alok.giri@ncit.edu.np

Agile KPIs

What are KPIs in Agile?

- **Metrics** that help teams measure progress, efficiency, and predictability
- Track **delivery performance** and highlight opportunities for

improvement

Common Agile KPIs:

- Velocity
- Cycle Time
- Lead Time
- Throughput
- Defect Rate
- Team Happiness (qualitative)

Velocity

Velocity is the amount of work a team completes in a sprint, typically measured in story points or work items.

How to Calculate Velocity

Step-by-step:

1. At the end of each sprint, **sum the story points** for all completed stories.
2. **Do not count** incomplete stories.
3. Track the velocity **across multiple sprints** to find a stable average.

Example:

- Sprint 1: 23 story points
- Sprint 2: 26 story points
- Sprint 3: 25 story points

Average Velocity = (23 + 26 + 25) / 3 = 24.7 ≈ 25 points

Velocity – Capacity Planning and Forecasting

- **Plan future sprints**: Use average velocity to forecast how much work can fit
- **Estimate delivery dates**: If backlog has 100 story points and velocity is 25 → 4 sprints
- Helps balance **team workload** and **avoid over-committing**

Caution:

Velocity is a **team-specific internal metric**—not meant for comparing teams.

Slide 6: Cycle Time – Definition

Definition:

Cycle Time is the **total time it takes to complete a work item**, from the moment it starts (in progress) to when it's done.

It answers:

"How long does it take to complete a task once we start it?"

How to Calculate Cycle Time

Formula:

```
Cycle Time = Completed Date − Start Date (in-progress)
```

Example:

- A story started on May 1 and completed on May 4 → Cycle Time = 3

days

Tools That Help:

- Jira, Azure DevOps, Trello with plugins
- Cumulative Flow Diagrams (CFDs)

Strategies to Reduce Cycle Time

1. **Limit Work In Progress (WIP):**
    - Fewer tasks = more focus and faster flow
2. **Reduce bottlenecks:**
    - Identify stages where work piles up
3. **Improve team collaboration:**
    - Shared ownership and pairing
4. **Break down large stories:**
    - Smaller stories move through faster
5. **Automate testing and deployment:**

- Reduce delays caused by manual QA or release gates

Lead Time – Definition

Definition:
Lead Time is the **total time from when a request is made** (backlog entry) **to when it is completed**.

It answers:
"How long does a customer wait to get what they asked for?"

Slide 10: How to Calculate Lead Time

Formula:

```
Lead Time = Completion Date − Request Date
```

Example: A feature is requested on April 1, and delivered on April 10 → Lead Time = 9 days **Includes:**

- Waiting time before development

- Cycle Time (development phase)

Improving Lead Time

1. **Prioritize backlog frequently:**
    - Keep high-value work near the top
2. **Visualize work queues:**
    - Identify aging items and dependencies 3. **Reduce hand-offs and approvals:** ○ Empower teams to move work forward faster 4. **Manage external blockers:**
    - Communicate with outside teams early
5. **Automate workflows:**
    - Integrate CI/CD to cut delivery delays

Using All Three KPIs Together

KPI	Use For
Velocity	Sprint planning and delivery forecasting
Cycle Time	Team efficiency and flow improvement
Lead Time	Customer satisfaction and delivery speed

Introduction to Reporting Tools and Dashboards

Why Visualize Agile Metrics?

- Provide **real-time insights** into progress and issues
- Drive **data-informed decisions**
- Improve **transparency and accountability**

- Help in **retrospective analysis** and **future planning**

Goal:

Track how work flows, identify bottlenecks, and assess delivery performance.

Jira Dashboards

Overview:

Jira offers **custom dashboards** with real-time gadgets for:

- Velocity
- Burndown
- Sprint Health
- Cumulative Flow Diagrams
- Control Charts (Cycle Time)

Features:

- Filterable widgets using JQL
- Team-level and program-level dashboards
- Integration with Confluence, Bitbucket, GitHub

Tip: Use **“Filters + Gadgets”** to create focused dashboards per team or role.

Trello and Asana Dashboards

Trello

- Simple Kanban boards
- **Power-Ups** enable reporting (Charts by Vizydrop, Dashcards)
- Great for small teams and simple projects

Asana

- Timeline and Portfolio views
- Dashboards for:
  - Task status
  - Due date tracking
  - Goal progress

- Suited for cross-functional and non-engineering teams

Key Metrics to Visualize

Metric	Purpose
Velocity Chart	Forecast capacity based on past delivery
Burndown Chart	Track sprint progress and completion rate
Cycle Time	Measure efficiency from start to finish
Lead Time	Total time from request to delivery
WIP Limits	Prevent overload by capping active work
Epic/Release Burnup	Monitor value delivery toward strategic goals

Velocity Chart

- Shows the number of **story points** (or tasks) completed over sprints
- Helps with:
  - **Capacity planning**
  - Sprint forecasting
  - Tracking **team consistency**

Jira Tip: Use the **Velocity Report** to analyze across multiple sprints

Slide 8: Burndown Chart

- Plots **work remaining vs time**
- X-axis = sprint days, Y-axis = story points
- **Ideal line vs actual progress**
- Reveals:
  - Blockers
  - Scope creep
  - Overcommitment

Lead Time & Cycle Time Charts

- **Cycle Time:** From “in progress” to “done”
- **Lead Time:** From backlog to done

Visual Tools:

- **Control Charts** in Jira
- **Cumulative Flow Diagrams (CFD)** for identifying WIP congestion

Use cases:

- Optimize flow efficiency
- Reduce delays in development or review

Work in Progress (WIP) Limits

- Shows how many items are currently active ● **Limiting**

WIP = Faster flow + less context switching

Metrics to track:

- Average WIP count
- Items per column (e.g., Testing, Code Review)
- Age of WIP items

Tools: Kanban boards with **column limits and WIP alerts**

Epic and Release Burnup Charts

Epic Burnup Chart:

- Tracks how much work is completed toward an **epic** goal

Release Burnup Chart:

- Tracks progress toward **a product release**

Benefits:

- Shows **scope changes and additions**
- Provides a **clear picture of delivery pace** vs scope growth

Best in: Jira Advanced Roadmaps, Aha!, or Azure Boards

Best Practices for Dashboards

1. **Keep it simple and focused**
    - Tailor dashboards to team roles (Dev, PO, Management)
2. **Automate where possible**
    - Use real-time data; avoid manual entry
3. **Visualize trends, not just snapshots**
    - Compare sprint-over-sprint, epic trends, etc.
4. **Include actionable insights**

- Use charts that trigger retrospectives or improvements

5. **Review regularly**
    - Dashboards are only useful if **reviewed and acted upon**

Metrics and Continuous Improvement – The Connection

What is Continuous Improvement?

A systematic effort to seek and implement **incremental changes** that enhance quality, efficiency, and performance.

Role of Metrics:

- Provide **quantitative insight** into how a process performs
- Allow **data-driven decisions** instead of assumptions
- Highlight areas of waste, delays, or inconsistency
- Enable **experiments, validation, and learning**

Agile Principle #12:

“At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.”

Agile Metrics for Continuous Improvement

Metric	Purpose
Velocity	Understand team delivery capacity
Cycle Time	Optimize how long it takes to complete work
Lead Time	Improve customer satisfaction through faster delivery
Escaped Defects	Track quality of delivery
Throughput	Analyze how many items are completed over time
Cumulative Flow	Visualize bottlenecks and flow efficiency
Team Happiness	Assess morale and well-being

Using Metrics for Process Enhancement

Steps to Leverage Metrics Effectively:

1. **Collect Reliable Data:**
    - Automate tracking via tools (e.g., Jira, GitLab, Azure Boards)
2. **Interpret Trends:**
    - Look for patterns over time, not isolated data points
3. **Correlate with Behavior:**
    - Identify what practices drive improvements (or regressions)
4. **Adjust Processes:**
    - Tweak WIP limits, sprint length, review practices, etc.
5. **Validate Results:**
    - Remeasure after changes

Root Cause Analysis (RCA)

Purpose:

To go beyond symptoms and identify the **underlying cause** of inefficiencies or issues.

Techniques:

- **5 Whys Method:**
  - Ask “why?” repeatedly until the core cause is uncovered
- **Fishbone (Ishikawa) Diagram:**
  - Categorize causes into people, process, tools, etc.

Example (5 Whys):

- Defect found in production
  → Why? Missed in testing
  → Why? No test case for edge case
  → Why? Requirements unclear
  → Why? Inadequate PO review
  → Why? Lack of stakeholder collaboration

Improvement Cycles – PDCA Framework

1. Plan:

- Identify an area for improvement using metrics
- Set a goal or hypothesis for change

2. Do:

- Implement a small, testable change
- Apply in a sprint or workflow

3. Check:

- Review data after implementation
- Validate with team feedback and metrics

4. Act:

- Standardize if successful
- Adjust and retry if not

SMART Goals for Continuous Improvement