

JIRA - PRACTICAL IMPLEMENTATION

JIRA DASHBOARD OVERVIEW

When you sign in to Jira dashboard of an organization:

You can see a tab called "Projects".

PROJECTS IN JIRA

What Projects tab shows: All the projects related to the organization.

Example: Amazon organization

Possible projects:

- Payment
- Transaction
- Order
- UI/UX
- Customer Service
- Logistics
- Inventory
- Analytics

Each project: There can be a project dedicated to each of these teams.

Purpose: Organize work by team or functional area.

CREATING ITEMS IN JIRA

When we create something in the Jira dashboard:

Process: It will come and drop in the backlog.

WHAT IS BACKLOG?

Definition: Backlog is a collection of:

- User stories
- Bugs
- Requests
- Tasks
- Everything that needs to be done

Purpose: Central repository of all work items that haven't been scheduled yet.

BACKLOG REFINEMENT PROCESS

Product Owner's role:

Step 1: Review backlog Product Owner checks all items in backlog.

Step 2: Prioritize Decides the priority of each ticket.

Priority levels:

- P1: Critical/Highest
- P2: High
- P3: Medium
- P4: Low
- P5: Lowest

Step 3: Add to sprint Prioritized tickets are added to upcoming sprint.

This process: Backlog refinement takes place.

Result: The prioritized tickets are first added to the sprint as stories.

TICKET SIZING

What is ticket sizing:

Purpose: Estimate effort required for each ticket.

Process: Ticket sizing based on the type of work in each ticket.

Sizing scale: Usually uses Fibonacci sequence:

- 1: Very small (few hours)
- 2: Small (half day)
- 3: Medium (1 day)
- 5: Large (2-3 days)
- 8: Very large (1 week)
- 13: Huge (2 weeks)
- 21: Epic-level (should be broken down)

Team activity: Team will come up with the top 10 tickets (for example) and assign it to Sprint 1, and so on.

SPRINT PLANNING

What is sprint planning:

Purpose: Decide what work will be done in upcoming sprint.

Who attends:

- Scrum Master
- Product Owner
- Development Team

Activities:

1. Review sprint goal
 2. Select stories from backlog
 3. Estimate effort
 4. Assign to team members
 5. Commit to sprint
-

CAPACITY PLANNING

What it is: Calculate team's available capacity for the sprint.

Calculation:

- Team size: 5 developers
- Sprint duration: 10 working days
- Individual capacity: 6 hours/day (accounting for meetings, etc.)
- Total capacity: $5 \times 10 \times 6 = 300$ hours

Story points: Convert hours to story points and select work accordingly.

ACTIVE SPRINTS

What you can see in active sprints:

Information displayed:

- What are the user stories assigned to which user
 - Status of each story
-

Status columns:

To Do: Work not started yet.

In Progress: Currently being worked on.

In Review: Completed, awaiting review.

Done: Completed and approved.

Moving stories:

How it works: We can just move our user stories from one status to another.

Actions:

- Drag and drop stories between columns
- Update status with click
- Add comments on progress

Example: Developer starts work:

- Drag story from "To Do" to "In Progress"

Developer completes work:

- Drag story from "In Progress" to "In Review"

After code review:

- Drag story from "In Review" to "Done"
-

DAILY STANDUP

What is daily standup:

Purpose: Quick team sync meeting.

Duration: 15 minutes maximum (standing up encourages brevity).

Three questions asked to each team member:

Question 1: What you have done yesterday?

Question 2: What you are going to do today?

Question 3: Any blockers?

Who facilitates:

Every team has a Scrum Master.

Scrum Master takes this meeting.

Scrum Master responsibilities:

- Facilitate daily standup
 - Remove blockers
 - Track sprint progress
 - Coach team on Agile practices
-

MID-SPRINT REVIEW

What is mid-sprint review:

Another ceremony in Agile.

Purpose: Review where we currently are in the sprint.

Questions answered:

- How many stories completed?
- How many in progress?
- Are we on track?
- Any risks?
- Need to adjust scope?

Timing: Usually middle of sprint (Day 5 of 10-day sprint).

STORY POINTS IN FIBONACCI FASHION

How stories are estimated:

Stories are given points in Fibonacci fashion.

Fibonacci sequence used: 1, 2, 3, 5, 8, 13, 21

Rule: Highest the number, more critical or complex is the story.

Examples:

Story: Fix button color Points: 1 Reason: Very simple, few hours

Story: Add forgot password link Points: 2 Reason: Simple feature, half day

Story: Implement email verification Points: 5 Reason: Moderate complexity, 2-3 days

Story: Build payment gateway integration Points: 13 Reason: Complex, needs testing, 1-2 weeks

Story: Redesign entire checkout flow Points: 21 Reason: Very complex, should be broken into smaller stories

Why Fibonacci:

Purpose: As work gets larger, estimates become less precise.

Fibonacci reflects this:

- Difference between 1 and 2 is meaningful
 - Difference between 13 and 21 shows uncertainty
 - Forces breaking down large tasks
-

JIRA REPORTS

We have several types of reports:

Report 1: Velocity Chart

What it shows: Team's velocity (story points completed) per sprint.

Example:

Sprint 1: 30 points

Sprint 2: 35 points

Sprint 3: 32 points

Sprint 4: 38 points

Use:

- Predict future capacity
 - Track team performance
 - Identify trends
-

Report 2: Sprint Report

What it shows: Detailed view of completed sprint.

Information:

- Stories completed
- Stories not completed
- Total points completed

- Total points committed
- Completion percentage

Use:

- Sprint retrospective
 - Identify issues
 - Improve next sprint
-

Report 3: Burndown Chart

What it shows: Remaining work versus time in sprint.

Axes:

- X-axis: Days in sprint
- Y-axis: Remaining story points

Ideal line: Straight line from total points to zero.

Actual line: Shows real progress.

Use:

- Track if on schedule
- Identify if sprint at risk
- Daily progress monitoring

Example interpretation:

- Line above ideal: Behind schedule
 - Line below ideal: Ahead of schedule
 - Flat line: No progress (blocker)
-

JIRA DASHBOARDS

What are dashboards:

Apart from reports: We can also create dashboards.

Purpose: Dashboards give a very good view of your entire project.

Dashboard features:

With the help of JQuery: We can group issues or tasks together in dashboard.

What you can add:

- Velocity chart
- Burndown chart
- Issue statistics
- Recent activity
- Custom filters
- Sprint health

Customization:

- Drag and drop widgets
- Resize widgets
- Multiple dashboards
- Share with team

JIRA TERMINOLOGY

Issue in Jira:

In Jira, ticket is called an issue.

Types of issues:

- Story: User-facing feature
- Task: Technical work
- Bug: Something broken
- Epic: Large body of work
- Subtask: Part of story/task

JIRA QUERY LANGUAGE (JQL)

What is JQL:

Purpose: Query and filter issues in Jira.

Example queries:

Query 1: Find all bugs

issue type = Bug

Result: Shows all issues where issue type is Bug.

Query 2: Find high priority issues

priority = Highest

Query 3: Find issues assigned to you

assignee = currentUser()

Query 4: Find issues in current sprint

sprint in openSprints()

Query 5: Complex query

project = "Payment" AND status = "In Progress" AND priority in (Highest, High)

Result: All high-priority in-progress issues in Payment project.

Using JQL:

Where to use:

- Filters
- Dashboards
- Reports
- Search

Benefits:

- Powerful searching
 - Save filters
 - Share with team
 - Create custom views
-

CONFLUENCE INTEGRATION

Integration with Jira:

We can integrate Confluence and store all the documents.

What documents:

- Documents derived from Jira outcomes
 - Meeting notes
 - Technical specifications
 - Architecture diagrams
 - Any project documentation
-

How integration works:

Link Confluence to Jira:

- Confluence page linked to Jira epic
- Jira issue linked to Confluence documentation
- Requirements in Confluence
- Implementation tracking in Jira

Benefits:

- Single source of truth
- Traceability
- Easy navigation
- Centralized information

TYPICAL AGILE WORKFLOW IN JIRA

Week 1: Sprint Planning

Monday:

- Team meets for sprint planning
- Review backlog
- Product Owner prioritizes
- Team selects stories
- Estimate story points
- Commit to sprint goal

Stories moved: From backlog to Sprint 1.

Week 1-2: Active Sprint

Daily:

- Daily standup (15 minutes)
- Update story status
- Move cards on board
- Add comments
- Log work hours

Mid-sprint (Day 5):

- Mid-sprint review
- Check progress
- Adjust if needed

Continuous:

- Developers work on stories
- QA tests completed stories
- Code reviews happen

- Stories move through workflow
-

Week 2: Sprint End

Friday:

- Sprint review meeting
- Demo completed work
- Get stakeholder feedback

After sprint review:

- Sprint retrospective
- What went well?
- What didn't go well?
- What to improve?

Reports generated:

- Velocity chart updated
 - Sprint report created
 - Burndown chart finalized
-

Next Monday: New Sprint

Cycle repeats:

- Sprint 2 planning
 - Select new stories
 - Continue development
-

JIRA WORKFLOW EXAMPLE

Story lifecycle:

Step 1: Created Product Owner creates story in backlog.

Step 2: Refined Team discusses and estimates during backlog refinement.

Step 3: Planned Added to sprint during sprint planning.

Step 4: To Do Sprint starts, story ready to work.

Step 5: In Progress Developer starts working, moves to In Progress.

Step 6: In Review Code complete, moved to code review.

Step 7: In Testing QA testing the feature.

Step 8: Done Testing passed, feature complete.

Step 9: Deployed Released to production (optional status).

SERVICENOW WORKFLOW EXAMPLE

Incident management flow:

Step 1: Alert triggered Monitoring system (CloudWatch, Prometheus) detects issue.

Step 2: Incident created ServiceNow API called automatically.

Incident details:

- Title: "Payment Service Down"
- Priority: P1 (Critical)
- Team: Payments
- Auto-assigned: To payments on-call engineer

Step 3: Notification sent Engineer receives:

- Email
- SMS
- Slack message
- PagerDuty alert

Step 4: Engineer acknowledges Updates incident: "Acknowledged. Investigating root cause."

Step 5: Diagnosis Engineer finds issue: "Database connection pool exhausted."

Updates incident: "Root cause identified. Increasing connection pool size."

Step 6: Resolution Engineer makes fix: "Applied configuration change. Service restored."

Step 7: Verification Engineer tests: "Payment service operational. Monitoring for 30 minutes."

Step 8: Close incident Final update: "Issue resolved. No customer impact after fix. Post-mortem scheduled."

Incident closed.

Change management flow:

Step 1: Need identified From incident: Need to increase connection pool permanently.

Step 2: Change request created In ServiceNow:

- Summary: "Increase database connection pool"
- Reason: "Prevent payment service outages"
- Risk: Low
- Impact: Low
- Change window: Saturday 2 AM - 4 AM

Step 3: Change details Document:

- Current config: pool size = 50
- New config: pool size = 100
- Affected systems: Payment database
- Rollback plan: Revert config file

Step 4: Approvals requested Sent to:

- Technical lead
- Database admin
- Product owner

Step 5: Approvals received All approvers review and approve.

Step 6: Schedule Change scheduled for Saturday 2 AM.

Step 7: Implementation Saturday 2 AM:

- Take database backup
- Update configuration
- Restart service

- Verify connections

Step 8: Verification Test:

- Payment service working
- Connection pool at 100
- No errors in logs

Step 9: Close change request Update: "Change implemented successfully. Service running normally."

Change closed.

SUMMARY

Jira dashboard:

- Projects tab shows all projects
- Each team has dedicated project

Backlog:

- Collection of all work items
- User stories, bugs, requests
- Unscheduled work

Backlog refinement:

- Product Owner prioritizes
- Team estimates
- Stories sized (Fibonacci points)
- Top items selected for sprint

Sprint planning:

- Select stories for sprint
- Capacity planning
- Assign to team members
- Commit to sprint goal

Active sprint:

- User stories assigned
- Status: To Do, In Progress, In Review, Done
- Drag and drop to update status

Daily standup:

- 15-minute meeting
- 3 questions per person:
 1. What did you do yesterday?
 2. What will you do today?
 3. Any blockers?
- Scrum Master facilitates

Mid-sprint review:

- Check progress
- Review where we are in sprint
- Adjust if needed

Story points:

- Fibonacci sequence (1, 2, 3, 5, 8, 13, 21)
- Higher number = more critical/complex

Reports available:

- Velocity chart (team speed)
- Sprint report (sprint summary)
- Burndown chart (remaining work vs time)

Dashboards:

- Visual view of entire project
- Widgets for charts and reports
- Use JQL to group issues
- Customizable

Jira terminology:

- Ticket = Issue

- Filter by: issue type = Bug

Confluence integration:

- Store documents derived from Jira
- Link documentation to issues
- Centralized knowledge base

ServiceNow integration:

- Incident management (alert → assign → resolve → close)
- Change management (request → approve → implement → verify → close)

This completes the practical understanding of how Jira is used in organizations for Agile project management.