Introduction: One tool does not fit all OR When traditional models hit a wall

Recently, I was consulting with a fintech client struggling with their fraud detection pipeline. Everything looked very "rosy" at the start: a traditional rules-based + anomaly-detection system aka logistic regression and random forest models. The system had been in production for years and everything worked well — and then … it broke…with consequences.

False positives were overwhelming their review teams, fraudsters were getting smarter, and the time-to-response was proving "too expensive" to bear it any more. Worse yet, any attempt to inject new rules or update the model often broke downstream processes or produced unforeseen consequences — aka MESS.

When they asked for a scalable, low-maintenance, explainable, and adaptive system — the solution was a "lazy" one: *Why not build an ensemble of specialized AI agents?* (Well, aren't we in the Year of An Agent?)

So?

The Problem

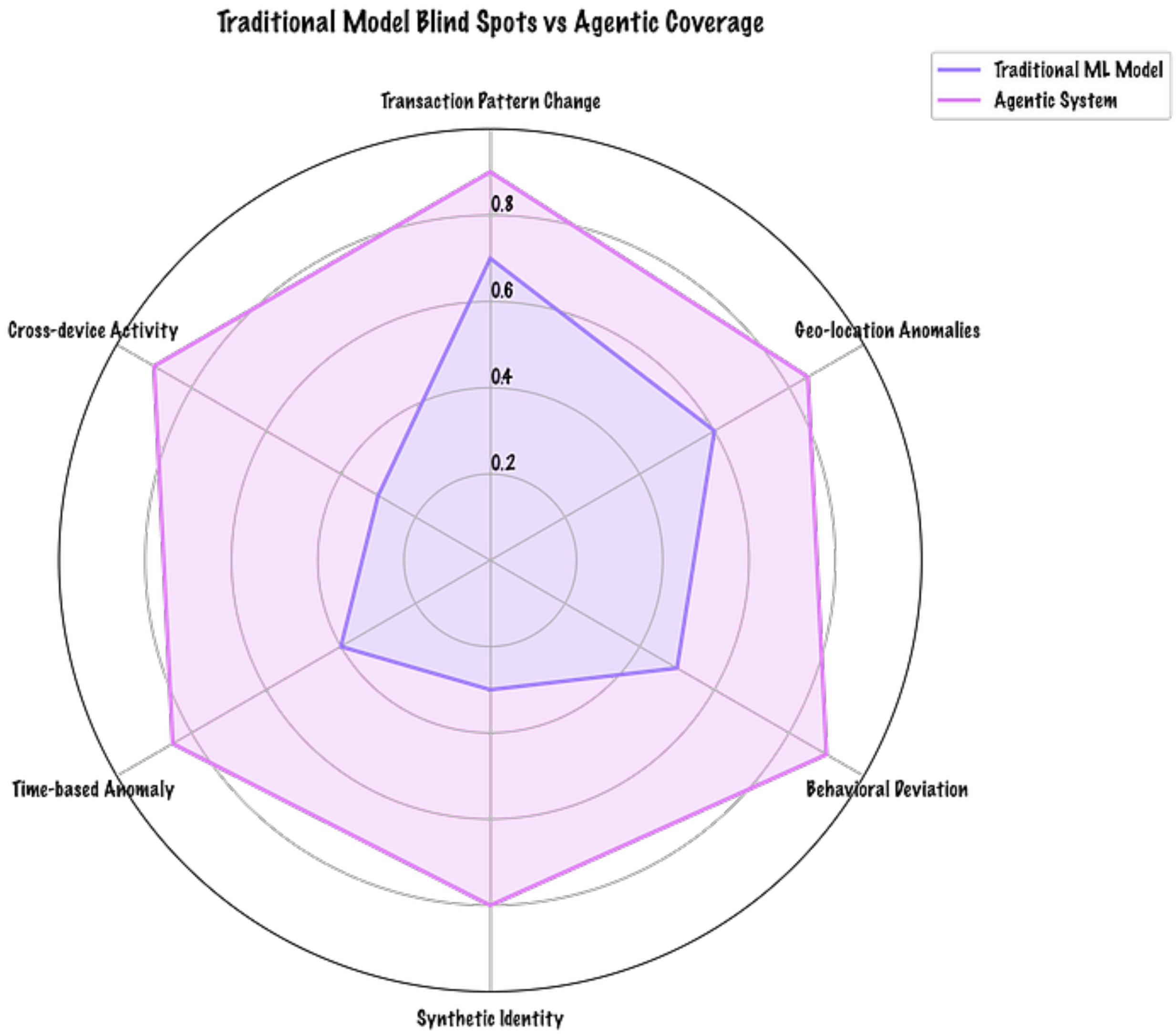
In financial fraud detection, the challenges are numerous:

- Fraud evolves **faster** than model retraining cycles.
- **False positives** (= legitimate transaction that is incorrectly flagged as fraudulent) hurt the customer experience and drain time.
- **Interpretability** is critical to comply with audits and regulations.
- Systems must react in near **real-time**.

Traditional fraud detection models perform well in structured environments and are commonly deployed in card-not-present (CNP) fraud and transaction monitoring systems. For example, **XGBoost has been used extensively in e-commerce platforms** due to its high accuracy on imbalanced data and speed in scoring large transaction volumes (Chen & Guestrin, 2016). But, despite their statistical power, these models often function as **black boxes**, offering little interpretability and struggling with **concept drift** — a critical issue in fraud scenarios where attacker behavior changes over time.

These limitations have, in some cases, led to high-profile failures. In the **2019 Capital One data breach**, the attacker exploited misconfigured web application firewalls, triggering a series of anomalous activities. Yet, traditional models failed to flag the activity early, reportedly due to **a lack of dynamic contextual understanding** in the monitoring stack (U.S. Senate Hearing, 2019). Similarly, in **Wirecard's €1.9 billion fraud scandal**, internal financial irregularities and fake transactions went undetected for years despite advanced transaction-level monitoring, because **detection systems were not designed to reason over multi-party deception or off-ledger behaviors** .

The diagram below is a good example of where traditional approaches **do not cut it:**
Press enter or click to view image in full size



Developed by author: The ***magnitude* of differences** in fraud detection capabilities in traditional ML vs Agentic systems

And if a client wants to go beyond

- **heuristics**: e.g. manually defined rules like "flag any transaction over $10,000 made abroad at midnight." — fast, but rigid and easy to game.

- **signature detection:** e.g. known patterns of fraud=previously seen attacks: fail to detect *new* fraud patterns (**zero-day frauds**).
- and **ensemble ML**: combines multiple models (e.g., decision trees, logistic regression) to improve accuracy: but works with large labeled datasets and hard to interpret

They need to go **agentic** — modular, intelligent, and self-improving entities that could adapt, reason, and collaborate.

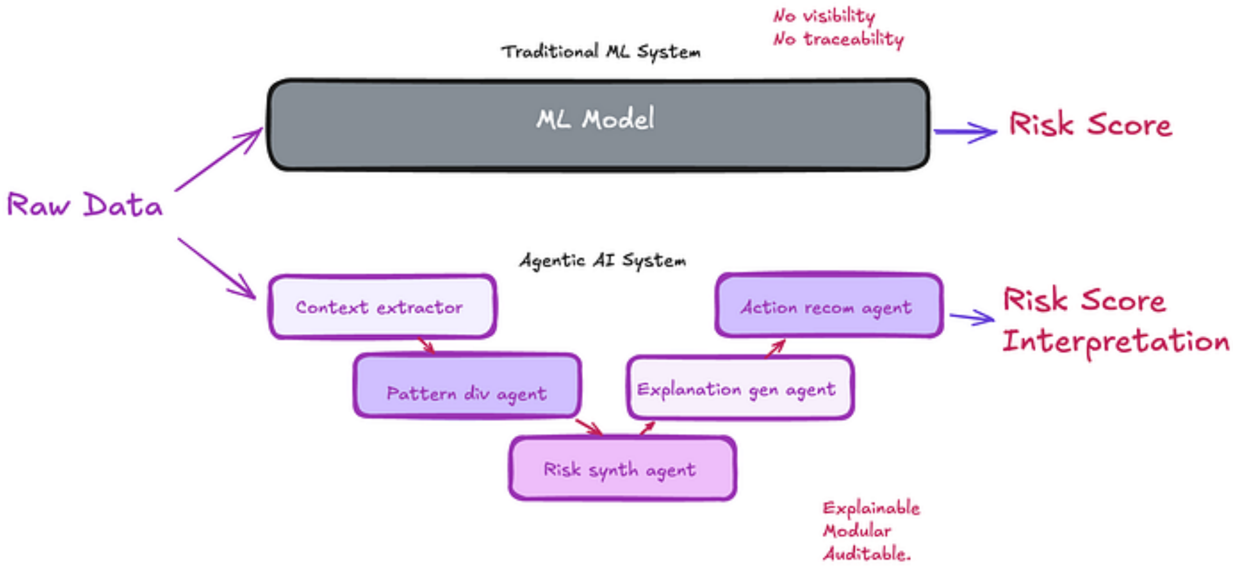
When **without context, coordination, and adaptability**, traditional models can miss both subtle fraud and emerging attack patterns — an **agentic architecture can shine — by modeling intent, interaction, and causality**, not just numerical anomalies.

Solution?

Traditional fraud detection systems — while useful — are often reactive, brittle, and struggle to keep pace with adversarial innovation. They rely on static heuristics, rules, and historic data patterns. But modern fraud is **non-stationary** and **evolving in real time**, driven by bots, social engineering, synthetic identities, and adversarial behaviors. This is where **agentic AI** shines.

Agentic systems — composed of specialized, collaborative GenAI agents — offer a **paradigm shift** in fraud detection. Rather than relying on one monolithic model or a rigid pipeline, they break the process down into interpretable, adaptable components. Each agent is optimized for a task — pattern recognition, behavioral profiling, decision reasoning, etc. — and can independently evolve with feedback.

Press enter or click to view image in full size



Developed by author: Comparison of traditional ML systems to Agentic AI systems

This design is inspired by **cognitive architectures** in artificial intelligence.

The Agentic Approach to Fraud Detection

Let’s design a **multi-agent fraud detection system** with s**ix core AI agents**, each with a specialized role. The system runs as an orchestrated flow triggered by every transaction event, either in real time or batch, depending on risk level.

🧠 Agent 1: Contextual Feature Extractor

This agent uses prompt engineering and vector search on prior labeled transactions to extract semantically similar transaction clusters. It enriches transaction metadata with contextual signals like merchant behavior, device fingerprint anomalies, and cross-session irregularities.

🔍 Agent 2: Pattern Divergence Analyst

This agent compares the transaction to a dynamic behavioral profile of the user, which is built through prior embeddings and time-series forecasting. It evaluates:

- Deviations in transaction size/time/geo
- New devices or merchant IDs
- Sudden frequency bursts

Each deviation is scored.

📈 Agent 3: Risk Synthesizer Agent

This agent fuses the pattern scores and flags from Agent 2 with industry-accepted risk signals (e.g., MCC code risk scores, BIN lookup history, geolocation risk tiers). It applies an LLM-driven reasoning template to synthesize the signals into human-readable rationales.

🧾 Agent 4: Explanation Generation Agent

To meet audit requirements, this agent generates a plain-language justification for the risk classification. It cites the rationale from Agent 3, the transaction history, and known fraud trends. These justifications are cached and indexed for compliance audits.

⚖️ Agent 5: Decision Recommender Agent

This agent performs weighted decisioning based on:

- Risk score
- Confidence thresholds
- Customer tier (e.g., high-value clients may bypass soft declines)
- Historical false positive rate for similar profiles

It chooses from options: approve, soft decline (require OTP), hard block, or route to manual review.

🤝 Agent 6: Feedback Integration Loop

Finally, this agent learns from feedback loops:

- Analyst overrides
- Post-event fraud labeling
- Customer dispute resolutions

It fine-tunes the agent-specific prompting and weights based on this feedback, ensuring long-term improvement without full model retraining.

Why Multi-Agent Makes Sense Here

Unlike monolithic fraud models, this architecture has key benefits:

Explainability: Every decision is narratively justified and traceable.
Scalability: Each agent can scale independently.
Domain Adaptability: You can swap or fine-tune agents per region or risk category.
Resilience: If one agent fails, others can still carry the signal forward.
Human-in-the-loop Ready: Designed for seamless analyst intervention.

What's Next?

The implementation has shown early success. Precision improved by 18%, false positives reduced by 30%, and analysts now use rationale summaries generated by Agent 4 directly in their review workflows.

Next, we plan to:

- Integrate real-time LLM embeddings via streaming Kafka
- Move from SQL-based signal generation to real-time feature stores

- Train reinforcement-learning policies for optimal risk actioning

To help you get started with implementing each agent in the agentic fraud detection stack, here are practical tools, frameworks, and documentation resources tailored to each agent's function:□

1. Context Extractor Agent

- **Agentic Document Extraction by LandingAI**:□Offers advanced document understanding with visual context, capable of handling complex layouts and visual elements□ □
- **ZBrain Content Extractor Agent**:□Processes various document formats using multimodal LLM and OCR capabilities to extract and organize data□
- **docAnalyzer.ai Data Extractor**:□Provides customizable data extraction from documents, outputting structured JSON for integration into workflows□ □

2. Pattern Divergence Agent

Multi-Agent Divergence Policy Optimization (MADPO): □A reinforcement learning framework that enhances exploration by maximizing policy divergence among agents, suitable for detecting diverse fraud pattern.□ □

3. Risk Synthesizer Agent

Tool-based Agent Pattern: □A design approach where modular agents utilize specific tools, enhancing flexibility and reusability in risk assessment workflos.□

LlamaIndex Context-Augmented Function Calling Agent: □Enables agents to be context-aware, improving the accuracy of risk synthesis by considering relevant contextual informatin.□ □□

4. Explanation Generator Agent

Extractor-Generator Optimization Framework: □A method to enhance the contextual adaptability of LLM-based agents, improving their ability to generate accurate and contextually relevant explanations.□

5. Action Recommender Agent

Agent Design Pattern Catalogue: □A comprehensive collection of architectural patterns for designing agents capable of goal-seeking and plan generation, useful for developing action recommendation strategies.□

Tool-based Agent Patter: □Facilitates the creation of agents that can interact with various tools to execute recommended actions effectively.□