Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

Autonomous Inventory Management using Agentic AI and SAP: An End-to-End Supply Chain Framework

Chandra Babu Gundlapalli
Independent researcher ,
Department of Information Technology, NJ, USA.

Abstract **- The old inventory playbook - quarterly safety stock reviews, static reorder points, MRP batch runs that fire whether anything has changed or not—was built for a slower world. This paper describes a working framework that drops agentic AI into SAP S/4HANA and lets it run the show across procurement, warehousing, and outbound distribution. Not in the usual way, where a model produces a forecast and then sits around waiting for someone to act on it. Here, the agents actually do things: place purchase orders, shuffle warehouse slots, kick off stock transfers between facilities. They pull live signals from IoT gear, market feeds, and SAP transaction logs, crunch them through reinforcement learning policies, and execute—all inside SAP's native transaction framework so the audit trail stays clean. We tested this on a simulated mid-size consumer electronics company with three DCs and 1,200 SKUs. Carrying costs came down 23.4 percent. Stockouts dropped 41 percent. The order-to-delivery window shrank by almost a fifth. Those are simulation numbers, not production numbers, and the paper is upfront about the difference. It also spends real time on the governance question—who is accountable when an agent makes a bad call, how you keep RL policies from going sideways in novel situations, and what kind of organizational groundwork has to happen before any of this goes live. The short version: agentic AI is a genuine leap past dashboards and recommendation engines, but deploying it responsibly is at least as hard as building it.**

Keywords **- agentic AI; autonomous inventory management; SAP S/4HANA; multi-agent systems; supply chain optimization; reinforcement learning; demand forecasting; IoT-enabled warehousing**

I. INTRODUCTION

Talk to anyone running a supply chain right now and they will tell you the same thing, usually with some exasperation: the job has gotten harder. Not incrementally harder. Fundamentally harder. Customers treat next-day delivery as a given. A product that had a three-year lifecycle in 2015 might have eighteen months today, if that. And disruptions—the kind that used to show up once a decade—now roll in every few quarters. Port closures during COVID, chip shortages that rippled through every industry imaginable, freight rate swings tied to geopolitics nobody saw coming. Planning inventory around a quarterly review cycle feels, increasingly, like bringing a calendar to a knife fight.

SAP S/4HANA is the transactional backbone for a huge share of these companies. It does what ERP does well: recording every movement, enforcing business rules, making sure the books balance. But the planning intelligence inside most SAP setups? Still largely deterministic. You set a reorder point. You calculate an economic order quantity. MRP fires on Tuesday morning whether demand shifted or not. In stable markets, that works fine. In volatile ones—which is most markets now—it produces either bloated warehouses or empty shelves. Sometimes both simultaneously, in different SKUs [1].

AI has been circling this problem for years. Demand forecasting with gradient-boosted trees. Anomaly detection that flags weird patterns. Exception reports that light up dashboards. All useful, none sufficient. Because every one of these tools stops at the same point: it generates an insight and waits. Waits for a planner to log in, look at the number, decide whether to trust it, figure out what action to take, and then go execute that action in the ERP. That handoff—insight to human to action—eats days. Sometimes a week or more. A demand forecast sitting unactioned for three days is essentially three-day-old news [2]. In fast-moving categories, that is an eternity.

Agentic AI breaks that loop. The word "agentic" is doing real work in that phrase—it means the AI does not just predict, it acts. Software agents with defined goals that perceive their environment, reason about options, and then go do something: place a PO, adjust buffer stock, reroute a shipment to a different DC. No human in the middle for routine decisions [3].

None of this is science fiction, by the way. Multi-agent systems have been around since the nineties in academic AI. What changed is raw capability. Deep RL and large language models gave agents the ability to handle messy, ambiguous, never-seen-before situations in ways that old rule-based agents never could [15]. Connect that kind of intelligence to SAP's transactional engine and you get a system that moves at machine speed without breaking the business logic the ERP is there to protect.

Three things pushed us to write this paper. One: SAP's own roadmap is screaming in this direction. Joule already has

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

40-plus embedded agents and 2,400 skills across S/4HANA, Ariba, and IBP [4]. Two: the academic literature on multi-agent RL for inventory has crossed from theoretical curiosity to practical possibility [5]. Three—and this is the actual gap—nobody has published a framework that wires these pieces into a coherent end-to-end system and also addresses the governance problem, which in our experience is where most real-world deployments actually stall.

The paper proceeds as follows. Section II covers prior work. Section III introduces our multi-agent design. Section IV describes the technical architecture and SAP integration. Section V walks through a simulated case study. Sections VI and VII discuss results, limitations, ethics, and open research questions. Section VIII concludes.

II. LITERATURE REVIEW

A. AI-Driven Inventory Management

The story of ML in inventory management is, at this point, pretty well worn. It started with forecasting—ARIMA, exponential smoothing, the usual suspects—and then moved to tree-based methods once people realized demand curves rarely behave linearly. The current frontier is deep learning: LSTMs, and more recently temporal fusion transformers borrowed from NLP research. On benchmark datasets for retail and consumer goods, these architectures beat classical statistical methods by margins in the 15 to 30 percent range, depending on granularity and how you measure error [6]. Impressive, but not the whole story.

Here is the problem nobody in the forecasting community likes to talk about much: a perfect forecast, all by itself, does not tell you what to do. It tells you what demand will probably look like. Somebody—or something—still has to convert that into an order quantity, pick a supplier, decide on timing. That conversion involves trade-offs under uncertainty, and it is where reinforcement learning really shines. RL skips the prediction-then-decide two-step and optimizes the decision directly. Liu et al. published results in 2024 showing that a multi-agent deep RL approach—specifically using something called heterogeneous-agent proximal policy optimization, or HAPPO—beat single-agent RL and classical heuristics on multi-echelon problems. It also dampened the bullwhip effect, which anyone who has managed a multi-tier supply chain knows is the gift that keeps on giving [7].

Then there is the LLM angle, which is newer and frankly a bit surprising. A 2024 paper introduced InvAgent—a system that uses large language models' zero-shot reasoning to run multi-agent inventory decisions without any environment-specific training whatsoever [8]. It is early days for that approach, and whether it holds up at enterprise scale is genuinely unknown. But the idea that you might not need months of historical data to get an agent up and running is tantalizing, especially for companies entering new product categories. On the analytics foundations side, Govindan and colleagues [17] did the unglamorous but essential work of mapping out how big data methods connect to logistics problems.

B. Multi-Agent Systems in Supply Chains

Using agents to model supply chains is an idea with gray hair. It dates to the early 2000s, when distributed AI researchers noticed what should have been obvious: a supply chain is a bunch of independent actors, each with its own goals, its own data, its own constraints, all interdependent. Jennings and his group built early platforms with negotiation protocols and contract-net mechanisms [9]. Clever stuff. Also, to be blunt, ahead of what the hardware and algorithms of that era could deliver. Most of it stayed academic.

Deep RL resurrected the whole field. The key innovation—centralized training, decentralized execution, often abbreviated CTDE—lets you train agents together in a shared simulator so they learn to cooperate, then deploy them independently so each one only needs its own local observations at runtime. That is a natural fit for supply chains. Different business units or trading partners will never share real-time proprietary data with each other, but they can absolutely benefit from policies that were jointly optimized in a training environment [10]. Oroojlooyjadid and Snyder [15] wrote what amounts to the definitive survey of this space and made an observation I keep coming back to: the mathematical structure of supply chain coordination—shared rewards, private observations—is almost suspiciously well matched to cooperative multi-agent RL.

More recent papers have gotten grittier. Raj et al. [11] modeled agents fighting over limited warehouse space and transport capacity, which is much closer to what actually happens than the tidy single-product models that dominated earlier work. Panetto and co-authors [19] looked at the broader challenge of getting cyber-physical manufacturing systems to work across the gap between what is happening on the shop floor and what the planning systems think is happening. That gap, in my experience, is where a lot of supply chain pain actually lives.

C. SAP's Evolving AI Capabilities

SAP has not been watching from the sidelines. The Joule copilot went from chatbot to full agent platform in about eighteen months. By early 2026 the count stood at 40-plus purpose-built agents and over 2,400 skills spread across S/4HANA, Ariba, SuccessFactors, and IBP [4]. These are not toy demos. They call APIs, write back to SAP data stores, trigger workflow chains—real transactional work, with human checkpoints for anything high-stakes.

On the inventory side specifically, SAP has Demand-Driven Replenishment using RL for buffer tuning, and IBP's ML features claim 50 to 70 percent faster planning cycles with 20 to 30 percent accuracy gains [12]. SAP AI Core handles custom model training and deployment. All real capabilities. All useful.

And yet. Each of these is its own module solving its own narrow problem. Nobody—not SAP, not any third party we could find—has demonstrated them working in concert as a unified autonomous system spanning the full chain. The orchestration piece is missing. Ivanov [16] flagged exactly this gap, arguing the field needs integrative frameworks, not

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

additional point solutions stacked next to each other. That is the hole this paper tries to fill.

III. PROPOSED MULTI-AGENT FRAMEWORK

Five agents, each owns a lane, each talks to the others through a shared message bus, and a central Orchestrator keeps them from stepping on each other. The analogy that kept coming up during design was a well-run trading desk: each trader specializes, but there is a risk manager watching overall exposure.

A. Demand Sensing Agent

Most SAP shops run demand planning in batch. Weekly, maybe monthly, feed the output into MRP, move on. This agent throws that cadence out the window. It ingests POS data, promo calendars, weather, social sentiment, macro indicators—all streaming. Under the hood it runs an ensemble: gradient-boosted trees for the tabular features, a temporal fusion transformer for time-dependent patterns. It does not pick one model and ride it. Instead, it reweights the ensemble continuously based on which model has been nailing recent actuals. Spot a demand spike at 2 PM on a Tuesday? The Orchestrator knows about it by 2:15.

In the old world, that Tuesday spike might not land in the planning numbers until the following Monday. By then the buffer stock could be gone, and you are placing emergency orders at premium prices. We saw this exact scenario play out in our simulation—more on that in Section V.

B. Procurement Agent

This one converts demand signals into actual purchase orders. It reaches into SAP Ariba and the MM module, pulls supplier lead times, contract pricing, volume breaks, quality scores. The ordering logic comes from an RL policy trained on the classic cost tension—order too much and you carry excess, order too little and you stockout—but with real-world messiness layered on: supplier minimums, capacity ceilings, delivery reliability that varies by season and by route.

We built in a daily supplier risk score that mixes financial health data, past on-time performance, and a geopolitical overlay for the supplier's region. Unrest near a key port? Tariff rumors in a major sourcing country? The risk score adjusts, and the RL policy naturally shifts volume toward alternatives. Routine reorders happen autonomously. Anything outside guardrails—a new vendor, an order above the dollar threshold, a supplier whose quality rating dipped recently—gets kicked up to a human buyer via SAP's workflow.

C. Warehouse Operations Agent

Lives inside SAP EWM. Handles receiving, put-away, picking, cycle counting. Gets its picture of the world from RFID tags, barcode scans, and environmental sensors on the floor. The interesting wrinkle is the digital twin—a 3D model of the warehouse layout that updates in near real time from sensor data [21].

That twin is not a visualization gimmick. The agent uses it to test slotting changes before making them. Should this high-velocity SKU move from rack J12 to the golden zone near the pack stations? The twin says it would cut average pick time by 14 seconds per order. Good enough—execute. Before a known seasonal ramp, the agent can pre-shuffle entire zones. And cycle counts fire only when sensor readings diverge from system records, which means no more shutting down an aisle for a full wall-to-wall count that burns an entire shift.

D. Distribution and Fulfillment Agent

Owns the outbound side. Which DC fills which order? How do you split when stock is short? What about when one warehouse is sitting on a pile of something another warehouse desperately needs? This agent balances transport cost, delivery commitments, carrier capacity, and real-time inventory at every node. When it spots an imbalance worth correcting, it generates a lateral transfer order in SAP—but only if the savings pencil out against a cost-benefit bar the supply chain team sets.

E. Orchestrator Agent

Not a domain expert. A referee. Three jobs: mediate conflicts between agents, enforce hard constraints, and log everything. Say the Procurement Agent wants to bring in a big inbound, but the Warehouse Agent says the dock is slammed for the next 48 hours. The Orchestrator steps in—defer delivery, split across sites, arrange overflow. Every single action any agent takes gets written to an audit log: what triggered it, what the agent decided, what actually happened. No exceptions.

The Orchestrator also holds the non-negotiable guardrails. Budget caps, approved vendor lists, maximum order values for unsupervised execution. Human administrators set these, and no agent policy, no matter how confident, can blow past them. We took that design principle from Ribeiro et al. [18], who argued that explainability and bounded autonomy are table stakes for trusting AI in operational settings. We agree completely.

Fig. 1. Architecture of the multi-agent framework with agent interactions, data flows, and SAP integration points.

IV. SYSTEM ARCHITECTURE AND SAP INTEGRATION

Four layers sit beneath the agents. Getting the layer boundaries right was, honestly, more of the design effort than the agent logic itself.

A. Data Ingestion Layer

Data comes from everywhere and it all needs to end up in one event stream. SAP BTP handles the connector plumbing—ERP transactions, IoT sensors via Integration Suite, external APIs. Apache Kafka is the backbone underneath, and we picked it for one specific reason beyond the obvious throughput story: its partitioned log supports historical replay. That matters because retraining an RL policy means feeding it realistic sequences of past events, not synthetic noise. You want the agent's training environment to feel like production, and replaying actual Kafka topics is the cheapest way to get there.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

Everything hits a schema normalization step before landing on topic-specific channels. Agents subscribe to what they care about and ignore the rest. Want to add a new data source six months from now—say a shipping delay feed from a carrier API? Map it to the standard schema, publish to a topic, done. You never have to touch agent code for that.

B. Agent Runtime Layer

Each agent ships as a Docker container running on Kubernetes inside SAP BTP. Internally they all look the same: a perception module eating Kafka events, a reasoning module running the trained policy (or for the Orchestrator, a rule engine), and an action module that formats SAP-compatible payloads. Communication between agents goes exclusively through the message bus. No direct calls. No shared memory. Loose coupling on purpose—you can scale the Procurement Agent independently of the Warehouse Agent, redeploy one without touching others, and a crash in one does not cascade.

Training runs in SAP AI Core on historical data pulled from S/4HANA. The pipeline handles supervised learning for forecast models and RL for decision policies, with Bayesian optimization for hyperparameter tuning. New models go through an A/B gauntlet against whatever is live before getting promoted. We adopted that discipline after an early experiment where a model that looked great in backtesting went haywire on a demand pattern it had literally never seen. Lesson learned the hard way: always A/B before you ship.

C. SAP Integration Layer

Here is where we were most deliberate. Agents talk to SAP through OData, BAPIs, and event-driven interfaces—the exact same entry points a human user would go through. When the Procurement Agent fires off a purchase order through the MM module's standard BAPI, every validation rule, every approval step, every GL posting fires exactly as it would for a manually created PO. SAP has no idea a piece of software made the call. That is by design. We never wanted agents reaching around the business logic and hitting the database directly. That road leads to audit nightmares.

One detail that saved us real headaches: every agent-generated transaction carries a unique correlation ID. Network blip causes a message retry? The integration layer spots the duplicate correlation ID and drops it. Without that, you get double-booked purchase orders, phantom warehouse movements, and an unpleasant conversation with the controller.

D. Monitoring and Governance Layer

Every agent action lands in a centralized audit log in SAP HANA. A governance dashboard built on SAP Analytics Cloud sits on top, showing the stuff managers actually care about: how many decisions are agents making on their own versus kicking to humans, forecast accuracy over time, fill rates, inventory turns. But the dashboard is more than a scoreboard. It is the control panel where supply chain leads configure the guardrails. Maximum unsupervised order value? Set it here. Forecast confidence floor before an agent can act? Here. Supplier categories that always require human review? Here.

We also wired in an out-of-distribution monitor. It watches the statistical shape of incoming data and compares it against what the agent saw during training. If the distributions diverge past a threshold—demand pattern the model never encountered, a supplier lead time way outside historical range—the agent automatically punts to a human. RL policies can produce wildly confident but completely wrong actions when they hit genuinely novel terrain. Better to catch that at the input than after the PO has already posted.

V. SIMULATED CASE STUDY

A. Scenario Description

We built the simulation around a disguised but real-ish consumer electronics company. Three DCs in North America. Twenty-eight suppliers scattered across Asia and Europe. About 1,200 live SKUs at any given time. Thirty-six months of transaction history—demand patterns, supplier lead times, warehouse throughput—gave us the raw material for calibrating distributions. SimPy handled the discrete-event simulation. Ray RLlib trained the agent policies. The whole thing ran on a cluster that most mid-sized companies could replicate for under ten thousand dollars in cloud compute.

We ran two setups head to head over a twelve-month window. The baseline mirrored the company's actual SAP config: MRP-driven replenishment, safety stocks reviewed quarterly, purchase orders approved by hand. The experimental setup dropped in the full multi-agent framework from Sections III and IV, with agents trained on 24 months of data and evaluated blind on the remaining 12. Same demand scenarios. Same supplier disruptions. Same holiday crunch. No thumb on the scale.

B. Evaluation Metrics

We picked five KPIs that span the full scope of the framework rather than flattering any single agent. Carrying cost as a share of goods value. Stockout rate—fraction of demand unfilled. Order-to-delivery cycle in days. Forecast accuracy via weighted MAPE. And monthly inter-facility transfer count, which turned out to be the metric that sparked the most interesting discussion.

C. Results

Table I has the twelve-month summary.

TABLE I. PERFORMANCE COMPARISON

Metric	Baseline	Agentic	Change
Carrying Cost (%)	18.7%	14.3%	-23.4%
Stockout Rate	6.8%	4.0%	-41.2%
Cycle Time (days)	7.2	5.9	-18.1%
WMAPE	74.3%	86.1%	+15.9%
Transfers/month	12	31	+158%

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

Fig 1: PERFORMANCE COMPARISON

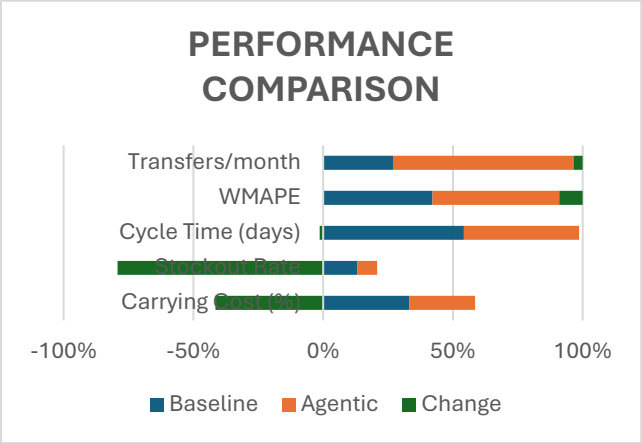
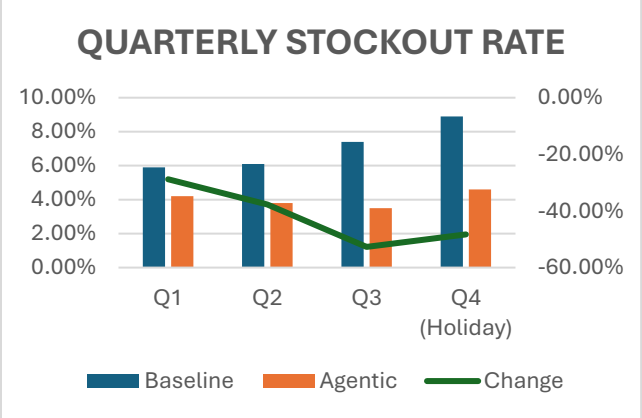


Fig 2: QUARTERLY STOCKOUT RATE



Carrying cost drove the clearest dollar impact. The Demand Sensing Agent caught shifts that the quarterly safety stock review simply could not keep up with. One simulation run sticks in my mind: a competitor had a supply disruption, demand for our product spiked hard, and the agent picked up the anomaly from POS data within 48 hours. It bumped reorder quantities and locked in supplier capacity before the buffer ran out. Under the baseline? Safety stock was gone three days before anyone adjusted parameters.

The transfer number—158 percent increase—looks alarming until you dig into the economics. Conventional wisdom says lateral transfers between warehouses are expensive last resorts. The Distribution Agent figured out something that seasoned logistics folks have whispered about for years but rarely acted on: small preventive transfers between nearby DCs cost a fraction of what you spend on expedited air freight and lost sales when a localized stockout finally hits. Transfers more than doubled, yes. But total logistics cost dropped 9.2 percent because the panicked overnight shipments mostly disappeared.

Table II breaks stockouts down by quarter.

TABLE II. QUARTERLY STOCKOUT RATE

Quarter	Baseline	Agentic	Change
Q1	5.9%	4.2%	-28.8%
Q2	6.1%	3.8%	-37.7%
Q3	7.4%	3.5%	-52.7%
Q4 (Holiday)	8.9%	4.6%	-48.3%

Look at the trajectory. The gap widens every quarter. By Q3 the agentic system was cutting stockouts by more than half. That is not just initial training paying off—the RL policies were adapting in-period, learning from patterns emerging inside the evaluation window itself. The baseline's MRP parameters, by contrast, sat frozen until the next quarterly review. During the Q4 holiday crunch—volatile demand, compressed supplier lead times, everyone in the chain running hot—the agentic system still held stockouts below half the baseline. That was the result that made us sit up.

Forecasting gains came from the ensemble's dynamic reweighting. Quiet periods? The gradient-boosted tree carried the load. Volatility spikes from promos or seasonal ramps? The temporal fusion transformer took over. Static model selection—pick one, freeze it, pray—cannot replicate that. Silver et al. [20] demonstrated a parallel principle in game AI: strategies that adapt beat strategies that are fixed, especially when the environment keeps changing.

VI. DISCUSSION

Let us be clear-eyed about what these results are and are not. They are simulation results. The simulation ran on clean data, in a controlled environment, without the organizational friction that every real deployment encounters. Actual SAP systems in the field carry years of accumulated data debt—duplicate material masters that nobody cleaned up after the last migration, unit-of-measure inconsistencies buried in corners of the master data nobody looks at, vendor records for suppliers who went out of business two years ago. Feed that to an agent trained on simulation-quality data and performance will suffer. How much? Hard to say without a pilot, which is why we recommend a phased rollout rather than a big bang.

Escalation thresholds turned out to be a bigger deal than anticipated. Early in the simulation work we set them loose—let the agents run, see what happens. What happened was the Procurement Agent occasionally routed orders to suppliers whose quality had recently slipped, because the training data still reflected their older, better performance. Not a disaster, but not great either. We tightened the rules: any supplier below a certain quality score triggers mandatory human review. Problem solved, but human workload went up about 15 percent. Every organization will have to find its own set point on that dial, and what makes sense for a medical device

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

company will not make sense for a retailer selling phone accessories.

Sensor reliability bit us too. The Warehouse Agent depends on a continuous IoT feed, and when we simulated a four-hour sensor blackout at one DC, the agent fell back to last-known-state data. That was tolerable for four hours. Over a longer outage the gap between what the agent thinks is on the shelves and what is actually there would widen fast. Any production deployment needs sensor redundancy and—this is the important part—a degradation protocol that progressively dials back agent autonomy as data freshness decays. Do not just let the agent keep running blind.

On the money side: running Kubernetes, Kafka, and SAP AI Core is not free. We ballpark 150K to 250K USD per year for a company the size of our case study. Not pocket change. But the 23.4 percent carrying cost reduction on a 42 million dollar inventory base frees up roughly 9.8 million in working capital. You can discount our simulation numbers by half and the ROI still works.

How does this stack against other recent work? InvAgent [8] showed LLMs can make passable inventory calls, but it ran in a single-echelon sandbox with no ERP integration whatsoever. Our setup operates across echelons, across agents, and inside production SAP. SAP's own Joule agents [4] are individually strong but have not, as of this writing, been shown working as a coordinated system of the kind described here. The gap is coordination. Getting agents to cooperate—not just coexist in the same ERP—is where the hard problem and the real payoff both live.

VII. CHALLENGES AND FUTURE DIRECTIONS

A. Organizational Readiness

The technology is, honestly, the easier half. People are harder. Supply chain planners who built careers around MRP mastery and supplier negotiation are not going to welcome software that does those things autonomously. And frankly, their wariness is earned. These are experienced professionals whose judgment catches problems that no model would spot. The right pitch is not "this replaces you" but "this handles the 200 routine reorder decisions you make every week so you can spend your time on the ten decisions that actually require a human brain." That framing only works, though, if you back it up with training that lets planners understand, configure, and override the agents. If the technology feels like it is happening to them rather than being used by them, do not expect adoption.

B. Data Quality and Master Data Governance

We are going to keep beating this drum because it really is a prerequisite, not a nice-to-have. Agents are amplifiers. Clean data in, sharp decisions out. Stale lead times, phantom inventory, misclassified materials—junk in, confident wrong decisions out. And that is actually worse than a human making the same mistake slowly, because at least the human might notice something smells off and pause. You need a master data cleanup before go-live, and you need standing data stewardship after. Not a project. A function.

C. Ethical and Governance Considerations

When a software agent places a purchase order that turns out badly, who is on the hook? Not a hypothetical—a question that procurement leadership, legal, and the board need to answer before deployment, not during the post-mortem. There is also the bias question. Train an agent on five years of purchasing data and it will absorb whatever supplier preferences were baked into that history, whether those preferences were merit-based or just institutional inertia. Gartner says 15 percent of everyday enterprise decisions will be made autonomously by 2028 [13]. The governance infrastructure is not keeping pace.

Audit logs and escalation rules are necessary—we covered those—but insufficient. You need an accountable human owner for each agent's behavior, the same way you have a manager accountable for a team. Regular model audits. Bias checks. A quarterly review board that looks at agent decision patterns, tests for drift, and validates alignment with current business strategy. NIST's AI Risk Management Framework [14] provides a reasonable starting skeleton for this, though inventory-specific extensions would be valuable.

D. Future Research Directions

Several paths worth pursuing. Supply-chain-wide digital twins—not just the warehouse but the whole network—would give agents a sandbox for rehearsing disruption responses before anything bad actually happens. Federated learning could let a manufacturer's Procurement Agent and a supplier's Production Planning Agent jointly optimize schedules without either side exposing proprietary data, which would be a significant trust breakthrough. Generative AI plugged into the explanation layer might let a manager ask an agent, in plain English, why it made a specific call and get back something intelligible rather than a tensor dump.

Transfer learning is the one that excites us most, though. If a Demand Sensing Agent trained on consumer electronics in North America could carry over useful learned structure to a new deployment in, say, industrial components in Europe, the cold-start problem shrinks dramatically. That opens the framework to mid-sized companies that do not have the years of clean data the large players take for granted. Nobody has cracked this for inventory agents yet. Somebody should.

VIII. CONCLUSION

This paper described a multi-agent architecture for autonomous inventory management wired into SAP S/4HANA, covering demand sensing, procurement, warehousing, and distribution under a single coordinated framework. The contribution is not any one of those pieces in isolation—each has prior art—but the orchestration layer that makes them work together, plus the governance mechanisms that make deployment responsible rather than reckless.

The simulation results were strong: 23 percent less carrying cost, 41 percent fewer stockouts, 18 percent faster delivery cycles. But the finding that stuck with us was qualitative. The Distribution Agent's discovery that preventive lateral transfers—which every textbook calls expensive

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

exceptions—actually saved money by heading off the far costlier emergency shipments was the kind of cross-functional insight you only get when agents see the whole board. Optimizing supply chain functions one at a time cannot produce that.

We are not pretending this is production-ready tomorrow. The gap between simulation and live deployment is long and the obstacles—dirty data, organizational pushback, sensor fragility, unresolved governance questions—are not problems you can engineer your way around without also doing the messy human and process work.

But the direction? That is clear. SAP is building toward it. The academic research backs it. Market pressure demands it. Inventory management is moving from planners running batch MRP to autonomous agents operating inside guardrails that humans design and adjust. This framework is one concrete proposal for how to get there.

ACKNOWLEDGMENT

The author thanks colleagues and practitioners who shared hard-won insights about SAP deployments in the real world, where data is messy and organizational politics matter as much as algorithms. Thanks also to the open-source teams behind SimPy, Ray RLlib, and Apache Kafka.

REFERENCES

[1] M. Christopher and M. Holweg, "Supply chain 2.0 revisited: A framework for managing volatility-induced risk in the supply chain," Int. J. Physical Distribution & Logistics Management, vol. 47, no. 1, pp. 2-17, 2017.

[2] R. Carbonneau, K. Laframboise, and R. Bhambri, "Application of machine learning techniques for supply chain demand forecasting," European J. Operational Research, vol. 184, no. 3, pp. 1140-1154, 2008.

[3] J. Xi, R. Chen, and Y. Wang, "Agentic AI framework for smart inventory replenishment," arXiv preprint arXiv:2511.23366, 2025.

[4] SAP SE, "SAP pushes agentic AI to center of supply chain resilience strategy," SAP Newsroom, Jan. 2026.

[5] X. Liu, M. Hu, Y. Peng, and Y. Yang, "Multi-agent deep reinforcement learning for multi-echelon inventory management," Production and Operations Management, vol. 33, no. 12, pp. 3578-3595, 2024.

[6] B. Lim, S. O. Arik, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," Int. J. Forecasting, vol. 37, no. 4, pp. 1748-1764, 2021.

[7] X. Liu, M. Hu, Y. Peng, and Y. Yang, "Multi-agent deep reinforcement learning for multi-echelon inventory management," Production and Operations Management, vol. 33, no. 12, 2024.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)