

~1,000 JIRA USERS · ENTERPRISE SOFTWARE

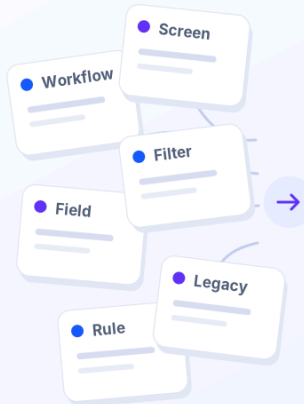


CASE STUDY

Jira governance: Back under control.

⚙️ ~1,000 Jira users

Governance first. Cleanup second.
Keep the legacy that still does real work.



Governed configuration

- ✓ **Workflows**
Defined variation
- ✓ **Fields & screens**
Cataloged purpose
- ✓ **Filters**
Sharing standards
- ✓ **Automation**
Accountable owner

Evidence. Dependencies. Named owners.

✓ Evidence before deletion

✓ Named owners

✓ Legacy kept deliberately

AVARATAK

Governance first. Cleanup without breaking legacy.

A governance-first cleanup of workflows, screens, filters, and automation that preserved the legacy still doing real work.

~1,000 Jira users

Jira Cloud Platform

An enterprise software company had reached the point where nobody could change anything with confidence. Avaratak brought the environment back under control with an evidence-based configuration inventory, shared standards, and a named owner for everything retained.

The change that mattered. Administrators could evaluate a change's blast radius against documented dependencies instead of discovering them.

01 / THE CHALLENGE & OUR APPROACH

Governance before cleanup. Evidence before deletion.

Some of the configuration sprawl was still doing essential work. Removing it safely required more than a list of things to delete.

Change had become a risk

An enterprise software company with roughly 1,000 Jira users had reached the point where nobody could change anything with confidence. Years of decentralized administration had produced hundreds of near-duplicate workflows, cloned screens, thousands of saved filters, and an automation estate that behaved unpredictably.

The environment had grown the way most large Jira environments grow: one reasonable local decision at a time, for years. Each new team received a cloned configuration and adjusted it, leaving workflows that differed by a renamed status, screens carrying fields nobody filled in, and three separate custom fields recording when work was due. Filters numbered in the thousands, many broken or shared with the entire site, silently feeding dashboards whose owners had long since changed roles.

Automation was the sharpest pain: overlapping triggers, rules reacting to edits made by other rules, global rules written for one project's problem, and rules whose creators had left the company. Administrators practiced a policy of add, never remove, because the blast radius of any deletion was unknowable. Change requests stalled, trust in Jira data eroded, and spreadsheet workarounds started to reappear.

“Zero unowned configuration,
not zero legacy.”

Avaratak's approach

Make the dependencies visible

The obvious instruction was to delete the junk. Two things made that dangerous: nobody could say what the junk was, and some of the mess was load-bearing. Legacy automation encoded business logic that predated everyone in the room, and legacy fields fed reporting leadership relied on. So governance came before cleanup. We built a complete configuration inventory, with usage evidence and dependencies for every workflow, screen, field, filter, and automation rule, and interviewed administrators and team leads about how work actually flows.

Every object was then classified into one of four outcomes: retain, consolidate, retire, or redesign. Two rules governed the exercise: nothing was retired without usage evidence, and nothing was retained without a named owner. The design that followed established a governed workflow library with defined variation points, a field catalog with naming standards, automation standards with deliberate scope and accountable owners, and a legacy exception register that let legacy automations and fields with real dependencies stay, deliberately and on the record.

Implementation ran in staged waves, each rehearsed in the Jira Cloud sandbox before production: obsolete projects archived first, projects moved onto governed workflow schemes in groups, duplicate fields merged with values migrated where history mattered, filters retired with notice, and automation removed on a disable, observe, delete pattern.

02 / THE GOVERNANCE MODEL

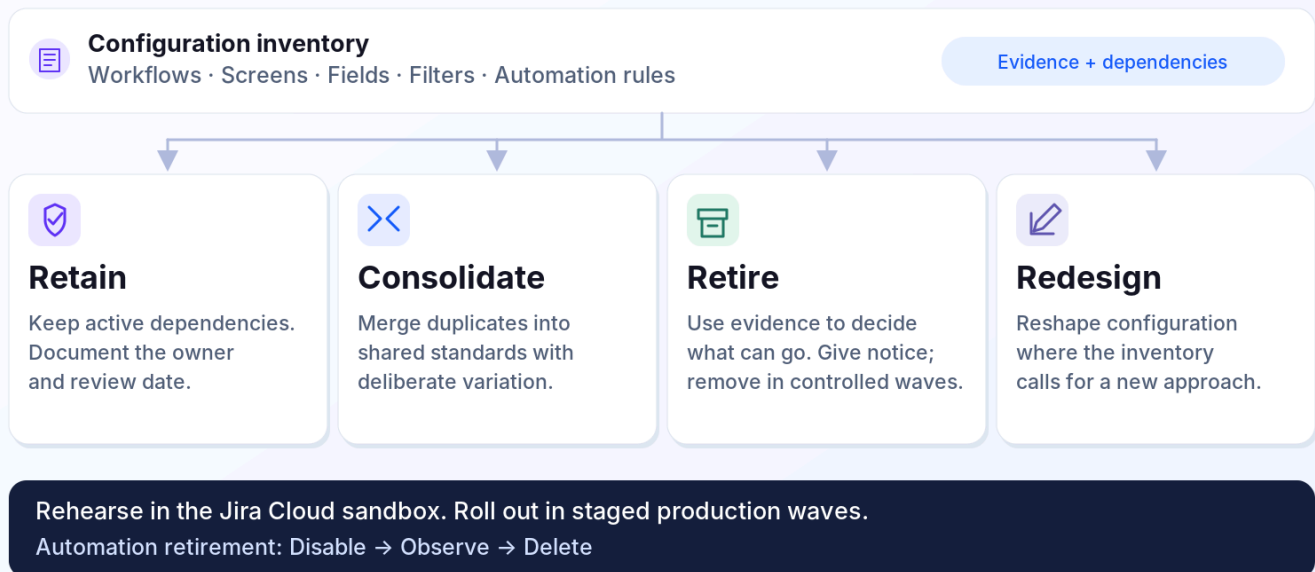
Four decisions. A reason behind every one.

Every object was classified as retain, consolidate, retire, or redesign. Usage evidence and dependencies informed the decision.

GOVERNANCE BEFORE CLEANUP

A decision for every configuration.

Usage evidence and dependencies turn cleanup into a controlled process.



Nothing retired without usage evidence.
Nothing retained without a named owner.

Nothing retired without evidence

Configuration decisions started with usage evidence and dependencies. Filters were retired with notice; automation followed a disable, observe, delete pattern.

Nothing retained without an owner

Legacy fields and automation with active dependencies stayed deliberately, with a documented dependency, a named owner, and a review date.

03 / THE SOLUTION

Six connected foundations. One governable environment.

Shared standards, accountable ownership, and a deliberate place for the legacy the business still needed.

01

A complete configuration inventory

Every workflow, screen, field, filter, and automation rule mapped with usage evidence and dependencies, plus the retain, consolidate, retire, redesign decision record.

02

A governed workflow library

A small standard set with defined variation points, replacing per-project cloning without forcing identical workflows on teams that genuinely work differently.

03

A field catalog

Naming standards, context discipline, merged duplicates, and a request path for anything new.

04

Automation standards

Project scope by default and global by exception, a naming convention, standardized rule actors, and a named human owner for every active rule.

05

A legacy exception register

Legacy automations and fields with active dependencies formally retained: dependency documented, owner assigned, review date set.

06

An administration playbook and change intake

Standards built into the request itself, with a quarterly configuration review to catch drift early.



04 / THE RESULTS

Change became evaluable, not unknowable.

What changed once ownership and evidence replaced guesswork.

-
- 01** **Workflows consolidated into a governed library**
Covering every active project, with deliberate variation instead of per-project cloning.

 - 02** **Every remaining field owned and cataloged**
A stated purpose, a documented owner, and known screens for each custom field.

 - 03** **Automation people can rely on**
Overlapping rules consolidated, redundant executions eliminated, and an accountable person behind every active rule.

 - 04** **Filters worth trusting**
Broken, orphaned, and over-shared filters retired or corrected, with sharing standards for everything new.

 - 05** **Legacy kept on purpose**
Automations and fields with genuine dependencies retained under documented exceptions, with owners and review dates.

 - 06** **Unowned configuration reduced to effectively zero**
Administrators now evaluate a change's blast radius against documented dependencies instead of discovering them.



05 / KEEPING CONTROL

Keep the legacy. Lose the guesswork.

The objective was not to remove every older field or rule. It was to understand what each one did, who owned it, and when it should be reviewed.

LEGACY BY EXCEPTION, NOT BY ACCIDENT

A deliberate place for work that still matters

Legacy automations and fields with genuine dependencies were formally retained. The exception register recorded the dependency, owner, and review date so that keeping them became a conscious decision.

STANDARDS IN THE CHANGE REQUEST

A repeatable way to keep the estate under control

A field catalog, workflow library, automation standards, and an administration playbook made ownership and purpose visible. Change intake carried the standards into the request itself, with quarterly configuration reviews to catch drift.

Bring your Jira environment back under control.

If nobody can say what is safe to change, cleanup is guesswork. We'll help you determine what should be retired, what should be consolidated, and what genuinely deserves to stay, before cleanup decisions become outages.

[Book a 30-minute conversation](#) →**About Avaratak**

Avaratak is an Atlassian Solution Partner. Engagements are delivered by Senior-level or higher consultants with at least five years of hands-on Atlassian experience.

[Read the full case study](#) · [Explore Atlassian consulting](#)