**PROJECT REPORT**

*on*

**Software Development Lifecycle (SDLC) Analysis of Shopify**

***A comparative study of different models in relation to Shopify’s software development***

**Akshatha Kini**

**NNM23IS009**

|  |  |
| --- | --- |
| Keywords: | SDLC, Shopify, AWS, Integration, Testing, Scalability, Requirements validation |

|  |
| --- |
| **Abstract:** |
| Software development is an evolving discipline requiring structured approaches that could build scalable, secure, and efficient systems. This report seeks an analysis of Software Development Life Cycle (SDLC) models that are applicable to Shopify as a globally known e-commerce system. It attempts a comparative analysis of SDLC methodologies by bringing forward the overview requirements engineering. The goal of this study is to provide insight into the selection of an appropriate SDLC model for a large-scale cloud-based platform with special focus on the waterfall model, incremental development and, spiral model approaches. This report will also discuss challenges and strategies regarding requirements validation and software deployment at Shopify. The findings of this document are drawn from extensive research, industry best practices, and insights from Shopify websites. It is hoped that the material examined herein will provide a good theoretical basis for software engineers, architects, and researchers interested in SDLC and requirements engineering methodologies toward large-scale streaming platforms. This paper is followed by a conclusion and short remarks on future work. |

**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**

1. **Introduction** 3-4
2. **Analysis of SDLC Models**

* 2.1 Incremental Development Model 4-6
* 2.2 Waterfall Model 6-7
* 2.3 Spiral Model 8-9

1. **Requirements Engineering for Shopify**

* 3.1 Functional Requirements 9-10
* 3.2 Non-Functional Requirements 11
* 3.3 Requirements Validation Strategy 12

**4.Conclusion** 13

**5.Refrences**  13

**1. Introduction**

Shopify is an e-commerce platform that helps small businesses to create and manage online store. It provides tools for setting up an online store, processing payments, managing inventory and handling shipping. It’s a software as a service (Saas) which is a cloud computing model that provides software applications to user over the internet. Saas platform offers tools for designing, setting up and growing an online business.

**1.1 Reasons for Selecting Shopify**

* **User-Friendly**: Given its simple interface, it is easily navigated by a beginner.
* **Broad-Scale Software System**: It operates at a massive scale, having large scale transactions worldwide.
* **Complex Requirements**: Shopify has multiple user needs, from small businesses to bigger ones.
* **Open Source**: It operates on GitHub for managing its open-source projects.
* **Transaction Processing Requirement**: It supports various payment gateways and third-party options like PayPal and Stripe making transactions easy for customers.
* **Security**: It is PCI DSS compliant, which means that the customer payment detail is completely secure.

**1.2 Purpose of This Report**

The report discusses SDLC stages at Shopify and how the choice of a different development model governs growth in an organization. This is aimed at realizing how Shopify adapts quickly to users' needs through the construct of a large system of software.

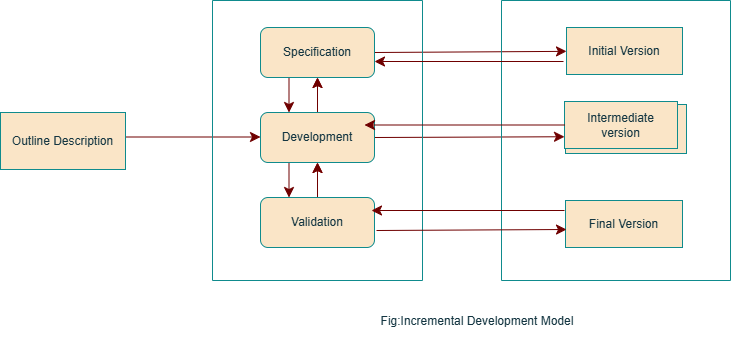
1. **SDLC Model Comparisons**: The instantiation of the comparison points between the waterfall, incremental, and spiral models, to show which suits Shopify's development approach the best.
2. **Requirement Management**: Looking into how the organization itself relates to other ventures to provide services catered precisely for the client's needs.
3. **Version Control**: Observing the usage of GitHub in the context of the version control systems used by Shopify to keep everything right in coding management.
4. **Software Quality**: This talks about Shopify using automated testing to show and integrate accountability across its platform.
5. **Requirements Document**: This is a simplified, systematic, and straightforward guide to defining and managing software requirements.  
   The report highlights the processes through which Shopify builds and runs one of the strongest e-commerce platforms on the planet.

**2.Software Developmental Lifecycle Models (SDLC)**

Software Development lifecycle (SDLC) is a structured process. It is a practice that defines entire procedure of software development step-by step. The goal of SDLC is to deliver good-quality, maintainable software that meets the user’s requirements. SDLC models plans for each stage so that stage of the software development model can perform task efficiently. It consists of detailed plan that describes how to maintain replace and enhance specific software. There are various SDLC models which helps in in software development.

**2.1 Incremental Development Model**

Shopify primarily follows Incremental Development Model. The Incremental Development model is method of building software products in the system is developed in small, manageable chunks called increments rather than a complete product in one go. These increments are planned developed and tested individually. Each increments adds new functionality to the system until the final product is fully developed. After each increment it is reviewed, allowing modifications. Requirements can be changed without affecting the entire system.



**2.11 Phases of Incremental Development Model for Shopify:**

1. **Requirements Gathering and Analysis:**

This involves gathering and analysing the requirements for the Shopify store. This includes understanding the business needs, audience and desired functionalities. This can be divided into smaller and manageable modules.

1. **Design:**

Design phase includes the user interface design, database design and system architecture. The design of the model should be flexible as per the shareholders requirements and scalable to accommodate future increment.

1. **Development:**

Development phase is where the actual coding and development of the system takes place. Every module is developed and tested independently. Shopify’s developers write code in backend, frontend and database.

1. **Testing:**

After developing phase it is thoroughly tested to ensure that it meets the requirements and is free of bugs. This may involve unit testing, integration testing and user acceptance testing. Shopify runs automated tests to check payment gateway functionality, load testing to handle high traffic, and security tests to prevent fraud.

1. **Deployment:**

After testing, the module is deployed to Shopify store. This involves updating, adding new functionalities and configuration settings.

1. **Evaluation:**

After model is deployed, it is evaluated to ensure that it is meeting the stakeholder’s business needs and user expectation.

1. **Iteration:**

Iteration process is repeated for each module until the entire Shopify software is built. New iteration builds upon previous one adding new features and functionalities to previous one.

**2.1.2 Functional and Non-Functional Requirement**

* **Functional Requirements:**

Incremental models requires needs complete requirement specifications before development begins. It is suitable foe well defines and static requirements but it lacks flexibility for evolving e-commerce needs.

* **Non-Functional Requirements**:

The non-functional requirements of this model is performance optimization, security updates, and user interface improvements which can be incremented in each iteration.

**2.13 Risk and Change Management**

**Risk Management:** Lower risk due to early and continuous testing of small modules.

**Change Management:** It is moderately flexible as incremental changes are possible which impacts time and cost.

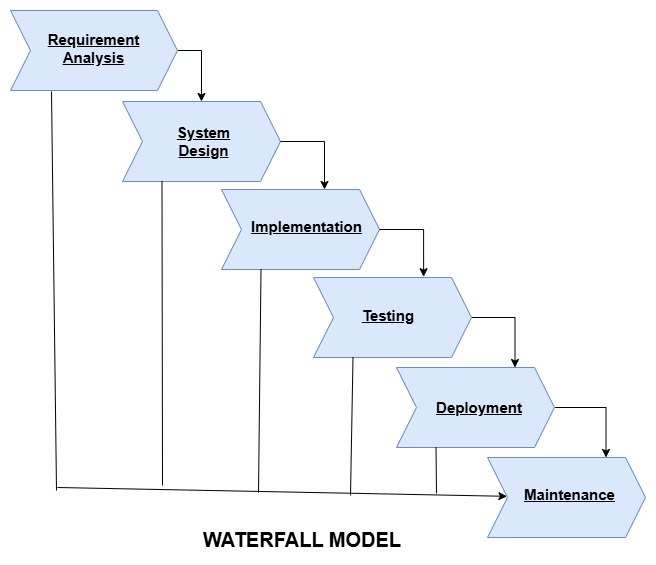
**2.14 Time and Cost Constraints**

**Time constraints:** It is faster with initial releases and also with incremental improvements due to this it is good choice for Shopify.

**Cost Constraints:** Cost of this model is moderate and manageable but the cost may increase if there is multiple iterations.

**2.2 Waterfall Model**

A Waterfall model is a methodical and step-by-step manner of proceeding from one phase of creating software to another. This approach insists on the completion of one phase before the next can be started. Therefore, the main objective of a Waterfall model is to make sure that all software requirements are met and that software operates as per expectations. This is a structured, rational method to software construction that ensures good quality and effective work throughout the engineering process.



**2.21 Phases of the Waterfall Model for Shopify Development**

The waterfall model for Shopify is applied as follows:

1. **Requirements gathering and analysis:**  
   Business requirements for Shopify are collected and analysed. For instance, Shopify should provide support to online merchants with their product listing, payments and order management. A Software Requirement Specification report encapsulating all functional and non-functional requirements must be mentioned.
2. **System design:**  
   This phase is concerned with system architecture, database structure, and technology. As an example, during backend development, Ruby on Rails may be leveraged by Shopify to achieve scalability. The frontend has to be responsively designed with React.js. A secured MySQL/PostgreSQL database structure is set out for order and customer management.
3. **Implementation:**Input from system design allows a system to be first developed as small independent programs known as units, which are put together in later phases, with development having implemented a checkout system that integrates with third-party payment gateways, PayPal, Stripe.
4. **Integration and testing:**  
   At the conclusion of the implementation phase, all units of the work done during implementation are integrated to form a complete system after all individual components have been tested. Different types of testing phases include unit testing, integration testing, and performance testing.
5. **Deployment:**  
   Once functional and non-functional testing is complete, the product is deployed into the customer environment or officially released into the market. Once the Shopify platform goes live it becomes available to its users.
6. **Maintenance and Support:**  
   For the bugs raised in the client's environments, we need maintenance. Moreover, there might be releases in order and bug fixes in better versions of the product.

**2.22 Functional and Non-Functional Requirements**

* **Functional Requirements:** Functional specifications would include core features such as product browsing, adding items to the cart, the checkout process, payment integration, order tracking, and user account management.
* **Non-Functional Requirements:** Early assessments of this include the performance, security, and scalability aspects; however, these will be hard to adjust afterward.

**2.23 Risk and Change Management**  
**Risk Management:** It performs minimal risk assessment throughout. It is high risk if the requirements are altered.  
**Change Management:** Once development begins, it is impossible to change anything.  
This model is definitely not ideal for use with Shopify.

**2.24 Time and Cost Constraints**Time constraints: A little more time is required before deployment because everything has to happen sequentially.  
Cost Constraints: Cost is low but it may cost dearly if any big changes are done later.

**2.3. Spiral Model**

The risk-driven software development model that combines iterative development with systematic risk analysis is the Spiral Model. It is ideal for large-scale and complex projects like Shopify, which continuously evolve and feature continuous improvements.



**2.31 Phases of the Waterfall Model for Shopify Development**

1. **Planning (Originate Objectives and requirements):**Summarizes business needs, risks, and core requirements. For instance, the Shopify user envisions an e-commerce store wherein he can integrate some form of payment.
2. **Risk Analysis (Identify and Mitigate risks):**  
   In the next steps, the prospective risks are identified and plans are formulated. This may include assessing if the AWS-based cloud architecture of Shopify is capable of sustaining millions of users.
3. **Engineering (Building and Prototyping):**This is the phase when we get to build the actual Shopify store; this can go from customizing the store theme, specific features, to integrating third-party services and configuring the settings in the store. Prototypes and incremental builds of the store are developed and tested.
4. **Evaluation:**

Finally, during this phase, the developed increment or prototype is evaluated against the project's objectives and requirements. It determines whether to go to the next iteration or revise the project plan.

**2.32 Functional and Non-Functional Requirements**

* **Functional requirements**: product listing, friend knitting, shopping cart management, checkout, user account management, and payment processing.
* **Non-functional requirements:** Performance, security, scalability, usability, compatibility with mobile devices and other browsers/devices.

**2.33 Risk and Change Management**

**Risk Management:** Strong risk assessment at each phase, ensuring proactive issue mitigation.

**Change Management:** It is difficult to accommodate changes once development has started.

Spiral model is not ideal for Shopify due toned for frequent updates and growing business needs.

**2.34 Time and Cost Constraints**

**Time Constraints:** The long modifications and development processes would lead to a delay in the launch.

**Cost Constraints:** Higher costs due to extensive risk management and iterative refinements.

**3.Reqirement Document**

A Requirement Document is a formal document that outlines the essential features, functions, and constraints of a software system. It serves as a blueprint for developers, designers, testers, and stakeholders to understand what needs to be built. It's also known as a software requirements specification (SRS).

**3.1 Functional Requirement**

Functional requirement describes what the software should do, how it should react to inputs as well as outputs.

**3.11 User Management:**

* The users should be able to login and manage profiles securely.
* There should be different user roles like store owner, admin and customer roles.

**3.12 Store Management:**

* Store owners should be able to create, update, and delete online stores.
* Ability to customize store appearance using themes and templates.

**3.13 Product Management**

* Support for inventory tracking and stock management.
* The product availability in different variants like size, colour etc

**3.14 Order Management**

* Customers should be able to place orders ad track order status.
* Store owner should be able to process orders.
* The Software should process order cancellation and support refund.

**3.15 Payment Processing**

* It should support multiple payment gateways.
* The payments information should be securely handled.

**3.16 Shipping and Delivery**

* It should define proper shipping rates and methods of shipping.
* It should also provide information about logistics providers and have a real-time tracking system.

**3.17 Marketing and SEO**

* It should have built in SEO tools, and should provide discount codes .
* To increase the user base marketing is done through integrating with social media and other marketing tools.

**3.18 Reporting and Analysis**

* It should maintain various reports like sales inventory and customer insights in the database.

**3.19 Security and Compliance**

* The user’s data should be secure and data should be encrypted.

**3.2 Non-Functional Requirements**

Non-functional requirements specifies how the system performs and refers to its qualities like security, reliability, and usability.

**3.21 Performance**

* The system should handle high traffic loads efficiently.
* The pages of the websites should not buffer and should run smoothly.

**3.12 Scalability**

* The software should support millions of users and should handle the transactions.

**3.23 Availability and Reliability**

* The website should be highly available and data should backup regularly.

**3.24 Usability**

* Even for first-time visitors, the websites should remain intuitive, easy to navigate through, and checkout short, smooth, and user-friendly.

**3.25 Security**

* Customer data must always be encrypted and stored in accordance with the data privacy provisions of the applicable laws and regulations. Besides, the system must fully comply with PCI DSS to ensure the safe handling of payment-related information. The website must be secured against common web vulnerabilities.

**3.26 Maintainability and Upgradability**

* The Shopify theme and the customizations shall be very well-documented so that users can maintain and upgrade them easily.

**3.3 Requirements validation**

Requirements Validation is the process of ensuring whether or not the requirements gathered for a system meet the needs and expectations of stakeholders. Such validation involves determining whether the requirements are clear, complete, consistent, feasible, and testable. The aim overall is to determine if the right requirements have been identified before any development phase begins so that costly changes can be avoided at later project stages.

**3.31 Requirement Validation Strategy**

* **Stakeholder Review and Approval:** Requires formal reviews with stakeholders to ensure business needs are being met and confirm sign-offs prior to key decision-makers for decision-making before developments may begin.
* **Prototyping and User Feedback:** A tool to craft and develop prototypes to visualize what functionality can do and receive feedback from the users on the views or refinements of the **implementation** requirements.
* **Requirement Testing:** Conducts testing in a number of domains to check whether any of the requirements are being implemented correctly.
* **Traceability and Change in Management:** Maintains a Requirement Traceability Matrix (RTM) to track the requirements during all development stages. Initiates a change control process to document each change made and its impact.
* **Feasibility and Performance Validation**: Assess the technical feasibility ensuring the realistic implementation of requirements. Validate performance metrics such as scalability, security, and response time.
  1. **Potential Challenges in Requirements Validation**
* **Unclear requirements and requirements which are not complete:** Poorly defined requirements may lead to misinterpretation and wrong implementation.
* **Changes in business:** Some change in requirements can still emerge over futures in a life cycle phase of projects, thus increasing project distress in terms of contraction schedule and increased rework time
* **Conflicting interests from those stakeholders involved:** Numerous concerns and expectations are generated among different teams (business, developers, marketing), thereby creating a turmoil of how to prioritize.
* **Constraints in terms of money and time:** Limitations such as budgets, skilled resources, or time might not permit rigorous validation.
* **Problems with integration and compatibility:** Ensuring compatibility between various development environments with currently existing systems and third-party tools is daunting.

**4.Conclusion**

In conclusion, the SDLC model in Shopify has done quite well in terms of a balance between adaptability and scalability through the feature of Incremental Development, which accommodates rapid iterations and fast reflections from clients. This further grooming under the Spiral Model nature of risk mediation helps for constant improvements and agile adjustments to match e-commerce environments. Where Waterfall brings on such tight structure, Shopify relies heavily on adaptability to keep the platform audibly fresh, top quality, and consumer-oriented against constant cuts from the competition.

**5.References**

1. Software-Engineering-9th-Edition-by-lan-Sommerville.pdf-book.
2. Shopify-website- <https://www.shopify.com/in/blog/product-development-life-cycle>
3. https://aws.amazon.com/what-is/sdlc/ -website