PolyAPI

Global Hotel Alliance Replaces Mulesoft with PolyAPI Using a Governed Multi-Agent Migration Framework

Luis Weir • Chief Architect, PolyAPI

Andrea Angiolini • SVP of Systems, Global Hotel Alliance

Abstract

This paper examines how Global Hotel Alliance (GHA) and PolyAPI partnered to migrate GHA's integration flows from MuleSoft to PolyAPI using a purpose-built, governed multi-agent framework. It explores why enterprise integration migrations have historically been so difficult to execute and how AI-assisted engineering is fundamentally changing that calculus. The paper describes the Governed Multi-Agent Migration Framework the two teams implemented together, presents the outcomes achieved, and reflects on what this model of human and agent collaboration means for the future of enterprise engineering.

About the Authors

Luis Weir is Chief Architect at **PolyAPI**, where he sets the technical direction for strategic client engagements spanning integrations, orchestrations, data pipelines, and AI services across hospitality, food and beverage, and retail.

Before joining PolyAPI, Luis served as **Head of Applied AI for Oracle's Hospitality** business. Prior to that, he launched and led the scale-up of the **Oracle Hospitality Integration Platform (OHIP)** — a project originally conceived by PolyAPI CEO and founder Darko Vukovic. His broader career includes a range of senior architectural and technical leadership roles, most notably as CTO of the Oracle practice at Capgemini.

Andrea Angiolini is the SVP of Systems at Global Hotel Alliance , where he has been for nearly 13 years overseeing the organisation's technology architecture and systems, spanning loyalty management, distribution, integration, and business intelligence.

Andrea brings over thirty years of experience in hospitality technology. Before joining GHA, he spent almost a decade at **Oracle MICROS**, where he held senior roles spanning product development, business intelligence, pre- and post-sales support, and SOA architecture. His career has given him a rare combination of deep systems architecture expertise and hands-on experience navigating the complexity of **large-scale enterprise technology** in the hospitality sector.

Introduction

Enterprise technology modernisation projects have long been constrained by cost, delivery speed, platform complexity, and operational risk. For decades, SaaS and PaaS vendors have justified premium pricing not only through the business value they create, but also through the switching costs associated with their products. Nowhere is this more pronounced than in the iPaaS category, where years of business logic become deeply embedded in proprietary languages and runtimes that are expensive and complex to migrate away from. For enterprises, the challenge has rarely been a lack of desire to modernise — it has been that the tools to make large-scale migrations economically and temporally viable simply have not existed. The rise of AI-assisted engineering and AI agents has changed that, unlocking approaches to modernisation projects that were previously too expensive and too time-intensive to justify.

Global Hotel Alliance (GHA) found themselves at exactly this modernisation decision point in early 2025. Their legacy iPaaS, MuleSoft, had become expensive and its architecture had begun to constrain their ability to innovate. They wanted a fundamentally more modern integration platform — one built on open architecture and designed to support development in native languages rather than proprietary ones. Equally important was the ability to flexibly leverage best-in-class AI tools and open source models as they continue to be released and improved.

PolyAPI offered exactly that. Built as a Kubernetes-native, cloud-agnostic operating system for integration and microservice development, the platform allowed business logic to be written in native TypeScript or Python, owned and controlled entirely by GHA. Beyond the out-of-the-box tooling Poly provides to streamline and simplify development, the operating system's core architecture was built to enable the controlled use of AI as a first-class capability. Rather than being dependent on a vendor's embedded AI offerings, GHA can determine which models to use, when, and for what purpose, with granular control over how agents interact with underlying systems and control over token routing and consumption. Poly addressed not only GHA's immediate migration needs, but provided the architectural foundation for building innovative new capabilities going forward.

To accelerate the migration, architecture leaders from both organisations, Andrea Angiolini (GHA) and Luis Weir (PolyAPI), partnered to pioneer a new approach to integration modernisation: a deliberately designed agentic AI migration framework with human-in-the-loop governance.

Early in the project, Andrea highlighted one of his favorite quotes from Abraham Lincoln, which became a central maxim for the creation of this new agentic framework, "Give me six hours to chop down a tree and I will spend the first four sharpening the axe".

Rather than pursuing full autonomy, the framework was built around a new model of human and agent collaboration.

The framework introduced:

- Agent-agnostic architecture
- Deterministic "Skills" with executable scripts
- Specialised analysis and implementation agents
- Structured manifests and intermediate blueprints
- Human-in-the-loop architectural validation and approval processes

The framework resulted in migrated flows completed in roughly one-third (⅓) of the original timeline estimates. The project has already successfully migrated approximately 70 API operations and workflows across 18 work packages in just over 130 days following the project's inception. Migration activities are ongoing, with efficiencies compounding with each additional flow.

The Modernisation Challenge

Migrations of this kind carry a cost structure that makes them uniquely difficult to greenlight. Two platforms running in parallel, engineers split between keeping existing integrations stable and building on the new platform, and normal delivery priorities competing for the same bandwidth throughout. Any delays in a migration of this scale extend the window during which an organisation pays for both systems — with no guarantee of when the transition will be completed.

MuleSoft is also a specialised platform with a proprietary development language, and accurately interpreting existing flows requires someone who knows it deeply. That person is rarely the right one to implement on a new platform as well. In practice, you need two specialists working in close coordination throughout the transition, compounding cost and complexity further. This is why so many modernisation projects get deferred — not because the case for change is unclear, but because the economics have historically been very difficult to justify and execute.

For GHA, MuleSoft's pricing model created a cost curve that penalised growth and began influencing architectural decisions in ways that were no longer sustainable.

"We could predict the cost and we could see that it would soon become unmanageable." — Andrea Angiolini, SVP Systems, GHA

Beyond cost, the platform itself had become a constraint on innovation. Developing new capabilities on MuleSoft was slow and expensive, with complex multi-system flows frequently requiring specialist SI or consulting engagements. At the same time, business

demand for new integrations and capabilities was not slowing down. The longer GHA remained on MuleSoft, the deeper the technical debt grew and the harder it became to address it without disrupting ongoing delivery.

Sharpen The Axe - The Case for Modern Tools

Andrea understood that the migration to PolyAPI would ultimately lead to more sustainable costs, faster deployment times, and greater flexibility in building future capabilities. He also recognised that writing code is most valuable, and most rewarding, when engineers can focus on solving business problems rather than wrestling with boilerplate and infrastructure work. PolyAPI's underlying architecture was designed to give GHA the ability to use AI as a core capability without being locked into a vendor-specific agent or language. Business logic would be written in native Python (GHA's preferred language), owned fully by GHA, with the freedom to evolve as their objectives changed. Critically, the Governed Multi-agent Migration Framework eliminated the need for the third-party MuleSoft specialists or systems integrators that migrations of this kind would typically require. The expertise bottleneck that has historically made these projects so expensive was addressed by the framework itself.

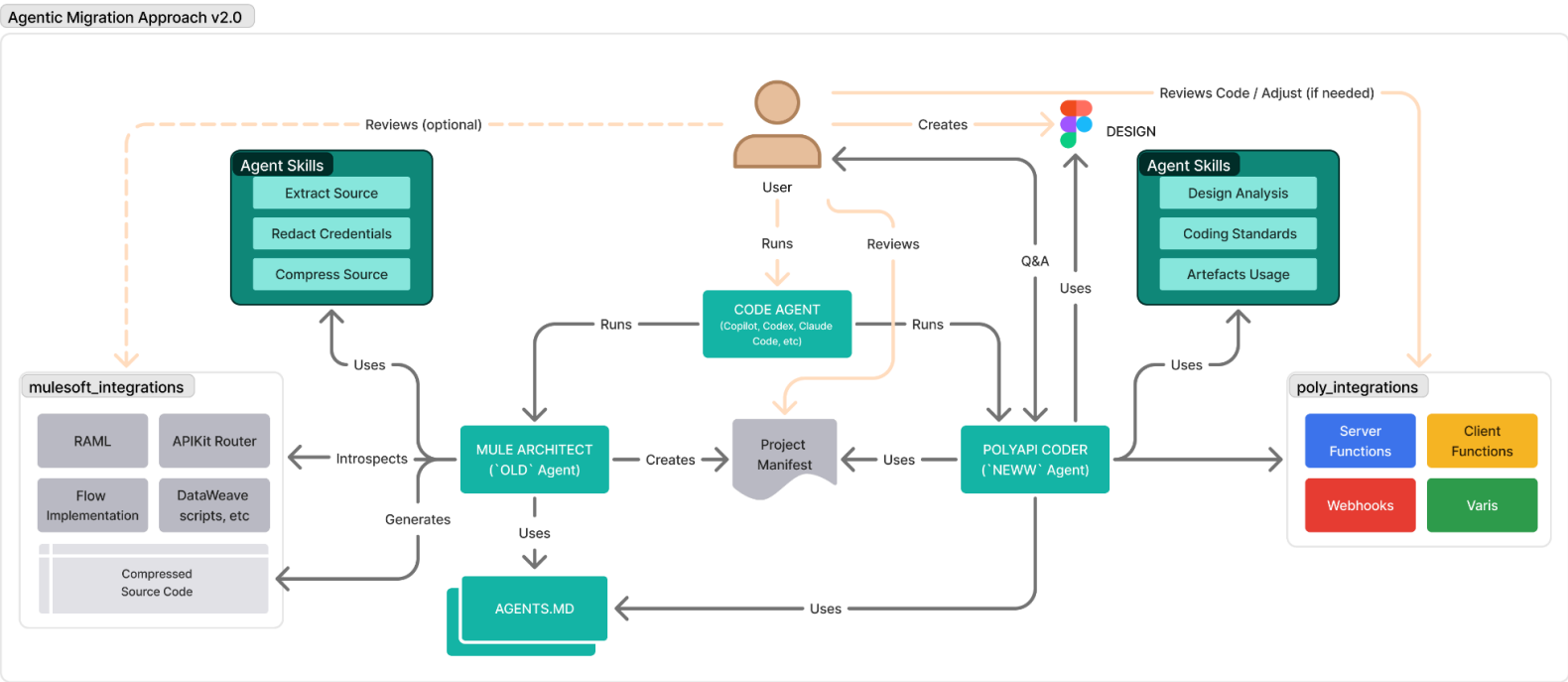
Rather than treating this as a conventional migration effort, Andrea and Luis made a deliberate decision to invest heavily in building the migration framework before jumping into code. Agent design, prompt engineering, Skills creation, manifest structures, architectural patterns, reference implementations, governance workflows — none of this produced a migrated integration directly, but all of it determined how fast and how reliably every subsequent integration could be delivered.

"We spent a lot more time building the environment than we did using it to generate code. That investment is exactly what made the speed possible." — Andrea Angiolini, SVP Systems, GHA

One decision in particular proved foundational: the first integration was implemented manually, without AI, as a standard-setting exercise. This defined the quality benchmark for integrations running on PolyAPI before any agent was asked to reproduce similar results at scale. With that foundation in place, the teams were ready to build the framework itself.

The Governed Multi-Agent Migration Framework

One of the defining characteristics of the project was the deliberate decision not to pursue fully autonomous migration. Instead, GHA and PolyAPI designed a governed multi-agent migration framework that balanced AI-driven acceleration with human architectural governance. The objective was not to automate everything, but to maximise delivery speed while maintaining enterprise-grade architectural oversight, governance, and implementation quality.



The teams designed the framework around a core insight: specialisation produces better outcomes than generalisation. By splitting responsibilities across two purpose-built agents with tightly defined roles, each agent could be deeply optimised for its specific task, with a structured manifest serving as the contract between them. Human validation checkpoints were built into the workflow at precisely the points where architectural judgement mattered most.

The framework was built around four interconnected components. An agent-agnostic design ensured the entire framework could operate across any AI tooling environment, free from dependency on any single model or vendor. A library of deterministic Skills provided consistent, repeatable execution patterns that reduced the inherently nondeterministic nature of large language models. Two specialised agents — the Origin Language Decoder (OLD) and the Native Enterprise Workflow Writer (NEWW) — handled the interpretation and implementation work respectively, with a structured manifest

serving as the contract between them. And a human-in-the-loop governance model ensured that architectural decision-making remained firmly in human hands.

Agent-Agnostic Design

One of the most important architectural decisions made early in the project was ensuring the framework had no dependency on any specific AI model or coding assistant. Tightly coupling the migration framework to a single vendor would have introduced exactly the kind of lock-in the project was designed to mitigate.

Instead, the migration intelligence lived in the framework itself: the reusable patterns, structured manifests, Skills, and governance workflows that could operate consistently across any AI coding environment. This proved its value in practice, as GHA and PolyAPI were running different tooling ecosystems throughout the project. GHA primarily leveraged Microsoft Copilot while PolyAPI engineers worked across OpenAI Codex and Claude Code. The framework performed consistently across all of them, with no need for adaptation or additional vendor dependencies.

Because the abstraction sits at the framework level rather than the model level, the methodology is portable, flexible, and compatible with however the AI landscape continues to develop.

Deterministic Skills

A core challenge in any agentic framework is the inherently nondeterministic nature of large language models. To address this, the teams built a library of "Skills" — structured, reusable execution patterns that give agents deterministic guardrails for the tasks where consistency matters most. Skills combined contextual guidance, embedded scripts, validation logic, and platform-specific implementation standards into a single reusable unit.

This proved especially valuable for tasks such as:

- Credential redaction
- Source code compression
- MuleSoft structure interpretation
- Figma design interpretation
- PolyAPI primitive usage
- Retry patterns
- API orchestration standards

One of the most powerful Skills developed was the ability to interpret architectural designs directly from visual inputs. The workflow used a lightweight but effective approach: the OLD Agent produced its output, a human architect used that output to produce a design in Figma, and a PNG export was passed directly to the NEWW Agent. The NEWW Agent interpreted the visual — colour codings, workflow notations, architecture relationships — asked clarifying questions, presented an implementation plan for human approval, and only then proceeded to implement.

Through the use of Skills the team achieved a meaningful and measurable improvement in consistency and reliability across migration cycles.

Origin Language Decoder (OLD) Agent

The OLD Agent was purpose-built to do what had previously required a seasoned MuleSoft specialist: read and interpret existing integration flows accurately enough to migrate them. Rather than passing raw source code directly to the NEWW Agent, the OLD Agent produced a clean, structured migration manifest that could serve as a reliable foundation for the NEWW Agent to build off of.

Its responsibilities included:

- Analysing MuleSoft source code
- Understanding orchestration logic and workflows
- Removing noise and irrelevant artefacts
- Redacting sensitive information
- Compressing and simplifying source structures
- Producing structured migration manifests

Native Enterprise Workflow Writer (NEWW) Agent

Working from the structured manifest produced by the OLD Agent, the NEWW Agent was responsible for translating that blueprint into production-ready, native code written in Python, that would run on PolyAPI. Crucially, rather than making architectural assumptions independently, it was designed to surface ambiguities and validate key decisions with human architects before proceeding.

Its responsibilities included:

- Reading migration manifests
- Interpret Figma designs using standard PolyAPI conventions
- Understanding PolyAPI primitives and standards
- Generating implementation plans

- Producing PolyAPI-native implementations
- Validating assumptions with human architects

Human Governance

A central design principle of the framework was the deliberate inclusion of humans in the decision-making process. Rather than allowing agents to make unchecked architectural assumptions, the framework explicitly required:

- Human-provided target architecture design
- Assumption validation
- Clarification of ambiguities
- Architectural review before implementation
- Human intervention on design deviations and critical architectural decisions

This balance proved essential in achieving both acceleration and enterprise-grade reliability. The agents were deliberately instructed never to make independent architectural assumptions — every consequential decision was surfaced to a human architect for validation before implementation proceeded. AI handled the high-speed work: analysis, orchestration understanding, implementation planning, and code generation. Human architects retained ownership of architectural intent, governance, and the critical design decisions that could not be delegated.

Andrea described the experience of collaborating with the agents this way:

"The process was similar to training a junior engineering team member and then asking them to take over the development once their results align with your standards. You invest in training them to produce results at a set standard, and to ask questions when there is lack of clarity or assumptions." — Andrea Angiolini, SVP Systems, GHA

The approach significantly reduced hallucination risk, deepened the team's understanding of the target platform, and produced a migration workflow that became more deterministic and more reliable with every integration completed.

Project Outcomes

The results validated the investment in the framework extremely quickly. The most immediate outcome was delivery speed. Implementation cycles that would previously have taken weeks were completed in days or hours, with velocity increasing as the framework matured and the agents built up context across successive migrations.

"It understood the way our other applications were built and applied that understanding without us having to spell it out. It asked the right questions and took our directions exactly as intended." — Andrea Angiolini, SVP Systems, GHA

Across the migration programme, GHA observed the following improvements:

- Significantly faster implementation cycles (flows completed in ~1/3 original estimate timelines)
- Improved consistency across integrations
- Easier onboarding to the new platform
- Lower implementation friction throughout

The framework enabled the delivery of production-grade integrations written in native Python, running on PolyAPI at a pace that neither architecture team had considered achievable at the outset. The agents also demonstrated a capacity that went beyond raw speed. Rather than treating each integration in isolation, they inferred architectural patterns from existing implementations and applied them consistently to subsequent ones without explicit instruction.

The discipline invested in building the framework correctly from the outset proved to be more than a migration accelerator. The same principles that governed the migration — specialised agents, human oversight at critical decision points, deterministic Skills, and a governance model designed to scale — are equally applicable to building net new capabilities.

Looking Ahead

Building this framework together shifted how both teams think about the future of integration engineering. The most valuable application of AI in this domain is not full autonomy, but a carefully designed collaboration between specialised agents and human architects who retain ownership of the decisions that matter most.

The emerging model for modern engineering teams is one of agentic organisations — networks of specialised AI workers coordinated through structured workflows, with humans operating at the governance layer.

In this Governed Multi-Agent Model, the engineering organisation evolves into a team of:

- AI orchestration designers
- Agent supervisors
- Architectural reviewers
- Governance operators
- Business-context translators

The engineers who will thrive in this next era are those who can design agent frameworks, govern multi-agent workflows, and bring the business context and architectural judgement to produce quality outputs at a speed and cost previously thought to be impossible.

"This project fundamentally changed how I think about what is achievable. The speed at which we were able to deliver production-grade integrations, and the quality of what the framework produced, has opened up possibilities that I would not have considered realistic before we started. We're very pleased with the partnership with Poly and the results it's produced, and are excited for what comes next after the migration." — Andrea Angiolini, SVP Systems, GHA

If your organisation is weighing the cost and complexity of moving off a legacy integration platform, or exploring how a governed multi-agent framework could accelerate your own modernisation programme, we would welcome the conversation.

Reach out directly or at hello@polyapi.io.

Thank you for reading.

Luis Weir Chief Architect, PolyAPI

Members of the GHA and PolyAPI teams enjoying time together after the kick-off workshop.