**Microsoft Dynamics 365: The Journey through SDLC Models**

*Exploring SDLC Models: Navigating Microsoft Dynamics 365 from Concept to Deployment*

**Devika Shetty**

**NNM23IS046**

|  |  |
| --- | --- |
| **Keywords:** | Software Development Life Cycle (SDLC), Microsoft Dynamics 365, Incremental Development, Spiral Model, Waterfall Model, Requirements Engineering, Risk Management, Change Management, Functional Requirements, Non-Functional Requirements, Deployment, Validation Strategy, Cost and Time Constraints. |
|  | Software Development Life Cycle (SDLC), Microsoft Dynamics 365, Incremental Development, Spiral Model, Waterfall Model, Requirements Engineering, Risk Management, Change Management, Functional Requirements, Non-Functional Requirements, Deployment, Validation Strategy, Cost and Time Constraints. |
| **Abstract:** | Microsoft Dynamics 365, a powerful enterprise resource planning (ERP) and customer relationship management (CRM) solution, undergoes a structured development process following various Software Development Life Cycle (SDLC) models. This paper explores the application of Incremental Development, Spiral, and Waterfall models in the context of Dynamics 365 implementation, analyzing their impact on requirements engineering, risk management, and change control. The study highlights how each model addresses functional and non-functional requirements, cost, and time constraints while ensuring efficient validation and deployment.  A comparative analysis of these models demonstrates their suitability for different project scenarios, considering factors such as scalability, flexibility, and stakeholder involvement. The research also presents a simplified requirements document and a robust validation strategy to ensure the accuracy and completeness of requirements before deployment. Additionally, a version-controlled GitHub repository is utilized to track document revisions and support collaboration. By evaluating the advantages and limitations of each SDLC model, this paper provides valuable insights into selecting the optimal approach for Microsoft Dynamics 365 projects, balancing efficiency, risk, and business objectives. |
| **Publishing:** | This paper was submitted to *Dr. Jason Elroy Martis, Associate Professor, Department of Information Science and Engineering, NMAM Institute of Technology. Nitte Karnataka, India.* This paper is also hosted on a GitHub repository, along with the material used for preparing this research. The link to the GitHub Repository is given in the endnote. |
|  |  |

# **Table of Contents**

# **Introduction 1-2**

# 1.1 Rationale for Selecting Microsoft Dynamics 365 in SDLC Analysis 1

# 1.2 Purpose of the Report 2

# **Software Development Life Cycle (SDLC) Models 3-10**

# 2.1 Incremental Model 3-5

# 2.2 Waterfall Model 5-7

# 2.3 Spiral Model 7-10

# **Requirements Engineering Phase for Microsoft Dynamics 365 11-12**

# 3.1 Simplified Requirements Document 11

# 3.2 Requirements Validation Strategy 11

# 3.3 Potential Challenges in Requirements Validation 11-12

# **Conclusion 13**

# **References 14**

**1.Introduction:**

Microsoft Dynamics 365 is a cloud-based Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM) solution used across industries like finance, manufacturing, and retail to manage business processes efficiently. Because it’s so complex and has to scale to fit different needs, picking the right Software Development Life Cycle (SDLC) model is key for a successful extend. The report aims to compare the Incremental, Spiral, and Waterfall models, analysing their impact on requirements management, risk handling, and cost or time efficiency. Understanding these models helps optimize Microsoft Dynamics 365 development for better performance and adaptability.

**1.1** **Rationale for Selecting Microsoft Dynamics 365 in SDLC Analysis:**

Microsoft Dynamics 365 is an exceptional choice for studying software process models and requirements engineering because:

* **Large-Scale system with Complex Requirements:**

As a cloud-based Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM) solution, Dynamics 365 amalgamates multiple business functions, requiring a well-defined Software Development Life Cycle (SDLC) to handle evolving requirements.

* **Industry-Wide Pertinency:**

Dynamics 365 is widely used in manufacturing, supply chain, finance and trade, making it multipurpose case study for comparing diverse process models.

* **Continuous Updates and Modification:**

Microsoft habitually updates Dynamics 365, requiring on adaptable development approach making it an exceptional subject for estimating risk management and change control schemes.

* **Various Functional and Non-Functional Requirements:**

The system includes modules for finance, sales, HR, and operations each with distinct recitals, security and scalability needs ideal for studying requirements engineering and authentication strategies.

**1.2 Purpose of the Report:**

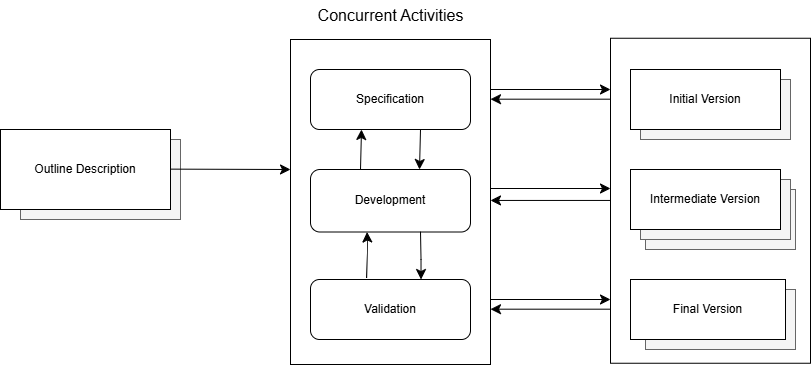
This report aims to examine and compare different Software Development Life Cycle (SDLC) models-Incremental Development, Spiral Model, and Waterfall Model-in the milieu of Microsoft Dynamics 365 implementation. The objective is to determine the most appropriate approach for developing, modifying, and deploying, Dynamics 365 solutions while considering key factors such as functional and non-functional requirements, risk and change management, time or cost constraints. Moreover, the report will outline a simplified requirements document, validation strategy, and version control practices using GitHub to support the structured software development.

**2. Software Development Life Cycle (SDLC) Models:**

The SDLC (Software Development Life Cycle) for Microsoft Dynamics 365 typically includes phases like Initiate, Plan, Design, Develop, Test, deploy and Operate following a organized approach to design, develop, test, and deploy Dynamics 365 solutions, with key features like requirement gathering, formation, customization, combination, through testing and ongoing maintenance throughout the lifecycle.

**2.1 Incremental Model:**

Incremental model also known as the successive version model is a process of software development where requirements divided into multiple standalone modules of the software development cycle. The Incremental Model is well-suited for implementing Microsoft Dynamics 365 as it allows business to grow and deploy the system in manageable increments. This method aligns with ERP and CRM solutions, where gradual deployment and integration are necessary.



*Fig. 1. Incremental Model*

**2.1.1 Functional Requirements and Non- Functional Requirements:**

**Functional Requirements:**

* + The Incremental Model allows Dynamics 365 to be installed module by module, safeguarding that business-critical functions are available early.
  + Each increment adds new functionalities based on business needs, making it easier to manage dense implementations.
  + Organizations can progressively implement power automate workflows and third-party integrations without tremendous implementations.
  + Business logic and mechanization rules can be added in further increments.

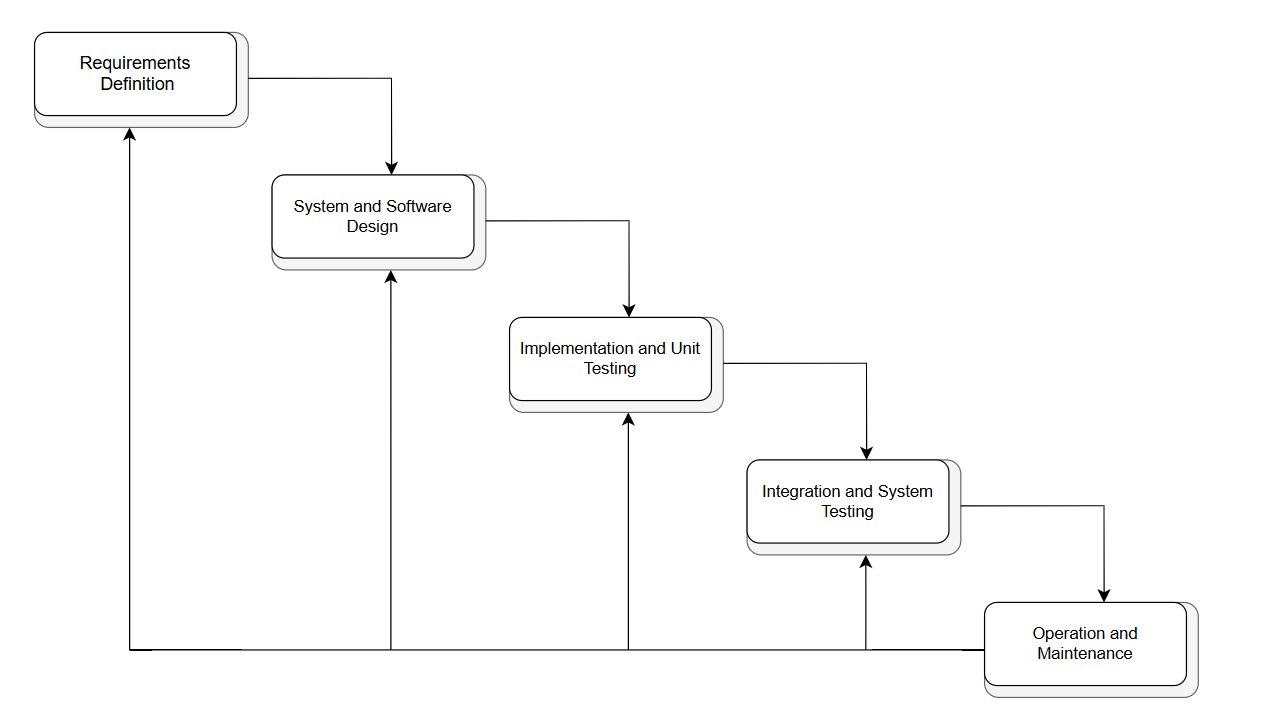
**Non-Functional Requirements:**

* Performance can be enhanced in each increment by nursing system load and user interactions.
* Scalability is obtained incrementally by intensifying cloud resources as usage raises.
* Security policies can be implemented in stages decreasing security risks.
* Consent with industry standards can be safeguarded incrementally.
  + 1. **Risk and Change Movement:**
* Since each increment is verified before the next is implemented, risks can be noticed and alleviated early.
* Errors in business process automation, data relocation or integrations can be modified before affecting the entire system.
* Unlike the Waterfall Model, where changes are expensive, the Incremental Model allows modification to be integrated in future increments.
* Users can provide real-time feedback after each increment, letting enhancements to improve system usability.
  + 1. **Time and Cost Constraints:**
* The most serious Dynamics 365 features can be installed first, permitting businesses to start promoting initially.
* Spreads prices over multiple phases, lessening upfront investment.
* Highlights high-influences features first, making sure businesses invest in what matters utmost.
* Teams can work on various increments in parallel and testing is done incrementally, avoiding large-scale revise at the end.
  + 1. **Incremental Model - Advantages and Disadvantages:  
       Pros:**
* Portions of an application can be delivered first, allowing the business to utilise key features such as CRM before the whole application goes live.
* It is possible to incorporate last-minute changes and new features into later increments.
* Each increment undergoes tests before the next one is initiated, which serves to running the risk of having to perform additional corrective measures.
* Resources could get time to be managed efficiently for expenses spread over time.

**Cons:**

* Potential integration problems. Newer features may not integrate smoothly with legacy ones.
* Delayed full-scale implementation. Typically, getting the whole system off the ground is a real delay.
* More costly overhead in management - each phase calls for its separate test, refresh, and training.

**2.2 Waterfall model:**Waterfall Model is old style. It is linear and sequential. Each phase has to be completed before the next phase can commence. This fits very well for projects that have well-defined and stable requirements. It is a linear and sequential approach to software development that consists of several phases. It must be completed in a specific order. This classical waterfall model is simple and idealistic. It is often used for large-scale projects with long timelines, where there is little room for error and the project stakeholders need to have a high level of confidence in the outcome. The Waterfall Model can be effective for Microsoft Dynamics 365 implementation when requirements are stable and predefined. However, for projects with evolving needs, Incremental or Agile models may offer greater flexibility.



*Fig. 2. Waterfall Model*

**2.2.1 Functional requirements and non-functional requirements:**

**Functional requirements:**

* Use CRM to manage customer data, sales and marketing automation.  
  ERP manages execution of financial transactions, payroll, and inventory.
* Set up user permissions and roles.
* Integrate seamlessly with Microsoft 365, SharePoint, and Azure.

**Non-Functional Requirements:**

* Scalable for new users and modules.
* Implementation of multi-factor authentication and encryption in the data.
* Targeting an uptime of 99.9% through an automated backup and recovery solution.
  + 1. **Risk and Change Management:**
* Requirements must be clearly defined ahead of time, and late errors could carry serious risks.
* Integration with Microsoft services may lead to unexpected technical problems.
* Changes need formal approvals, which can make them long and costly.
* Complete system deployment may have user resistance and problems with training.
  + 1. **Time and Cost Constraints:**
* Having high initial costs due to high-level planning, design, and development before deployment.
* Setting demanding timelines, since each phase must finish first before moving to the next phase, thus causing delays when any issues arise.
* Late-stage errors can be costly to fix, thereby increasing the overall project costs.
* There could be a larger budget overrun if requirements drift after development has started.

**2.2.4 Waterfall Model - Advantages and Disadvantages:  
 Pros:**

* Clear structure and well-defined phases ensure systematic development.
* Detailed documentation helps in future maintenance and upgrades.
* Suitable for projects with stable requirements and minimal expected changes.
* Early-stage planning reduces uncertainty and miscommunication.

**Cons:**

* Limited flexibility makes it difficult to accommodate changes once development starts.
* Late-stage testing increases the risk of costly fixes if issues are found.
* Long development cycles delay deployment and business benefits.
* High initial costs due to extensive planning and upfront resource allocation.
  1. **Spiral Model:**The Spiral Model is a risk-driven software development approach that combines iterative development with organised risk management. It is suitable for complex and budding projects like Microsoft Dynamics 365, where requirements may alter over time.

The Spiral Model is a powerful approach for Microsoft Dynamics 365 implementations where:

* Requirements are evolving, and risks need continuous assessment.
* Prototyping and phased delivery are essential.
* The project involves multiple integrations and high business impact.

For high-risk, large-scale ERP and CRM implementations, the Spiral Model ensures a structured yet adaptable development process.



*Fig. 3. Spiral Model*

**2.3.1 Functional requirements and non-functional requirements:  
 Functional requirements:**

* Permits incremental development of Microsoft Dynamics 365 modules.
* The modularization approach requires inspiration from phased testing and validation before final integration into the whole.
* Enable end-to-end data traversal and integration with existing business systems.
* Users collect their responses at the end of each development cycle to provide feedback about further improvements.

**Non-functional Requirements:**

* Provide a continuous estimation and justification for the risks involved, in order to maintain system security.
* Scale up to the changing business demands and growth.
* Foster stability within the system by instilling passive tests for all instances served.
* Achieve higher availability through automated backing up and disaster recovery planning.

**2.3.2 Risk and Change Management:**

* Risks are continuously identified and mitigated in each development cycle.
* With the help of frequent user feedback, requirement changes can be detected early in the life-cycle process. The chance for rework can then be substantially reduced.
* Flexibility in adapting to changing business needs reduces project failure risk.
* Experienced teams are required to assess and manage risks in each iteration, effectively.

**2.3.3 Time and Cost Constraints:**

* Iterative development lengthens both project duration and cost considerations.
* Continuous risk analysis and stakeholder feedback take additional time and resources.
* Every single iteration would require a certain amount of funding set aside, making it difficult to estimate proper costs.
* Longer planning and testing phases tend to delay full deployment of the system.

**2.3.4 Spiral Model - Advantages and Disadvantages:**

**Pros:**

* **Effective Risk Organization –** At first, it spots all risks and then lessens them in each round, modifications are allowed. They are focused on evolving needs in business.
* Firm feedback from Stakeholders undoubtedly guarantees the system attracts possible users.
* Prototyping early brings forth multiple working models throughout its stages, decreasing any ambiguity.

**Cons:**

* The need for more properties due to repeated risk examination and testing results in a higher cost.
* Complex Management requires highly skilled teams for active scheduling. Assessment is needed as well.
* Complete system deployment could lag given many iterations that lengthen development time.
* Accounting is hard because constant changes occur.

**3. Requirements Engineering phase for Microsoft Dynamics 365:  
3.1 Simplified Requirements Document:**

**Functional requirements:**

* That will allow customer data management, sales mechanisation, and advertising campaign chasing.
* Financial dealing, billing, workforce, and inventory management.
* Role-based admission control with a user verification and authorisation system.
* Complete Microsoft services integration work, such as Azure SharePoint and Power BI.

**Non-Functional Requirements:**

* The application service should respond within 2 seconds for ordinary assignments.
* System uptime 99.9% and automated backups, disaster recovery.
* Encryption of the data and two-stage authentication for protection.
* Capable of scalability to adjust to future business expansion and increased user.

**3.2 Requirements Validation Strategy**To clearly assure the correctness and completeness of requirements, the ensuing validation strategies can be implemented:

* **Shareholder Review:** Hold periodical review sessions with users, management, and developers.
* **Prototyping:** Provide for validating system behaviour of a model as it moves forward into development.
* **Requirement Testing:** In test cases, verify whether the requirements have been properly implemented.
* **Traceability Matrix:** Ensuring that every functional requirement tracks back to the business need.

**3.3 Potential Challenges in Requirements Validation:**

* **Changing Needs:** Business needs are continually changing.
* **Conflicting Stakeholder Expectations**: Different departments may have conflicting priorities
* **Unclear or Incomplete Specifications:** If specifications are not clearly defined, there may be misunderstanding.
* **Technical Constraints:** Integration with legacy systems might present unknown challenges.

**4. Conclusion:**

While deciding on the best software process model for Microsoft Dynamics 365, the nature of enterprise-level solutions, the modifications, changing customer needs, and complex integration requirements must all be taken into deliberation.  
One of the oldest models in the software development field that is still widely in use today, Waterfall offers a systematic linear approach that fully defines the requirements with few changes that could arise. In contrast, Microsoft Dynamics 365 releases new features, integrations, and customizations regularly. An inflexible Waterfall approach would hardly adjust and thus fail to accommodate this intense change, leading to inefficiencies and expensive rework.  
Incremental Development presents the system in smaller models, delivering working software in stages. This is apt because of the modular nature of Dynamics 365 and allows the company first to implement CRM, Finance, and so on, and then build on. This decreases risks by providing validation to the earlier increment before working on a new one. Nevertheless, if complex risks come into play, such as security concerns, compliance issues, or major architectural changes, using this model alone may prove inadequate when it comes to structured risk management.  
Taking all this into account, the Spiral Model juggles iterative stages of software development with risk assessment in each part of a spiral. This seems an ideal method for Dynamics 365 implementations because businesses would need to assess risks in security, compliance, and performance before attaining full performance. The spiral method is flexible regarding new requirements as organizations continue to grow. Due to the fact that the Spiral may use many resources, this is counteracted by a structured risk management process, which makes the Spiral model best for large-scale ever-changing systems such as Dynamics 365.  
**Final Verdict**: For balancing the flexibility, risk management coupled with phased deployment, the Spiral Model is most appropriate for Microsoft Dynamics 365. And it provides a way for the organizations in question to evolve in terms of needs, whilst carefully attending to risk management for making smoother implementations and future-proofing itself.

**5. References:**

1. <https://engineering.futureuniversity.com/BOOKS%20FOR%20IT/Software-Engineering-9th-Edition-by-Ian-Sommerville.pdf> - (Sommerville, 2010) *Software Engineering, 9th Edition*

2. <https://outsmartteam.com/sdlc-models-for-microsoft-dynamics-crm-development-which-one-should-you-pick/> - (Outsmart Team, n.d.) *SDLC Models for Microsoft Dynamics 365*

3. <https://www.tpointtech.com/software-engineering-incremental-model> - (TPoint Technologies, n.d.) *Software Engineering: Incremental Model*