LTCE

SOFTWARE ENGINEERING LABORATORY

Experiment 1

Applications of Process Models & Packet Sniffer Case Study

Subject: Software Engineering

Course: Third Year CSE (IoT & CSBT)

College: Lokmanya Tilak College Of Engineering (LTCE)

University: Mumbai University

Academic Year: 2025-26

Group Members:

1. Raj Dubey - Roll No: 11

2. Yash Maurya - Roll No: 25

3. Piyush Kanojiya - Roll No: 23

Date: August 11, 2025

TABLE OF CONTENTS

AIM AND OBJECTIVES

🎯 Primary Aim:

To study and analyze various software process models and their practical applications in real-world software development projects.

📋 Specific Objectives:

A. **Study and document applications of various process models:**
   - Waterfall Model
   - Incremental Model
   - Evolutionary Model
   - Agile Model

B. **Develop a detailed problem statement** for a selected case study (Packet Sniffer Software)

C. **Analyze and justify the selection** of the most suitable process model for the case study

🎓 Learning Outcomes:

- Understand the characteristics and applications of different software process models
- Develop skills in selecting appropriate development methodologies
- Learn to justify technical decisions with proper reasoning
- Gain experience in software project planning and analysis

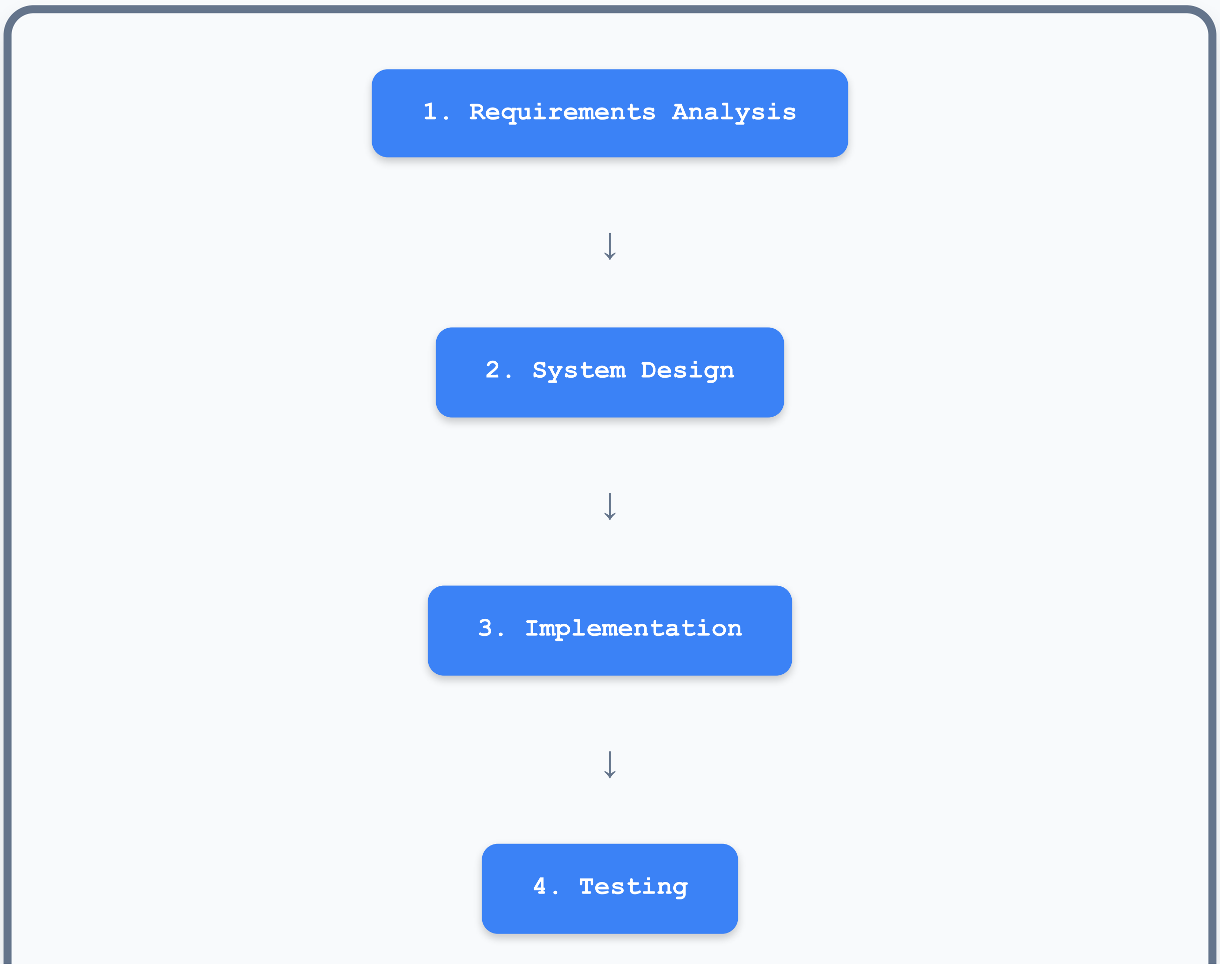
PART A: APPLICATIONS OF VARIOUS PROCESS MODELS

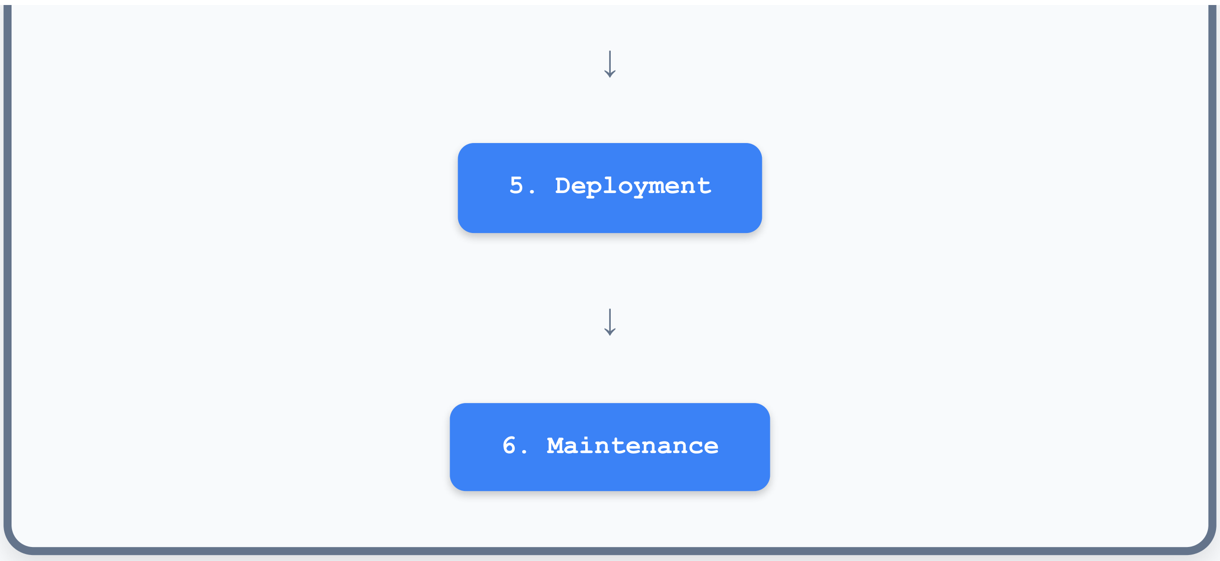
1. 🏗️ WATERFALL MODEL

Definition:

The Waterfall model is a linear sequential design process where progress flows steadily downward through distinct phases: Requirements Analysis, System Design, Implementation, Testing, Deployment, and Maintenance. Each phase must be completed before the next phase begins.

Process Flow Diagram:





Key Characteristics:

- **Sequential Phases:** No overlap between phases
- **Documentation Heavy:** Extensive documentation at each stage
- **Linear Progression:** Cannot return to previous phases
- **Late Testing:** Testing occurs only after implementation
- **Fixed Requirements:** Changes are difficult and expensive

Real-World Applications:

Domain	Example Projects	Why Waterfall?
Defense Systems	Missile guidance systems, Military communication software	Strict security requirements, Fixed specifications
Banking	Core banking systems, ATM software	Regulatory compliance, High reliability needs
Healthcare	Medical device software, Hospital management systems	FDA approvals, Patient safety critical
Aerospace	Flight control systems, Navigation software	Safety critical, Extensive testing required
Government	Tax processing systems, Voter registration	Stable requirements, Audit trails needed

✅ **Advantages:**

- Simple and easy to understand
- Well-defined project milestones
- Good for projects with stable requirements
- Extensive documentation helps maintenance
- Quality gates at each phase
- Easy to manage and track progress

❌ **Disadvantages:**

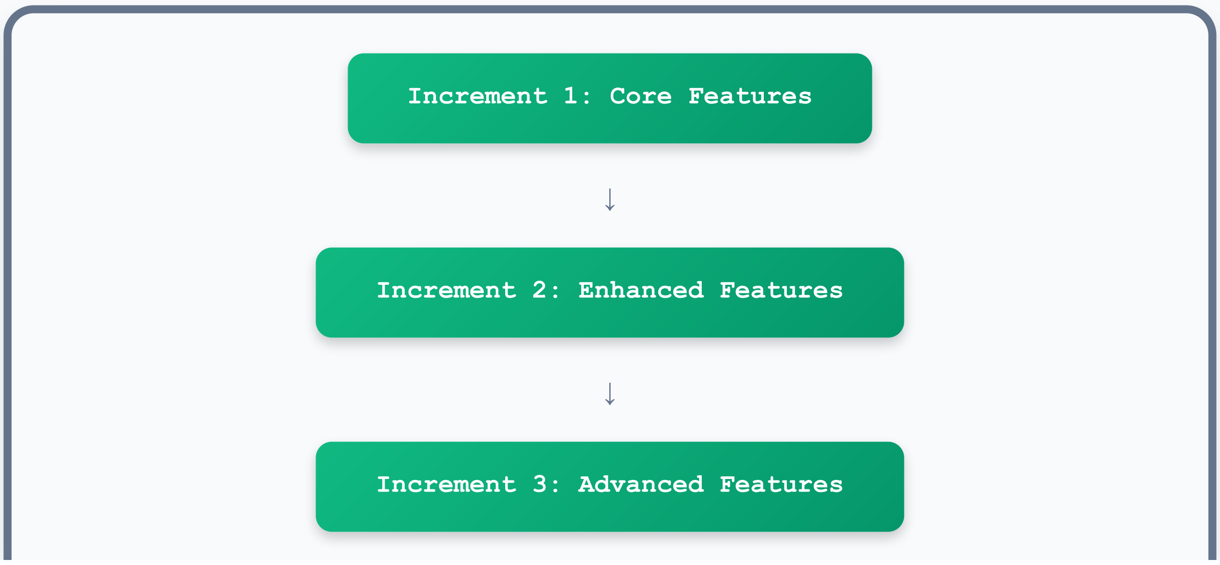
- No flexibility for requirement changes
- Late discovery of defects
- Customer sees product only at the end
- High risk for complex projects
- Cannot accommodate changing technology
- Long development cycles

2. 🔄 INCREMENTAL MODEL

Definition:

The Incremental model combines the linear sequential model with iterative philosophy. The software is developed in incremental builds, where each increment adds new functionality to the previous version, allowing early delivery of working software.

Process Flow Diagram:


Increment 1: Core Features
Increment 2: Enhanced Features
Increment 3: Advanced Features

↓

Continue until complete...

↓

Final Integrated System

Key Characteristics:

- **Multiple Increments:** Software delivered in multiple working versions
- **Early Delivery:** Core functionality available early
- **Parallel Development:** Multiple increments can be developed simultaneously
- **Incremental Integration:** Each increment builds upon previous ones
- **User Feedback:** Early user input guides subsequent increments

Real-World Applications:

Domain	Example Projects	Increment Strategy
ERP Systems	SAP, Oracle ERP	Module-wise: HR → Finance → Inventory → Sales
E-commerce	Amazon, Flipkart	Catalog → Cart → Payment → Shipping → Reviews
CRM Systems	Salesforce, HubSpot	Contacts → Leads → Opportunities → Reports
LMS Platforms	Moodle, Canvas	User Management → Content → Assessment → Analytics
Payroll Systems	Workday, ADP	Employee Data → Salary → Tax → Reports

✅ **Advantages:**

❌ **Disadvantages:**

- Early delivery of working software
- Easier testing and debugging
- Lower initial delivery cost
- Reduced risk of project failure
- Customer feedback incorporation
- Parallel development possible

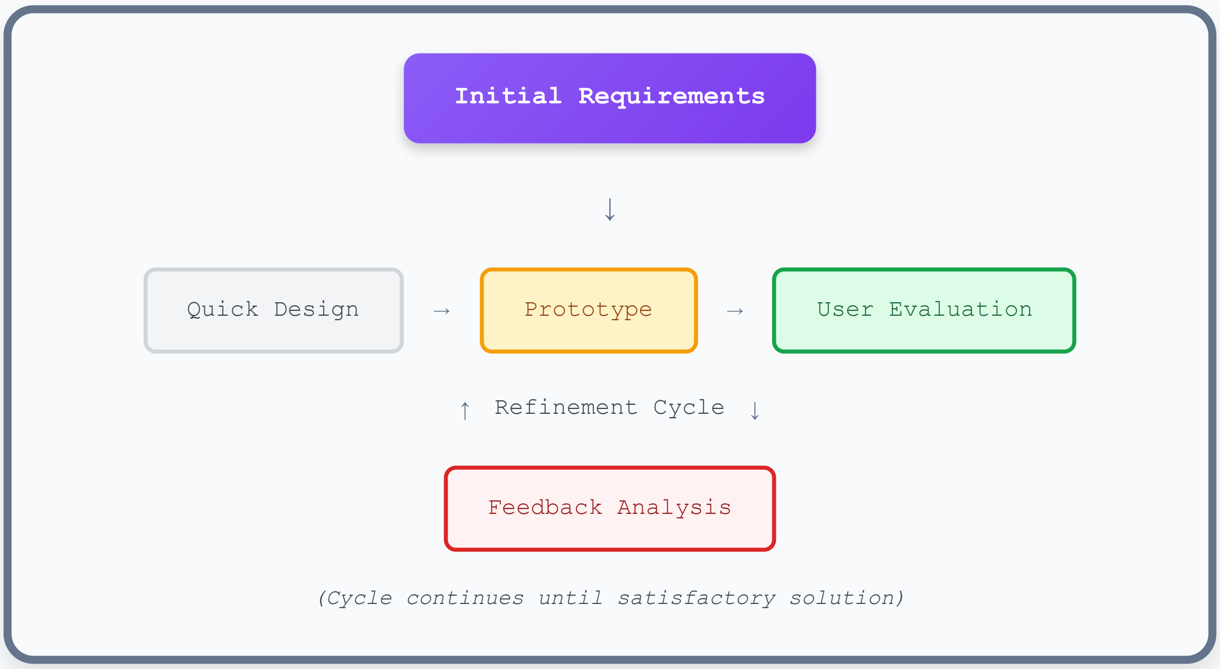
- Requires careful planning and design
- May need more resources
- Integration complexity increases
- Total cost may be higher
- Architecture must support increments
- May lead to rushed increments

3. 🌱 EVOLUTIONARY MODEL

Definition:

The Evolutionary model allows software to evolve over multiple cycles of development. Each cycle involves rapid development of a prototype, user evaluation, and refinement based on feedback. The system grows and evolves through successive iterations.

Process Flow Diagram:



Key Characteristics:

- **Iterative Refinement:** Continuous improvement through cycles
- **User-Centric:** Heavy emphasis on user feedback
- **Rapid Prototyping:** Quick development of working models
- **Flexible Requirements:** Requirements evolve with understanding
- **Risk Reduction:** Early identification of problems

Real-World Applications:

Domain	Example Projects	Evolution Strategy
AI/ML Systems	Recommendation engines, Chatbots	Model training → Testing → Refinement
Gaming	Video games, Mobile games	Core mechanics → Features → Polish
Social Media	Facebook, Twitter features	MVP → User feedback → Feature evolution
Research Tools	Scientific simulations, Data analysis tools	Initial model → Experiments → Improvements
Startups	MVP development, Product validation	Concept → MVP → Market feedback → Pivot

✅ **Advantages:**

- Flexible and adaptive to changes
- Continuous user involvement
- Reduced development time
- Better risk management
- Suitable for research projects
- Innovation-friendly approach

❌ **Disadvantages:**

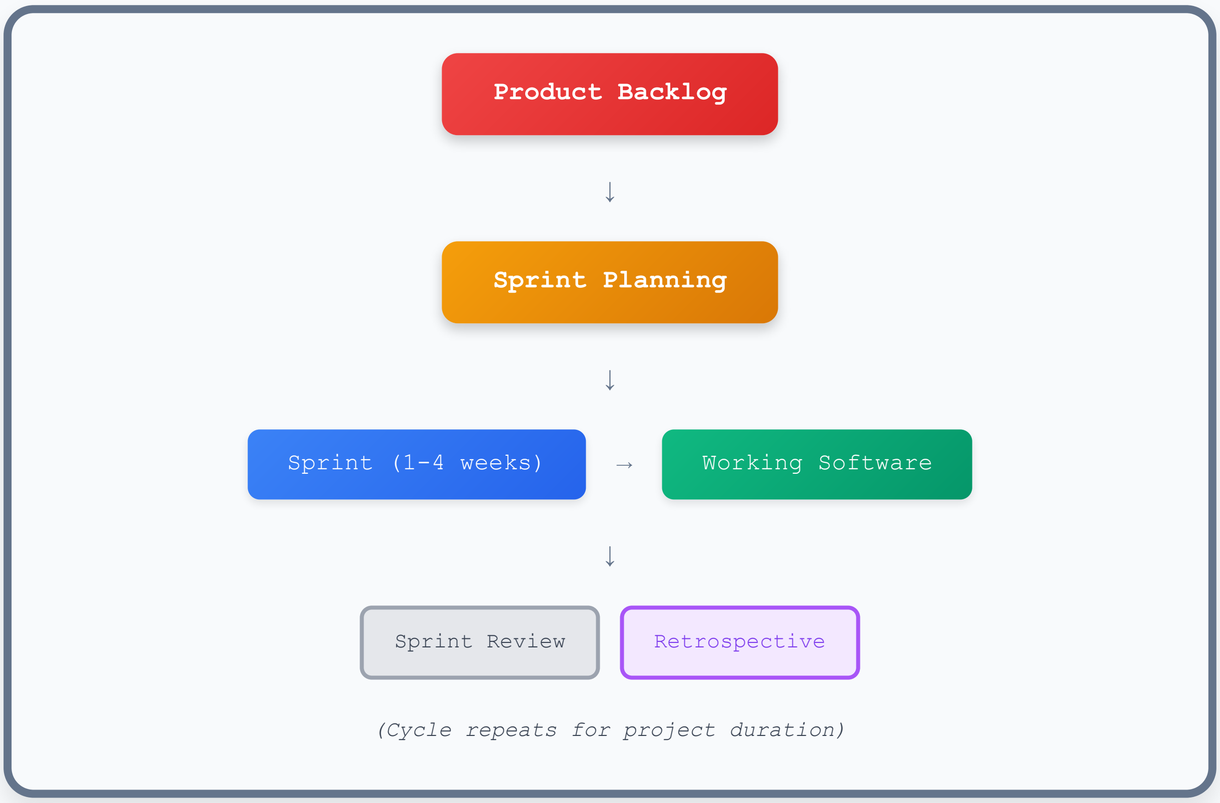
- Difficult to measure progress
- May lead to scope creep
- Requires experienced developers
- Documentation may be insufficient
- Cost estimation is difficult
- May become endless cycles

4. 🚀 AGILE MODEL

Definition:

The Agile model emphasizes iterative development, customer collaboration, and the ability to respond quickly to changing requirements. It focuses on delivering working software frequently through short development cycles called sprints (typically 1-4 weeks).

Agile Process Flow:



Agile Principles:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

Real-World Applications:

Domain	Example Projects	Agile Framework Used

Web Development	Netflix, Spotify, Airbnb	Scrum, Kanban
Mobile Apps	WhatsApp, Instagram, Uber	Scrum, Lean
SaaS Products	Slack, Zoom, Dropbox	Scrum, SAFe
Fintech	PayPal, Stripe, Robinhood	Scrum, XP
E-commerce	Shopify, WooCommerce	Scrum, DevOps

✅ **Advantages:**

- Quick delivery of working software
- High customer satisfaction
- Adaptive to changing requirements
- Improved team collaboration
- Continuous improvement
- Reduced project risk

❌ **Disadvantages:**

- Less emphasis on documentation
- Requires experienced team members
- Needs active customer involvement
- May lead to scope creep
- Difficult for fixed-price contracts
- Can be overwhelming for large projects

PART B: PROBLEM STATEMENT - PACKET SNIFFER CASE STUDY

📡 Network Packet Analyzer for Educational Purposes

1. 🌐 Project Overview

In computer networks education, students and educators need practical tools to understand how data flows across networks. A packet sniffer is an essential learning tool that captures and displays network packets, helping users understand network protocols, troubleshoot connectivity issues, and learn network security concepts.

2. 🎯 Problem Description

Educational institutions lack accessible, easy-to-understand packet analysis tools for teaching network concepts. Existing tools like Wireshark, while powerful, can be overwhelming for beginners. There's a need for a simplified, educational-focused packet sniffer that provides:

- **Educational Focus:** Clear explanations of network protocols and packet structures
- **Simplified Interface:** User-friendly design for students and educators
- **Real-time Monitoring:** Live packet capture and analysis
- **Learning Features:** Built-in tutorials and protocol explanations

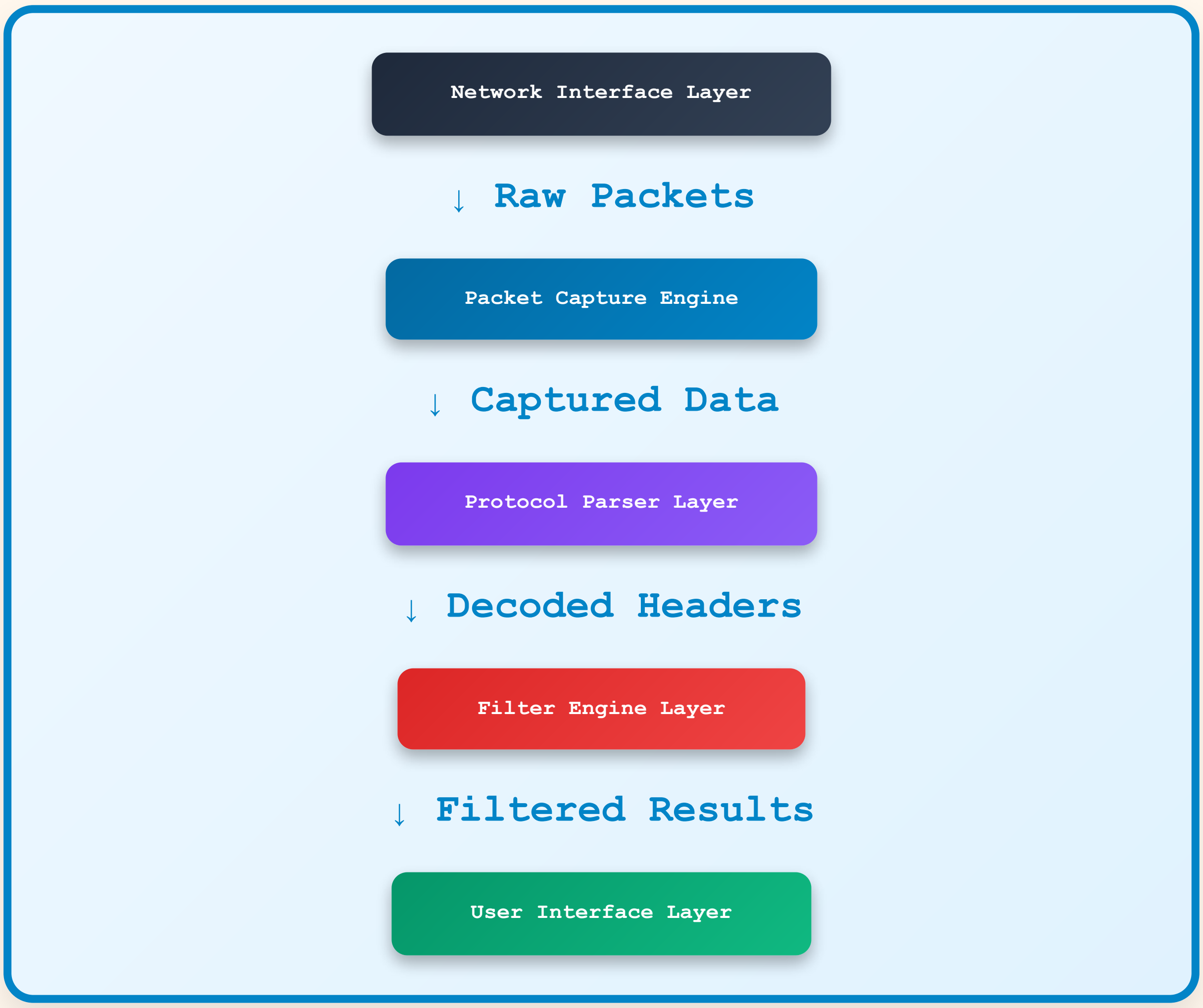
3. 📋 Functional Requirements

Category	Requirement	Description
Core Features	Packet Capture	Capture packets from network interfaces (Ethernet, WiFi)
	Protocol Analysis	Decode common protocols (TCP, UDP, HTTP, ICMP, DNS)
	Real-time Display	Show captured packets in real-time with basic details
	Basic Filtering	Filter by IP address, port number, and protocol type
Educational	Protocol Explanations	Built-in help explaining each protocol and header field
	Packet Structure	Visual representation of packet headers and

Features	View	data
	Learning Mode	Guided tutorials for understanding network concepts
Data Management	Save/Export	Save captured data to files (CSV, text format)
	Session Management	Start, stop, and manage capture sessions

4. 🔧 Technical Requirements

System Architecture:



Performance Specifications:

- **Packet Rate:** Handle up to 100 packets per second
- **Memory Usage:** Maximum 256 MB RAM usage

- **Response Time:** Display packets within 500ms of capture
- **Storage:** Store up to 10,000 packets per session
- **Platform:** Windows and Linux compatibility

5. 👥 Target Users

- **Computer Science Students:** Learning networking concepts
- **Network Instructors:** Teaching network protocols and analysis
- **IT Support Staff:** Basic network troubleshooting
- **Hobbyist Developers:** Understanding network communications

6. 📊 Success Criteria

Metric	Target Value	Measurement Method
Packet Capture Accuracy	95% of packets captured	Comparison with reference tools
User Interface Usability	4/5 user satisfaction rating	User surveys and testing
Educational Effectiveness	80% improvement in learning	Pre/post knowledge assessment
System Stability	Run continuously for 2+ hours	Stress testing and monitoring
Cross-platform Compatibility	Works on Windows and Linux	Testing on multiple platforms

7. ⚠️ Project Constraints

- **Development Time:** 10-12 weeks (academic semester)
- **Team Size:** 3-4 students
- **Budget:** Zero budget (open-source tools only)
- **Technology Stack:** Python with standard libraries
- **Legal Requirements:** Educational use only, no malicious capabilities

PART C: SELECTED PROCESS MODEL - JUSTIFICATION

🎯 Selected Model: INCREMENTAL PROCESS MODEL

1. 📊 Model Selection Analysis

Comparison Matrix:

Criteria	Waterfall	Incremental	Evolutionary	Agile
Fixed Timeline	Excellent	Good	Poor	Fair
Modular Architecture	Poor	Excellent	Fair	Good
Early Testing	Poor	Excellent	Good	Excellent
Student Team Suitability	Good	Excellent	Poor	Fair
Documentation	Excellent	Good	Poor	Poor
Risk Management	Poor	Excellent	Good	Good

DETAILED JUSTIFICATION

2. 🔍 Detailed Justification

A) Project Characteristics Favoring Incremental Model:

📦 **Natural Modularity:**

The packet sniffer has distinct functional modules that can be developed independently:

- **Core Capture Engine:** Network interface access and raw packet capture
- **Protocol Parser:** Decoding different network protocols
- **Filter System:** Packet filtering and search functionality
- **User Interface:** Display and interaction components
- **Data Export:** File saving and export features

⏰ **Academic Timeline Constraints:**

The incremental approach fits perfectly with academic requirements:

- **Semester Duration:** 10-12 weeks allows for 5 clear increments
- **Milestone Submissions:** Each increment provides demonstrable progress
- **Professor Reviews:** Regular evaluation at increment completion
- **Student Learning:** Gradual skill building through increments

B) Implementation Strategy:

Increment	Duration	Features	Deliverable
1	Weeks 1-2	Basic packet capture, Network interface selection	Command-line tool that captures and displays raw packets

2	Weeks 3-4	TCP/UDP/ICMP parsing, Header extraction	Enhanced tool showing protocol information
3	Weeks 5-6	Basic filtering, IP/Port filters	Packet sniffer with filtering capabilities
4	Weeks 7-8	File export, Session management	Complete CLI packet analyzer
5	Weeks 9-10	Graphical interface, Educational features	Full GUI packet sniffer with learning features

3. ❌ Why Other Models Are Less Suitable:

🚫 Waterfall Model - Not Suitable:

- **Late Testing Risk:** Network programming is complex and requires early testing
- **No Early Feedback:** Students need continuous guidance and evaluation
- **Integration Challenges:** Network protocols have many edge cases
- **Inflexible Timeline:** Academic deadlines don't allow for major revisions

🚫 Agile Model - Not Optimal:

- **Overhead for Students:** Sprint ceremonies consume valuable development time
- **Limited Customer Access:** Professor availability is restricted
- **Documentation Requirements:** Academic projects need formal documentation
- **Scope Creep Risk:** Students may add unnecessary features

🚫 Evolutionary Model - Not Practical:

- **Time Constraints:** Academic semester is too short for multiple evolution cycles

- **Resource Limitations:** Students lack experience for rapid prototyping
- **Unclear Requirements:** Educational objectives are well-defined
- **Assessment Challenges:** Difficult to grade evolving prototypes

4. 🎯 Expected Benefits of Incremental Approach:

✅ Academic Benefits:

- **Progressive Learning:** Students build skills incrementally
- **Regular Assessment:** Professors can evaluate at each stage
- **Risk Mitigation:** Problems identified early in development
- **Motivation:** Working software at each increment
- **Teamwork:** Clear division of responsibilities

✅ Project Benefits:

- **Quality Assurance:** Testing at each increment
- **Modular Design:** Clean, maintainable architecture
- **Early Functionality:** Basic packet capture available early
- **Flexible Scope:** Advanced features can be adjusted
- **Documentation:** Incremental documentation development

5. 📋 Success Metrics for Incremental Development:

- **Increment Completion Rate:** Target 100% on-time completion
- **Code Quality:** Maintain consistent quality across increments
- **Integration Success:** Smooth integration between increments
- **Learning Objectives:** Meet educational goals at each stage
- **Team Collaboration:** Effective teamwork and communication

CONCLUSION

📝 Summary of Findings:

This experiment provided comprehensive analysis of four major software process models and their practical applications. Through detailed examination of the Waterfall, Incremental, Evolutionary, and Agile models, we identified their strengths, weaknesses, and suitable application domains.

🎯 Key Learning Outcomes:

- **Model Characteristics:** Each process model has distinct characteristics suited for specific project types
- **Application Domains:** Different industries and project types benefit from different approaches
- **Selection Criteria:** Project constraints, team experience, and requirements stability are key factors
- **Trade-offs:** Every model involves trade-offs between flexibility, documentation, risk, and timeline

🔍 Case Study Analysis:

The packet sniffer project analysis demonstrated the importance of matching process models to project characteristics. The **Incremental Model** emerged as the optimal choice due to:

- Natural modularity of the packet sniffer architecture
- Academic timeline and assessment requirements
- Student team capabilities and learning objectives
- Risk management needs for technical challenges

🎓 Professional Implications:

Understanding process model selection is crucial for software engineering success. This analysis provides a framework for evaluating and selecting appropriate development methodologies in real-world scenarios.

📚 Future Applications:

The knowledge gained from this experiment will be valuable in:

- Industrial software development projects
- Academic research and development
- Project management and planning
- Team leadership and technical decision making

REFERENCES

1. Sommerville, Ian. *Software Engineering*. 10th Edition, Pearson Education Limited, 2015. ISBN: 978-0133943030
2. Pressman, Roger S., and Bruce R. Maxim. *Software Engineering: A Practitioner's Approach*. 8th Edition, McGraw-Hill Education, 2014. ISBN: 978-0078022128
3. Beck, Kent, Mike Beedle, Arie van Bennekum, et al. *"Manifesto for Agile Software Development"*. 2001. Available: https://agilemanifesto.org/
4. Boehm, Barry W. *"A Spiral Model of Software Development and Enhancement"*. Computer, vol. 21, no. 5, pp. 61-72, May 1988. DOI: 10.1109/2.59
5. Royce, Winston W. *"Managing the Development of Large Software Systems"*. Proceedings of IEEE WESCON, vol. 26, pp. 1-9, August 1970.
6. Larman, Craig, and Victor R. Basili. *"Iterative and Incremental Development: A Brief History"*. Computer, vol. 36, no. 6, pp. 47-56, June 2003. DOI: 10.1109/MC.2003.1204375
7. Kurose, James F., and Keith W. Ross. *Computer Networking: A Top-Down Approach*. 7th Edition, Pearson Education, 2016. ISBN: 978-0133594140
8. Stevens, W. Richard. *TCP/IP Illustrated, Volume 1: The Protocols*. 2nd Edition, Addison-Wesley Professional, 2011. ISBN: 978-0321336316
9. Schwaber, Ken, and Jeff Sutherland. *"The Scrum Guide: The Definitive Guide to Scrum"*. Scrum.org, November 2020.
10. IEEE Computer Society. *"IEEE Standard for Software Life Cycle Processes"*. IEEE Std 12207-2017, 2017. DOI: 10.1109/IEEESTD.2017.8100771

📋 END OF DOCUMENT 📋

Software Engineering Laboratory - Experiment 1

Applications of Process Models & Packet Sniffer Case Study

Lokmanya Tilak College Of Engineering (LTCE)

Mumbai University | CSE (IoT & CSBT) | Academic Year 2025-26

Group Members:

1. Raj Dubey (Roll No: 11)
2. Yash Maurya (Roll No: 25)
3. Piyush Kanojiya (Roll No: 23)

Date: August 11, 2025