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

A Comparative Study of Different Models in Relation to Walmart's Software Development

Akash A Mogaveera

NMAMIT

[nnm23is007@nmamit.in](mailto:nnm23is007@nmamit.in)

GitHub Repository: https://github.com/akashmogaveera/Walmart

**Abstract**

Software development is a critical component for modern retail operations, requiring structured approaches for building scalable, secure, and efficient systems. This report explores the Software Development Life Cycle (SDLC) models applicable to Walmart, a global leader in retail and e-commerce. It provides a comparative analysis of SDLC methodologies and an overview of requirements engineering. The study aims to offer insights into the selection of appropriate SDLC models for large-scale retail platforms, emphasizing the waterfall, incremental development, and spiral model approaches. The report also discusses challenges and strategies involved in requirements validation and software deployment at Walmart.

**1. Introduction**

Walmart is one of the world's largest retailers, operating both physical stores and e-commerce platforms globally. The company relies on robust software systems to manage inventory, process transactions, analyze customer data, and operate its e-commerce platform. Given its large-scale nature, Walmart requires an efficient software development lifecycle (SDLC) to manage continuous updates, new features, and system stability across its retail technology stack.

**2. Overview of Walmart**

2.1 System Overview

Walmart operates a complex technological ecosystem that includes:

- Point of Sale (POS) systems across thousands of stores

- E-commerce platform (Walmart.com)

- Inventory management systems

- Supply chain management software

- Customer relationship management (CRM) systems

- Data analytics platforms

- Mobile applications for customers and employees

2.2 Technologies Used

- Cloud Platform: Hybrid cloud infrastructure using private data centers and public cloud services

- Architecture: Microservices-based architecture

- Database: Various databases including Oracle, MongoDB, and Cassandra

- Development Stack: Java, Python, React, Node.js

- DevOps: Jenkins, Docker, Kubernetes

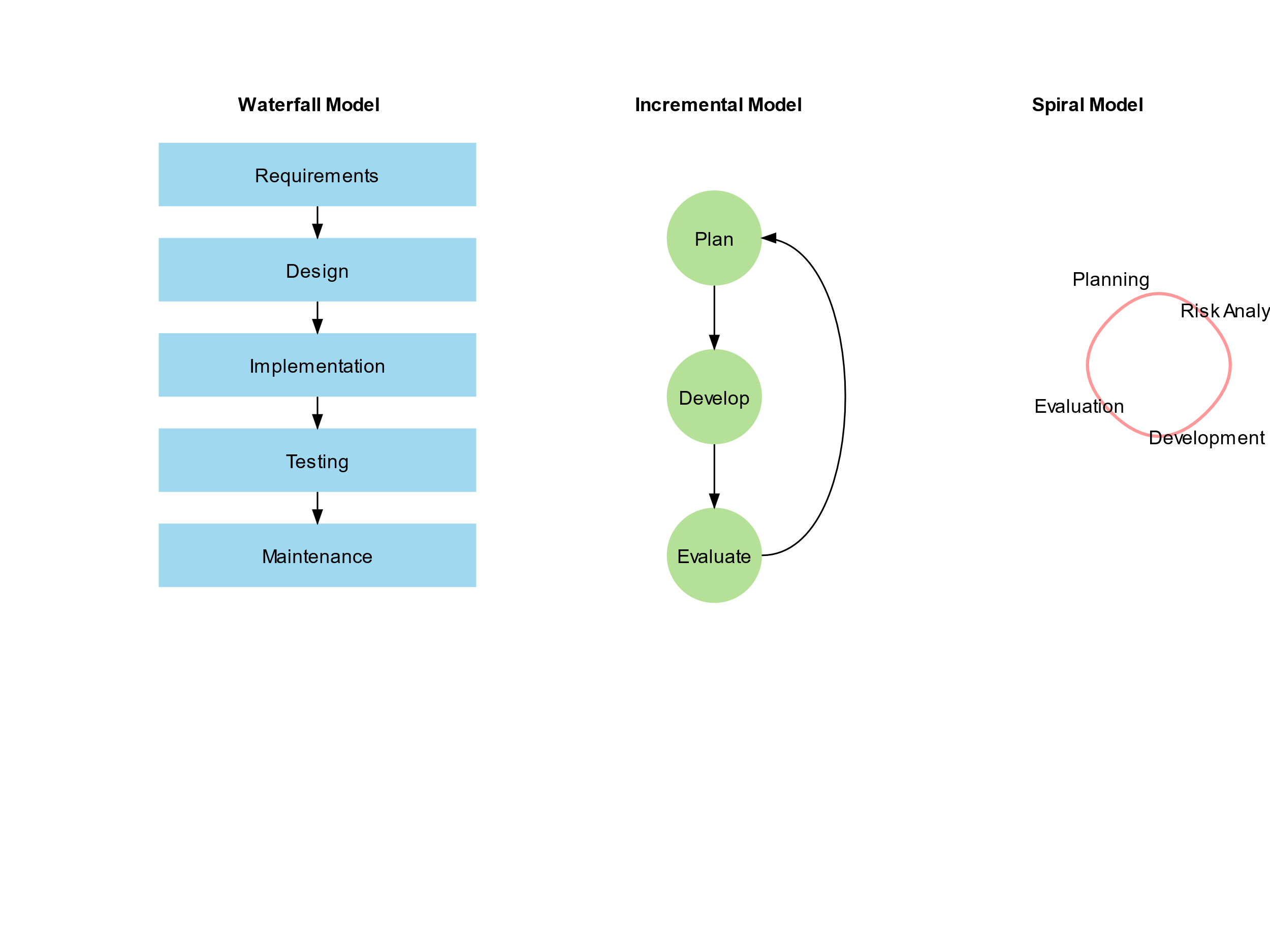
- Analytics: Hadoop ecosystem, Machine Learning platforms

- Security: Advanced encryption and authentication systems

**3. Comparative Analysis of SDLC Models**

3.1 Waterfall Model

**How Walmart Would Be Developed Using Waterfall:**

****

1. Phase 1: Requirements Definition

- Define all system requirements upfront

- Document POS system specifications

- Detail inventory management requirements

- Specify e-commerce platform features

- Outline security and compliance needs

2. Phase 2: System & Software Design

- Design complete system architecture

- Plan database structures

- Define API endpoints

- Design user interfaces

- Plan security implementations

3. Phase 3: Implementation

- Develop all system components

- Build database systems

- Implement security measures

- Create user interfaces

- Develop integration points

4. Phase 4: Integration and System Testing

- Integrate all components

- Test system functionality

- Perform security testing

- Conduct performance testing

- Validate compliance requirements

5. Phase 5: Operation and Maintenance

- Deploy to production

- Monitor system performance

- Address issues and bugs

- Maintain system stability

**Suitability for Walmart:**

Pros:

- Clear documentation and structure

- Well-defined phases

- Suitable for stable, well-understood requirements

Cons:

- Lack of flexibility for changing retail needs

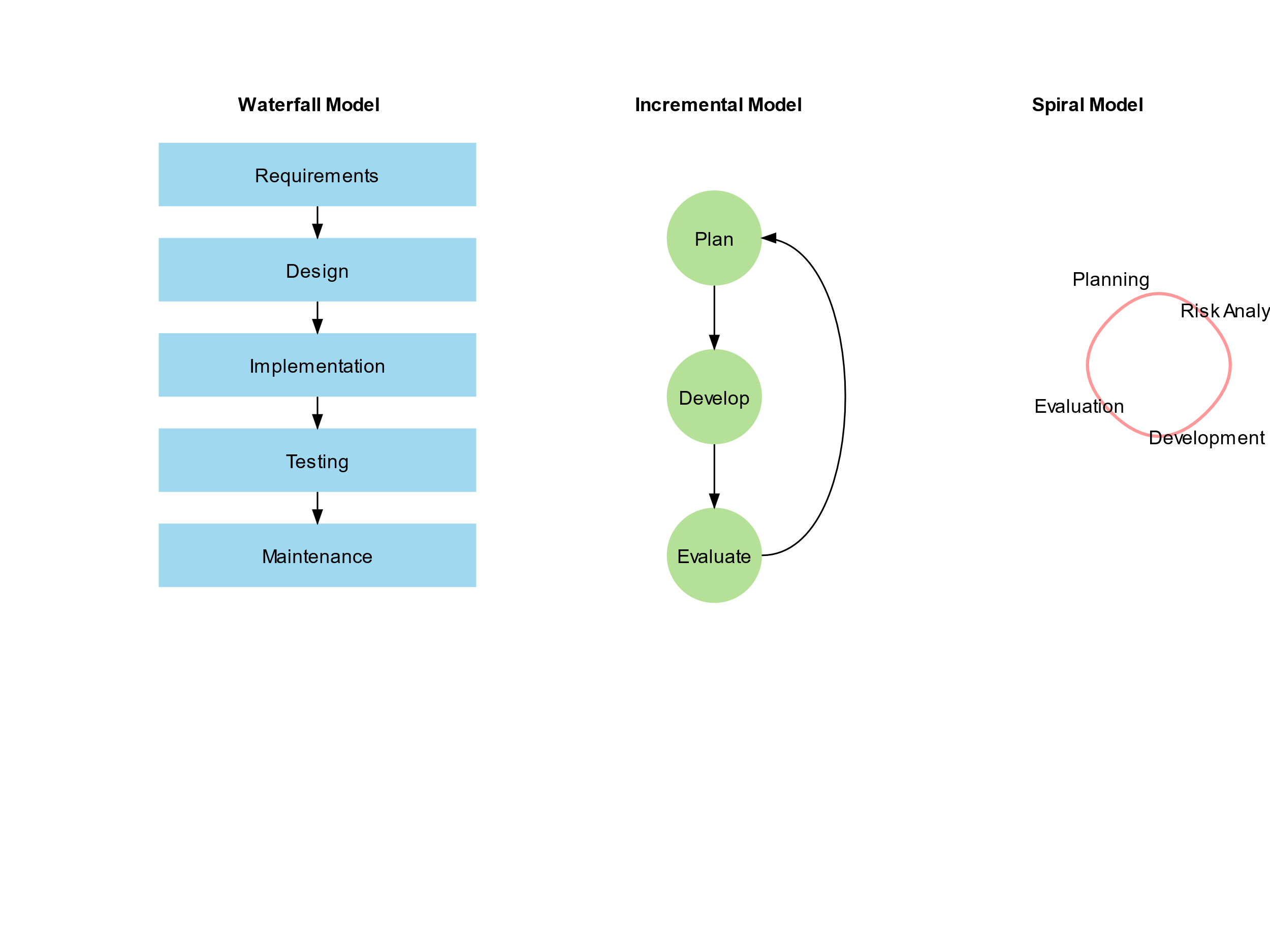
- Slow response to market changes

- Difficult to modify requirements once set

Verdict: Not suitable for Walmart's dynamic retail environment

3.2 Incremental Development Model

**How Walmart Uses Incremental Development:**

****

1. Phase 1: Planning and Initial Requirements

- Identify core features (POS, inventory)

- Prioritize development needs

- Plan initial architecture

2. Phase 2: Feature-wise Development

- Develop core features first

- Add additional capabilities incrementally

- Continuous integration of new features

3. Phase 3: Continuous User Feedback

- Gather feedback from store employees

- Monitor customer interactions

- Analyze system performance

4. Phase 4: System Testing and Refinement

- Test new features

- Perform integration testing

- Security validation

5. Phase 5: Deployment and Monitoring

- Phased rollout to stores

- Monitor system performance

- Gather usage metrics

6. Phase 6: Iteration and Maintenance

- Continuous improvement

- Feature enhancement

- System optimization

**Suitability for Walmart:**

Pros:

- Rapid deployment of features

- Flexible to changing retail needs

- Continuous improvement

- Better risk management

Cons:

- Complex integration requirements

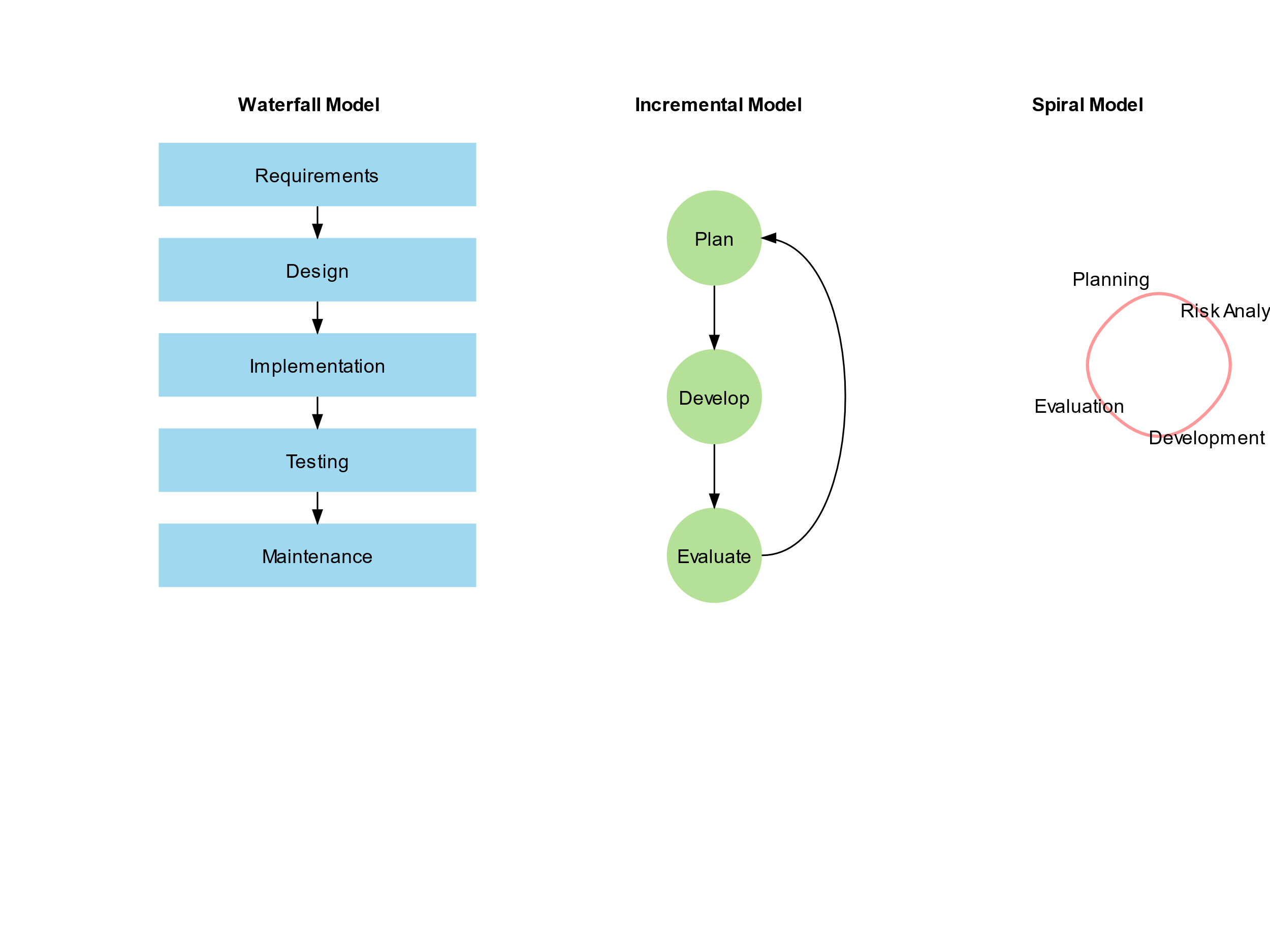
- Needs strong version control

- Resource intensive

Verdict: Highly suitable for Walmart's retail operations

3.3 Spiral Model

**How Walmart Would Be Developed Using Spiral:**

****

1. Phase 1: Risk Analysis and Prototyping

- Analyze security risks

- Evaluate performance requirements

- Create prototypes

2. Phase 2: Concept Validation

- Test prototypes

- Validate business requirements

- Assess technical feasibility

3. Phase 3: Development and Testing

- Iterative development

- Continuous testing

- Risk mitigation

4. Phase 4: Deployment

- Phased rollout

- Performance monitoring

- Security validation

**Suitability for Walmart:**

Pros:

- Strong risk management

- Suitable for complex features

- Thorough validation process

Cons:

- Time-consuming

- Expensive

- Complex for simple features

Verdict: Suitable for high-risk components like payment systems

**4. Requirements Engineering for Walmart**

4.1 Functional Requirements

- Point of Sale Systems

- Transaction processing

- Payment integration

- Receipt generation

- Returns processing

- Inventory Management

- Real-time stock tracking

- Automated reordering

- Supplier integration

- Warehouse management

- E-commerce Platform

- Product catalog

- Shopping cart

- Checkout process

- Order management

- Customer Management

- Customer profiles

- Loyalty programs

- Purchase history

- Customer support

4.2 Non-Functional Requirements

- Scalability

- Handle high transaction volumes

- Support peak shopping periods

- Scale across multiple locations

- Security

- PCI compliance

- Data encryption

- Fraud detection

- Access control

- Performance

- Fast transaction processing

- Quick page load times

- Responsive interfaces

- Availability

- 24/7 system uptime

- Disaster recovery

- Backup systems

4.3 Requirements Validation Strategy

- User Testing

- Store employee feedback

- Customer usability testing

- Performance monitoring

- Security Validation

- Penetration testing

- Compliance audits

- Security reviews

- Performance Testing

- Load testing

- Stress testing

- Scalability validation

4.4 Challenges in Requirements Validation

- Complex Integration Needs

- Diverse User Base

- Security Compliance

- Performance at Scale

- Global Operations

**5. Conclusion**

Based on the analysis, Walmart's software development benefits most from a combination of Incremental Development for regular features and the Spiral Model for high-risk components. The Waterfall model proves too rigid for the dynamic retail environment. Success depends on strong requirements engineering practices and effective validation strategies.

**6. References**

1. Walmart Technology Blog

2. Retail Software Development Best Practices

3. Enterprise SDLC Methodologies

4. Retail Technology Standards

5. E-commerce Platform Development Guidelines