AI VOID FIELD GUIDE

Technical Case Studies

In-depth technical case studies exploring real-world architecture decisions, implementation challenges, and engineering solutions from production systems.

AI VOID

2026-07-29 15:30 IST

Contents

CHAPTER 01

Agentic Application Modernization at Scale with Strands and AWS Transform Custom: Technical Case Study

Executive Summary

The escalating complexity of legacy IT estates, characterized by diverse technologies, undocumented code, and significant technical debt, poses a critical challenge for organizations striving for agility and innovation. This case study delves into how agentic applications, specifically leveraging Strands agents orchestrated by Amazon Bedrock AgentCore and powered by AWS Transform Custom, are revolutionizing large-scale application modernization on AWS. This approach demonstrates a significant leap in efficiency, enabling modernization efforts up to 5 times faster and reducing maintenance costs by as much as 70%. By automating high-effort, repeatable tasks and providing deep insights into legacy systems, agentic AI empowers organizations to tackle hundreds of applications in parallel, freeing skilled professionals to focus on strategic architectural and optimization work.

The Modernization Imperative: Addressing Technical Debt

Many enterprises grapple with monolithic applications and disparate IT landscapes comprising various technologies, languages, and frameworks. These legacy systems often suffer from undocumented code, a lack of comprehensive test coverage, and accumulating technical debt. Such challenges hinder innovation, increase operational costs, and impede the adoption of modern cloud-native architectures. The drivers for modernization typically include:

- **Reducing Operational Costs:** Eliminating expensive legacy licenses and maintenance.
- **Improving Agility:** Enabling faster feature development and deployment.
- **Enhancing Security and Compliance:** Migrating to more secure, managed cloud environments.

- **Unlocking Innovation:** Leveraging modern cloud services and AI/ML capabilities.

Traditional modernization efforts are often manual, time-consuming, and resource-intensive, making large-scale transformations prohibitively complex. This is where agentic AI offers a transformative solution.

Agentic Architecture for Large-Scale Transformation

The core of this modernization strategy lies in an agentic orchestration architecture that combines specialized AI agents with a robust transformation engine. Figure 1 illustrates the interplay between Strands agents, Amazon Bedrock AgentCore, and AWS Transform Custom.

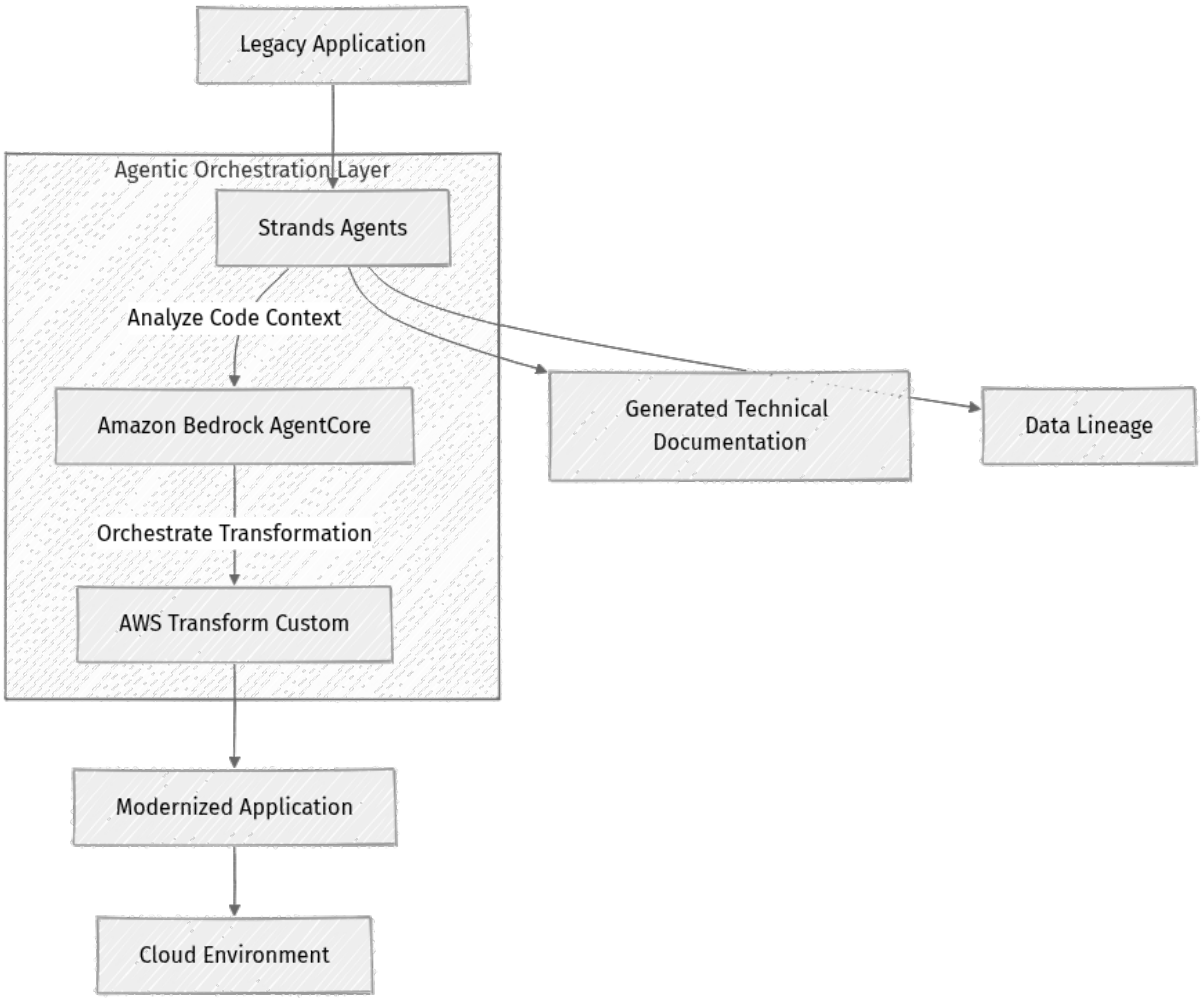

Figure 1: AWS Transform Custom Agentic Orchestration Architecture using Strands Agents and Amazon Bedrock AgentCore

Key Architectural Components:

1. **Strands Agents:** These specialized agents initiate the modernization process by performing in-depth analysis of legacy application code. They are designed to understand complex code structures, identify dependencies, and extract business logic. A critical capability of Strands agents is their ability to generate:
    - **Automated Data Dictionaries:** Providing visibility into legacy data structures and their business meanings.
    - **Data Lineage Maps:** Illustrating how data flows through the application, crucial for understanding impact and ensuring data integrity during migration.
    - **Test Coverage Generation:** Addressing the common lack of test coverage in legacy systems.
2. **Amazon Bedrock AgentCore:** This service acts as the intelligent orchestrator, managing the lifecycle and interactions of the Strands agents. It takes the analysis from Strands, understands the modernization goals, and directs AWS Transform Custom to execute the necessary transformations. AgentCore ensures that the entire workflow—from assessment to deployment—is customized, iterative, and flexible.
3. **AWS Transform Custom:** This is the transformation engine that performs the actual code changes. It offers:
    - **Pre-built Transformations:** For common patterns across languages like Java, Node.js, and Python upgrades, as well as specific modernization paths for Windows, mainframe, and VMware applications.
    - **Custom Capabilities:** Allowing organizations to define and execute transformations for company-specific programming languages, frameworks, APIs, and architectural patterns. This ensures the solution is adaptable to virtually any legacy system.

This multi-agent, orchestrated approach allows for consistent, repeatable, and high-quality transformations at a scale previously unattainable.

Overcoming Modernization Challenges

The agentic architecture effectively addresses several persistent challenges in large-scale modernization:

- **Complexity of Diverse Legacy Estates:** By offering custom transformation capabilities, AWS Transform Custom can handle any code, API, framework, runtime, architecture, or language, including proprietary ones. Strands agents provide the initial deep analysis required to understand these diverse systems.
- **Lack of Test Coverage:** A common pain point in legacy systems, this is mitigated by Strands agents generating test coverage, which is vital for validating transformations.
- **Manual Effort and Repetitive Tasks:** The automation inherent in agentic AI drastically reduces the manual effort involved in code analysis, planning, documentation, and transformation, allowing teams to deliver larger, more complex projects with ease.
- **Visibility into Legacy Data Structures:** The generation of Data Lineage maps and Automated Data Dictionaries by Strands agents provides unprecedented insight into the often-undocumented intricacies of legacy data.

Quantifiable Benefits and Impact

The adoption of agentic modernization with Strands and AWS Transform Custom has yielded significant and quantifiable benefits for organizations:

- **Accelerated Modernization:** Enterprises like IDEMIA and Thomson Reuters have modernized .NET and Windows applications up to 4x faster. Overall, the agentic AI–powered automation can accelerate modernization tasks by up to 5x across analysis, planning, documentation, and transformation.
- **Cost Reduction:** By streamlining the modernization process and enabling migration away from legacy infrastructure, customers can eliminate up to 70% of their maintenance and licensing costs.

- **Scalability:** The automated nature of the solution allows for the parallel transformation of hundreds of applications, making large-scale modernization initiatives feasible and efficient. Bridgestone, for instance, modernized its mainframe in seven months.
- **Improved Quality and Consistency:** Automated transformations ensure a consistent application of modernization patterns, reducing human error and improving the overall quality of the modernized codebase.
- **Strategic Resource Allocation:** By offloading repetitive, time-intensive tasks to agents, skilled migration experts can dedicate their efforts to higher-value architectural design, optimization, and strategic decision-making.

Key Performance Indicators Achieved

Metric	Impact	Source
Modernization Speed	Up to 5x faster	AWS Transform, About Amazon News
Cost Reduction	Up to 70% in maintenance & licensing costs	About Amazon News
Scalability	Hundreds of applications in parallel	AWS Transform
VMware Migration Planning	10x acceleration for wave planning	AWS Transform
Mainframe Modernization Timeline	7 months for Bridgestone	AWS Transform

---

Real-World Application Examples

Several prominent organizations are actively leveraging AWS Transform with its agentic capabilities to eliminate technical debt:

- **IDEMIA:** Modernized .NET applications 4x faster.
- **Thomson Reuters:** Modernized Windows applications 4x faster.
- **Bridgestone:** Modernized its mainframe in seven months.
- **CSL:** Accelerated VMware migration wave planning by 10x.
- **ADP:** Reimagined compliance infrastructure.

- **Air Canada, Experian, QAD, Teamfront, Verisk:** All using AWS Transform to address their tech debt.

Lessons Learned and Future Outlook

The implementation of agentic applications for modernization highlights several critical insights:

- **Automation is Key to Scale:** For truly large-scale modernization involving hundreds of applications, intelligent automation driven by agents is indispensable. It transforms what was once a multi-year, manual undertaking into a streamlined, accelerated process.
- **Human Expertise Remains Crucial:** While agentic AI handles the repetitive heavy lifting, it does not eliminate the need for skilled migration expertise. Professionals are required for higher-value architectural decisions, complex problem-solving, and ensuring the business context is correctly applied to transformations. The tool empowers experts by allowing them to focus on strategic work.
- **Applicability Beyond Code Transformation:** The architectural pattern demonstrated—using intelligent agents for deep analysis and automation—is highly adaptable to other large-scale code analysis and automation workflows, such as security vulnerability remediation, compliance checks, or architectural refactoring.

The future of application modernization is increasingly agentic, with AI taking on more complex analysis and transformation tasks, allowing businesses to adapt faster, innovate more freely, and significantly reduce the burden of legacy systems.

References

- Agentic application modernization at scale with Strands and Amazon Transform custom | AWS DevOps & Developer Productivity Blog
- Smash tech debt and get AI ready – AWS Transform – AWS
- AWS Transform and Agentic Modernisation: The End of Technical Debt | Devoteam
- New agentic capabilities in AWS Transform enable rapid modernization of any code or application
- AWS re:Invent 2025 - Modernize Legacy Applications with Agentic AI (DEV333) - YouTube

---

Transparency Note: This case study was generated by an AI Expert, synthesizing information from provided AWS and partner documentation, blog posts, and public announcements regarding AWS Transform, Strands agents, and Amazon Bedrock AgentCore. All technical details, architectural patterns, and quantifiable results are based on the provided search context.