outsystems

Introduction to Agentic Systems Engineering

Build, orchestrate, govern, and evolve agentic systems for the enterprise

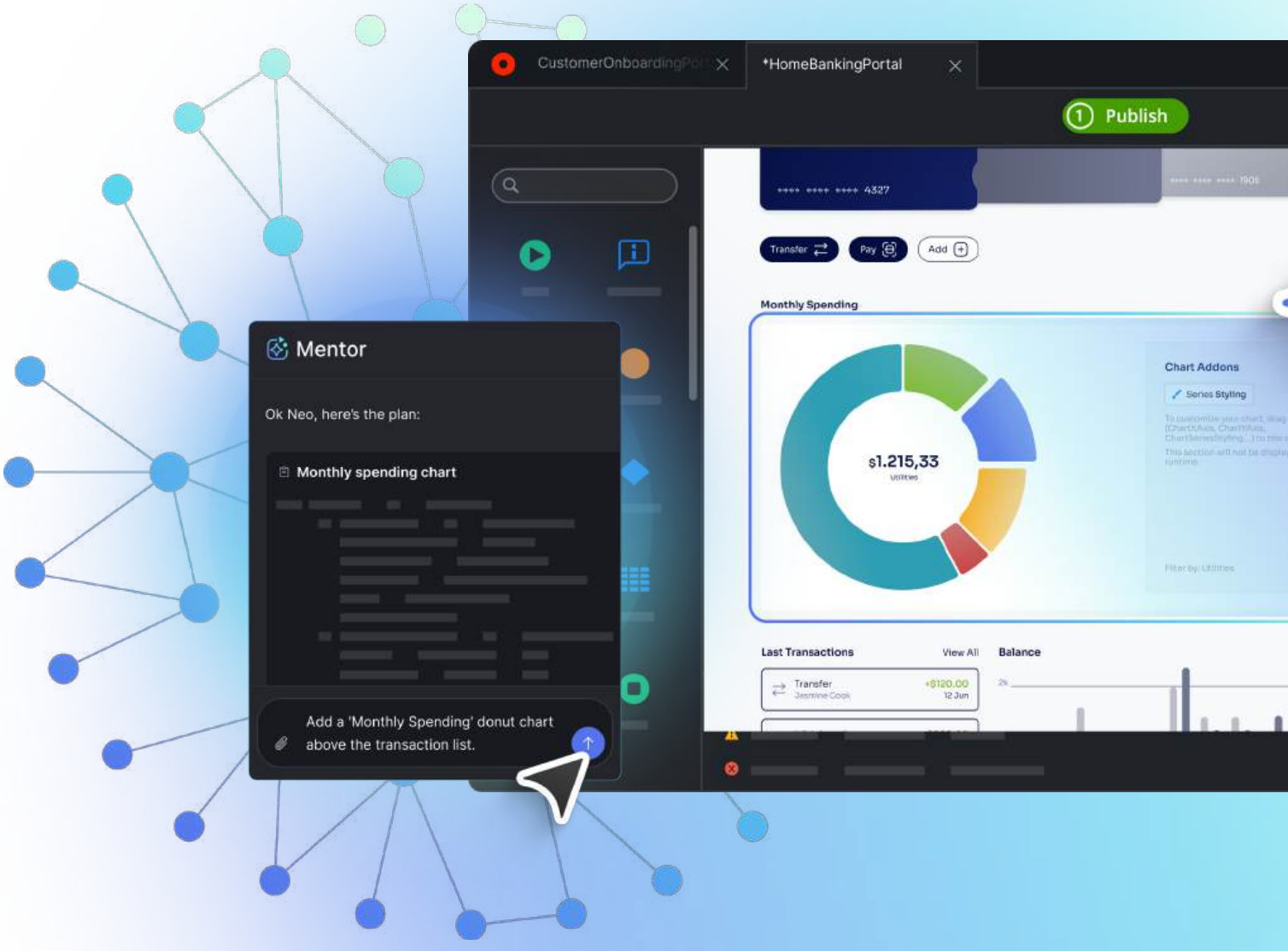


TABLE OF CONTENTS

Executive Summary

In a 2026 survey, 94% of IT leaders said they were concerned about AI sprawl. Nearly 60% of their developers are already running three or more different AI tools simultaneously. Ecosystems are fragmenting in real time because these tools are disconnected. Yet, only 12% have done something concrete about it.[1]

The easier it gets to generate code, apps, and agents, the harder it gets to control their downstream impact. So while AI is accelerating pilot projects and experimentation, few projects actually make it into enterprise production. This is the "enterprise gap," and it's preventing many enterprise IT teams from fulfilling their organizations' potential with AI and agentic technology.

The leaders who have hit the brakes on ungoverned AI projects aren't wrong. It's their job to worry about unpredictable performance, architecture and data drift, and inconsistent interaction with other systems. But waiting for the next AI model or compromising their unique needs for an off-the-shelf solution isn't the answer, either.

The race is on to become an agentic enterprise. The ones that get there first will have the power to scale their business light years faster than their competition. What's missing is a framework for governing the full lifecycle of AI and agents for true, reliable enterprise performance.

Agentic Systems Engineering (ASE) is a new, AI-native framework that helps organizations build, manage, and orchestrate agentic systems for the enterprise on an open and trusted platform.

ASE was designed specifically for AI-first development, where the old ways of building applications are giving way to conversational app generation and agents that participate at every phase of the process. ASE addresses the technical complexity and fragmented architectures of modern enterprises that are often difficult to reason over. By developing the rich context and guardrails agents require, it ensures the delivery of reliable, secure, and compliant systems.

This primer explains how ASE works, why its underlying technology is different, and how teams can begin using it today.

97% of IT leaders said they were concerned about AI sprawl

36% of respondents have a centralized approach to AI governance

12% Concrete action taken

Woodson Martin,
CEO, OutSystems

"AI is creating more change, across more tools and surfaces, than ever before. Enterprises still need that change to be governed, secure, and production-ready. Agentic Systems Engineering is our answer to that challenge."

[1] The State of AI Development 2026: Navigating the Shift to Agentic AI, OutSystems, April 2026.

Chapter 1
Why enterprise AI keeps stalling in production

Agentic AI offers an unprecedented opportunity to dramatically improve and innovate the systems that enterprises use to run their businesses. But deploying agents built in a vacuum without a control layer can stall AI initiatives, damage trust, and slow growth.

Ungoverned MVPs are no match for enterprise systems and their dependencies

AI development tools can't navigate enterprise complexity alone. There are countless data connections and data sources, along with integrations and dependencies to consider and manage. A retail bank building a customer onboarding agent has to reach into core banking, KYC services, credit bureau APIs, document verification, and the CRM. A patient intake agent must bridge electronic health records, scheduling, insurance eligibility, and consent management. For an inventory forecasting agent to be effective, it should connect to the ERP, warehouse management system, supplier portals, and demand planning software.

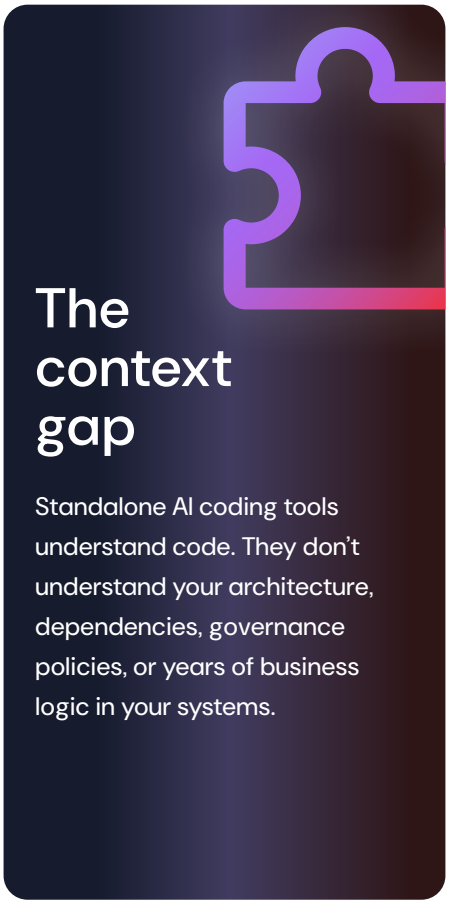
It's a rare enterprise that can ditch its existing systems for a bright, brand-new system developed from scratch. Instead, IT teams and their business stakeholders want to build AI and agents to improve or enhance existing applications and workflows. Yet, these systems and the data they run on are a primary reason AI projects stall.

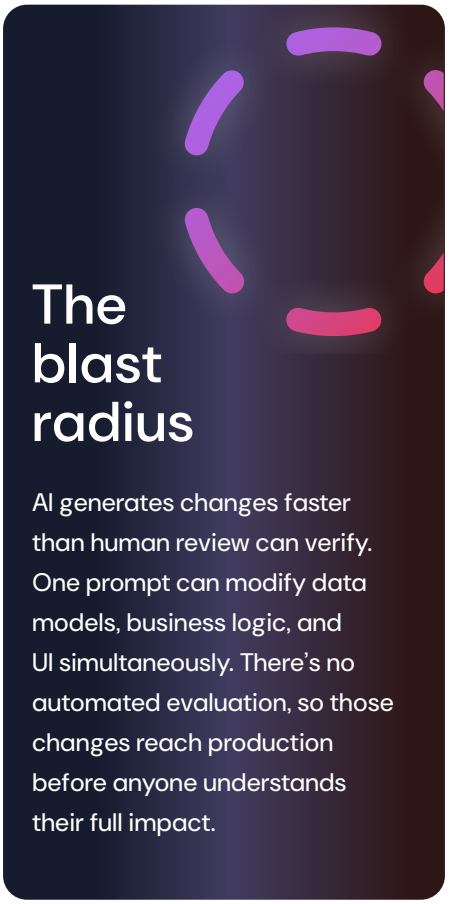
The operational reality of deploying specialized agents across key industries

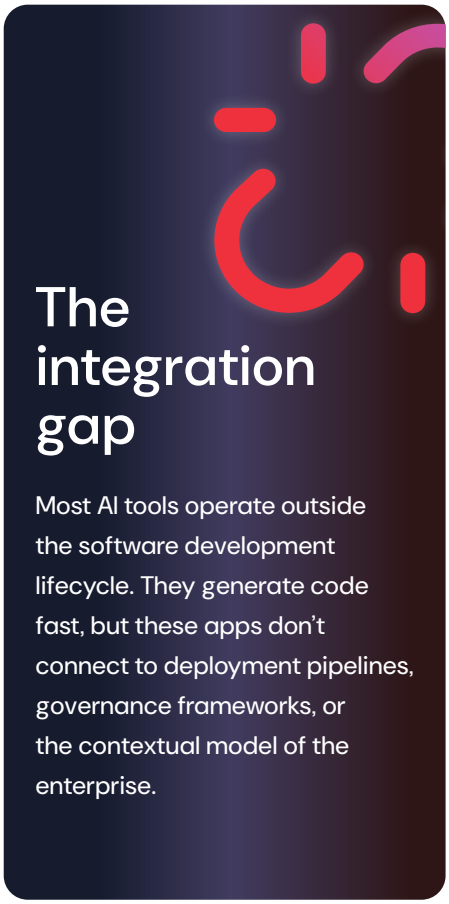
INDUSTRY SECTOR	AUTONOMOUS AGENT TYPE	REQUIRED SYSTEM INTEGRATIONS & TOUCHPOINTS
Retail Banking	Customer Onboarding Agent	Core Banking Systems, KYC Compliance Engines, Credit Bureau APIs, Document Verification Tools, Enterprise CRM
Healthcare	Patient Intake Agent	Electronic Health Records (EHR), Central Scheduling Systems, Insurance Eligibility Verification, Consent Management
Supply Chain	Inventory Forecasting Agent	Enterprise Resource Planning (ERP), Warehouse Management Systems (WMS), Supplier Portals, Demand Planning Engines

Friction points created by AI tools hinder progress

Consider a procurement system built on Lotus Notes over 20 years ago that relies on manual approvals, exposes vendor data, and has no audit trails. The people who built the system are gone, and its rules are buried in code. The design team uses Figma, and the backend is C#. The CEO wants it rebuilt as an agentic system. In this scenario, there are three friction points.





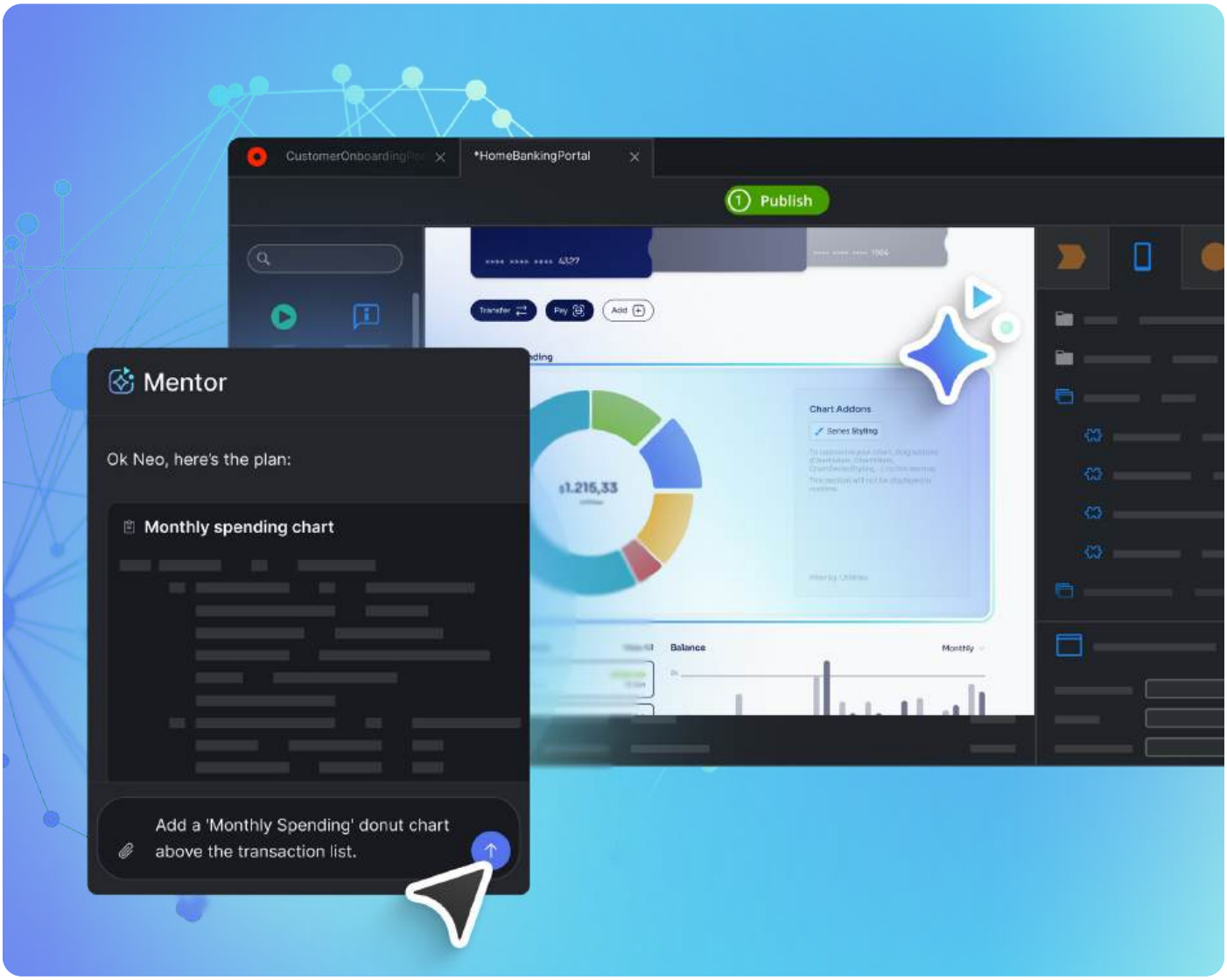


Agentic systems can remove this friction and drive enterprises forward. They consist of applications and autonomous agents that understand dynamic enterprise context to function reliably and autonomously in a business ecosystem. Building them requires a fresh approach to enterprise development.

Chapter 2
Introducing Agentic Systems Engineering

Agentic Systems Engineering (ASE) is a new plaform-based approach to enterprise AI development from OutSystems. ASE extends a company's unique enterprise architecture to AI-generated agents and applications and provides the context, governance, and openness needed to deploy them in complex environments.

ASE is AI-native, using a conversational experience and agents to automate development and deploy-ment. Developers can build custom AI apps and agents, modernize legacy processes, and ship agentic solutions that wrap and extend core systems and data.



What makes Agentic Systems Engineering different from AI coding assistants

AI coding assistants generate code. ASE makes businesses agentic at every level. Instead of starting from a blank slate, ASE begins with an understanding of the dynamic context of your enterprise environment, including nonfunctional requirements, dependencies, identity and permissions. Teams can build working agentic applications in minutes in a natural language conversation, and iterate them with the confidence that they will perform according to your organization's standards.

ASE is open, so it can support a range of AI and agentic coding tools like Claude Code and Kiro. Developers can choose their preferred LLM for each application, including customized models for specific industries, and even swap out models in future versions of an application as business needs evolve.

Using ASE, enterprise teams can deliver a repeatable governance and safety loop where every change passes through automated evaluations and policy validations to ensure the delivery of reliable, secure, and compliant systems. It also removes the risk of managing infrastructure and fragmented LLM stacks.

Modernizing Lotus Notes legacy in minutes with Agentic Systems Engineering

A development team can use ASE to approach the Lotus Notes modernization project described in Chapter 1. The original developers are long gone, leaving behind manual approvals, putting vendor data at risk, and no audit trails. **But in 10 minutes, using ASE in the OutSystems platform, it's possible for two engineers to:**

- ✓ Migrate the Lotus Notes procurement system into a modern enterprise environment.
- ✓ Add an intake agent that extracts and structures data from purchase orders automatically.
- ✓ Activate agent guardrails to block prompt injection and prevent sensitive data exposure.
- ✓ Enable agent decisions to become training data for future accuracy.

The CEO gets the agentic system they requested in just 10 minutes, thanks to a **context layer** with dynamic tooling, **conversational app generation**, and **an open ecosystem**. Let's explore these components of ASE.

Chapter 3

The context layer: The missing link

Most AI tools produce code or text files. When teams use multiple tools, it's not possible to share context. Each is operating on a partial picture. It's like a jigsaw puzzle when you first open the box.

OutSystems generates a structured model. It uses OutSystems Model Language (OML) to capture business intent, data relationships, UI structure, access controls, and integration dependencies in a single, machine-readable representation. OML is the core file format for designing, editing, and sharing modules in the OutSystems platform IDE.

Every component or application built on the OutSystems platform shares this model-driven foundation. This reusability powers full lifecycle governance, deterministic outcomes, and performance. An example is an OutSystems extension built by a C# developer that becomes immediately available to every application and agent in the ecosystem. For ASE to deliver on its promises, a context layer was necessary, one that includes the dynamic tooling that agents need to be highly productive over complex systems. And so the Enterprise Context Graph was born.

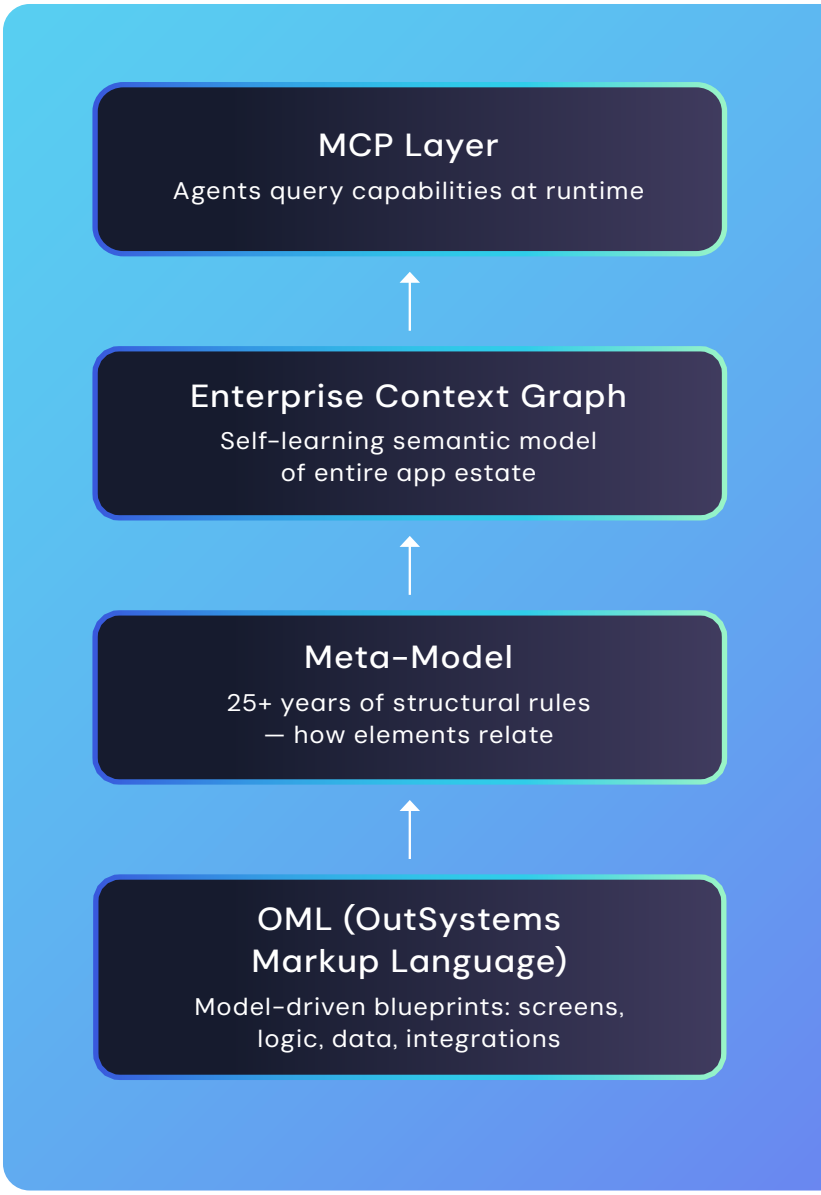
Forrester's Total Economic Impact study **found a 363% ROI for OutSystems customers, with payback in under six months.** The OutSystems model-driven foundation is what makes that compounding return possible. Teams are not rebuilding foundational architecture with every new application or agent.

The Enterprise Context Graph: The foundation for Agentic Systems Engineering

The Enterprise Context Graph is an evolution of OML. An active map of a customer's data, business logic, and dependencies provided by OML delivers the high-fidelity grounding that AI agents need to run safely and productively in complex systems. The graph extends the OutSystems platform's unique contextual architecture with the dynamic and external tooling that your teams need to build highly productive agents that run in complex systems.

The Enterprise Context Graph interconnects user interfaces and design systems, workflows and business logic, integrations and data structures, and agentic behaviors and skills. It can reason over dependencies, enforce governance policies, and understand how a change to a data model cascades across 20 connected applications. Everything is brought into a unified platform, and each action is versioned, auditable, and governed.

This significantly speeds up delivery while keeping teams in full control. Backend developers, designers, and external AI tools can all participate in building and evolving these systems.

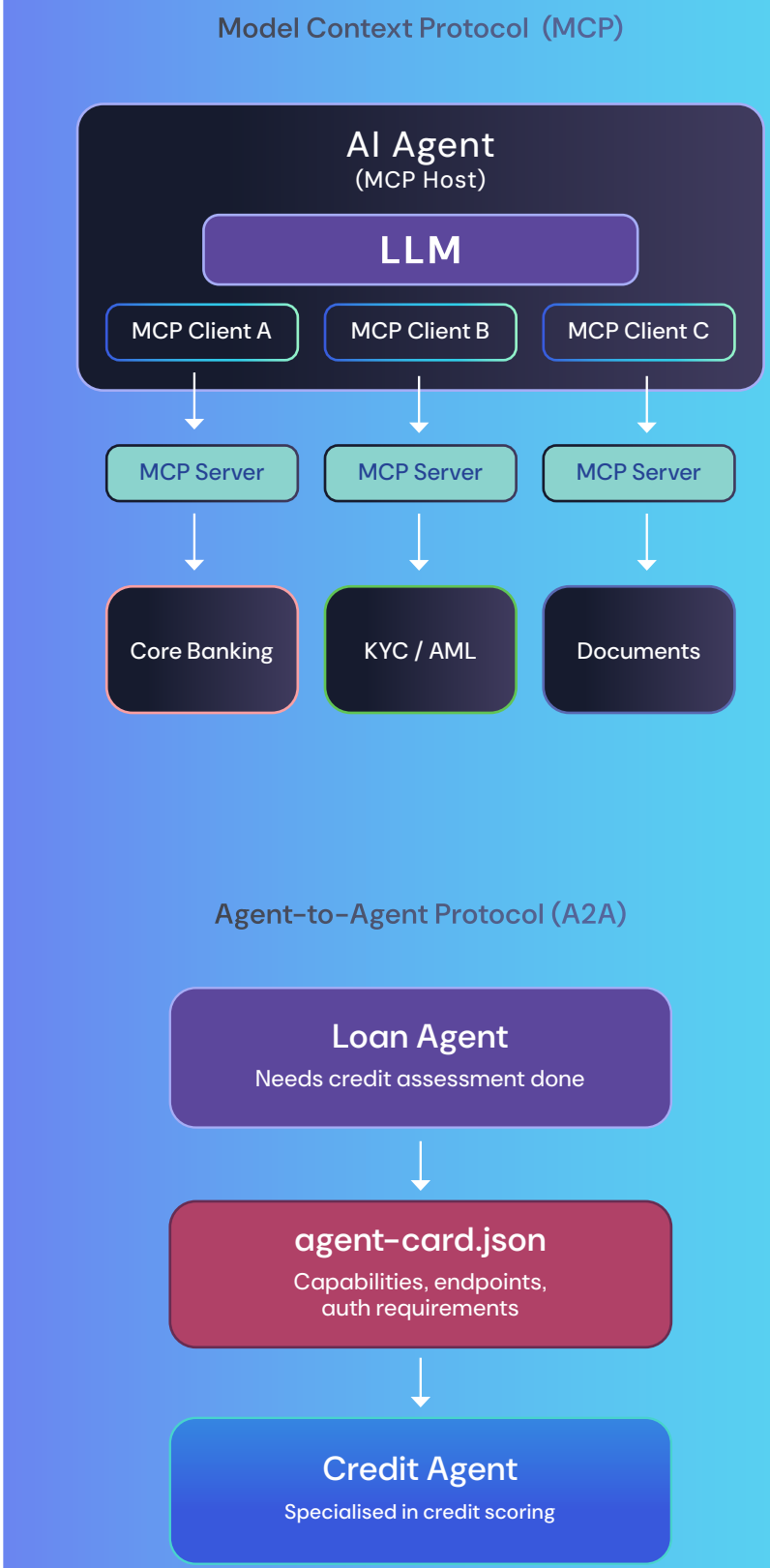


API, A2A, and MCP services are key to the graph

The Enterprise Context Graph is formalized with a services API, which opens it to the agentic development tools used in your organization. Add ASE's agent-to-agent (A2A) and MCP services, and external agentic tools such as Claude Code, Cursor, and Codex can read the model, contribute code, deploy extensions, and apply design systems.

In fact, because of the Enterprise Context Graph, a design system applied through Claude Code can propagate across your entire portfolio. With A2A, agents can talk to app generation tools. Not only does every contribution from every tool flow into this graph, but each change is validated against it before deployment.

All of this is done within the permissions granted to that tool and that team. Role-based access controls govern what each tool and each team can see and change within the graph. For example, a design team given access through Claude Code can apply themes and deploy extensions without changing data models or business logic.



How context affects agentic development

Changing an application changes the model. As a result, the change is immediately understood in the context of the full application graph, including its dependencies, downstream effects, and compliance with governance policies. Automated evaluations run against the model before any changes deploy. Agents can't introduce an architectural inconsistency that the model does not surface immediately.

The extension of the OML by the Enterprise Context Graph also creates a self-learning loop. When a human or an agent makes a decision in a workflow, it becomes a decision trace. Those traces accumulate and codify the business rules embedded in tribal knowledge. The result is context the agent can use to improve future decision-making. For example, an intake agent that processes 2,300 order submissions and has learned that routing rules can apply a level one escalation path if the amount is under $50,000.

> "Without the Enterprise Context Graph, agents are just guessing."

Gonçalo Borrêga

Senior Vice President of Product Management, AI & AppDev, OutSystems

Chapter 4

Build, iterate, and orchestrate with conversational app generation and automation

There are a lot of vibe coding tools, but so many of them produce prototypes. A key aspect of the ASE framework is empowering teams to teams generate full-stack, production-grade applications using natural language right where they work. For example, Mentor, an in-IDE AI app generation tool native to OutSystems, turns building and iterating over complex enterprise systems into a conversation. Because it has real-time understanding of enterprise architecture, it can generate and evolve applications and agents that can be easily integrated into business systems. It also automates additional parts of the software development lifecycle, such as iteration, monitoring, validating quality, and security checks.

With Mentor, developers and architects provide requirements in natural language right in the OutSystems Developer Cloud IDE. Mentor accesses existing data models, application logic, UI patterns, style guides, and governance policies. It asks clarifying questions and proposes a plan. This plan is a blueprint that covers the full data model, business logic, UI screens, and role permissions. Developers review, iterate or change if necessary, approve, and ship the application.

ACM

"Working with Mentor is like having a programmer by your side, a true partner that always helps you break out of technical walls whenever you hit one."

—Erik Looi, CDO,
AllianceCorp Manufacturing

HEINEKEN

"We uploaded the exact business requirements for an event engagement app to Mentor, and it automatically generated almost everything needed—from the Data Model and server-side logic to the essential screens. In seconds, we had about 80% of the MVP version built."

—Praveen Kumar Natarajan, OutSystems architect, HEINEKEN

Mentor respects your guardrails, role-based access control, and existing dependencies because it's reading all of that from the Enterprise Context Graph in real time. It brings the speed of AI and the precision of visual development together in a trusted, agentic-first environment. With Mentor, developers move from writing every line of code to orchestrating a team of agents.

Automated orchestration drives production-ready agentic AI

Production-ready agentic AI requires orchestration. ASE capabilities like Mentor coordinate multiple autonomous, specialized AI agents so developers and engineers don't have to. ASE automates the processes of ensuring that agents complete complex, multi-step tasks while also managing communication, role allocation, and workflow. There are also automated controls that track agent reasoning paths, response quality, and operational costs in real-time; controls also define technical and ethical constraints to ensure agents operate within company policy.

Automated lifecycle management elevates development

Mentor and ASE automate iteration, monitoring, validating quality, and security checks to support rapid, high-quality evolution and maintenance of applications and agents. Instead of spending time on the repetitive and time-consuming tasks of lifecycle management, teams can trust Mentor to handle them.

Generated apps and agents are already integrated into an application ecosystem before a developer reviews the first line of output. This integral component of the ASE framework elevates developers into agentic systems engineers as they move from writing every line of code to orchestrating a team of agents. They have what they need to architect, build, and evolve the agentic systems of tomorrow. And the developer stays in control throughout.

KENT STATE UNIVERSITY

"Mentor is a major time-saver. I can ask it to implement my request and it does it with high accuracy. I never want to go back to not using Mentor—two thumbs up!"

—Wayne Fincher, Developer, Kent State University

Vopak

"OutSystems is the enterprise platform that provides the governance and trust layer we need to have on top of agentic AI-led development. It allows us to benefit from the speed of AI in a controlled and secure way, ultimately making it possible to use AI without spending hours reviewing the code behind its output."

—Cristiano Marques, Director of Software Development, Vopak

Chapter 5
Open by design: A real-world scenario

Developers and architects shouldn't have to use a different AI tool when they're proficient in one already. That's why ASE is open by design with the Enterprise Context Graph APIs, A2A, and MCP, enabling designers and developers to use tools like Claude Code, Cursor, and Codex. The procurement modernization story we introduced in Chapter 1 can show how this works.

Design teams using Claude Code

The design team for the procurement app stays in Figma and Claude Code. Through the Context Graph Services API, Claude Code has access to a permissions-scoped subset of the enterprise context so it can apply themes, deploy extensions, and update design systems across all applications in the portfolio. The designer used ASE to apply a new purple design system across the entire ProcureOS application using a single natural language prompt without ever opening the OutSystems Developer Cloud IDE.

Design teams using Codex

The C# backend team used Codex to wrap an existing EDI parser library and deploy it as an OutSystems extension. Through the extension framework, that library becomes available to every application, agent, and workflow in the ecosystem. The backend developer did not leave their IDE, and the extension landed in a governed, versioned platform.

What governance looks like across all paths

Every contribution from every tool flows into the same code quality dashboard, lifecycle management framework, and observability layer. ASE surfaces architecture recommendations, security risks, and maintainability scores. The governed path from development to production is the same regardless of which tool generated the code.

The benefits of being open

With ASE, internal and external user interfaces, workflows and business logic, integrations and data, and agentic behaviors and skills are all connected and governed. Your design team doesn't need to learn ODC Studio. They use their own tools, and their design system is applied across all apps in the portfolio instantly. Your backend team doesn't need to be OutSystems developers. They work in their own IDEs, in C#, with their preferred coding agent. Their output flows into the platform, managed, governed, and versioned while respecting your architecture, security policies, and standards.

30,000+
Agentic apps developed on OutSystems

Supported by an ecosystem of over **3,700+** certified developers.

Chapter 6
How to get started with Agentic Systems Engineering

Agentic Systems Engineering meets teams where they are and doesn't require dismantling existing infrastructure or retraining teams. Whether you are a systems architect, a developer, or an IT leader, you can get started with ASE today.

Developer or architect checklist

- ☐ Sign up for the OutSystems Personal Edition.
- ☐ Enable OutSystems Developer Cloud (ODC), which is where agentic features live. If you are an existing OutSystems 11 customer, you can request an ODC instance through your customer portal.
- ☐ Download ODC Studio, the primary IDE where you will design and publish your agentic apps.
- ☐ Use Agent Workbench in ODC Studio to create and manage agents.
- ☐ Ground your agents by connecting them to your enterprise data through the Enterprise Context Graph, which allows agents to understand your specific business logic and dependencies.
- ☐ Conversationally generate UI screens and data models your agents will interact with.
- ☐ Once published, your agent becomes a reusable service action. You can call it from any web or mobile app by adding it as a dependency.

IT Leader checklist

- ☐ Map your current development surfaces, including teams, tools, and the governance gaps between them.
- ☐ Identify the 12% opportunity, which is the part of your portfolio that would benefit most from centralized lifecycle governance.
- ☐ Evaluate your enterprise context graph before adding new development surfaces.
- ☐ Begin with one modernization candidate, use it to establish the governance model, then extend.

Build your agentic future with OutSystems

- ✓ **Request a personalized demo** to see ASE applied to a use case specific to your industry and stack.
- ✓ Take a closer look at **Agentic Systems Engineering in action.**
- ✓ Sign up for a **free personal environment.**