considered complex and uncertain but the risk is minimized substantially as the cutover often takes place during off-peak hours. Both end-users and system analysts should see a refined project with all necessary changes implemented at this time.
7. Maintenance: The V-Model_ Another Take on the System Development Life Cycle3In the seventh and final phase, end users can fine-tune the completed system as necessary if they want to improve performance. Through maintenance efforts, the team can add new capabilities and features and meet new requirements set by the client. This stage ensures the system stays usable and relevant by regularly replacing outdated hardware, inspecting performance, improving software, and implementing new updates so all standards are met. This also equips the system with the latest technologies to face new and stronger cybersecurity threats. All 7 stage shown in figure 1.

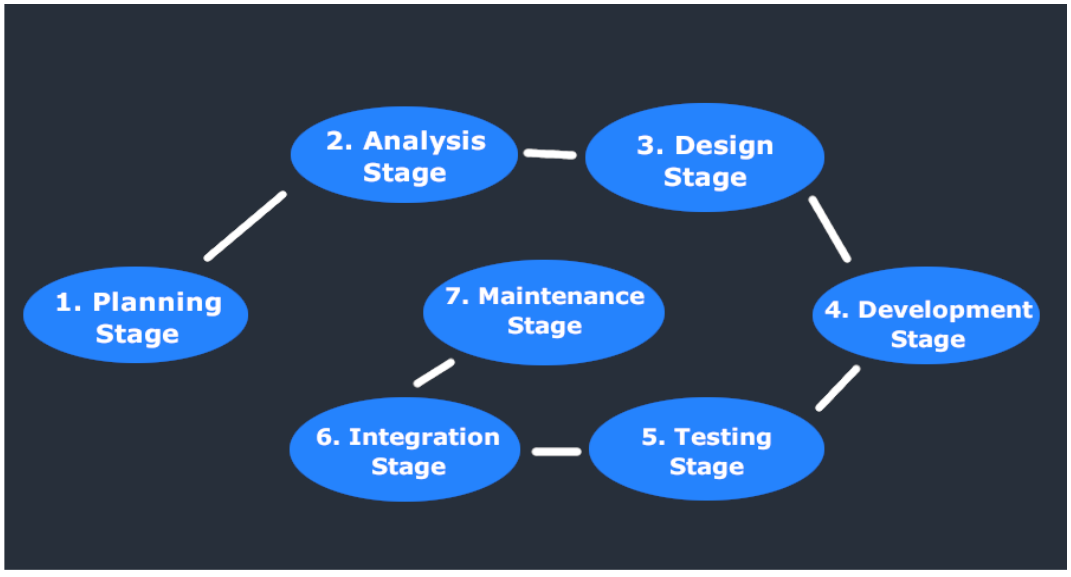

Fig . 1. Stages of SDLC.

5. EXAMPLE OF DSLS MODEL

a. Puts test specification as the critical design activity
   - Understands that deployment comes when the system passes testing

b. Clearly defines what success means
   - No more guesswork as to what "complete" means

c. The act of defining tests requires one to understand how the solution works.

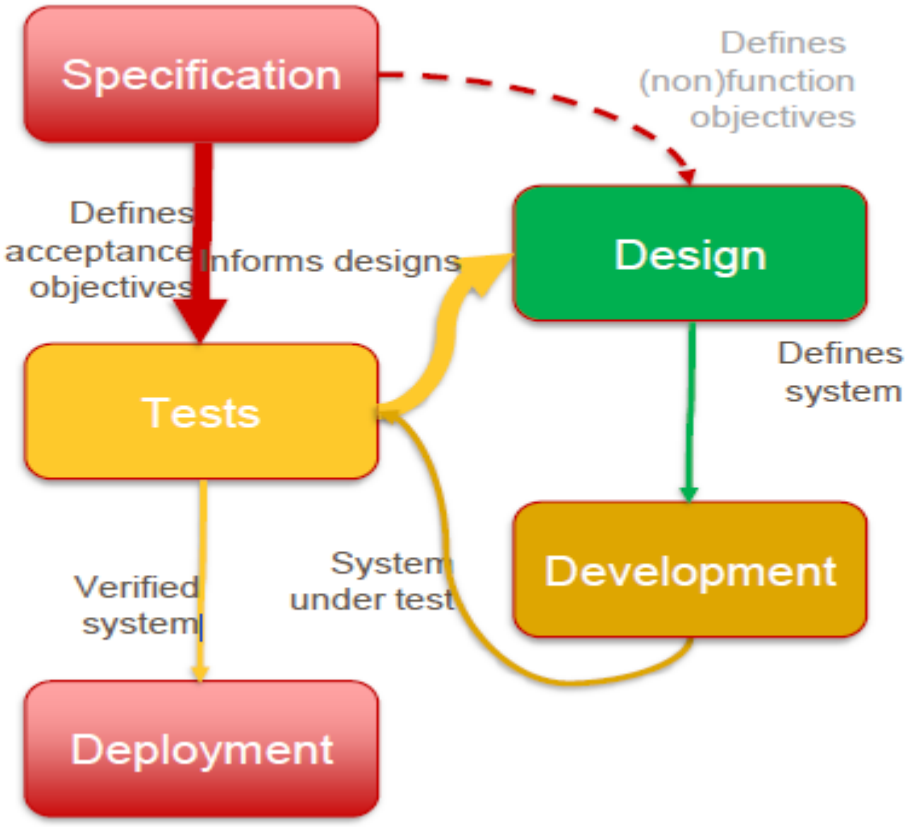

Fig . 2. Example of SDLC model.

6. CONCLUSION

Following the system development life cycle is crucial each time a new project or phase of a software project is released. Doing so gives teams a systematic approach that in turn enables them to come up with new solutions to existing issues in a standardized and controlled manner. How the SDLC will cover and satisfy overall requirements should be determined before embarking on a new project so you can achieve the best results. Once that step is done, you can select the right SDLC methodology or a hybrid of models that is perfectly suited to your main project requirements and expected end result. In conclusion, the SDLC must continue to evolve to keep up with the changing technological landscape. By embracing Agile methodology, integrating DevOps practices, incorporating security earlier, leveraging AI and ML, and shifting left testing, companies can build more secure, robust, and efficient systems.

The study for this survey involves a comprehensive review of existing literature, analysis of case studies, and identification of key themes related to network integration in SDLC. This approach provides a holistic view of current practices and emerging trends in network-centric system development. Integrating network considerations into the SDLC is crucial for developing robust and efficient information systems. By adapting traditional SDLC models to incorporate network requirements, organizations can enhance system performance, security, and scalability. This survey highlights the importance of a network-centric approach to system development and provides a foundation for future research and practice in this evolving field.

The Systems Development Life Cycle (SDLC) must adapt to rapid technological changes and evolving business needs. Future advice includes adopting Agile methodology, which allows for faster project completion and better outcomes. Integrating DevOps practices, which involve continuous integration, delivery, testing, and deployment, will ensure seamless software delivery. Integrating security proactively, as cyber threats increase, will reduce vulnerabilities in later stages. Leveraging Artificial Intelligence and Machine Learning can automate repetitive tasks, reducing costs and improving outcomes. Shifting left testing, which involves testing early in the SDLC process, can help identify bugs earlier, reducing

costs and improving software quality. These strategies will help ensure seamless software delivery and reduce the risk of vulnerabilities in later stages.

Conflicts Of Interest

The author's disclosure statement confirms the absence of any conflicts of interest.

Funding

The author's paper clearly indicates that the research was conducted without any funding from external sources.

Acknowledgment

The author extends appreciation to the institution for their unwavering support and encouragement during the course of this research.

References

[1] S. Sen, M. Patel, and A. K. Sharma, "Software Development Life Cycle Performance Analysis," in *Emerging Trends in Data Driven Computing and Communications: Proceedings of DDCIoT 2021*, Singapore: Springer, 2021, pp. 311-319.

[2] T. K. Tia, "Simulation Model for Rational Unified Process (Rup) Software Development Life Cycle," *Sistemasi*, vol. 8, no. 1, pp. 176-184, 2019.

[3] A. A. Adanna and O. F. Nonyelum, "Criteria for choosing the right software development life cycle method for the success of software project," *IUP Journal of Information Technology*, vol. 16, no. 2, pp. 39-65, 2020.

[4] D. A. Arrey, "Exploring the integration of security into software development life cycle (SDLC) methodology," Ph.D. dissertation, Colorado Technical Univ., 2019.

[5] L. Llerena, C. Almeida, J. W. Castro, and D. Buenaño, "Identification of Ethical Issues in the Phases of the Software Development Life Cycle: A Preliminary Secondary Study," in *2022 IEEE International Conference on Automation/XXV Congress of the Chilean Association of Automatic Control (ICA-ACCA)*, 2022, pp. 1-6.

[6] B. K. Jeong, S.-Y. Ji, and D. H. Jeong, "Lean IT With Value Stream Mapping Analysis: A Case Study in Software Development Life Cycle Process," *Information Resources Management Journal (IRMJ)*, vol. 35, no. 1, pp. 1-18, 2022.

[7] H. Cho, S. Kang, Y. Shin, and K. Cho, "A Study on the Application Method of Fuzz Testing to Domestic Weapon Systems Considering the Software Development Life Cycle (SDLC)," *Journal of the Korea Institute of Information Security & Cryptology*, vol. 31, no. 2, pp. 279-289, 2021.

[8] S. Al-Saqqa, S. Sawalha, and H. AbdelNabi, "Agile software development: Methodologies and trends," *International Journal of Interactive Mobile Technologies*, vol. 14, no. 11, 2020.

[9] H. P. Maryani, F. L. Gaol, and A. N. Hidayanto, "Comparison of the System Development Life Cycle and Prototype Model for Software Engineering," *Int. J. Emerg. Technol. Adv. Eng.*, vol. 12, no. 4, pp. 155-162, 2022.

[10] N. B. Ruparelia, "Software development lifecycle models," *ACM SIGSOFT Software Engineering Notes*, vol. 35, no. 3, pp. 8-13, 2010.

[11] G. Dlamini, S. Ergasheva, Z. Kholmatova, A. Kruglov, A. Sadovykh, G. Succi, A. Timchenko, X. Vasquez, and E. Zouev, "Metrics for software process quality assessment in the late phases of SDLC," in *Intelligent Computing: Proceedings of the 2022 Computing Conference, Volume 1*, Cham: Springer, 2022, pp. 639-655.