Journal of Computer Science and Technology Studies
ISSN: 2709-104X
DOI: 10.32996/jcsts
Journal Homepage: www.al-kindipublisher.com/index.php/jcsts

Agentic Site Reliability Engineering for Self-Healing FHIR Workflow Engines

Ankita Banerjee

Independent Researcher, California, USA

Corresponding Author: Ankita Banerjee**, E-mail:** reachankitab@gmail.com

ABSTRACT

Healthcare FHIR workflow engines have complicated operational requirements, including high-volume, time-sensitive transactions and complete adherence to the HL7 FHIR Release 4 (R4) standard. Traditional Site Reliability Engineering (SRE) approaches rely on reactive runbooks and manual triage, which are inadequate for the speed and specificity of FHIR fault resolution. This article proposes the FHIR Agentic SRE for Engine Reliability (FASER) conceptual framework, a proposed architectural model for agentic SRE in FHIR workflow engines. The framework introduces a FHIR-specific fault taxonomy covering resource validation errors, transaction rollback failures, subscription notification failures, and consent enforcement errors. FASER employs a four-stage closed-loop operational cycle—Observe, Reason, Act, and Verify—grounded in FHIR-native telemetry and bounded-autonomy remediation. The Reason layer applies LLM-based fault classification with a confidence gating mechanism to constrain autonomous action and ensure human escalation for uncertain or high-risk fault classes. The Verify stage closes the reliability loop by confirming that remediation actions have successfully restored the expected operational state. This article presents the design and analytical justification of the FASER framework. Implementation and empirical validation are identified as future work.

KEYWORDS

FHIR workflow engines; agentic site reliability engineering; self-healing systems; autonomous fault remediation; healthcare cloud infrastructure

ARTICLE INFORMATION

ACCEPTED: 16 October 2026 **PUBLISHED:** 16 November 2026 **DOI:** 10.32996/jcsts.2026.8.17

1. Introduction

Healthcare cloud services face operational challenges that current traditional operational methods cannot address. Fast Healthcare Interoperability Resources (FHIR) workflow engines, which direct and coordinate authorization and notification steps in patient care and other business processes involving cloud-based applications, represent some of the most complex healthcare IT systems. FHIR workflow engines must support high transaction volumes at low latency while adhering to the HL7 FHIR Release 4 (R4) standard [21]. Disruptions to FHIR workflow engines can negatively affect IT system access and other aspects of patient care and compliance and undermine confidence in the healthcare interoperability infrastructure in which substantial healthcare IT investments have been made [1].

Site reliability engineering provides practices and strategies to improve system reliability at scale. Traditional SRE strategies tend to be more reactive. Strategies based on runbooks for responding to system problems depend on human intervention to respond to problems and may not meet FHIR transaction SLAs. Problems responding to member communication system outages in a health care network serving millions of members resulted in additional outages in other systems used for clinical decision support. Slow response to member communication outages is partly due to human limitations in responding quickly to outages in distributed FHIR environments [5, 6].

Agentic AI systems that integrate large language models with other technologies and memory capabilities to perform tasks may help address this challenge. Implementation examples in healthcare and cloud computing suggest advantages to using agentic AI for monitoring and remediation, including shorter average remediation times [8, 10]. Other implementation options to consider in healthcare FHIR use cases include strategies beyond deploying AIOps tools in healthcare settings. Specific errors and alerts common to FHIR, including invalid FHIR resources, failed transactions, invalid or unexpected CapabilityStatement data, and consent violations, have particular implications for healthcare that are not covered by AIOps remediation tools.

This article proposes the FHIR Agentic SRE for Engine Reliability (FASER) conceptual framework, which is designed to help workflow engines improve reliability by incorporating an Observe step to gather FHIR-standard information in real time, a Reason step for

Copyright: © 2026 the Author(s). This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) 4.0 license (https://creativecommons.org/licenses/by/4.0/). Published by Al-Kindi Centre for Research and Development, London, United Kingdom.

LLM-based fault classification and confidence-gated action selection, an Act step to execute bounded remediation playbooks, and a Verify step that confirms recovery before returning to the Observe phase. FASER is presented as a conceptual architectural framework. This study does not claim empirical performance results for FASER; instead, the framework is analytically examined using operational requirements derived from FHIR workflow characteristics and findings reported in the autonomous cloud operations literature. Implementation and experimental validation are identified as future work.

The primary focus of the article is on architecture. The article discusses a fault classification scheme related to HL7 FHIR, an observation reporting scheme for HL7 FHIR, a confidence-guided limited-action self-healing process guided by a playbook, and steps to prevent or mitigate human involvement after self-healing actions. The article provides references to research on AIOps and other self-healing tools to support its claims about architecture. References to research should not be interpreted as performance information about FASER. The article compares HL7 FHIR specification requirements and other behaviors with those proposed for FASER. The article characterizes FASER as an architecture. The article provides information other than empirical performance data about FASER, including information drawn from HL7 FHIR workflow characteristics and research. Additional development and testing of FASER are recommended.

The remainder of the paper examines other published research and the FASER model and framework. A framework analysis of the expected performance of the FASER model is also presented. The discussion also covers other implications for the development of healthcare platforms.

2. Literature Review

2.1 FHIR Workflow Engines in Cloud Healthcare Infrastructure

FHIR is the most widely used interoperability standard in healthcare and is increasingly being adopted for clinical research and other healthcare applications. A systematic review of the literature identified 998 records initially; following screening, 49 studies met inclusion criteria, of which 36 (73%) involved clinical research applications [1]. Workflow engines implementing FHIR-based standards help streamline care coordination and data exchange in healthcare IT systems [2, 3, 4, 15].

Operational challenges arise with workflow technologies that implement policies for distributed systems. Studies of HL7 FHIR implementations in health IT systems indicate that automation benefits care coordination, but a lack of IT connectivity and communication poses challenges for care coordination. Other studies of HL7 FHIR implementations in health IT systems have identified other challenges, including the need to verify standards documented in implementation guides used by workflow technologies to implement policies. Inadequate verification of these standards can lead to the exchange of improperly formatted FHIR resources between health IT systems and problems with care coordination [3, 4].

2.2 Site Reliability Engineering: From Runbook-Driven to Intelligent Operations

The Site Reliability Engineering field was created to adopt software engineering concepts and best practices, including error budgets and postmortems, in IT operations. Machine learning and other automated methods are common in Site Reliability Engineering, particularly in light of cloud computing and other trends that make it challenging for IT workers to maintain IT services [5]. Practices and trends in the growing field of Site Reliability Engineering, such as the adoption of policy-driven automation, reflect an emerging appreciation of the IT worker's limitations regarding the use of failure information.

Research on intelligent reliability engineering highlights efforts to automate failure detection and remediation processes. Benefits to cloud users of intelligent reliability engineering include faster failure detection and remediation, including automatic remediation of some common types of failures [6]. Drawbacks of existing SRE automation tools include limited use cases. Existing SRE automation tools can detect and remedy some common types of failures based on general cloud metrics, such as CPU use and request latency, but they cannot properly detect and analyze anomalies involving specific FHIR resources, such as AuditEvent and CapabilityStatement.

2.3 Agentic AI in Cloud Operations

Multiagent architectures have been used to enhance fault tolerance in distributed systems since the early days of cloud computing. Studies indicate potential benefits of a multiagent architecture for performing and coordinating fault recovery activities and enabling self-healing responses to specific types of resource and availability faults in distributed systems [7, 16]. New studies are exploring ways to use large language models as part of AIOps solutions to automate the analysis of natural language content, including log files, that is often analyzed by humans.

A survey of autonomous AIOps and generative AI solutions in cloud-native contexts suggests applications such as anomaly and root cause detection for LLM solutions [9]. Autonomous agents using LLMs in Infrastructure as Code applications have a 96.8% success rate in anomaly detection and a 95.2% success rate in identifying misconfigured security controls, with a mean time to resolution of 6.9 minutes in a closed-loop system [10]. Publications on autonomous medical AI applications are growing at a compound annual growth rate of 32% from 2021 to 2026 [8].

2.4 Gap Synthesis

The literature review identified knowledge gaps between the three research areas. Studies of methods and trends in FHIR-based workflow processes do not discuss self-healing capabilities. Studies of SRE automation include methods for building self-healing

capabilities but do not provide a specific fault classification scheme for FHIR implementations. Studies of agentic AI include methods for building self-healing capabilities in cloud computing but do not discuss other relevant topics for FHIR implementations in healthcare. FASER is necessary to address these knowledge gaps because no other framework includes agentic LLM and other capabilities, such as FHIR-specific fault classification and methods for verifying confidence in and completion of other remediation actions.

3. Methodology

3.1 Research Design

FASER was developed using a framework development approach involving both FHIR service failure analyses and other research on multi-agent systems and autonomous SRE. The current paper does not offer an empirical analysis of the FASER framework but instead provides other analyses related to each step in developing the FASER framework, comparing these analyses with other published analyses of FHIR service failures and published information about SRE runbooks. Examples of empirical techniques that have been applied to cloud computing frameworks include controlled fault injection and other data collection techniques [11, 14]. Empirical testing is part of planned future research on FASER (Section 5.5).

The methodology has three parts. The first part involved creating a fault taxonomy for FHIR applications using information about faults in FHIR workflow engine processes and other sources, including the HL7 FHIR R4 specification [21] and other published sources of information about faults in healthcare cloud operations. The second part involved creating the FASER framework and outlining its four phases (Observe, Reason, Act, and Verify) based on needs revealed through development of the fault taxonomy. The third part involved assessment of potential FASER framework functions and comparison with those of traditional runbooks used in SRE. FASER development used other published data on autonomous cloud operations and agentic AI applications, which are not FASER data.

3.2 FHIR Fault Taxonomy Development

Four kinds of faults were identified as distinct from other faults that might occur in a FHIR workflow engine. Unique approaches to addressing these faults are necessary, but SRE tools do not provide them. Additional information on the behavior of these fault types under the FHIR R4 specification and in various server and platform implementations, as well as FASER guidance on addressing these faults, is included below.

Resource validation errors occur when resources used in a workflow through the workflow engine do not conform to the profile specified in an implementation guide. Resource validation errors cannot be fully detected through HTTP status codes. Conforming systems return OperationOutcome resources or other information about resource validation errors, including the error's severity and where it occurred in a resource. Special knowledge of FHIR is required to interpret information returned in OperationOutcome resources. Documentation about the FHIR standard, including the OperationOutcome resource and $validate operation, is available. Documentation about a system's particular approach to validating resources received is implementation-specific.

Transaction errors involve issues with multi-resource bundles and dependencies between them. Transactions in Bundles are atomic under FHIR R4, meaning that all Bundle resources are created on the server or none of the Bundle resources are created, and an error response, including an OperationOutcome resource, is returned to the sender in the Bundle [21]. No partially created resources will be found on the server as a result of a successful atomic transaction. Other FASER-related operation processing errors include (i) a failed transaction that prevents a business process step from being completed and results in inconsistent data on the sender's and receiver's systems and (ii) errors associated with nonatomic batch bundles that result in some resources being successfully created on the server and others not being created; and (iii) a successful transaction that results in errors with later dependent processes, such as notifications to other systems, preventing completion of the overall business process. Solutions to these problems are needed in addition to resending resources.

Subscription notification failures occur when notifications to subscribed endpoints for a live FHIR subscription do not reach or are not accepted by those endpoints. In R4, subscription notifications can be delivered to subscribed endpoints via REST, WebSocket, email, and other mechanisms. Minimal subscription state information, including an error code and text description of an error condition, is available in R4. The R4 standard does not include mechanisms such as heartbeats to detect missed subscription notifications. Topic-based subscriptions in R5, implemented on R4 servers using the Subscriptions R5 Backport Implementation Guide, offer additional capabilities to detect missed subscription notifications, such as heartbeats and subscription event counters [22]. Approaches to retrying failed subscription notifications differ. Factors affecting subscription notification failures and possible remedies include problems with transport and other aspects of the subscribed endpoint and subscription or topic criteria.

Consent enforcement errors are types of errors specific to FHIR applications but not generally seen in other cloud applications, including cases in which resources transferred by a workflow engine do not match a patient's consents and resources with existing consents are not transferred. FHIR R4 includes a Consent resource that can be used to represent consents given by patients, including fields to record other details about consents, and an AuditEvent resource that can be used to represent security events involving patient data [21]. FHIR R4 does not require consent and audit features for all resource transfer events. These features are provided by features of the access control system on the FHIR server and may depend on other resources and profiles implemented on the server. Identifying consent-related errors is a goal of the FASER program; the Observe component of FASER uses AuditEvent

and other records to identify cases in which resource transfers to and from the patient did not comply with the patient's consents for the intended use of the data.

3.3 FASER Framework Architecture

FASER is a closed-loop system with three levels and four steps. At the Observe level, systems continuously send telemetry data in FHIR format and document issues. At the Reason level, systems use an LLM diagnostic tool to diagnose and classify issues and recommend specific playbooks and actions, subject to policies and other limitations. At the Act level, systems attempt to resolve issues by using recommended playbooks or, if the system cannot resolve them, notify human responders of the issue. At the Verify level, systems check FHIR and other operational status information to confirm that desired workflow conditions have been restored and relay this information to the Observe level. The feedback loop follows the sequence Observe → Reason → Act → Verify → Observe.

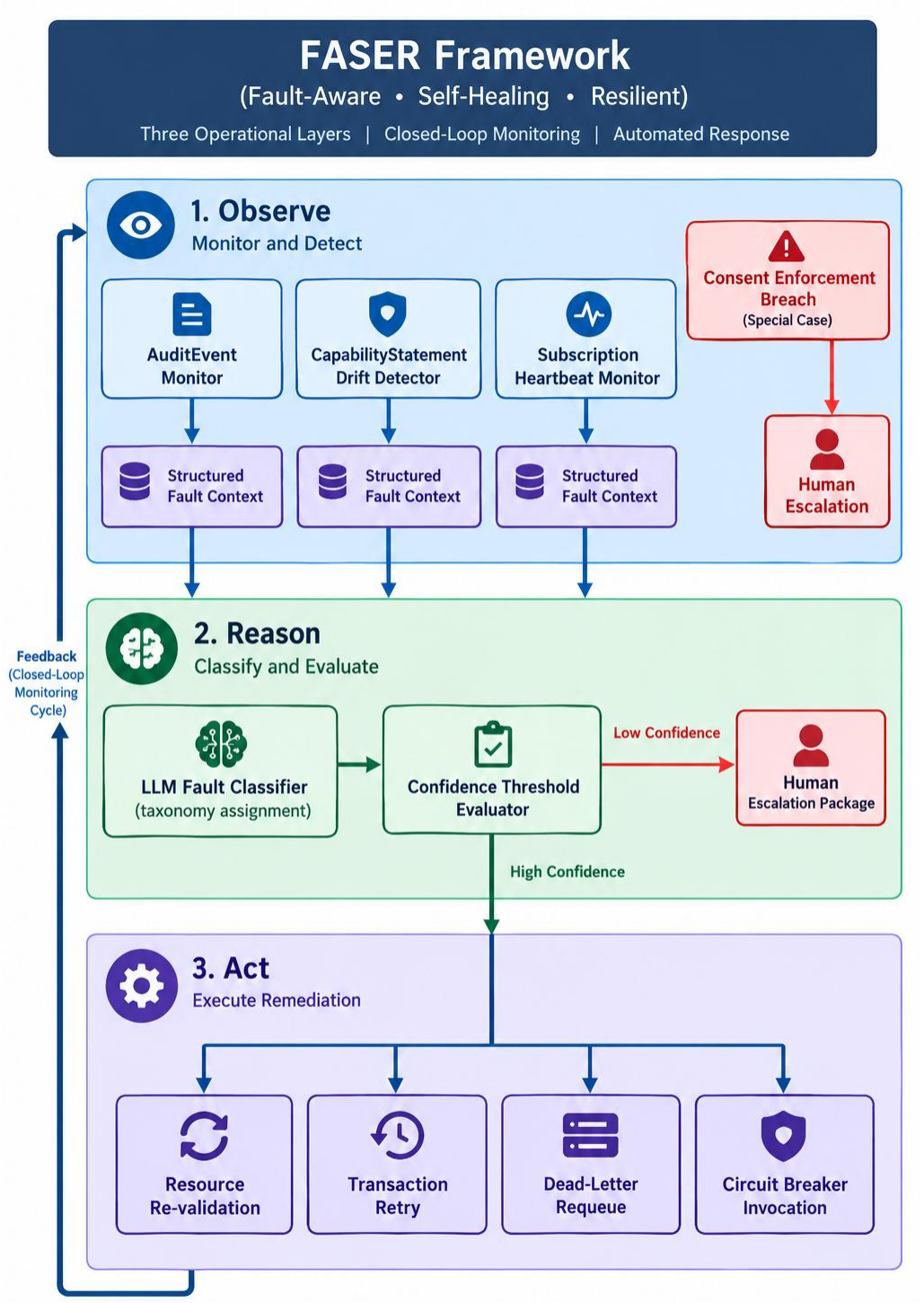


Figure 1. FASER Closed-Loop Architecture (Observe → Reason → Act → Verify → Observe)

3.4 Analytical Evaluation Criteria

The framework is assessed against these four factors: the effects of the tooling on SREs' ability to decrease resolution times for SLO violations versus traditional paper-based SRE processes; steps taken to evaluate and rate confidence in autoresponse decisions to reduce autoresponse actions; effects on SLO performance; and changes to SRE processes involving human incident responders, depending on the nature of the incident. Studies of autoresponse in cloud environments lend further support to the analysis of the framework. Section 5.5 discusses reporting requirements for FASER deployments.

4. FASER Framework Design

4.1 Observe Layer

The Observe layer monitors three types of native FHIR log data. AuditEvent log data involves logging events sent by the server as FHIR resources. Comparisons with expected AuditEvent data are possible without translating log data into other formats. The extent of available and useful AuditEvent log data may depend on server settings. CapabilityStatement comparisons are a monitoring technique described in the FASER guidance but not yet part of FHIR monitoring tools. A CapabilityStatement monitoring tool in the Observe layer pulls the server's CapabilityStatement resource using the metadata operation at regular intervals and compares it to an expected CapabilityStatement. Differences between the two may indicate issues with server setup or deployment. Subscription log data involves logging subscription notifications, including heartbeats and other data that may help detect missed notifications for topic-based subscriptions. Other data reported by the notification engine, along with details in the Subscription resource, may be used to detect missed notifications for simpler FHIR R4 criteria-based subscriptions [22].

Signal systems provide fault object data, not input data from their systems, such as fault type, related system IDs, timestamp data, and other FHIR-related data. Fault object data provided by signal systems are an input to the reasoning step, intended to help the language model infer fault information without requiring direct knowledge of the system providing the input data that the language model may lack. The same signal systems used in the reasoning step are used in the verification step (Section 4.4) to gather information about postremediation conditions.

4.2 Reason Layer

In the Reason layer, fault context information collected in the Observe layer is analyzed in two steps. First, an agentic language model uses the FHIR fault taxonomy and other resources to identify and classify the nature of the fault and the appropriate remediation actions from a preapproved playbook library. Second, the language model assesses its confidence in the fault classification against a threshold, and the Reason layer assesses whether remediation actions can be performed autonomously according to applicable policies. Remediation actions that meet confidence and policy requirements are forwarded to the Act layer for automated remediation. Cases where confidence and policy requirements are not met are referred to human SREs with a package of information about the fault.

This confidence threshold was introduced to reduce the risks of autonomous repair, including wrong responses to system faults caused by LLM misinformation. Human oversight of LLM responses with low confidence can help address the risk of cascading mistakes involving LLM misinformation during autonomous repair [8].

4.3 Act Layer

The Act layer provides playbooks for conducting remediation tasks that have been tested and confirmed to work with FHIR resources. Playbooks are associated with specific entries in the fault class taxonomy. Resource re-validation playbooks resubmit FHIR resources that previously failed validation, making corrections based on the playbook according to the implementation guide's profile. Automated corrections available through resource re-validation playbooks include only structural and formatting corrections. Clinical corrections may need to be made manually as part of the resource re-validation process. Transaction re-validation playbooks query state information about previously used resources in incomplete or failed steps and resubmit corrections as a new transaction bundle to preserve atomicity. Available corrections for batch bundles and other downstream issues are limited to unavailable or incomplete Bundle entries or steps and will use conditional create operations to avoid creating duplicate resources. Dead-letter queue reprocessing playbooks (an implementation technique, not part of the FHIR specification) retrieve dropped notification messages from the dead-letter queue, check endpoint availability, and resubmit available notifications with appropriate backoff. Circuit breaker invocation playbooks prevent degradation of the FHIR service interface by invoking circuit breaker functions [11, 12]. After making corrections with a playbook, the Verify step checks for conditions defined in that playbook.

Table 1: FHIR Fault Taxonomy with FASER Remediation and Verification Actions

Fault Class	Detection Signal	FASER Remediation Action	Verification Check (Verify Stage)	Escalation Trigger
Resource validation error	OperationOutcome issue with severity 'error' or 'fatal'	Profile-aware re-validation + structural (non-clinical) corrective transform	Resource accepted by the server and re-validates against the target profile	Repeated failure after 2 retries, or any transform that would alter clinical content
Transaction processing failure	Rejected transaction (4xx + OperationOutcome), mixed per-entry outcomes in a batch bundle, or failed downstream workflow step	Server-state re-read + re-validation + resubmission as a new atomic transaction, or targeted re-execution of incomplete entries/steps	Expected resources present in server state and downstream workflow step confirmed complete	Unresolvable dependency conflict or failed verification
Subscription notification drop	Missed heartbeat or event-count gap (topic-based subscriptions), or subscription status 'error' / engine delivery-log failure (R4 criteria-based)	Endpoint availability check + dead-letter requeue	Subsequent notifications delivered (2xx response) with a continuous event sequence	Endpoint unreachable after 3 attempts or recovery not verified
Consent enforcement breach	AuditEvent–Consent correlation mismatch (performed by FASER)	Transaction hold pending human review; consent policy re-evaluation by operator	Performed by the human reviewer	Any detected breach (zero-tolerance)

Note: retry and attempt limits are illustrative configuration parameters of the proposed design, not empirically derived values.

Autonomous remediation is only possible for the first three fault classes because they have relatively small and contained action spaces and observable symptoms that can be used to verify successful remediation. Consent policy violations are always remediated by SREs, as they are particularly sensitive in the healthcare field.

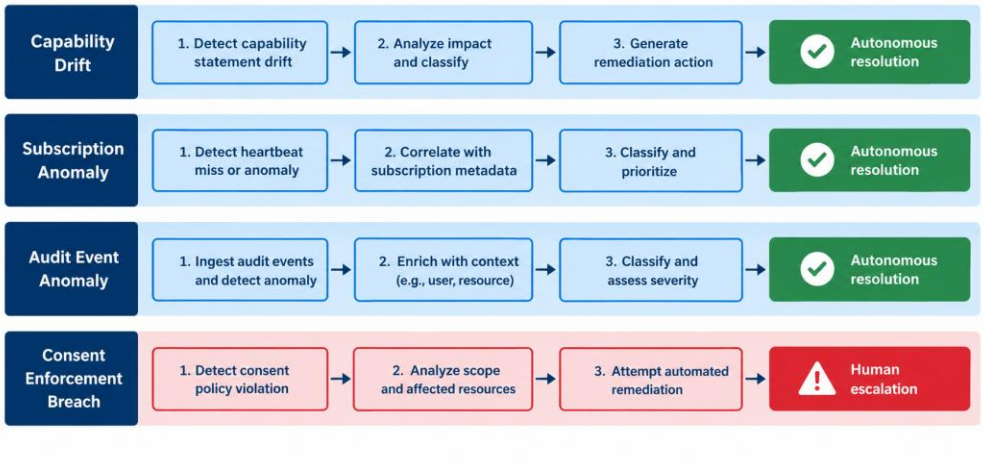


Figure 2. FHIR Fault Taxonomy — Detection, Remediation, and Verification Flow

4.4 Verify Stage

The Verify step is the final step in the reliability improvement process. Taking a recovery action by itself is not enough to close an alert about a workflow; a self-healing system must take additional steps to confirm the successful implementation of a recovery action and return the workflow to its desired state. The Verify step performs post-recovery checks after each Act step, using the same FHIR alert data and access to resources as the Observe step to check for recovery. Successful verification might include confirmation that a recovered resource or transaction has the desired attributes according to a FHIR profile within a certain time frame set out in the playbook, that other resources and steps in a recovered workflow transaction exist, and that notifications are being sent in the correct order by a recovered subscription workflow. If verification is successful, the Verify step documents the recovery and returns to regular monitoring. If verification is unsuccessful, the system may take further recovery actions as directed in the playbook or share information about the recovery attempt with a human reviewer for additional action.

A conceptual subscription notification example outlines the FASER process. A missed subscription notification to an active subscription is detected by the Observe phase. Actions taken by the Reason phase include identifying a missed subscription notification and selecting a standard dead-letter queue recovery process. Actions taken by the Act phase include identifying the status of the endpoint being monitored and attempting to resend missed subscription notifications at increasing intervals to the dead-letter queue. Actions taken by the Verify phase include checking whether the missed and future subscription notifications are being received in order by the endpoint. If the problem is not resolved within an acceptable time, more details about it are provided to a supervisor. For FASER, self-healing refers to successfully recovering subscription notifications, not just performing other corrective actions.

5. Analytical Evaluation and Expected Operational Characteristics

This section outlines the expected operational characteristics of the FASER framework based on its proposed architecture and design rationale. This section does not present empirical test results. Where quantitative figures appear, they are cited from the referenced literature and describe results from those prior studies; they should not be interpreted as measured or projected performance outcomes of FASER. The purpose of this section is to analyze how the architecture is expected to behave based on its design, drawing on operational requirements and findings from related autonomous cloud operations research.

5.1 Expected Impact on Remediation Latency

A principal operational advantage of the FASER system design is the intended reduction of human intervention in addressing errors. Human intervention is common in traditional SRE approaches using runbooks and other tools. Agentic systems are designed to eliminate human involvement in error detection, classification, and remediation for defined fault classes. Prior studies have reported reductions in mean time to repair (MTTR) through automated detection and remediation in cloud operations [13]. FASER applies a related automation principle to FHIR-specific operational failures by removing routine operator handoffs for fault classes eligible for autonomous remediation. The magnitude of any resulting MTTR improvement would require empirical evaluation of a FASER implementation.

In addition to the recovery time for resource validation and subscription notification errors, other variables that affect recovery time include retry and delivery delays for the workflow engine and the verification period for the Verify step in the recovery process. These and other variables must be considered to complete recovery successfully. The inclusion of the Verify step in the recovery loop allows for measurement of lower latencies based on successful recovery, in addition to successful completion of actions.

5.2 Fault Classification and Confidence Gating

The Reason layer's two-tier fault classification process helps mitigate hallucination risks for autonomous LLM fault repair. Human repair of faults with low confidence in classification limits autonomous repair activities to those that are likely to involve accurate LLM fault classification. Other research has demonstrated high accuracy among LLM-based infrastructure agents in detecting and classifying faults and other issues in IaC [10]. This other research supports studies of LLM-based classification in FHIR application settings but does not guarantee similar accuracy for FASER. There are four types of faults relevant to FHIR applications. The FASER fault classification problem may be simpler than general problems of classifying faults in IaC. Other research could explore whether simpler problems produce more accurate results.

The confidence threshold approach reduces human expert work by restricting it to high-value tasks, including identifying new and uncommon failure types not covered by the taxonomy, analyzing telemetry, and taking other actions to address consent violations. Consent violations are not automatically fixed, even when correctly identified, because of the critical nature of addressing consent violations in healthcare applications. An optional verify step adds an extra safeguard against misclassification errors by ensuring that incidents misclassified by the automated system are not closed during the automated response process but referred for additional action.

5.3 Implications for SLA Adherence

FHIR transaction SLAs are limited by bottlenecks in FHIR fault detection and resolution processes. Bottlenecks in production operations following documented procedures include human involvement, measured in minutes, between fault detection and response at production sites. Research on auto-healing methods and tools using machine learning technologies and other technologies to automate fault resolution in cloud computing systems has shown benefits to SLAs, including reduced human

involvement [14, 17]. FASER seeks to provide similar benefits to FHIR transaction SLAs by automating the FHIR Act and Verify processes, eliminating human involvement in response to certain levels of fault detected during the FHIR Reason process, and taking steps to resolve faults identified in the FHIR Verify process to meet SLAs.

Autonomous cloud management tools powered by AI have shown advantages for service continuity in multi-cloud applications [18]. FASER's architecture is designed to include features that could support SLA adherence for FHIR workflow engines, including autonomous remediation for defined fault classes and confidence-gated escalation to limit the risk of incorrect autonomous actions. Whether these design properties would translate to measurable SLA improvements in practice would require empirical evaluation of a deployed FASER instance.

5.4 Human-Escalation Boundaries

FASER does not eliminate the need for human SRE intervention but aims to reduce it. Under traditional runbook models, SREs are involved in responding to all fault events, including triage. Under FASER, the role of the SRE in responding to specific faults is reduced in cases not handled by the FASER system, including intentional consent failures and other types of faults not addressed in the FASER taxonomy model, as well as low reason confidence levels and unsuccessful attempts at automated remediation. Adoption of cloud monitoring and other automated remediation tools has reduced other SRE interventions needed to resolve day-to-day issues, freeing up SRE time for other reliability work [17]. FASER can provide similar advantages in the FHIR environment by using a taxonomy to limit SRE participation in addressing certain types of faults.

5.5 Limitations and Empirical Validation Requirements

FASER relies on telemetry data. The Observe layer uses telemetry data types native to FHIR, such as AuditEvent resources and subscription and CapabilityStatement conformance information, which are published by the workflow engine and other systems. Some types of telemetry data may not be present in all FASER implementations. AuditEvent data may be limited by server configuration, and data about failed subscription notifications may not be available unless the server supports topic-based subscriptions, a functionality requirement that is not part of FHIR R4. Incomplete or unavailable telemetry data in an implementation may lead to incomplete or unavailable data for the Reason layer, which in turn may reduce accuracy in fault classification and increase the transmission of fault reports to the Escalate layer. Large language models are more likely to hallucinate in the absence of context. Use of a confidence filter can help reduce the risk of hallucination but cannot prevent it. New fault types outside the four-category FHIR fault taxonomy are also more likely to be hallucinated. New fault types may be classified with low confidence until the taxonomy is updated using experience.

Most importantly, the FASER architecture has not been built or tested, and no test results are reported in this paper. To test the FASER architecture, testers would need to build prototype implementations for one or more implementations of a FHIR server, perform focused fault injection tests for different types of faults, and test various metrics related to fault classification and remediation, including successful and unsuccessful remediation and escalation actions, MTTR, SLA compliance, and other metrics, compared with actions taken according to a runbook. Testers would also need to test parameter values discussed in this paper to determine optimal values.

6. Discussion

Analytical advantages of FASER include benefits in addition to architectural advantages. Native implementation of the FASER architecture for FHIR applications eliminates other obstacles to implementing automated diagnosis and repair capabilities for FHIR applications that can prevent diagnosis and repair of FHIR faults. Some actions that are appropriate for other IT applications can be harmful or otherwise counterproductive for FHIR applications, including applications that use consents and resource-validation features. Elements of FASER's Observe function support appropriate diagnosis and repair of FHIR faults by including FHIR-specific information, such as resource IDs and OperationOutcome and AuditEvent-Consent data, in fault data records instead of converting FHIR data to other formats for other uses.

The Verify step extends this process by implementing it as a postloop step. Domain knowledge is needed both to select an action and to evaluate the outcome of that action; not all infrastructure-level indicators of action completion, such as resubmission of a resource and notification and reconciliation of a transaction workflow, necessarily indicate that the FHIR workflow is complete. Verifying the status of the FHIR workflow after taking an action can help with FASER service restoration.

In health IT systems, limited health information interoperability can cause patient harm by interfering with care and medication and treatment management. Metrics of FHIR workflow engine performance might not capture all patient safety risks. Studies have found patient harms due to limited health information exchange [3, 4]. The FASER approach can help health IT systems prevent patient harms by incorporating a taxonomy of errors, automated responses, and safeguards to ensure that a human reviews system activity in certain cases involving sensitive patient information.

FASER is an independent SRE and AIOps tool with healthcare applications. Other LLM-based tools for troubleshooting faults provide general infrastructure information about faults, including CPU, memory, and network data and configuration information [9, 10]. Problems with workflow engines handling FHIR data include FHIR-specific issues related to patient authorization. General telemetry tools cannot provide this information about faults. Features in the Observe component of FASER help ensure that the Reason component has the FHIR-related information needed to properly categorize faults.

The framework applies to a number of use cases. The four error categories were defined using operations defined in the FHIR standard, including those involving resource checking, transactions and batches, and consent and subscriptions. Ways of displaying and reporting errors in each error category may differ across FHIR server implementations and versions or implementation guides. Other organizations using FHIR in cloud services are expected to encounter similar error categories. Many examples in the FASER playbook library use FHIR API operations instead of other vendor API operations to make the framework more broadly useful for FHIR implementations. Section 5.5 provides citations to empirical studies on the transferability of the framework.

Responsible deployment of the FASER system involves audit features. Actions performed using the Act component of the FASER system are documented in the Reason component, including details about the classification confidence judgment and chosen action plan. Verification findings and other evidence are documented in the Verify component. Audit information collected through the FASER system can help improve the Act component over time and inform updates to the confidence threshold and action plans. Audit features of the type offered by FASER can be valuable for agentic systems deployed in healthcare and other highly regulated fields [8].

Table 2: FASER vs. Related Autonomous SRE Approaches

Capability	**General AIOps**	**LLM Infrastructure Agents**	**FASER**
Fault taxonomy	Generic infrastructure metrics	IaC configuration drift	FHIR-native (4 fault classes)
Telemetry source	Generic logs/metrics	IaC state files	FHIR AuditEvent, CapabilityStatement, Subscription delivery, and status signals
Remediation scope	Alert routing, scaling	Config remediation	FHIR playbook library
Confidence gating	Threshold-based	Rule-based	LLM confidence score + domain threshold
Post-action verification	Implementation-dependent	Implementation-dependent	Dedicated verify stage (post-action state check)
Consent-class handling	None	None	Zero-tolerance escalation
Healthcare compliance	None	None	FHIR R4 conformance-aware

Note: entries for related approaches summarize design characteristics described in the cited literature; FASER entries describe proposed design properties, not measured capabilities.

7. Conclusion

Healthcare FHIR workflow engines require reliability strategies capable of responding to the speed and specificity of FHIR faults in live clinical environments. The combined requirements of low latency, FHIR-native fault detection, and bounded autonomous remediation are not well addressed by conventional SRE tools. The FASER framework is proposed as a conceptual foundation for addressing this gap.

This article describes FASER, an approach to agentic SRE for FHIR workflow engines. One benefit of FASER mentioned in this article is the integration of LLM-based and other agentic and FHIR-specific techniques, such as FHIR monitoring and a system for classifying common errors, that has not been widely covered in prior research on agentic SRE and FHIR workflow engines. Techniques from FASER enable agentic approaches to SRE for FHIR workflow engines by taking advantage of FHIR data sources during the error diagnosis and resolution process. FASER techniques can be used to diagnose and resolve errors in healthcare cloud services. The Observe → Reason → Act → Verify → Observe cycle can help confirm resolutions to errors and trigger escalations as needed. Design decisions about the boundary for enforcing consents can help ensure that consent and other errors are always escalated to human reviewers, even when automated systems are very confident in their assessments, for patient safety and other regulatory reasons, at the cost of some operational efficiency.

The main contribution of this paper is its design and analysis of FASER, which provides a conceptual foundation for further research and development. Other potential research studies include building and testing a prototype to evaluate the accuracy of fault classification and repair, as well as other measures, such as successful and verified repairs, unnecessary and excessive upticketing of faults, speed of fault repair, and service-level agreement adherence, compared with runbooks. Other potential research activities include gathering data on additional types of faults to expand the taxonomy, as well as testing and adjusting confidence and verification parameters for various FHIR server implementations. Collaboration with FHIR Implementation Guide tool developers may be useful to update playbooks.

Statements and Declarations

Statement of Human and Animal Rights: This article does not involve any human participants, animal subjects, or identifiable personal data. No ethics committee approval was required.

Informed Consent: Not applicable, as no human subjects were involved in this research.

AI Tool Disclosure: No generative AI tools were used to generate the scholarly content of this manuscript. The author affirms responsibility for the entirety of the content.

Funding: This research received no external funding.

Conflicts of Interest: The author declares no conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers.

References

[1] Vorisek, C. N., Lehne, M., Klopfenstein, S. A. I., Mayer, P. J., Bartschke, A., Haese, T., & Thun, S. (2022). Fast Healthcare Interoperability Resources (FHIR) for interoperability in health research: Systematic review. *JMIR Medical Informatics*, *10*(7), e35724. https://doi.org/10.2196/35724

[2] Jukanti, M. (2025). Interoperability-driven workflow engines in healthcare IT: Enhancing care coordination through standards-based automation. *International Journal of Computational and Experimental Science and Engineering*, *11*(4), 8993–9003. https://doi.org/10.22399/ijcesen.4345

[3] Gazzarata, R., Almeida, J., Lindsköld, L., Cangioli, G., Gaeta, E., Fico, G., & Chronaki, C. E. (2024). HL7 Fast Healthcare Interoperability Resources (HL7 FHIR) in digital healthcare ecosystems for chronic disease management: Scoping review. *International Journal of Medical Informatics*, *189*, 105507. https://doi.org/10.1016/j.ijmedinf.2024.105507

[4] Amar, F., April, A., & Abran, A. (2024). Electronic health record and semantic issues using Fast Healthcare Interoperability Resources: Systematic mapping review. *Journal of Medical Internet Research*, *26*, e45209. https://doi.org/10.2196/45209

[5] Boosa, S. (2024). The future of site reliability engineering: AI-driven observability and autonomous operations in multi-cloud environments. *International Journal of Emerging Trends in Computer Science and Information Technology*, *5*(2), 204–214. https://doi.org/10.63282/3050-9246.IJETCSIT-V5I2P122

[6] Muthyam, R. (2025). From SRE to intelligent reliability engineering: Revolutionizing the discipline with AI. *International Journal of Computational and Experimental Science and Engineering*, *11*(4). https://doi.org/10.22399/ijcesen.4119

[7] Rajput, P. K., & Sikka, G. (2021). Multi-agent architecture for fault recovery in self-healing systems. *Journal of Ambient Intelligence and Humanized Computing*, *12*(2), 2849–2866. https://doi.org/10.1007/s12652-020-02443-8

[8] Alhazba, S. A., Ali, M. H., Alawi, D. A., Al-Betar, M. A., & Abd Elaziz, M. (2026). *From reactive AI to agentic systems: A review of autonomous medical AI agents in healthcare*. Archives of Computational Methods in Engineering. https://doi.org/10.1007/s11831-026-10655-y

[9] Srivastava, P. K., & Agrawal, S. (2026). A systematic survey of autonomous AIOps and generative AI in cloud-native infrastructure. *International Journal of Emerging Research in Engineering and Technology*, *7*(2), 203–209. https://doi.org/10.63282/3050-922X.IJERET-V7I2P125

[10] Apuri, H., Chinthala, M. M. R., Goel, S., Aurangabadkar, M., & Yepuri, C. (2026). Self-healing infrastructure: Autonomous LLM agents for real-time remediation of configuration drift and security misconfigurations in IaC deployments. *International Journal of Innovative Technology and Exploring Engineering*, *15*(4), 25–32. https://doi.org/10.35940/ijitee.D4757.15040326

[11] Hlybovets, A., & Paprotskyi, I. (2024). Increasing the fault tolerance in microservice architecture. *Cybernetics and Systems Analysis*, *60*(3), 480–488. https://doi.org/10.1007/s10559-024-00689-0

[12] Ganesan, M. (2026). Circuit breaker pattern in modern distributed systems: Implementation, monitoring, and best practices. *International Journal of Research and Applied Innovations*, *9*(1), 13580–13589. https://doi.org/10.15662/IJRAI.2026.0901011

[13] Verma, D. (2025). *Beyond reactive IT: Quantifying the transformative impact of AIOps on service management*. International Journal of Emerging Research in Engineering and Technology, ICRCEDA2025 Conference Proceeding. https://doi.org/10.63282/3050-922X.ICRCEDA25-102

[14] Syed, A. A. M., & Anazagasty, E. (2024). AI-driven infrastructure automation: Leveraging AI and ML for self-healing and auto-scaling cloud environments. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *5*(1), 32–43. https://doi.org/10.63282/3050-9262.IJAIDSML-V5I1P104

[15] Le, T. K. H., Kongsiriwattana, W., Meny, A., Kaewsaeng-On, R., Nimyungdee, C., & Namee, K. (2025). *Enhancing healthcare interoperability with FHIR: A systematic approach to online data management. In Proceedings of the 2024 9th International Conference on Information Systems Engineering (pp. 35–40)*. ACM. https://doi.org/10.1145/3711954.3711958

[16] Azaiez, S., & Chainbi, W. (2016). *A multi-agent system architecture for self-healing cloud infrastructure*. https://doi.org/10.1145/2896387.2896392 [Note: retain only after verifying title, authors, and proceedings information independently]

[17] Muthusamy, M. (2025). Cloud-based intelligent monitoring and self-healing systems using machine learning and reliability engineering. *International Journal of Research and Applied Innovations*, *8*(6), 13172–13181. https://doi.org/10.15662/IJRAI.2025.0806040

[18] Banu, M. A. R. (2026). Autonomous cloud operations: Integrating AI for reliability and scalable system performance. *International Journal of Technology, Management and Humanities*, *12*(1), 2050–2062. https://doi.org/10.21590/ijtmh.v12i01.297 [Note: retain pending independent publisher confirmation]

[19] Tabari, P., Costagliola, G., De Rosa, M., & Boeker, M. (2024). State-of-the-art Fast Healthcare Interoperability Resources (FHIR)-based data model and structure implementations: Systematic scoping review. *JMIR Medical Informatics*, *12*, e58445. https://doi.org/10.2196/58445

[20] Marfoglia, A., Nardini, F., Arcobelli, V. A., Moscato, S., Mellone, S., & Carbonaro, A. (2025). Towards real-world clinical data standardization: A modular FHIR-driven transformation pipeline to enhance semantic interoperability in healthcare. *Computers in Biology and Medicine*, *187*, 109745. https://doi.org/10.1016/j.compbiomed.2025.109745

[21] HL7 International. (2019). *HL7 FHIR Release 4 (v4.0.1)*. https://hl7.org/fhir/R4/

[22] HL7 International. (2023). *Subscriptions R5 Backport Implementation Guide, version 1.1.0 (STU 1.1)*. https://hl7.org/fhir/uv/subscriptions-backport/STU1.1/