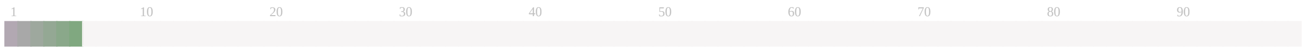
The Report is Generated by DrillBit Plagiarism Detection Software

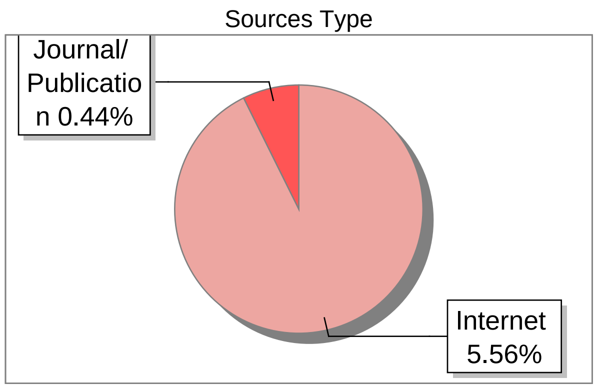
Submission Information

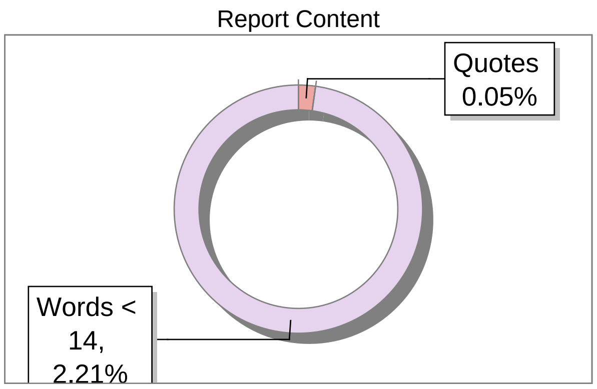
Author Name	Darren D'Souza
Title	Task
Paper/Submission ID	3335354
Submitted by	nnm23is030@nmamit.in
Submission Date	2025-02-16 10:19:27
Total Pages, Total Words	19, 2040
Document type	Project Work

Result Information

Similarity **6 %**







Exclude Information

Quotes	Not Excluded
References/Bibliography	Not Excluded
Source: Excluded < 14 Words	Not Excluded
Excluded Source	**0 %**
Excluded Phrases	Not Excluded

Database Selection

Language	English
Student Papers	Yes
Journals & publishers	Yes
Internet or Web	Yes
Institution Repository	Yes

A Unique QR Code use to View/Download/Share Pdf File

DrillBit

DrillBit Similarity Report

6	**7**	**A**	**A-Satisfactory (0-10%)**
SIMILARITY %	MATCHED SOURCES	GRADE	**B-Upgrade (11-40%)**
			C-Poor (41-60%)
			D-Unacceptable (61-100%)

LOCATION	MATCHED DOMAIN	%	SOURCE TYPE
1	www.softwaretestingstuff.com	3	Internet Data
2	fdokumen.id	1	Internet Data
3	jurnal.upmk.ac.id	1	Internet Data
4	mdpi.comjournalacoustics	<1	Internet Data
5	www.geeksforgeeks.org	<1	Internet Data
6	dochero.tips	<1	Internet Data
7	www.jstage.jst.go.jp	<1	Publication

Analysis of SDLC of Blinkit

-Darren D'souza(NNM23IS040)

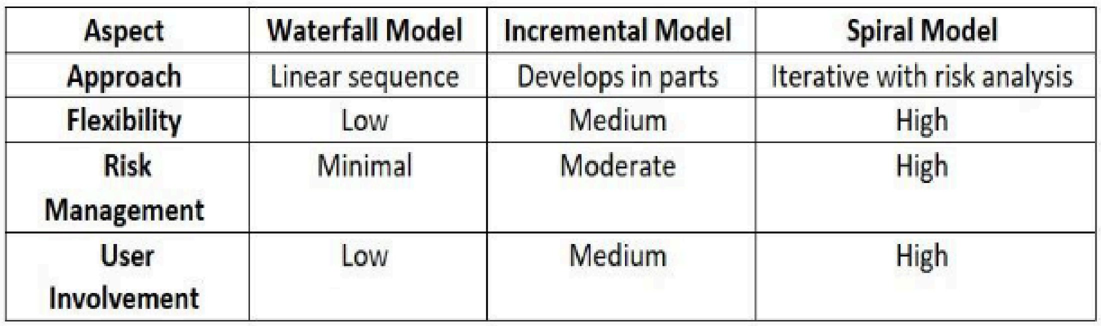
Case study selection:

For the analysis of Software development life cycle (SDLC), a large scale e-commerce platform, healthcare management system or an ERP system is required. I chose "Blinkit", an e-commerce platform to analyse the software development.

About Blinkit:

Blinkit is an Indian quick commerce service and was launched in December 2013. It delivers all kinds of goods, groceries, accessories, etc. One of the main speciality of this platform is that you can have your product delivered anytime and within minutes of ordering, which makes it different from other large scale e-commerce platforms.

Comparison of SDLC models



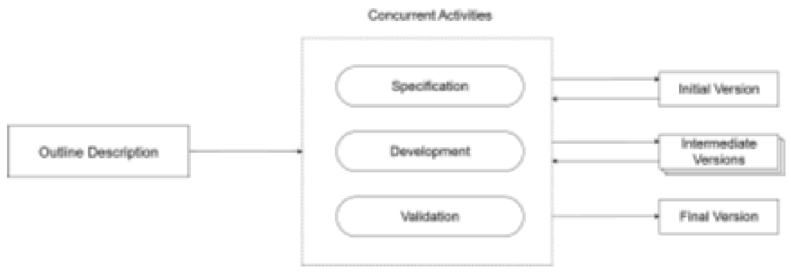
Aspect	Waterfall Model	Incremental Model	Spiral Model
Approach	Linear sequence	Develops in parts	Iterative with risk analysis
Flexibility	Low	Medium	High
Risk Management	Minimal	Moderate	High
User Involvement	Low	Medium	High

Comparative analysis of SDLC models for Blinkit

As a large e-commerce platform, Blinkit should have a well structure SDLC models to ensure efficiency, reliability and sustainability.

The following report comparatively analyses incremental development model, waterfall model and spiral model.

Incremental development



The Incremental Development Model is an iterative SDLC approach where the system is built in small functional increments, each delivering a working version of the software.

Phases of Incremental Development:

1. **Planning**: Identify core functionalities and create a roadmap.

2. **Increment 1:** Develop and deploy the Minimum Viable Product (MVP).

3. **Increment 2, 3, etc.:** Add more features and refine existing ones based on customer feedback.

4. **Final Product:** A complete, optimized, and fully functional system.

Characteristics:

- The system is developed incrementally with each increment giving new features.
- User feedback integration-allows user testing to collect feedback
- Parallel development-Different teams can work on same module simultaneously.
- Risk reduction- Each increment can be tested, hence less risk.
- Adaptability to Change-We can implement new features without redesigning the whole system.

Functional and Non-functional requirements

- **Handling Evolving Functional requirements**
  It introduces new features such as real-time order tracking, personal recomendations, etc. Incremental development allows these to be built and deployed gradually.
- **Non functional requirement optimization**
  Requirements like performance, scalability and security can be Improved incrementally based on real world usage.

1. The first increment may introduce a basic order placement system.

2. Later increments optimize speed by referring to frequently accessed products.

3. Security enhancements like multi-factor authentication (MFA) can Implemented further.

Risk and Change Management

Lower Risk Due to Early Testing
Since each increment is tested individually, critical failures are found out early, leading to less failure in the software.

Flexibility in Change Management
Unlike the Waterfall Model, where late changes are costly, Blinkit can change requirements as per the necessity.

Time and Cost Constraints

Faster Time to Market
Initial increments can be launched early, allowing Blinkit to start user testing before the full system is built.

Cost Efficiency

- Reduces cost by developing only requiredl features first.
- Later increments are built based on actual business needs, avoiding unnecessary expenses.
- Teams work in parallel, optimizing resources.

Example of Incremental Development for Blinkit

Increment 1 – Basic System Launch

User registration, product browsing, and order placement.

Increment 2 – Advanced Features

Real-time tracking and dynamic pricing.

Increment 3 – Performance & Security Upgrades

Enhanced server response time, fraud detection mechanisms.

Increment 4 – AI & Personalization

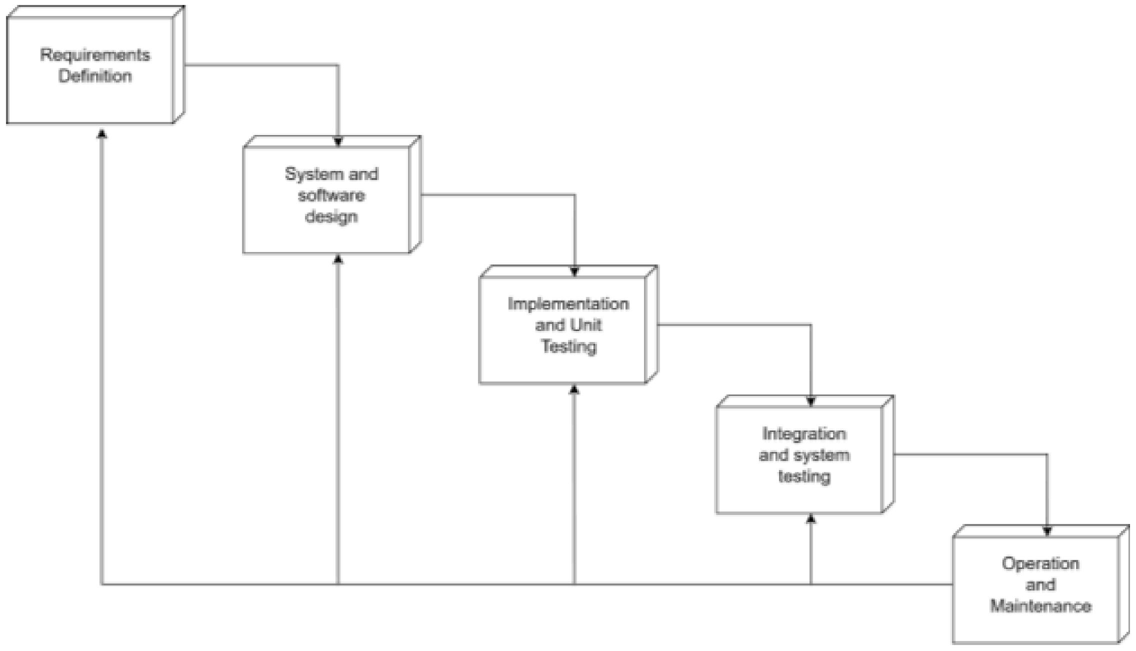
AI-based product recommendations, voice-assisted search.

<u>Why Incremental Model is a Good Fit for Blinkit</u>

- Supports continuous updates
- Allows early deployment of essential features
- Reduces the risk of project failure

However, poor planning of increments can lead to inefficiencies (e.g., if fundamental changes are needed later).

Waterfall model


Requirements Definition
System and software design
Implementation and Unit Testing
Integration and system testing
Operation and Maintenance

The Waterfall Model is a linear and sequential approach to software development where each phase must be completed before the next phase begins. The six main phases of the Waterfall Model are:

1. **Requirement Analysis** – Detailed documentation of requirements.

2. **System Design** – Architecture, UI/UX, and database structure planning.

3. **Implementation** – Coding and development of the software.

4. **Testing** – Debugging and verification of the software.

5. **Deployment** – Releasing the product for customer use.

6. **Maintenance** – Updating and fixing issues post-deployment.

Suitability for Blinkit

While the Waterfall model provides a structured approach, it is not ideal for Blinkit due to its rigid structure and inability to accommodate frequent changes. Since Blinkit operates in a dynamic business environment, requirements frequently evolve based on customer demand, competitor activity, and technological advancements.

Functional and Non-Functional Requirements

Strengths: Well-suited for projects where requirements are clearly defined from the beginning.

Weaknesses: Since Blinkit's software needs continuous updates, the lack of flexibility makes it unsuitable for accommodating evolving requirements.

Risk and Change Management

- High risk as changes late in the development process are expensive and difficult to implement.
- Minimal scope for iteration, making it unsuitable for Blinkit's fast-paced environment.

Time and Cost Constraints

Time: Lengthy development cycles make it difficult to release updates quickly.

Cost: While initial development may be lower in cost, modifications and bug fixes after deployment can be expensive.

Advantages of Waterfall for Blinkit:

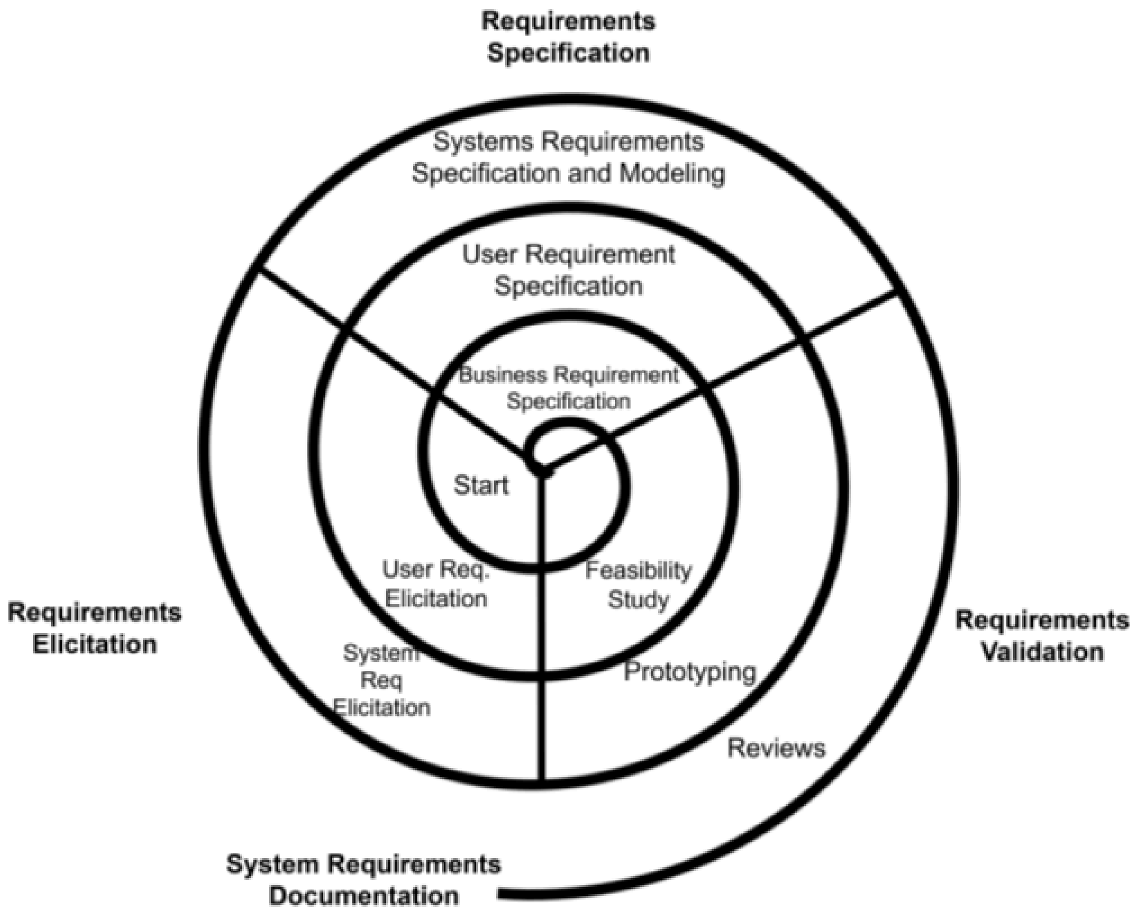
Well-structured development – Suitable if Blinkit were developing a fixed enterprise solution (e.g., warehouse management system).

Clear documentation – Helps new developers understand the system easily.

Why Waterfall is NOT Recommended for Blinkit

Since Blinkit requires agility, real-time updates, and a scalable infrastructure, the Waterfall model's lack of flexibility makes it unsuitable.

Spiral Model


Requirements Specification
Systems Requirements Specification and Modeling
User Requirement Specification
Business Requirement Specification
Start
User Req. Elicitation
Feasibility Study
Requirements Elicitation
Requirements Validation
System Req Elicitation
Prototyping
Reviews
System Requirements Documentation

The Spiral Model is an iterative SDLC approach that combines prototyping with risk management. It is structured into four major phases in each iteration:

1. Planning – Define requirements, objectives, and constraints.

2. Risk Analysis – Identify potential risks (e.g., system overload, security threats).

3. Engineering – Develop and test a prototype or system component.

4. Evaluation – Gather feedback and refine the system before moving to the next iteration.

Each cycle improves upon the previous one, allowing incremental risk reduction.

Suitability for Blinkit

The Spiral Model is highly suitable for Blinkit due to its ability to handle complex requirements and high-risk factors. Blinkit frequently updates its features based on customer behavior, so prototyping and risk assessment in each iteration ensure the platform evolves smoothly.

Functional and Non-Functional Requirements

Strengths: Allows continuous refinement of both functional and non-functional requirements.

Weaknesses: Requires expert risk analysis, which increases complexity.

Risk and Change Management

- Best among the three models due to its built-in risk assessment process.
- Accommodates changes efficiently at every iteration.

Time and Cost Constraints

Time: Can be slower initially due to risk evaluation.

Cost: Higher upfront cost but prevents expensive post-deployment fixes.

Example of Spiral Model for Blinkit

Iteration 1 – MVP Development

Basic grocery ordering platform, limited delivery locations.

Iteration 2 – Risk Evaluation & Expansion

Analyze system load under high user traffic.

Expand operations to more cities.

Iteration 3 – AI & Automation

Introduce AI-driven order suggestions based on user preferences.

Improve inventory prediction models for warehouses.

Iteration 4 – Performance & Security Enhancements

Implement fraud detection algorithms.

Upgrade servers for faster response time.

Why Spiral Model is the Best Fit for Blinkit

- Best risk management strategy
- Supports frequent iterations and feature updates
- Prevents costly redesigns

However, it requires skilled project management and can be complex to implement.

Case Studies of Software Development Models for Blinkit

4 This section presents three case studies illustrating the Waterfall, Incremental, and Spiral models applied to different Blinkit software systems. Each case study analyzes the development process, challenges, and outcomes to determine the suitability of each model.

Case Study 1: Waterfall Model for Blinkit's Warehouse Management System

Background

Blinkit operates a network of warehouses and dark stores to ensure fast and efficient deliveries. A Warehouse Management System (WMS) is crucial for tracking stock levels, managing supplier interactions, and optimizing order fulfillment. Since warehouse operations follow a structured, predefined workflow, the Waterfall Model was chosen for its systematic approach.

SDLC Implementation

The Waterfall Model was implemented in six sequential phases:

1. Requirement Analysis

- Define stock tracking, inventory replenishment, and warehouse automation needs.
- Identify integration points with supplier systems and delivery networks.

2. System Design

- Create database schemas for inventory data and an API structure for communication with Blinkit's mobile and web platforms.
- Plan barcode scanning and automated stock updates.

3. Implementation

Developers build inventory dashboards, supplier portals, and automated order allocation features.

4. Testing

QA team verifies accuracy in stock updates, order fulfillment speed, and system security.

5. Deployment

Initial launch in selected warehouses, followed by a nationwide rollout.

6. Maintenance

Post-launch updates include AI-powered demand forecasting and better stock optimization tools.

Results and Challenges

Success:

- The system provided accurate, real-time inventory tracking.
- Well-documented processes allowed easy training for new employees.

Challenges:

- Slow adaptability to changing business needs.
- Required long development cycles, delaying new feature implementations.

Conclusion

The Waterfall Model worked well for a structured system like warehouse management, where requirements were clear and stable. However, it was too rigid for dynamic business needs, making it unsuitable for customer-facing applications requiring frequent updates.

Case Study 2: Incremental Development Model for Blinkit's Mobile App

Blinkit's mobile app is the primary platform for customer interactions, handling order placement, tracking, payments, and customer support. Given the need for frequent updates, feature enhancements, and real-time fixes, the Incremental Development Model was chosen to allow gradual improvements while keeping the app fully functional.

SDLC Implementation

The Incremental Model allowed Blinkit to release core features first, followed by continuous improvements:

1. **Increment 1** – Core Functionality Launch

- Basic features like user registration, product browsing, cart, and checkout were developed.
- The app was launched with an MVP (Minimum Viable Product), ensuring early market entry.

2. **Increment 2** – Real-Time Order Tracking & Notifications

- A GPS-based tracking system was introduced.
- SMS and push notifications were added for order status updates.

3. **Increment 3** – Payment Enhancements

- Secure integration with UPI, wallets, credit cards, and COD options.
- Additional fraud prevention measures were implemented.

4. **Increment 4** – AI and Personalization

- AI-powered product recommendations based on user preferences.
- Dynamic pricing based on demand and supply conditions.

Results and Challenges

Success:

- Allowed rapid feature deployment without affecting app stability.
- Continuous customer feedback helped optimize user experience.

Challenges:

- Some increments required major refactoring, leading to technical debt.
- Scalability issues emerged as new features were added.

Conclusion

The Incremental Model proved highly effective for Blinkit's app, ensuring fast updates and customer-driven improvements. However, poor planning of increments could lead to inefficiencies. Regular refactoring was required to maintain performance and scalability.

Case Study 3: Spiral Model for Blinkit's Fraud Detection System

Background

With thousands of daily transactions, Blinkit faced challenges related to payment fraud, fake orders, and account takeovers. A Fraud Detection System (FDS) was needed to analyze transaction patterns, detect anomalies, and prevent fraudulent activities. Due to the high-risk nature of fraud prevention, the Spiral Model was chosen, allowing continuous risk assessment and iterative improvements.

SDLC Implementation

The Spiral Model followed an iterative approach, combining prototyping and risk analysis:

1. Iteration 1 – Planning & Risk Analysis

Identify major fraud risks:

- Fake orders from bots
- Unauthorized payment attempts
- Account hacking

Set risk mitigation strategies (e.g., real-time fraud alerts).

2. Iteration 2 – Prototype Development

- Develop a basic fraud detection algorithm using rule-based detection.
- Apply user behavior analysis to flag suspicious activities.

3. Iteration 3 – Testing & Refinement

- Train machine learning models on real transaction data.
- Optimize for speed and accuracy in fraud detection.

4. Iteration 4 – Deployment & Continuous Monitoring

- Implement real-time fraud detection in the production environment.
- Continuously update the AI model based on new fraud patterns.

Results and Challenges

Success:

- High fraud detection accuracy, reducing fraudulent transactions.
- Adaptive learning mechanism improved fraud detection over time.

Challenges:

- Required constant monitoring and regular updates.
- Initial high development cost due to extensive risk assessment.

Conclusion

The Spiral Model was the best choice for Blinkit’s Fraud Detection System due to its iterative risk evaluation and flexibility. However, it required skilled risk analysis and continuous updates, making it more resource-intensive than other models.

Overall Conclusion of the SDLC Models

This report analyzed three SDLC models for Blinkit. The Waterfall Model is unsuitable due to its rigid structure. The Incremental Model is a good alternative, but the Spiral Model offers the best combination of risk management, flexibility, and iterative refinement.

By adopting the Spiral Model, Blinkit can:

- Ensure high reliability and security.
- Release features in small, controlled iterations.
- Minimize system failures and improve customer satisfaction.

Thus, the Spiral Model is the most effective SDLC approach for Blinkit's evolving business and technological needs. By adopting the Spiral Model, Blinkit can deliver a scalable, high-performance, and secure platform, meeting its business and user needs effectively.

Links:

Visit my github