FAILURE SEMANTICS: WHY SYSTEMS FAIL DESPITE CORRECT DIAGNOSTICS

Anand Wanjari

Independent Researcher, USA

ABSTRACT

Modern complex systems increasingly demonstrate a paradox: they detect faults with high accuracy yet still experience unsafe, degraded, or mission-impacting failures. This paper introduces Failure Semantics as a missing systems-engineering construct that explains why correct diagnostics do not guarantee system correctness. It argues that failures emerge not from detection errors but from semantic mismatches between detection, interpretation, decision, and action layers. Drawing on diagnosability theory, resilience engineering, and functional safety research, the paper presents a taxonomy of semantic failures and proposes a layered Failure Semantics Framework comprising Detection, Interpretation, Decision, Action, and Feedback & Learning layers. By enforcing semantic contracts and leading semantic health indicators, the framework enables context-aware interpretation, intent-aligned decisions, and proactive fault management beyond reactive diagnostics.

KEYWORDS

Failure Semantics, Systems Engineering, Functional Safety, Diagnostics, Resilience, Layered Architecture

1. INTRODUCTION

Modern engineered systems employ increasingly sophisticated diagnostic techniquesincluding model-based reasoning, machine learning, and dense sensor networksto detect faults with high accuracy. Yet across multiple domains, systems continue to experience unsafe, degraded, or mission-impacting failures even when underlying faults are correctly detected. These failures arise not from detection gaps, but from semantic mismatches between fault detection, interpretation, decision-making, and action, amplified by temporal dynamics, context loss, human factors, and lifecycle evolution. To address this gap, this paper introduces Failure Semantics as a formal construct explaining why correct fault detection does not ensure correct system response. Failure semantics is defined as the contextual interpretation of detected conditions in terms of severity, urgency, scope, and required response, and is supported by a taxonomy of semantic failure modes grounded in diagnosability theory, resilience engineering, and functional safety research [10–12, 18]. A layered semantic framework is proposed to align diagnostics, decision-making, and system intent across the system lifecycle, shifting the focus from whether faults are detected to whether systems respond correctly to what they detect [5, 14, 15, 17–20].

David C. Wyld et al. (Eds): CSML, AISCA, DNLP, SOEA, NET, BDHI, SIPO – 2026
pp. 92-106, 2026. CS & IT - CSCP 2026 DOI: 10.5121/csit.2026.160208

2. BACKGROUND AND DEFINITIONS

2.1. Fault, Error, Failure Revisited

The foundational terminology for reasoning about system failures originates from dependability theory, particularly the work of Avizienis, Laprie, and colleagues on dependable computing [17], with later refinements addressing diagnosability and observability by Bozzano [12] and Gao et al. [1]. Within this framework, a *fault* is a defect or condition in hardware, software, design, or the environment that can lead to incorrect behavior and may be permanent, transient, or latent [1, 17]. Activation of a fault produces an *error*, defined as an incorrect internal system state that may propagate through components or be masked by redundancy [12, 17]. A *failure* occurs when such an error becomes externally observable, resulting in service interruption, incorrect output, or violation of safety or mission requirements [17]. Together, the fault–error–failure chain describes a causal progression that underpins reliability engineering, safety analysis, and fault-tolerant design [1, 17]. However, this chain characterizes *how* failures occur, not *how systems should respond* once faults are detected, and therefore provides limited guidance on response correctness.

Fault detection, while necessary, is insufficient for failure prevention because detection alone does not ensure that a system can interpret and act on fault information appropriately. A detected fault may be identified too late for effective mitigation or followed by delayed responses due to processing or communication latency [12]. The meaning of a fault is highly context-dependent, varying with operational mode, mission phase, environmental conditions, and system history [13]. Moreover, a system may lack the actuators, redundancy, or operational margin required to execute the intended response [4], or may prioritize one objective, such as hardware protection, at the expense of safety or mission success [21]. Interactions among multiple concurrent faults further invalidate single-fault response assumptions [12], while human operators may interpret diagnostic outputs differently from designers' intentions, leading to inappropriate manual interventions [14, 15].

2.2. What Are Failure Semantics?

This paper defines **Failure Semantics** as the contextual interpretation of a detected condition that determines its meaning in terms of severity, urgency, scope, and required response within the operational and mission context of a system. Severity reflects how seriously a condition threatens safety, mission success, or efficiency and is not intrinsic to the fault itself, but dependent on system state, redundancy, and operational margins [2]. Urgency captures how rapidly the system must respond, shaped by fault propagation dynamics, time to failure, and available intervention windows [12]. Scope determines which parts of the system are affected or at risk and whether local mitigation is sufficient or coordinated system-level action is required [13]. The required response must be feasible, effective, and aligned with system intent, ensuring that safety and mission objectives are preserved rather than inadvertently compromised [23]. Failure semantics is inherently contextual, as identical fault codes can have markedly different meanings depending on operational mode, mission phase, environmental conditions, and system history.

Modern diagnostic systems typically achieve **syntactic correctness**, meaning that faults are accurately detected, coded, and reported according to predefined standards such as SAE J1939, OBD-II, or IEC 61508 diagnostic metrics [10]. However, syntactic correctness does not guarantee **semantic correctness**, which requires that diagnostic outputs are interpreted and acted upon appropriately to achieve system-level objectives. Traditional fault-management architectures are therefore **diagnostics-centric**, prioritizing detection coverage, fault isolation

accuracy, and reporting fidelity through methods such as FMEA, FMECA, and HAZOP [2]. Faults are treated as discrete events that trigger logging or predefined responses, implicitly if correct detection ensures correct behavior. In contrast, a **semantics-centric** design treats fault management as an intent-aware reasoning process in which detected conditions are interpreted in context and responses are selected to balance safety, mission success, efficiency, and resource constraints. Achieving this shift requires explicit modeling of system intent [21], context-aware interpretation through state estimation and environmental awareness [13], temporal reasoning that accounts for fault evolution and response latency [12], semantic contracts between system layers that convey meaning rather than raw codes [19], and human-centered diagnostic outputs aligned with operator mental models [14, 15]. The remainder of this paper builds on these principles to demonstrate why diagnostics-centric approaches are insufficient and how semantics-centric design can be systematically integrated into systems engineering practice.

3. THE FAILURE GAP: WHERE DIAGNOSTICS BREAK DOWN

The **failure gap** is the conceptual and operational space between correct fault detection and correct system response. It is in this gap that systems fail despite accurate diagnostics. It identifies four primary mechanisms through which the failure gap manifests: detection-response decoupling, temporal semantics failure, context loss across system boundaries, and human-system semantic mismatch.

3.1. Detection–Response Decoupling

In complex systems, fault detection and fault response are frequently designed, implemented, and validated in isolation by different engineering disciplines, each optimizing for local objectives such as observability, control stability, or hazard mitigation. While this separation of concerns is necessary to manage complexity, it creates significant opportunities for semantic misalignment. As a result, systems may detect faults with high accuracy yet respond incorrectly by over-protecting (unnecessary shutdowns), under-protecting (continued operation under unsafe conditions), or mis-protecting (executing responses suited to different fault conditions) [4, 13, 23]. These failures arise because response logic embeds latent assumptions about fault meaning assumptions that may hold under nominal conditions but become invalid as configurations change, operating environments evolve, software is updated, or multiple faults interact [12]. When such assumptions are violated, the system behaves as designed but produces semantically incorrect outcomes.

Temporal factors further exacerbate semantic failures. The significance of a detected fault depends not only on what is detected but also on when detection and response occur relative to fault propagation. Faults may be detected too late for effective mitigation, responses may be delayed by processing, communication, or actuation latency, or recovery actions may be initiated prematurely before underlying causes are resolved, leading to oscillatory or deferred failures [12, 13]. Static diagnostic thresholds compound this problem by failing to adapt to dynamic operating conditions, producing false positives during benign transients or false negatives during atypical but valid operation [1]. Context loss across subsystem boundaries also contributes to semantic failure, as local anomalies may be benign, compensated, or irrelevant at the system level, yet are interpreted conservatively due to missing global context [13]. This issue is reinforced by interface specifications that define syntactic data exchange without conveying semantic meaning, forcing subsystems to infer intent from incomplete information [19].

Human–system interaction introduces additional semantic risk. Operators and technicians often interpret diagnostic outputs differently than designers intended, influenced by training,

experience, user-interface design, and cognitive biases such as confirmation, recency, availability, and automation bias [14, 15]. Diagnostic interfaces that emphasize fault codes and severity indicators without contextual explanation amplify ambiguity, increasing the likelihood of inappropriate manual intervention. Together, these factors demonstrate that semantic failures emerge not from incorrect diagnostics, but from misalignment between detection, interpretation, decision-making, action, and human understanding even when fault detection itself is correct.

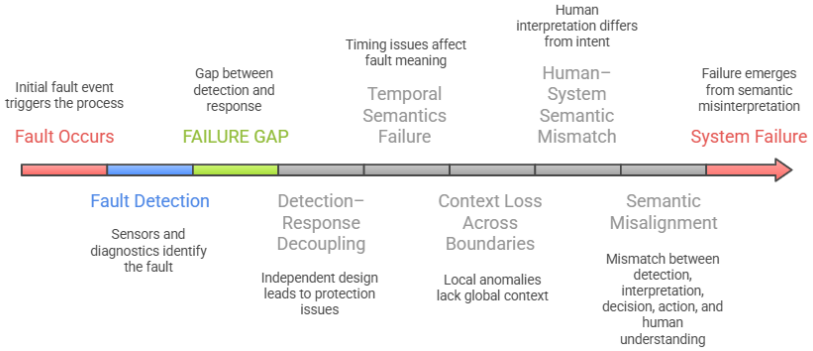

Figure 1: System Failure – A step by step analysis

4. TAXONOMY OF FAILURE SEMANTICS IN COMPLEX SYSTEMS

This section presents a taxonomy of semantic failure modes that explain how systems fail despite correct diagnostics, grounded in diagnosability theory [12], failure-mode reasoning [22], and resilience engineering [18, 20,21]. **Severity semantics failures** occur when fault severity is misjudged, either triggering excessive responses to minor faults or insufficient responses to catastrophic conditions due to conservative or optimistic assumptions that ignore context and evolving system state [2, 21]. **Scope semantics failures** arise when the perceived impact of a fault does not reflect its actual propagation potential, leading either to unnecessary system-wide shutdowns or to the masking of global degradation as localized anomalies, thereby enabling cascading failures [12, 13]. **Intent semantics failures** occur when fault responses optimize a single objective, such as hardware protection or regulatory compliance, while violating higher-level safety or mission goals, highlighting the tension between prescribed responses and real-world operational demands [10, 11, 21]. Finally, **recovery semantics failures** result from premature or incomplete recovery actions that mask unresolved root causes, defer consequences, and increase the likelihood of repeated or catastrophic failures, particularly in systems that prioritize availability over diagnostic depth [1, 13].

5. WHY CORRECT DIAGNOSTICS STILL FAIL SYSTEMS

Having characterized the failure gap and the associated semantic failure modes, this section explains why correct diagnostics are insufficient for ensuring system correctness. The root causes lie in the intrinsic complexity of modern systems, the limitations of prevailing diagnostic paradigms, and the effects of system evolution over the operational lifecycle. Most diagnostic systems are fundamentally rule-based, relying on predefined logic derived from FMEA, FTA, and expert knowledge to associate detected conditions with specific faults and responses [2]. Such approaches are effective in deterministic environments where cause–effect relationships are linear, faults occur independently, system state is fully observable, and operating conditions remain within design assumptions. However, contemporary systems operate in non-deterministic environments characterized by non-linear dynamics, interacting faults, partial observability, and

emergent behavior that cannot be predicted from component specifications alone [1, 12, 19]. In these contexts, rule-based diagnostics may correctly detect individual faults yet misinterpret their system-level implications, resulting in semantically incorrect responses.

Model-based diagnostics seek to address these limitations by reasoning over explicit behavioral models to infer faults from deviations between observed and predicted behavior [1, 3]. While this approach enables physically interpretable root-cause analysis, formal diagnosability assessment [12], and reasoning about fault propagation, it is constrained by model completeness, modeling accuracy, computational complexity, and the need to continuously maintain models as systems evolve [1, 3, 16]. As highlighted by Gao et al. [1], both model-based and signal- or data-driven diagnostic paradigms remain limited in their ability to translate fault detection into actionable semantic interpretation. Data-driven diagnostics, particularly those based on machine learning, further scale diagnostic capability by identifying complex patterns in sensor data [3, 7], but introduce additional challenges, including lack of causal explanation, brittleness under distribution shift, and the risk of silent failures high-confidence yet incorrect inferences that can lead to safety violations [3, 7]. Yang and Wang [3] demonstrate this phenomenon in autonomous systems and show that even when runtime monitoring and formal verification are applied, semantic interpretation remains a fundamental challenge.

These limitations are compounded by assumption drift over the system lifecycle. Diagnostic logic and response strategies embed design-time assumptions regarding operating ranges, fault independence, reset effectiveness, and environmental conditions, which may become invalid due to usage evolution, aging effects, configuration changes, and software updates [16]. Although diagnostic algorithms continue to function as designed, the meaning of their outputs degrades as underlying assumptions no longer hold. Modern systems further exacerbate this issue through high configurability, where software updates, parameter adjustments, and hardware replacements are often introduced independently by different stakeholders without corresponding updates to the semantic models underlying fault management [16]. Finally, toolchain fragmentation across design, diagnostics, control, safety analysis, and service domains introduces additional semantic mismatches, as each tool employs its own ontology and assumptions [2, 10, 11, 14, 19]. As noted by Häring [2], failure analysis artifacts such as FMEA tables rarely maintain formal semantic links to control logic and operational procedures, leading to situations in which individual tools produce correct outputs, yet the integrated system exhibits semantically incorrect behavior due to the absence of a shared semantic framework.

6. A FAILURE SEMANTICS FRAMEWORK

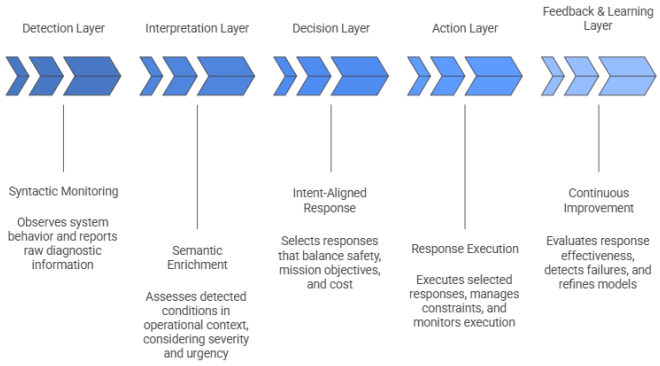

Figure 2: Failure Semantics Framework Layers and Processes

To address the failure gap and mitigate the semantic failure modes identified earlier, this paper proposes a **Failure Semantics Framework** composed of five layers connected through explicit semantic contracts. The framework guides the design of fault-management systems toward **semantic correctness**, ensuring not only accurate fault detection but also context-aware interpretation, intent-aligned decision-making, and effective response.

The framework comprises five layers with distinct roles. The **Detection Layer** ensures syntactic correctness by observing system behavior and reporting raw diagnostic information using conventional techniques such as model-based detection, data-driven anomaly detection, and signal processing [1, 3, 7]. The **Interpretation Layer** provides semantic enrichment by assessing detected conditions in operational context, accounting for severity, urgency, and scope through context-aware reasoning methods such as Bayesian inference [14]. The **Decision Layer** embodies system intent by selecting responses that balance safety, mission objectives, availability, and cost, potentially using multi-objective optimization or decision-theoretic approaches [21]. The **Action Layer** executes selected responses while managing feasibility constraints, monitoring execution, and invoking fallback strategies as needed. The **Feedback & Learning Layer** closes the loop by evaluating response effectiveness, detecting semantic failures, identifying assumption drift [16], and refining models and procedures using operational data and human-factors insights [14, 15].

A key contribution of the framework is the formalization of semantic contracts between layers, which define the semantics, assumptions, uncertainty bounds, and temporal characteristics of exchanged information in addition to its data structure.These contracts enable downstream layers to assess detection confidence, latency, and observability limitations, and to detect violations of underlying assumptions before incorrect behavior propagates [12]. Finally, the framework emphasizes **leading indicators** over traditional lagging diagnostics by promoting semantic health indicators such as operational margins, degradation trends, prognostic estimates, and contextual state to support proactive decision-making. By integrating prognostics and health management techniques, the framework shifts fault management from reactive response to proactive, intent-aware system health management [1, 18].

7. FAILURE SEMANTICS

7.1. Failure Semantics and Functional Safety

Functional safety standards such as IEC 61508 [10] and ISO 26262 provide rigorous processes for developing safety-critical systems. These standards emphasize systematic hazard analysis and risk assessment, safety integrity levels (SIL) and automotive safety integrity levels (ASIL), diagnostic coverage and safe failure fraction (SFF), and verification and validation of safety functions. Functional safety standards ensure that systems detect faults and execute protective actions to prevent hazards [11]. However, they focus primarily on compliance with process requirements and achievement of quantitative metrics such as SFF, diagnostic coverage, and probability of failure on demand.

A system can comply with functional safety standards yet still exhibit semantic failures, demonstrating that compliance does not guarantee correctness. Diagnostic coverage metrics measure the probability of detecting faults but do not ensure that detected faults are interpreted correctly or that responses are appropriate. Signoret and Leroy [10] critique the reliance on simplified probabilistic calculations in IEC 61508, arguing for systemic models such as fault trees, Markov chains, and Petri nets that capture complex, interacting failure modes. However, even these sophisticated models focus on what happens with fault propagation and failure

probabilities rather than why responses may be semantically incorrect. This limitation highlights the need for an explicit semantic framework.

Failure semantics complements functional safety by providing a framework for reasoning about the meaning of detected faults, not merely their probability. It emphasizes alignment between detection, interpretation, decision, and action, while highlighting the importance of operational context in determining correct responses. Most critically, it encourages verification of semantic correctness rather than syntactic compliance alone, addressing gaps that functional safety standards leave unexamined.

7.2. Failure Semantics as a Precursor to Resilience

Resilience engineering defines resilience as the ability of a system to anticipate, absorb, adapt to, and recover from disruptions [18, 20]. Resilient systems exhibit four key capabilities: anticipation of potential failures and preparation of responses; continuous monitoring of system state to detect anomalies; execution of appropriate actions to mitigate disruptions; and adaptation based on experience to improve future performance.

Failure semantics serves as a precursor to resilience because resilience fails when semantics are incorrect. A system cannot respond appropriately to a disruption if it misinterprets the meaning of detected conditions. Anticipation requires semantic models to understand how faults propagate and what they mean in different contexts [12]. Monitoring requires semantic interpretation, as detecting anomalies alone is insufficient; the system must interpret their severity, urgency, and scope [13]. Response requires semantic alignment, ensuring that executed actions align with system intent and address the objectives at stake [21]. Finally, learning requires semantic feedback to understand why responses succeeded or failed, enabling meaningful adaptation [16].
Irshad and Hulse [20] propose integrating human error and functional failure reasoning (HEFFR) into resilience modeling, emphasizing joint machine-human failure dynamics. Their work highlights that resilience depends on semantic alignment between automated systems and human operators, a central theme of the failure semantics framework. Similarly, Mishra et al. [15] presents a framework integrating reliability, resilience, and human factors for trustworthy AI systems. They argue that trustworthiness requires not just technical robustness but semantic correctness: AI systems must interpret their environment correctly and align their actions with human values and intentions. This perspective directly supports the objectives of the failure semantics framework, demonstrating its relevance across diverse application domains.

7.3. Reliability Metrics that Ignore Semantics

System dependability in reliability engineering is quantified using well-established metrics, including Mean Time Between Failures (MTBF), failure rate (λ), availability, and maintainability, which collectively describe failure occurrence, operational continuity, and repair efficiency. These metrics are essential for lifecycle cost analysis, maintenance planning, and design optimization. However, they are fundamentally context-free: they do not distinguish between failures occurring in different operational contexts or mission phases.

This limitation can be illustrated by comparing two systems, each with an MTBF of 1000 hours. System A fails randomly throughout operation, with failures equally likely in all mission phases. System B fails primarily during mission-critical phases when consequences are severe. Although both systems have identical MTBF values, System B is far more problematic because its failures occur when they matter most. Traditional reliability metrics fail to capture this semantically significant difference, potentially leading to misguided design and operational decisions. Failure

semantics suggests augmenting traditional reliability metrics with context-aware variants that align reliability analysis with system intent and operational reality.

8. FUTURE DIRECTIONS

The failure semantics framework opens several promising research directions that leverage emerging technologies and methodologies. The failure semantics framework enables several promising research directions. **Semantic digital twins** extend conventional digital twinsfocused primarily on physical fidelity [19]by integrating semantic models of operational context, system intent, severity assessment, and response strategies, enabling real-time interpretation of system states and proactive fault management [16, 19]. **AI-assisted interpretation layers** combine data-driven anomaly detection [3, 7] with causal reasoning [1], contextual enrichment [13], and formal verification to improve semantic trustworthiness and detect silent failures, as demonstrated by hybrid approaches such as the FAME framework [3]. **Intent-aware diagnostics** further advance fault management by interpreting detected conditions related to safety goals, mission objectives, and operational constraints, enabling intent-aligned responses optimized across safety, availability, and cost rather than fixed reaction logic [21]. Finally, **Cross-domain semantic standards** could address fragmentation across industries by defining shared ontologies and semantic contracts, improving tool interoperability, knowledge transfer, and certification efficiency across safety-critical domains [10, 11].

9. CASE STUDY: AUTONOMOUS VEHICLE FLEET FAILURE DURING URBAN POWER OUTAGE

9.1. Incident Description

On December 21, 2025, a fire at a San Francisco electrical substation caused widespread power loss affecting 130,000 customers and disabling hundreds of traffic signals. Waymo's autonomous vehicle fleet correctly detected non-functional signals and invoked four-way stop protocols. However, multiple vehicles stalled at intersections, blocking traffic and emergency access routes. Each vehicle requested human operator confirmation before proceeding with a response designed for isolated signal failures. The simultaneous requests overwhelmed operator capacity, creating a fleet-wide bottleneck. Waymo suspended service after six hours, resuming only after deploying emergency software updates[24].

Alignment with Failure Semantics: This incident exemplifies the paper's central thesis: accurate fault detection does not guarantee correct system response. The vehicles achieved syntactic correctness (detected faults accurately) but failed semantic correctness (responded inappropriately). The failure emerged from misalignment between detection, interpretation, decision, and action layers precisely the diagnostic-response decoupling described in Section 3.1.

9.2. System Architecture Analysis

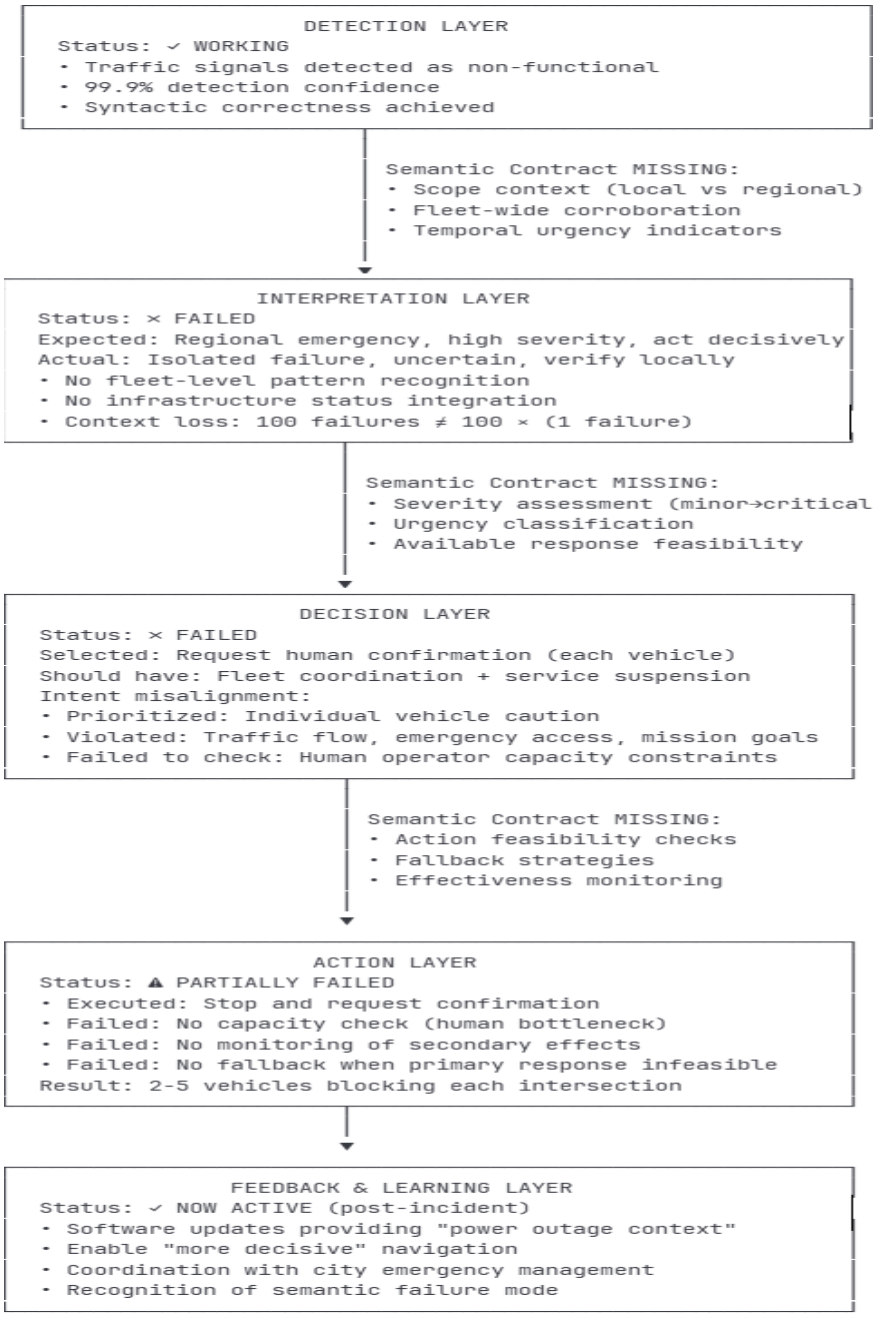

Figure 3: Semantic failure propagation through Waymo's autonomous vehicle architecture during the San Francisco power outage.

Note - (✓) indicates correct operation; (✗) indicates semantic failure; (⚠) indicates partial failure. The missing semantic contracts between layers allowed semantically incorrect interpretations and decisions to propagate unchecked.

9.3. Failure Semantics Characteristics

Table 1 compares the semantic characteristics required for correct system response against what the Waymo system exhibited during the incident:

Table 1: Comparison of required semantic characteristics versus actual system behavior during the Waymo San Francisco incident.

Semantic Dimension	Required for Correctness	Waymo System Behavior	Failure Mode
Severity Assessment	Critical: regional infrastructure failure requires immediate decisive action	Moderate: treated as isolated equipment issues requiring cautious verification	**Severity Semantics Failure**: System assessed each signal independently without recognizing cumulative severity
Scope Determination	Regional: 33% of city infrastructure affected, coordinated response needed	Local: each vehicle treated its intersection as isolated event	**Scope Semantics Failure**: No fleet-level pattern recognition or scope aggregation
Urgency Classification	Immediate: act decisively to clear intersections, enable emergency response	Delayed: extended verification periods acceptable ("stationary longer than usual")	**Temporal Semantics Failure**: Response latency appropriate for isolated failures, not mass events
Response Feasibility	Scale aware: human operator capacity constraints must be evaluated	Scale-blind: assumed confirmation request system could handle arbitrary load	**Action Feasibility Failure**: No capacity constraints checked before selecting response
Intent Alignment	Multi-objective: balance vehicle safety, traffic flow, emergency access, mission success	Single objective: optimized exclusively for individual vehicle caution	**Intent Semantics Failure**: Local optimization violated system-level safety and mission goals
Context Integration	Infrastructure-aware: integrate power grid status, emergency declarations, fleet patterns	Context-isolated: each vehicle evaluated traffic signals without external context	**Context Loss Failure**: No semantic information exchanged across system boundaries
Operational Success	7,000+ intersections navigated successfully	~dozens of intersections blocked	**Recovery Semantics Failure**: Successful responses masked underlying semantic brittleness

9.4. Analysis and Interpretation

Table 1 reveals a systematic pattern: the Waymo system operated under design-time semantic assumptions that became invalid at runtime. The detection layer functioned correctly, identifying non-functional traffic signals with high accuracy. However, each subsequent layer inherited and amplified semantic misinterpretations due to missing semantic contracts that would have conveyed context, constraints, and intent.

Three critical insights emerge from this comparison:

First, scope transforms severity. A single non-functional traffic signal is a minor anomaly; 100 simultaneous failures constitute a regional emergency. The system lacked mechanisms to reassess severity based on the scope of detected anomalies, treating 100 independent observations as 100

instances of the same low-severity event rather than recognizing the emergent high-severity pattern.

Second, response feasibility is context dependent. The human confirmation protocol was feasible under design assumptions (occasional isolated failures, low request rate) but became infeasible during the mass event. The Decision layer selected this response without checking whether the action layer could execute it at scale a violation of semantic contracts between these layers.

Third, local optimization can violate global intent. Each vehicle is correctly optimized for local safety (don't proceed without certainty), but the aggregate fleet behavior violated system-level objectives: traffic flow maintenance, emergency vehicle access, and mission completion. This demonstrates the intent semantics failure described in Section 4: responses that optimize a single objective while compromising higher-level goals.

9.5. Implications for Failure Semantics Framework

This case study validates the necessity of the five-layer framework with explicit semantic contracts proposed in Section 6:

Validation 1: Detection ≠ Response Correctness. Waymo achieved perfect detection accuracy yet experienced system failure, confirming that syntactic correctness is insufficient.

Validation 2: Semantic contracts are essential. The missing contracts between layers allowed semantically incorrect interpretations to propagate unchecked. Had the detection layer communicated scope context and fleet-wide patterns, the interpretation layer could have recognized the regional emergency.

Validation 3: Intent must be formalized and multi-objective. Without explicit representation of competing objectives (vehicle safety, traffic flow, emergency response), the decision layer could not balance trade-offs appropriately.

Validation 4: Context loss causes cascading failures. Information boundaries (vehicle-to-fleet, autonomous system-to-city infrastructure) caused critical context to be unavailable where needed.

Validation 5: Feedback loops enable learning. Waymo's post-incident software updates demonstrate the feedback and learning layer in action, incorporating "power outage context" to enable "more decisive navigation" precisely the semantic enrichment the framework advocates.
The incident also highlights a limitation not fully addressed in the original framework: the scalability of semantic reasoning. Even with perfect semantic contracts, the human-in-the-loop confirmation system created a bottleneck during the mass event. This suggests that semantic correctness must be achievable autonomously at scale, with human oversight for monitoring and exception handling rather than routine decision-making.

9.6. Lessons for Systems Engineering Practice

This case study yields seven actionable lessons directly aligned with the failure semantics framework:

Lesson 1: Implement explicit semantic contracts between system layers (Section 6). The detection layer reported "traffic signal non-functional" without conveying scope, corroboration, or urgency. Semantic contracts must communicate meaning, assumptions, and constraintsnot just

diagnostic codes. The detection-to-interpretation contract should have included: fault type, confidence, fleet-wide occurrence patterns, and geographic scope.

Lesson 2: Validate semantic assumptions across the operational envelope. Design-time assumptions (isolated failures, sufficient human capacity) became invalid on a scale. Apply the feedback and learning layer proactively: validate semantic models under rare but high-consequence scenarios using fault injection testing that evaluates interpretation appropriateness, not just detection accuracy.

Lesson 3: Enable context-aware interpretation through distributed reasoning (Section 6, Interpretation Layer). Individual vehicles lacked fleet-level awareness and infrastructure context, causing scope semantics failure (Section 4). Implement distributed semantic reasoning that aggregates fleet observations, integrates external context (power grid status, emergency declarations), and distinguishes local anomalies from system-wide events through V2V and V2I semantic information exchange.

Lesson 4: Formalize system intent and multi-objective decision-making (Section 6, Decision Layer). The system exhibited intent semantics failure (Section 4) by optimizing exclusively for individual caution while violating traffic flow and emergency access objectives. Define explicit intent hierarchies: normal operation prioritizes vehicle safety over traffic flow; emergencies prioritize emergency response access. Enable context-shift recognition to adapt priorities dynamically.

Lesson 5: Incorporate feasibility constraints and fallback strategies (Section 6, Action Layer). The system selected human confirmation without verifying feasibility at scale. The action layer must evaluate response feasibility (operator capacity, timing requirements), monitor for unintended consequences, and invoke fallbacks when primary responses fail: "IF confirmation queue > threshold THEN suspend service."

Lesson 6: Build semantic digital twins for proactive validation (Section 8.1). Traditional simulation validates detection and control but not semantic correctness. Semantic digital twins integrating physical and semantic models enable what-if analysis: simulating "100 simultaneous signal failures" would have revealed the interpretation bottleneck and capacity constraints before deployment.

Lesson 7: Establish feedback mechanisms for semantic drift detection (Section 6, Feedback Layer). Semantic models valid at design time become invalid as contexts evolve. Implement continuous monitoring of semantic correctness through metrics including context ambiguity frequency, contract violation rates, response effectiveness, and prediction-versus-actual divergence to detect emerging misalignments before incidents occur.

10. CONCLUSION

This paper has introduced Failure Semantics as a critical yet underrepresented dimension of systems engineering, explaining why modern complex systems can fail despite accurate fault detection. It demonstrated that failures arise not from diagnostic inaccuracies, but from semantic mismatches between detection, interpretation, decision-making, and action, amplified by temporal dynamics, context loss, human factors, and lifecycle evolution. By formalizing failure semantics, presenting a taxonomy of semantic failure modes, and proposing a layered semantic framework with explicit semantic contracts, this work shows that diagnostics-centric approaches are necessary but insufficient for system correctness. The Waymo San Francisco case study provided empirical validation of these concepts, demonstrating how syntactically correct fault

detection combined with semantic interpretation failures produced system-level breakdowns with safety and operational consequences. However, significant limitations constrain the framework's immediate applicability and highlight directions for future research. The framework lacks formal mathematical foundations, preventing rigorous verification of semantic correctness and limiting integration with existing formal methods tools. Computational complexity remains unanalyzed, raising questions about real-time feasibility in safety-critical embedded systems with strict timing constraints. The framework introduces a meta-problemsemantic reasoning itself may failyet provides no mechanisms for detecting when interpretation layers produce silent semantic failures that are more subtle than traditional diagnostic errors. Semantic correctness depends critically on complete and accurate context, but the framework does not address reasoning under context uncertainty, partial observability, or information acquisition constraints inevitable in real deployments. Constructing valid semantic models requires eliciting tacit knowledge, resolving stakeholder disagreements, and validating requirements correctnesssociotechnical challenges the framework acknowledges but does not directly solve.

Most critically, the proposal lacks empirical validation through prototype implementations, controlled experiments, or quantitative assessments demonstrating that semantic contracts reduce failures or justify their implementation complexity in production systems. Finally, practical guidance on incremental adoption, integration with existing toolchains, and management of increased design complexity remains underdeveloped. Despite these limitations, the failure of semantics framework establishes essential conceptual foundations for addressing a fundamental gap in systems engineering practice. Ensuring reliable and resilient system behavior requires elevating semantic alignment across system layers, tools, and human interactions to a first-class systems-engineering concern. Future research must address the identified limitations through formal specification of semantic correctness, computational complexity analysis, methodologies for semantic model validation, empirical studies quantifying semantic failure prevalence and intervention effectiveness, and industrial case studies demonstrating practical deployment. By bridging the gap between correct diagnostics and correct system response, failure semantics offers a path toward systems that not only detect what is wrong but understand what it means and respond appropriately toa capability increasingly critical as autonomous systems assume greater responsibility in safety-critical and mission-critical domains.

REFERENCES

[1] Gao, Z., Cecati, C., & Ding, S. X. (2015). A survey of fault diagnosis and fault-tolerant techniques Part I: Fault diagnosis with model-based and signal-based approaches. *IEEE Transactions on Industrial Electronics*, 62(6), 3757-3767. https://doi.org/10.1109/TIE.2015.2417501

[2] Häring, I. (2021). Failure mode and effects analysis. In *Advanced Sensing, Fault Diagnostics, and Structural Health Management* (pp. 89-112). MDPI. https://doi.org/10.3390/books978-3-0365-6182-0

[3] Yang, G.-Y., & Wang, F. (2025). Taming silent failures: A framework for verifiable AI reliability. *arXiv preprint arXiv:2510.22224*. https://arxiv.org/abs/2510.22224v1

[4] Seo, K.-M., & Park, K.-P. (2018). Interface data modeling to detect and diagnose intersystem faults for designing and integrating system of systems. *Complexity*, 2018, Article 7081501. https://doi.org/10.1155/2018/7081501

[5] Mansoor, A., van Geritn, E., et al. (2023). Backward failure propagation method for functional safety analysis. In *Advanced Fault Diagnosis and Health Monitoring Techniques for Complex Engineering Systems* (pp. 145-168). MDPI. https://doi.org/10.3390/books978-3-0365-6463-0

[6] Rostamabadi, A., Jahangiri, M., Zarei, E., Kalatpour, O., &Alanjari, P. (2020). A novel fuzzy Bayesian network approach for safety analysis of process systems: An application of HFACS and SHIPP methodology. *Journal of Cleaner Production*, 244, 118761. https://doi.org/10.1016/J.JCLEPRO.2019.336315

[7] Dowdesitll, B., Sinha, R., &MacDonell, S. G. (2021). A scoping study on fault diagnosis and health monitoring techniques for industrial cyber-physical systems. In *Proceedings of the 2021 IEEE International Conference on Industrial Cyber-Physical Systems* (pp. 1-8). IEEE.

[8] Echtle, K., Hammer, D., &Poitll, D. (Eds.). (1994). *Dependable Computing EDCC-1: First European Dependable Computing Conference*. Springer-Verlag.

[9] Kvalo, M., Torrealba, A., & Duarte, M. A. H. (2025). Built to bend: Strengthening drilling operations with resilience engineering & MPD. In *Proceedings of the SPE/IADC Drilling Conference and Exhibition*. https://doi.org/10.2118/spe-228388-ms

[10] Signoret, J.-P., & Leroy, A. (2021). Functional safety related modelling and calculations. In *Reliability Assessment of Safety and Production Systems* (pp. 705-742). Springer. https://doi.org/10.1007/978-3-030-64708-7_36

[11] ISO 26262: 2018 Standard International Organization for Standardization (ISO) Road vehicles - Functional safety.

[12] Bozzano, M. (2017). Causality and temporal dependencies in the design of fault management systems. *Electronic Proceedings in Theoretical Computer Science*, 259, 23-35. https://doi.org/10.4204/EPTCS.259.4

[13] Guo, L. (2022). From function to failure: A formal method for reasoning about program-related failure modes. *arXiv preprint arXiv:2210.08667*. https://doi.org/10.48550/arxiv.2210.08667

[14] Rostamabadi, A., et al. (2020). A novel fuzzy Bayesian network approach for safety analysis of process systems: An application of HFACS and SHIPP methodology. *Journal of Cleaner Production*, 244, 118761. https://doi.org/10.1016/J.JCLEPRO.2019.336315

[15] Mishra, S., Rao, A. B., Krishnan, R., et al. (2024). Reliability, resilience and human factors engineering for trustworthy AI systems. *arXiv preprint arXiv:2411.08981*. https://doi.org/10.48550/arxiv.2411.08981

[16] van Geritn, E., et al. (2022). Practical method to integrate diagnostic models into MBSE for design-time and runtime diagnosis support. *Systems Engineering*, 25(4), 289-307.

[17] Avizienis, A., Laprie, J.-C., Randell, B., &Landithr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1), 11-33.

[18] B. Werner and B. Schumeg, "Leveraging Traditional Design for Reliability Techniques for Artificial Intelligence," *2022 Annual Reliability and Maintainability Symposium (RAMS)*, Tucson, AZ, USA, 2022, pp. 1-6, doi: 10.1109/RAMS51457.2022.9893957.

[19] Konstantinos Mokos, Panagiotis Katsaros, Preben Bohn, Model-based safety analysis of requirement specifications,Journal of Systems and Software,Volume 219,2025,112231,ISSN 0164-1212,https://doi.org/10.1016/j.jss.2024.112231.

[20] Irshad, L., & Hulse, D. (2022). Resilience modeling with HEFFR: Human error and functional failure reasoning for joint machine-human failure modeling. In *Proceedings of the 2022 Annual Reliability and Maintainability Symposium* (pp. 1-8). IEEE.

[21] Chu, C., et al. (2024). Dynamic fault tree generation from SysML with temporal characteristics for embedded systems analysis. *Reliability Engineering & System Safety*, 241, 109634.

[22] Guo, L. (2022). From function to failure: Failure Mode Reasoning (FMR) method for formal analysis of program-related failure modes. *arXiv preprint arXiv:2210.08667*. https://doi.org/10.48550/arxiv.2210.08667

[23] [23] Brtis, John &McEvilley, Michael & Pennock, Michael. (2021). Resilience Requirements Patterns. INCOSE International Symposium. 31. 570-584. 10.1002/j.2334-5837.2021.00855.x.

[24] Waymo's San Francisco outage raises doubts over robotaxi readiness during crises(2025) https://www.reuters.com/business/autos-transportation/waymos-san-francisco-outage-raises-doubts-over-robotaxi-readiness-during-crises-2025-12-27/

AUTHOR

Anand Wanjari is a Systems Engineer with over 15 years of experience in automotive and heavy-duty powertrain systems, specializing in system safety, diagnostics, and resilience engineering. He has played a key role in defining system-level requirements, failure in semantics and diagnostic strategies supporting diesel, natural gas, and hybrid powertrain platforms, with a strong focus on tool integration and verification efficiency. His work bridges industry practice and research, addressing challenges in ISO 26262 functional safety, ISO 21448 and distributed system development.He is an active member of the International Council on Systems Engineering (INCOSE). His research interests include predictive diagnostics, system resilience patterns, failure semantics, and safety assurance for autonomous and connected vehicle systems. He actively contributes to technical publications and thought leadership in systems engineering and automotive safety.