

PROJECT MONITORING

PROJECT MONITORING

PMG333

Lecture 09

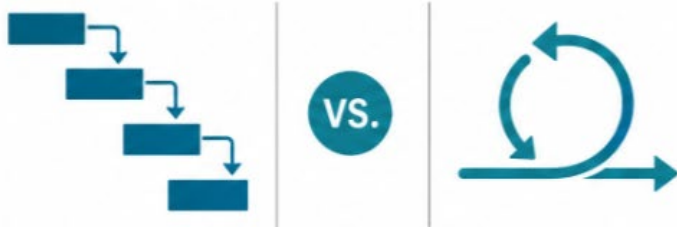
AGILE PROJECTS MONITORING TOOLS

Prepared by
Dr. Bassel alhassan

01

Methodologies Overview

- Waterfall vs. Agile
- Key differences and applications



02

Agile Monitoring Tools



1. Burndown Chart

Tracking remaining work



2. Velocity

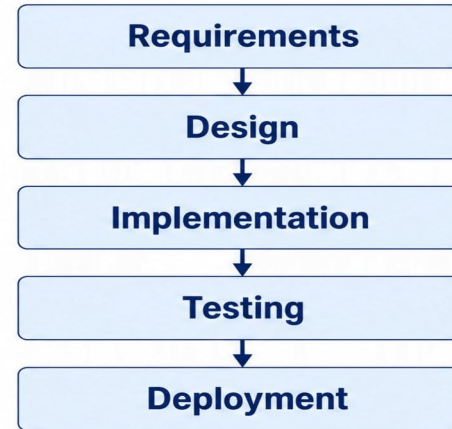
Measuring team capacity



3. Cycle Time

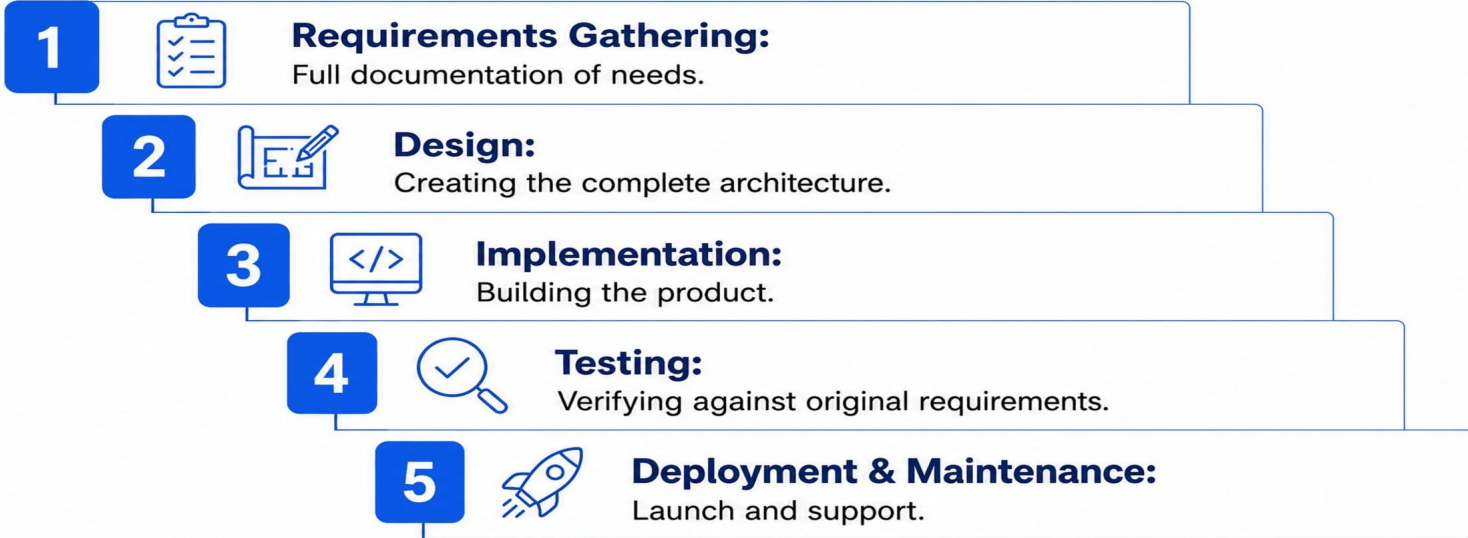
Analyzing workflow efficiency

Waterfall: Definition



- Waterfall is a linear and sequential approach to software development.
- In Waterfall, work flows downward like a waterfall from one phase to the next.
- Each phase must be completed before the next phase begins, making each phase dependent on the previous one.

Waterfall: Key Phases

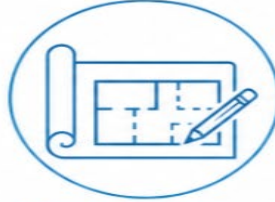


Construction: The Purest Waterfall



1 The Ideal Environment:

Construction is the purest example of Waterfall in action. Physical dependencies create a natural, unchangeable linear progression. You cannot pour a foundation before clearing the site, nor hang drywall before wiring.



2 Strict Phase Gates:

Enforces "Phase Gates" to prevent costly rework. You cannot move from site preparation to structural framing without structural inspections.



3 Predictability & Structure:

Built on predictability. Exhaustive upfront planning minimizes risk and ensures the final deliverable precisely matches the original vision.



1. Site Preparation



2. Foundation



3. Structural Framing



4. MEP Rough-In



5. Drywall & Finishes



Waterfall: Core Characteristics



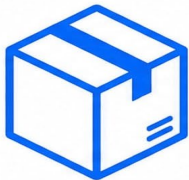
Fixed Scope

Requirements are frozen early.



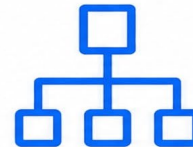
Detailed Planning

Extensive upfront documentation.



Single Delivery

One final product at the end.



Low Flexibility

Changes are costly and difficult.

The \$170M Disaster

FBI Virtual Case File Project Failure (2000-2005)

Project Overview

In 2000, the FBI launched the Virtual Case File (VCF) project as part of the Trilogy initiative to modernize its case management system.

The project used classical Waterfall methodology: gather all requirements, design the system, develop it, test it, then deploy it.

The approach seemed logical on paper. In practice, it became a catastrophic failure.

Financial Impact

\$170M

Total Investment

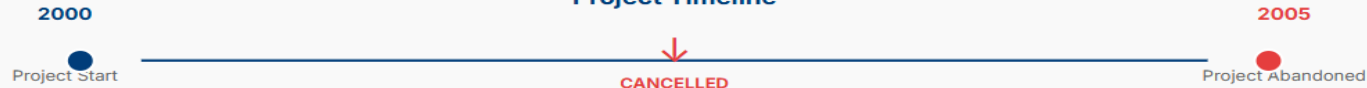
\$104.5M

Actual Loss (61% wasted)

700,000

Lines of Code Written (0% used)

Project Timeline



Six Critical Failures

Problems that led to the project's collapse

1

Political Pressure

After 9/11, the project faced enormous pressure. Timeline compressed from 3 years to 22 months without reducing scope.

2

Unclear Requirements

No clearly defined formal requirements at project start. Result: 800 pages of ambiguous and contradictory specifications.

3

400 Change Requests

In the first year of development, 400 change requests arrived. In Waterfall, every change means complete redesign and retesting.

4

Integration Issues

Team split into 8 parallel teams. When they tried to merge work, discovered fundamental incompatibilities in system architecture.

5

Late Testing

Real testing didn't start until after 3 years. Discovered the system didn't work properly—far too late to fix.

6

No Flexibility

When the project failed, there were no options. Could not revert to old system or fix new system quickly.

Waterfall vs Agile

Why Agile Would Have Succeeded

Waterfall



Assumes a predictable future: plan everything, then execute.

Agile



Embraces uncertainty: build, learn, and adapt continuously.

Success Rates: Standish Group CHAOS Report

Outcome	Waterfall	Agile
Success	14%	42%
Challenged	61%	49%
Failed	25%	9%

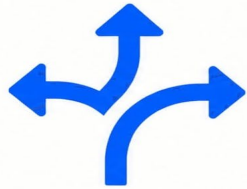
Conclusion: Waterfall projects are three times more likely to fail than Agile projects. For projects with uncertain requirements, changing stakeholder needs, or complex integration challenges, Agile is the superior approach.

Agile: Definition

- Iterative and incremental approach
- Breaks projects into small cycles called Sprints (2-4 weeks)
- Focuses on continuous delivery of value



Agile: Core Philosophy



Flexibility & Adaptation

Agile embraces change and adapts quickly to evolving requirements and priorities.



Continuous Collaboration

Agile fosters teamwork and open communication across all stakeholders.



Frequent Deliveries

Agile delivers working software in short cycles to provide value early and often.



Continuous Improvement

Agile encourages ongoing reflection and refinement to enhance processes and outcomes.

Waterfall vs. Agile: Comparison

Feature	Waterfall	Agile
Flow	Linear & Sequential	Iterative & Cyclic
Planning	Upfront & Detailed	Progressive & Flexible
Delivery	Single Final Delivery	Multiple Frequent Deliveries
Changes	Difficult & Costly	Welcomed & Expected
Client Role	Beginning & End	Continuous Involvement

Case Study 1: Saudi Insurance Co.

Waterfall Project



Time: 24 months (33% delay)



Cost: 750k SAR (50% overrun)



Satisfaction: 65%

Agile Project



Time: 15 months (early delivery)



Cost: 485k SAR (on budget)



Satisfaction: 92%

Case Study 1: Lessons Learned



Waterfall Issues

- Delayed requirements gathering
- Inability to adapt to new needs discovered during design



Agile Advantages

- 8 sprints of 2 weeks each
- Immediate feedback from department representatives
- High alignment with actual user needs

Case Study 2: Brazilian Pharma

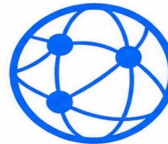
Implementing a complex Product Lifecycle Management (PLM) system using Scrum (Agile framework).

Key Challenges



1.

Complex Integrations



2.

Distributed Teams



3.

Unclear Initial Requirements

Case Study 2: Results & Impact



Early Delivery

First working version
in 6 weeks.



Higher Quality

Defects reduced by 40%.



Team Morale

Satisfaction increased
from 5.2 to 8.5 out of 10.



Better Communication

Faster issue resolution
via daily standups.

Case Study 3: ACM Study (2024)

Empirical study comparing 15 software projects.

Defects per 1000 lines of code

8.3



Waterfall

2.1



Agile

Post-release defects

65%



Waterfall

15%



Agile

Case Study 3: Cost & Time Impact



Resolution Time

Waterfall:

14 days

Agile:

2 days



Cost per Defect

Waterfall:

\$5,000

Agile:

\$800

When to Use Waterfall

Fixed Requirements



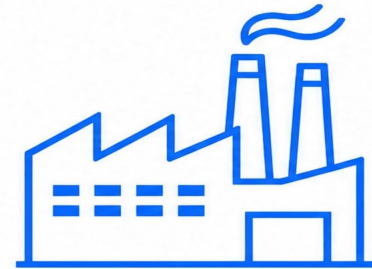
(e.g., construction)

Strict Regulations



(e.g., aerospace)

Physical Products



(e.g., manufacturing)

When to Use Agile



Software & Tech

Rapidly changing technological environments.



Unclear Requirements

When the client's needs are expected to evolve.



Fast Time-to-Market

Startups and competitive markets needing quick releases.



High Customer Focus

Projects requiring continuous user feedback.

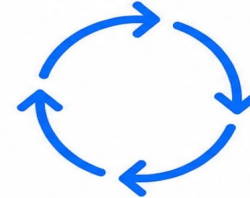
Summary & Conclusion

Waterfall



Assumes a predictable future:
plan everything, then execute.

Agile



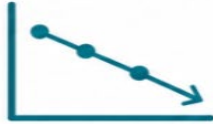
Embraces uncertainty:
build, learn, and adapt continuously.

Conclusion: Both methodologies are valid tools. Success depends on matching the right approach to the project's specific context and constraints.



Three Agile Monitoring Tools

01



Burndown Chart

Tracks the amount of work remaining versus the time available in a sprint or release.

02



Velocity

Measures the amount of work a team can successfully complete during a single sprint.

03



Cycle Time

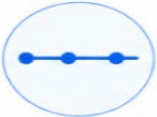
Calculates the total time taken from when work starts on an item until it is completed.

The Story Behind the Chart



Context

Context: A cross-functional Agile team (developers & UX designers) working in 2-week Sprints.



The Problem

The Problem: The burndown chart was completely flat for the first week, then dropped vertically in the last 3 days.



The Discovery

The Discovery: The team was suffering from "Last-Minute Delivery Syndrome" due to isolated, siloed work.



The Result

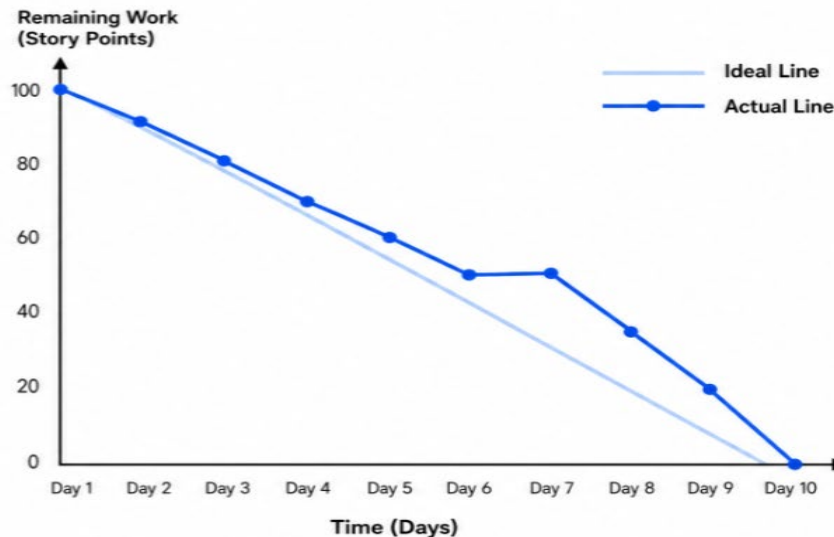
The Result: The visual evidence forced the team to collaborate better, significantly improving flow and quality.

What is a Sprint Burndown Chart?

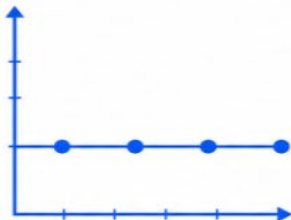
A visual representation of work remaining versus time in a given Sprint.

Core Components:

- **X-Axis (Horizontal):** Time, usually measured in days.
- **Y-Axis (Vertical):** Remaining work, usually measured in Story Points.
- **Ideal Line:** A straight, diagonal guideline showing theoretical steady progress.
- **Actual Line:** A fluctuating line showing the team's real, day-to-day progress.

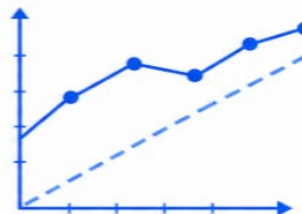


Reading the Chart: Warning Signs



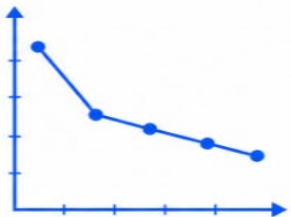
Flat Line (No Progress)

Work is stuck in progress; tasks are too large or blocked.



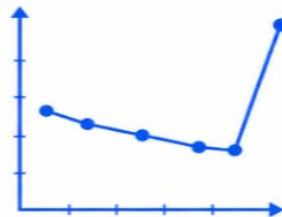
Actual Above Ideal

The team is moving slower than planned. Risk of missing the Sprint goal.



Early Drops

The team overestimated the work complexity or under-committed.



Line Goes Up

Scope creep! New work was added to the Sprint after it started.

Practical Application: Best Practices



Daily Updates

The chart is useless if tasks aren't updated daily. Move items to "Done" immediately.



The Daily Standup

Use the chart as the visual centerpiece during the daily 15-minute meeting.



Don't Change Estimates

Avoid changing Story Points mid-sprint; it hides the real issues.



Continuous Learning

Review past charts in the Sprint Retrospective to improve future planning.

Summary: Why It Matters



Transparency

Provides an immediate, honest view of the team's actual progress.



Early Detection

Identifies bottlenecks and scope creep before it's too late.



Predictability

Helps Scrum Masters and Product Owners forecast delivery accurately.



Team Empowerment

A diagnostic tool that helps the team self-correct and improve their workflow.

The Speedometer: A Real Transformation



Context: Namecheap faced an 18-month complex project with exhausting 3-hour Sprint planning meetings.



The Solution: The team implemented "Velocity-Based Agile" using historical data to plan future Sprints.



The Impact: Planning meetings shrank from 3 hours to 30 minutes!



The Result: The massive project was delivered within two weeks of the original 18-month timeline with zero budget overruns.

What is Velocity in Agile?

Definition:

A metric that measures the amount of work a team can successfully complete during a single Sprint.

Unit of Measurement:

Measured in Story Points, not hours or days.

The Three Factors of Story Points

**Complexity:**

How difficult is the task?

**Effort:**

How much actual work is required?

**Risk/Uncertainty:**

Are there unknown variables?

How to Calculate Velocity

1**Estimate**

Assign Story Points to tasks before the Sprint begins.

**2****Execute**

The team works on the tasks during the Sprint.

**3****Sum the "Done"**

At the end, add up the points of ONLY fully completed tasks. (90% done = 0 points).

**4****Average**

Calculate the Average Velocity over the last 3 Sprints for reliable future planning.

Why Velocity is a Powerful Tool



1. Accurate Forecasting

If the backlog has 200 points and average velocity is 50, the project needs 4 Sprints.



2. Realistic Planning

Prevents teams from over-committing and burning out.



3. Early Warning System

A sudden drop in velocity indicates hidden problems like technical debt or blocked tasks.



4. Continuous Improvement

Tracking trends helps teams measure their growing efficiency over time.

The Dark Side: Common Mistakes



Velocity $\neq$ Quality

High velocity with buggy code is a failure. It measures quantity, not quality.



Never Compare Teams

Team A's 15 points might equal Team B's 30 points. Velocity is strictly team-specific.



The Pressure Trap

Pushing a team to artificially increase velocity leads to burnout and corner-cutting.



The Golden Rule

Use it as a planning compass, not a performance punishment.

Cycle Time

Measuring Workflow Efficiency in Agile Projects



Measures time from actual work start to completion



Essential Agile tool for measuring team efficiency



Focuses on flow quality, not just quantity

Understanding Cycle Time: Components & Metrics

Four Types of Work



Features

New work that adds customer value



Defects

Bug fixes and issue resolution



Debt

Internal improvements without direct customer value



Risks

Security and compliance work

Five Flow Metrics



Flow Velocity

Number of completed tasks



Flow Time

Total time from start to finish



Flow Efficiency

Ratio of actual work to total time



Flow Load

Number of tasks in progress



Flow Distribution

Balance across work types

Benefits of Measuring Cycle Time



Discover Real Bottlenecks

Identify where time is actually wasted and pinpoint critical waiting points in your process.

Example: 14-day Cycle Time but only 3 days of actual work = 11 days of waiting



Set Realistic Goals

Build expectations on historical data to improve forecast accuracy and planning.

If average Cycle Time is 5 days, don't expect complex tasks to complete in one day



Improve Customer Satisfaction

Deliver features faster, creating happy customers and gaining better competitive advantage.

Happy customers = loyal customers. Faster delivery = competitive edge

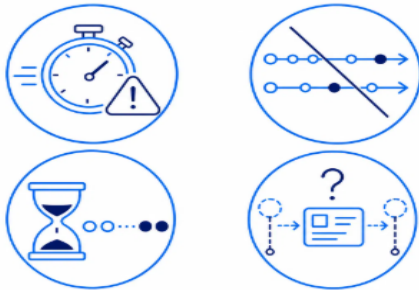


Discover Hidden Problems

Sudden spikes may indicate technical issues, resource shortages, or complex processes.

Early warning signals help identify problems before they become critical

The Dark Side: Common Mistakes



1. Confusing Cycle Time with Lead Time

Cycle Time measures from actual work start to completion. Lead Time includes waiting in Backlog. Mixing them leads to inaccurate measurements.

2. Ignoring Wait Time

Focusing only on actual work time while overlooking 70-80% of time wasted in waiting. Fails to discover real bottlenecks in the process.

3. Unclear Start and End Points

Inconsistent definitions of "work start" and "completion" lead to inconsistent measurements across different tasks and difficulty in process improvement.

4. Comparing Different Teams

Each team has different context and complexity. Team A's Cycle Time differs completely from Team B's. Comparisons create unhealthy pressure.

Encoparts: The Logistics Challenge



1. High Complexity

International purchasing involved multiple stages, from supplier search to final registration, taking 15-40 days to complete.



2. Lack of Visibility

The team struggled to identify where delays occurred. Over 70% of the process time was spent on non-value-added waiting periods.



3. Siloed Teams

Poor communication between purchasing, shipping, and finance departments led to unnecessary rework and bottlenecks.

Encoparts Purchasing Process — BEFORE Cycle Time Measurement



The Solution: Kanban & STATIK Implementation



- 1 STATIK Approach:** Mapped the entire value stream to identify bottlenecks and eliminate non-value-added activities.

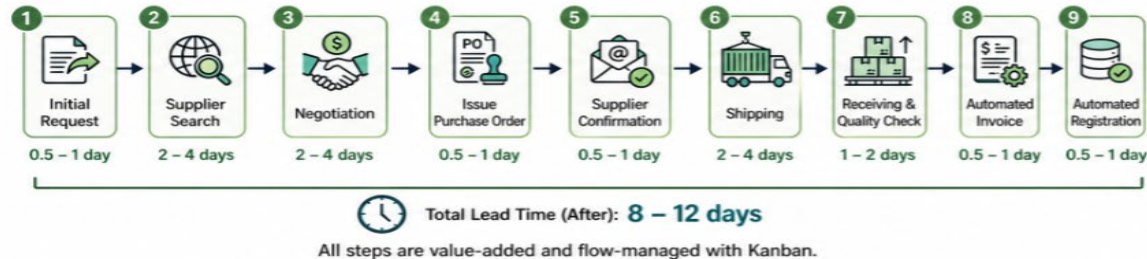


- 2 Kanban & WIP Limits:** Visualized the workflow and restricted Work-In-Progress to ensure continuous flow and prevent team burnout.



- 3 Process Optimization:** Reduced cycle time from 20–30 days to 8–12 days by automating invoices and standardizing suppliers.

Encoparts Purchasing Process – AFTER Cycle Time with Kanban



Kanban Board & WIP Limits

Ready	In Progress	Done
	3 WIP Limit	



Visualize the work

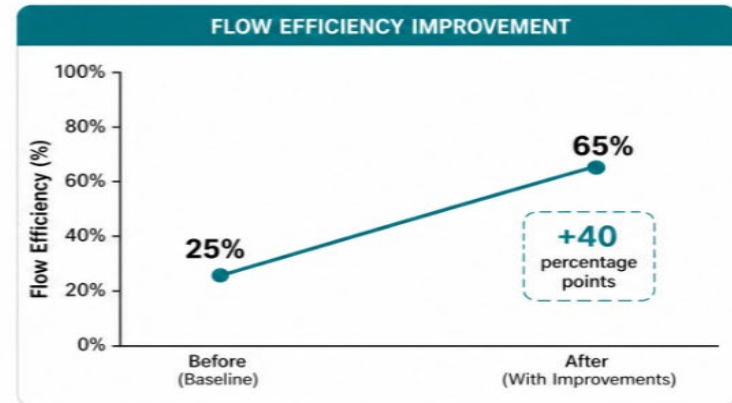
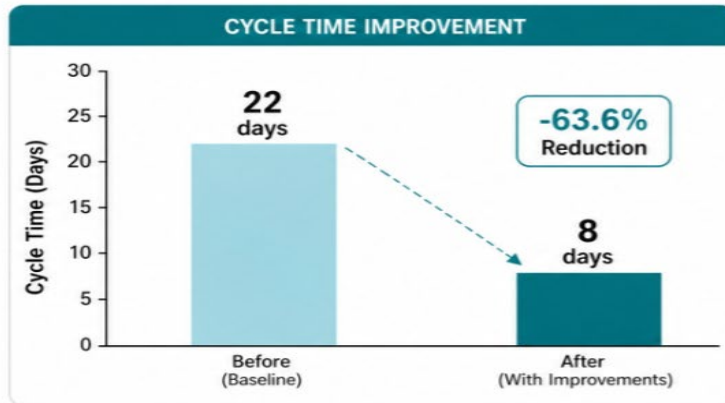


Limit work in progress



Remove avoidable waiting

Results: 63.6% Cycle Time Reduction



1. CYCLE TIME

22 → 8 Days
(-63.6%)



2. FLOW EFFICIENCY

25% → 65%
(+160%)



3. THROUGHPUT

5 → 15 req/day
(Doubled)



4. ERROR RATE

8.0% → 1.5%
(-81%)

Results based on measured operational data after implementation.

Strategies to Improve Cycle Time



Reduce Waiting Points

Eliminate bottlenecks in the workflow process.

- Review code immediately
- Automate tests
- Reduce approval layers



Improve DoR & DoD

Define clear task readiness and completion criteria.

- DoR: Ensure task clarity before starting
- DoD: Set measurable completion criteria
- Reduce rework and confusion



Enhance Communication

Break down silos and improve team collaboration.

- Reduce silos between teams
- Improve cross-functional communication
- Conduct effective standups



Increase Flow Efficiency

Maximize productive work time and resource utilization.

- Reduce wasted waiting time
- Increase actual work time
- Better resource allocation

Thank You