AI-Assisted Software Workflow Playbook

CONTENTS

OUR PERSPECTIVE

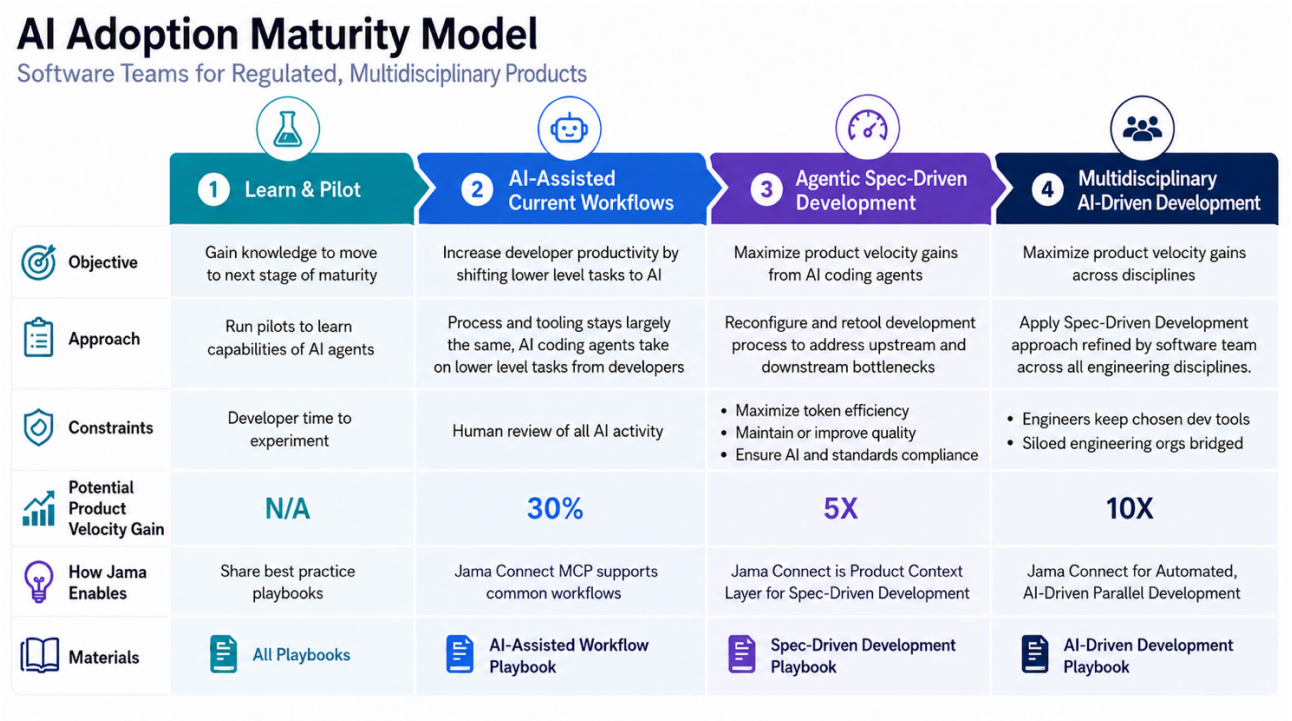
We work with over a thousand software development organizations transitioning to AI-Driven Development in regulated, multidisciplinary product industries. Our focus on the multidisciplinary product context layer from requirements through release provides a unique vantage point for best practices in AI adoption. The fundamental patterns we see are:

- **AI product velocity gains are real** | Companies must adopt AI-Driven Development to stay competitive.
- **Software leads** | Software teams are leading the way in AI adoption at regulated multidisciplinary product companies, and their success patterns are likely to be leveraged across engineering disciplines.
- **Coding is no longer the bottleneck** | The speed of AI coding agents has shifted the constraints upstream (requirements, context engineering) and downstream (testing, integration, compliance)
- **Phased adoption** | The most successful teams make specific choices of how much process reconfiguration and retooling they undertake in a given timeframe.

To support our customers, we have developed an AI Adoption Maturity Model and playbooks for each level of adoption leveraging Jama Connect. Jama Connect supports all upstream (requirements, context engineering) and downstream (integration, testing, compliance) activities from AI coding agents.

AI ADOPTION MATURITY MODEL

Below is an AI maturity model for software teams in regulated multidisciplinary product companies. It outlines 4 levels of adoption and the objectives, approaches, constraints, potential gains and how Jama Connect supports each level.



This playbook is focused on Level 2, AI-Assisted Current Workflows. Level 2 adoption maintains current processes and tooling and seeks to maximize developer productivity within these constraints.

The next level of adoption, Level 3, Spec-Driven Development reconfigures and retools to maximize product velocity gains from the speed of AI coding agents. This is a more significant transformation of the software development process. Some organizations choose to jump to level 3 from level 1 while others choose to progress through level 2. Both approaches are feasible, the choice depends on the organization's level of urgency and capacity for change.

Core technical enablers of level 2 adoption are the Jama Connect MCP™ and Jama Connect Advisor™. Jama Connect MCP enables AI coding agents (Claude, Codex, etc.) and AI IDEs (Cursor, GitHub Copilot, Virtual Studio, etc.) to engage with Jama Connect's product context layer to make high quality, consistent and token efficient inferences, reads and writes. Jama Connect MCP enables significantly increased efficiency in many existing workflows. Jama Connect Advisor provides AI enabled functionality for users that choose a UI interface (typically not software developers).

To achieve level 2 adoption requires:

1. A review of relevant workflows through which AI could deliver efficiency gains.
2. Specification of the AI enabled workflow elements via Jama Connect MCP and Jama Connect Advisor.
3. Compliance with AI governance and industry standards.

SUPPORTED AI-ASSISTED WORKFLOWS

For most software organizations there are still a significant number of workflows that require developers to switch context, search for information, refine requirements, plan sprints, transform information from one system to another, document for compliance, create test cases and more. All of these manual tasks impact productivity, overall quality and job satisfaction.

AI coding agents enabled by Jama Connect MCP can transform these manual tasks into ones performed by AI under developer direction and review to ensure compliance with AI governance and industry standards.

Below are a list of common workflows that are currently supported by Jama Connect MCP for implementation today.

Supported AI-Assisted Workflows:

- **Requirement Lookup**: Developers search for a requirement directly from their IDE and instantly retrieve the latest requirements details instantly without leaving the development environment.
- **Requirement Quality Review with AI Analysis**: An AI agent searches for requirements in a project, reads each one back, analyzes quality (completeness, testability, ambiguity, INCOSE compliance), and adds review comments with improvement suggestions.
- **Contextual Requirement Updates**: Developers can propose updates to requirements based on implementation insights. All changes follow Jama Connect lifecycle rules and version history.
- **Requirements Decomposition from Concept to Subsystem**: Engineers can describe a feature in natural language and have their AI agent decompose it into a full requirements hierarchy: stakeholder requirements → system requirements → system architecture →subsystem requirements, with full traceability at every level.
- **Sprint Planning from Jama Connect Requirements**: An AI agent reads the approved requirements from Jama Connect, estimates effort, and creates a sprint's worth of work items (User Stories, Tasks) in chosen external tool (Jira, ADO, etc.) with proper linking back to Jama requirement IDs.
- **Test Case Authoring with Verification Traceability**: Pull system requirements from Jama Connect, generate test cases for each, linking them via chosen relationship type and direction.

- **Cross-Iteration Requirement Refinement**: After a design review, an AI agent receives feedback, updates requirement descriptions and fields in Jama Connect, adds comments documenting the rationale, and once approved by an engineer, creates a new baseline to capture the revision.
- **Code-to-Requirement Linking**: Engineers create trace links between code changes and requirements at the moment implementation occurs, ensuring traceability stays synchronized with development activity.
- **Change Request with Impact Visibility**: An engineer identifies a change request. An AI agent automatically traces all impacted items in Jama Connect, creates a change request item, links it to affected requirements, creates corresponding bugs/tasks, and comments on all impacted items.
- **Change Awareness**: When requirements change in Jama Connect, developers can view the updated content and compare differences directly from their development environment.

USE CASE 1: REQUIREMENT LOOKUP

Value: Instantly retrieve the latest requirements details without leaving IDE, saving time and context switching costs

Trigger: Developer needs to confirm requirement acceptance criteria while reviewing software code in IDE.

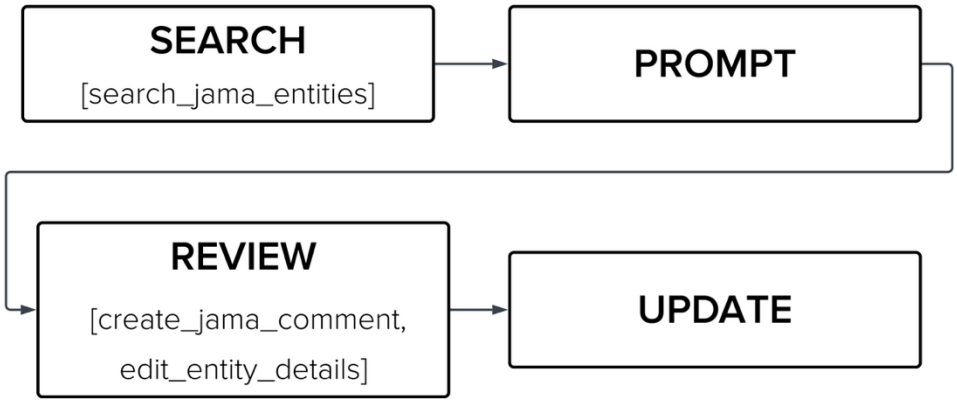
Steps:

1. SEARCH: Developer finds approved requirements based on the project and function using natural language
2. VIEW: Developer requests specific requirement from search result to read the detailed acceptance criteria

Next Steps: Quality Review and Test Case Authoring

USE CASE 2: REQUIREMENT QUALITY REVIEW

Value: Improve requirement clarity and completeness by safely leveraging AI to identify gaps, and ambiguities with human-in-the-loop to review and validate AI suggestions



Trigger: Developer receives new batch of requirements to implement (e.g. baseline or release package)

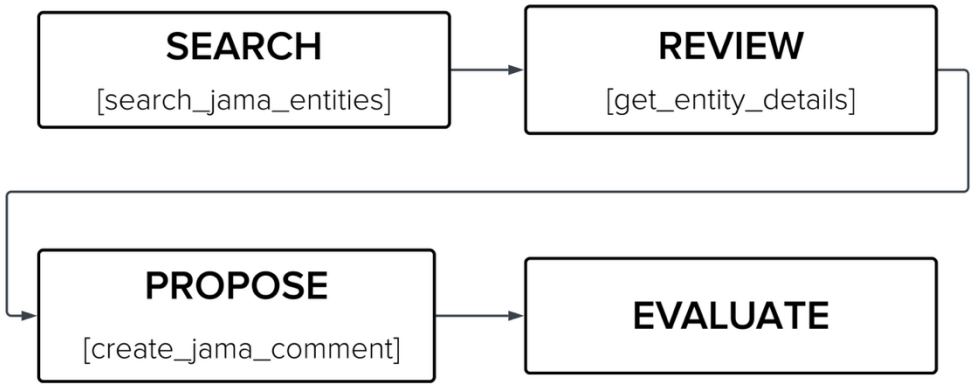
Steps:

1. SEARCH: Developer searches for relevant requirements using natural language within IDE.
2. PROMPT: Developer invokes AI to evaluate requirement quality (e.g., ambiguity, testability, completeness). The can prompt AI to use relevant skills e.g. “as an ISO26262 expert, please review the requirements for safety compliance…”
3. REVIEW: AI returns structured feedback highlighting issues such as unclear language, missing acceptance criteria, etc. AI posts feedback as comments on the Requirement and updates status to “Needs Review” without modifying original content.
4. UPDATE: Requirement owners see the list of Requirements that need review in Jama Connect, validate AI suggestions, and make changes.

Next Steps: Test Case Authoring

USE CASE 3: CONTEXTUAL REQUIREMENT UPDATES

Value: Developers can propose updates to requirements based on implementation insights. All changes follow Jama Connect lifecycle rules and version history.



Trigger: Developer identifies a gap, inconsistency, or improvement opportunity in a requirement while implementing or reviewing code in their IDE

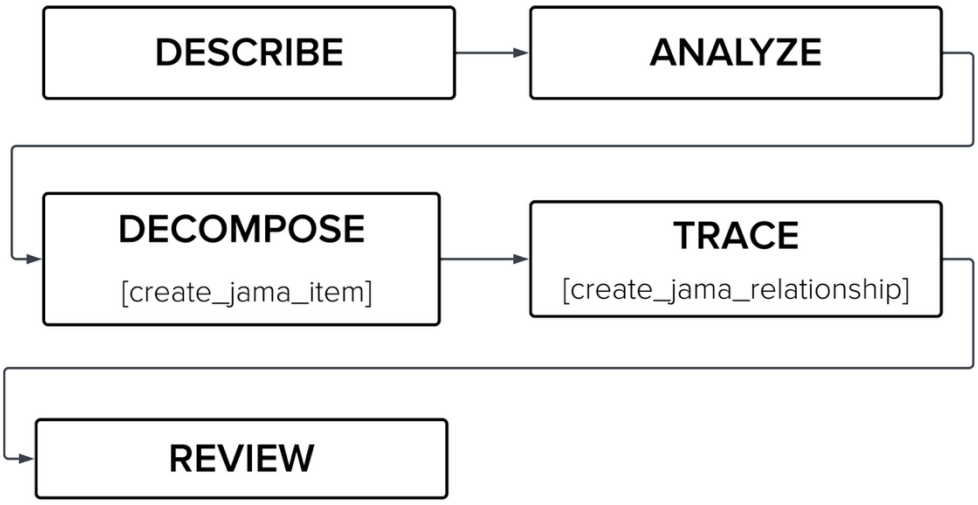
Steps:

1. SEARCH: Developer searches for relevant requirements using natural language within IDE (or by specific ID if already traced in code comments)
2. REVIEW: Developer reviews requirement details alongside implementation context in IDE
3. PROPOSE: Developer suggests updates or refinements to the requirement in natural language, AI via MCP adds comment directly in Jama Connect attributing comment to developer
4. EVALUATE: Requirement stakeholders evaluate and approve or reject the proposed update following their defined change workflow

Next Steps: Change Request with Automated Impact Analysis

USE CASE 4: CONCEPT TO SPEC DECOMPOSITION

Value: accelerate early-phase definition while maintaining traceability and governance in Jama Connect



Trigger: Engineers have a conceptual prototype and describe it in natural language for AI to decompose into a full requirements hierarchy.

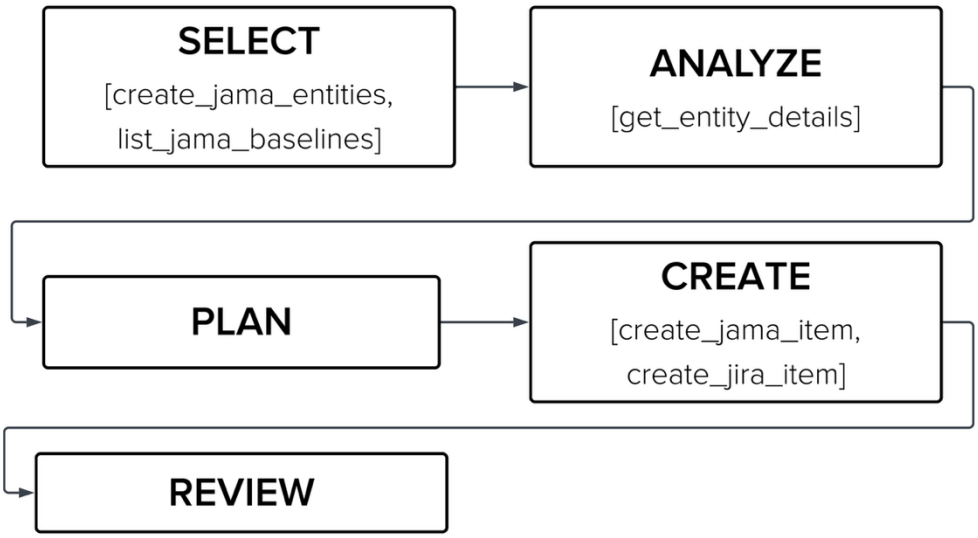
Steps:

1. DESCRIBE: Engineer inputs a natural language description of the concept or prototype in IDE
2. ANALYZE: AI interprets the concept and identifies key functional areas, constraints, and system boundaries
3. DECOMPOSE: AI generates a hierarchical breakdown (e.g., system → subsystem → component requirements) [create_jama_component, create_jama_set, create_jama_folder, create_jama_item]
4. TRACE: AI creates traceability links according to Jama Connect traceability information model (TIM) [list_jama_relationship_types, create_jama_relationships]
5. REVIEW: Relevant stakeholders see the content in Jama Connect was AI generated so is flagged for human review and acceptance. Stakeholders iterate and approve requirements through defined lifecycle workflows

Next Steps: Sprint Planning

USE CASE 5: SPRINT PLANNING FROM JAMA CONNECT REQUIREMENTS

Value: automatically translate approved requirements into estimated, ready-to-execute work, reducing manual effort, increasing alignment between product and engineering



Trigger: Baseline set of approved requirements are released for implementation

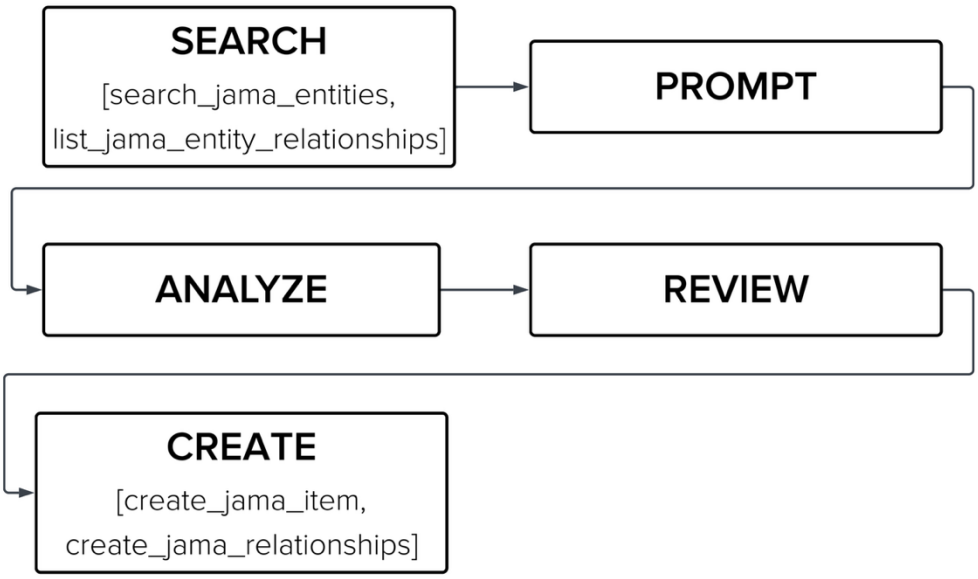
Steps:

1. SELECT: Engineering lead selects approved requirements in scope for upcoming sprint planning [search_jama_entities, list_jama_baselines]
2. ANALYZE: AI retrieves detailed requirement data and acceptance criteria. AI evaluates requirements to determine scope, complexity, and dependencies [get_entity_details]
3. PLAN: AI decomposes requirements into user stories and tasks aligned to development workflow
4. CREATE: AI creates work items in Jama Connect with trace link to source Requirement. The work items flow to software task used by developer (e.g., Jira or Azure DevOps) with recommended sprints [create_jama_item, create_jira_item]
5. REVIEW: Team reviews, adjusts, and commits work items into the sprint backlog

Next Steps: Test Case Authoring with Verification Traceability

USE CASE 6: TEST CASE AUTHORING WITH VERIFICATION TRACEABILITY

Value: Improve product quality and audit readiness by automatically generating comprehensive test cases



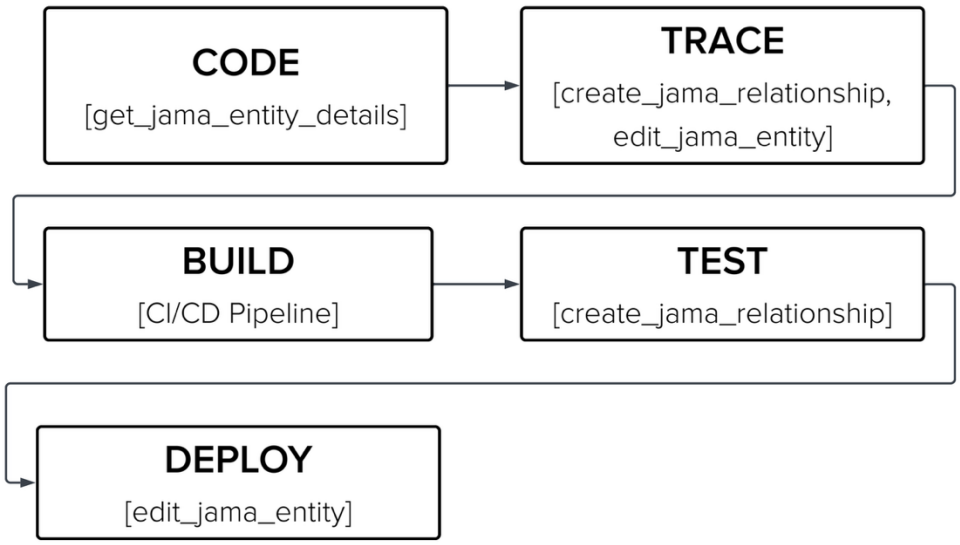
Trigger: QA engineer or developer needs to create test coverage for approved system requirements

1. SEARCH: QA Engineer/Developer working in IDE uses natural language to find specific approved system requirements that are missing verification traceability [search_jama_entities, list_jama_entity_relationships]
2. PROMPT: QA Engineer/Devleoper selects requirements to generate corresponding test cases and provides specific test types and industry context
3. ANALYZE: AI evaluates requirement details and acceptance criteria to determine test scenarios. AI recommends test cases including steps, expected results, and edge conditions.
4. REVIEW: QA Engineer/Developer reviews the recommended test cases, iterates where necessary, and then prompts AI to create the test
5. CREATE: AI creates test case items in Jama Connect with link to source requirement(s) [create_entities,create_jama_relationships]. Test Coverage established and ready for execution

Next Steps: Code to Requirement Linking

USE CASE 7: CODE TO REQUIREMENT LINKING

Value: Engineers create trace links between code changes and requirements at the moment implementation occurs, ensuring traceability stays synchronized with development activity



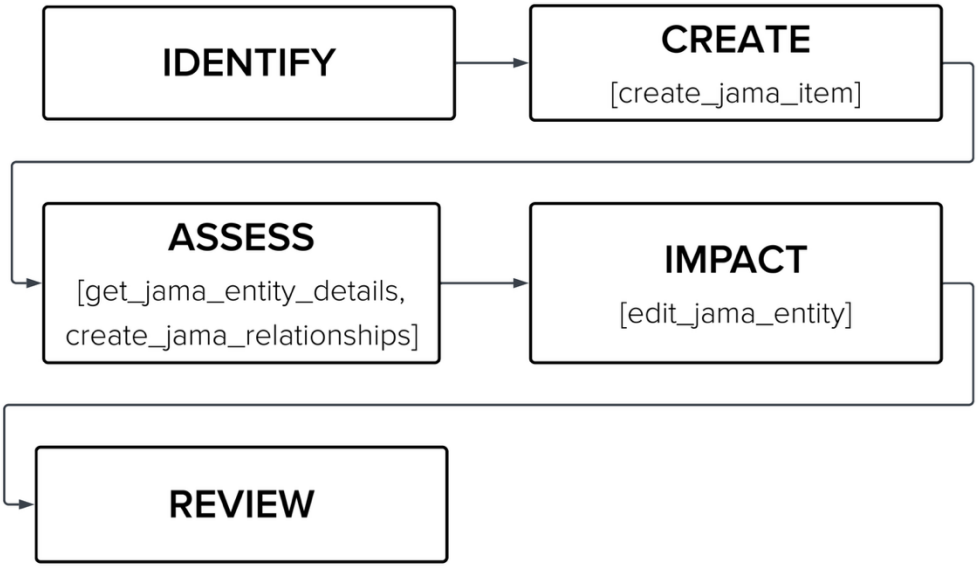
Trigger: Developer has finalized requirements and test cases from Jama Connect baseline and is ready for implementation [list_jama_baselines, get_jama_entity_details]

1. CODE: Developer prompts AI within IDE to create new or update existing code per the requirements from Jama Connect baseline. AI starts updating the code per new and changed requirements
2. TRACE: AI applies requirement ID comment tag directly by relevant code while making changes. AI updates requirement in Jama Connect with link to relevant code that implements the requirement [create_jama_relationship, edit_jama_entity]
3. BUILD: AI commits code to repo and includes relevant requirement IDs in the code commit
4. TEST: AI builds and executes unit tests. For failed unit tests it retries prior steps until successful. AI summarizes work and test results and ask Developer to review changes for publishing test results. Developer confirms to publish test results [create_jama_relationship]
5. DEPLOY: Developer confirms for AI to deploy the changes. AI updates Jama Connect with verification test results and adds deployment details on the relevant impacted requirements [edit_jama_entity]

Next Steps: Change Request with Automated Impact Analysis

USE CASE 8: CHANGE REQUEST WITH AUTOMATED IMPACT PROPAGATION

Value: accelerate change management and reduce risk by automatically identifying impacted requirements, maintaining traceability, and coordinating follow-on work



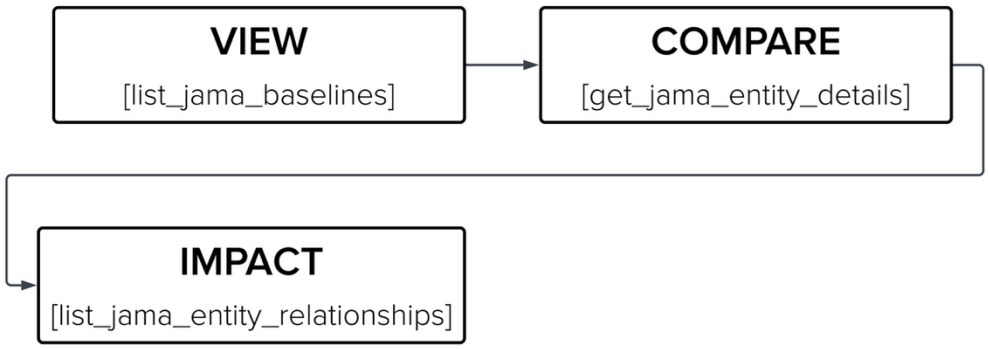
Trigger: Developer working in IDE recognizes a change is needed based on early user feedback.

1. IDENTIFY: Engineer documents the change request context and rationale within IDE and prompts AI to do a change impact analysis
2. CREATE: AI generates a structured change request and submits it to Jama Connect [create_jama_item]
3. ASSESS: AI evaluates existing requirements to determine if impacted by change request. For those that have potential impact, it creates trace relationships to the Change Request [get_jama_entity_details, create_jama_relationships]
4. IMPACT: AI reviews all upstream and downstream related artifacts for the impacted requirements. It updates each relevant item with a comment of proposed changes. AI summarizes full impact in Change Request description [edit_jama_entity]
5. REVIEW: Project stakeholders review the change request and impacted items for approval or rejection decision

Next Steps: Change Awareness and Impact

USE CASE 9: CHANGE AWARENESS AND IMPACT

Value: Reduce regression risk by clearly identifying what requirements have changed with impacted tests and code.



Trigger: Approved change request and updated requirements

1. VIEW: Developer views list of change request baselines directly in IDE [list_jama_baselines] and asks to compare prior baseline with current
2. COMPARE: AI summarizes the changes between 2 specific baselines. AI offers to show details of specific changes [get_jama_entity_details]
3. IMPACT: Developer asks AI to show impacted tests and code libraries, AI provides list of impacted assets [list_jama_entity_relationships]

Next Steps: Change Awareness and Impact

SUMMARY

AI-assisted workflows are the easiest way to get started with AI for productivity gains with existing process, tools and compliance constraints. Our playbook provides guidance on workflows to focus on for implementation. Clients may also leverage their Success Program to gain help from our consultants through the process.

Once level 2 maturity is reached with AI-assisted workflows, it provides an easier transition to level 3 maturity – Spec-Driven Development.