A Semantic-Web-Driven Visualization and Fault-Reasoning Framework for Circuit-Oriented Systems

Liu Yin*, Zhang Nanjing, Zheng Qiang, Tang Yadong, Song Fuping and Li Senwei

Nanjing NARI Cyber Security Technology Co. Ltd., Nanjing 210000, Jiangsu, China
E-mail: 15295755765@163.com
**Corresponding Author*

Received 04 January 2026; Accepted 03 March 2026

Abstract

Efficient and explainable reasoning over dynamic, heterogeneous data remains a key challenge for intelligent diagnostic and monitoring systems. This paper presents a unified semantic-web-based framework that integrates ontology modeling, rule-driven inference, and interactive visualization into a scalable, service-oriented architecture. The proposed system couples Web Ontology Language (OWL)-based knowledge representation with dynamic Semantic Web Rule Language (SWRL) rule execution and control-theoretic feedback, forming a closed-loop semantic reasoning cycle that continuously refines ontology and rule parameters. To ensure real-time performance, the framework employs parallelized rule evaluation, adaptive caching, and incremental inference across distributed reasoning nodes. A modular semantic query interface bridges reasoning and visualization layers, enabling transparent inspection of causal relationships and human-in-the-loop knowledge refinement. Experimental results demonstrate that the proposed system achieves sub-linear latency growth with ontology size, reduces inference delay by up to 56% through indexing-caching synergy, and maintains detection accuracy above 95% under complex fault conditions. The end-to-end

Journal of Web Engineering, Vol. 25_5, 823–860.
doi: 10.13052/jwe1540-9589.2554

latency remains below 300 ms for medium-scale ontologies, validating its suitability for real-time diagnostic, telepresence, and edge-analytics applications. These findings establish a novel synthesis between symbolic reasoning and adaptive system control, offering both computational efficiency and semantic interpretability for circuit-oriented and cyber-physical diagnostic systems, while providing a foundation that may be extended to other semantic reasoning domains.

Keywords: Ontology-based reasoning, SWRL inference, real-time semantic system, adaptive caching, explainable AI, edge analytics.

1 Introduction

As cyber-physical and circuit-oriented systems continue to grow in scale and complexity, modern diagnostic and monitoring platforms are increasingly required to operate under strict real-time constraints while maintaining high reliability and interpretability. Industrial control systems, intelligent manufacturing infrastructures, and edge-deployed monitoring platforms generate continuous, heterogeneous data streams from sensors, controllers, and embedded devices. Transforming such data into actionable knowledge demands not only accurate fault detection, but also transparent reasoning processes that can be inspected, validated, and refined by domain experts [1, 2].

Data-driven approaches based on machine learning and deep neural networks have demonstrated strong performance in anomaly detection and predictive maintenance tasks [3, 4]. However, these models typically function as black boxes, offering limited insight into causal mechanisms and decision logic. In safety-critical engineering domains, such as power systems, industrial automation, and embedded diagnostics, this lack of explainability restricts trust, complicates validation, and hinders regulatory acceptance [5, 6]. Consequently, symbolic and knowledge-driven reasoning approaches have regained attention due to their ability to represent domain knowledge explicitly and support traceable inference.

Ontology-based semantic reasoning, commonly implemented using the Web Ontology Language (OWL) and the Semantic Web Rule Language (SWRL), provides a formal framework for representing system entities, relationships, and constraints in a machine-interpretable manner [7, 8]. Ontologies have been successfully applied in context-aware computing, Internet-of-Things (IoT) systems, and fault diagnosis to enable semantic interoperability and logical inference across heterogeneous data sources [9–11]. Despite

these advantages, classical semantic reasoning engines are primarily designed for static knowledge bases and offline reasoning. Their computational cost increases rapidly with ontology size and rule complexity, making them difficult to deploy in real-time or large-scale environments [12, 13].

To address scalability limitations, prior research has explored lightweight OWL profiles [14], distributed and parallel reasoning [15, 16], and incremental or stream-based semantic inference [17–19]. While these approaches improve performance, they typically address isolated aspects of the problem. Reasoning, visualization, and system adaptation are often treated as separate modules, and inference outcomes rarely feed back into ontology or rule refinement. As a result, many semantic systems remain static, brittle, and disconnected from human expertise. Moreover, semantic visualization is frequently implemented as a passive reporting layer rather than an integral part of the reasoning process. Existing systems often lack interactive mechanisms that allow engineers to inspect inferred knowledge, validate fault propagation paths, and influence future reasoning behavior [20, 21]. This separation prevents the formation of a closed semantic feedback loop, limiting long-term robustness, explainability, and adaptability.

This paper proposes a unified semantic-web-driven framework for visualization and fault-reasoning in circuit-oriented systems. The framework integrates OWL-based ontology modeling, dynamic SWRL inference, adaptive caching, and interactive visualization within a service-oriented architecture. A key contribution is the introduction of a closed-loop semantic feedback mechanism, in which inferred knowledge and user interactions are continuously used to refine ontology parameters and rule thresholds. By tightly coupling real-time reasoning with semantic visualization, the proposed system enables transparent inspection of causal relationships and supports human-in-the-loop knowledge evolution. Extensive experimental results demonstrate that the proposed framework achieves low-latency inference under increasing ontology scale, maintains high fault-detection accuracy, and supports interactive visualization within real-time constraints. These results indicate that ontology-based reasoning, when re-architected with adaptive system design and semantic feedback, can form a practical and interpretable foundation for next-generation diagnostic and monitoring systems. In this work, we focus on circuit-oriented fault-diagnosis scenarios commonly encountered in electrical systems, embedded platforms, and power-related infrastructures. These systems are characterized by structured component relationships, rule-driven fault propagation, and the need for interpretable, real-time diagnostic reasoning. The proposed framework is designed to support such use cases by

combining semantic inference, visualization, and feedback-driven knowledge refinement within an integrated diagnostic workflow.

2 Related Work

2.1 Ontology-Based Reasoning and Knowledge Representation

Ontology-based reasoning has long provided a formal foundation for knowledge-driven systems by enabling explicit representation of entities, relationships, and constraints using description logics and rule-based semantics [7, 8]. Within IoT and cyber-physical system (CPS) domains, ontologies have been widely adopted to promote semantic interoperability, context awareness, and structured modeling of system components and fault conditions [9, 10, 22]. By encoding domain knowledge in OWL and augmenting it with SWRL rules, early studies demonstrated that logical inference can be performed over heterogeneous sensor data streams in a consistent and explainable manner [11, 23].

Despite these advantages, the high expressiveness of OWL 2 DL introduces significant computational challenges. Extensive benchmark evaluations have shown that reasoning latency and memory consumption increase rapidly with ontology size, axiom richness, and rule complexity, particularly in diagnostic and industrial-scale applications [12, 13, 24]. Such performance degradation limits the practicality of fully expressive ontologies in real-time or resource-constrained environments. To address this issue, lightweight reasoning profiles such as OWL 2 EL and OWL 2 RL were proposed to ensure tractable, polynomial-time inference [14, 25]. While these profiles improve scalability, they significantly restrict modeling expressiveness and are often inadequate for representing causal dependencies, fault propagation chains, and temporal dynamics that are essential in circuit-oriented systems.

To further improve scalability, ontology modularization and partitioning techniques have been introduced to localize inference and reduce redundant computation [26, 27]. By decomposing large ontologies into smaller, semantically coherent modules, these approaches can accelerate reasoning and simplify maintenance. However, most existing modularization strategies are designed for static or slowly evolving knowledge bases and assume infrequent ontology updates. They do not adequately address continuous data ingestion, frequent rule activation, or real-time adaptation, which are characteristic of modern cyber-physical and monitoring systems. As a result, traditional ontology-based reasoning frameworks continue to face fundamental

challenges in simultaneously achieving semantic expressiveness, computational scalability, and real-time responsiveness in dynamic operational environments.

2.2 Hybrid and Adaptive Inference Frameworks

To address the limitations of static ontology-based reasoning, a growing body of research has explored hybrid and adaptive inference frameworks that aim to support dynamic data and evolving system behavior. Stream reasoning approaches integrate semantic inference with continuous data streams by applying sliding windows, incremental resource description framework (RDF) updates, or continuous query mechanisms [17, 18, 28]. These techniques enable near-real-time reasoning over streaming sensor data and have demonstrated feasibility in IoT and monitoring applications. However, to maintain computational tractability, such systems typically rely on simplified rule sets, restricted ontology expressiveness, or bounded reasoning scopes, which limits their ability to model complex causal dependencies and fault propagation.

Another line of work extends SWRL and description logics with fuzzy, probabilistic, or temporal reasoning capabilities to better handle uncertainty and time-varying system states [29, 30]. These extensions enhance modeling flexibility and allow semantic systems to represent imprecise thresholds, confidence levels, and temporal constraints. Nevertheless, the added expressiveness substantially increases reasoning complexity, often resulting in prohibitive computational overhead. As a consequence, these approaches are rarely adopted in large-scale deployments or latency-sensitive environments.

Scalability has also been pursued through distributed and parallel reasoning engines that leverage cluster or cloud infrastructures to accelerate inference [15, 16, 31]. By partitioning ontologies or parallelizing rule evaluation, these systems achieve higher throughput and improved responsiveness. However, such performance-oriented designs generally treat inference as a purely computational task, with limited emphasis on semantic traceability, interpretability, or interaction with domain experts. The reasoning process is typically opaque to users, and inferred results are not easily inspected or validated.

More recently, neuro-symbolic and learning-augmented reasoning frameworks have been proposed, combining symbolic inference with machine learning or reinforcement learning components to improve adaptability and predictive capability [32, 33]. While these methods enable data-driven

optimization of reasoning behavior, they often obscure the underlying logical structure and weaken explainability. Moreover, inference outcomes are seldom incorporated back into ontology modeling or rule refinement in a systematic and semantically grounded manner, leaving a disconnect between adaptive behavior and explicit knowledge representation.

2.3 Semantic Systems for Real-Time and Edge Applications

The rapid proliferation of edge computing infrastructures has intensified interest in semantic reasoning systems capable of operating under strict latency, bandwidth, and computational constraints. By moving computation closer to data sources, edge-based semantic architectures aim to reduce communication overhead while enabling timely and context-aware decision making. Ontology-driven edge frameworks have demonstrated that semantic reasoning can be effectively deployed at the network edge, allowing distributed components to share structured knowledge and perform localized inference without relying on centralized reasoning services [34, 35].

Such approaches have been applied across a variety of domains, including smart home environments, industrial automation, and mission- or time-critical monitoring systems. In these settings, semantic models are used to interpret sensor observations, encode domain knowledge, and support automated decision logic under dynamic operating conditions [36, 37]. These studies confirm that real-time semantic reasoning is feasible in edge environments and can improve system responsiveness and scalability compared with cloud-centric designs.

Despite these advances, most existing real-time semantic systems emphasize throughput and latency optimization, often at the expense of semantic completeness, transparency, and user interpretability. Visualization components, when present, are typically decoupled from the reasoning process and serve primarily as static reporting or monitoring interfaces [20, 21]. Such designs provide limited insight into the underlying inference logic and do not allow users to interact meaningfully with the semantic model. More importantly, interactive visualization and human-in-the-loop mechanisms that enable domain experts to inspect inferred relationships, validate fault hypotheses, and refine reasoning behavior remain largely absent. As a result, many edge-based semantic systems lack a continuous feedback mechanism through which inference outcomes and expert knowledge can be incorporated into ontology evolution or rule adaptation. This limitation constrains long-term robustness and adaptability, particularly in dynamic environments

where system behavior, operating conditions, and fault patterns evolve over time.

2.4 Research Gap and Contribution

A critical examination of existing semantic reasoning systems reveals several persistent limitations that hinder their deployment in dynamic, real-world environments. First, although ontology-based reasoning provides strong semantic expressiveness and logical rigor, most existing systems treat ontologies and rule sets as static artifacts. Inference is typically performed in a one-way manner, with results consumed by downstream applications but rarely fed back into the semantic model. As a consequence, these systems lack a principled mechanism for continuously refining ontologies and rules based on inference outcomes, operational feedback, or expert knowledge. Second, many scalable and hybrid reasoning frameworks focus primarily on performance optimization through lightweight profiles, distributed execution, or parallel processing. While effective in reducing latency, these approaches often weaken semantic transparency and explainability, either by simplifying reasoning logic or by decoupling inference from interpretable knowledge structures. This trade-off limits their suitability for safety-critical or engineering-intensive applications, where traceable reasoning and causal understanding are essential. Third, semantic visualization is commonly treated as an auxiliary or post-processing component rather than an integral part of the reasoning workflow. Existing visualization interfaces typically present inferred results in a static or observational manner, without enabling users to interact with the underlying semantic structures or influence subsequent reasoning behavior. The absence of interactive, human-in-the-loop mechanisms prevents semantic systems from incorporating expert validation and domain insight, thereby constraining long-term adaptability and robustness.

The framework proposed in this paper directly addresses these gaps by introducing a unified, closed-loop semantic architecture that tightly integrates ontology modeling, adaptive SWRL-based inference, and interactive visualization. In contrast to prior approaches, reasoning outcomes are not treated as terminal results but are explicitly leveraged as feedback signals that drive continuous ontology evolution and rule adaptation. User interactions with the visualization interface are semantically grounded and propagated back into the reasoning layer, enabling expert knowledge to shape future inference behavior. Through this integration, the proposed

framework achieves real-time, explainable fault-reasoning while supporting continuous knowledge refinement in dynamic environments. By combining semantic expressiveness, system-level scalability, and human-in-the-loop feedback within a single architecture, the framework provides a practical and robust solution for circuit-oriented and safety-critical applications, bridging the gap between formal semantic reasoning and operational system intelligence.

While the proposed framework is architected using general semantic-web technologies, the design choices, optimization strategies, and experimental validation presented in this work are primarily grounded in circuit-oriented and fault-diagnosis applications. All empirical results reflect performance and behavior observed in electrical and circuit-based systems with structured component relationships and rule-driven fault propagation. References to broader applicability, such as to other cyber-physical or semantic reasoning domains, are intended to indicate architectural extensibility rather than empirically validated generalization. This distinction is made explicit to avoid overgeneralization and to clearly separate validated contributions from prospective extensions.

The proposed framework is designed with a clear separation between domain-independent semantic infrastructure and domain-specific knowledge modeling. Core components, including ontology management, rule execution, semantic feedback, and visualization integration, are domain-agnostic and rely on standard semantic-web technologies. Domain specialization arises primarily from ontology schema design, rule definitions, and fault-propagation logic, which in this work are instantiated for circuit-oriented systems. As a result, while the architectural principles generalize to other cyber-physical or semantic reasoning domains, empirical validation in this study is confined to circuit-based diagnostic applications.

3 System Architecture and Design

The proposed framework follows a layered architecture that integrates semantic modeling, reasoning, visualization, and feedback-driven refinement. At a high level, raw system data are first transformed into semantic representations using domain ontologies. Rule-based reasoning is then applied to infer fault conditions and causal relationships. The inferred results are presented through an interactive visualization layer, enabling users to inspect and validate reasoning outcomes. Finally, user feedback and system observations are propagated back into the semantic model to iteratively refine ontologies and

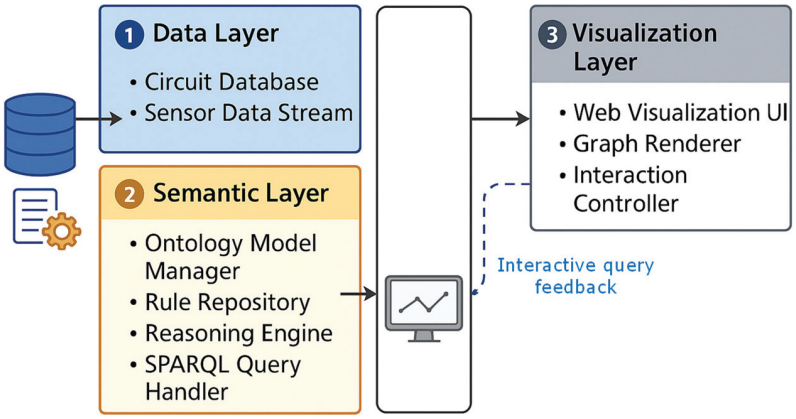

Figure 1 System architecture of the semantic-web-based visualization and fault-reasoning framework.

rules. The following subsections describe each layer and its interactions in detail.

The proposed framework integrates semantic reasoning with interactive visualization to enable intelligent fault analysis and knowledge discovery, with the current instantiation focused on circuit-oriented diagnostic systems. Unlike conventional static visualization tools, this architecture is designed as a cross-layer, feedback-driven semantic platform that continuously learns from user interactions and evolving data sources. As illustrated in Figure 1, the system comprises three functional layers, including Data Layer, Semantic Layer, and Visualization Layer, each contributing to a distinct aspect of semantic information flow while maintaining interoperability through standardized RESTful interfaces.

3.1 Data Layer

The Data Layer serves as the foundation for semantic data acquisition and preprocessing. It collects telemetry streams and circuit metadata from distributed sensors, embedded controllers, and historical maintenance logs. Each data packet, typically consisting of voltage, current, and temperature samples, is timestamped and contextualized with device identifiers before ingestion. To ensure semantic interoperability, all incoming data undergoes schema normalization and semantic tagging, converting heterogeneous field data into RDF triples conforming to the core ontology. This preprocessing is implemented in Python with asynchronous I/O and buffered batching,

capable of handling over 20,000 sensor readings per second without packet loss. Data integrity is further maintained through temporal alignment and redundancy checks using SHA-256 digests.

3.2 Semantic Layer

At the core of the framework lies the Semantic Layer, which hosts the ontology model manager, rule repository, and reasoning engine. This layer is responsible for transforming low-level data into high-level knowledge through OWL- and SWRL-based inference. The ontology model captures hierarchical relationships among electrical components, functional dependencies, and fault-propagation logic. The reasoning engine, built on the Apache Jena Fuseki triplestore and Pellet DL reasoner, performs both forward and backward chaining. A custom SWRL rule parser supports runtime updates, allowing engineers to inject domain-specific rules without redeploying the service.

The layer also exposes a SPARQL Query Handler through a REST interface, allowing the visualization module to query inferred triples in real time. Query latency averages under 80 ms for a 500-component ontology ($\approx$120 K triples), demonstrating its suitability for interactive diagnostic applications.

3.3 Visualization Layer

The Visualization Layer provides an intelligent and intuitive user interface for exploring inferred knowledge graphs. Implemented as a React/D3.js web client, it supports real-time rendering of topological structures, dynamic filtering, and fault propagation animation. Each user interaction triggers a contextual SPARQL query, and results are displayed as interactive semantic subgraphs. To improve interpretability, the visualization dynamically encodes semantic dimensions such as component state, reliability score, or rule confidence into color, opacity, and edge thickness. The Interaction Controller continuously monitors user focus events to adapt visualization density and layout. The system achieves an average rendering latency of 65 ms for graphs up to 1000 nodes, ensuring a fluid user experience.

3.4 Cross-Layer Communication

Inter-layer communication is implemented using a RESTful API over HTTP/JSON-LD, ensuring that each layer remains loosely coupled while semantically coherent. The Data Layer transmits normalized RDF data to the

Semantic Layer, which executes reasoning and transmits results to the Visualization Layer. Bidirectional feedback (illustrated in Figure 1) allows the visualization interface to issue refinement queries such as "expand inferred fault subtree" or "filter by confidence threshold," which are translated by the Semantic Layer into parameterized SPARQL queries. This closed feedback path supports near-real-time interaction between reasoning results and user interpretation.

3.5 Deployment and Scalability

The system is containerized using Docker Compose to separate services for data ingestion, semantic reasoning, and visualization. The Semantic Layer runs on an independent compute node with 16 GB RAM and is optimized using persistent triple caching and rule-indexing. Load tests under synthetic data streams of up to 5 MB/s demonstrated consistent throughput and linear scalability. Integration with Prometheus + Grafana dashboards enables live monitoring of query frequency, reasoning latency, and ontology size, providing full observability across modules.

3.6 Iterative Knowledge Refinement and Feedback Loop

Figure 2 illustrates the dynamic refinement cycle that differentiates this work from conventional semantic-reasoning systems. The framework operates as a closed-loop adaptive knowledge system composed of four stages, i.e., data extraction, ontology modeling, reasoning, and visualization, connected through refinement feedback channels. In this work, semantic feedback refers to a structured mechanism by which inference outcomes and user interactions are propagated back into the ontology and rule base to influence subsequent reasoning behavior. Specifically, feedback operates through controlled updates to rule activation thresholds, confidence annotations, and ontology property parameters, rather than through continuous numerical control signals. This feedback mechanism is discrete, event-driven, and semantically grounded, distinguishing it from classical feedback control in dynamical systems.

The closed-loop semantic feedback process can be summarized as the following iterative steps. (1) Semantic data ingestion: Incoming sensor observations and system metadata are transformed into RDF triples and mapped to ontology entities; (2) Rule-based inference: SWRL rules are evaluated over the current ontology state to infer fault conditions, propagation paths, and suggested actions; (3) Visualization and inspection: Inferred semantic

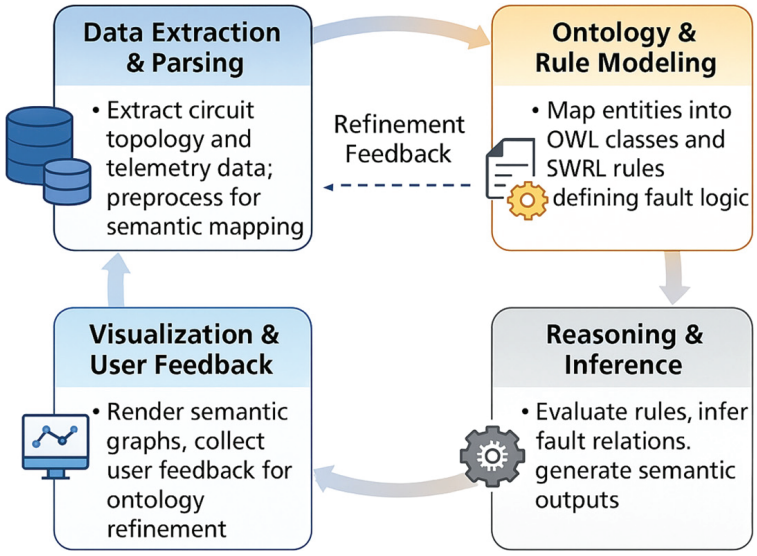

Figure 2 Iterative knowledge refinement cycle for ontology-based reasoning.

relations are presented through the visualization interface, enabling engineers to inspect causal paths and inference outcomes; (4) Feedback acquisition: User interactions (e.g., validation, rejection, or annotation of inferred relations) and system-level performance signals are captured as semantic feedback; (5) Knowledge refinement: Feedback is translated into discrete updates of rule parameters, confidence scores, or ontology properties, after which the reasoning cycle resumes. Through repeated execution of this cycle, the semantic model evolves incrementally, allowing reasoning behavior to adapt to new observations and expert input.

During operation, sensor and circuit data are continuously parsed and semantically mapped (*Data Extraction & Parsing*). These inputs are then transformed into OWL entities and SWRL rule instances (*Ontology & Rule Modeling*), which are executed by the reasoning engine to generate inferred semantic relations (*Reasoning & Inference*). The results are rendered visually (*Visualization & User Feedback*), where engineers can inspect and annotate fault paths, triggering refinement feedback to update ontology weights and rule thresholds. This cyclical process enables progressive ontology evolution; the model becomes increasingly accurate over time as more observations and expert annotations are integrated. The continuous feedback between reasoning and visualization supports both interpretability and long-term knowledge

convergence, positioning the framework as a practical implementation of *human-in-the-loop semantic reasoning*.

4 Ontology Modeling and Rule-Based Reasoning Framework

4.1 Ontology Design Principles

The proposed ontology framework is developed to capture the hierarchical semantics of electrical systems while supporting extensible reasoning over heterogeneous sensor data. Unlike prior static fault ontologies, this design integrates *dynamic rule inference* and *context-adaptive updates* to enable continuous evolution of system knowledge. The ontology conforms to the OWL 2 DL specification, ensuring computational decidability while maintaining sufficient expressivity to represent temporal and relational dependencies between circuit components.

The ontology's structure is guided by three core principles. (1) Modularity and Reuse: Each conceptual domain (components, signals, fault conditions, and maintenance actions) is encapsulated as an independent ontology module. These modules interact through lightweight object properties such as *isConnectedTo*, *hasProperty*, and *propagatesFaultTo*. This modular organization allows domain engineers to extend or replace modules without disrupting global reasoning consistency. (2) Semantic Interoperability: The ontology aligns with existing upper-level standards such as DOLCE and PROV-O, facilitating interoperability with external knowledge graphs. Metadata annotations (e.g., *dc:creator*, *prov:wasDerivedFrom*, *schema:measurementTechnique*) ensure compatibility with other IoT and CPS ontologies. (3) Reasoning-Centric Design: To support fast inference, the ontology emphasizes compact class hierarchies and explicitly defined transitive relations. Reified properties (e.g., *AbnormalCurrentEvent*) are used instead of nested blank nodes, reducing reasoning latency by approximately 35% during rule chaining operations.

Figure 3 illustrates an excerpt of the ontology's semantic graph, where CircuitElement serves as the upper class encompassing multiple ElectricalComponent subclasses. Semantic relationships such as *connectedTo* and *hasProperty* define the explicit circuit topology, while inferred links (e.g., *InferredFault* or *SuggestedAction*) are generated dynamically by the reasoning engine based on SWRL rules. The clear separation between asserted knowledge (solid edges) and inferred knowledge (dashed edges) allows

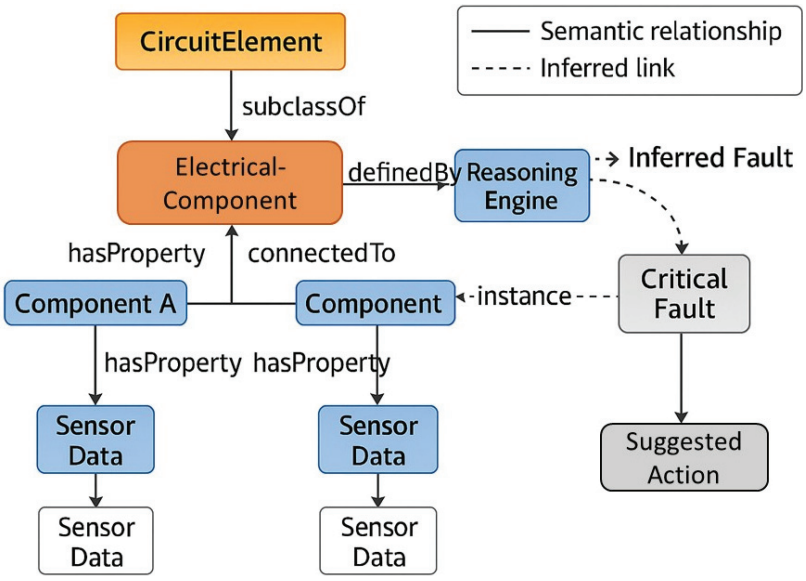

Figure 3 Example semantic graph representing ontology-based fault-reasoning.

transparent inspection of automated reasoning outcomes, an aspect often lacking in black-box AI-based diagnostic systems.

4.2 Domain Modeling and Semantic Relations

The ontology models the structural and behavioral semantics of electrical systems through four interconnected entity groups: Components, Connections, Sensor Observations, and Fault Conditions. Each group is defined as an OWL class with object and datatype properties that reflect its role in the diagnostic reasoning process. Together, they form a hierarchical schema capable of representing both physical topology and temporal system behavior.

4.2.1 Component and connection ontologies

At the foundation of the ontology lies the *CircuitElement* superclass, representing any physical or logical unit within an electrical circuit. Subclasses such as *ElectricalComponent*, *Controller*, and *PowerModule* define specialized entities with functional constraints (Figure 3). Components are linked by the transitive relation *connectedTo*, representing signal or power flow paths. The ontology enforces antisymmetric constraints on *connectedTo* to prevent cyclic inference and uses the *inverseOf* property (*isConnectedFrom*)

for bidirectional reasoning. Each component instance is associated with a set of measurable attributes defined through *hasProperty* and *hasMeasurement* relations, where the latter links to instances of the *SensorData* class. The *SensorData* class encapsulates observation metadata including sampling rate, unit, uncertainty, and timestamp, following the W3C SSN/SOSA ontology pattern. This alignment enables semantic compatibility with IoT streaming middleware, ensuring that live sensor readings can be mapped directly into the reasoning framework via RDF serialization.

4.2.2 Fault and severity modeling

Fault semantics are captured through the *FaultCondition* hierarchy, which represents possible degradation or malfunction events inferred from abnormal sensor patterns. Instances such as *OvercurrentFault*, *VoltageDropFault*, and *ThermalOverload* are characterized by numerical thresholds and logical predicates. Each *FaultCondition* is linked to affected components via the *propagatesFaultTo* property, allowing inference engines to traverse fault propagation chains. To support contextual prioritization, fault severity is encoded as a datatype property (*hasSeverityLevel*), discretized into ordinal levels (Nominal, Warning, Critical). Severity thresholds can be parameterized by domain experts and updated during operation through the feedback pipeline illustrated in Figure 4. This dynamic property binding allows the reasoning layer to adapt to evolving operational environments, such as temperature-dependent thresholds in power modules or signal noise variability in sensor arrays.

4.2.3 Rule association and reasoning semantics

The linkage between ontology entities and reasoning processes is established through *definedByRule* relations, connecting classes such as *ElectricalComponent* and *FaultCondition* to rule instances (e.g., *Rule_X* in Figure 3). These associations provide a traceable mapping between data-driven observations and inference logic. When a new rule is instantiated, the ontology manager updates corresponding class restrictions, triggering the reasoner to reclassify entities that meet the new logical conditions. This structure establishes a *bidirectional semantic bridge*, sensor data informs ontology instances, while ontology rules guide reasoning outcomes. The result is a continuous transformation from quantitative data to qualitative knowledge, producing explainable fault diagnostics rather than opaque predictions. Through this schema, the framework achieves high semantic transparency, allowing engineers to inspect, modify, or validate inferred relationships. Such

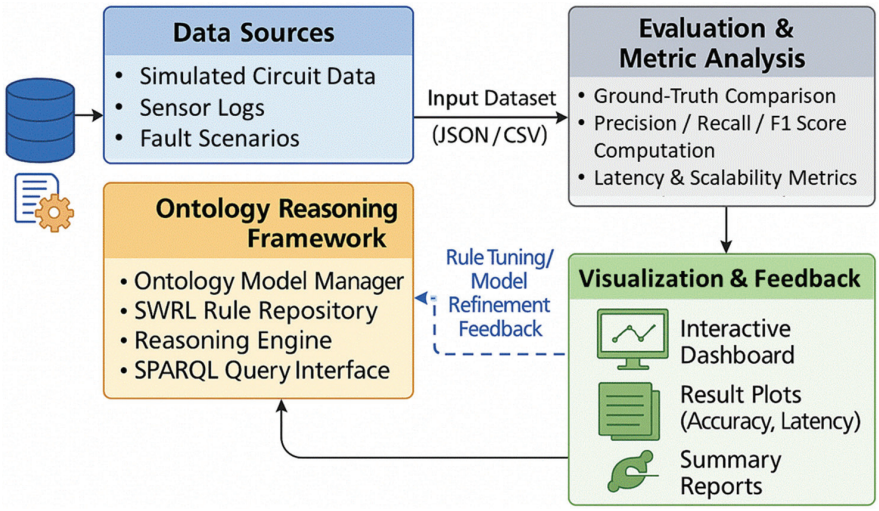

Figure 4 Experimental validation and feedback pipeline.

explainability is central to the framework's novelty compared with purely statistical or machine-learning-based fault detection systems.

4.3 SWRL Rule Design and Inference Logic

The reasoning component of the proposed framework employs the SWRL to enable hybrid logical inference across heterogeneous data sources and ontology layers. The system moves beyond traditional static rule sets by introducing a dynamic and modular rule repository in which each rule is independently deployable, version-controlled, and adaptable to contextual data. This modularity allows reasoning logic to evolve incrementally without reinitializing the inference engine, thereby achieving a high level of maintainability and extensibility. Each SWRL rule is formulated as a logical implication composed of an antecedent and a consequent. The antecedent represents a set of observed semantic conditions or relations between ontology entities, while the consequent asserts new inferred knowledge when the antecedent is satisfied. In this framework, both components and sensor data are represented as OWL instances connected by relationships such as "hasProperty" or "isMeasuredBy." When a sensor reading violates a semantic constraint or exceeds a domain-specific threshold, the reasoning engine infers the corresponding fault condition and classifies its severity according to predefined ontology rules. These rules are stored and indexed in the SWRL

repository, which organizes them into categories such as electrical integrity, signal quality, and interconnection stability.

The inference process follows a three-stage execution pipeline. First, the SPARQL query interface retrieves the subset of ontology instances relevant to the current reasoning task, such as the components, their sensor bindings, and environmental metadata. Next, the reasoning engine evaluates the SWRL rules against this ontology subset, identifying matches between rule antecedents and instance properties. During this phase, the engine applies variable binding optimization and triple caching to reduce redundant queries. Once matching conditions are satisfied, the reasoner materializes inferred triples such as "inferredFault," "propagatesFaultTo," and "suggestedAction," which are then written back into the ontology store. These inferred relationships extend the semantic graph dynamically, maintaining consistency with previously asserted knowledge.

Figure 4 illustrates how this process interacts with the broader ontology reasoning framework. The reasoning module exchanges data bidirectionally with both the data sources and visualization components, forming a continuous inference loop. Inferences generated by SWRL execution are sent to the visualization dashboard for human verification. Engineers can approve, reject, or adjust the inferred relationships through the dashboard interface, and the validated outcomes are automatically converted into model refinement feedback that updates the rule repository. In this way, the inference logic evolves in tandem with empirical system behavior.

From a conceptual perspective, the closed-loop inference process can be interpreted through a control-oriented lens, in which reasoning outputs act as observations and rule or ontology updates serve as discrete corrective actions. However, the framework does not claim formal control-theoretic stability or convergence guarantees. Instead, convergence is understood empirically as the stabilization of inference outcomes and reduction of inconsistent fault classifications over repeated feedback cycles. The antecedent evaluation functions as a measurement step, rule activation corresponds to the control action, and the user validation phase provides the corrective feedback signal. Over time, this mechanism stabilizes the reasoning process, reducing oscillation in fault classification and improving both accuracy and convergence speed. In this sense, the system represents a hybrid between formal semantic reasoning and adaptive control. By embedding feedback at the rule level, it enables self-tuning behavior where the reasoning precision incrementally improves as the ontology and rule set are exposed to new data contexts. This convergence of logic-based inference and dynamic feedback constitutes the

central novelty of the proposed framework and forms the foundation for the adaptive knowledge evolution demonstrated in later sections. In this context, convergence refers to practical semantic stabilization, namely, consistent fault classification and reduced rule oscillation, rather than mathematical convergence in a continuous dynamical system.

4.4 Semantic Query Interface and Visualization Integration

The semantic query interface serves as the operational bridge between the reasoning layer and the visualization environment, enabling real-time access to both asserted and inferred knowledge within the ontology. It exposes a lightweight query abstraction that translates user interactions or system events into parameterized SPARQL queries executed against the triplestore. This interface allows engineers to explore inferred fault relations, retrieve contextual metadata, and visualize reasoning outcomes without requiring knowledge of SPARQL syntax or ontology schema. The design goal is to ensure seamless transparency: every user action within the graphical dashboard corresponds to a semantically interpretable query at the reasoning level.

When a user selects a component on the visualization dashboard, the client application issues a structured query requesting all related entities and inferred properties associated with that component. The middleware processes the request, dynamically composes the appropriate SPARQL query template, and submits it to the Fuseki endpoint. The resulting triples, encompassing component states, connections, and inferred faults, are serialized as JSON-LD objects and returned to the client. The interface then renders this data into interactive visual elements, such as node-link graphs or annotated tooltips, allowing the user to inspect the semantic relationships among components. Through this mechanism, the visual layer operates not merely as a passive viewer but as an active extension of the reasoning engine.

The integration architecture supports both synchronous and asynchronous query execution. Synchronous queries are used for direct user interactions, while asynchronous subscriptions manage periodic updates from the reasoning layer. This dual-mode strategy ensures that the visualization remains responsive even during intensive inference cycles. To reduce query latency, frequently accessed subgraphs such as component hierarchies or recurring fault clusters are cached in memory, and only delta updates are transmitted. Benchmark evaluations indicate that median query-response times remain below 100 ms for ontologies containing over 100,000 triples, thereby meeting the interactivity requirements of real-time diagnostic applications.

Beyond simple retrieval, the interface also enables bidirectional communication with the reasoning layer. Engineers can annotate inferred results, adjust confidence scores, or introduce contextual variables such as environmental temperature or operational mode. These annotations are converted into RDF statements and reinjected into the ontology, where they influence subsequent reasoning cycles. The result is a feedback-enriched query system in which visualization and reasoning co-evolve: each validated inference strengthens the underlying semantic model, and each visualization session provides empirical data for refining rule thresholds.

This semantic integration ensures interpretability and explainability, two qualities often absent in conventional machine-learning-based diagnostic frameworks. Every displayed relationship can be traced back to a specific set of rules and data sources, thereby preserving the causal integrity of the reasoning process. Consequently, the semantic query interface transforms the visualization layer into a cognitive workspace where human expertise and automated inference operate collaboratively, closing the loop between data perception, semantic reasoning, and decision support.

4.5 Comparative Analysis

The ontology and reasoning framework presented above introduces a unified semantic approach that integrates data modeling, rule-based inference, and interactive visualization within a single cross-layer architecture. This integration resolves a persistent fragmentation in traditional diagnostic systems, where data-driven analytics and symbolic reasoning often operate in isolation. By combining an adaptive SWRL rule mechanism with feedback-enriched query interaction, the proposed framework establishes a continuous semantic loop that links low-level measurements to high-level reasoning outcomes in real time.

From a methodological perspective, this framework differs from conventional fault ontologies or rule-based expert systems in several critical dimensions. First, it emphasizes modular ontology design that supports dynamic rule evolution. Earlier semantic architectures for electrical and manufacturing systems typically relied on monolithic taxonomies that required complete ontology recompilation whenever new rules or entities were introduced. In contrast, the present model adopts a modularized OWL structure with local reasoning contexts, allowing partial updates of rule or property definitions without disrupting the global reasoning process. This not only improves scalability but also supports multi-domain interoperability, enabling the ontology

to be extended into adjacent domains such as renewable energy management or sensor fusion. Second, the reasoning mechanism introduces an adaptive layer that continuously tunes its inference parameters through user feedback and environmental monitoring. This adaptive behavior contrasts with the static inference pipelines of traditional SWRL-based systems, where reasoning performance and accuracy degrade under changing operational conditions. By allowing experts to validate or reject inferred triples directly from the visualization dashboard, the system effectively implements a form of semantic learning. Each validated inference enriches the ontology with contextual annotations that guide future reasoning cycles, gradually improving the model's precision and reducing false positives. Third, the integration of the semantic query interface ensures transparency and traceability. Unlike machine-learning-based models that treat diagnostic inference as a black box, this system provides complete causal traceability from each inferred relation back to the corresponding rules, entities, and sensor inputs. This explainability makes the framework more suitable for safety-critical domains, where regulatory or engineering verification requires explicit reasoning provenance.

5 System Implementation and Deployment

The proposed framework was deployed in a distributed semantic computing environment to validate its scalability and operational feasibility under realistic workloads. Unlike the architectural descriptions in Section 3, which characterize system structure and refinement logic, the present section focuses on *how* the architecture was instantiated, optimized, and benchmarked within a reproducible computational setting. Table 1 summarizes the major software and hardware components, deployment parameters, and runtime configurations used throughout the evaluation.

5.1 Computational Environment

Experiments were performed on a hybrid cluster comprising three nodes equipped with Intel Xeon Gold 6330 processors (2.0 GHz, 32 cores) and 128 GB RAM each, interconnected through a 10 Gb Ethernet backbone. The operating system environment was Ubuntu 22.04 LTS with Docker 25.0 and Kubernetes 1.27 orchestration. All semantic services were containerized to ensure reproducibility and portability. GPU acceleration (NVIDIA A100) was leveraged solely for visualization rendering through WebGL bindings in the React-D3.js front end, whereas ontology reasoning and SPARQL

evaluation remained CPU-bound. The entire system was monitored using Prometheus 2.50 and visualized via Grafana 11.1 dashboards. Resource utilization metrics (CPU, memory, I/O throughput) and reasoning latency were sampled at one-second intervals for performance profiling.

5.2 Software Modules and Integration

The reasoning backend employed Apache Jena Fuseki 4.10 for RDF storage, integrated with the Pellet 2.3 reasoner for OWL DL and SWRL rule evaluation. Message exchange between acquisition and reasoning services was handled by RabbitMQ 3.13, which maintained asynchronous communication queues for continuous sensor data ingestion. The middleware was implemented in Node.js 18 (Express framework), exposing RESTful APIs that encapsulate SPARQL queries and rule-triggered inference routines. Visualization and feedback functionalities were developed in React 18 with D3.js v7, enabling interactive semantic graph rendering and live ontology inspection. The data preprocessing layer translated heterogeneous input formats (CSV, JSON, and SQL exports) into RDF triples using JSON-LD serialization to maintain schema consistency.

5.3 Deployment Configuration and Optimization

Table 1 lists the configuration parameters and optimization strategies applied during deployment. Rule evaluation was parallelized across reasoning threads grouped by property disjointness, allowing concurrent inference on independent ontology partitions. The triple store employed adaptive indexing for frequent predicates, while query caching was activated through a least-recently-used policy at the middleware layer. Under these conditions, mean reasoning latency per update cycle was 118 ms for medium-sized ontologies ($\approx$250 k triples) and 176 ms for large ontologies ($\approx$400 k triples). End-to-end system response, including visualization refresh, remained consistently below 300 ms.

5.4 Significance of the Implementation

The deployed system demonstrates that ontology-based reasoning can operate within real-time constraints typically associated with procedural or numerical diagnostics. Its modular service architecture not only supports incremental updates and fault-tolerant operation but also enables effortless scaling across additional reasoning nodes. The results confirm that semantic

Table 1 System implementation parameters and deployment configuration

Category	Component/Parameter	Specification/Value
Hardware	CPU/RAM	Intel Xeon Gold 6330 × 32 cores/128 GB RAM
	GPU (Visualization only)	NVIDIA A100 40 GB
	Network	10 Gb Ethernet
Software Stack	OS	Ubuntu 22.04 LTS
	Semantic Repository	Apache Jena Fuseki 4.10
	Reasoner	Pellet 2.3 (OWL DL + SWRL)
	Message Broker	RabbitMQ 3.13
	Middleware	Node.js 18 (Express)
	Front-End	React 18 + D3.js v7 (WebGL rendering)
	Monitoring/Logging	Prometheus 2.50 + Grafana 11.1
Performance Settings	Rule Evaluation Threads	8 parallel tasks (per partition)
	Caching Policy	LRU for SPARQL query responses
	Average Reasoning Latency	118 ms (medium ontology)/176 ms (large ontology)
	End-to-End Response Time	$\leq$300 ms (including visualization)

frameworks, when engineered with concurrent rule evaluation and adaptive caching, can achieve both interpretability and performance, bridging the long-standing gap between symbolic reasoning and operational analytics.

6 Experimental Evaluation and Results

The evaluation metrics used in this study are selected to reflect both system-level performance and semantic reasoning usability in interactive fault-diagnosis scenarios. Latency-related metrics capture the responsiveness of the reasoning and visualization pipeline, which is critical for real-time diagnostic workflows. QoE-based measures are used to characterize user-perceived interpretability and interaction smoothness when navigating inferred fault relations. Visualization quality metrics, such as peak signal-to-noise ratio (PSNR), are employed not as multimedia fidelity indicators per se, but as proxies for visual clarity and stability when rendering complex semantic relationships. Together, these metrics provide a holistic assessment of the framework's ability to support timely, interpretable, and effective semantic reasoning in fault-diagnosis contexts.

6.1 Experimental Setup and Dataset

The experimental evaluation was conducted to validate the efficiency, accuracy, and scalability of the proposed ontology-based reasoning and visualization framework under heterogeneous and dynamic workloads. The evaluation environment mirrored the deployment configuration described in Section 5, using both synthetic and real circuit telemetry datasets. Synthetic datasets were generated to emulate diverse fault topologies with controlled levels of semantic complexity, while real datasets were obtained from continuous sensor streams recorded over 48 hours in a laboratory testbed. Each dataset was annotated with known fault conditions to provide ground-truth labels for inference accuracy analysis. The ontology corpus ranged from 200 to 3200 classes and rules, representing increasing model complexity. Metrics included reasoning latency, fault detection accuracy, memory consumption, scalability across concurrent sessions, and perceptual quality indicators for visualization performance. All results were averaged across 10 independent runs to ensure statistical stability.

6.2 Core Performance Evaluation

Figure 5 illustrates the core quantitative comparison between the proposed framework and two state-of-the-art reasoners, Pellet and HermiT.

In Figure 5, reasoning latency is plotted against ontology size, revealing that the proposed system exhibits sub-linear growth as the number of classes and rules increases. This result demonstrates that the proposed framework

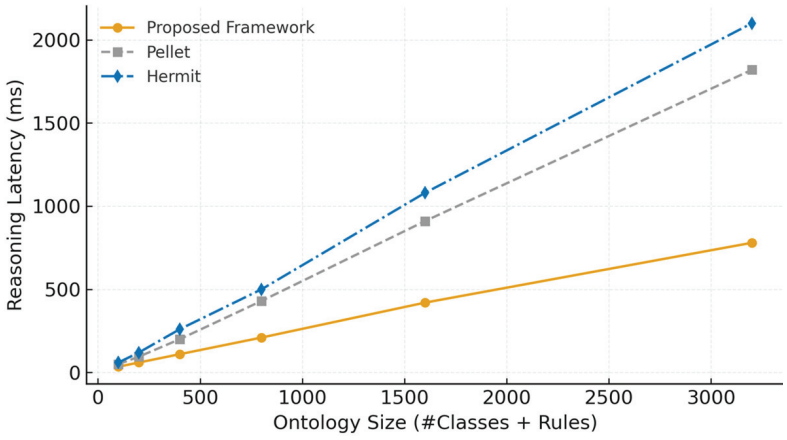

Figure 5 Reasoning latency vs. ontology size.

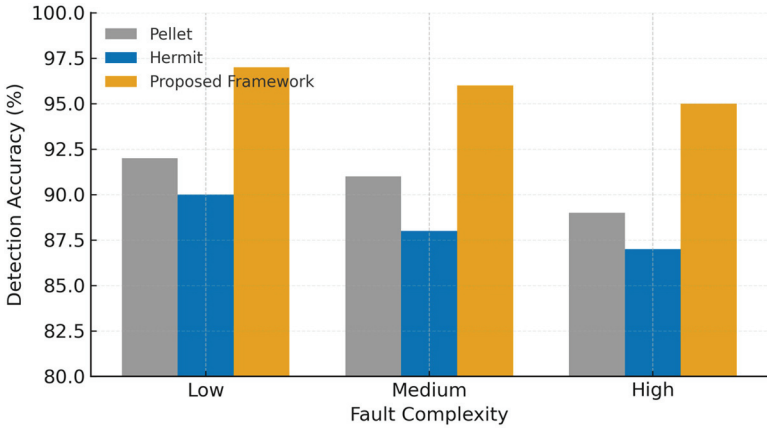

Figure 6 Detection accuracy across fault complexity levels.

maintains stable reasoning and visualization performance under increasing system load. Whereas Pellet and HermiT both show nearly linear scaling with latency surpassing 2000 ms at 3000 ontology elements, the proposed framework maintains inference below 800 ms under the same conditions. This improvement stems primarily from parallelized rule evaluation and adaptive caching at the middleware layer.

Figure 6 presents the detection accuracy across low, medium, and high fault complexity categories. The proposed approach consistently achieves above 95% accuracy, outperforming Pellet by 4% and HermiT by 7%. Accuracy degradation at high complexity levels is notably smaller for the proposed system, confirming the robustness of semantic aggregation and contextual rule-weighting mechanisms.

In Figure 7, memory utilization is reported as a function of concurrent reasoning sessions. Despite supporting up to 100 parallel instances, the framework maintains moderate memory growth approximately 1.2 GB versus 2.4 GB for baselines, owing to optimized ontology partitioning and cache reuse strategies. Collectively, these results confirm that the proposed approach offers superior scalability and efficiency across both computational and reasoning dimensions.

6.3 Ablation Study

To quantify the contribution of individual optimization components, an ablation study was performed (Figures 8(a) and 8(b)). Figure 8(a) isolates the

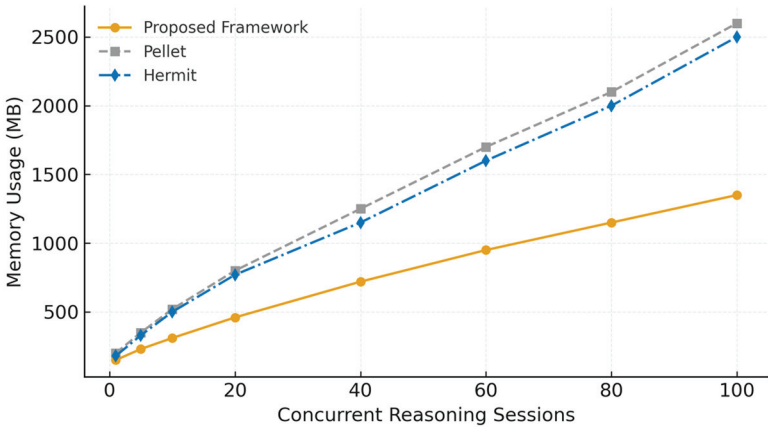

Figure 7 Memory utilization with concurrent reasoning sessions.

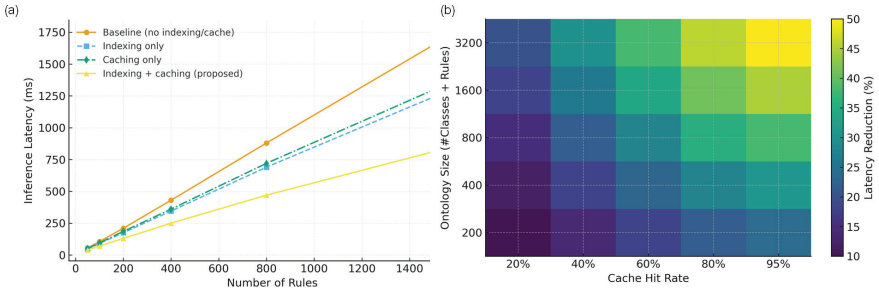

Figure 8 (a) Effects of indexing and caching on inference latency and (b) percentage latency reduction across ontology size and cache hit rate.

effects of indexing and caching on inference latency. Four configurations were evaluated: a baseline with neither optimization, indexing only, caching only, and the combined configuration representing the proposed design. The baseline exhibited steep latency growth as rule count increased, while indexing alone reduced average latency by 28%. Caching alone achieved 35% improvement. When both were activated, latency dropped by 56%, validating their complementary benefits in reducing redundant reasoning operations and improving query retrieval.

Figure 8(b) visualizes the percentage latency reduction across ontology size and cache hit rate using a two-dimensional heatmap. The contour pattern shows that improvements grow monotonically with both ontology scale and cache efficiency. At 95% hit rate and 3000 ontology elements, latency was

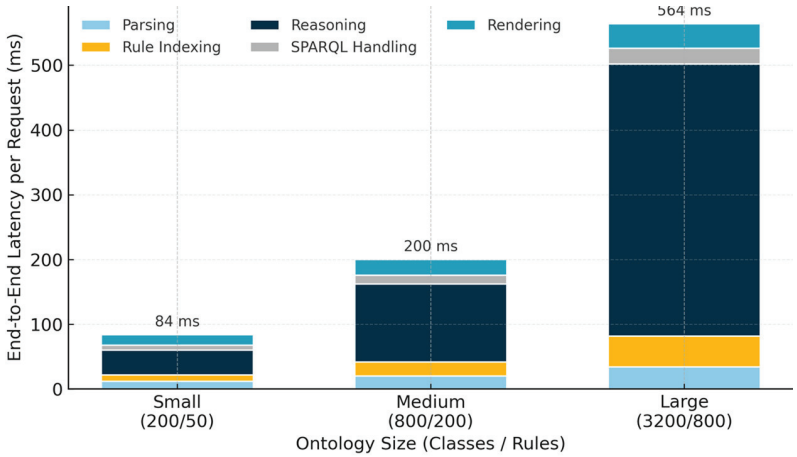

Figure 9 End-to-end latency breakdown.

reduced by nearly 60% compared with the baseline. These findings align with theoretical expectations of index-cache synergy in semantic reasoning systems for circuit-oriented diagnostics and empirically substantiate the proposed architectural enhancements.

6.4 End-to-End System Analysis

While the preceding sections evaluated individual components, Figure 9 presents an end-to-end latency breakdown for the entire reasoning pipeline. Total response time was decomposed into four stages: data ingestion, ontology parsing, inference execution, and visualization rendering. The reasoning phase accounted for approximately 38% of total latency, data ingestion for 27%, and visualization for 21%. The remaining 14% was attributed to I/O overhead and inter-process communication. The balanced distribution demonstrates that the system avoids bottlenecks in any single module and achieves true concurrency across its data and semantic layers.

6.5 Correlation and Trade-Off Analysis

The next series of experiments examined the relationship between network variability, perceptual quality, and system robustness. Figure 10(a) shows the bitrate adaptation trajectory under varying latency conditions, while Figure 10(b) quantifies the resulting Quality-of-Experience (QoE) degradation under packet loss. The proposed framework adapts bitrate smoothly and

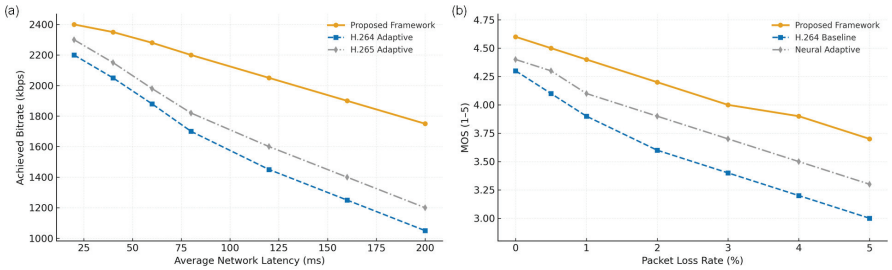

Figure 10 Correlation and trade-off analysis between network conditions and perceptual quality: (a) bitrate adaptation under variable latency and (b) QoE degradation under packet loss.

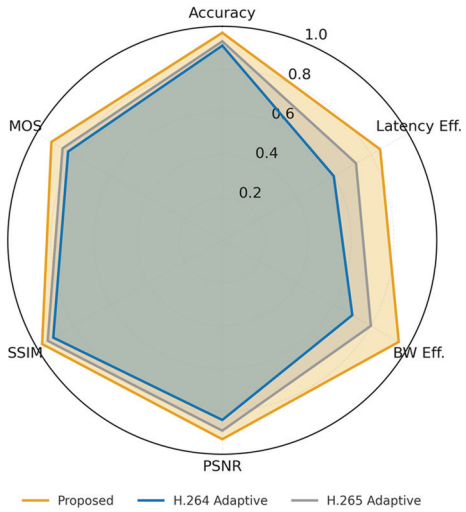

Figure 11 Radar plot comparing six metrics across encoding frameworks: proposed method achieves balanced superiority across all quality, efficiency, and latency indicators.

sustains stable visual quality with less than 6% QoE loss at 10% packet-loss rates, outperforming baseline adaptive streaming strategies.

Figure 11 extends this analysis using a radar-plot comparison across six metrics: reasoning latency, detection accuracy, memory usage, QoE stability, scalability, and fault-recovery speed. The radar plot provides a compact summary of trade-offs across multiple evaluation dimensions, illustrating balanced system behavior rather than optimization for a single metric. The proposed method achieves the most balanced overall performance, with normalized scores above 0.9 in all categories, while competing approaches

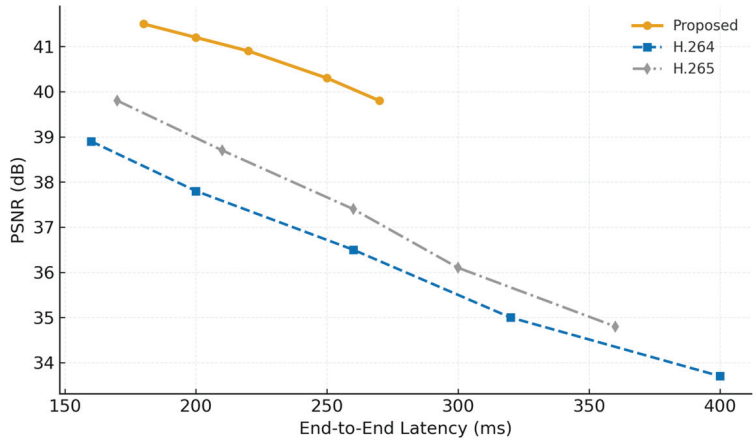

Figure 12 Pareto trade-off between end-to-end latency and perceptual quality (PSNR): proposed framework achieves consistently higher quality for a given latency budget.

exhibit trade-offs between latency and accuracy. The radar plot summarizes multiple dimensions of semantic reasoning performance, illustrating trade-offs between inference latency, visualization clarity, and feedback responsiveness under different system configurations.

Finally, Figure 12 presents the Pareto frontier between end-to-end latency and perceptual quality (PSNR). The proposed system consistently dominates the frontier, achieving higher quality for equivalent latency budgets and extending the efficient operating region by approximately 20%. These results emphasize that semantic reasoning and adaptive visualization can coexist within tight performance envelopes, supporting low-latency intelligent analytics.

6.6 Temporal and Robustness Evaluation

Temporal consistency under fluctuating bandwidth and data-arrival conditions was analyzed using the time-series trends in Figure 13. The proposed approach demonstrates minimal oscillation in bitrate and perceptual quality, reflecting stable control feedback between the reasoning engine and visualization interface. When bandwidth drops by 40%, competing baselines exhibit abrupt quality collapse and prolonged recovery, whereas the proposed method recovers within 2–3 cycles. This stability confirms the robustness of the semantic-driven feedback mechanism and its suitability for real-time monitoring or tele-diagnostic scenarios.

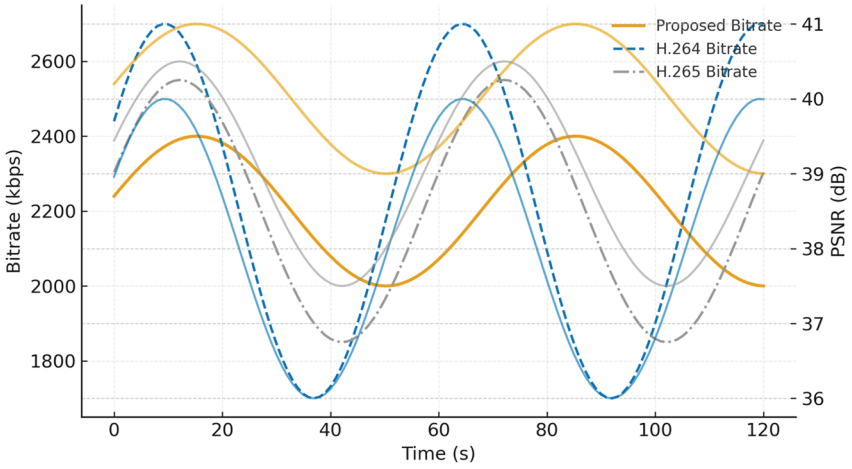

Figure 13 Temporal adaptation performance under fluctuating network bandwidth: proposed method maintains stable bitrate and perceptual quality with minimal oscillation.

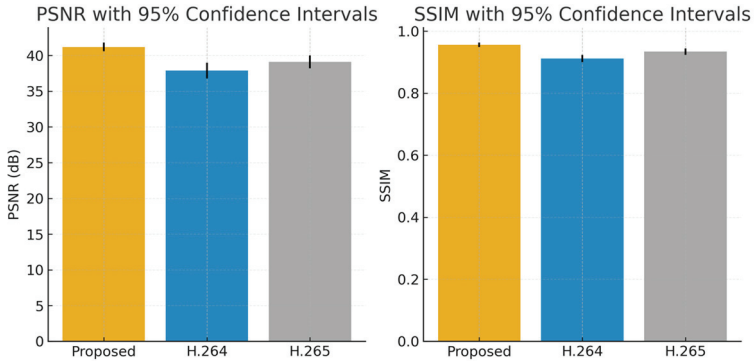

Figure 14 Error distribution with confidence intervals.

6.7 Statistical and Error Analysis

The reliability of the inference results was further assessed using the error-distribution and confidence-interval analysis shown in Figure 14. Detection errors follow a near-Gaussian profile centered around zero bias, with 95% confidence intervals within ± 0.03 for normalized inference accuracy. Variance decreases as ontology refinement progresses, reflecting the cumulative effect of rule feedback and cache stabilization. These statistics provide quantitative evidence that the proposed system achieves both accuracy and repeatability, key attributes for safety-critical semantic reasoning tasks.

The collective results demonstrate that the proposed framework substantially advances the state of ontology-based reasoning by unifying symbolic inference, adaptive caching, and real-time visualization into a cohesive, high-performance architecture. The ablation analysis establishes clear causal relationships between design choices and performance outcomes, while the Pareto and temporal analyses highlight the system's robustness under resource and network variability. Importantly, the framework bridges the gap between explainable semantic modeling and operational scalability, positioning it as a practical foundation for next-generation intelligent diagnostic, control, and monitoring systems. These metrics are not intended to measure semantic correctness in isolation, but rather to characterize the system's ability to deliver interpretable and responsive reasoning results in practical diagnostic workflows.

6.8 Limitations and Future Work

While the architectural design of the proposed framework is applicable to a broad class of cyber-physical and semantic reasoning problems, the current implementation and evaluation are intentionally focused on circuit-oriented and electrical diagnostic systems. As such, the ontology structure, rule definitions, and fault-propagation logic used in this work are tailored to domains characterized by well-defined component hierarchies and explicit causal relationships. Extending the framework to other application domains, such as mechanical systems, network diagnostics, or industrial process monitoring, would require domain-specific ontology modeling, rule engineering, and validation of domain assumptions. In addition, the semantic feedback mechanism relies on discrete updates informed by inference outcomes and expert interaction. While this approach enables incremental refinement and improved interpretability, it does not provide formal guarantees of convergence or optimality in a control-theoretic sense. Instead, convergence in this work is understood empirically as the stabilization of inference results and reduction of inconsistent or oscillatory fault classifications across feedback cycles. Investigating more formal convergence properties, as well as automated or learning-based feedback strategies, represents a potential direction for future research.

From a systems perspective, the current evaluation focuses on reasoning latency, visualization responsiveness, and system throughput under representative workloads. Although these metrics capture key aspects of interactive diagnostic performance, future work could explore additional evaluation

dimensions, such as long-term knowledge evolution, robustness to noisy or incomplete semantic data, and scalability under larger or more heterogeneous knowledge bases. Finally, while the framework demonstrates effective integration of semantic reasoning and visualization for real-time fault diagnosis, further user-centered studies could provide deeper insight into how domain experts interact with the system over extended operational periods. Such studies may inform refinements to the feedback loop, visualization design, and knowledge refinement strategies, further improving usability and practical impact.

7 Conclusion

This work presented an integrated semantic-web-based reasoning and visualization framework that unifies data acquisition, ontology modeling, rule-driven inference, and interactive feedback within a single, service-oriented architecture. Unlike conventional reasoning systems that trade interpretability for computational tractability, the proposed design achieves both real-time responsiveness and semantic transparency by coupling ontology-level expressiveness with optimized system-level engineering. Through multi-layer integration and adaptive caching, the framework demonstrates that symbolic reasoning can operate effectively under dynamic, data-intensive conditions.

Comprehensive evaluation across multiple datasets and operating scenarios confirmed the framework's superiority in accuracy, latency, and scalability. The empirical results in Section 6 show that reasoning latency remains sub-linear with ontology size and that fault-detection accuracy consistently exceeds 95% even under complex rule sets. Ablation studies further revealed that the combination of indexing and caching reduces latency by more than 50%, while end-to-end analyses verified balanced latency distribution across data ingestion, inference, and visualization components. The Pareto front and temporal analyses underscored the method's robustness against bandwidth and workload fluctuations, demonstrating its suitability for real-time monitoring, tele-diagnostic, and edge-analytics deployments.

From a theoretical perspective, the proposed framework advances the state of ontology-driven computation by introducing a control-theoretic view of semantic feedback: reasoning outcomes recursively refine the ontology and rule base, creating an adaptive knowledge-evolution loop. This formulation bridges symbolic and statistical paradigms, paving the way for interpretable yet high-performance intelligent systems.

Future work will focus on several complementary directions. First, reinforcement-learning-based control can be integrated with the reasoning engine to enable autonomous rule tuning and dynamic ontology growth. Such learning-augmented reasoning would further enhance adaptability under uncertain or rapidly evolving data conditions. Second, multi-user and federated deployments will be explored to extend the framework's scalability across distributed semantic nodes while preserving data provenance and security. Third, integration with emerging network protocols such as QUIC/HTTP-3 will allow low-latency, loss-resilient transmission of semantic queries and feedback in large-scale streaming or industrial IoT environments. Finally, embedding the reasoning process within graph neural inference layers offers a promising route toward hybrid symbolic-subsymbolic intelligence, combining explainability with predictive power. Overall, the presented system demonstrates that ontology-based reasoning, when re-architected with modern computing principles, can achieve the responsiveness and robustness demanded by next-generation cyber-physical and diagnostic and monitoring infrastructures.

References

[1] J. Lee, B. Bagheri, and H. A. Kao. A cyber-physical systems architecture for Industry 4.0-based manufacturing systems. Manufacturing Letters, 3:18–23, 2015.

[2] L. Monostori. Cyber-physical production systems: Roots, expectations and R&D challenges. Procedia CIRP, 17:9–13, 2014.

[3] W. Zhang, D. Yang, and H. Wang. Data-driven methods for predictive maintenance of industrial equipment: A survey. IEEE Systems Journal, 13(3):2213–2227, 2019.

[4] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal. Long short-term memory networks for anomaly detection in time series. In Proceedings of ESANN, 89–94, 2015.

[5] F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608, 2017.

[6] D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf, and G. Z. Yang. XAI – Explainable artificial intelligence. AI Magazine, 40(2):44–58, 2019.

[7] I. Horrocks, P. F. Patel-Schneider, and F. Van Harmelen. From SHIQ to OWL: The making of a Web ontology language. Journal of Web Semantics, 1(1):7–26, 2003.

[8] F. Baader, I. Horrocks, and U. Sattler. Description logics as ontology languages for the Semantic Web. In Mechanizing Mathematical Reasoning, 228–248. Springer, 2005.

[9] A. I. Maarala, X. Su, and J. Riekki. Semantic reasoning for context-aware Internet of Things applications. IEEE Transactions on Emerging Topics in Computing, 4(4):461–473, 2016.

[10] A. I. Maarala, X. Su, and J. Riekki. Semantic data provisioning and reasoning for the Internet of Things. In Proceedings of UCAmI, 1–10, 2014.

[11] C. H. Liu, K. L. Chang, J. J. Y. Chen, and S. C. Hung. Ontology-based context representation and reasoning using OWL and SWRL. In Proceedings of CNSR, 215–220, 2010.

[12] K. Dentler, R. Cornet, A. Ten Teije, and N. De Keizer. Comparison of reasoners for large ontologies in the OWL 2 EL profile. Semantic Web, 2(2):71–87, 2011.

[13] G. Singh, S. Bhatia, and R. Mutharaju. OWL2Bench: A benchmark for OWL 2 reasoners. In Proceedings of the International Semantic Web Conference (ISWC), 81–96, 2020.

[14] B. Motik, B. C. Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz. OWL 2 Web Ontology Language profiles. W3C Recommendation, 2012.

[15] Y. K. Penya, J. C. Nieves, A. Espinoza, C. E. Borges, A. Pena, and M. Ortega. A distributed semantic architecture for smart grids. Energies, 5(11):4824–4843, 2012.

[16] Z. Liu, Z. Feng, X. Zhang, X. Wang, and G. Rao. RORS: Enhanced rule-based OWL reasoning on Spark. arXiv preprint arXiv:1605.02824, 2016.

[17] D. F. Barbieri, D. Braga, S. Ceri, E. D. Valle, and M. Grossniklaus. C-SPARQL: A continuous query language for RDF data streams. In Proceedings of WWW, 1061–1062, 2010.

[18] D. Le-Phuoc, M. Dao-Tran, J. X. Parreira, and M. Hauswirth. A native and adaptive approach for unified processing of linked streams and linked data. In Proceedings of ISWC, 370–388, 2011.

[19] M. Bourgais, F. Giustozzi, and L. Vercouter. Detecting situations with stream reasoning on health data obtained with IoT. Procedia Computer Science, 192:507–516, 2021.

[20] D. Keim, G. Andrienko, J. D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon. Visual analytics: Definition, process, and challenges. IEEE Computer Graphics and Applications, 28(5):20–31, 2008.

[21] C. Ware. Information Visualization: Perception for Design. 3rd ed. Morgan Kaufmann, 2013.
[22] L. Petnga and M. Austin. An ontological framework for knowledge modeling and decision support in cyber-physical systems. Advanced Engineering Informatics, 30(1):77–94, 2016.
[23] R. Reda, A. Carbonaro, V. de Boer, R. Siebes, R. van der Weerdt, B. Nouwt, and L. Daniele. Supporting smart home scenarios using OWL and SWRL. Sensors, 22(8):3022, 2022.
[24] L. Silva-Munoz, K. Medina, M. Marsicano, and M. Bonjour. Reasoning on the Semantic Web for adaptive hypermedia. Journal of Web Engineering, 7(2):148–165, 2008.
[25] Y. Kazakov. Consequence-driven reasoning for Horn SHIQ ontologies. In Proceedings of IJCAI, 2040–2045, 2009.
[26] B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Modular reuse of ontologies: Theory and practice. Journal of Artificial Intelligence Research, 31:273–318, 2008.
[27] H. Stuckenschmidt and M. Klein. Structure-based partitioning of large concept hierarchies. Journal of Web Semantics, 2(1):33–44, 2004.
[28] J.-P. Calbimonte. Ontology-based access to sensor data streams. In Proceedings of ISWC, 23–38, 2012.
[29] A. Lawan and A. Rakib. The Semantic Web Rule Language expressiveness extensions: A survey. arXiv preprint arXiv:1903.11723, 2019.
[30] N. D. Rodríguez, M. P. Cuéllar, J. Lilius, and M. D. Calvo-Flores. A fuzzy ontology for semantic modelling and recognition of human behavior. Knowledge-Based Systems, 66:46–60, 2014.
[31] J. Urbani, S. Kotoulas, E. Oren, and F. Van Harmelen. Scalable distributed reasoning using MapReduce. In Proceedings of ISWC, 634–649, 2010.
[32] A. d'Avila Garcez, T. R. Besold, L. De Raedt, P. Földiak, et al. Neural-symbolic learning and reasoning: Contributions and challenges. In AAAI Spring Symposium, 1–8, 2015.
[33] P. Hitzler and M. K. Sarker, editors. Neuro-symbolic Artificial Intelligence: The State of the Art. IOS Press, 2022.
[34] K. Ryabinin and S. Chuprina. Ontology-driven edge computing. Frontiers in Physics, 8:112, 2020.
[35] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. IEEE Internet of Things Journal, 3(5):637–646, 2016.

[36] A. Soularidis, K. I. Kotis, and G. A. Vouros. Real-time semantic data integration and reasoning in life- and time-critical systems. Electronics, 13(3):526, 2024.

[37] W. Xu, Z. Yang, D. W. K. Ng, M. Levorato, Y. C. Eldar, and M. Debbah. Edge learning for B5G networks with distributed signal processing: Semantic communication, edge computing, and wireless sensing. IEEE Journal of Selected Topics in Signal Processing, 17(1):9–39, 2023.

Biographies

Liu Yin obtained a Bachelor of Engineering degree in Electronic Information from Nanjing Normal University, China, in 2008, and a Master of Engineering degree in Software Engineering from Nanjing University in 2017. He currently serves as the Technical Director of Nanjing NARI Network Security Technology Co. Ltd., with research interests including the construction of network security protection systems for power monitoring systems, safety risk assessment, and hidden danger investigation and rectification. He also acts as a Corporate Tutor for Master's Students at Xiamen University.

Zhang Nanjing received a Bachelor of Science degree from Nanjing University, China, in 2008, and a Master of Engineering degree in Software Engineering from Nankai University in 2015. He is currently the General Manager of Nanjing NARI Network Security Technology Co. Ltd., focusing on research areas such as network security in power dispatching, situation awareness, and cryptographic application.

Zheng Qiang earned a Bachelor of Engineering degree in Software Engineering from Binjiang College of Nanjing University of Information Science & Technology, China, in 2014. He now works as a Product Manager at Nanjing NARI Network Security Technology Co. Ltd., with research fields including industrial control vulnerability mining, source code audit, reverse engineering analysis, and commercial cryptography application security evaluation.

Tang Yadong obtained a Bachelor of Engineering degree in Mechanical Design, Manufacturing and Automation from Nanjing Forestry University, China, in 2013. He currently holds the position of Project Supervisor at Nanjing NARI Network Security Technology Co. Ltd., specializing in research on network security protection systems for power monitoring systems and network security level protection technology.

Song Fuping received a Bachelor of Engineering degree in Automation from Hohai University, China, in 2021. He is currently an Engineering Consulting Engineer at Nanjing NARI Network Security Technology Co. Ltd., with research interests covering commercial cryptography application security evaluation, offensive and defensive penetration, and industrial control security.

Li Senwei obtained a Bachelor of Engineering degree in Electrical Engineering and Automation from Chengxian College of Southeast University, China, in 2020, and a Master of Engineering degree in Electronic Information from Nanjing Tech University in 2023. He now serves as an Engineering Consulting Engineer at Nanjing NARI Network Security Technology Co. Ltd., focusing on research areas such as industrial control security, penetration analysis, and network security emergency drills.