

System Safety for Health Information Technology Report

September 2025

Authors:

John Thomas, PhD

Eugenia Kim, MS

Polly Harrington, MS

Nancy Leveson, PhD

Massachusetts Institute of Technology

Partnership for Systems Approaches to Safety and Security

and supported by

Stephen Powell, DHA

Alana Keller, MBA, PMP

Abigail Harte-Williams, MPH, MS

Braden Burnett, BBA

Synensys, LLC

Contact: Dr. Stephen Powell

spowell@synensysglobal.com

A System-Theoretic Process Analysis (STPA) of the U.S. Department of Veterans Affairs Health Information Technology Systems

Executive Summary

Clinical decision support systems (CDSS) are increasingly being used to support decision-making in diagnosis and treatment. The proliferation of automatic alerts and reminders, condition-specific order sets, patient-specific data reports, and other features have introduced both benefits and pitfalls for healthcare. This report examines weaknesses and gaps related to health information technology (HIT), focusing on CDSS, and identifies actionable recommendations to improve the implementation and operation.

In practice, most CDSS deficiencies are caught by chance—often in a live healthcare situation involving a patient (83%) or in a routine test case (31%)¹. Many deficiencies are only identified and fixed long after they were first introduced. Voluntary user reports and finite test cases do not provide a comprehensive way to identify critical CDSS deficiencies in a system.

Hazard analysis is commonly used in safety-critical industries to efficiently and systematically identify weaknesses in a system before it is deployed. In this project, a modern hazard analysis technique called System-Theoretic Process Analysis (STPA) was used to proactively identify weaknesses in common CDSS technologies and the environment in which they are used. STPA was applied to the HIT systems of the U.S. Department of Veterans Affairs (VA), focusing on the VA's CDSS environment using Oracle Cerner Millennium. The analysis examined how control structures, responsibilities, decision-making processes, and feedback loops can interact to create unsafe outcomes.

Systemic Factors

The following systemic factors were identified along with recommendations to address them.

1. Unused, Delayed, and Missing Feedback
2. Reactive vs. Proactive
3. Resource Constraints
4. Inadequate Oversight
5. Competing Priorities
6. Lack of Clinical Rule Ownership
7. Inadequate Learning from Past Events and Root Cause Analyses

1. Unused, Delayed, and Missing Feedback

There is a large amount of data that is available related to the VA's CDSS rules, but currently, there is no group that actively monitors that data or is responsible for ensuring the CDSS rules are useful

¹ Wright et al, 2016. "Analysis of clinical decision support system malfunctions: a case series and survey." Journal of the American Medical Informatics Association : JAMIA 23 (6): 1068-1076.

or effective over time. There is also relevant CDSS rule audit data that is available, but those lookback views are only available temporarily, preventing long-term historical assessments. Certain CDSS data, therefore, either exists as unused feedback or becomes inaccessible.

As part of the current total rule catalog, there are also model rules, which are vendor-supplied CDSS rules. In some instances of these model rules, the VA is several versions behind the vendor's latest release of the rule due to untimely or unreliable communication with the vendor. This delay creates the potential for outdated or incomplete model rules to remain active within the clinical environment.

Additionally, documentation of a rule's rationale exists and is captured during the approval process of the rule, but there is currently no formal mechanism to link CDSS rules to their original intent and rationale. The rule's rationale is not systemically associated after the rule is deployed.

Sections 4.1.1, 4.1.3, and 4.1.8 of the report go into further detail on the types of data that are available but unused, the model rules that are delayed or unimplemented, and the linking mechanism that is missing for CDSS rules and their rationale.

2. Reactive vs. Proactive

The current process for monitoring CDSS rule performance is predominantly reactive. The health information technology (IT) team is made aware of a rule that is broken primarily through a ticket submission mechanism. This process is entirely voluntary, dependent on individual reporting behavior, and is therefore subject to variability and underreporting. Section 4.1.2 describes this reactive feedback loop and the need for proactive monitoring.

3. Resource Constraints

People and technical resources are both constrained. For the thousands of rules that the VA currently has in its total catalog, there are currently only three to four people managing those rules. Moreover, from a technical perspective, the current infrastructure presents constraints on the scalability of the CDSS, especially with additional sites coming online with the new electronic health record (EHR). Section 4.1.4 explains several contributing factors for both the human and technical resource constraints.

4. Inadequate Oversight

Although other safety-critical industries outside healthcare have hazard analyses and human factors analyses as standard practices, there currently appears to be no effective hazard analyses or human factors analyses being performed by the VA's EHR vendor to identify and prevent the types of software deficiencies that are identified by STPA. There seems to be no government requirement for either the vendor or the client to perform a hazard analysis before CDSS is used or conduct a human factors analysis before CDSS is implemented. This lack of a hazard analysis requirement can be costly. Costs to fix errors substantially increases as a project matures, particularly when errors are discovered during operational settings. A proactive hazard analysis could save both money and lives. Section 4.1.5 further explains the impact of hazard and human factors analyses from a safety and cost standpoint.

5. Competing Priorities

Stakeholders from different clinical specialties within the VA have differing priorities that sometimes conflict. The VA also has shared ecosystems with the Department of Defense (DoD), which introduces additional competing priorities, particularly when it comes to CDSS rule design. Section 4.1.6 provides a specific example of a rule design conflict resulting from competing priorities between different VA clinical domains.

6. Lack of Clinical Rule Ownership

There is a missing responsibility in the sense that there is no clinical rule ownership. This is significant because there is no tracking of changes, such as clinical guidelines, that affect CDSS rules; no continuous monitoring of clinical relevance and only perhaps a periodic reassessment; no checks of rule effectiveness; and no checks for unintended consequences after implementation. Section 4.1.7 goes in-depth into this systemic factor and describes the current fragmented, informal, voluntary, and undocumented state of clinical rule ownership.

7. Inadequate Learning from Past Events and Root Cause Analyses

The health IT team does not receive structured feedback or communication from root cause analyses (RCAs), even in cases where adverse events are directly attributable to CDSS rules. The subject matter experts who were interviewed for this analysis informed the MIT team that as the health IT team, they have not received any feedback from RCAs. There is no formal tracking mechanism to establish a historical on the health IT team record of rule-related adverse safety events. Section 4.1.9 explains the resulting gaps and lack of organizational learning.

Recommendations

The following recommendations were developed to address the systemic factors summarized above.

Recommendation 1: *Establish a clear, formal rule ownership role.*

An entity must be assigned the responsibility of and accountability for owning the rules. Rule ownership must include responsibilities for monitoring and maintaining the rules after implementation. Specifically, this would include monitoring relevant clinical guidelines for changes that require rule updates, monitoring clinical effectiveness of rules, and monitoring unintended consequences of the rules. Regarding clinical effectiveness, this rule-owning entity should identify measures to evaluate clinical impact and, if a rule is determined to be ineffective, provide an appropriate response. In other words, the rule owner should create ways to analyze metrics on the catalog of rules and consider ways to collect and use feedback from clinicians on rule usefulness to guide decisions on rule refinement and reduce alert fatigue.

One responsibility of this rule owner should be to improve rule management by clearly documenting and linking rule metadata. This metadata might include the original requestor, rule rationale, clinical objective, source documentations, and change history. The metadata also needs to be documented in a structured, easily searchable format. Building and/or expanding upon reporting tools that would allow solution teams and super users to proactively analyze rules is

recommended. This approach has already shown success for pharmacy teams and should continue to be expanded. Given the systemic factor meant to be addressed by this recommendation, this formal rule ownership must be established in a way that survives the organizational state of the larger ecosystem, including surviving reorganizations, staff turnover, and changing organizational structures.

Recommendation 2: *Incorporate strong proactive measures.*

There are meaningful measures that should be implemented to move beyond after-the-fact reports of problems during clinical care and reactive change requests. These measures might include developing systems that actively monitor external changes, such as clinical guideline updates, new drug codes, and position changes, to name a few examples. A hazard analysis on the CDSS should be performed or required to reduce cost and prevent deficiencies before they have a clinical impact. To prevent software behaviors that exacerbate confusion, human error, usability issues, and workload, a human factors analysis should be conducted or required, thereby reducing cost and improving safety. Automated monitoring tools should be considered that link rules to source guidelines or terminologies, allowing for proactive identification of affected rules.

In a similar vein, coordination between the sites, working groups, and deployments should be strengthened to improve proactive alignment. This coordination should prevent rule divergence across sites by improving coordination between deployment teams and partners. There should be a proactive effort to avoid redundant or duplicate rules across VA sites.

Recommendation 3: *Improve the model rule coordination with the vendor.*

The underlying causes of current communication gaps should be thoroughly examined. This investigation can be done by looking into the reason behind why relevant VA teams are not always being notified of an update. Additional questions should be answered such as whether technology solutions can be created for automatic notification, whether the onus should be on the health IT team to reach out to the vendor and contact them first to get a status instead of waiting for information, and whether there are better ways for the vendor to roll out updates. There should be a simple and easy-to-use process established for the vendor to notify the health IT team and send any model rule changes. This process should be clearly documented and used for all vendor updates.

Recommendation 4: *Ensure adequate resources.*

The health IT team must be appropriately staffed and be able to not only react to tickets and build rules based on those tickets, but also monitor what's already in the system in a prospective manner.

Recommendation 5: *Improve learning from past events and root cause analyses (RCAs).*

RCA integration with CDSS review should be improved. The entity responsible for RCAs must ensure that RCA investigations examine whether CDSS rules contributed to adverse events or could prevent similar events in the future.

Conclusions

The results of the hazard analysis show that the safety and effectiveness challenges in CDSS cannot be solved with piecemeal fixes or isolated technical upgrades. They arise instead from systemic factors in feedback, governance, ownership, management, and interactions that shape how rules are designed, implemented, and maintained across the VA. As is the nature of STPA, the recommendations outlined in this report are not about assigning fault to individuals, but about designing the larger system so that safe and effective behavior is the outcome. By implementing clearer ownership, stronger feedback loops, proactive monitoring, better vendor coordination, and formal hazard analysis practices, the VA can move from a reactive model to a safer one. While some of these changes will require cultural and organizational shifts, these recommendations and changes are proven approaches in other safety-critical industries. The opportunity is not simply to fix broken rules, but to change the CDSS rule management system into a foundation for improved patient safety.

Table of Contents

1.	Background	8
1.1.	Introduction to Clinical Decision Support Systems	8
1.2.	A Brief Introduction to Systems Theory	8
2.	Method.....	8
2.1.	System-Theoretic Process Analysis.....	8
3.	Findings.....	10
3.1.	Losses and Hazards	10
3.2.	Boundary of the System	11
3.3.	Control Structure	12
3.4.	Unsafe Control Actions	15
3.5.	Scenarios	16
4.	Discussion and Recommendations	19
4.1.	Systemic Factors	20
4.1.1.	Delayed/Missing Feedback and Lack of Customization for Model Rules	20
4.1.2.	Reactive vs. Proactive.....	22
4.1.3.	Unused and Missing Feedback	22
4.1.4.	Resource Constraints.....	25
4.1.5.	Inadequate Oversight.....	26
4.1.6.	Competing Priorities	28
4.1.7.	Lack of Clinical Rule Ownership	29
4.1.8.	Missing/Inaccessible Feedback: Rule Intent and Rationale	31
4.1.9.	Inadequate Learning from Past Events and Root Cause Analyses (RCA).....	31
4.2.	Recommendations	32
5.	Conclusions	33
6.	Appendix	36
4.1.	Appendix A: List of Unsafe Control Actions.....	36
4.2.	Appendix B: List of Scenarios.....	45

1. Background

This section provides background information on clinical decision support systems and systems theory.

1.1. Introduction to Clinical Decision Support Systems

Clinical decision support systems (CDSS) are a set of health information technology (HIT) tools that provide clinicians and staff with knowledge and information to support effective diagnosis, treatment, and outcomes by supporting effective clinical decision-making. CDSS components include computerized alerts and reminders, clinical guidelines, condition-specific order sets, patient-specific data reports, documentation templates, and contextually relevant reference information, among other tools.² However, challenges have constrained the value of CDSS.

The goal of this study was to identify potential weaknesses and gaps related to HIT, focusing on CDSS, and to generate actionable recommendations that address those gaps. This was done using System-Theoretic Process Analysis (STPA), a hazard analysis technique based on systems theory.

1.2. A Brief Introduction to Systems Theory

Organizations and technologies operate as complex sociotechnical systems where safety and performance are shaped not only by individual components, but also by their interactions and interdependencies. Traditional safety approaches often focus narrowly on component unreliability or human error, but these methods do not account for hazards that emerge from the emergent behavior of the system. Systems theory reframes safety as a control problem: Accidents occur not simply because components fail, but because control over system behavior, including interactions between components, is inadequate or ineffective in a given context.

From this perspective, ensuring safety requires analyzing how decisions, controls, and feedback are structured across the organization from technology and processes to human operators and policy. This shift allows leaders to understand vulnerabilities that arise from weaknesses like missing feedback, delayed actions, or poorly coordinated responsibilities, which cannot be uncovered by component-level analyses alone. By applying this framework, organizations like the VA can identify systemic causes of hazards and implement controls that strengthen the entire system.

2. Method

The following section explains the methodology used for this analysis.

2.1. System-Theoretic Process Analysis

The System-Theoretic Process Analysis (STPA) technique provides a structured method for identifying hazards, unsafe control actions, and causal scenarios within complex sociotechnical

² <https://www.healthit.gov/topic/safety/clinical-decision-support>

systems. Rather than relying on only retrospective investigations or probabilistic component failures, STPA proactively examines how control structures, responsibilities, decision-making processes, and feedback loops can interact to create unsafe outcomes. The following section outlines the steps of STPA and how they can be applied to the VA's clinical decision support environment to reveal vulnerabilities and generate actionable recommendations for improved safety.

The first step of STPA identifies losses and hazards, defined as follows.³

- A **loss** involves something of value to stakeholders and may include a loss of human life or human injury, property damage, environmental pollution, loss of mission, loss of reputation, loss or leak of sensitive information, or any other loss that is unacceptable to the stakeholders.
- A **hazard** is defined as a system state or set of conditions that, together with a particular set of worst-case environmental conditions, will lead to a loss.
- A **system-level constraint** specifies system conditions or behaviors that need to be satisfied to prevent hazards (and ultimately prevent losses).

Once losses of concern are identified, the next step is to define the hazards related to these losses.

A control structure is then created as the next step. The control structure provides the central model for understanding how safety is managed within a system. It represents the hierarchical arrangement of controllers, controlled processes, and the feedback loops that connect them. The purpose of the control structure is not to model every technical detail, but to capture the functional relationships and flows of information and control that determine system behavior. As emphasized in the STPA Handbook, accidents often occur not because individual components fail, but because the control structure as a whole does not provide adequate or effective control. By visualizing and analyzing the control structure, we can identify unsafe control actions, gaps in feedback, and interactions between components that create systemic hazards.

An unsafe control action (UCA) is a control action that, in a particular context and worst-case environment, will lead to a hazard.⁴ There are four ways a control action can be unsafe:

1. **Not providing** the control action leads to a hazard
2. **Providing** the control action leads to a hazard
3. Providing a potentially safe control action but **too early, too late, or in the wrong order**
4. The control action is **applied too long or stopped too soon** (for continuous control actions, not discrete ones)

Every UCA contains four parts: **<Source> <Type> <Control Action> <Context>**. The first part is the controller that can provide the control action. The second part is the type of unsafe control action (provided, not provided, too early or too late, stopped too soon or applied too long). The third part is the control action or command itself (from the control structure). The fourth part is the context discussed above.

³ Leveson and Thomas. STPA handbook. 2018.

⁴ Leveson and Thomas. STPA handbook. 2018.

After the UCAs are identified, the next step of STPA is to identify loss scenarios, i.e., why those UCAs might occur. A loss scenario describes the causal factors that can lead to the unsafe control actions and hazards.⁵ There are two types of questions answered in the loss scenarios:

- a) Why would unsafe control actions occur?
- b) Why would control actions be improperly executed or not executed, leading to hazards?

To build these scenarios, we can start with four types of scenario archetypes. The four scenario classes can be described as follows:

- Scenario Class 1: Unsafe Controller Behavior
- Scenario Class 2: Unsafe Feedback Path
- Scenario Class 3: Unsafe Control Path
- Scenario Class 4: Unsafe Controlled Process Behavior

These four scenario classes are visualized in Figure 1 below:

STPA: Four Classes of Formal Scenarios

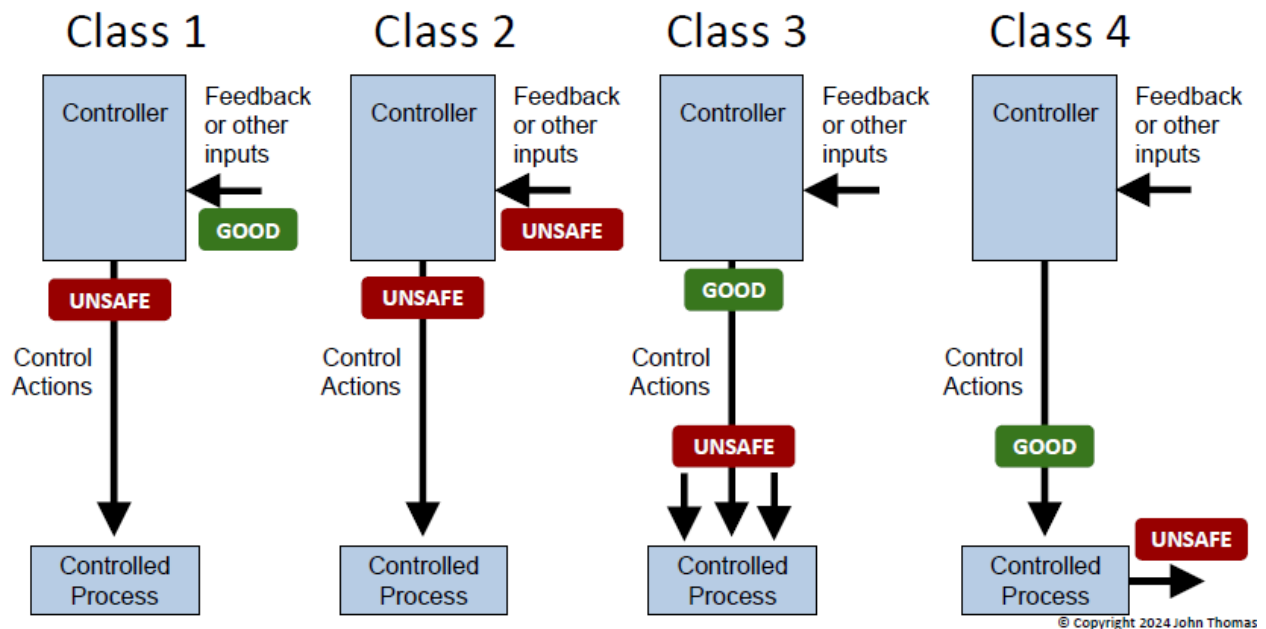


Figure 1. Scenarios can be formally categorized into four classes.

3. Findings

This section presents the results of the analysis of the U.S. Department of Veterans Affairs health information technology systems using the steps of STPA described above.

3.1. Losses and Hazards

Accidents (losses) and hazards of importance to healthcare stakeholders:

⁵ Leveson and Thomas. STPA handbook. 2018.

- **L-1:** Loss of life or serious injury to the patient, resulting in patient harm and/or decline in the patient's quality of life (QoL)
- **L-2:** Loss or reduction in providers' quality of work life (burnout)
- **L-3:** Loss of reputation of the healthcare organization
- **L-4:** Loss of patient throughput (efficiency)
- **L-5:** Loss or reduction of ability to work with peer and academic institutions
- **L-6:** Loss of financial feasibility or sustainability (unacceptable cost of care)

The system-level hazards for this STPA are identified in the list below and formatted in the following manner: <Hazard specification> = <System> & <Unsafe Condition> & <Link to Losses>.

- **H-1:** Patient receives less than acceptable standard of care [L-1, L-3]
 - H-1-1: Health information technology (HIT) does not reflect patient's condition accurately [L-1, L-2, L-3]
 - H-1-2: HIT degrades provider QoL, leading to reduced staffing levels for providing acceptable care [L-1, L-2, L-3, L-4, L-5]
- **H-2:** HIT is not usable in operational, clinical contexts [L-2, L-4, L-6]
 - H-2-1: Necessary HIT data cannot be accessed and entered in a timely manner (in an intuitive manner) [L-1, L-2, L-3, L-4, L-6]
 - H-2-2: HIT data is misinterpreted by the user(s) [L-1, L-3]
- **H-3:** Patient data is accessed and used by unauthorized parties [L-3, L-5]

3.2. Boundary of the System

The boundary of the system considered for this research needed to be limited for practical considerations to ensure the work would be finished within the time constraints of the research contract period. The focus areas for the STPA research team included the health IT team and the clinicians within healthcare providers and staff as well as the CDSS and EHR within the health information technology system ecosystem, specifically scoped to Oracle Millennium.

3.3. Control Structure

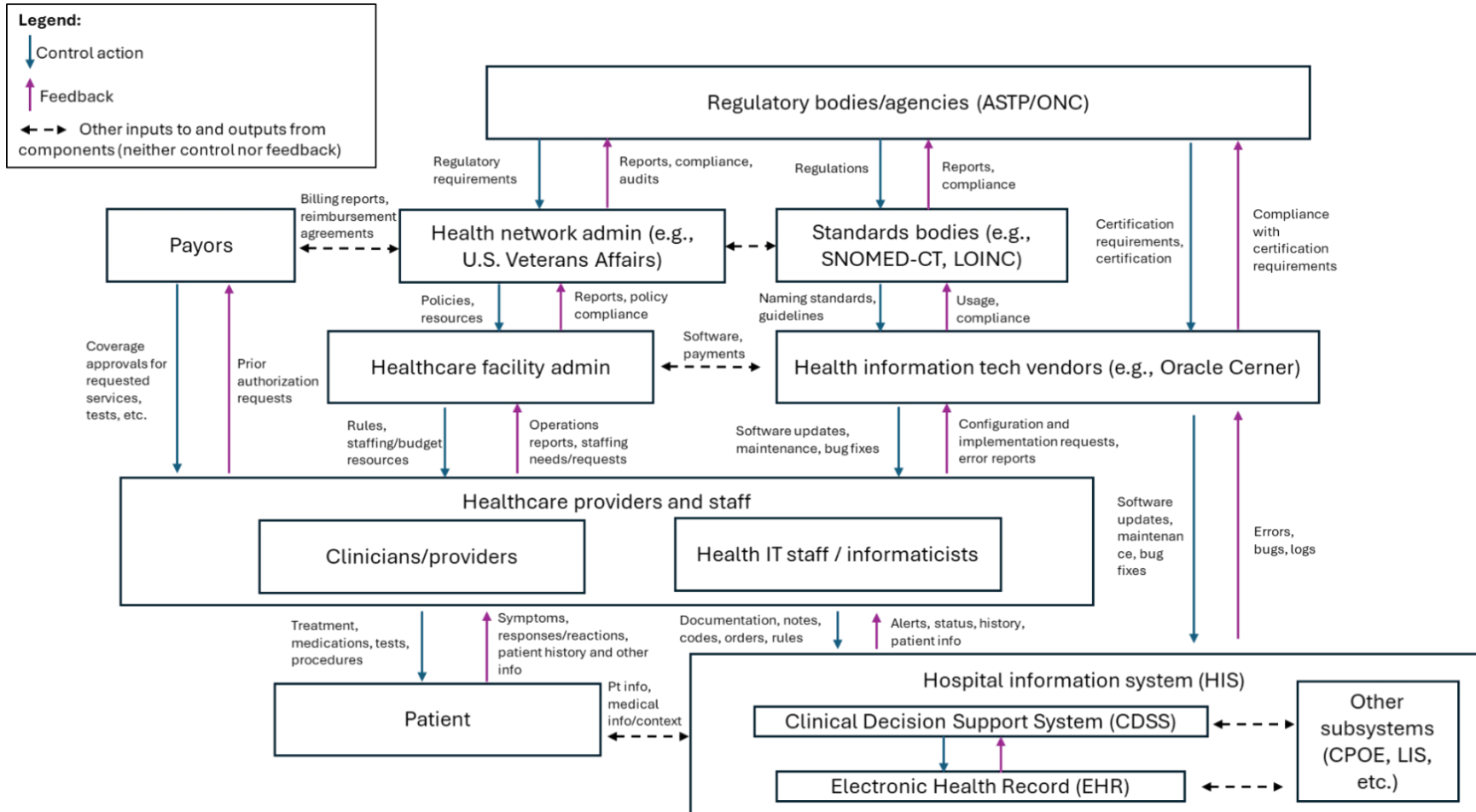


Figure 2. A high-level, abstracted control structure is first modeled.

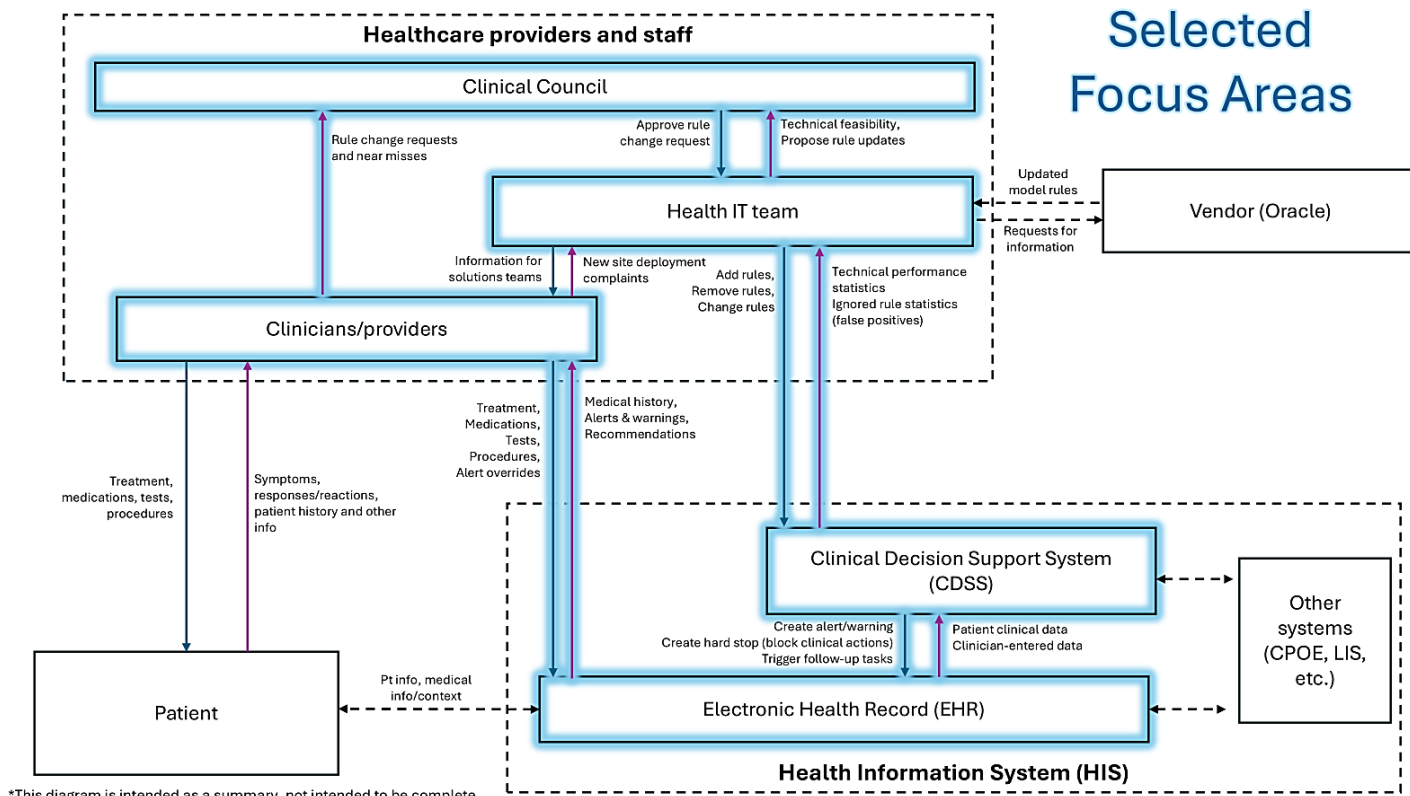


Figure 3. An iterated control structure is created with the selected research focus areas highlighted.

Description of Controllers

Controllers in the Focus Areas

Clinical Council

This is the team of clinicians within a specialty that meets regularly to review any decisions relating to their clinical specialty and health information technology. For example, clinical councils may review or provide feedback on order sets or HIT workflows. Clinical councils are also responsible for reviewing clinicians' requests for new rules. If the clinical council believes the rule is valid and useful, they pass on the official request to the health IT team to create the new rule.

Health IT Team

The health IT team is responsible for maintaining and improving the health IT systems from the care facility side. There are several subgroups within the health IT team who focus on various aspects of the HIT infrastructure. The subgroup that is relevant to this analysis is the group that updates and maintains the rule infrastructure. The main responsibilities of this subgroup are determining the feasibility of implementing the rule requests from the clinical councils and then creating the rules if they are possible to implement.

Clinicians

Clinicians are the individuals who provide medical treatment to patients. This category includes physicians and nurses.

Clinical Decision Support System (CDSS)

The CDSS system is a component of the electronic health record system. This component is responsible for firing alerts and providing guidance to clinicians based on information in a patient's health record and the existing rules infrastructure.

Electronic Health Record (EHR)

The EHR is a digital record of patient history and treatment plans. All data about a patient, including test results, visit notes, medication lists, and others, are kept in the EHR.

Controllers Beyond the Focus Area

Health IT (HIT) Vendor

The HIT vendor is the company responsible for providing the base infrastructure for the EHR. The vendor is also responsible for updating the EHR infrastructure to ensure it remains operational.

Healthcare facility administrators

The healthcare facility administrators are responsible for managing the operations of the healthcare facility. Their responsibilities include staffing allocations, budget allocations, and creating the policies that govern staff roles and responsibilities.

Regulatory Bodies

The regulatory bodies are responsible for the regulation of any tool or process used in the healthcare facility. This includes the regulation of medicines, HIT tools, hospital policies, and clinical laboratory policies.

Payors

Payors are the bodies that provide reimbursement to the care facility for care provided. This category includes private insurance providers and public insurance providers (Medicaid/Medicare).

Standards Bodies

Standards bodies and organizations set standards across the healthcare system, including the naming and coding standards that are used in HIT systems. For example, the standards body Regenstrief Institute maintains the LOINC® code set.

Health Network Administrators

These may be the high-level administrators that set policies and procedures across a wide network of healthcare facilities. They may make decisions about procurement and contracting with outside companies, including HIT vendors.

Patient

The patient is the individual receiving treatment.

Other Health Information Technology Systems

The main EHR may receive and share data with other HIT systems, including lab information systems (LIS) or computerized physician order entry (CPOE).

3.4. Unsafe Control Actions

For each controller and identified control action that was focused on for the purposes of this report, the four UCA types were analyzed to identify the contexts in which the control actions became unsafe. The full set of UCAs are included in Appendix A: List of Unsafe Control Actions. Table 1 below lists a handful of the UCAs that contributed to a range of the scenario results. Please note that neither the list in this table nor the set of UCAs in the appendix should be considered exhaustive in nature as additional analysis is required beyond the current scope of this report for a more comprehensive list to be produced.

Table 1. Consolidated list of unsafe control actions (UCAs)

ID	Controller	Control Action	UCA
1	Health IT team	Accept rule request	Health IT team does not accept a rule request when it is actually technically feasible
2	Health IT team	Write rule	Health IT team does not write the rule when the rule has been accepted by the team as a technically feasible request
3	Health IT team	Fix rule	Health IT team does not fix a rule when the rule is broken
4	Health IT team	Fix rule	Health IT team does not fix a rule when the rule is computationally intensive
5	Health IT team	Pull data/audit reports	Health IT team pulls a data audit report too late after a rule is not firing as intended
6	Health IT team	Provide human-readable version of rules	Health IT team does not provide a human-readable version of rules when users review existing rules and/or validate new rules
7	Health IT team	Implement or update model rules from EHR vendor	Health IT team does not implement or update a model rule when the EHR vendor has released a (newer version of the) model rule
8	CDSS	Generate an alert	CDSS does not generate an alert when there is too much server latency or not enough compute

9	Clinician	Provide treatment	Clinician provides treatment that has a known negative interaction with a known patient condition
10	Clinician	Order diagnostic test	Clinician does not order diagnostic test to patient when patient has known/documented symptom that standard best practice recommends testing
11	Clinician	Enter patient data into HIT system	Clinician enters patient data into the HIT system such that it cannot trigger necessary rules and alerts
12	Clinician	Order treatment	Clinician orders necessary treatment from internal provider when internal provider cannot provide the ordered treatment
13	Clinical Council	Request new rule	Clinical council does not provide new rule request when current rule does not meet updated clinical guidelines

3.5. Scenarios

From the UCAs in Table 1. Consolidated list of unsafe control actions (UCAs) above, we identified causal scenarios, many of which were discussed and originated from the interview process of subject matter experts. The following set of refined scenarios are examples selected for the health IT team, CDSS, and clinician controllers. These are selected as illustrative scenarios with systemic factors from across the control structure. The complete information for the scenarios, including the class of the scenario and the category of the UCA, can be found in

Appendix B: List of Scenarios. Similar to the UCAs, there are other scenarios that may not have been included in the scope of this study and may require further effort for additional comprehensiveness.

UCA-1: Health IT team does not accept a rule request when it is actually technically feasible.

Scenario 1: The health IT team receives a technically sound rule request from one of the clinical councils to implement a new alert. However, they deny the request. Although the inputs were adequate and accurate, the health IT team’s decision-making rationale deprioritizes rules that are not mandated by national policy, required for regulatory compliance, or deemed to have as much of an impact as another rule request. The team has an informal decision-making rationale that determines “strategic alignment,” but there is no formal prioritization criteria or process.

Scenario 2: A clinical council requests implementation of a model rule that exists within the EHR vendor’s CDSS library. The health IT team evaluates the request and decides that it is technically feasible but does not accept the rule request for implementation because no clear documentation or metadata is available from the EHR vendor about the model rule (e.g., the required data elements, rule logic, compatibility with the current system version). Feedback/information is missing from the EHR vendor about the model rule.

Scenario 3: The health IT team accepts a rule request that is technically feasible, and the CDSS platform receives the rule request acceptance. However, the CDSS rule deployment process for a specific site/facility does not receive the rule request due to the facility not being selected or captured within the rule's deployment scope. So, it is as if the rule request was not accepted by the health IT team when it was technically feasible.

UCA-2: Health IT team does not write the rule when the rule has been accepted by the team as a technically feasible request.

Scenario: The health IT team receives a request from a clinical council for a new rule with all the required information. The rule is technically feasible, and the health IT team's manager accepts the rule request. However, the health IT team's rule writers/implementers do not write the rule and instead add it to the rule request queue in a huge backlog due to other competing priorities, such as implementing federal mandates and patching system vulnerabilities. Although the health IT team intends to approve the rule, the decision is delayed by several weeks.

UCA-3: Health IT team does not fix a rule when the rule is broken.

Scenario: A rule is broken and not working as intended, but the health IT team does not know that the rule is broken because a ticket has not been submitted to report the broken rule.

UCA-4: Health IT team does not fix a rule when the rule is computationally intensive.

Scenario: The health IT team receives feedback that general lab rules put a lot of strain on the server due to volume. However, in order to consolidate labs, the machines would need to be consolidated as well, but there is too much variation within lab equipment for that to happen. So, the labs at each site continue to add their own rules, and the health IT team does not fix the general lab rules even though they know the rules are computationally intensive.

UCA-5: Health IT team pulls a data audit report too late after a rule is not firing as intended.

Scenario: The data audit report that the health IT team pulls does not indicate when a rule is not firing as intended. The log is temporary and only contains historical data going back 30 days before disappearing. If there is a problem with a rule that was deployed in the system more than one month in the past, the health IT team has difficulty in diagnosing the problem since there are no diagnostic archives. The report is not able to be used for prospective analysis of which rules are firing as intended.

UCA-6: Health IT team does not provide a human-readable version of rules when users review existing rules and/or validate new rules.

Scenario: The health IT team does not provide a human-readable version of rules when existing rules are reviewed and/or new rules need to be validated because: a) the amount of rules is overwhelming, b) there are many different stakeholders with competing interests, c) the granular fidelity for the rule is lost after one month, and/or d) alerts data is not clean.

UCA-7: Health IT team does not implement or update a model rule when the EHR vendor has released a (newer version of the) model rule.

Scenario: The health IT team receives feedback that there is a new or newer version of a model rule from the EHR vendor available. However, ingestion of the rule requires not only the rule itself, but also other packages associated with it, and the health IT team cannot modify the model rule for customizations or optimizations to specific sites or workflows because it is built by the EHR vendor. This results in the potential for the model rules to be easily broken if there is any change thereafter (e.g., to event sets, order names, code sets).

UCA-8: CDSS does not generate an alert when there is too much server latency or not enough compute.

Scenario: Due to gaps in the feedback loop (e.g., infrastructure monitoring system does not send real-time latency and compute utilization data to the CDSS, or the CDSS's process model is built to assume that compute resources are always sufficient), the CDSS does not receive feedback/input that the system is in a high-latency, resource-constrained state. However, the server latency is critically high and/or the compute capacity is insufficient. Because of this, the CDSS continues operating as though resources are adequate, does not detect the performance degradation, and does not trigger any fallback mechanisms, warnings, or prioritization of critical alerts, ultimately not generating required alerts because it silently times out or skips rule checks under the heavy load.

UCA-9: Clinician provides treatment that has a known negative interaction with a known patient condition.

Scenario 1: Clinicians may receive the same style of alert for concerns that have radically different levels of clinical concern. Therefore, an alert that may be triggered for a high-risk interaction may look no different than other alerts that have minimal clinical significance. The clinician may not realize that the alert contained useful information.

Scenario 2: The contraindicated treatment or medication the patient is currently receiving may be provided by an outside provider who codes treatment differently than the home EHR. Therefore, despite the treatment being logged in the patient file, the alerts may not identify the conflicting medication and therefore will not trigger an alert. This may be particularly dangerous when the interacting medication is treating a condition outside of the clinician's specialty and they do not regularly treat patients that have this particular interaction risk.

UCA-10: Clinician does not order diagnostic test for patient when patient has known/documented symptom that standard best practice recommends testing.

Scenario: The clinician may not trust that the alerts are valid for the patient in question. Previous rule updates have caused incorrect alerts in the past, and the clinician may have lost trust in the alert system. For example, clinicians often have to accept or sign off on rules that represent an impossible task in order to move to the next screen. As the clinicians see more alerts they believe to be incorrect, they lose trust in the HIT system to provide usable information.

UCA-11: Clinician enters patient data into the HIT system such that it cannot trigger necessary rules and alerts.

Scenario: The clinician may not receive any alerts that entering data as free text will prevent rules from firing based on that data. For example, a clinician may overestimate the EHR's ability to search through free text for medications. A medication prescribed by another physician may be mentioned by a patient and entered in free text. However, if that medication is not formally entered, it may not trigger interaction alerts.

UCA-12: Clinician orders treatment from internal provider when internal provider cannot provide the ordered treatment.

Scenario: The clinician may believe that the information shown is incorrect. The treatment may be offered on occasion in the clinic. The clinician may believe that if they can submit the order, then it is available. The clinician may not get feedback that the order was not sent to a valid recipient or that the treatment is no longer available, which could delay treatment to the patient.

UCA-13: Clinical council does not provide new rule request when current rule does not meet updated clinical guidelines.

Scenario: The clinical council may only have feedback about the functionality of previous rules they have approved/provided when they encounter them in their clinical practice. Because they submit so many rules for development, they may not track which rules were tied to certain clinical guidelines. Therefore, when clinical guidelines are updated, they may not be able to identify that the updated guidelines will impact existing rules in the system. Additionally, tracking previously submitted rules may not be in their specific list of responsibilities.

4. Discussion and Recommendations

The scenarios outlined above highlight several key systemic flaws and patterns from across the control structure. These systemic factors are clear opportunities to strengthen the safety management system (SMS) as summarized in Figure 4 below.

Systemic Factors

- Delayed / Missing Feedback: Model Rules
- Lack of Customization: Model Rules
- Reactive vs. Proactive
- Unused & Missing Feedback
- Resource Constraints
- Inadequate Oversight
- Competing Priorities
- Lack of Rule Ownership
- Missing / Inaccessible Feedback: Rule Intent & Rationale
- Inadequate Learning From Root Cause Analysis (RCA)

Opportunities to Strengthen SMS

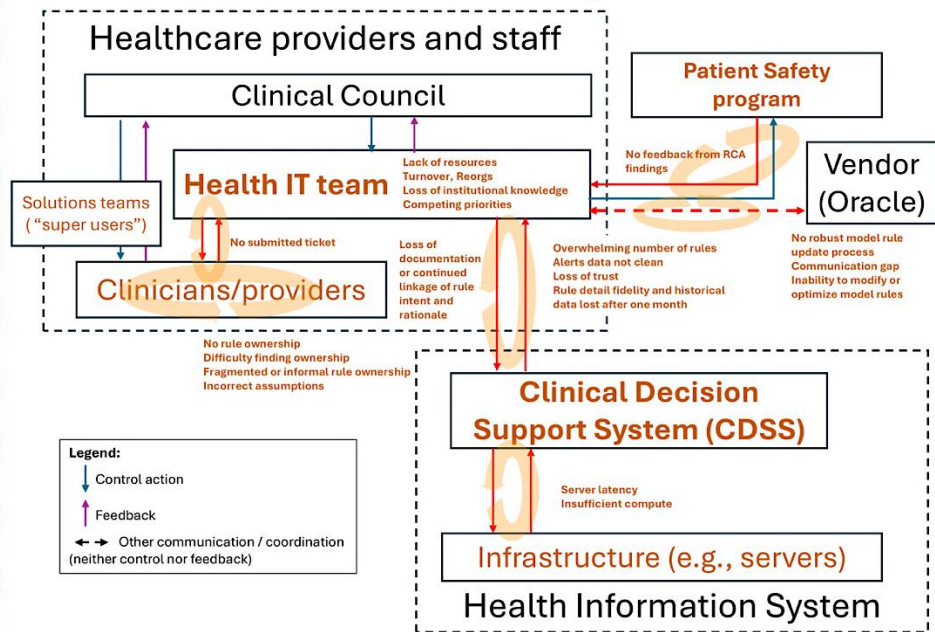


Figure 4. Systemic factors provide opportunities to strengthen the safety management system across the control structure.

These systemic causal factors are discussed in greater detail in the section below followed by recommendations generated from analysis of the scenarios above to address these systemic factors.

4.1. Systemic Factors

4.1.1. Delayed/Missing Feedback and Lack of Customization for Model Rules

Model rules are vendor-supplied clinical decision support rules that are distributed as standardized content across all the vendors' EHR sites. Model rules account for approximately one third of the total rule catalog currently in use, and the VA is in some instances several versions behind the vendor's most recent release of these model rules. This lag is attributable to a recurring communication gap between the EHR vendor and the health IT team: The vendor does not consistently provide timely or reliable notifications regarding the release of new model rules or updated versions of existing rules. As a result, critical updates are not always implemented, creating the potential for outdated or incomplete model rules to remain active within the clinical environment (Figure 5).

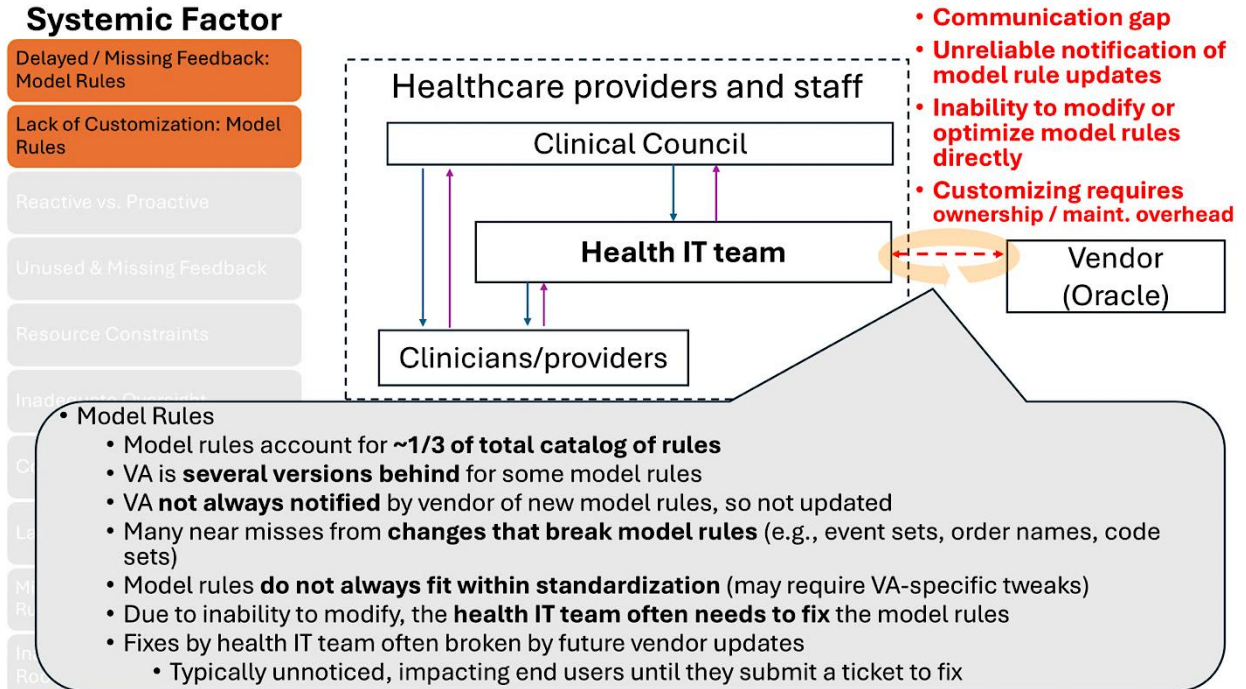


Figure 5. Two systemic factors involve delayed and/or missing feedback as well as lack of customization of model rules.

In addition, near misses have been observed in cases where changes to event sets, order names, or code sets introduced incompatibilities that disrupted the function of implemented model rules. These incidents demonstrate that rule integrity is highly sensitive to upstream vendor-driven changes.

A further complicating factor is that vendor-provided model rules often do not conform to the VA's established standardization framework. In practice, the health IT team is frequently required to "fix" model rules by developing supplemental rules to close functional gaps. This then increases the overall number of rules and introduces additional complexity into the rule catalog. When subsequent vendor updates are released, these downstream modifications can themselves break or conflict with the updated model rules.

The interaction of these issues presents a systemic vulnerability: Reliance on vendor-delivered model rules without a robust communication, validation, and adaptation process creates conditions in which the CDSS logic can drift from both the vendor specifications and clinical requirements, which increases the risk of incorrect or missing alerts at the point of care (Figure 6).

4.1.2. Reactive vs. Proactive

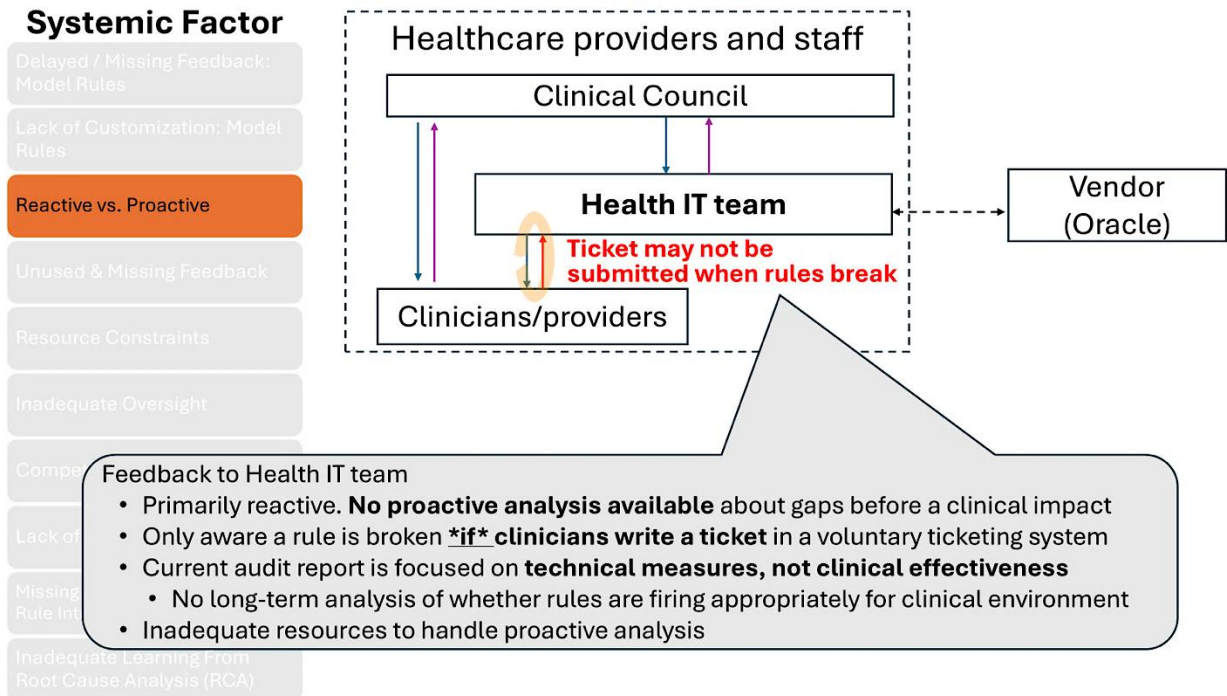


Figure 6. The reactive vs. proactive systemic factor highlights the primarily reactive nature of the current system.

The current feedback loop to the health IT team regarding CDSS rule performance is predominantly reactive. Currently, the primary mechanism through which the health IT team becomes aware of a broken (e.g., malfunctioning or misfiring) rule is the submission of a ticket by an end-user clinician. This process is voluntary, dependent on individual reporting behavior, and is therefore subject to variability and underreporting. Critically, there is no systematic or proactive monitoring in place to identify rule degradation, logic failures, or gaps in coverage prior to the occurrence of a clinical impact. As a result, issues are often discovered only after a clinician has encountered an alert failure or inconsistency during patient care. Inadequate resources, which have been separately identified as another systemic factor, have been cited as a key causal factor in this gap.

4.1.3. Unused and Missing Feedback

While rule performance data is available (e.g., alerts overridden, firing rate, adherence percentage) and interest in evaluating it exists, the health IT team is currently overextended with existing responsibilities and the large number of rules. The responsibility of evaluating rule performance data, which contains important clinically relevant insights, is outside the health IT team’s current scope and resource capacity. Therefore, feedback about the performance of the rules is not available.

There are several notable points around the rule performance data. Data on clinical performance indicators, such as false alert rates or downstream clinical impact, is not routinely monitored or used for decision-making. Historical trends are unavailable because relevant data is periodically deleted,

preventing any kind of long-term historical assessment. Moreover, no dedicated group is responsible for monitoring the current false alert rate, reducing future false alerts, reporting effectiveness metrics across organizational units, or examining secondary indicators of rule effectiveness or utility. This absence of governance and sustained data stewardship creates a systemic blind spot where ineffective rules may persist undetected and opportunities to improve rule fidelity and reduce clinician burden are not pursued (Figure 7).

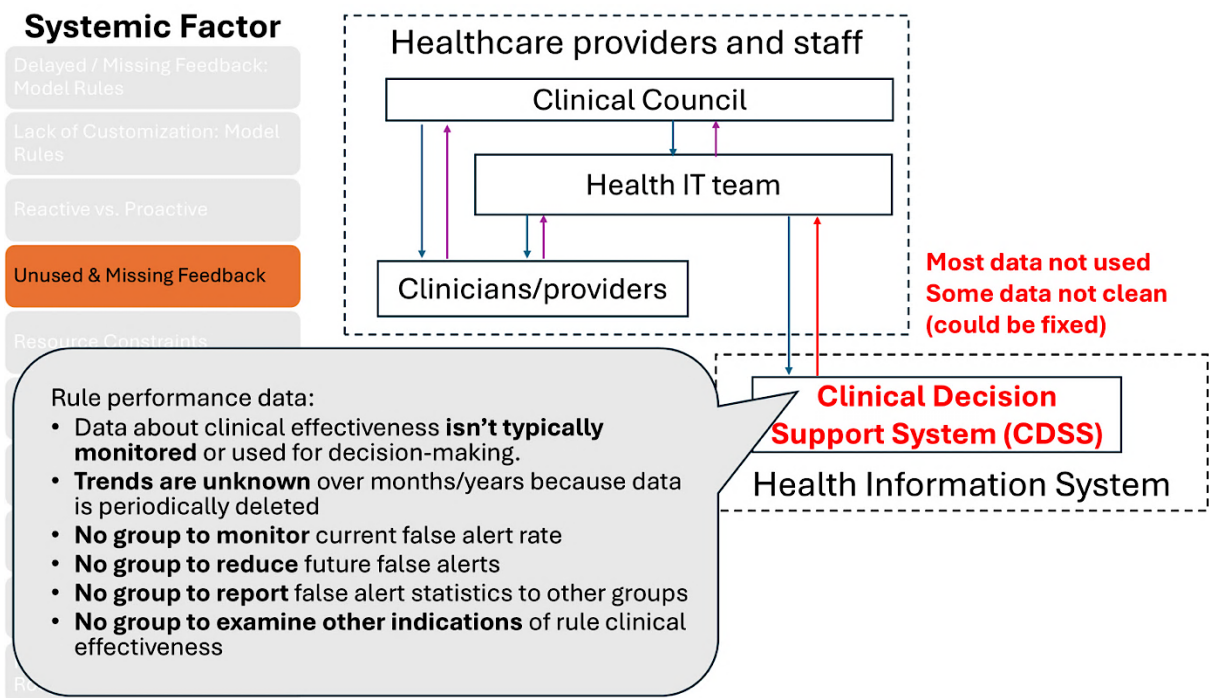


Figure 7. Unused and missing feedback systemic factor is evident by rule performance data not being used or monitored.

The figure below captures two data points that further illustrate the key points previously described, in which informational alerts with only an “OK” button are still counted as overrides. Therefore, the second data point of the alert override percentage of ~90% is not clean and can be further parsed for usable data around actionable alerts. This has safety impacts that exacerbate alert fatigue and can be addressed as a human factors issue. There are also implications to cost and productivity. The alerts are assumed to make the jobs of the clinicians easier, but the data suggests that the alerts are in fact getting in the way of clinical care and making clinicians’ jobs more difficult (see Figure 8 below).

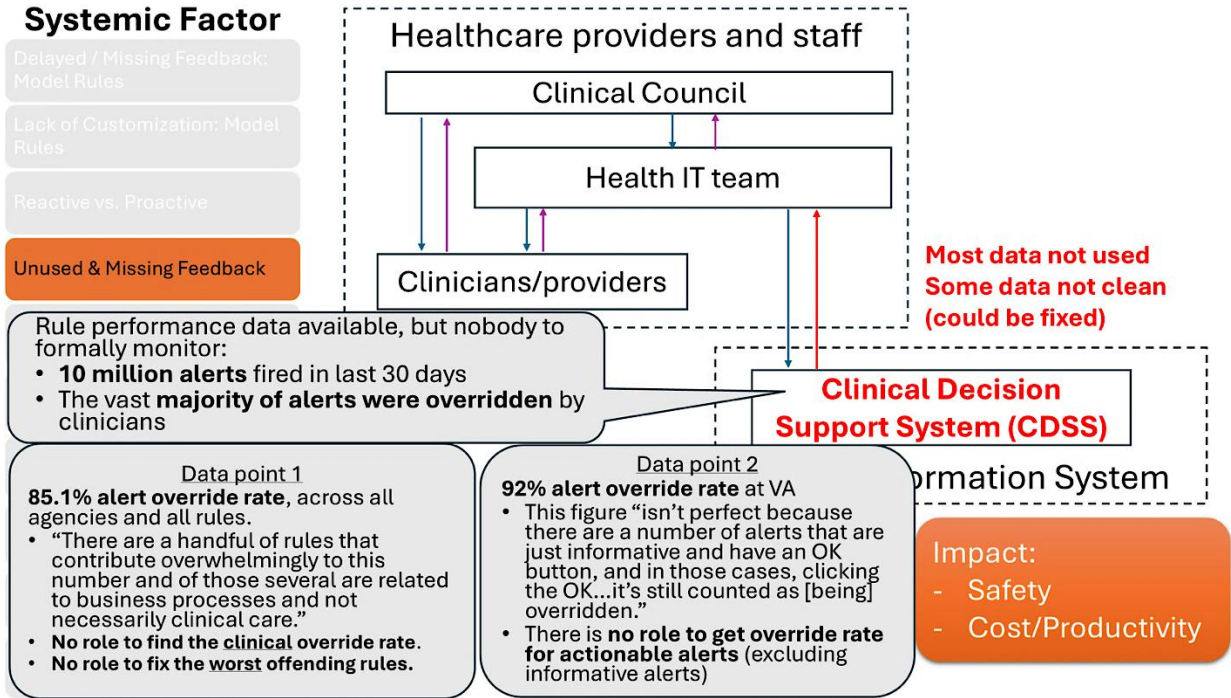


Figure 8. Two data points on alert override rates illustrate the availability of but unused nature of rule performance data.

Furthermore, to substantiate the lack of historical trends data and the resulting diagnostic data challenges, the health IT team can access and pull audit tables and rule logs, but those lookback views are temporary, disappearing after only 30 days. Because the module audit table does not get archived, with no diagnostic archives and only a simple log of popup events being accessible, diagnosing rule issues becomes difficult after 30 days (Figure 9).

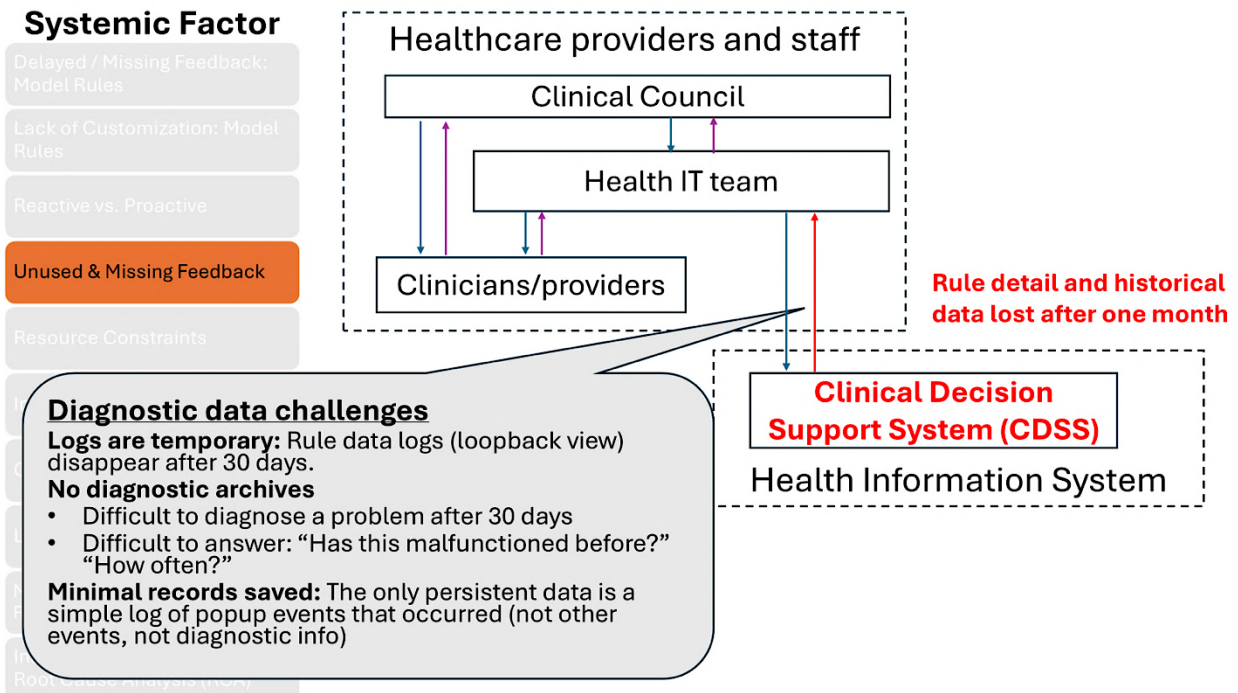


Figure 9. Issues from unused feedback include diagnostic data challenges.

4.1.4. Resource Constraints

The VA currently has approximately 4,000 rules. However, those thousands of rules are being managed by only three to four people at present. There is a significant strain from this lack of dedicated headcount, time, and capacity. High turnover and attrition along with organizational changes in the form of reorganization and policy changes also result in potential loss of institutional knowledge (Figure 10).

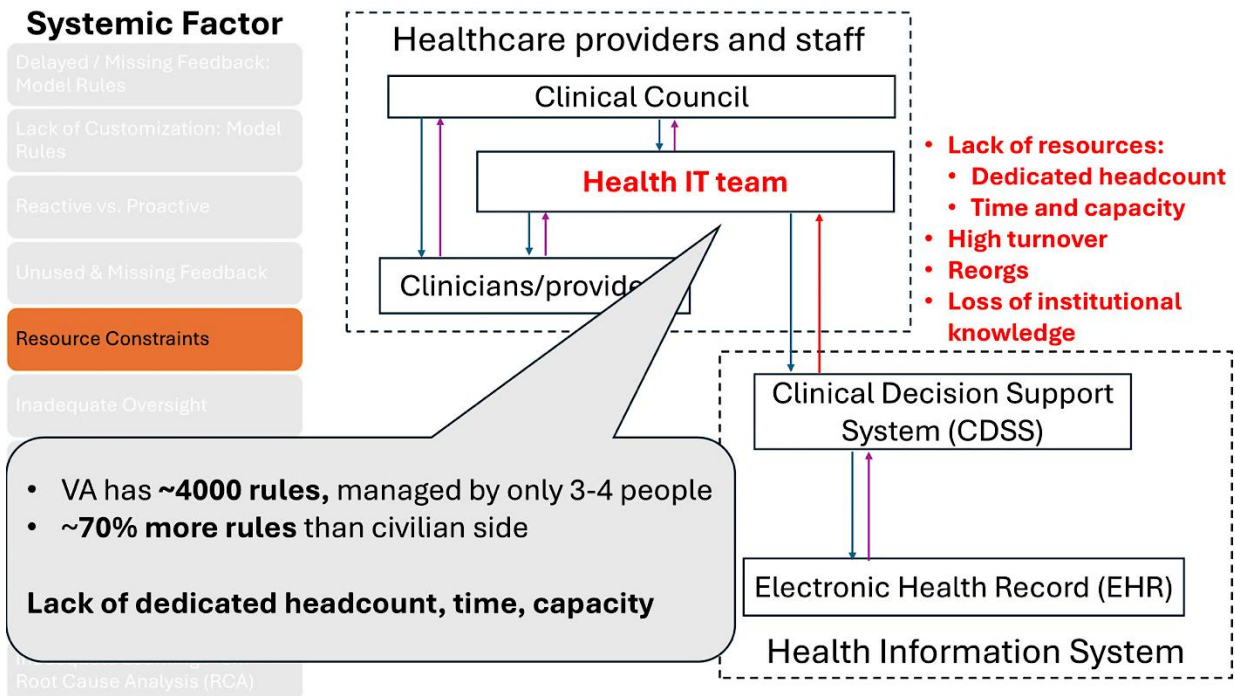


Figure 10. The resource constraints systemic factor involves people as human resources.

Resource constraints also go beyond human resources to technical resources. The current technical infrastructure presents constraints on the scalability of the CDSS. With the VA giving the go-ahead for additional sites to come online with the new EHR and as rule complexity increases, the available compute power becomes increasingly strained. Latency can also be expected to worsen as the demand on the system increases, raising the risk of delayed or even the lack of alert generation. While a planned migration to a cloud-based system may mitigate these issues, the existing environment remains as a vulnerability (see Figure 11 below).

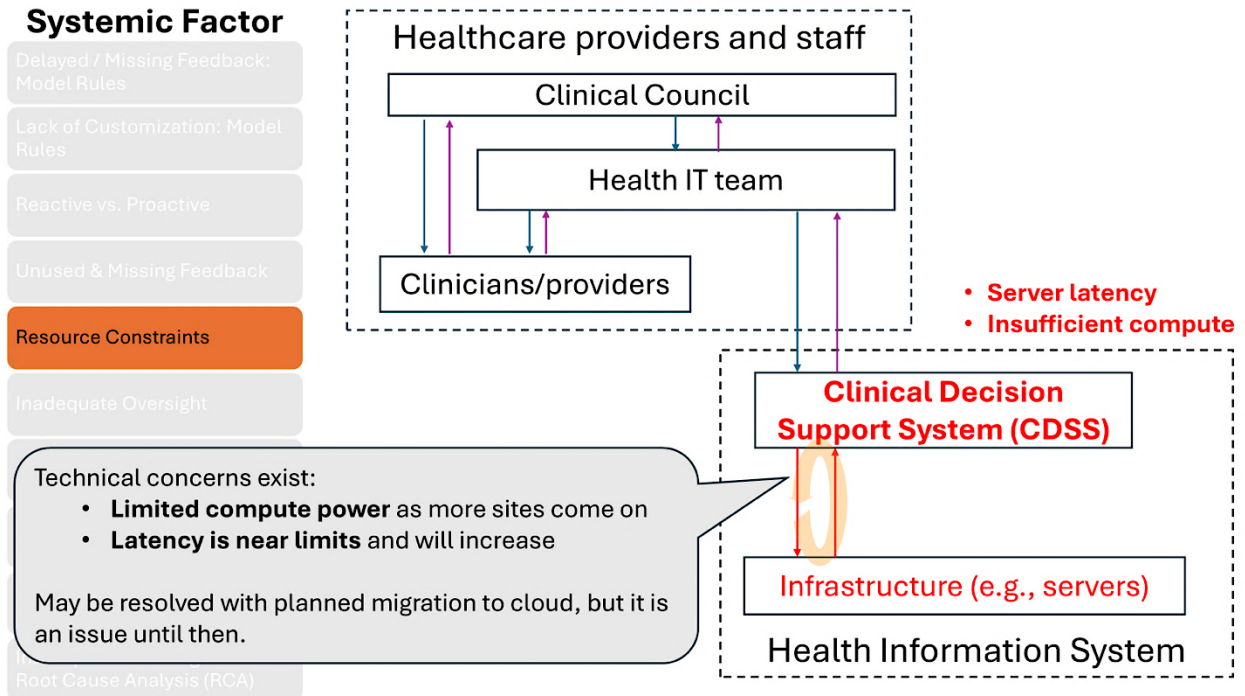


Figure 11. Resource constraints systemic factor also includes technical resource considerations.

4.1.5. Inadequate Oversight

Hazard analysis is standard practice in all safety-critical industries outside healthcare. It is far less costly to do a hazard analysis than to wait for a flaw to be discovered in operation. However, no effective hazard analysis is being performed by the vendor or VA staff to anticipate and prevent the types of software deficiencies identified by STPA. Currently, many of these software deficiencies are only being found after a CDSS malfunction or error happens in a clinical environment. This is in spite of reports, such as the survey conducted by Wright et al., in which 93% of the surveyed Chief Medical Information Officers shared that their hospital or health system experienced at least one CDSS malfunction in the last year.⁶ The vast majority of CDSS malfunctions are overlooked during rule development, design, and testing and are not found by reviewing reports of existing CDSS performance metrics; most errors are, in fact, only caught in operation when end users voluntarily decide to submit a ticket (see Figure 12 below). However, costs to fix errors substantially increases as a project matures and particularly when errors are discovered during operational settings, which can be up to more than 1500 times the cost of fixing an error compared to if the errors were discovered earlier during the requirements phase.⁷

⁶ Wright et al, 2016. "Analysis of clinical decision support system malfunctions: a case series and survey." Journal of the American Medical Informatics Association : JAMIA 23 (6): 1068-1076.

⁷ "Error Cost Escalation Through the Project Life Cycle", Haskins et al, Proceedings of INCOSE International Symposium 2004

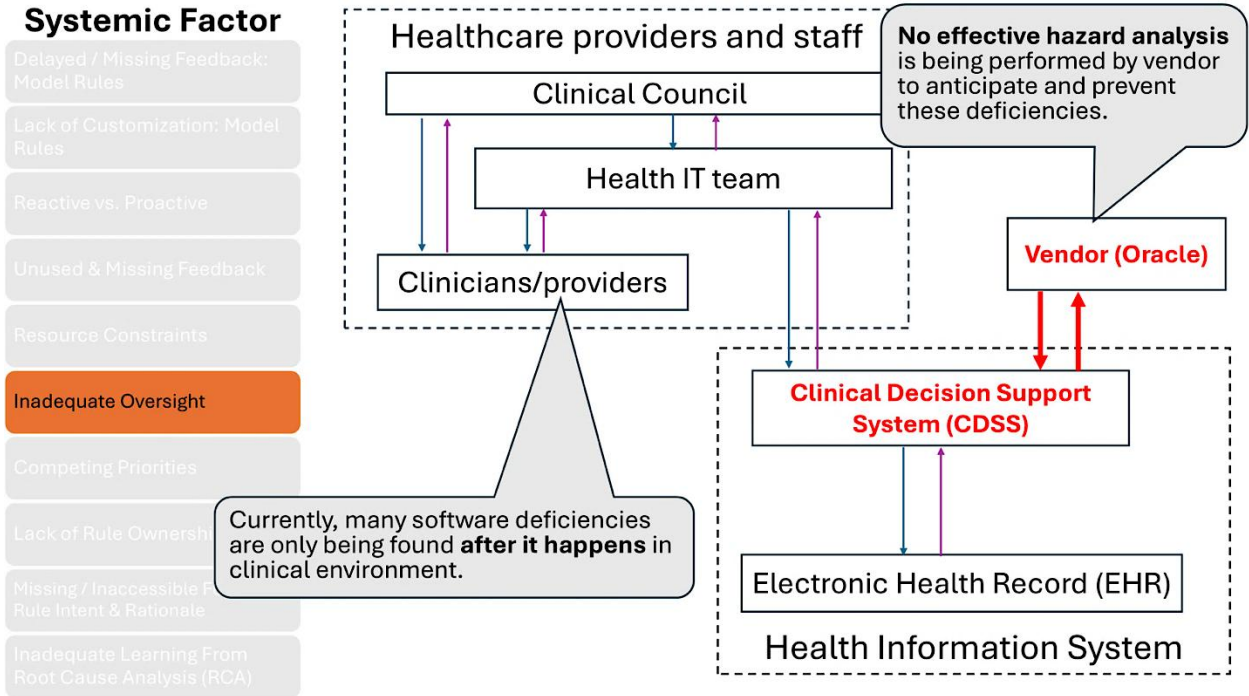


Figure 12. Inadequate oversight includes lack of hazard analysis requirements and after-the-fact identification of software deficiencies.

At present, there appears to be no government requirement for the vendor or the VA staff themselves to perform a hazard analysis before the CDSS is used to prevent some of these software deficiencies. Hazard analysis identifies deficiencies before they occur in operation and provides information to design proactive measures to prevent or mitigate the hazardous behavior (Figure 13).

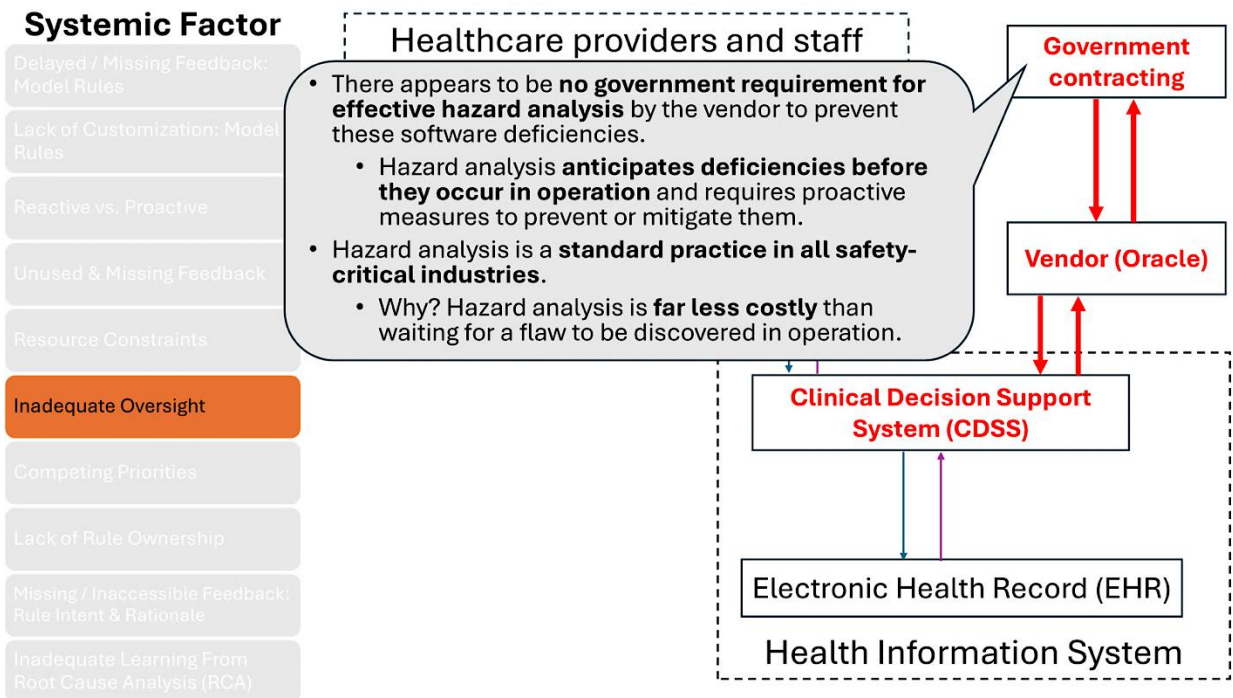


Figure 13. There appears to be no government requirement for effective hazard analysis of CDSS.

The inadequacy in oversight seems to also extend to the lack of human factors analysis. Much like hazard analysis, human factors analysis is a standard practice in safety-critical industries outside of healthcare because software without human factors considerations can increase human workload, errors, etc. Human factors analyses can capture problems like EHR alert fatigue and high false-positive rates. Information from human factors analyses can be used to produce the requirements that are needed to limit these well-known flaws. These clear requirements must be measured, monitored, and fixed ideally prior to the CDSS system operating (Figure 14).

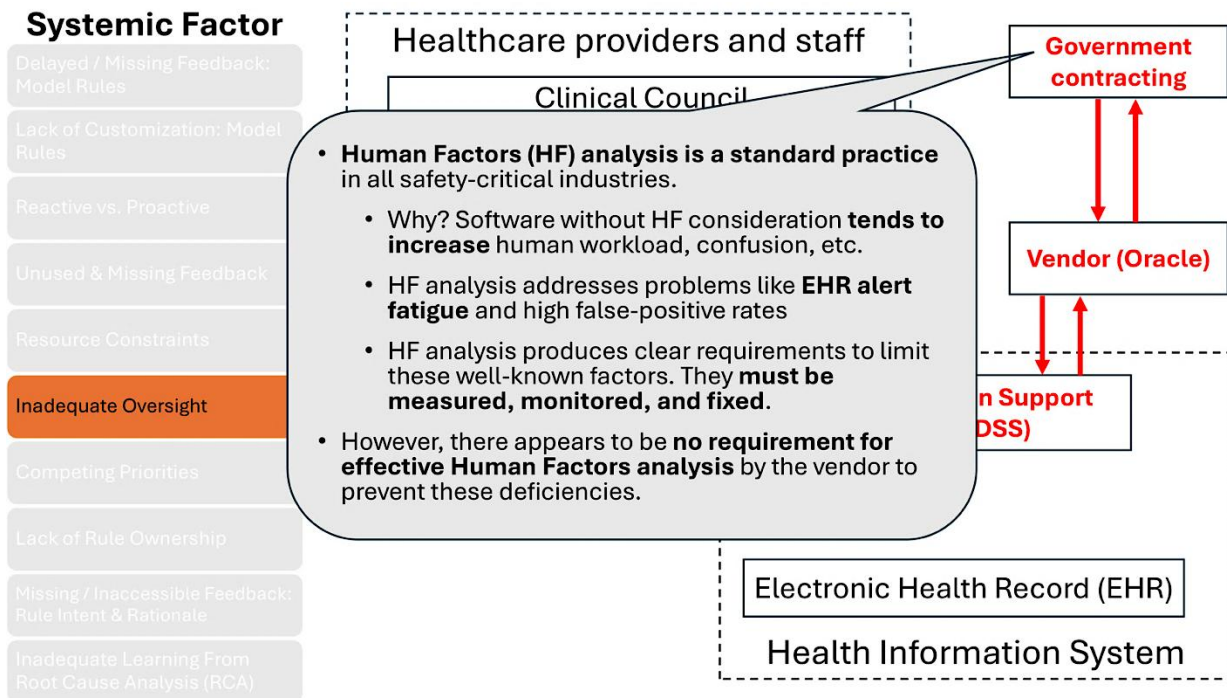


Figure 14. There appears to be no government requirement for effective human factors analysis.

4.1.6. Competing Priorities

Both within VA clinical specialties as well as between the VA and Department of Defense (DoD) shared ecosystems, competing priorities exist. Although rules have to be standardized to a certain extent across the VA and DoD, different stakeholders have disparate and competing interests. One example of competing priorities between VA clinical specialties is in rule design, such as with the “weight update” rule, which automatically updates the dosing weight data element based on the most recently measured weight. This functionality is highly valued in inpatient settings where providers often neglect to enter dosing weights, and nurses at the VA are restricted from entering in the dosing weights themselves. However, the same rule creates problems in oncology where they want a dosing weight that is locked in for patients’ chemotherapy regimens. When the rule automatically updates this weight value after a primary care visit, the oncology treatment plans become inaccurate and must then be manually re-reviewed. Conflicts and competing priorities between clinical domains like this example show how well-intentioned rules can create safety vulnerabilities when competing priorities are not reconciled (see Figure 15 below).

Systemic Factor

- Delayed / Missing Feedback: Model Rules
- Lack of Customization: Model Rules
- Reactive vs. Proactive
- Unused & Missing Feedback
- Resource Constraints
- Inadequate Oversight
- Competing Priorities**
- Lack of Rule Ownership
- Missing / Inaccessible Feedback: Rule Intent & Rationale
- Inadequate Learning From Root Cause Analysis (RCA)

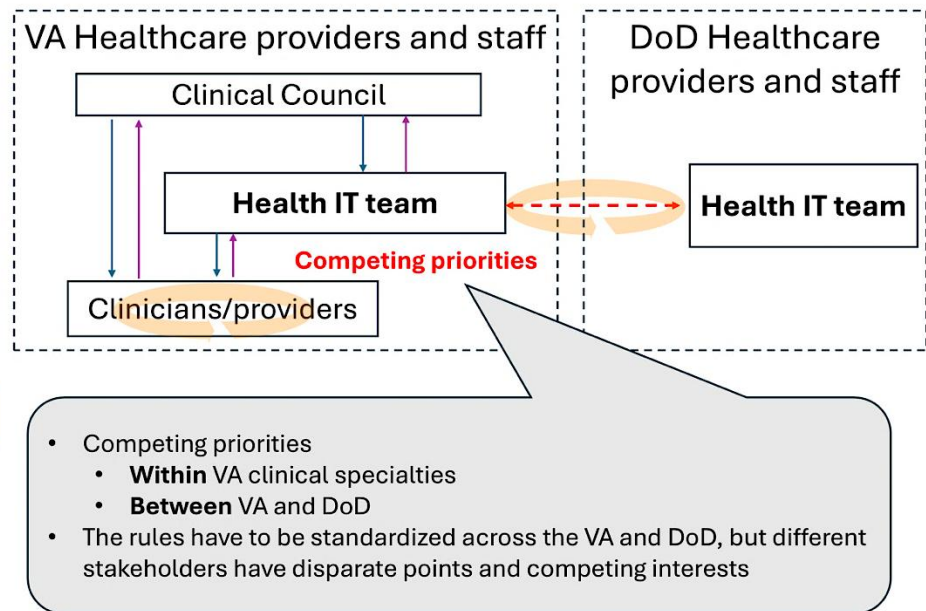


Figure 15. Competing priorities both within VA clinical specialties and between the VA and DoD exist as a systemic factor.

4.1.7. Lack of Clinical Rule Ownership

There is a missing responsibility in the sense that there is no clinical rule ownership. This is significant because there is no tracking of changes, such as clinical guidelines, that affect rules; no continuous monitoring of clinical relevance and only perhaps a periodic reassessment; no checks of rule effectiveness such as false-positive and false-negative rates; and no checks for unintended consequences after implementation. This systemic factor has been identified consistently across the set of interviews conducted for this research work as a major pain point (see Figure 16 below).

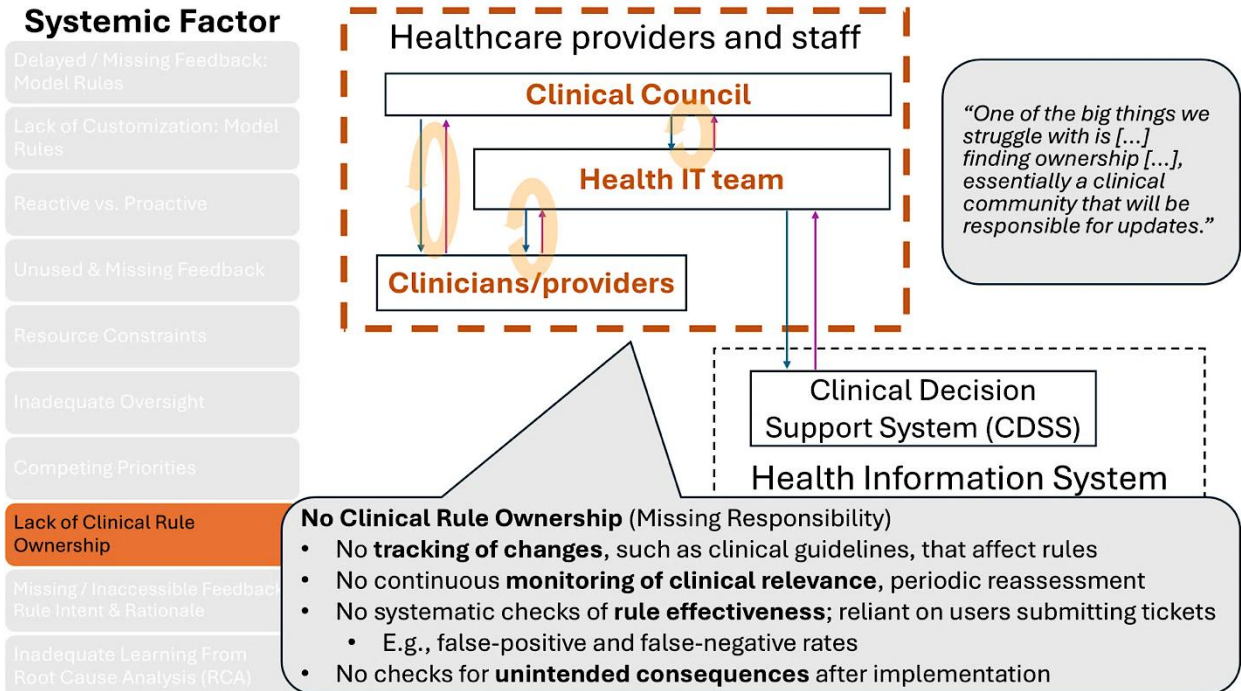


Figure 16. Lack of clinical rule ownership systemic factor

There are certain teams across the VA who serve as “super users.” They sometimes take an ownership role and coordinate with the health IT team. There are only a handful of specialties that have super users, such as the pharmacy group. However, this ownership is fragmented, purely informal and/or voluntary, and not clearly documented. In fact, many groups have incorrect assumptions about rule ownership wherein they assume the health IT team are the owners of the entire catalog of rules with the full scope of responsibilities (see Figure 17 below).

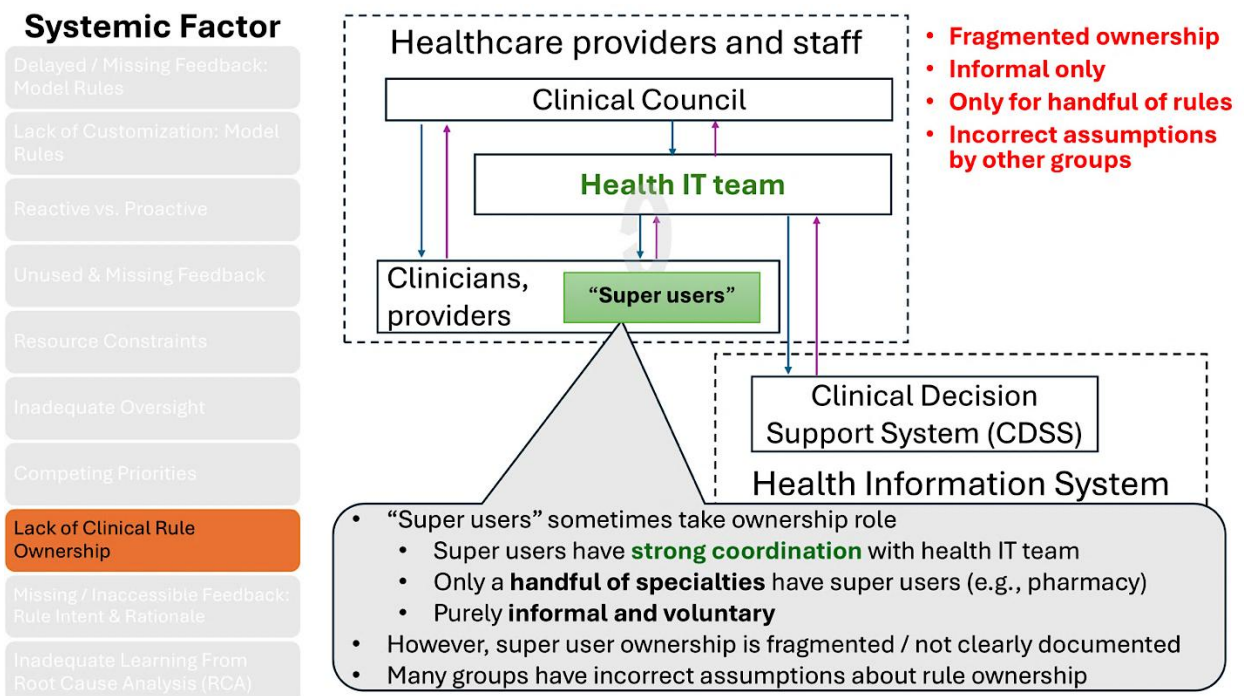


Figure 17. Clinical rule ownership as a formal, well-documented responsibility is fundamentally missing as a responsibility.

4.1.8. Missing/Inaccessible Feedback: Rule Intent and Rationale

The VA demonstrates a notable strength in that the intent and rationale of CDSS rules are carefully documented during the approval process, a practice that many other organizations struggle to maintain. However, once rules are deployed, there is no formal mechanism to link the operational rule to its original rationale. Although the documentation exists, it is not systematically associated with the live rule set. As a result, the underlying intent and reasoning cannot be readily recovered or referenced after deployment, limiting transparency, impeding rule validation, and complicating future updates or audits (see Figure 18 below).

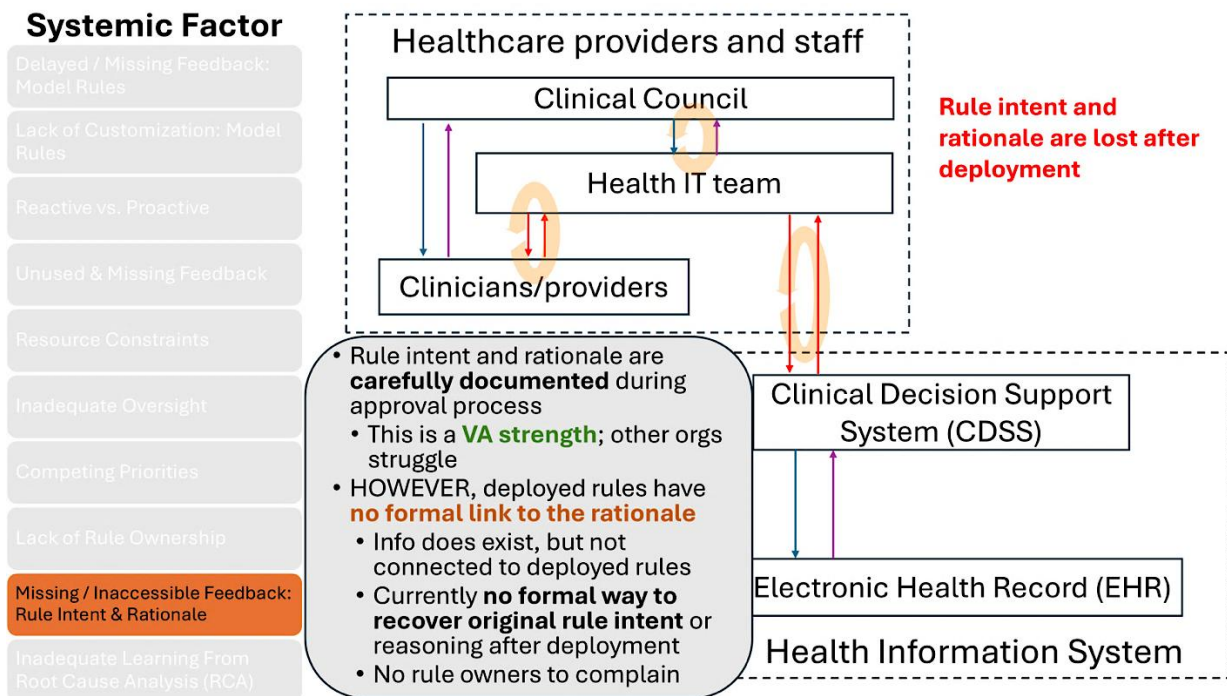


Figure 18. Systemic factor involving missing/inaccessible rule intent and rationale

4.1.9. Inadequate Learning from Past Events and Root Cause Analyses (RCA)

The health IT team does not receive structured feedback or communication from root cause analyses (RCAs), even in cases where adverse events are directly attributable to CDSS rules. There is no formal tracking mechanism to establish a historical record of rule-related adverse safety events. As a result, many gaps identified in clinical practice are either absent from RCA investigations or, when reported, are not systematically analyzed in relation to CDSS performance. This disconnect prevents organizational learning, impedes targeted improvements to decision support logic, and limits the ability to close feedback loops between safety investigations and rule management (see Figure 19 below).

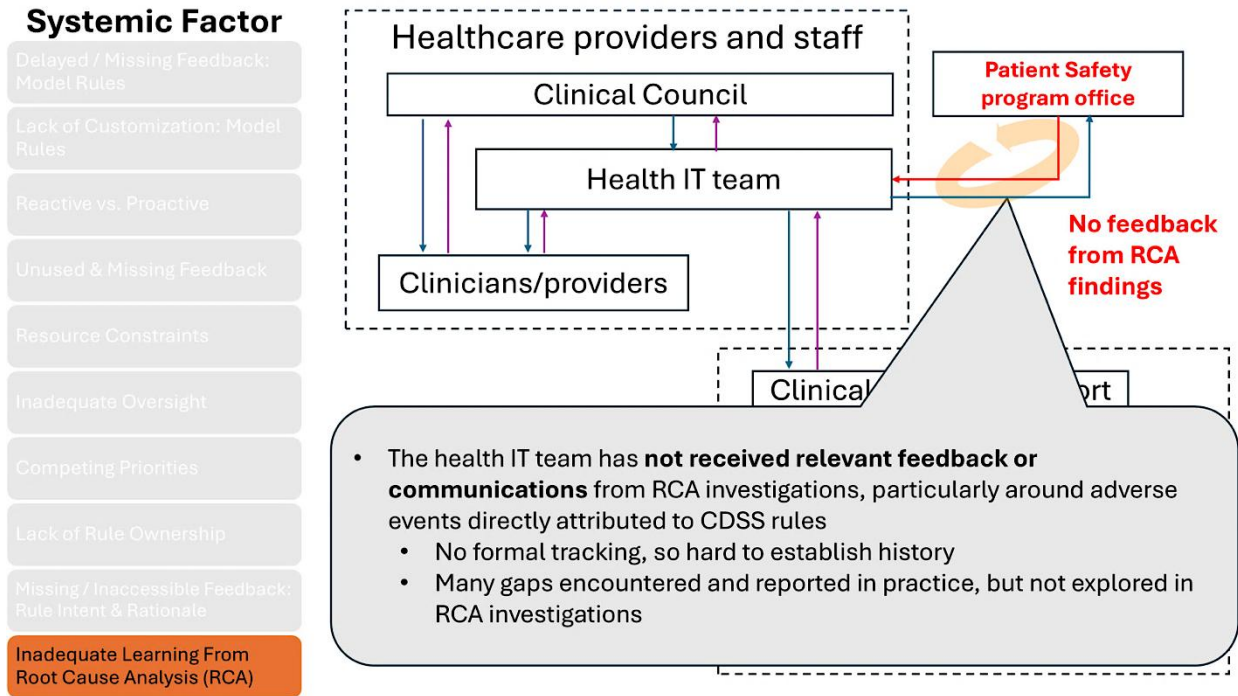


Figure 19. Systemic factor for inadequate learning from past events and root cause analyses

4.2. Recommendations

To address the systemic factors described above, we propose the following recommendations:

Recommendation 1: *Establish a clear, formal rule ownership role.*

An entity must be assigned the responsibility of owning the rules. Rule ownership must include responsibilities around monitoring and maintaining the rules after implementation. Specifically, this would include monitoring relevant clinical guidelines for changes that require rule updates, monitoring clinical effectiveness of rules, and monitoring unintended consequences of the rules. Regarding clinical effectiveness, this rule-owning entity should identify measures to evaluate clinical impact and, if a rule is determined to be ineffective, provide an appropriate response. In other words, the rule owner should create ways to analyze metrics on the catalog of rules and consider ways to collect and use feedback from clinicians on rule usefulness to guide decisions on rule refinement and reduce alert fatigue.

One responsibility of this rule owner should be to improve rule management by clearly documenting and linking rule metadata. This metadata might include the original requestor, rule rationale, clinical objective, source documentations, and change history. The metadata also needs to be documented in a structured, easily searchable format. Building and/or expanding upon reporting tools that would allow solution teams and super users to proactively analyze rules is recommended. This approach has already shown success for pharmacy teams and should continue to be expanded. Given the systemic factor meant to be addressed by this recommendation, this formal rule ownership must be established in a way that survives the organizational state of the larger

ecosystem, including surviving reorganizations, staff turnover, and changing organizational structures.

Recommendation 2: *Incorporate strong proactive measures.*

There are meaningful measures that should be implemented to move beyond after-the-fact reports of problems during clinical care and reactive change requests. These measures might include developing systems that actively monitor external changes, such as clinical guideline updates, new drug codes, and position changes, to name a few examples. A hazard analysis on the CDSS should be performed or required to reduce cost and prevent deficiencies before they have a clinical impact. To prevent software behaviors that exacerbate confusion, human error, usability issues, and workload, a human factors analysis should be conducted or required, thereby reducing cost and improving safety. Automated monitoring tools should be considered that link rules to source guidelines or terminologies, allowing for proactive identification of affected rules.

In a similar vein, coordination between the sites, working groups, and deployments should be strengthened to improve proactive alignment. This coordination should prevent rule divergence across sites by improving coordination between deployment teams and partners. There should be a proactive effort to avoid redundant or duplicate rules across VA sites.

Recommendation 3: *Improve the model rule coordination with the vendor.*

The existing communication gaps should be investigated for the reasons as to why they exist. This investigation can be done by looking into the reason behind why relevant VA teams are not always being notified of an update. Additional questions should be answered as to whether technology solutions can be created for automatic notification, as to whether the onus should be on the health IT team to reach out to the vendor and contact them first to get a status instead of waiting for information, and as to whether there are better ways for the vendor to roll out updates. As part of this recommendation, there should be a simple and easy-to-use process established for the vendor to notify the health IT team and send any model rule changes. This process should be clearly documented and used for all vendor updates.

Recommendation 4: *Ensure adequate resources.*

The health IT team must be appropriately staffed and be able to not only react to tickets and build rules based on those tickets, but also monitor what's already in the system in a prospective manner.

Recommendation 5: *Improve learning from past events and root cause analyses (RCAs).*

RCA integration with CDSS review should be improved. The entity responsible for RCAs must ensure that RCA investigations examine whether CDSS rules contributed to adverse events or could prevent similar events in the future.

5. Conclusions

The goal of this work was to identify potential weaknesses and gaps related to HIT, focusing on the VA's CDSS, and to propose actionable recommendations where there are opportunities for

improvement. The VA has a robust foundation in many areas, such as documenting CDSS rules during the approval process. However, there are weaknesses in governance, feedback, and infrastructure of the VA's current HIT system. These weaknesses, which have been identified as systemic factors, have left the CDSS operating environment vulnerable to malfunctions and errors that undermine both safety and effectiveness.

The following systemic factors were identified based on the STPA work conducted and are summarized below:

- **Unused, Delayed, and Missing Feedback**

There is a large amount of data that is available related to the VA's CDSS rules, but currently, there is no group that actively monitors that data or is responsible for ensuring the CDSS rules are useful or effective over time. There is also relevant CDSS rule audit data that is available, but those lookback views are only available temporarily, preventing long-term historical assessments. Certain CDSS data, therefore, either exists as unused feedback or becomes inaccessible.

As part of the current total rule catalog, there are also model rules, which are vendor-supplied CDSS rules. In some instances of these model rules, the VA is several versions behind the vendor's latest release of the rule due to untimely or unreliable communication with the vendor. This delay from the communication gap creates the potential for outdated or incomplete model rules to remain active within the clinical environment.

Additionally, documentation of a rule's rationale exists and is captured during the approval process of the rule, but there is currently no formal mechanism to link CDSS rules to their original intent and rationale. It is not systemically associated after the rule is deployed.

- **Reactive vs. Proactive**

The current process for monitoring CDSS rule performance is predominantly reactive. The health IT team is made aware of a rule that is broken primarily through a ticket submission mechanism. This process is entirely voluntary, dependent on individual reporting behavior, and is therefore subject to variability and underreporting.

- **Resource Constraints**

Resources both in terms of people and technical resources are constrained. For the thousands of rules that the VA currently has in its total catalog, there are currently only three to four people managing those rules. Moreover, from a technical perspective, the current infrastructure presents constraints on the scalability of the CDSS, especially with additional sites coming online with the new EHR.

- **Inadequate Oversight**

Although other safety-critical industries outside healthcare have hazard analyses and human factors analyses as standard practices, there currently appears to be no effective hazard analyses or human factors analyses being performed by the EHR vendor to identify and prevent the types of software deficiencies that are identified by STPA. There seems to be no government requirement for either the vendor or the client to perform a hazard analysis before CDSS is used or conduct a human factors analysis before CDSS is implemented.

- **Competing Priorities**

Stakeholders from different clinical specialties within the VA have differing priorities that sometimes conflict. The VA also has shared ecosystems with the DoD, which introduces additional competing priorities, particularly when it comes to CDSS rule design.

- **Lack of Clinical Rule Ownership**

There is a missing responsibility in the sense that there is no clinical rule ownership. This is significant because there is no tracking of changes, such as clinical guidelines, that affect CDSS rules; no continuous monitoring of clinical relevance and only perhaps a periodic reassessment; no checks of rule effectiveness; and no checks for unintended consequences after implementation.

- **Inadequate Learning from Past Events and Root Cause Analyses**

The health IT team does not receive structured feedback or communication from RCAs, even in cases where adverse events are directly attributable to CDSS rules. There is no formal tracking mechanism to establish a historical record of rule-related adverse safety events.

Addressing these challenges requires strengthening the safety management system, governance, resourcing, and analytical capabilities. Key recommendations include establishing formal and accountable clinical ownership of rules; expanding staffing for capacity to do more proactive rule management; requiring hazard and human factors analyses for both VA and vendor systems; and building robust monitoring tools to assess clinical effectiveness over time. Improved coordination and contracting with the EHR vendor coupled with standardized processes across the VA and DoD will reduce fragility and fragmentation. Extending diagnostic logging, linking rules to their rationale and clinical guidelines, and enhancing feedback loops between RCA findings and health IT teams will improve learning from adverse events or near misses.

Ultimately, the findings of this hazard analysis report reaffirm that the safety and effectiveness challenges in CDSS cannot be addressed through piecemeal fixes or isolated technical upgrades. They arise instead from systemic factors in feedback, governance, ownership, management, and interactions that shape how rules are designed, implemented, and maintained across the VA. As is the nature of STPA work, the recommendations outlined in this report are not about assigning fault to individuals, but about redesigning the larger system so that safe and effective behavior is the outcome. By implementing clearer ownership, stronger feedback loops, proactive monitoring, better vendor coordination, and formal hazard analysis practices, the VA can move from a reactive model to a safer one. While some of these changes will require cultural and organizational shifts, these recommendations and changes are proven approaches in other safety-critical industries. The opportunity is not simply to fix broken rules, but to change the CDSS rule management system into a foundation for improved patient safety.

6. Appendix

4.1. Appendix A: List of Unsafe Control Actions

Below is a list of all the UCAs identified during the course of this study. The UCAs described in the body of the report are indicated with a number and are highlighted.

Controller: Health IT Team

Table 2. Complete list of UCAs for health IT team controller

Control Action	Not providing causes hazard	Providing causes hazard	Too early, too late, out of order	Stopped too soon, applied too long
Accept rule request	UCA-1: Health IT team does not accept a rule request when it is actually technically feasible	UCA: Health IT team accepts a rule when the rule is not technically feasible	<p>UCA: Health IT team accepts a rule request too early before the clinical council reviews and gives approval</p> <p>UCA: Health IT team accepts a rule request too late after the clinical council reviewed and gave approval</p>	N/A
Write rule	<p>UCA: Health IT team does not write the rule when the rule has been approved by the clinical council</p> <p>UCA-2: Health IT team does not write the rule when the rule has been accepted by the team as a technically feasible request</p>	UCA: Health IT team writes the rule when the rule logic does not match with the rule's specified clinical function	<p>UCA: Health IT team writes the rule too early before getting all required clinical and governance approvals</p> <p>UCA: Health IT team writes the rule too late after it has been clinically and technically accepted/ approved as a rule request</p>	UCA: Health IT team stops writing the rule too soon before the logic is completely written and linkages all made

<p>Request the clinical communities test rule</p>	<p>UCA: Health IT team does not request testing of a rule by the clinical community when the rule is awaiting deployment</p>	<p>UCA: Health IT team requests testing of a rule by the clinical community when no clear instructions or testing scenarios have been provided</p>	<p>UCA: Health IT team requests testing of a rule by the clinical community too early before the rule logic has been fully written</p> <p>UCA: Health IT team requests testing of a rule by the clinical community too late after the rule has been implemented in the live environment</p>	<p>UCA: Health IT team stops requesting for testing of a rule by the clinical community too soon before a complete round of feedback is received</p>
<p>Deploy rule to live production environment</p>	<p>UCA: Health IT team does not deploy a rule to live production environment when the rule is ready for production</p>	<p>UCA: Health IT team deploys a rule into live production environment when the rule conflicts with existing or proposed rules or other shared ecosystem systems</p>	<p>UCA: Health IT team deploys a rule to the live production environment too early before it has been tested</p> <p>UCA: Health IT team deploys a rule out of order, deploying in the live production environment first before the development, testing, or staging environments</p>	<p>N/A</p>
<p>Fix (update) rule</p>	<p>UCA-3: Health IT team does not fix a rule when the rule is broken</p> <p>UCA-4: Health IT team does not fix a rule when the rule is computationally intensive</p>	<p>UCA: Health IT team fixes a rule when the fix introduces unintended consequences/results</p>	<p>UCA: Health IT team fixes a rule too early before clinician feedback has been validated</p> <p>UCA: Health IT team fixes rule too late after the rule has already caused high system resource utilization and degraded HIS technical performance</p>	<p>UCA: Health IT team stops fixing a rule too soon before the fix is fully implemented</p>

<p>Document clinical community ownership of a rule that is responsible for update requests for a rule</p>	<p>UCA: Health IT team does not document clinical community ownership of a rule when a rule needs to be updated based on clinical guidelines</p> <p>UCA: Health IT team does not document clinical community ownership when ownership exists</p>	<p>UCA: Health IT team documents ownership of a rule when the incorrect clinical community is assigned as the owner</p> <p>UCA: Health IT team documents clinical community ownership when ownership does not exist</p>	<p>UCA: Health IT team documents clinical community ownership of a rule too late after clinical guidance has changed and the rule is outdated</p>	<p>N/A</p>
<p>Get buy-in from clinical teams to help monitor and update rules</p>	<p>UCA: Health IT team does not get buy-in from clinical teams when a rule is being prepared for implementation/ deployment</p>	<p>UCA: Health IT team gets buy-in from clinical teams when the scope of engagement is too broad</p>	<p>UCA: Health IT team gets buy-in from the clinical teams too late after rules have been implemented or updated</p>	<p>N/A</p>
<p>Pull data/audit reports</p>	<p>UCA: Health IT team does not run data extraction jobs when rules are not deployed</p> <p>UCA: Health IT team does not pull data audit reports of rules when rules are associated with specific user roles</p>	<p>UCA: Health IT team runs technical jobs during business hours when systems are in use for clinical care</p>	<p>UCA-5: Health IT team pulls a data audit report too late after a rule is not firing as intended</p> <p>UCA: Health IT team pulls rule data audit reports too early before key data parameters are loaded</p> <p>UCA: Health IT team pulls data audit reports of rules too late after the data was needed for the solution teams</p>	<p>UCA: Health IT team runs technical jobs that are applied for too long and run continuously after reports have been generated</p>

<p>Generate documents or a dashboard for super users to see a human-readable version of rules</p>	<p>UCA-6: Health IT team does not provide a human-readable version of rules when users review existing rules and/or validate new rules</p>	<p>UCA: Health IT team generates a document with a human-readable version of rules when the document contains incorrect or outdated metadata</p>	<p>UCA: Health IT team generates a dashboard with a human-readable version of rules before the rule logic is finalized</p>	<p>N/A</p>
<p>Assign facilities to rules</p>	<p>UCA: Health IT team does not assign facilities to a relevant and deployed rule when the rule is intended to fire at specific sites</p>	<p>UCA: Health IT team assigns facilities to a rule when the facility is not appropriate for the rule's logic or target population</p>	<p>UCA: Health IT team assigns a facility to a rule too early before testing has been completed for that site</p>	<p>UCA: Health IT team stops assigning all the relevant facilities to a rule too soon before all relevant sites have been assigned</p>
<p>Write requirements about the information needed to create a new rule</p>	<p>UCA: Health IT team does not write rule requirements when input data definitions or business rules are needed for correct rule authoring</p>	<p>UCA: Health IT team writes requirements for rule requests when the requirements are too restrictive</p> <p>UCA: Health IT team writes requirements for rule requests when the requirements are too burdensome for rule requesters</p>	<p>UCA: Health IT team writes rule information requirements too early before knowing what constraints are imposed by the EHR vendor</p>	<p>N/A</p>
<p>Link a specific new rule to the rationale and other metadata</p>	<p>UCA: Health IT team does not link a new rule to rationale/metadata when users need to understand the purpose and intended usage of the rule</p>	<p>UCA: Health IT team links a new rule to rationale/metadata when it is incorrect or misleading</p>	<p>UCA: Health IT team links a new rule to its metadata too late after the rule has changed context</p>	<p>N/A</p>

<p>Implement or update model rules from EHR vendor</p>	<p>UCA-7: Health IT team does not implement or update a model rule when the EHR vendor has released a (newer version of the) model rule</p>	<p>UCA: Health IT team implements a new model rule when communication or synchronization with the EHR vendor has not occurred</p> <p>UCA: Health IT team implements or updates a model rule as-is when no local site adaptation for the site's specific workflow or patient population</p>	<p>UCA: Health IT team implements a model rule before aligning with local policies and resolving conflicts with existing rules</p>	<p>N/A</p>
---	--	--	---	------------

Controller: CDSS

Table 3. Complete list of UCAs for CDSS controller

Control Action	Not providing causes hazard	Providing causes hazard	Too early, too late, out of order	Stopped too soon, applied too long
<p>Create hard stop (block clinical action) in the EHR</p>	<p>UCA: CDSS does not create a hard stop in the EHR and allows the provider to perform a clinical action when the patient is not actively registered at the facility</p> <p>UCA: CDSS does not create a hard stop in the EHR and allows the provider to perform a clinical action when the provider lacks the required authorization</p>	<p>UCA: CDSS creates hard stop when a medically necessary treatment is ordered, preventing submission of the order</p>	<p>UCA: CDSS creates a hard stop too early before (> TBD s before) critical lab results confirming contraindication are received</p> <p>UCA: CDSS creates a hard stop too late after (> TBD s after) the clinician has already signed and submitted the order</p>	<p>UCA: CDSS continues to create a hard stop too long after (> TBD s after) the provider has documented and provided clinical justification to override</p>

<p>Generate an alert</p>	<p>UCA-8: CDSS does not generate an alert when there is too much server latency or not enough compute</p> <p>UCA: CDSS does not generate an alert for a required follow-up action when an order or note needs to be signed or lab result viewed</p>	<p>UCA: CDSS generates an alert for a required follow-up action when it takes the user to the incorrect tab to complete the action</p> <p>UCA: CDSS generates an alert when the alert is for a clinically irrelevant issue</p> <p>UCA: CDSS generates an alert when the underlying data is no longer current</p> <p>UCA: CDSS generates multiple alerts simultaneously</p>	<p>UCA: CDSS generates an alert too early before (> TBD s before) a test result is received in the patient's record</p> <p>UCA: CDSS generates an alert too late after (> TBD s after) the provider has already signed and submitted the order</p>	<p>UCA: CDSS stops generating an alert too soon before the provider is able to acknowledge the alert</p> <p>UCA: CDSS continues generating an alert too long after the provider has already acknowledged the alert</p>
<p>Suppress alert</p>	<p>UCA: CDSS does not suppress an alert when a provider has already documented an override and explanation/justification</p> <p>UCA: CDSS does not suppress an alert when the clinical context is resolved (e.g., inactive patients, already completed order)</p>	<p>UCA: CDSS suppresses an alert when the user is processing notifications and alerts due to the alert belonging to a sensitive record that the user cannot view and blocks access to that record</p> <p>UCA: CDSS suppresses a processed alert when the limit has been reached for the number of alerts in the table</p>	<p>UCA: CDSS suppresses an alert out of order before checking higher-priority alerts (e.g., suppresses a lower-priority informational alert when a critical alert is also triggered)</p>	<p>UCA: CDSS continues suppressing an alert too long after the clinical condition has changed and the alert should re-fire</p>

		<p>UCA: CDSS suppresses an alert when the provider is unavailable and the surrogate is incorrectly configured</p> <p>UCA: CDSS suppresses an alert when it was overridden once before but should be fired again</p>		
Send alert/ notification from original user to the surrogate	<p>UCA: CDSS does not send the alerts to the surrogate when the original provider is unavailable and the surrogate has not processed the alert</p> <p>UCA: CDSS does not send alert to the original user when the surrogate removes/deletes the alert</p>			
Recommend dosage	<p>UCA: CDSS does not recommend a dosage (erases all complex dosages) when a user switches tabs before finishing the order</p>	<p>UCA: CDSS recommends incorrect dosage when the user inputs a decimal without a leading zero</p>		
Substitute value for parameter	<p>UCA: CDSS does not substitute the correct value for a parameter when there is more than one value available for that parameter</p>	<p>UCA: CDSS substitutes a value for a parameter when it is not the correct value (e.g., from an incorrect document or time, from a different domain or EHR/software environment)</p>		

Controller: Clinician

Table 4. Complete list of UCAs for Clinicians

Control Action	Not providing causes hazard	Providing causes hazard	Too early, too late, out of order	Stopped too soon, applied too long
Provide Treatment to Patient	<p>UCA: Clinician does not provide treatment to patient with a known condition who requires treatment</p>	<p>UCA: Clinician provides treatment to patient that does not match patient condition</p> <p>UCA: Clinician provides treatment when there is a better treatment option available</p> <p>UCA-9: Clinician provides treatment that has a known negative interaction with a known patient condition</p> <p>UCA: Clinician provides treatment that has already been provided to patient, when multiple treatments reduces the efficacy of the treatment or has unsafe side-effects</p>	<p>UCA: Clinician provides Treatment after condition is untreatable</p> <p>UCA: Clinician provides treatment before treatment is able to be effective</p> <p>UCA: Clinician provides treatment before a required pre-step was completed</p> <p>UCA: Clinician provides treatment after a later step in the treatment has already been completed, when such an order reduces the efficacy of the treatment</p>	<p>UCA: Clinician stops providing treatment before the treatment is fully effective</p> <p>UCA: Clinician provides treatment too long after treatment was fully effective, when too long a treatment duration has unsafe side-effects</p>
Provide Diagnostic Test to Patient	<p>UCA-10: Clinician does not order diagnostic test to patient when patient has known/documented symptom</p>	<p>UCA: Clinician provides diagnostic test to patient who has already received diagnostic test when</p>	<p>UCA: Clinician provides diagnostic test too late after condition is in the most manageable stage</p>	

	that standard best practice recommends testing	diagnostic test has negative side effects	UCA: Clinician provides diagnostic test too early to detect condition of concern	
Enter Patient Data into HIT system	UCA: Clinician does not enter patient data into HIT system when data is relevant to future treatment	UCA: Clinician enters patient data into HIT system when that data is inaccurate UCA-11: Clinician enters patient data into the HIT system such that it cannot trigger necessary rules and alerts	UCA: Clinician enters patient data into HIT system after it is required for decision-making by other clinicians	
Order Treatment from external provider	UCA: Clinician does not order necessary treatment from an external provider when that treatment is not available internally	UCA: Clinician orders treatment from external provider when patient cannot access external provider UCA: Clinician orders treatment from external provider when external provider cannot provide the ordered treatment	UCA: Clinician orders treatment from external provider too late to address patient condition	
Order Treatment from internal provider	UCA: Clinician does not order treatment from internal provider when treatment is available and necessary for patient condition	UCA-12: Clinician orders necessary treatment from internal provider when internal provider cannot provide the ordered treatment	UCA: Clinician orders treatment from internal provider too late to address patient treatment	

Controller: Clinical Council

Table 5. Example of UCAs for the Clinical Council

Control Action	Not providing causes hazard	Providing causes hazard	Too early, too late, out of order	Stopped too soon, applied too long
Request new rule	<p>UCA: Clinical council does not provide new rule request when clinician requests new rule and rule would improve patient care</p> <p>UCA-13: Clinician council does not provide new rule request when current rule does not meet updated clinical guidelines</p>	<p>UCA: Clinical council provides rule request when rule already exists in system</p> <p>UCA: Clinical council provides rule request when requested rule does not reflect standard clinical guidelines</p> <p>UCA: Clinical council requests new rule that is too specific to be helpful for all clinicians who will be impacted</p>	<p>UCA: Clinical council provides request for updated rule too long after clinical guidelines have been updated</p>	

4.2. Appendix B: List of Scenarios

Controller: Health IT Team

Table 6. List of loss scenarios for health IT team controller

Unsafe Control Action (UCA)	Class of Scenario Archetype	Category of UCA	General Scenario Class (Basic Scenario)	Refined Scenario
UCA-1: Health IT team does not accept a rule request when it is actually technically feasible	Class 1: Unsafe Controller Behavior	<Feedback/input> was adequate BUT <control action> was unsafe	<ul style="list-style-type: none"> - UCA: Health IT team does not accept a rule request when it is actually technically feasible - Health IT team receives feedback/indication that 	The health IT team receives a technically sound rule request from one of the clinical councils to implement a new alert. However, they deny the request. Although the inputs were adequate and accurate, the health IT team's decision-making

			the rule is technically feasible	rationale deprioritizes rules that are not mandated by national policy, required for regulatory compliance, or deemed to have as much of an impact as another rule request. The team has an informal decision-making rationale that determines “strategic alignment,” but there is no formal prioritization criteria or process.
UCA-1: Health IT team does not accept a rule request when it is actually technically feasible	Class 2: Unsafe Feedback Path	<Feedback/input> to <controller> does not adequately indicate <context>, and <context> is true.	<ul style="list-style-type: none"> - Feedback/other inputs received by the health IT team does not adequately indicate that the rule is technically feasible. - The rule is technically feasible 	A clinical council requests implementation of a model rule that exists within the EHR vendor’s CDSS library. The health IT team evaluates the request and decides that it is technically feasible but does not accept the rule request for implementation because no clear documentation or metadata is available from the EHR vendor about the model rule (e.g., the required data elements, rule logic, compatibility with the current system version). Feedback/information is missing from the EHR vendor about the model rule.
UCA-1: Health IT team does not accept a rule request when it is actually technically feasible	Class 4: Unsafe Controlled Process Behavior	<Process> does not receive <control action> when <context>, and <process> effectively performs <control action> when <context>.	<ul style="list-style-type: none"> - Acceptance of the rule request is received by the CDSS when the rule is technically feasible - CDSS does not respond for a specific site as if the rule is not accepted 	The health IT team accepts a rule request that is technically feasible, and the CDSS platform receives the rule request acceptance. However, the CDSS rule deployment process for a specific site/facility does not receive the rule request due to the facility not being selected or captured within the rule’s deployment scope. So, it is as if the rule request was not accepted by

				the health IT team when it was technically feasible.
UCA-2: Health IT team does not write the rule when the rule has been accepted by the team as a technically feasible request	Class 1: Unsafe Controller Behavior	<Feedback/input> was adequate BUT <control action> was unsafe.	<ul style="list-style-type: none"> - Health IT team's rule writers/implementers do not write the rule when the rule has been accepted by the team's manager as a technically feasible request - Health IT team's rule writers/implementers receive feedback/indication that the rule has been accepted as a technically feasible request 	The health IT team receives a request from a clinical council for a new rule with all the required information. The rule is technically feasible, and the health IT team's manager accepts the rule request. However, the health IT team's rule writers/implementers do not write the rule and instead add it to the rule request queue in a huge backlog due to other competing priorities, such as implementing federal mandates and patching system vulnerabilities. Although the health IT team intends to approve the rule, the decision is delayed by several weeks.
UCA-3: Health IT team does not fix a rule when the rule is broken	Class 2: Unsafe Feedback Path	<Feedback/input> to <controller> does not adequately indicate <context>, and <context> is true.	<ul style="list-style-type: none"> - Feedback/other inputs received by the health IT team does not adequately indicate that the rule is broken - The rule is broken 	A rule is broken and not working as intended, but the health IT team does not know that the rule is broken because a ticket has not been submitted to report the broken rule.
UCA-4: Health IT team does not fix a rule when the rule is	Class 1: Unsafe Controller Behavior	<Feedback/input> was adequate BUT <control action> was unsafe.	<ul style="list-style-type: none"> - Health IT team does not fix a rule when the rule is computationally intensive - Health IT team received feedback/indication that 	The health IT receives feedback that general lab rules put a lot of strain on the server due to volume. However, in order to consolidate labs, the machines need to be consolidated, but

<p>computationally intensive</p>			<p>the rule is computationally intensive</p>	<p>there is so much variation within lab equipment. So, the labs at each site continue to add their own rules, and the health IT team does not fix the GL rules even though they know the rules are computationally intensive.</p>
<p>UCA-5: Health IT team pulls a data audit report too late after a rule is not firing as intended</p>	<p>Class 2: Unsafe Feedback Path</p>	<p><Feedback/input> to <controller> does not adequately indicate <context>, and <context> is true.</p>	<ul style="list-style-type: none"> - Feedback/other inputs received by the health IT team does not indicate when a rule is not firing as intended (too late) - The rule is not firing as intended 	<p>The data audit report that the health IT team pulls does not indicate when a rule is not firing as intended. The log is temporary and only contains historical data going back 30 days before disappearing. If there is a problem with a rule that was deployed in the system more than one month in the past, the health IT team has difficulty in diagnosing the problem since there are no diagnostic archives. The report is not able to be used for prospective analysis of which rules are firing as intended.</p>
<p>UCA-6: Health IT team does not provide a human-readable version of rules when users review existing rules and/or validate new rules</p>	<p>Class 1: Unsafe Controller Behavior</p>	<p><Feedback/input> was adequate BUT <control action> was unsafe.</p>	<ul style="list-style-type: none"> - Health IT team does not provide a human-readable version of rules when users review existing rules and/or validate new rules - Health IT team received feedback/indication that users review existing rules and/or validate new rules 	<p>The health IT team does not provide a human-readable version of rules when existing rules are reviewed and/or new rules need to be validated because:</p> <ul style="list-style-type: none"> - the amount of rules is overwhelming - there are many different stakeholders with competing interests - the granular fidelity for the rule is lost after one month - alerts data is not clean

<p>UCA-7: Health IT team does not implement or update a model rule when the EHR vendor has released a (newer version of the) model rule</p>	<p>Class 1: Unsafe Controller Behavior</p>	<p><Feedback/input> was adequate BUT <control action> was unsafe.</p>	<ul style="list-style-type: none"> - Health IT team does not implement or update a model rule when the EHR vendor has released a (newer version of the) model rule - Health IT team received feedback/indication that there is a new(er version of the) model rule available 	<p>The health IT team receives feedback that there is a new or newer version of a model rule from the EHR vendor available. However, ingestion of the rule requires not only the rule itself, but also other packages associated with it, and the health IT team cannot modify the model rule for customizations or optimizations to specific sites or workflows because it is built by the EHR vendor. This results in the potential for the model rules to be easily broken if there is any change thereafter (e.g., to event sets, order names, code sets).</p>
--	--	---	--	--

Controller: CDSS

Table 7. Example of loss scenario for CDSS controller

Unsafe Control Action (UCA)	Class of Scenario Archetype	Category of UCA	General Scenario Class (Basic Scenario)	Refined Scenario
<p>UCA-8: CDSS does not generate an alert when there is too much server latency or not enough compute</p>	<p>Class 2: Unsafe Feedback Path</p>	<p><Feedback/input> to <controller> does not adequately indicate <context>, and <context> is true.</p>	<ul style="list-style-type: none"> - Feedback/other inputs received by the CDSS does not indicate when there is too much server latency or not enough compute - There is too much server latency or not enough compute 	<p>Due to gaps in the feedback loop (e.g., infrastructure monitoring system does not send real-time latency and compute utilization data to the CDSS, or the CDSS's process model is built to assume that compute resources are always sufficient), the CDSS does not receive feedback/input that the system is in a high-latency, resource-constrained state. However, the server latency is critically high and/or the compute capacity is insufficient. Because of this, the CDSS continues operating as though resources are</p>

				adequate, does not detect the performance degradation, and does not trigger any fallback mechanisms, warnings, or prioritization of critical alerts, ultimately not generating required alerts because it silently times out or skips rule checks under the heavy load.
--	--	--	--	---

Controller: Clinician

Table 8. List of loss scenarios for clinician controller

Unsafe Control Action (UCA)	Class of Scenario Archetype	Category of UCA	General Scenario Class (Basic Scenario)	Refined Scenario
UCA-9: Clinician provides treatment that has a known negative interaction with a known patient condition	Class 1: Unsafe Controller Behavior	<Feedback/input> was adequate BUT <control action> was unsafe.	Clinician provides treatment that has a known negative interaction with a known patient condition when there is feedback that shows the negative interaction	<p>Clinicians are constantly bombarded with alerts as they navigate the EHR. Therefore, they may not pay as much attention to each alert and develop a habit of dismissing them in order to get the work they need to do done in a reasonable time frame. The VA systems have approximately 70% more rules than civilian side which might contribute to physicians ignoring alerts habitually.</p> <p>Clinicians may receive the same style of alert for concerns that have radically different levels of clinical concern. Therefore, an alert that may be triggered for a high-risk interaction may look no different than other alerts that have minimal clinical significance. The clinician may not realize that the alert contained useful information.</p>

<p>UCA-9: Clinician provides treatment that has a known negative interaction with a known patient condition</p>	<p>Class 2: Unsafe Feedback Path</p>	<p><Feedback/input> to <controller> does not adequately indicate <context>, and <context> is true.</p>	<p>Clinician provides treatment that has a known negative interaction with a known patient condition When there is no available feedback that shows the negative interaction</p>	<p>The clinician may order two meds that are contraindicated at the same time. The system may only check against medications that are in the record and not against those in the same order. While pharmacists may be able to catch this error, patients may not always pick up medicines from the same pharmacy, so the pharmacy is not guaranteed to know the patient is taking both medications.</p> <p>The contraindicated treatment or medication the patient is currently receiving may be provided by an outside provider who codes treatment differently than the home EHR. Therefore, despite the treatment being logged in the patient file, the alerts may not identify the conflicting medication and therefore will not trigger an alert. This may be particularly dangerous when the interacting medication is treating a condition outside of the clinician's specialty and they do not regularly treat patients that have this particular interaction risk.</p>
<p>UCA-9: Clinician provides treatment that has a known negative interaction with a known patient condition</p>	<p>Class 3: Unsafe Control Path</p>	<p><Controller> provides a safe control action but <Controlled Process> receives an unsafe control action</p>	<p>Clinician does not provide negative interacting treatment, but patient receives negative treatment.</p>	<p>A doctor may prescribe a medication that does not have contraindicated properties. However, the pharmacy may swap out a medication due to supply constraints. The medication may only have rare counterindications with treatments that the pharmacy is unaware the patient is receiving.</p>
<p>UCA-10 : Clinician does not order</p>	<p>Class 1: Unsafe</p>	<p><Feedback/input> was adequate BUT</p>	<p>Clinician does not order diagnostic testing to</p>	<p>The clinician may not trust that the alerts are valid for the patient in question. Previous</p>

<p>diagnostic test to patient when patient has known / documented symptom that standard best practice recommends testing.</p>	<p>Controller Behavior</p>	<p><control action> was unsafe.</p>	<p>patient when there is feedback indicating recommended tests.</p>	<p>rule updates have caused incorrect alerts in the past and the clinician may have lost trust in the alert system. For example, clinicians often have to accept or sign off on rules that represent an impossible task in order to move to the next screen. As the clinicians see more alerts they believe to be incorrect, they lose trust in the HIT system to provide usable information.</p> <p>There may be alerts or rules that show contradictory information. Because rules are mostly evaluated only upon creation, it is possible that a rule addition overlaps with an existing rule. The clinician may be unable to tell which rule shows the most up-to date protocol and may choose an action that they are familiar with even if it is out of date.</p>
<p>UCA-10 : Clinician does not order diagnostic test to patient when patient has known / documented symptom that standard best practice recommends testing.</p>	<p>Class 2: Unsafe Feedback Path</p>	<p><Feedback/input> to <controller> does not adequately indicate <context>, and <context> is true.</p>	<p>Clinician does not order diagnostic testing to patient when there is no feedback indicating recommended tests.</p>	<p>The rule that triggers the suggestion of a particular diagnostic test may not fire for all roles within the VA. A trainee doing the same task as a full doctor may not see the same alerts, even if their actions within the HIT system are identical.</p> <p>A clinician may not know which medications that a patient is taking are being evaluated for interactions. They may believe that the EHR is able to evaluate all medications a patient is taking, but it may only consider those prescribed within the specific VA hospital. The ability to consider all medications or conditions may be hindered because individual sites may be more easily</p>

				<p>able to create or add new code for things like allergens. Therefore, different sites may have different codes for the same medications/allergens/treatments. While there is still technically a requirement that new codes should be requested from a central group, it may not be well known or required.</p> <p>Clinicians may have requested a rule to ensure that the correct test is always suggested. However, after clinicians ask for a rule to be implemented, the request goes to clinical council, then to the IT team, (fuller). This process may take a long time, and the request may not be prioritized. There may not be feedback to the clinicians if a rule request was accepted and implemented or if it was rejected.</p>
<p>UCA-10 : Clinician does not order diagnostic test to patient when patient has known / documented symptom that standard best practice recommends testing.</p>	<p>Class 3: Unsafe Control Path</p>	<p><Controller> provides a safe control action but <Controlled Process> receives an unsafe control action</p>	<p>Clinician provides diagnostic test order, but order is not received by the appropriate party</p>	<p>The EHR may not always communicate to the user when certain actions are completed successfully. This may be particularly likely if an action has a time delay between entry and success/failure, for example the receiving clinician accepting or rejecting an order.</p>
<p>UCA-11: Clinician enters patient data into the HIT system such that it cannot trigger necessary rules and alerts</p>	<p>Class 1: Unsafe Controller Behavior</p>	<p><Feedback/input> was adequate BUT <control action> was unsafe.</p>	<p>Clinician enters patient data into the HIT system such that it cannot be easily found when there is feedback indicating</p>	<p>The only feedback clinicians may receive about ensuring data entered is easy to locate in a patient record may be in staff training. The physicians may not believe that they have time to complete all steps as</p>

			that the data will be difficult to find	shown in training or may not believe it is worth the time.
UCA-11: Clinician enters patient data into the HIT system such that it cannot trigger necessary rules and alerts	Class 2: Unsafe Feedback Path	<Feedback/input> to <controller> does not adequately indicate <context>, and <context> is true.	Clinician enters patient data into the HIT system such that it cannot be easily found when there is no feedback indicating that the data will be difficult to find	<p>The clinician may not receive any alerts that entering data as free text will make it difficult to find or use later. For example, a clinician may overestimate the EHR's ability to search free text for medications. A medication prescribed by another physician may be mentioned by a patient and entered in free text. However, if that medication is not formally entered it may not trigger interaction alerts. This may be particularly likely if a patient is utilizing non-standard medications like herbal remedies that can occasionally have severe interactions with prescription medications.</p> <p>The clinician may believe information reported by a patient is unnecessary to enter formally. However, the way the information is included in the patient record may be very difficult for future physicians to locate in the future. The patient may not repeat information to future physicians because they believe the information is already in their record.</p>
UCA-12: Clinician orders necessary treatment from internal provider when internal provider cannot provide the ordered treatment	Class 1: Unsafe Controller Behavior	<Feedback/input> was adequate BUT <control action> was unsafe.	Clinician orders treatment from internal provider when internal provider cannot provide the ordered treatment when there is feedback showing the internal	The system may not have rules set up to alert when treatments are ordered that do not exist in that facility. The healthcare organization may have some facilities that offer the treatment and others that do not. The ability to order the treatment may exist on all computer systems even when not applicable. There may be other documents

			provider cannot provide the treatment.	that show what is orderable in the facility, but the clinician may not know where to look for that information when putting in an order. This may be especially true after any changes to what a particular facility offers.
--	--	--	--	--

Controller: Clinical Council

Table 9. List of loss scenarios for the Clinical Council

Unsafe Control Action (UCA)	Class of Scenario Archetype	Category of UCA	General Scenario Class (Basic Scenario)	Refined Scenario
UCA-13: Clinical council does not provide new rule request when current rule does not meet updated clinical guidelines.	Class 1: Unsafe Controller Behavior	<Feedback/input> was adequate BUT <control action> was unsafe.	Clinical council does not provide a new request when there is feedback showing that the current rule does not meet updated clinical guidelines.	<p>The clinical council may have been consulted on the original alert/rule. There may be feedback that exists in the system that shows that the rule is alerting in inappropriate contexts, but there may be no one on the clinical council who is responsible for checking that data. Clinicians on the clinical council may even receive inappropriate alerts or CDSS interactions but do not have the time to initiate a rule update or may not realize that the rule needs to be updated.</p> <p>The clinical councils may have provided early feedback to the HIT vendor for CDSS behavior or EHR functionality. However, they may have signed off at a very early point in the development process. For example, they may sign off on a “workflow.” without the context of what the feature will look like in the real system. The clinical councils are not always familiar with the specifics of the EHR system in question, which makes it more</p>

			<p>difficult for them to imagine how a feature will be implemented without a concrete example. Therefore, the way the rule is implemented may not be what the clinical council anticipated. However, because of the earlier sign-off, the rule may be implemented with no further checks from clinicians.</p> <p>The EHR vendor may have provided some clinical council members with a fake account to test with. However, any patients or data they store for testing purposes are wiped every time an update occurs which makes testing time intensive even if it is possible.</p> <p>Because the EHR is used in many different contexts, many of the rules or alerts must be kept vague in order to be generally applicable. For example, clinical councils need to decide on what orders are seen by all clinicians in that sub-specialty. Generic alerts that do not apply in all cases they appear in may contribute to alert fatigue and distrust in the system. There may be no one monitoring how often a rule alerts in an appropriate context versus inappropriate context even if the data is available.</p>
--	--	--	--