Project-X: Architectural Evolution - Old vs. New

Alberto Espinosa
KSquare Group

July 18, 2025

This document outlines the key changes, adaptations, and equivalences between the original Project-X architecture and the newly proposed agent-based architecture leveraging Ollama and LangGraph, updated as of 06:57 AM CST. The platform supports multiple use cases, including fraud detection, supply chain optimisation, and customer analytics.

1 Original Architecture Overview

The original Project-X architecture focused on distinct components for data ingestion, predictive modelling, business rule application, and a centralised decision engine, all feeding into a user interface. It laid a solid foundation for data-driven decision-making.

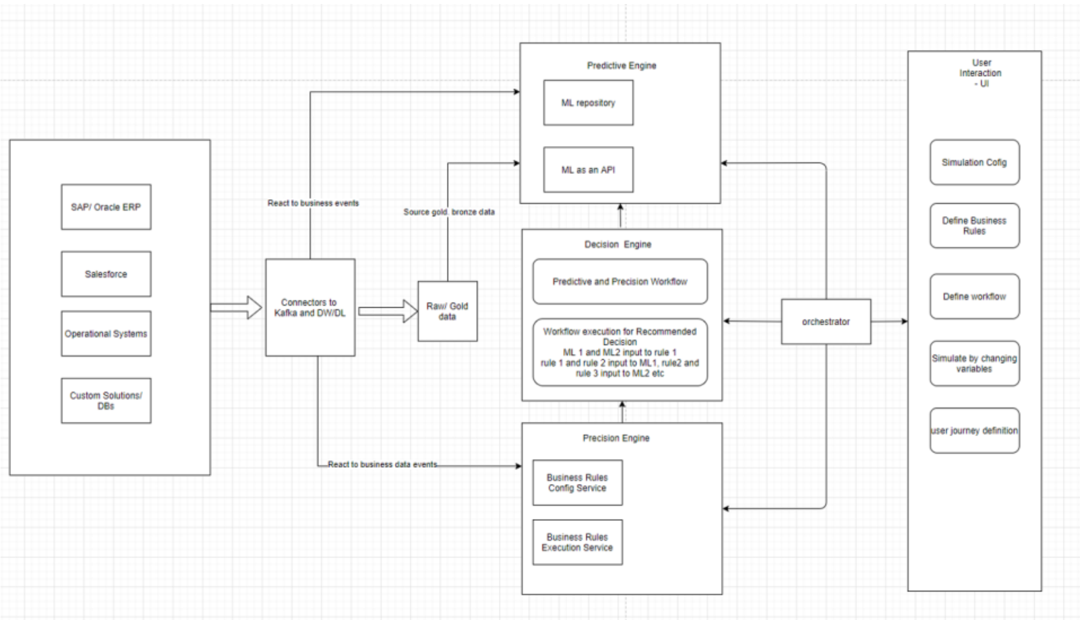

Figure 1: Original Project-X Architecture Diagram

2 Key Driving Principles for Adaptation

The adaptation to an agent-based architecture is driven by the desire for:

- **Enhanced Intelligence & Adaptability:** Utilising LLMs for more nuanced understanding and dynamic behaviour.
- **Increased Automation:** Automating complex tasks like code generation and intelligent data handling.
- **Transparency & Explainability:** Providing clear reasoning for decisions and actions.
- **Scalability & Efficiency:** Leveraging LLMs for reasoning while offloading heavy computation.

- **Empowered User Interaction:** Enabling more intuitive configuration and understanding for business users.

3 Architectural Changes, Adaptations, and Equivalences

The table below details how each module from the original architecture has evolved in the agent-based design.

Original Module	New Agent-Based Adaptation	Changes/Equivalences
Data Sources	**Data Sources**	No fundamental change in external data origin.
SAPI Oracle ERP, Salesforce, Operational Systems, Custom Solutions/DBs	*SAPI Oracle ERP, Salesforce, Operational Systems, Custom Solutions/DBs*	These remain the foundational data providers (integration planned post-MVP with synthetic data focus initially).
Connectors to Kafka and DIVIDL	**Connectors (to Kafka and DIVIDL) & Data Ingestion Agents (Stream Processor Agent)**	**Adaptation:** Connectors remain, but are now intelligently augmented by Data Ingestion Agents (post-MVP integration). **Validation:** Hybrid validation (rules + LLM-driven semantics) ensures data integrity across use cases. Tools: SciPy, pandas. Success metric: 98% accuracy in routing and metadata generation. **Performance:** Stress-tested with high-throughput streams (e.g., 1000 events/second). Tools: JMeter, Locust. Metrics: Latency ¡ 100ms, throughput ¿ 1000 events/second. **Integration:** Post-MVP real-time integration with Kafka and DIVIDL, using exponential backoff retries, dead-letter queues, and batch processing. Metrics: Ingestion latency ¡ 50ms, error rate ¡ 1%.
React to business events, Raw Gold Data, Source Gold Bronze Data	*Data Ingestion Agents (Stream Processor Agent)*	**New Capabilities:** LLM-driven intelligent parsing, hybrid validation, dynamic routing, rich metadata generation, summarisation (initially with synthetic data).
	Raw Gold Data, Source Gold Bronze Data	Data output layers remain, enhanced by agent processing post-MVP.
N/A (New)	**Synthetic Data Generation Agent (Code Generator)**	**New Addition:** An entirely new capability for programmatically generating synthetic data for MVP across multiple use cases. **Validation:** Statistical tests (e.g., Kolmogorov-Smirnov) ensure data aligns with use case distributions. Tools: SciPy, pandas. Success metric: p-value ¿ 0.05. Code validated via pytest (95% pass rate). **Performance:** Tested for code generation efficiency under load.
No equivalent in old architecture	*Agent generates code for data creation using libraries like Faker, SciPy.*	Addresses the need for robust test/training data, especially when real data is scarce or sensitive.

Continued on next page

Table 1 – continued from previous page

Original Module	New Agent-Based Adaptation	Changes/Equivalences
Predictive Engine	**Predictive Engine (AutoML Orchestration Agent)**	**Adaptation:** Enhances the model creation process with AutoML intelligence. **Validation:** Generated pipelines evaluated using cross-validation and use case-specific metrics (e.g., F1 ¿ 0.85). Tools: PyCaret, scikit-learn. **Performance:** Tested for model inference speed under high load.
ML repository, ML as an API	*ML Repository, ML as an API, AutoML Orchestration Agent*	**Changes:** The AutoML Agent (powered by Ollama) can analyse synthetic datasets and tasks to generate PyCaret code, automating model building, tuning, and evaluation. The ML Repository and ML as an API remain the storage and serving layers.
Precision Engine	**Precision Engine (Business Rule Management Agent)**	**Adaptation:** Streamlines rule definition and management. **Validation:** LLM-translated rules cross-checked against test suites for diverse use cases. Tools: Custom validation scripts. Success metric: 98% accuracy. **Performance:** Tested for rule execution efficiency.
Business Rules Config Service, Business Rules Execution Service	*Business Rules Config Service, Business Rules Execution Service, Business Rule Management Agent*	**Changes:** The Business Rule Management Agent (Ollama-powered) allows natural language input for rules, translating them to structured formats. The core services are retained as tools for the agent.
Decision Engine	**Decision Engine (LangGraph Core)**	**Fundamental Transformation:** Moves from a fixed workflow to a dynamic, intelligent agent graph. **Validation:** Explanations evaluated for clarity (90% rated clear). Conflicts resolved without human intervention in 95% of cases. **Performance:** Stress-tested with complex workflows (e.g., 100 concurrent decisions). Tools: Locust, Prometheus. Metrics: Latency ¡ 100ms, throughput ¿ 100 decisions/second.
Predictive and Precision Workflow, Workflow execution for Recommended Decision	*LangGraph-Powered Orchestration, Decision Orchestration Supervisor Agent, Sub-Agents (Prediction Request, Rule Application, Data Query, Human-in-the-Loop, Conflict Resolution Agents)*	**Changes:** LangGraph defines stateful workflows with agent collaboration. Decisions use adaptive reasoning with synthetic data initially.
Orchestrator	**Orchestrator (API Gateway)**	**Adaptation:** Becomes primarily an API gateway to the new agentic backend. **Performance:** Tested for API routing efficiency under high request volumes.
Links backend processing with UI	*Routes UI requests to LangGraph workflows, translates agent responses.*	**Changes:** Focuses on communication mediation, with workflow logic in LangGraph.

Continued on next page

Table 1 – continued from previous page

Original Module	New Agent-Based Adaptation	Changes/Equivalences
User Interaction - UI	**User Interface (UI)**	**Enhancement:** Becomes a more powerful interaction point. **Performance:** Tested for responsiveness under high user interaction loads.
Simulation Config, Define Business Rules, Define workflow, Simulate by changing variables, user journey definition	*Simulation Config, Define Business Rules, Define Workflow (Visual LangGraph Editor), Simulate by Changing Variables, User Journey Definition, Agent Interaction Interface*	**New Capabilities:** Basic Agent Interaction Interface for MVP, with Visual LangGraph Editor planned post-MVP.

4 Roadmap Overview

The development of the new agent-based architecture follows a structured roadmap, detailed in the separate document `project_x_roadmap.tex`. Key phases include MVP Development (July–August 2025), Validation (August–September 2025), Performance Testing (September–October 2025), Integration (October–November 2025), and Post-MVP Enhancements (December 2025 onward), with dependencies ensuring a phased rollout.

5 Conclusion

The shift to an agent-based architecture with Ollama and LangGraph transforms Project-X into a more intelligent, adaptive platform, starting with synthetic data for MVP. It leverages LLMs for reasoning and code generation, integrating existing components as tools, with plans for real-data integration post-MVP for greater flexibility and scalability across multiple use cases.