CodeLogic

Case Study:
Growing Software Company

Overview

A dev team at a small software company was wrapping up a rescue project for a failing application: they arrived at the painful conclusion that they needed to re-write it. **Team leadership had to reconcile two contradictory needs: moving fast while keeping things clean.** For speed, they needed to empower several squads to make strategic development and architecture decisions quickly. At the same time, they needed to keep the architecture solid to avoid tanking this kind of critical application again.

Highlight

After throwing out a whole codebase, this dev team **decreased triage time by 53%** and established a continuous flow that allows them to move fast and deliver predictably while ensuring that debilitating technical debt never swallows up productivity — or their codebase — ever again.

Continuous Scanning

Application Dashboard

IDE Integration

Approach

For the new effort, the team chose to implement the CodeLogic software intelligence platform. The lead architect regularly reviewed CodeLogic's dependency and complexity insights. If the data access layer showed inappropriate coupling across the system, he'd work with the team to keep things clean. As methods got long or overly complex, the team would refactor. When similar integrations diverged on which API endpoints they used, the team leads could be brought together briefly to decide on the right path and move forward with more clarity and confidence. *(continued)*

"Before CodeLogic, when we'd overhaul a shared library, it would always be a mess. We'd miss things that would only hit us when building other modules, or even in testing. It was difficult to know when we were truly done. Now it's so much easier to see the 'blast radius' of a change and plan accordingly"

Over the course of a few months, the frequency of interventions decreased. The developers learned that taking on technical debt to get a task done a little quicker resulted in extra work when being asked to clean up that debt. While occasionally taking on tech debt is the right decision, the team learned to consciously make that decision together rather than individually.

Results

Unlike architects who draw diagrams and hope for the best, this team's lead architect used the CodeLogic Application Dashboard to identify bad patterns as they emerged and address them quickly.

"Some of this we should have caught in code review, but many of our issues looked fine commit to commit. Only when you're able to see the full picture across libraries and services do these issues become obvious." - Lead Architect

Highlight

17% increase in dev team productivity

53% decrease in weekly triage mtgs

Using CodeLogic when planning bigger changes also helped the team execute those changes better. "Before CodeLogic, when we'd overhaul a shared library, it would always be a mess. We'd miss things that would only hit us when building other modules, or even in testing. It was difficult to know when we were truly done. Now it's so much easier to see the 'blast radius' of a change and plan accordingly" observed a lead developer.

Too often, smaller changes to a shared library still caused issues. While checking CodeLogic before making a change would have prevented those issues, adopting the new habit of checking the platform was difficult for many team members. Following a retrospective, the team decided to add CodeLogic's integration to their IDE: JetBrains IntelliJ. Now, when the team checked references in the IDE as part of their existing workflow, CodeLogic provided additional telemetry within the IDE. When an interesting usage surfaced, they could slide into the CodeLogic WebUI to learn more.

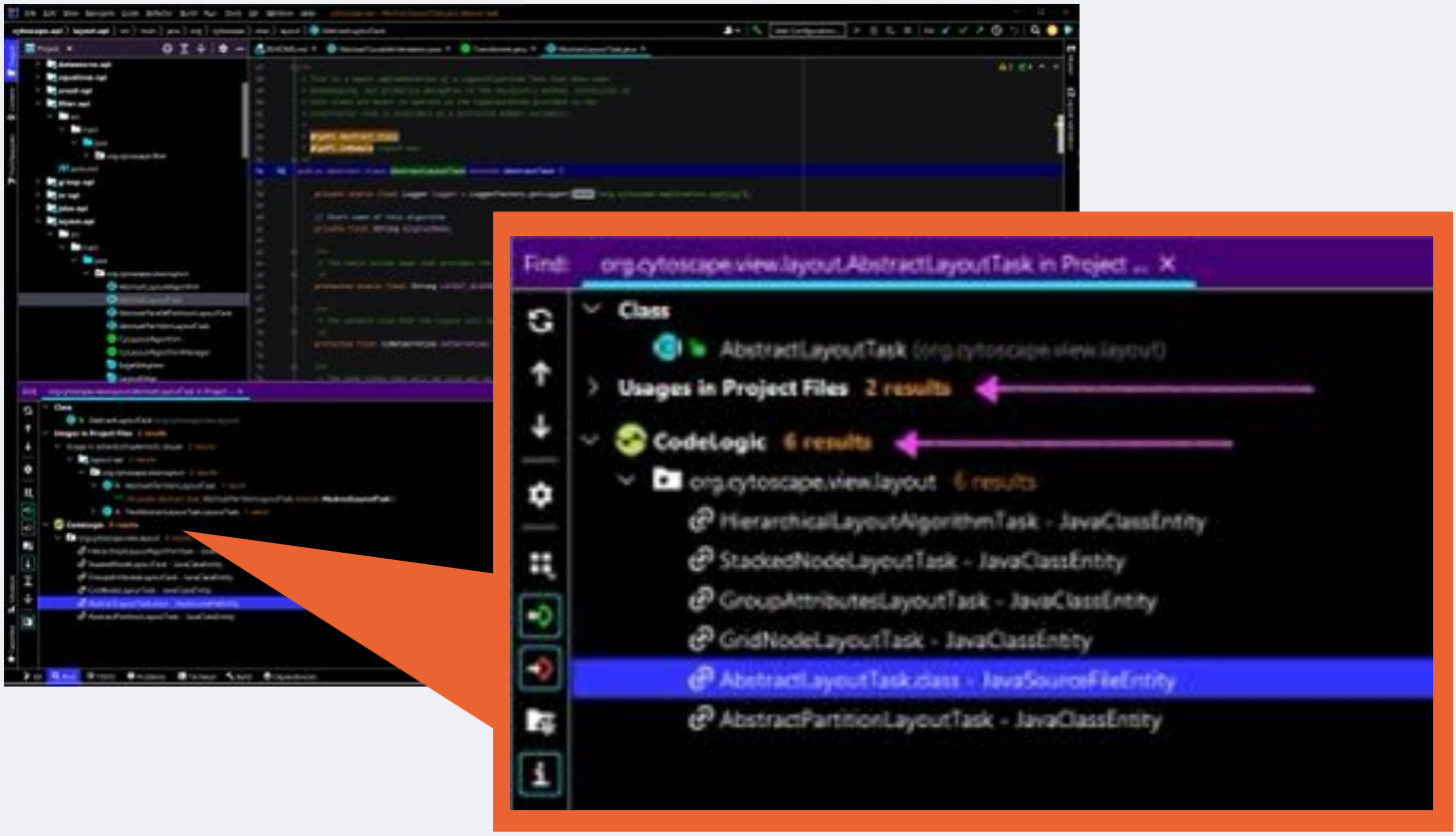
As one developer put it, **"Accessing CodeLogic within IntelliJ means we don't have to break our focus to find the information we need – We're already there."**

Conclusion

With CodeLogic now part of their continuous workflow, the source of defects dried up, saving significant development and testing time, and shortening weekly triage meetings by 53 percent. Because those errors often impacted people on other squads, reducing them helped build confidence across teams and keep the larger organization harmonious and happy.

For the executive overseeing the team, the CodeLogic benefits are clear, **“Day-in and day-out, we’re saving 15% of the engineering team’s time. But the real benefit is that we’re keeping the code clean enough to go fast, deliver predictably and never throw out a whole codebase again.”**

Today, this organization is still moving quickly. The foundations of good architecture have been well maintained. They have avoided creating the sort of creeping technical debt that swallows productivity and leaves everyone miserable. Instead, the teams are empowered to make good decisions, with the right information at their fingertips.



The CodeLogic IntelliJ integration identifies usages that the native IDE typically does not detect.