ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov552

Agentic AI for Autonomous Detection and Resolution of Hold Transactions in Integrated Multi-Vendor Hospital Pharmacy Systems

Samiyullaha Sayyed[1]

[1]Healthcare IT Consultant, Pharmacy and Supply Chain Applications Independent Researcher, India

Publication Date: 2026/11/24

Abstract: This paper presents an agentic artificial intelligence framework that detects, diagnoses and resolves hold transactions, which are stuck or failed transactions, in integrated hospital pharmacy environments. Large health systems often run order entry, medication processing, billing, stock and inventory, laboratory, blood bank and automated dispensing software from different vendors, and holds arise at the interfaces where no single vendor sees the whole transaction. Analysts resolve holds manually today, so backlogs build up and stock records drift from physical reality. The research problem is how to automate the repetitive part of this work without weakening audit, inventory accuracy or patient safety. Using a design science approach, the paper develops a taxonomy of five hold causes, a five agent architecture based on a monitor, analyze, plan, execute and knowledge control loop, a tiered autonomy policy with approval gated reprocessing, and an artificial intelligence governance model. A 30 day simulation with 20 replications and a sensitivity analysis evaluate the design. Under stated assumptions, about 82 percent of holds are resolved automatically and mean handling time falls from about 43 to 11 minutes, but the human escalation queue stays bounded only above a break even of about 80 percent automatic resolution. The work indicates that governed agentic automation is a promising option for multivendor hospital pharmacy operations, provided that automatic resolution and false fix rates are confirmed through the staged pilot protocol proposed here.

Keywords: *Agentic AI; AI Governance; Healthcare Information Systems; HL7 Interfaces; Hold Transactions; Hospital Pharmacy; Multi-Vendor Integration; Supply Chain Integration.*

How to Cite: Samiyullaha Sayyed (2026) Agentic AI for Autonomous Detection and Resolution of Hold Transactions in Integrated Multi-Vendor Hospital Pharmacy Systems. *International Journal of Innovative Science and Research Technology*, 11(11), 1324-1343. https://doi.org/10.38124/ijisrt/26nov552

I. INTRODUCTION

Integrated hospital pharmacy operations depend on the seamless coordination of multiple, often vendor-distinct, software systems: a drug dispensing / pharmacy application, an inventory and supply-chain management system, a patient medication billing system, and an automated drug dispensing system (ADS) such as cabinet- or carousel-based dispensing hardware. These systems are connected through an integration engine that exchanges HL7 and related messages for orders, charges, inventory movements, and dispensing events. When a large healthcare organization operates at scale — spanning multiple hospital facilities and numerous outpatient health centers — the daily volume of dispensing and inventory transactions is enormous, and even a small percentage failure rate produces hundreds of transactions that fail to post correctly and instead land in a “hold” or exception queue.

A hold transaction is operationally dangerous because it frequently represents a disconnect between physical reality and system-of-record data: the pharmacist has physically dispensed the medication, the patient has been billed, and the medication has left the shelf, yet the inventory system has not deducted the stock because the transaction failed validation. Left unresolved, these discrepancies accumulate into significant inventory inaccuracy, complicate reconciliation, obscure true drug utilization, and can affect replenishment and controlled-substance accounting. Today, resolving these holds is a manual, expert-driven activity performed by application and interface support analysts who must identify the specific failure category, locate the missing or incorrect configuration, correct it, and resubmit the transaction — a repetitive but judgment-intensive task performed anew every day across dozens of facilities.

This paper proposes replacing the repetitive portion of that human workflow with an Agentic AI system: a set of cooperating software agents that monitor hold queues in real time, diagnose the underlying cause using a combination of rule-based logic and large-language-model (LLM) reasoning over system metadata and error messages, apply the appropriate corrective configuration change, and resubmit the transaction automatically — while escalating genuinely ambiguous or clinically sensitive cases to a human analyst.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov552

The goal is not to remove human oversight from the system but to absorb the high-volume, pattern-recognizable share of hold resolution, so that support staff can focus on novel or high-risk exceptions.

➢ *Research Questions, Contribution and Method*

This paper addresses three questions. RQ1: what failure modes account for hold transactions in an integrated pharmacy, inventory, billing and automated-dispensing environment, and which of them are suitable for automated remediation? RQ2: what agent architecture and governance controls allow such remediation to be applied without weakening audit, inventory-accuracy or patient-safety assurances? RQ3: under what conditions would the approach deliver a bounded backlog and faster resolution, and how fragile are those conditions?

The paper makes four contributions:

- A pharmacy-specific root-cause taxonomy of hold transactions across five failure categories (Table 1).
- A five-agent, governed remediation architecture formalised through the MAPE-K control loop and BDI agent model, with a shared event schema and explicit routing logic (Section V).
- A tiered autonomy policy and AI-governance model that assigns every fix type to automatic, automatic-with-review, approval-required or always-escalate handling, following the levels-of-automation model of Parasuraman et al. [22], together with the approval workflow, runtime safeguards and assurance cycle that govern the agents (Sections 5.4, 5.6 and 5.7; Tables 5 and 8).
- A reproducible simulation, sensitivity analysis and pilot validation protocol that state the conditions under which the benefits hold, including an analytic break-even for the escalation queue, and define the measurements needed to confirm or refute them (Sections VII and 8.4).

Methodologically, the work follows a design-science approach [21]: the framework is the designed artifact, the taxonomy and governance controls are its design rationale, and the simulation provides a preliminary, assumption-driven evaluation that is explicitly not a substitute for field validation. The paper is scoped to administrative and configuration-level failures of already-completed transactions, in single-suite and in multi-vendor (best-of-breed) environments (Section 2.3). It does not address clinical decision support, order verification, or any change to what was prescribed, dispensed or administered.

➢ *Background and Problem Statement*

- *Integrated Pharmacy Ecosystem Architecture*

A representative large-hospital-system architecture consists of four to six interconnected application domains: (1) the pharmacy / drug dispensing application, which records what was dispensed against a prescribed order; (2) the inventory and supply-chain management system, which tracks stock levels, requisitions, and distributions across stores and pharmacy locations; (3) the patient billing system, which charges the patient or payer for the dispensed medication; and (4) the automated drug dispensing system (ADS), which physically stores and releases medication at the point of care. These systems are connected by an integration engine (interface engine) that translates and routes messages — commonly HL7 v2.x [18] — between systems using configured inbound and outbound interface "aliases." A single dispensing event may generate five or more downstream messages across these systems.

- *Nature and Scale of the Problem*

In a large healthcare organization of the scale considered here — spanning multiple hospital facilities and numerous specialty and outpatient health centers — the pharmacy and inventory teams process an extremely high daily volume of dispensing, requisition, and distribution transactions. Application and integration support teams currently review hold queues manually each day, transaction by transaction, to identify why a given record failed. This is time-consuming, inconsistent across analysts, delayed (holds may sit for hours before review), and produces a persistent gap between physical and system inventory that compounds over time across every facility.

The author's earlier published work examined configurable, mistake-proofed approaches to adding and removing medications across hospital pharmacy inventory locations [19], and the reasons stock discrepancies persist in hospital pharmacy supply chains despite robotics, system integration and artificial intelligence [20]. The present paper builds on those observations by moving from the diagnosis of build and configuration causes to governed, automated remediation of the resulting holds. The transactions considered in this paper fail primarily for one of five reasons, summarized in Table 1.

Table 1 Root-Cause Taxonomy of hold Transactions in an Integrated Pharmacy Environment.

Category	Typical Trigger	Example	Primary System(s) Involved
Master-data / build gaps	Required linkage missing at build time	Medication not mapped to the dispensing pharmacy location	Pharmacy, Inventory
Missing financial attributes	New item build incomplete	Newly built medication missing a cost/price record	Inventory, Billing
Fractional / negative stock	Unit-of-issue vs. unit-of-dispense mismatch	Medication dispensed in a fraction of the stocking unit	Inventory, ADS
Multi-batch / lot conflicts	Concurrent distribution across batches	Requisition-distribution spans multiple batches of the same drug	Supply Chain / Requisition-Distribution
Interface / interoperability errors	Alias, mapping, or format mismatch	HL7 inbound/outbound message bounces at the interface engine	Integration Engine, ADS, Billing

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov552

- *Multi-Vendor (Best-of-Breed) Environments*

Many organizations do not run a single vendor suite. They deliberately choose the application that best fits each department's daily operations and reporting needs: one system for order entry, another for medication processing in the pharmacy, a third for billing and revenue cycle, a fourth for stock and inventory, and separate systems for the laboratory, the blood bank and automated dispensing (Figure 1). The same pattern also results from mergers, phased procurement, and the difficulty of replacing a well-liked departmental system. Whether best-of-breed or a single suite is preferable is a long-standing debate without a universal answer; this paper takes the multi-vendor environment as given and asks what it implies for transaction reliability.

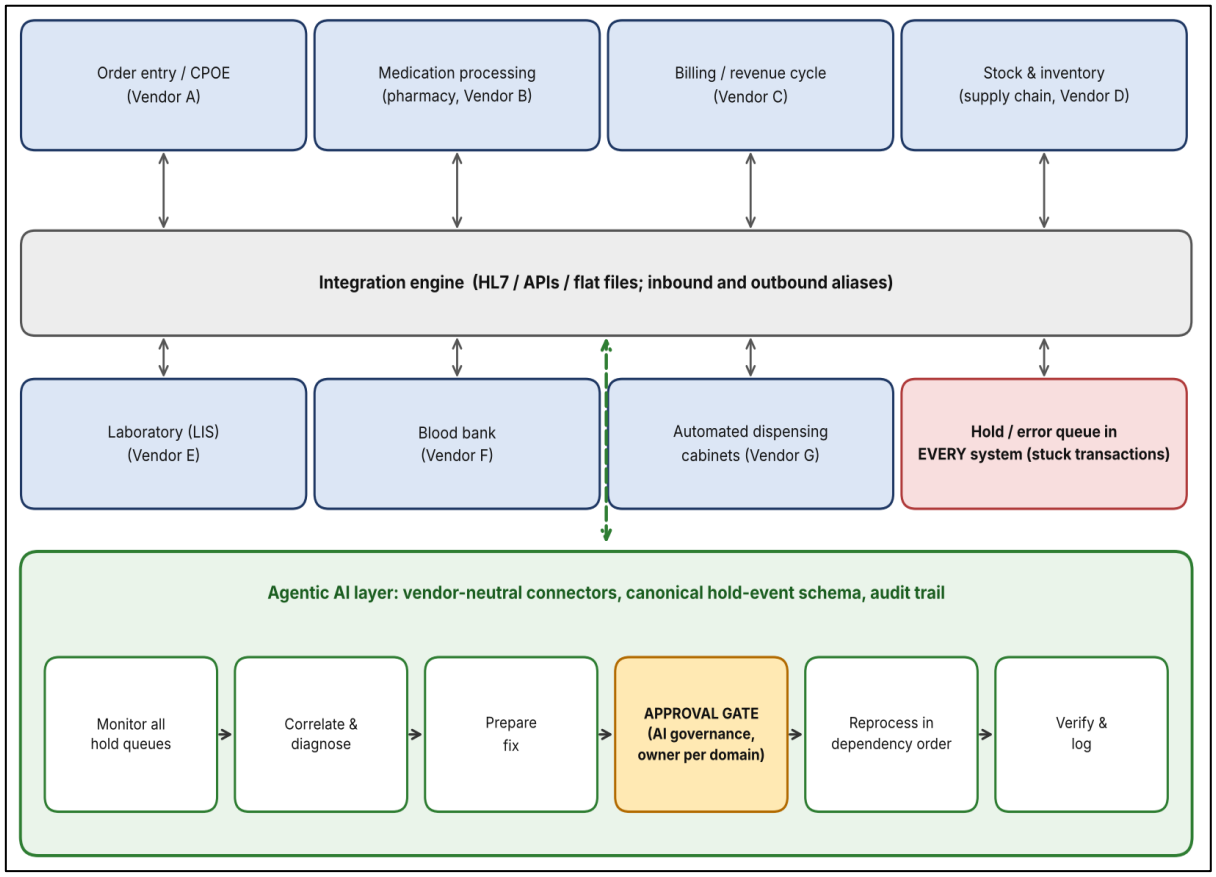

Fig 1 A Multi-Vendor Environment: Vendor Applications Exchange Transactions Through an Integration Engine, Holds Accumulate in Each System's Own Queue, and a Vendor-Neutral Agentic AI Layer Works them Through a Governed Approval Gate.

The consequence is that the interfaces become the operational seams of the organization. Each vendor keeps its own master data, identifiers, units of measure, validation rules and error semantics, and each support team sees only its own queue. A hold that appears in one system frequently has its cause in another: a charge held in billing because a code was never mapped, a stock deduction held in inventory because an item was built in the pharmacy system but not in the supply chain system, or a message rejected because a vendor changed a field format in an upgrade. No single vendor sees the whole business event, so diagnosis depends on staff who know several systems and can trace one order across them by hand. This is consistent with the persistence of stock discrepancies in integrated pharmacy supply chains discussed in the author's earlier work [20].

These are the conditions in which an agentic layer is most useful. A vendor-neutral layer can watch every system's hold queues, correlate related holds to one upstream cause and, once approval is recorded under the organization's AI-governance policy, reprocess them quickly (Sections 5.6 and 5.7). It supplies the cross-system visibility that no individual vendor provides, and it reduces the human contribution to the decision, not the investigation. Table 2 summarises the application domains, their typical hold causes and the proposed handling.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov552

Table 2 Application Domains in a Multi-Vendor Estate, Typical Hold Causes and the Proposed Agent Handling.

Application domain	Role in a multi-vendor estate	Typical hold or stuck-transaction causes	Proposed agent handling (Table 5)
Order entry / CPOE	Captures orders and releases them downstream	Order released with an item or catalog code unknown downstream; duplicate or out-of-sequence order messages	Detect and correlate; administrative mapping fixes only; order content as written is never altered (Tier X)
Medication processing (pharmacy)	Verification, dispensing and dispense events	Item not mapped to a dispensing location; fractional dispense; dispense event rejected by inventory or cabinet	Tier A for mappings; Tier C for unit-of-measure changes
Billing / revenue cycle	Charge capture and posting	Charge code or price missing; charge rejected for an unmapped item or location	Tier C for charge-master changes, approved by the revenue-cycle owner
Stock and inventory (supply chain)	Stock, requisition, distribution and valuation	Missing cost record; negative or fractional stock; multi-batch distribution conflict	Tier B for cost within tolerance; Tier C for quantity and lot changes
Laboratory (LIS)	Orders, results, reagent and consumable usage	Consumable or reagent usage posting held for an unmapped item; charge posting rejected	Administrative posting holds only; results and specimen records are Tier X
Blood bank	Product inventory, issue and transfusion records	Product stock or charge posting held for an unmapped product code or unit	Administrative postings only, after approval; clinical blood bank records are Tier X
Cabinets (ADS) and interface engine	Cabinet transactions; message routing	HL7 alias or field-mapping mismatch; receiving system unavailable	Tier A for retry; Tier B for an alias correction that matches a documented change

II. RELATED WORK

Agentic AI for autonomous inventory management has recently been proposed outside healthcare: Syed et al. [1] describe a multi-agent framework that forecasts retail demand, negotiates with suppliers, and adjusts replenishment without step-by-step human direction, illustrating the general pattern of sense-decide-act agent loops applied to inventory operations. Within healthcare, commercial interface-engine monitoring platforms such as iTechSmart HL7 [6] already perform AI-driven root-cause analysis on stalled HL7 messages and can retry failed messages or restart stalled interface services automatically, which is closely related to — but narrower than — the interface-level remediation described in Section 6.5 of this paper. Work on agentic business process management [2, 3] discusses agents that detect and autonomously correct master-data errors such as mismatched product codes, directly supporting the master-data-gap remediation pattern used throughout Sections IV and VI. A recent taxonomy of agentic AI in healthcare and medicine [4] evaluates error-recovery capability — detection, diagnosis, and corrective action — as a distinct dimension of agent maturity across published clinical LLM-agent systems, none of which specifically targets pharmacy or inventory hold-transaction remediation. Existing pharmacy-focused AI research has concentrated on demand forecasting and stock-level optimization rather than transaction-level failure remediation [5, 7], and industry reporting on agentic AI adoption in hospital pharmacy operations remains largely limited to communications, diversion detection, and cross-vendor data synthesis rather than autonomous transaction repair [10]. Multi-agent system theory [9] and classical AI planning and reasoning frameworks [8] provide the conceptual basis for decomposing a monitoring-diagnosis-remediation-validation pipeline into specialized, cooperating agents rather than a single monolithic automation script.

The closest architectural analog identified in this review is outside healthcare entirely: Malik [16] describes a production-deployed agentic AI architecture for autonomous network-incident resolution at a major cloud provider, built around the same underlying lifecycle proposed here — specialized agents for intake/detection, planning/diagnosis, execution/remediation, and closed-loop verification, coordinated under a progressive-autonomy safety framework with bounded authority, blast-radius containment, and automatic rollback. That system reports production outcomes (autonomous resolution exceeding 90% for well-understood incident categories, resolution time reduced by roughly two orders of magnitude) that are directly relevant here: they provide external, production-grade evidence that the general detect-diagnose-remediate-verify agentic pattern this paper proposes for pharmacy hold transactions is technically viable at scale, even though that evidence comes from a different operational domain (cloud network infrastructure, not healthcare supply chain). This paper's contribution relative to [16] is therefore not the general architectural pattern — which is convergently similar, reflecting a broader emerging consensus around MAPE-K-style autonomous remediation — but its adaptation to a domain with a materially different risk profile (patient-facing inventory and billing accuracy rather than network availability), a pharmacy-specific root-cause taxonomy (Table 1), and formal governance controls suited to clinical-supply-chain audit requirements (Section 5.4). It should also be stated plainly that unlike [16], the framework in this paper has not yet been production-validated; Section VII's simulation is offered as a directional plausibility check pending the pilot validation protocol in Section 8.4 and the future work in Section XI, and the production results in [16] should be read as evidence for the general pattern's feasibility rather than as a substitute for that validation.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov552

Three further bodies of literature shape the governance design. First, human-automation interaction research distinguishes levels and stages of automation and shows that the appropriate level depends on the consequence of error [22]; it also documents misuse, disuse and complacency when operators over-trust automation [23], which motivates the sampling and override controls in Sections 5.4 and X. Second, AI-safety work on avoiding unintended side effects and maintaining scalable oversight [24], together with the reasoning-and-acting pattern for LLM agents [25], motivates constraining the LLM to diagnosis while routing every state-changing action through a bounded, auditable API. Third, socio-technical studies of health information technology [26] and risk-management guidance for AI systems [27] frame the organizational and compliance controls in Sections VIII and X.

The empirical state of the healthcare agent literature reinforces the gap addressed here. The review of 49 healthcare LLM-agent studies in [4] reports that event-triggered activation and drift detection are largely unimplemented (roughly 92% and 98% rated "not implemented", respectively); both capabilities are central to the Monitoring Agent and the knowledge-drift controls proposed in this paper (Sections 5.2 and X). Table 3 positions the present work against the closest prior contributions identified in this review.

Table 3 Positioning of the Proposed Framework Against the Closest Prior Work Identified in this Review.

Work	Domain	Primary function	Transaction-level repair	Pharmacy hold taxonomy	Cross-system scope
Syed et al. [1]	Retail inventory	Demand forecasting, supplier negotiation, replenishment	No	No	Supply side only
iTechSmart HL7 [6]	Healthcare interface engine	AI root-cause analysis; retry or restart of stalled messages	Partial (interface layer)	No	Interface engine only
Agentic BPM [2, 3]	Business processes (general)	Agent governance; master-data error correction pattern	Conceptual	No	Generic
Vatsal et al. [4]	Healthcare LLM agents	Taxonomy and review of 49 agent studies	None targets hold remediation	No	n/a
Pharmacy AI [5, 7, 10]	Pharmacy operations	Demand forecasting, stock optimisation, diversion detection	No	No	Single domain
Malik [16]	Cloud network operations	Detect, plan, execute, verify with bounded authority and rollback	Yes (network incidents)	No	Network domain only
This paper	Hospital pharmacy supply chain	Detect, classify, remediate, verify and escalate hold transactions	Yes (governed, tiered)	Yes (Table 1)	Dispensing, inventory, billing, ADS, interface engine

This paper differs from the above work in combining, within a single framework, (a) cross-system hold-queue monitoring across dispensing, inventory, billing, and ADS platforms, (b) a pharmacy-specific root-cause taxonomy, and (c) governed, auditable auto-remediation and resubmission at the transaction level — a configuration not previously addressed in this combination in the literature reviewed.

III. ROOT CAUSE TAXONOMY OF HOLD TRANSACTIONS

Effective autonomous remediation depends on a precise taxonomy of failure modes, since the corrective action differs substantially by category. The five categories introduced in Table 1 are elaborated below.

➢ *Master-Data / Build Deficiencies*

The most common failure mode occurs when a medication, store, or pharmacy location has been built in one system but the corresponding cross-reference has not been completed in another. A frequent example is a medication that has not been mapped to the dispensing pharmacy location: the pharmacist dispenses the drug and the patient is billed, but the inventory system rejects the stock deduction because it has no record that this item is valid at that location. The physical and financial transaction has already occurred; only the inventory posting is blocked.

➢ *Missing Financial / Cost Attributes*

When a new medication is built into the formulary and item master but the associated cost record is not populated before the item becomes dispensable, transactions against that item fail inventory valuation checks and are held pending completion of the cost build.

➢ *Fractional and Negative Stock Conditions*

Certain medications are dispensed in fractional quantities relative to their stocking unit of issue (for example, a partial vial or a split tablet). If the system's unit-of-measure conversion is not configured for fractional dispensing, the resulting transaction can drive the calculated stock below zero or produce a non-integer quantity that fails validation, placing

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov552

the transaction on hold even though the physical dispense was correct.

➢ *Multi-Batch / Lot Distribution Conflicts*

Requisition-and-distribution transactions in the supply-chain module can span more than one batch or lot of the same medication when a single request is fulfilled from two partially depleted lots. Some system configurations do not gracefully support multi-batch fulfillment in a single distribution transaction, causing the transaction to hold pending manual batch reconciliation.

➢ *Interface and Interoperability Errors*

The final and often highest-volume category involves the integration engine itself. Outbound and inbound interface aliases — the configured endpoints and message-mapping definitions used to route HL7 messages between the pharmacy, inventory, billing, and ADS systems — can bounce a message when an alias is missing, a field mapping does not match the receiving system's expected format, or the receiving interface is temporarily unavailable. These messages accumulate in the interface engine's own error or hold queue, separate from but related to the application-level hold queues.

➢ *Cross-Vendor Hold Patterns*

In a multi-vendor environment the five categories of Table 1 still describe the immediate cause of a hold, but several patterns cut across them and determine how the agents should behave:

- One cause, many holds. A single upstream gap, such as an item built in one system but not in two others, produces separate holds in separate queues. The agents should treat them as one case with one root cause and one approval.
- Symptom and cause in different systems. The system that displays the hold is often not the system that must be corrected, so diagnosis must locate the owning system.
- Identifier and unit mismatches. Each vendor has its own codes and units, so the same item, location or charge may need an explicit mapping in each system.
- Dependency order. Some held transactions can be reprocessed only after another has posted, so reprocessing must follow dependency order and be verified end to end.
- Version drift. A vendor upgrade can change a field format or code set and cause a wave of holds, which links to the knowledge-drift risk in Section X.

IV. PROPOSED AGENTIC AI FRAMEWORK

➢ *Architecture Overview*

The proposed framework introduces an Agentic AI layer that sits alongside — rather than inside — the existing pharmacy, inventory, billing, ADS, and interface-engine systems, communicating with each through existing APIs, database views, or message queues. The layer is composed of five cooperating agents, described in Table 4, coordinated by an orchestration/workflow controller and backed by a shared knowledge base of master-data rules, historical remediation patterns, and interface specifications.

Table 4 Agent Roles within the Proposed Agentic AI Framework.

Agent	Function	Key Inputs / Outputs
Monitoring Agent	Continuously polls hold queues across pharmacy, inventory, billing, and interface-engine systems	Input: system hold-queue tables/APIs. Output: normalized hold-event records
Diagnostic Agent	Classifies each hold event against the root-cause taxonomy using rules plus LLM reasoning over error text and metadata	Input: hold-event record, error codes/messages. Output: root-cause category and confidence score
Remediation Agent	Retrieves the missing/incorrect configuration and applies the governed corrective action (e.g., add item-to-location mapping, populate cost record)	Input: root-cause category. Output: applied configuration change, change log
Validation / Resubmission Agent	Re-triggers the held transaction and confirms it clears the queue and posts correctly on both sides of the integration	Input: corrected record. Output: resubmission result, success/failure status
Escalation Agent	Routes low-confidence, clinically sensitive, or repeated-failure cases to a human analyst with full diagnostic context	Input: unresolved or low-confidence cases. Output: ticket with root-cause hypothesis and suggested fix

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov552

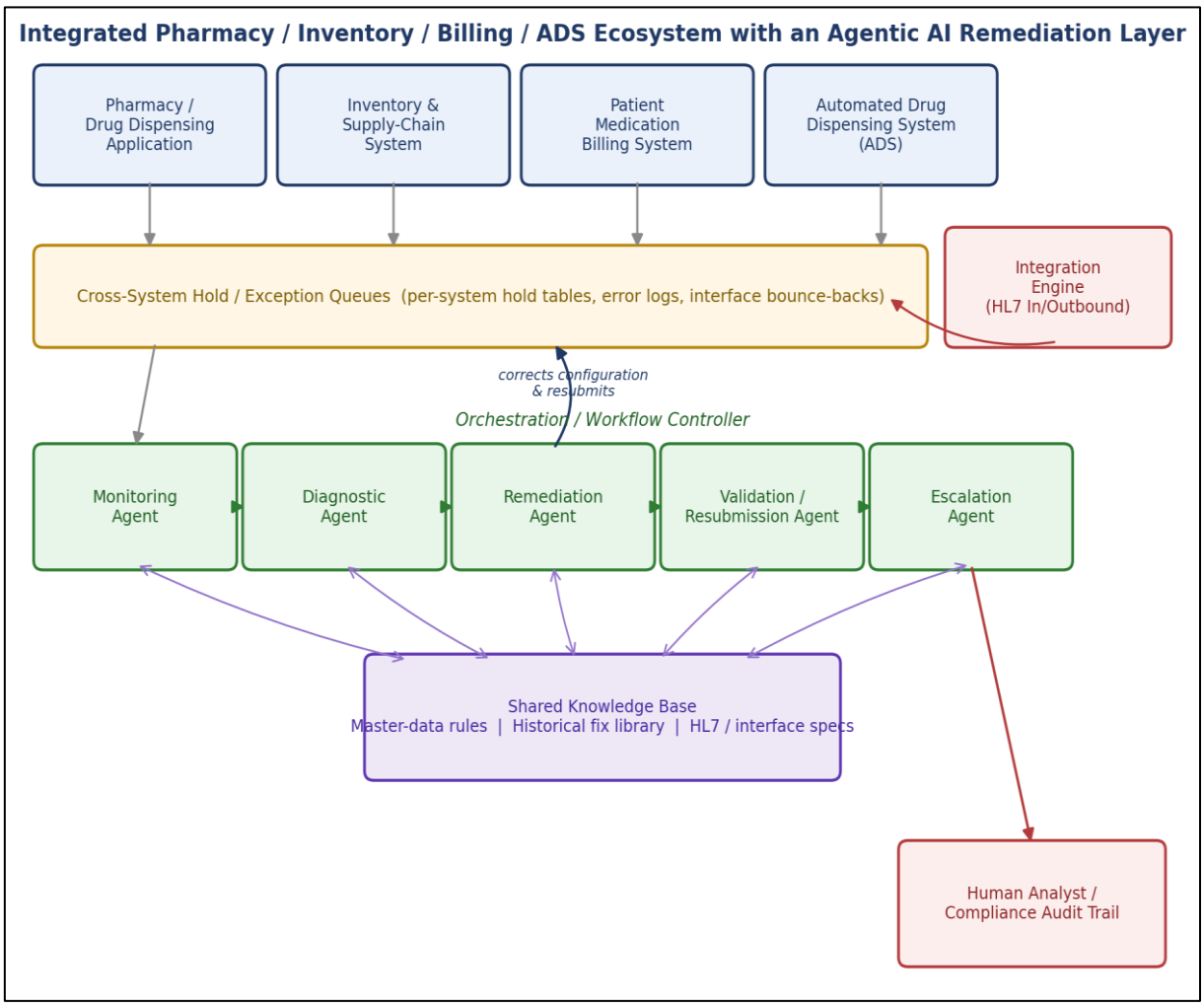

Fig 2 System Architecture: the Agentic AI Layer Sits Alongside Existing Pharmacy, Inventory, Billing, ADS, and Interface-Engine Systems, Reading from Shared hold/Exception Queues and Writing Corrections Back Through the Same Channels.

➢ *End-to-End Workflow*

The framework operates as a continuous loop, illustrated in Figure 3:

- Detect — the Monitoring Agent identifies new or aged records in hold queues across all connected systems and normalizes them into a common event schema.
- Classify — the Diagnostic Agent determines the failure category from Table 1 using deterministic rules where the error is unambiguous (e.g., a specific error code) and LLM-based reasoning over free-text error descriptions and surrounding metadata where it is not.
- Retrieve — the Remediation Agent queries the relevant master-data source (item master, location master, cost table, batch/lot ledger, interface alias configuration) to identify the specific missing or incorrect value.
- Remediate — the Remediation Agent applies the correction through the target system's supported configuration API (e.g., adds the item-to-location mapping, populates the missing cost, adjusts the unit-of-measure conversion, or corrects the interface alias) subject to pre-approved change rules and permission scopes.
- Resubmit — the Validation Agent triggers reprocessing of the held transaction through the normal transaction pipeline, exactly as if a human analyst had corrected the configuration and clicked resubmit.
- Verify — the Validation Agent confirms the transaction has cleared the hold queue and that stock, billing, and interface states are now consistent across systems.
- Escalate or Log — successful resolutions are logged with full audit detail for compliance review; unresolved or low-confidence cases are escalated to a human analyst with the diagnostic trail attached, avoiding “silent” automated changes in ambiguous situations.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov552

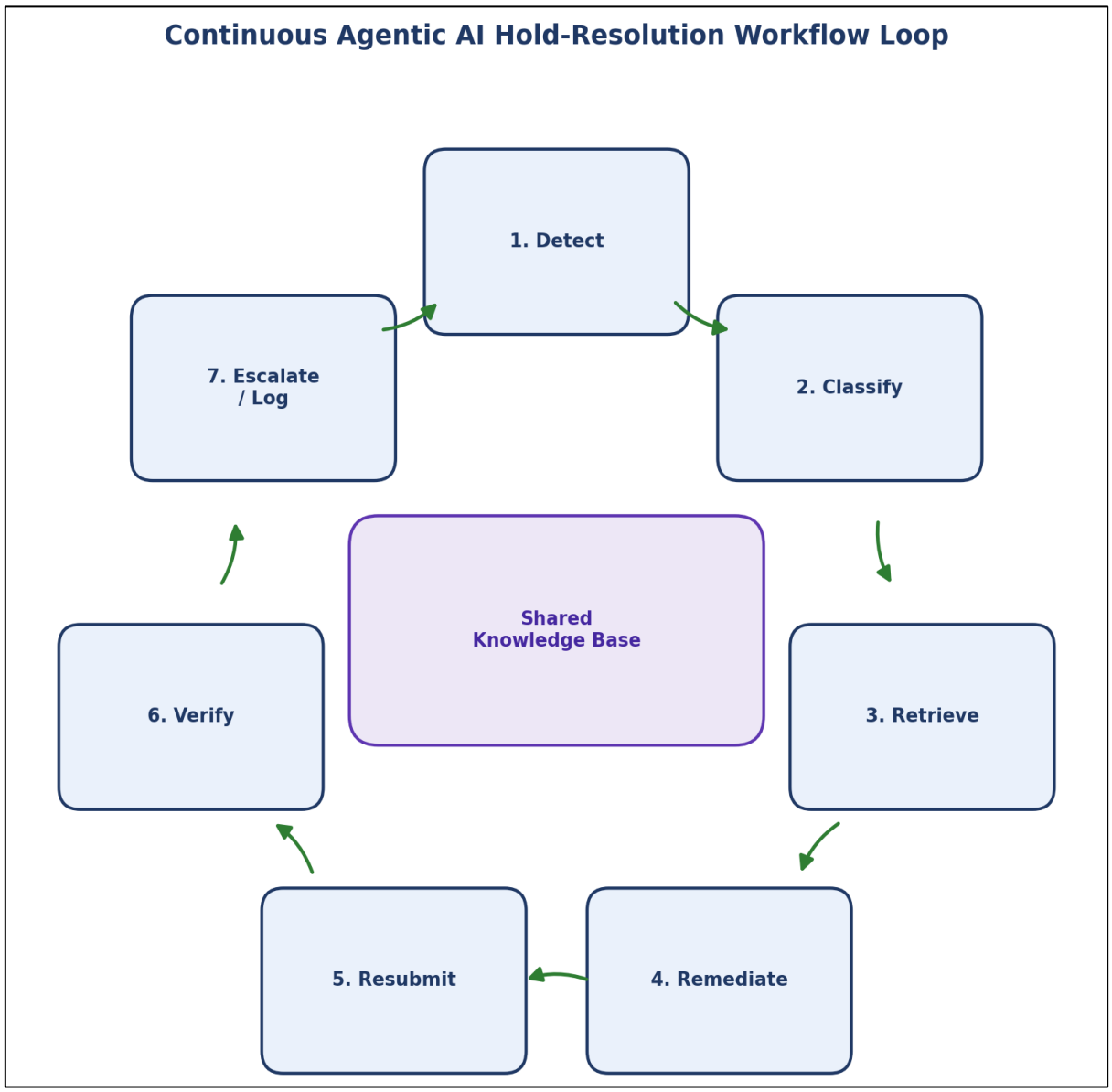

Fig 3 The Seven-Step Agentic Workflow Loop, Continuously Re-Entered as New Hold Events Arrive; Every Step Reads from and Writes to the Shared Knowledge Base.

➢ *Knowledge Base and Reasoning Layer*

The framework's reliability depends on a knowledge base that combines three elements: (a) deterministic master-data rules describing which fields are mandatory for an item, location, or interface alias to be valid; (b) a continuously updated library of historical hold cases and their confirmed fixes, used both to match new cases to known patterns and to retrain classification confidence over time; and (c) interface specifications (HL7 segment definitions, alias mapping tables) used to interpret interface-engine bounce-backs. An LLM-based reasoning component is used specifically where free-text error messages or ambiguous metadata require interpretation beyond a fixed rule set, while the actual system-changing action is always executed through a constrained, auditable action API rather than by the LLM directly manipulating production data.

➢ *Governance, Safety, and Human-in-the-Loop Controls*

Because the framework modifies master data and financial/inventory records in a clinical supply chain, governance controls are treated as a first-class design requirement rather than an afterthought:

- Scoped autonomy: the Remediation Agent is only permitted to act within a pre-approved catalog of fix types (e.g., add missing mapping, populate missing non-clinical attribute); anything outside that catalog is routed to a human.
- Confidence thresholds: the Diagnostic Agent's classification confidence score determines whether a fix is auto-applied, auto-applied with post-hoc human review, or escalated before any action is taken.
- Full audit trail: every diagnosis, retrieved value, applied change, and resubmission outcome is logged with a timestamp and linked to the responsible agent for compliance and regulatory review.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov552

- Segregation from clinical decision-making: the framework never alters what was prescribed, dispensed, or administered — it only corrects the administrative/configuration data that prevented an already-completed transaction from posting.
- Rollback capability: every automated configuration change is reversible, and the system maintains a pre-change snapshot to support rapid rollback if a fix is later found to be incorrect.

The three execution modes of Section 5.5.4 are made concrete by assigning every fix type in the catalog to an autonomy tier (Table 5). Assignment follows the levels-of-automation view that the permissible degree of autonomy should fall as the consequence and irreversibility of an error rise [22], and is a configurable policy, not a hard-coded rule. The criteria are reversibility, effect on stock quantity or financial valuation, effect on lot traceability, and proximity to clinical data.

Table 5 Autonomy Tiers for the Fix Catalog and the Conditions for Promoting a Category to a Higher Tier.

Tier	Representative fix types	Execution mode	Condition for promotion
A: Auto	Add an item-to-location mapping from a validated template; resubmit once the source error has cleared; re-send an interface message after transient unavailability	AUTO	Shadow-mode agreement and false-fix gates met (Table 12)
B: Auto with review	Populate a missing cost record from an approved source within a configured tolerance; correct an interface alias to match a documented specification change	AUTO_REVIEW	Low false-fix rate on audit sampling; no category-attributable rollbacks for a defined period
C: Prepare and approve	Change a unit-of-measure conversion; split a multi-batch distribution; any fix that alters stock quantity or lot assignment; any change to a charge-master mapping	Agent prepares the fix; an analyst approves before it is applied	Not promotable without clinical-pharmacy and internal-audit sign-off
X: Always escalate	Any hold involving controlled substances or high-alert medications; anything touching what was prescribed, dispensed or administered; laboratory results; blood product issue, crossmatch or transfusion records; low-confidence or repeated failures	ESCALATED	Not promotable

Unit-of-measure and lot corrections do not change what was dispensed, but they change stock quantities and traceability, so they are treated as Tier C even though they are administrative. Under this policy Cases 6.3 and 6.4 are agent-prepared and human-approved, while Cases 6.1, 6.2 and 6.5 are eligible for automatic handling. Section 7.3 quantifies the capacity consequence of this stricter policy.

➢ *Formal System Model*

To move beyond a purely descriptive architecture, this section grounds the framework in two established bodies of computer-science theory: autonomic computing's MAPE-K control loop and the Belief-Desire-Intention (BDI) model of rational agency. Doing so gives the framework a precise, falsifiable structure that can be implemented, tested, and reasoned about formally, rather than only described in prose.

- *Mapping to the MAPE-K Autonomic Control Loop*

Kephart and Chess's autonomic computing reference model [11] defines a self-managing system as a closed control loop of four functions — Monitor, Analyze, Plan, Execute — operating over a shared Knowledge base (MAPE-K). The seven-step workflow in Section 5.2 is a domain-specialized instantiation of MAPE-K, as shown in Table 6. This mapping is significant because MAPE-K is a mature, widely studied reference architecture for self-healing IT systems [12, 13, 29]; expressing the framework in these terms means its properties (stability, convergence, oscillation risk) can be analyzed using the existing self-adaptive-systems literature rather than treated as novel unknowns.

Table 6 Mapping of the Proposed Workflow onto the MAPE-K Autonomic Control Loop [11].

MAPE-K Function	Agentic Workflow Step(s)	Role
Monitor	1. Detect	Monitoring Agent senses hold-queue state across all connected systems
Analyze	2. Classify	Diagnostic Agent interprets sensed data against the root-cause taxonomy and knowledge base
Plan	3. Retrieve, 4. Remediate (planning phase)	Remediation Agent selects the specific corrective action from the pre-approved fix catalog
Execute	4. Remediate (action), 5. Resubmit, 6. Verify	Remediation and Validation Agents apply the change and confirm system state converges
Knowledge	Continuous, all steps	Shared knowledge base: master-data rules, historical fix library, HL7/interface specifications

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov552

- *Agent Formalism (BDI)*

Each agent in Table 4 is modeled as a Belief-Desire-Intention agent following Rao and Georgeff's formalism [14], in which an agent's behavior is governed by three internal states: Beliefs (B) — the agent's current view of the world, drawn from the knowledge base and live system state; Desires (D) — the goals the agent is trying to achieve; and Intentions (I) — the specific, committed course of action the agent is currently executing. For the Diagnostic Agent, for example: Beliefs = {error code, error text, transaction metadata, historical case library}; Desires = {correctly classify the hold event}; Intentions = {apply deterministic rule set; if inconclusive, invoke LLM-based reasoning over free-text fields; return category and confidence score}. Representing agents this way keeps their internal reasoning inspectable and testable in isolation — each agent's belief set, goal, and committed action can be logged and unit-tested independently of the other agents in the pipeline, which materially aids both correctness verification and the audit requirements described in Section 5.4. Coordinating several autonomic, BDI-style agents into one coherent self-managing system, as proposed here, follows precedent established in early multi-agent autonomic computing prototypes such as IBM's Unity system [15], which similarly composed multiple self-managing agents under a shared coordination layer rather than relying on a single monolithic controller.

- *Hold Event Data Schema*

To keep the formalism in this section concrete and implementable rather than purely abstract, Table 7 defines the minimum field set every agent operates on. Every hold event normalized by the Monitoring Agent (Section 5.2, step 1) is represented as a single structured record with this schema, regardless of which source system it originated from; this is the common event format referenced throughout Section 5.2.

Table 7 Minimum Hold-Event Data Schema Shared Across all Five Agents and the Knowledge Base.

Field	Type	Description
event_id	string	Unique identifier for the hold event, generated at detection time
source_system	enum	Pharmacy | Inventory | Billing | ADS | IntegrationEngine — the system the hold originated in
facility_id	string	Facility / health-center identifier, for cross-facility aggregation and reporting
transaction_ref	string	Reference to the underlying dispense, requisition, distribution, or billing transaction
error_code	string | null	Structured error/reason code returned by the source system, where one exists
error_text	string | null	Free-text error message, used by the Diagnostic Agent's LLM-based reasoning path
category (c*)	enum | null	Root-cause category from Table 1, populated by the Diagnostic Agent
confidence (score)	float [0,1]	Diagnostic Agent's confidence in category, used in the decision logic below
proposed_fix	object | null	Structured description of the corrective action identified by the Remediation Agent
mode	enum	AUTO | AUTO_REVIEW | ESCALATED — set by the decision logic in Section 5.5.4
status	enum	OPEN | REMEDIATED | VERIFIED | ESCALATED | ROLLED_BACK
audit_trail	array	Ordered list of {agent, action, timestamp} entries for every step taken on this event

- *Classification and Remediation Decision Logic*

The Diagnostic Agent's classification and the Remediation Agent's auto-fix-versus-escalate decision are the two points in the pipeline with the highest safety consequence, so both are specified formally here rather than left implicit. Let e denote an incoming hold event (an instance of the schema in Table 7), $C = \{c1, \ldots, c5\}$ the five root-cause categories from Table 1, and $f(e, c)$ the confidence score in [0, 1] produced by the Diagnostic Agent for event e against category c. Let τ_{auto} and τ_{review} be configured confidence thresholds ($\tau_{auto} > \tau_{review}$). The decision logic is:

```
FUNCTION classify_and_route(e):
# best-matching category and its confidence score
c*, score = argmax_c f(e, c), max_c f(e, c)

IF score >= τauto AND c* ∈ approved_fix_catalog:
# Section 5.2 steps 3-6, fully automatic
action = remediate(e, c*)
log(e, c*, score, action, mode='AUTO')

ELSE IF score >= τreview AND c* ∈ approved_fix_catalog:
# apply fix, but flag for human review
action = remediate(e, c*)
log(e, c*, score, action, mode='AUTO_REVIEW')
notify_human_reviewer(e, c*, score, action)
```

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov552

```
ELSE:
# Escalation Agent, Section 5.4
escalate(e, c*, score)
log(e, c*, score, action=None, mode='ESCALATED')
RETURN
```

This decision rule is the formal counterpart of the confidence-threshold governance control described qualitatively in Section 5.4, and it is the exact function evaluated in the simulation study in Section VII: τauto and τreview are configuration parameters, not hard-coded constants, and are expected to be tuned per health system as the historical fix library (part of the shared knowledge base) grows and per-category confidence calibration improves. Membership of approved_fix_catalog in the rule above corresponds to the Tier A and Tier B fix types of Table 5; Tier C fixes are routed to ESCALATED with a prepared fix attached, and Tier X holds bypass confidence-based automation altogether.

➢ *Operating Across Vendors: Connectors, Correlation and Approval-Gated Reprocessing*

The agentic layer is vendor-neutral. A connector for each application and for the interface engine reads that system's hold and error queues and writes approved corrections only through the vendor's supported interface (an API, configuration utility, database view or flat file). Connectors translate vendor-specific error codes and identifiers into the canonical hold-event schema (Table 7), so the Diagnostic Agent reasons over one format whatever the source. Adding a vendor means adding a connector and a vendor-specific playbook to the knowledge base, not redesigning the agents.

Before classifying, the Diagnostic Agent correlates holds that share a business key (order number, transaction reference, item code, facility) across systems. Holds that trace to one upstream cause are handled as one case with one proposed set of corrections and one approval. Correlation also identifies which system owns the cause, which may differ from the system that displays the hold.

- Approval-gated reprocessing. Approval is built into the workflow, not added afterwards. For Tier B and Tier C fixes the Escalation Agent raises a single request containing the diagnosis, the evidence, the exact proposed change and the transactions affected, addressed to the owner of the domain concerned: for example the pharmacy lead for unit-of-measure changes, the revenue-cycle owner for charge-master changes, and the supply chain owner for cost and lots. The approval, the approver and the time are written to the audit trail. On approval the agents apply the change through the supported interface, reprocess the held transactions in dependency order, verify that every affected system is consistent and close the related tickets. If approval is declined or expires, nothing is applied and the case stays with a human. Because investigation and preparation are complete when the approver sees the request, human time is spent on the decision itself; this is the expected source of the speed-up, and its size is an empirical question for the pilot (Section 8.4).

Cross-vendor operation adds three guardrails. Agents use only vendor-supported interfaces and never modify vendor code or bypass vendor validation. Changes are tested in a vendor's non-production environment first where one exists, and the fix catalog is versioned against each vendor release. Clinical content, meaning laboratory results, specimen and blood product records and orders as written, is excluded from automation entirely (Tier X, Table 5).

➢ *AI Governance for the Agentic Layer*

Sections 5.4 and 5.6 introduced the autonomy tiers and the approval gate. This section extends the five controls of Section 5.4 into one governance model for the agentic layer (Table 8). The model follows the four functions of the NIST AI Risk Management Framework [27]: Govern (policy, roles and accountability), Map (the taxonomy, the operating context and the tier assigned to each fix type), Measure (the metrics and acceptance gates of Table 12 and the sensitivity analysis of Section 7.3) and Manage (approval, runtime safeguards, rollback and incident response).

Accountability sits with a governance committee that includes clinical informatics, pharmacy, supply chain, finance or revenue cycle, integration and IT operations, compliance or internal audit, and the vendor liaison. Each agent and each fix type has a named owner, and each fix type has a named approver role. No agent may approve its own proposal, and the person who approves a fix is not the person who configured the agent (segregation of duties).

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov552

Table 8 AI-Governance Model for the Agentic Layer.

Governance element	How the agent acts	How it is controlled	How it is guided and improved
Authority and scope	Acts only on catalogued fix types, in the tier assigned to each	Tier policy (Table 5); least-privilege service accounts per connector; allow-list of permitted operations; no access to clinical content	Promotion or demotion between tiers only through the gates of Table 12, decided by the committee
Decision confidence	Scores each classification; acts only above the category threshold, otherwise escalates	Thresholds set and reviewed per category; repeated failure or conflicting evidence forces escalation	Monthly calibration review of confidence against confirmed outcomes
Approval	Sends one complete request (diagnosis, evidence, exact change, affected transactions) and waits	Domain-owner approval for Tier B review and Tier C; time-limited; agent cannot self-approve; decline or expiry means no change	Approval analytics (latency, decline reasons, overrides) feed playbook and threshold changes
Safe execution	Applies the change only through the vendor-supported interface and reprocesses in dependency order	Pre-checks, idempotent and rate-limited operations, post-fix verification, snapshot-based rollback (Section 5.4) and a pause switch per agent, category or vendor	Every rollback is root-caused and fed back to the fix catalog
Transparency and audit	Records evidence, diagnosis summary, action, approver, outcome and time for every case	Tamper-evident log retained under organizational policy and open to internal audit	Periodic audit sampling and blinded re-review of automatic and approved fixes
Monitoring and drift	Reports auto-resolve share, escalation load, false-fix rate, rollback rate and approval latency	A gate breach, anomaly or correlated burst demotes the affected category to recommendation-only and alerts the owner; a vendor release triggers playbook review	Live view of auto-resolve share against the break-even of Section 7.3
Learning	Adds confirmed fixes and rejected proposals, with reasons, to the historical fix library	Only reviewed cases enter the library, which is versioned against each vendor release	Quarterly review of catalog, tiers and thresholds, and after each major vendor upgrade
Data protection	Reads the minimum data needed for diagnosis	Minimised or de-identified hold records where possible; restricted model access; no clinical content sent to external services without agreement (Section 8.2)	Included in the security and privacy review cycle

- Guiding the agents toward best use. Governance also directs the agents to the work where they add most value. Queues are worked by age, volume, financial or stock impact and proximity to patient care, so the oldest and most consequential holds are handled first and anything near clinical content goes to a human. Related holds are bundled so that approvers receive one request per root cause, not one per hold. The learning loop lets approved and rejected proposals alike improve the fix library and thresholds, so the auto-resolve share can rise safely over time. Categories are promoted from shadow mode to Tier A and Tier B only as measured agreement and false-fix rates allow, and are demoted when they deteriorate.
- Operating cadence. A proposed cadence is a daily review of escalated and declined cases by the operations owner, a weekly review of approval latency and queue ageing, a monthly committee review of the Table 12 measures and any tier changes, and an immediate review after each vendor upgrade, incident or rollback. These intervals are proposals for the organization to set.
- Behaviour under uncertainty. The default is to stop and escalate, not to act. If a connector fails, evidence conflicts, a gate is breached or a correlated burst is detected, the layer falls back to recommendation-only mode for the affected scope, so the worst case is the current manual process and not an unsupervised change.

V. CASE STUDIES

The following representative scenarios, drawn from the failure taxonomy in Section IV, illustrate the end-to-end agentic workflow.

➢ *Case: Item Not Mapped to Dispensing Location*

A pharmacist dispenses a medication and the patient is billed and physically receives the drug, but the inventory system holds the transaction because the medication has not been mapped to that pharmacy location. The Monitoring Agent detects the hold; the Diagnostic Agent classifies it as a master-data mapping gap based on the specific error code returned by the inventory system; the Remediation Agent adds the item-to-location mapping using the item's existing formulary attributes as a template from a comparable location; and the Validation Agent resubmits the transaction, which then clears and posts the stock deduction correctly.

Table 9 traces this case through the hold-event schema (Table 7) and the seven-step workflow (Section 5.2), showing representative field values at each stage to make the abstract workflow concrete.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov552

Table 9 Worked Example: Field-Level Trace of Case 6.1 Through the Hold-Event Schema and Workflow Loop.

Workflow Step	Agent / Field(s) Updated	Representative Value
1. Detect	Monitoring Agent populates event_id, source_system, facility_id, transaction_ref, error_code	source_system=Inventory; error_code=ERR_ITEM_LOC_UNMAPPED; status=OPEN
2. Classify	Diagnostic Agent populates category, confidence	category=Master-data / build gaps; confidence=0.97 (deterministic rule match on error_code, → τauto)
3. Retrieve	Remediation Agent queries item master and a comparable location's mapping	candidate mapping = {item_id, target_location_id, uom, formulary_attrs} copied from nearest matching location
4. Remediate	Remediation Agent writes proposed_fix and applies it via inventory system's configuration API	proposed_fix={action:'add_item_location_map', item_id, location_id}; mode=AUTO
5. Resubmit	Validation Agent re-triggers the original transaction_ref through the normal pipeline	transaction_ref resubmitted; inventory system re-evaluates stock deduction
6. Verify	Validation Agent confirms status transition and cross-system consistency	status: OPEN → REMEDIATED → VERIFIED; stock, billing, and pharmacy records now consistent
7. Log	All agents append to audit_trail	audit_trail = [{Monitoring,detect,t0}, {Diagnostic,classify,t1}, {Remediation,fix,t2}, {Validation,verify,t3}]

➢ *Case: Missing Cost on a Newly Built Medication*

A newly added medication is dispensable but has no populated cost record, causing every transaction against it to hold at the inventory valuation step. The framework retrieves a reference cost (e.g., from the supplying wholesaler feed or a comparable pack-size record), flags it for automatic population subject to a configured tolerance (Tier B: applied with post-hoc review, and escalated if the reference cost is outside tolerance), and resubmits affected transactions in a batch once the cost is confirmed.

➢ *Case: Fractional Dispensing Causing Negative Stock*

A medication dispensed as a partial unit relative to its stocking unit of issue produces a fractional or negative on-hand quantity that the system rejects. The Remediation Agent identifies the correct unit-of-measure conversion factor from the item master and prepares the corrected dispense-unit configuration. Because this changes stock quantities, it is a Tier C fix (Table 5) and is applied only after analyst approval, after which the transaction is resubmitted and fractional quantities post correctly.

➢ *Case: Multi-Batch Distribution Conflict*

A requisition is fulfilled from two lots of the same medication in a single distribution transaction, which the supply-chain module does not support natively and therefore holds. The Remediation Agent prepares a split of the single distribution into batch-specific sub-transactions consistent with the actual lots dispensed. Because lot assignment affects traceability, this is a Tier C fix applied after analyst approval, after which each sub-transaction is resubmitted, preserving lot traceability while clearing the hold.

➢ *Case: HL7 Interface Alias Bounce-Back*

An outbound HL7 message from the pharmacy system to the automated drug dispensing system bounces at the interface engine because the configured alias does not match a recently changed field format on the receiving side. The Monitoring Agent detects the message sitting in the interface engine's error queue; the Diagnostic Agent parses the interface engine's error text to identify the specific segment/field mismatch; the Remediation Agent corrects the alias mapping (Tier B when the change matches a documented specification update, otherwise escalated); and the Validation Agent re-sends the queued message, confirming successful inbound acknowledgment.

➢ *Case: Cross-Vendor Hold with Approval-Gated Reprocessing*

A new medication is built in the order-entry and pharmacy systems and released for use, but the build is not completed in the separate inventory and billing systems. A pharmacist dispenses it. Within minutes the inventory vendor's system holds the stock deduction (item not mapped to the location) and the billing vendor's system holds the charge (charge code not mapped). Each team sees one hold and, without correlation, would investigate separately. With the agentic layer:

- Detect and correlate. The Monitoring Agent reads both queues through their connectors; the Diagnostic Agent links the holds by item code and transaction reference and classifies them as one master-data gap caused by incomplete build propagation.
- Prepare. The Remediation Agent prepares the item-to-location mapping (Tier A, applied automatically) and a charge-code mapping drawn from the item's attributes and a comparable item (a charge-master change, Tier C).
- Approve. The Escalation Agent sends the revenue-cycle owner one request containing the evidence, the exact mapping and the affected transactions. The owner approves, and the approval is logged.
- Reprocess and verify. The agents apply the mapping, resubmit the inventory and billing holds in dependency order, and the Validation Agent confirms that stock, billing and dispensing records agree.
- Learn. The case and its confirmed fix join the historical fix library, and a preventive alert goes to the build team so that future items are completed in all three systems.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov552

The same pattern applies to a laboratory reagent or a blood bank product whose stock or charge posting is held because the item or product code is unmapped in the inventory or billing system. After approval the agent may correct only the administrative mapping and reprocess the posting. Anything touching clinical records stays Tier X and is escalated to the responsible clinical-system owner with the diagnostic context.

VI. SIMULATION-BASED EVALUATION

Because no production deployment of this framework yet exists, this section reports an illustrative, discrete-event simulation used to sanity-check the framework's expected operational effect and to make its claims falsifiable rather than purely qualitative. The simulation is a transparent, assumption-driven model — not empirical data from a live health system — and every parameter is stated explicitly below so the model can be reproduced, challenged, or recalibrated with real data once a pilot is available (see Section XI).

➢ *Simulation Methodology and Assumptions*

The model simulates 30 operating days across a large healthcare organization consistent with the scale described in Section 2.2 (spanning multiple hospital facilities and health centers). Hold-transaction arrivals are drawn from a Poisson process with a mean of 500 new hold events per day system-wide, split across the five root-cause categories of Table 1 using the mix shown in Table 10, which was set proportionally to the emphasis given to each failure mode in the motivating problem description (master-data and interface errors assumed most frequent, multi-batch conflicts least frequent).

Table 10 Simulation Input Parameters (Illustrative, Expert-Judgment Based — Not Measured from a Live System).

Category	Share of Holds	Manual Mean Resolution Time	Assumed Agentic Auto-Resolve Rate
Master-data / build gaps	30%	35 min	90%
Missing financial attributes	15%	25 min	85%
Fractional / negative stock	20%	40 min	80%
Multi-batch / lot conflicts	10%	55 min	65%
Interface / interoperability errors	25%	60 min	78%

Two scenarios are compared. In the manual baseline, a support team with a fixed daily throughput of 400 resolved holds/day (reflecting the cumbersome, one-by-one manual triage described in Section 2.2) processes the queue first-in-first-out. In the agentic scenario, the classification-and-routing logic of Section 5.5.4 is applied to every arriving event: events classified above τ_{auto} are auto-resolved at a mean of 3.5 minutes each (reflecting Section 5.2 steps 1–6 executed by software), and the remainder are escalated to the same support team, whose effective capacity for escalations alone is modeled at 100 cases/day — a conservative fraction of their original 400/day capacity, reflecting that most of the team's time is freed for higher-value work rather than assuming full reallocation. Both scenarios were implemented as a day-stepped queueing simulation in Python (NumPy; 20 independent replications with fixed seeds 0 to 19, results reported as means with min-max ranges across replications, following standard discrete-event simulation practice [28]) and are described precisely enough here to be re-implemented independently. The base case treats every category as auto-eligible at the Table 10 rates, which is the optimistic design; Section 7.3 evaluates the stricter tiered policy of Table 5. Mean handling time is hands-on resolution time and excludes queue waiting, which understates the latency advantage over a manual queue that is growing.

➢ *Simulated Results*

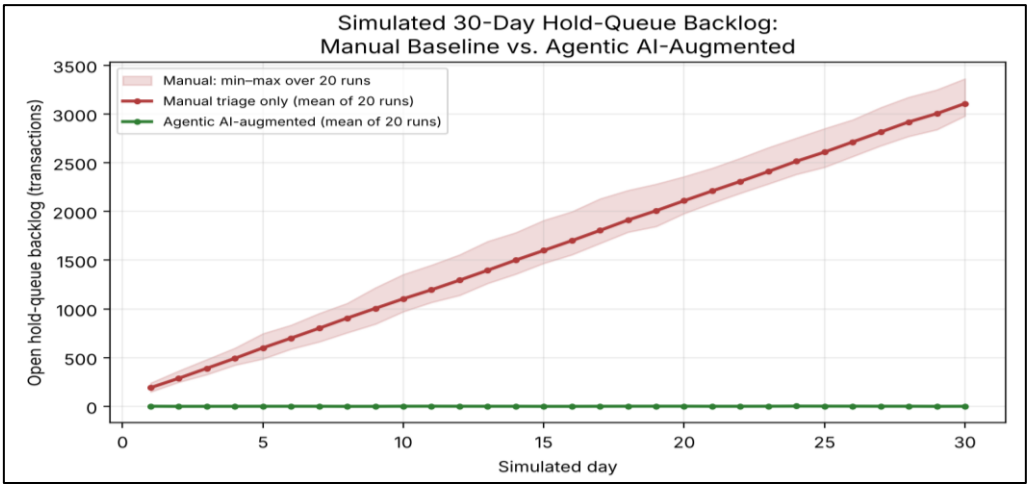

Fig 4 Simulated Open Hold-Queue Backlog Over 30 Days (20 Replications). The Manual Baseline Backlog Grows Roughly Linearly Because Arrivals (About 500 Per Day) Persistently Exceed Manual Throughput (400 Per Day); the Agentic Escalation-Only Queue Stays Near Zero while the Auto-Resolve Share Exceeds the Break-Even Level Derived in Section 7.3.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov552

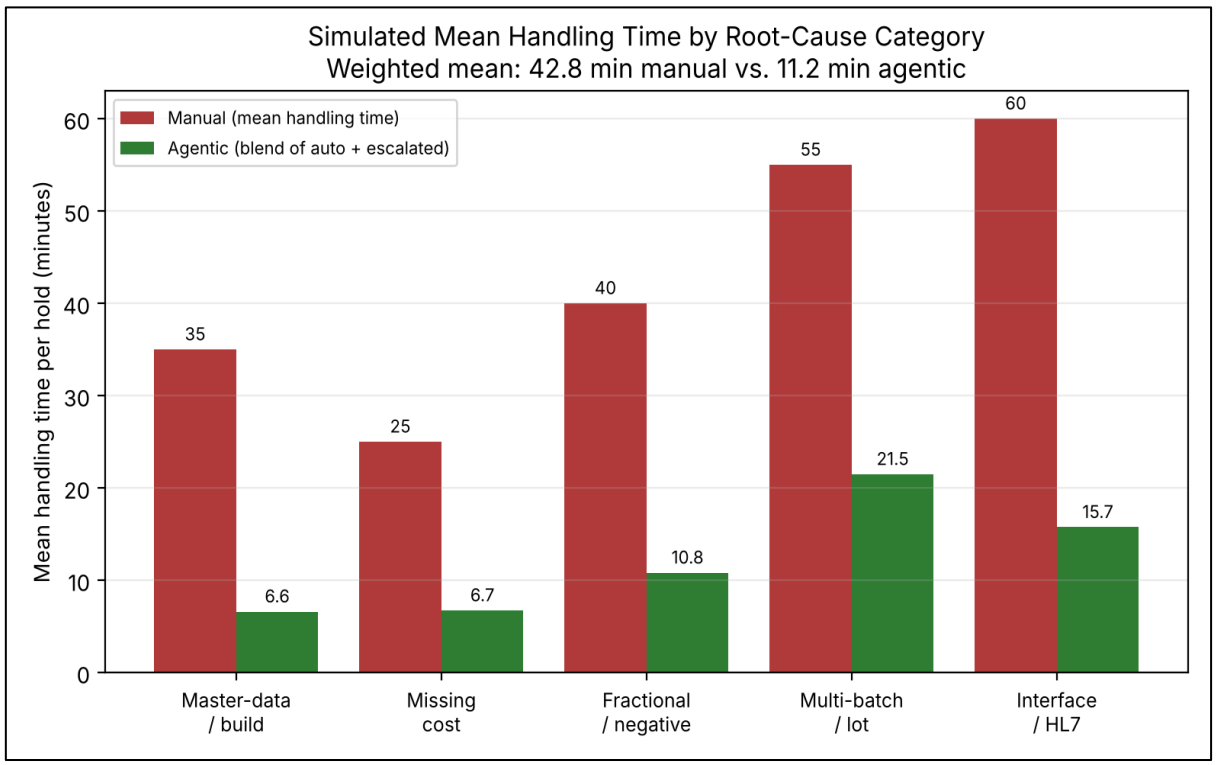

Fig 5 Simulated Mean Handling Time by Root-Cause Category (Mean of 20 Replications). Agentic Values Blend Fast Automatic Resolutions with Slower Human-Handled Escalations; Handling Time Excludes Queue Waiting.

Across the 30 simulated days and 20 replications, the manual baseline backlog grew from an initial 85 open transactions to a mean of 3,108 by day 30 (range 2,980 to 3,362), consistent with the accumulating discrepancy problem described in Section 2.2. The agentic scenario escalation queue stayed near zero (mean 0.8 open at day 30; mean daily peak 10, highest single-run peak 20), because about 82% of arriving holds were classified above τ_{auto} and resolved automatically (on average 410 auto-resolved and 91 escalated per day). Weighted across the category mix in Table 10, mean handling time fell from approximately 43 minutes under manual triage to approximately 11 minutes under the agentic scenario, a simulated speed-up of roughly 3.8x; category-level speed-ups ranged from 2.6x (multi-batch conflicts) to 5.3x (master-data gaps).

➢ *Sensitivity and Stress Analysis*

The central question is how much the results depend on the assumed auto-resolve rates in Table 10. Table 11 and Figure 6 scale all category auto-resolve rates down (keeping arrivals, manual times and escalation capacity fixed) and add two stress scenarios: correlated bursts and the stricter tiered policy of Table 5. All values are means over 20 replications.

Table 11 Sensitivity and Stress Results (20 Replications Per Scenario; Capacity of 100 Escalations Per Day; 500 Holds Per Day).

Scenario	Auto-resolve share	Escalations per day	Mean handling time (min)	Escalation backlog at day 30
Base case (Table 10 rates)	81.9%	91	11.2	1
Auto-resolve at 95% of assumed rates	77.8%	111	12.8	337
Auto-resolve at 90% of assumed rates	73.7%	132	14.4	954
Auto-resolve at 85% of assumed rates	69.6%	152	16.0	1,572
Auto-resolve at 70% of assumed rates	57.3%	214	20.7	3,412
Auto-resolve at 55% of assumed rates	45.0%	275	25.4	5,256
Correlated bursts: 3x volume on days 10 and 20, one category, confidence halved, batch auto-escalated	77.5%	128	12.4	919 (mean peak 1,005)
Tiered policy of Table 5 (Tier C approval at 8 min each; capacity held equivalent)	59.2%	53 + 150 approvals	10.4	0 (human workload 85% of capacity)

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov552

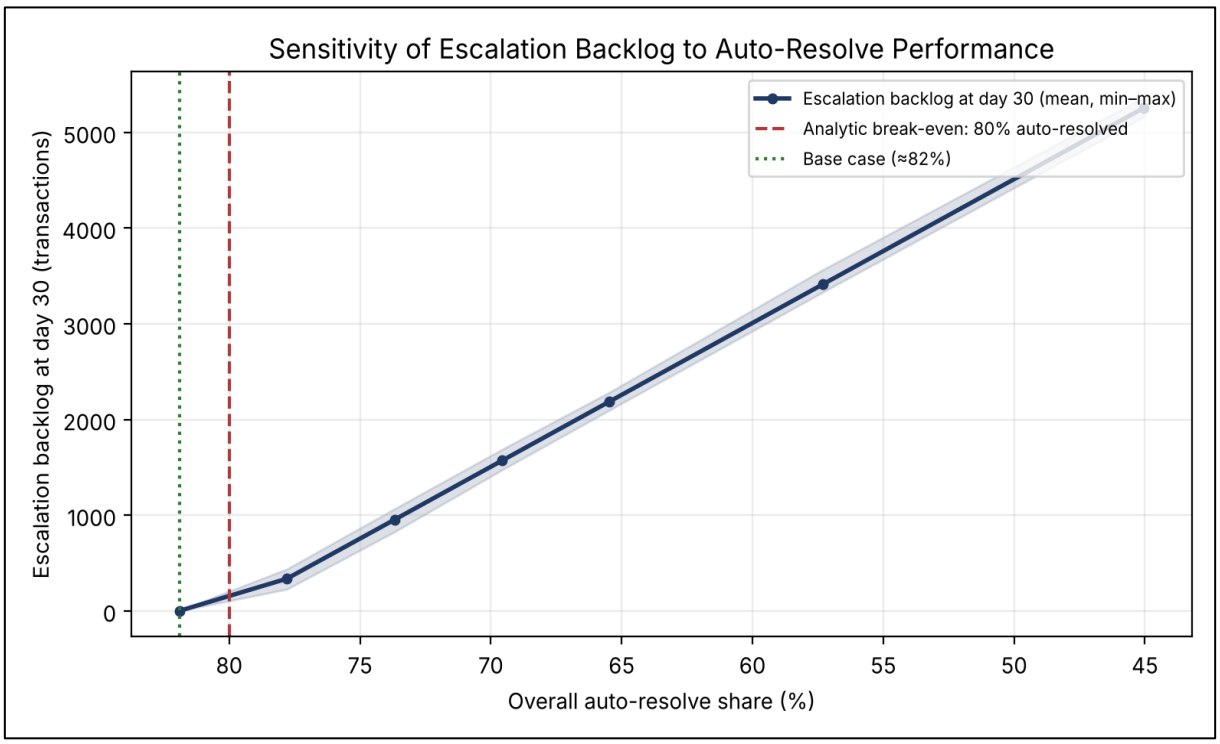

Fig 6 Escalation Backlog at Day 30 as a Function of the Overall Auto-Resolve Share (Mean and Min-Max Over 20 Replications). The Queue is Bounded Only Above the Analytic Break-Even of 80%.

- Break-even. With 500 holds arriving per day and capacity for 100 escalations per day, the escalation queue is bounded only if the overall auto-resolve share is at least 1 - 100/500 = 80%. The base case (82%) sits just above this threshold. This is the most important result of the sensitivity analysis: the backlog benefit is not robust to modest over-estimation. A 5% relative shortfall in auto-resolve performance already lets the queue grow to roughly 337 transactions in 30 days, and a 15% shortfall to roughly 1,572. Mean handling time still falls by 1.7x to 3.8x across the range tested, so the binding constraint is queue stability and not per-case speed.
- Incorrect automatic fixes. If a fraction of automatically applied fixes were wrong and undetected until audit, the base case (about 410 automatic fixes per day) would produce about 4 incorrect fixes per day at a 1% error rate, 8 at 2% and 20 at 5%. This is why Section 5.4 requires audit sampling and rollback, and why the pilot gates in Section 8.4 are set on the measured false-fix rate and not on the auto-resolve rate alone.
- Correlated bursts. When a single upstream cause produces a burst of same-category holds and the whole batch is escalated regardless of individual confidence, as required by the batch-level control in Table 13, the escalation backlog peaks at about 1,005 and about 919 holds remain open at day 30. The control trades speed for safety, and a deployment must provision surge capacity or a documented triage rule for such events.
- Tiered governance. Treating fractional-stock and multi-batch holds as Tier C lowers the fully automatic share to about 59% and creates about 150 approval tasks per day. If an approval takes about 8 minutes because the agent has already prepared the fix, and human capacity is held equivalent to 100 manual escalations per day, the queue remains bounded at about 85% utilisation (mean handling time 10.4 minutes). The result depends on the assumed approval time and shows that the stricter policy is feasible only if prepared fixes make approval materially cheaper than manual resolution; the pilot must measure that time.

➢ *Limitations and Threats to Validity*

These results should be read as a directional plausibility check, not a validated performance claim. Three limitations are important. First, the auto-resolve rates in Table 10 are expert-judgment estimates, not measurements; real Diagnostic Agent accuracy will depend on the quality and volume of the historical fix library available at deployment and should be expected to start lower and improve over time as described in Section 5.5.4. Second, the simulation assumes hold events are independent and identically distributed within a category; in practice, a single upstream cause (e.g., a bad formulary load) can produce correlated bursts of same-category holds, which would stress-test the Remediation Agent's throughput differently than modeled here. Third, the simulation does not model remediation errors (an incorrect automated fix) or their downstream cost, which Section X addresses qualitatively via failure-mode analysis. The error-rate sensitivity in Section 7.3 is a fixed-rate illustration, not a mechanistic model of the downstream cost of an incorrect fix. A pilot deployment against a real, de-identified hold-transaction dataset, as proposed in Section XI, is required before any of these figures should be treated as validated.

- Threats to validity. Construct validity: handling time excludes queue waiting, and "auto-resolve" is an assumed rate, not a measured classification accuracy. Internal validity: arrivals are Poisson and category outcomes independent, so correlation beyond the burst scenario is

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov552

not captured. External validity: the parameters describe a stylised organization of the scale in Section 2.2 and are not calibrated to any specific vendor configuration. Mitigations: every parameter is stated, the model is re-implementable, results are replicated across 20 seeds, and Section 7.3 reports where the conclusions change. The figures should therefore be read as conditions to be tested, not as predictions.

VII. IMPLEMENTATION CONSIDERATIONS

➢ *Integration Approach*

The framework is designed to integrate with existing systems through their existing supported interfaces — REST/SOAP APIs, database views, flat-file exports, or message-queue subscriptions — rather than requiring replacement of any pharmacy, inventory, billing, ADS, or interface-engine platform. This allows phased deployment: monitoring and diagnostic capability can be introduced first in a read-only, recommendation-only mode, with automated remediation enabled only for the highest-confidence, lowest-risk fix categories, and expanded over time as the historical case library grows.

➢ *Security and Compliance*

Because the framework operates on protected health information-adjacent financial and inventory data, it must operate within the health system's existing identity and access management controls, use service accounts with the minimum privilege necessary for each supported fix type, and log all activity to the organization's existing security information and event management (SIEM) and compliance audit systems.

➢ *Change Management*

Successful adoption requires close collaboration with the application support and interface teams whose work is being augmented. Framing the system as removing repetitive triage work rather than replacing staff, and involving these teams in defining the initial fix catalog and confidence thresholds, is important both for accuracy and for organizational acceptance.

➢ *Pilot Validation Protocol*

The simulation identifies what must be measured; the pilot determines whether the conditions hold. The protocol is staged so that no autonomy is enabled before it has been earned on local data. Phase 0 runs the Monitoring and Diagnostic Agents in recommendation-only (shadow) mode for a minimum observation window on a de-identified hold-transaction sample, comparing every proposed classification and fix with the analyst's actual resolution. Phase 1 enables Tier A fixes for categories that pass their gates. Phase 2 enables Tier B with audit sampling. Tier C remains human-approved, and Tier X is never automated. Rollout across facilities should use a stepped-wedge design so that facilities act as their own controls over time. Table 12 lists the measures and illustrative starting gates; the thresholds are proposals to be set with clinical informatics, pharmacy leadership and internal audit, not validated values.

Table 12 Pilot Measures and Illustrative Acceptance Gates.

Measure	Definition	Illustrative starting gate
Classification precision and recall	Per-category agreement of the Diagnostic Agent with the confirmed root cause	Precision of at least 95% in a category before any automatic mode
Fix agreement rate	Share of proposed fixes equivalent in effect to the analyst's fix in shadow mode	At least 98% for Tier A fix types
False-fix rate	Confirmed incorrect fixes per automatically applied fix, from audit sampling	Upper 95% confidence bound of 0.5% or less
Auto-resolve share and escalation load	Overall auto-resolve share and escalations per day against analyst capacity	Sustained at or above the break-even share of Section 7.3 and within capacity
Time to resolution	Median and 90th percentile including queue waiting, against the pre-pilot period	Statistically and operationally meaningful reduction
Rollback rate	Automated changes reversed per 1,000 and time to detect	No upward trend; each rollback root-caused
Inventory variance	Stock discrepancies and open holds at period close against historical baseline	Non-inferior to baseline, with reduction as the target
Override and sampling outcomes	Reviewer override rate on AUTO_REVIEW fixes; findings from blinded re-review of AUTO fixes	Override and error trends reviewed monthly
Approval latency	Time from approval request to decision, by tier and approver role, and from decision to verified reprocessing	Reported separately by tier; target set by the governance committee

Any breach of a gate returns the affected category to the previous mode. The monitoring dashboard should display the live auto-resolve share against the break-even of Section 7.3 so that deteriorating performance is visible before a backlog forms.

➢ *Multi-Vendor Operating Model*

Running the layer across several vendors needs an explicit operating model: (i) an ownership matrix naming, for each application domain and fix type, who approves, who is notified and who is the escalation owner, including the vendor support contact; (ii) confirmation that each vendor's support and licence terms permit the interfaces and

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov552

configuration changes the agents use; (iii) connector and playbook maintenance tied to each vendor's release calendar, because format or code changes after an upgrade are a recurring cause of holds; (iv) approval service levels, since approval latency becomes the dominant part of resolution time for Tier B and Tier C fixes and should be measured and reported (Table 12); and (v) a surge rule for correlated bursts (Section 7.3).

VIII. EXPECTED BENEFITS

At the scale of a large healthcare organization spanning multiple hospital facilities and health centers, the primary expected benefits of the proposed framework include:

- Reduced hold-resolution time: the simulation in Section VII suggests auto-resolvable holds could clear in minutes rather than during the next manual review cycle, with weighted mean handling time dropping roughly 3.8x (about 43 to 11 minutes) versus manual triage.
- Bounded backlog growth: where the manual-only baseline showed an unbounded, roughly linear backlog growth over 30 simulated days, the agentic scenario kept the human-facing escalation queue near zero, provided the auto-resolve share stays above the break-even of about 80% (Section 7.3) — directly addressing the accumulating stock-discrepancy problem described in Section 2.2.
- Improved inventory accuracy: faster clearance of holds reduces the window during which physical and system stock diverge, improving replenishment and reconciliation accuracy across all facilities.
- Reduced manual workload: application and integration support staff spend proportionally less time on repetitive, well-understood hold categories (an estimated 80–90% of volume in the highest-confidence categories, per Table 10) and more time on genuinely novel or high-risk exceptions.
- Cross-vendor visibility and faster approved reprocessing: a single view of held transactions across vendor boundaries, correlation of related holds to one cause, and approval requests that arrive complete, so that reprocessing follows immediately once the approver decides. These effects are expected, not yet measured; the approval-latency and time-to-resolution measures in Table 12 are designed to test them.
- Consistency: automated remediation applies the same corrective logic every time, reducing the variability introduced when different analysts resolve similar holds differently.
- Auditability: a complete, timestamped log of every automated diagnosis and fix (Section 5.5.4) provides stronger traceability than ad hoc manual resolution notes.

These figures are simulation-derived and directional, not measured outcomes; Section 7.4 and Section X set out the corresponding limitations and risks that a real deployment must validate against.

IX. CRITICAL ANALYSIS OF RISKS AND FAILURE MODES

Beyond the general governance controls in Section 5.4, this section applies a failure-mode-and-effects analysis (FMEA) [17, 30] — a structured risk-assessment method long established in safety-critical engineering — to the framework itself, since a system that autonomously modifies inventory, cost, and interface configuration in a hospital supply chain must be evaluated on how it can fail, not only on how it is intended to work. Table 13 examines five specific failure modes, their potential effect if unmitigated, a qualitative likelihood/severity rating, and the corresponding design mitigation already present in the architecture.

Table 13 Failure-Mode Analysis of the Proposed Agentic AI Framework.

Failure Mode	Potential Effect if Unmitigated	Likelihood / Severity	Mitigation in This Design
Diagnostic Agent misclassifies a hold with high confidence	Wrong fix applied automatically; may mask the true root cause or corrupt master data	Low / High	τauto set conservatively per category (Section 5.5.4); full rollback snapshot (Section 5.4); post-hoc audit sampling of AUTO-mode fixes
Remediation Agent applies a fix outside its intended scope (scope creep)	Unauthorized or unreviewed change to production master data	Low / High	Scoped autonomy restricts the Remediation Agent to a pre-approved fix catalog (Section 5.4); any action outside the catalog is hard-blocked, not just discouraged
Correlated burst of same-category holds (e.g., one bad formulary load)	Remediation Agent throughput saturated; automated fixes applied faster than a human could sanity-check the batch	Medium / Medium	Rate-limiting and batch-level anomaly detection on the Monitoring Agent; abnormal volume in one category auto-escalates the whole batch regardless of individual confidence scores
Knowledge base drifts out of date (e.g., vendor changes an HL7 field format)	Diagnostic Agent confidently applies a now-incorrect historical fix pattern	Medium / Medium	Validation Agent's post-fix verification step (Section 5.2, step 6) catches fixes that do not actually clear the hold and routes repeat failures to escalation, not repeated auto-retry

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov552

Over-reliance / deskilling of support staff over time	Erosion of institutional troubleshooting expertise needed for the genuinely novel cases still requiring humans	Medium / Low-Medium	Escalation Agent routes a deliberately non-zero share of cases to humans with full diagnostic context (Section 5.4), preserving exposure to real cases rather than eliminating it entirely
Auto-resolve performance falls below the break-even share (about 80% at 500 holds and 100 escalations per day)	Escalation queue grows without bound and holds age; the backlog benefit is lost even though handling time per case still falls	Medium / Medium	Pilot gate on measured auto-resolve share and escalation load (Section 8.4); staged enablement by tier; live share-versus-break-even monitoring with fallback to recommendation-only mode

➢ *Additional Organizational and Regulatory Risks*

- Data quality of the knowledge base: the framework's diagnostic accuracy depends on the completeness and correctness of the historical case library and master-data rules used to train and constrain the Diagnostic Agent.
- Interface heterogeneity: HL7 alias and mapping conventions vary across vendors and even across facilities within the same health system, requiring the framework's interface knowledge base to be maintained per facility/interface pair.
- Change control and regulatory considerations: automated modification of inventory and financial records may fall within the scope of existing change-control and internal-audit policies and should be reviewed with compliance stakeholders before deployment.
- Organizational adoption: as with any automation initiative affecting existing staff roles, success depends on transparent communication about scope, safeguards, and the continued role of human analysts for escalated cases.
- Automation bias and complacency: reviewers of AUTO_REVIEW fixes may approve without scrutiny, a recognised failure pattern in human-automation interaction [23]. Mitigations are random blinded re-review of AUTO-mode fixes, reporting of override rates and rotation of review duties (Table 12).
- Cross-vendor change control and support boundaries: automated configuration changes in a vendor system may fall under that vendor's support and licence terms and the organization's change-control process. Agents must use supported interfaces only, and the approval workflow must satisfy change control (Section 8.5).
- Approval latency and approver fatigue: slow, numerous or poorly evidenced requests erase the speed-up or lead to approvals without scrutiny. Mitigations are one complete request per root cause, approval service levels, delegation rules, and monitoring of approval time and override outcomes (Table 12).

X. CONCLUSION AND FUTURE WORK

Large, multi-facility health systems operating integrated pharmacy, inventory, billing, and automated dispensing platforms face a persistent, high-volume operational burden in manually resolving hold transactions caused by master-data gaps, missing financial attributes, fractional stock handling, multi-batch distribution conflicts, and HL7 interface errors. This paper has proposed an Agentic AI framework — comprising monitoring, diagnostic, remediation, validation, and escalation agents operating over a shared knowledge base, formally grounded in the MAPE-K autonomic control loop and BDI agent theory (Section 5.5) — that automates the diagnosis-and-fix workflow currently performed manually by application and integration support staff, while preserving human oversight for ambiguous or high-risk cases through confidence-based escalation and a fully auditable action trail. An illustrative, transparently-documented simulation (Section VII) indicates the direction and rough magnitude of benefit that such a system could plausibly deliver — materially bounded backlog growth and a 2.6x to 5.3x reduction in mean handling time by category, conditional on auto-resolve performance remaining above a break-even threshold of about 80% — while a failure-mode analysis (Section X) shows that its principal risks are addressable through scoped autonomy, confidence thresholds, and mandatory human review of anything outside a pre-approved fix catalog.

Agentic AI is still an actively developing field, and the specific engineering choices in this paper (confidence thresholds, fix catalogs, tier assignments, knowledge-base design) will need to evolve alongside it. The simulation and sensitivity results suggest, rather than demonstrate, that high-volume, pattern-recognisable, configuration-level failures are a promising candidate for governed automation. They also show that the decisive empirical questions are whether measured auto-resolve performance clears the break-even share, and whether measured false-fix rates clear the gates of Section 8.4. Health systems that begin instrumenting their hold queues and building the historical fix library described in Section 5.3 now will be positioned to test those questions on their own data, independent of any single vendor's roadmap.

Future work includes: (a) empirical evaluation of the framework in a staged pilot following the protocol of Section 8.4, against a defined, de-identified hold-transaction sample to measure resolution-time reduction and remediation accuracy against human-analyst baselines and replace the illustrative parameters of Section VII with measured ones; (b) extension of the root-cause taxonomy and fix catalog to additional supply-chain and revenue-cycle transaction types beyond pharmacy; (c) development of standardized confidence-calibration methods for the Diagnostic Agent across heterogeneous vendor systems, including formal precision/recall reporting per category; (d) a controlled A/B or stepped-wedge study comparing agentic-augmented and manual-only facilities within the same health system to isolate the framework's causal effect from other operational changes; (e) formal evaluation of governance and patient-safety controls in collaboration with clinical informatics and

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov552

compliance stakeholders prior to broader production rollout; and (f) extension of the connector layer and taxonomy to laboratory and blood bank administrative postings across additional vendors, with clinical sign-off.

REFERENCES

[1]. Syed, T. A., et al. (2025). Agentic AI Framework for Smart Inventory Replenishment. arXiv:2511.23366. https://arxiv.org/abs/2511.23366

[2]. Vu, H., Klievtsova, N., Leopold, H., Rinderle-Ma, S., & Kampik, T. (2025). Agentic Business Process Management: Practitioner Perspectives on Agent Governance in Business Processes. Responsible BPM 2025. arXiv:2504.03693. https://arxiv.org/abs/2504.03693

[3]. Calvanese, D., Casciani, A., De Giacomo, G., Dumas, M., Fournier, F., Kampik, T., et al. (2026). Agentic Business Process Management: A Research Manifesto. arXiv:2603.18916. https://arxiv.org/abs/2603.18916

[4]. Vatsal, S., Dubey, H., & Singh, A. (2026). Agentic AI in Healthcare & Medicine: A Seven-Dimensional Taxonomy for Empirical Evaluation of LLM-based Agents. IEEE Access, 14, 4840-4863. https://doi.org/10.1109/ACCESS.2026.3651218 (preprint arXiv:2602.04813).

[5]. Dhurwey, S. S., Kumar, P., Sendur, U., & Verma, A. AI-Driven Inventory Management System for Pharmacy Stores. Atlantis Press, Proceedings of ICAAAI-25. https://www.atlantis-press.com/proceedings/icaaai-25/126012591

[6]. Itechsmart Inc. iTechSmart HL7: Autonomous HL7 Interface Monitoring and Remediation [Product documentation]. https://itechsmart.dev/hl7

[7]. Use of Artificial Intelligence (AI) in Managing Inventory of Medicine in Pharmaceutical Industry. (2021). ResearchGate. https://www.researchgate.net/publication/348849015

[8]. Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach (4th ed.). Pearson.

[9]. Wooldridge, M. (2020). An Introduction to MultiAgent Systems (2nd ed.). Wiley.

[10]. Bluesight. (2026). 2026 Hospital Pharmacy Operations Report, cited in "Beyond the Hype: How AI is Being Used in Pharmacy," Becker's Hospital Review. https://www.beckershospitalreview.com/pharmacy/beyond-the-hype-how-ai-is-being-used-in-pharmacy-and-whats-coming-next/

[11]. Kephart, J. O., & Chess, D. M. (2003). The vision of autonomic computing. Computer, 36(1), 41–50. https://doi.org/10.1109/MC.2003.1160055

[12]. Mahdavi Hezavehi, S., Weyns, D., & Avgeriou, P. (2025). State of the Art on Self-adaptive Systems: An Essay. arXiv:2511.06352. https://arxiv.org/abs/2511.06352

[13]. A MAPE-K-Based Method for Architectural Conformance Checking in Self-Adaptive Systems. arXiv:2401.16382. https://arxiv.org/pdf/2401.16382

[14]. Rao, A. S., & Georgeff, M. P. (1995). BDI Agents: From Theory to Practice. Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), 312–319. The MIT Press.

[15]. Tesauro, G., Chess, D. M., Walsh, W. E., Das, R., Segal, A., Whalley, I., Kephart, J. O., & White, S. R. (2004). A Multi-Agent Systems Approach to Autonomic Computing. Proceedings of AAMAS 2004, 464–471.

[16]. Malik, A. (2026). Autonomous Incident Resolution at Hyperscale: An Agentic AI Architecture for Network Operations. arXiv:2606.09122. https://arxiv.org/pdf/2606.09122

[17]. McDermott, R. E., Mikulak, R. J., & Beauregard, M. R. (2009). The Basics of FMEA (2nd ed.). CRC Press. ISBN 9781563273773.

[18]. Health Level Seven International. HL7 Version 2 Product Suite [Standards documentation]. https://www.hl7.org

[19]. Samiyullaha, S. (2026). A Configurable, Mistake-Proofed Function for Adding and Removing Medications Across Hospital Pharmacy Inventory Locations. International Journal of Novel Research and Development (IJNRD), paper ID IJNRD2609140.

[20]. Samiyullaha, S. (2026). Persistent Stock Discrepancies in Hospital Pharmacy Supply Chains: Why Robotics, System Integration, and Artificial Intelligence Have Not Closed the Reconciliation Gap. International Journal of Scientific Research and Technology, 3(10), 80-87.

[21]. Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. MIS Quarterly, 28(1), 75-105.

[22]. Parasuraman, R., Sheridan, T. B., & Wickens, C. D. (2000). A model for types and levels of human interaction with automation. IEEE Transactions on Systems, Man, and Cybernetics - Part A, 30(3), 286-297.

[23]. Parasuraman, R., & Riley, V. (1997). Humans and automation: Use, misuse, disuse, abuse. Human Factors, 39(2), 230-253.

[24]. Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mane, D. (2016). Concrete Problems in AI Safety. arXiv:1606.06565.

[25]. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing Reasoning and Acting in Language Models. ICLR 2023. arXiv:2210.03629.

[26]. Sittig, D. F., & Singh, H. (2010). A new sociotechnical model for studying health information technology in complex adaptive healthcare systems. Quality and Safety in Health Care, 19(Suppl 3), i68-i74.

[27]. National Institute of Standards and Technology. (2023). Artificial Intelligence Risk Management Framework (AI RMF 1.0). NIST AI 100-1.

[28]. Law, A. M. (2015). Simulation Modeling and Analysis (5th ed.). McGraw-Hill Education.

[29]. [29] Weyns, D. (2020). An Introduction to Self-Adaptive Systems: A Contemporary Software Engineering Perspective. Wiley.

[30]. International Electrotechnical Commission. (2018). IEC 60812:2018 Failure modes and effects analysis (FMEA and FMECA).