Course Name: Introduction to Software Engineering

Course Code: IS1103-1

Title: Online Shopping

Student Name: Bhoomika Shetty

USN: NNM25IS504

Semester & Section: 4th Sem A

Department: Information Science & Engineering

Institution Name: NMAMIT Institute of Technology

Academic Year: 2026

Faculty Name: Dr. Jason Elroy Martis

Date of Submission: 13-02-2026

Abstract

This report analyzes how software development process models influence the success of a large scale real world system. The selected case study is a modern e commerce platform that behaves similar to Amazon or Flipkart. These platforms operate in highly dynamic environments where user expectations change constantly. Because of this, requirements engineering becomes a continuous activity rather than a single phase. The study compares three major process models, Waterfall, Incremental, and Spiral, focusing on how each model manages functional requirements, non functional requirements, risk, cost, and adaptability. A simplified requirements document is developed to simulate real project conditions. The analysis shows that rigid linear models struggle in fast changing systems. Iterative and risk driven approaches perform better. Among the compared models, Incremental development provides the best balance between flexibility, delivery speed, and resource control. The report concludes that successful systems depend not only on technical design but on how well requirements are understood, validated, and evolved over time. [1]

GitHub Repository Link

GitHub URL: http://github.com/Bhoomishetty/ISE-TASK-1

Introduction

Most people imagine software development as writing code until the application works. That idea is incomplete. Real software engineering is about planning for change. Every serious system changes after deployment. New features are demanded. Bugs appear. Market conditions shift. A system that cannot evolve becomes useless surprisingly fast.[1]

This assignment explores how different software process models react to change. The focus is not just theory. The goal is to see how these models behave when applied to a real large scale system. The selected case study is an e commerce platform. These platforms are perfect examples of living software. They expand continuously. They never freeze.[5]

The assignment compares three models that represent different philosophies. The Waterfall model believes in order and strict sequence. The Incremental model believes in gradual growth. The Spiral model believes in controlled risk and repeated evaluation. Each model reflects a different mindset about uncertainty.[2]

This study directly connects to the syllabus unit on Introduction and Requirements Engineering. It demonstrates that requirements are not static documents. They are conversations that continue throughout the lifecycle of the product. Ignoring this fact is usually the beginning of project failure.[7]

Problem Statement

The task is to evaluate how process models influence the management of requirements in a large evolving software system. The comparison must be practical, not just descriptive. The chosen case study should reveal strengths and weaknesses clearly.

The expected outcomes include a comparative study, a simplified requirements document, and an explanation of validation challenges. The report should also recommend which model suits the system best.

Certain assumptions are made to keep the analysis realistic. The system is assumed to serve millions of users. Requirements change frequently due to business competition. Security and reliability are mandatory. Budget exists but is limited. Deadlines are tight. These conditions mirror real industry projects.[1][7]

The challenge is not building the system once. The challenge is keeping it alive while everything around it changes.

Case Study Description

The selected case study is a large scale e commerce platform similar to Amazon. Such platforms are not simple websites. They are distributed ecosystems. Every feature depends on dozens of hidden components.

When a user searches for a product, multiple services activate at the same time. Inventory systems check stock. Recommendation engines predict preferences. Payment systems prepare transaction pipelines. Logistics systems estimate delivery. All this happens in seconds. The user only sees a clean interface. Underneath, it is chaos carefully organized.[5]

Functional requirements include user registration, authentication, product browsing, filtering, shopping cart management, payment processing, order tracking, and customer support integration. These features define the visible behavior of the platform.[3]

Non functional requirements are even more demanding. The system must support millions of concurrent users. It must remain available twenty four hours a day. Response time must stay low even during peak sales. Security must protect financial data. Reliability is critical. A single outage can damage brand trust permanently.[1][7]

These requirements do not remain stable. During festive seasons the system must scale aggressively. New payment methods must be added quickly. Regulatory policies may force changes in data handling. Competitors introduce features that customers expect immediately. This constant pressure makes the choice of process model extremely important.[2][4]

Methodology and Process Model Comparison

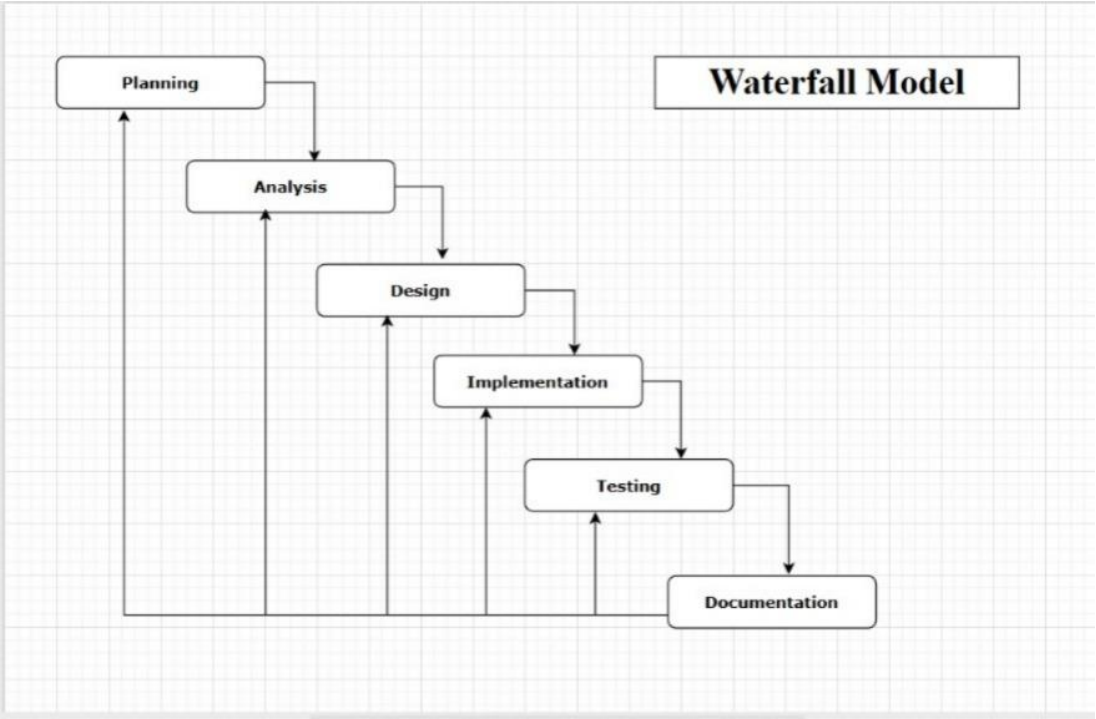
To compare the models, imagine building the e commerce system from scratch under each approach. Observe how each model reacts to uncertainty.

Waterfall Model

The Waterfall model follows a strict sequence. Requirements are collected fully before development begins. After approval, design is created. Then implementation. Testing happens near the end. Each phase assumes the previous phase was correct.[2]

This model feels safe at the beginning. Documentation is clear. Milestones are predictable. Management likes this structure. However, the illusion of certainty disappears once requirements change. In an e commerce environment they always change. Returning to earlier phases becomes expensive and politically difficult. Teams hesitate to admit mistakes. Problems hide until late testing stages.[1]

If Waterfall were used for the case study, the first release might already be outdated by launch. Competitors would release new features while the system is stuck in documentation cycles. The model punishes flexibility.[4]



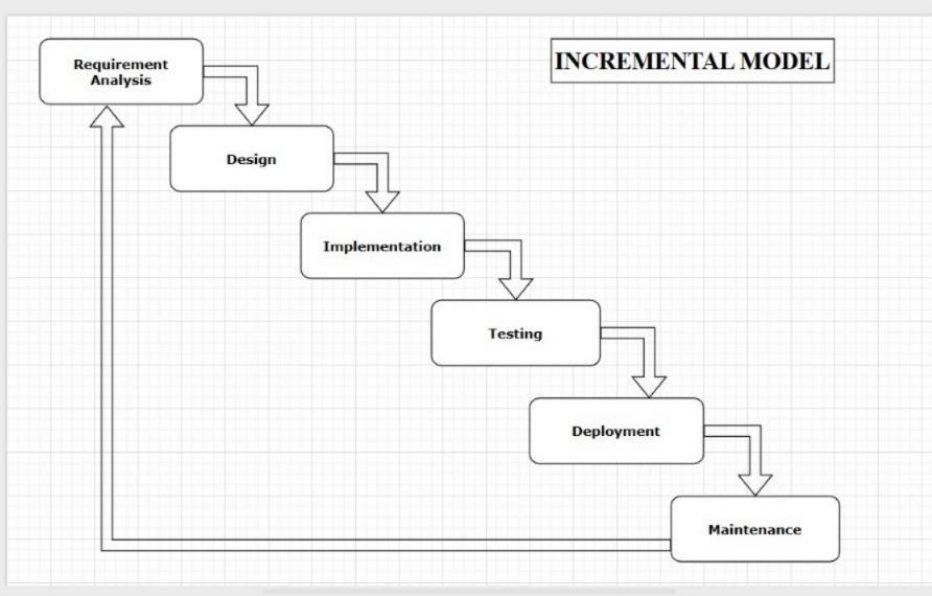
Incremental Development Model

The Incremental model breaks the system into smaller deliverable units. Each increment produces a working version of the platform. Users interact with early releases. Feedback arrives immediately.

In the first increment, basic browsing and registration may be implemented. The second increment adds payment integration. Later increments introduce recommendations, analytics, and optimization. Each cycle teaches the team something new.[2]

This approach matches how modern software actually grows. It accepts that understanding improves over time.[4] Mistakes are cheaper because they are discovered early. Stakeholders feel involved. Risk is distributed across releases instead of accumulating silently.

For the e commerce platform, Incremental development allows the business to launch quickly and expand gradually. Revenue starts earlier. Real user data guides improvement. The model embraces evolution rather than resisting it.[6]

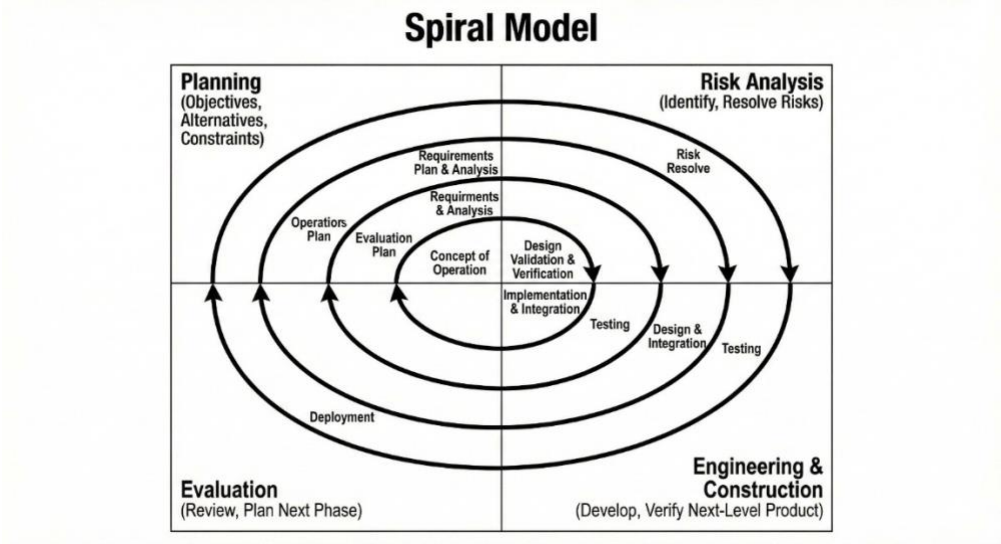


Spiral Model

The Spiral model combines iteration with structured risk analysis. Every cycle begins with planning and ends with evaluation. The model is powerful for complex high risk systems such as aerospace or defense software.[1]

Applying Spiral to the e commerce case provides excellent risk visibility. Security risks, scalability threats, and integration failures can be identified before implementation. However, this safety comes with cost. Spiral requires expert management and heavy documentation. Smaller teams may struggle to sustain its discipline.[7]

Spiral is ideal when failure is unacceptable. For commercial platforms, the cost may outweigh the benefits unless the organization is extremely large.[2]



Comparative Insight

Waterfall prioritizes order but sacrifices adaptability. Spiral prioritizes risk control but increases overhead. Incremental balances flexibility and practicality. For systems that grow continuously, balance matters more than perfection.

Implementation Details and Requirements Engineering

A simplified requirements engineering exercise was conducted in two stages to simulate project evolution.

The initial requirements version focused on essential functionality. User login, product listing, basic cart management, and manual payment confirmation formed the core system. The aim was to create a minimal viable product quickly.[3]

After stakeholder review sessions, the requirements expanded. Automated secure payment integration became necessary. Real time order tracking was requested. Recommendation features were added to increase engagement. Administrative analytics tools were included for business monitoring. Mobile responsiveness became mandatory because most users access platforms through phones.

These revisions demonstrate a simple truth. Requirements are never complete. They mature as understanding grows.[7]

Validation strategies included stakeholder interviews, walkthrough meetings, and prototype demonstrations. Even with structured validation, conflicts appeared. Marketing teams wanted aggressive feature expansion. Security teams demanded strict compliance. Budget managers resisted cost increases. Requirements engineering became a negotiation process, not just technical writing.[3][7]

Ambiguity was a constant problem. Words like fast, secure, and scalable mean different things to different stakeholders. Clarifying these expectations consumed significant effort.[1]

Results and Analysis

The comparative study revealed patterns that align with industry experience.

Waterfall performed poorly in dynamic scenarios. Late discovery of errors increased rework. Stakeholders felt disconnected from development progress. The model assumed stability that did not exist.[2]

Spiral handled risk extremely well. Critical threats were identified early. However, the management burden slowed delivery. The system advanced cautiously but expensively.[1]

Incremental development produced steady progress. Stakeholders remained engaged. Early releases generated valuable feedback. Risk remained manageable without overwhelming overhead. The model encouraged learning instead of pretending to know everything from the start.[4][6]

For an evolving e commerce platform, Incremental development emerges as the most practical choice. It supports continuous growth, adapts to market pressure, and controls cost. Spiral is technically superior in risk handling but financially heavy. Waterfall is too rigid for competitive environments.[5]

Incremental development matches the rhythm of modern software ecosystems.

Conclusion

This assignment demonstrates that process models shape the destiny of software systems. Requirements engineering is not paperwork. It is strategic thinking about uncertainty. Systems fail not because code is wrong but because assumptions were frozen too early.

Waterfall represents confidence in prediction. Spiral represents caution through analysis. Incremental represents acceptance of change. In fast moving industries, acceptance of change becomes an advantage.

The e commerce case study shows that flexibility is not optional. It is survival. Requirements evolve with user behavior, technology, and competition. A process model must allow evolution without chaos. Incremental development achieves this balance.

The main lesson is uncomfortable but honest. Software is never finished. It is maintained, adjusted, expanded, and repaired continuously. Engineering methods must respect this reality.

References

[1] I. Sommerville, Software Engineering, Lecture Notes, University of St Andrews, 2016. https://software-engineering-book.com/

[2] R. S. Pressman, "Software Engineering Notes and Slides," Open educational material. https://cs.gmu.edu/~offutt/classes/cs306/slides/

[3] IEEE Computer Society, IEEE Recommended Practice for Software Requirements Specifications, Overview Summary. https://standards.ieee.org/ieee/830/1222/

[4] Agile Alliance, "Manifesto for Agile Software Development," 2001. https://agilemanifesto.org/

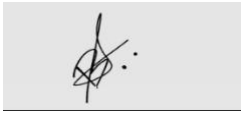
[5] Amazon Web Services, "AWS Architecture Center: Building Scalable E-commerce Systems." https://aws.amazon.com/architecture/

[6] M. Fowler, "The New Methodology," Martin Fowler Articles. https://martinfowler.com/articles/newMethodology.html

[7] Carnegie Mellon University, Software Engineering Institute, "Requirements Engineering Overview." https://www.sei.cmu.edu/our-work/requirements-engineering/

Declaration of Academic Integrity

I hereby declare that this assignment is my original work and has not been copied or plagiarized from any source. All references and resources used have been properly cited. Any ambiguities will lead to forfeiting my marks.

Student Signature: 

Date: 13-02-2026