Embracing intelligent Engineering for Accelerated Growth and Efficiency

By strategically adopting Sahaj's intelligent Engineering approach to AI-augmented software development, Axi Financial Services created a blueprint for operational agility, significantly reducing the average time to integrate new Payment Service Providers (PSPs) by over 20%.

November 2025

Axi, a global online broker specializing in forex and CFD trading, faced a serious bottleneck integrating multiple Payment Service Providers (PSPs) which was slowing growth. By leveraging Sahaj's intelligent Engineering approach to AI-augmented software development, Axi not only reduced the average time to integrate new PSPs by over 20%, integrating 8 new providers in less than 11 months, but also fostered a culture of continuous improvement and skill amplification across their Engineering teams.

This success story provides a compelling blueprint for how organizations can harness AI across the software development lifecycle to drive significant, measurable impact on their operational agility and market responsiveness.

Background

Axi Financial Services, a well-established global online broker headquartered in Sydney, Australia, serves tens of thousands of active clients in over 100 countries. Specializing in forex and CFD trading, Axi is committed to providing a smooth and reliable trading experience through user-friendly technology, comprehensive educational resources, and responsive customer support. Their strong emphasis on transparency and regulatory compliance ensures a trusted trading environment for both novice and experienced traders worldwide.

The Business Challenge

As Axi pursued strategic growth into new and existing markets, the integration of multiple PSPs became a critical bottleneck. This challenge was compounded by the diverse systems involved, necessitating improvements across websites, mobile applications, and various backend microservices. Ensuring a reliable payment experience across diverse geographies was not only essential for revenue growth and customer trust but also had to be delivered under stringent regulatory and time-sensitive market constraints.

The intelligent Engineering Solution

To address this challenge, Sahaj deployed a focused team of polyglot engineers. This team dedicated significant effort to understanding Axi's intricate business context, technical ecosystem, and the underlying architecture supporting the trading platform. They identified a common pattern in PSP integration workflows, which presented a prime opportunity for leveraging generative AI tools to accelerate development. The initial objective was to validate the efficiency gains offered by these AI tools and assess the viability of their continued adoption. The team strategically rolled out well-integrated tooling for specific tasks across various technology stacks and delivery concerns. This chosen tooling was seamlessly integrated into the Integrated Development Environment (IDE), establishing a fluid development workflow.

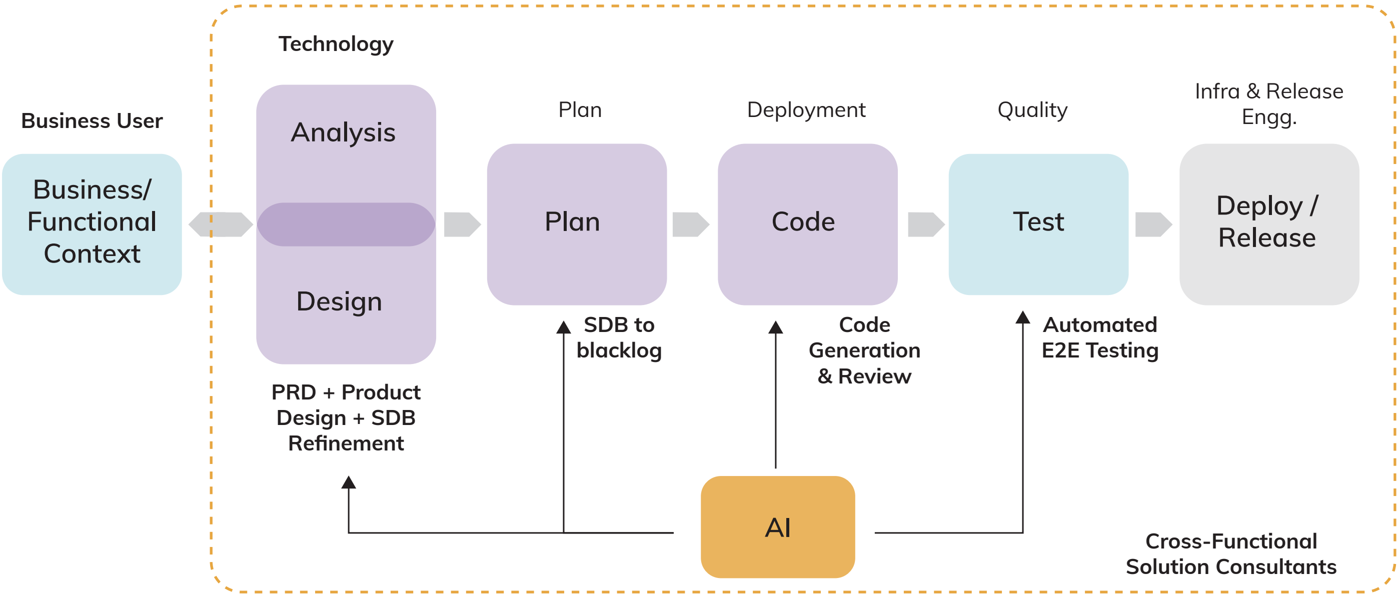
Beyond code generation, the application of generative AI tools has expanded to automate the creation of Solution Design Briefs (SDBs) and development stories directly from Product Design Documents (PDDs). This highlights AI's capability to browse relevant documents and refine designs, while still necessitating human oversight for formatting and optimal decision-making.

Tech Stack

The teams navigated a polyglot technology stack comprising:

- Frontend: React, React Native
- Runtimes: .Net, TypeScript
- Provisioning and Deployment: Azure DevOps, Azure AKS, FluxCD, Terraform
- Cloud Provider: Azure
- AI Tool Stack:
    - Claude Code
    - Copilot with various models and agentic mode

AI Beyond Code Generation - Product Lifecycle



SDB - Solution Design Brief
PRD - Product Requirement Document
E2E - End-to-end Testing

Impact

The strategic adoption of generative AI tools and innovative workflows delivered significant, measurable improvements in Engineering efficiency and development speed, facilitating rapid PSP integration. The team successfully reduced the average time to add a new PSP by more than 20%, leading to the integration of 8 new PSPs in less than 11 months.

Following on from this project, Axi has since:

- Rolled out Copilot and its instruction files to all other development teams, a testament to the successful results achieved in this use case.
- Developed newer workflows, effectively combining human intuition with the rapid and contextual generation capabilities of AI tools. Notable benefits included:
  - Improved development speed for engineers, with reduced time spent learning new frameworks.
  - The tooling acted as an amplifier, enabling developers to transfer existing skill sets to previously unfamiliar platforms and contribute effectively without losing momentum.
  - Time saved on fixing linting and formatting errors.
  - Enhanced developer learning from generated code.
  - Less time wasted on tedious tasks and boilerplate code.
  - Improved code quality.
  - Faster onboarding with minimal hand-holding.
  - Quicker and more insightful impact analysis.

Key Learnings

The successful integration of AI assistants necessitates a phased approach to proficiency, the critical incorporation of internal context for standardization, and the decomposition of complex tasks into smaller, manageable steps. While powerful for general coding and design tasks, these tools currently exhibit limitations in generating comprehensive test suites and refactoring highly complex or entangled code, thereby requiring human oversight and strategic planning.

- A steep learning curve exists to achieve proficiency with AI assistants. By gradually integrating AI assistants for minor changes and then progressively expanding their use to larger refactorings, planning, and SDB reviews, the team gained confidence in the AI assistants' capabilities.
- Adding Axi-specific internal context for AI assistants through instruction files across code repositories proved instrumental in standardizing best practices for development and enhancing the quality of generation and planning results.
- Working in small, iterative steps with the AI agent helps minimize rework and maximize benefits.
- Currently, AI tools are not optimal for generating comprehensive tests. Attempting to generate an entire test suite almost invariably leads to suboptimal outcomes.
- Manually defining storybook contracts before asking the AI to generate UI components significantly reduces iterative refinement.
- At times, the tools struggle to refactor lengthy methods with entangled responsibilities. Providing a high-level refactoring plan upfront helps streamline the process.

Conclusion

Sahaj’s rapid adaptation, swift execution, strong collaboration, innovative workflows, and the strategic deployment of generative AI tools have significantly enhanced Axi's delivery process. This enabled Axi to seamlessly integrate with multiple payment service providers across diverse geographies within a short timeframe. This initiative has built a strong case for the broader adoption of AI tools across Axi's entire product lifecycle and various departments, with the ultimate goal of making intelligent Engineering the default mode of operation.