Software Development Lifecycle Analysis of Redbus

A Comparative Study of Different Models in Context of Redbus's Software Development

Charanraj M

Nitte Mahalinga Adyantaya Memorial Institute of Technology

{ nnm24is501@nmamit.in , charanaikofficial@gmail.com }

https://github.com/sudo-charan/sdlc-redbus-analysis

Keywords: Software Development Lifecycle, Redbus, Online Booking, Integration, Testing, Scalability, Requirements Validation

Abstract:

The field of software development is extremely dynamic and requires a structured methodology to bring about a system that is scalable and secure. This report explores the Software Development Life Cycle (SDLC) models applicable to Redbus, the foremost bus ticketing platform online. It draws a comparative study of various methodologies along with their working application in the development for Redbus.

The research aims to shed light on exercising an appropriate type of SDLC model for a very complex online service platform named Redbus, focusing on the Waterfall, Incremental Development, and Spiral Model approaches. It also discusses the issues and techniques in the way of requirement validation, system integration, and secure deployment in Redbus.

It also spells out key insights and considerations for deploying SDLC models in real-world scenarios.

Publication:

The paper is hosted along with the reference material and resources used to prepare the research on GitHub.

Table of Contents

1. Introduction

The software development life-cycle (SDLC) encompasses creating and developing structured yet comprehensive procedures for forming premier-quality software through testing while maintaining it. The well-structured method enables better achievement of reliability while improving quality together with efficiency. Redbus functions as an ideal website to evaluate the fundamental aspects of Waterfall and Incremental Development and the Spiral Model approaches.

This media presentation evaluates the pros and cons of each model and provides substantial development paradigms for Redbus while explaining functional aspects, non-functional needs and presents requirement verification obstacles and potential solutions for each model.

2. Redbus System Overview

2.1 About Redbus

The Redbus system operates as an online platform that delivers a combined solution for bus users to make bus ticket reservations through the internet. Different layers form the architectural structure of Redbus due to its multiple components.

2.2 System Overview

- **User Interface Layer**
  The first visible section represents the company interface which provides a friendly experience for customers.
  Both web applications and mobile applications find support from this platform.
  The platform enables users to investigate available buses in addition to conducting seat availability checks before allowing them to complete their reservations.

- **Application Layer**
  The application layer provides business request reception and booking management services to customers.
  This part of the system manages user authentication as well as controlling session management for users.

- **Data Layer**
  User information along with booking data and payment information is stored as standard practice in this layer.
  The storage system within relational database management uses structured data structures.
  At this stage organizations fulfill all core data security and privacy criteria.

- **Integration Layer**
  Through its integration layer the system establishes hooks with the APIs from associated service providers including payment processors and hotel reservation systems.
  The platform gives access to Bus operator partner APIs.
  The different services can interact with each other through this mechanism in a simple manner.

- **Monitoring and Analytics**
  The system provides monitoring capabilities to observe platform activity together with user operations.
  The system presents recommendations to boost service delivery quality.

2.3 Technology Used

- **Front-end:** HTML, CSS, JavaScript, React.js for building user interfaces.
- **Back-end**: Node.js, Python, and Java for server-side logic.
- **Database:** The program implements MySQL as its database system while using MongoDB to handle unstructured content.
- **Security:** HTTPS encryption, OAuth for authentication, and advanced fraud detection algorithms.
- **Cloud Services:** Amazon Web Services provides AWS as their cloud service platform for scalable storage needs.

3. Comparative Analysis of SDLC Models

3.1 Waterfall Model

Quite the opposite, The Waterfall Model is characterized by being linear and sequential. These steps are cyclic and interrelated in the sense that one cannot proceed to the next step without attaining a completion on the previous one. The life cycle normally comprises of requirements definition, system architecture, development, integration, testing, installation and support stages.

How Redbus Would Be Developed Using Waterfall Model

The Waterfall Model can be well implemented during the early stage of Redbus because the requirements of the software were well defined and easy to implement. Requirements; The real sophisticated requirements that the development team would collect include online bus ticket booking, real time seat availability and secure payment gateway.

1. **Development**: The look and feel of the actual product, the internal architecture of the user interface, as well as the application architecture and the structure of the database would all be decided on at this stage.

2. **System Design:** The concept and structure of the whole program as well as the layout of the user interface, overall design of the application and the structure of the database would be determined.

3. **Implementation:** In the implementation process developers will solve each of the modules: registration, search for the bus, and payments one by one.

4. **Testing**: It means that all the above-said modules would be put into practice and tested over and over again to expose thebugs.

5. **Deployment**: The deployment of the system is also done as one package.

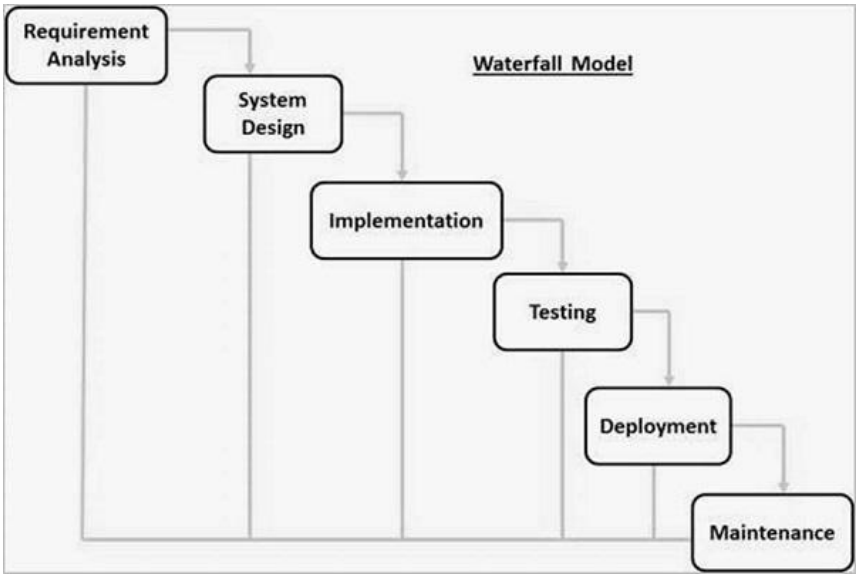
6. **Maintenance**: Changing or updating aspects in this type of model would be relatively cumbersome because of its structure.

Advantages

- Simple and easy to understand.
- Works well for small projects with well-defined requirements.
- Clear milestones and deliverables.

Limitations

- Poor adaptability to changes.
- High risk and uncertainty.
- Late discovery of defects.



3.2 Incremental Development Model

In Incremental Development, an assimilation of the system is done in small and measurable portions in the course of integration of the prototype. Every release brings only a part of the total functionality of the final product which help in delivering it more frequently with feedback from the next.

How Redbus Would Be Developed Using Incremental Model

This paper seeks to assess how Redbus would be developed using the incremental model of program development if it were to be implemented in the organization. Redbus adopted Incremental Model because of its flexibility as it allowed the gradual changes to the business needs to be incorporated step by step and the additional features to be implemented step by step:

1. **First Lunch:** Basic plans which any application must possess like registration of the users, search with respect to the buses and basic ticket booking were incorporated and put into the market.

2. **Second Increment:** The ability to link the system to payment gateway was implemented; real time availability of seats was also implemented.

3. **Third increment:** Additional services, for instance, booking of hotels and special offers were included.

4. **Ongoing Feedback:** While recommending the next steps of the platform, the feedback was collected after each release in order to address the problems.

Advantages

- Early delivery of partial working software.
- Easier to manage risks.
- Enables customer involvement in the whole process; it can be used to take their feedback all through the process from one stage to the other.

Limitations

- Requires detailed planning.
- May increase complexity.
- Integration issues may arise.

3.3 Spiral Model

The very concept of the Spiral Model is based on iteration and risk assessment. Every iteration consists of four steps, namely planning, risk assessment, engineering, and assessment. This model focuses on the identification of possible menace and then the taking of necessary precautions to avoid the menace.

How Redbus Would Be Developed Using Spiral Model

The specific details of how the Redbus would be developed based on the principles of the spiral model are as followed: 1. The development process of Redbus through the spiral model starts with a conception phase that is developmental in nature.

However it is in some high risk areas like data security and integrated marketing solutions the firm adopted the Spiral Model:

those high-risk areas which need to redefine during the planning phase include data privacy, user authentication and scalability.

1. **Planning Phase:** Establish high-raised areas, data privacy, user authentication, and scalability.

2. **Risk Analysis:** Identification of possible risks coupled with strategies to mitigate the occurrence of such risks e.g., deployment of encryption protocols and multi-factor authentication.

3. **Engineering Phase:** Small tests of some features through high-risk components.

4. **Evaluation Phase:** Make extensive reviews of the collected feedback and ensure that functionality meets expectations.

5. **Cycle:** Repeat the process until all critical components are in progress, thus also reducing the risk.

Advantages
- Effective for large, complex projects.
- Focuses on risk management.
- Allows for iterative refinement.

Limitations
- High cost due to repeated iterations.
- Requires expertise in risk management.
- Can be time-consuming.

4. Summary of Comparison:

Criteria	Waterfall Model	Incremental Development	Spiral Model
Flexibility	Low	Moderate	High
Risk Management	Low	Moderate	High
Time to Market	Long	Short for core features	Moderate
Cost	Predictable	Moderate	High
Customer Feedback	Limited	Frequent	Frequent

5. Functional Requirements
- To book an online bus ticket.
- Real-time checking of seat availability.
- Integration with hotel booking services.
- Secure payment gateway.

6. Non-Functional Requirements:
- The system should be highly available and scalable.
- The system should respond quickly.
- System data security and privacy.
- An intuitive user interface.

7. Requirements Validation Strategy:
1. **Review Session:** Review with stakeholders periodically.
2. **Prototyping:** Prototypes for extremely critical scenarios.
3. **Test Cases:** Prepare test cases in order to validate the requirements.
4. **Users Feedback:** Feedback from end-users.

8. Challenges in Requirement Validation:

- **Requirements' vagueness:** The most critical and key area is communication with stakeholders.
- **Resistance from the people against frequent changes:** Some stakeholders may always resist changes.
- **Disappearing Requirements:** Due to the nature of the constant change in the business area, it is subject to be changed continually.

9. Conclusion

Through the investigation, it was found out that no individually recognized SDLC can fit in a blanket requirement for all the scenarios. However, for Redbus, the combination of the Incremental and Spiral Models did wonders. The Incremental Model allowed Redbus to have an early delivery of the product and the opportunity of several releases and updates, whereas the Spiral allowed Redbus to manage its risks more effectively. By appropriately implementing the right SDLC techniques, Redbus was able to go through the scales and even was able to satisfy customer expectations.

10. References

- Redbus official website - https://www.redbus.in
- Wikipedia – Redbus; as edited on 10th February 2025 - https://en.wikipedia.org/wiki/Redbus
- AWS Whitepapers on Redbus's cloud infrastructure.