From Data Silos to Intelligent Operations: An AI-Based Approach to Ground Station Incident Investigation

Dimitri Accad and Nieves Salor Moral
Ground Segment & Operations Solutions
Starion Deutschland GmbH
Darmstadt, Germany
e-mail: {d.accad | n.salor}@stariongroup.eu

Andrea Di Luca
Fondazione Bruno Kessler
Trento, Italy
e-mail: adiluca@fbk.eu

Abstract—**The rapid increase in data generated by modern space systems has introduced significant challenges related to the scale, diversity, and fragmentation of operational information. Operators are increasingly overwhelmed by the daily influx of information, which is often loosely structured or spread across isolated systems. Although the integration of Artificial Intelligence (AI) into space ground segment operations offers promising opportunities, it demands addressing the inherent operational siloed contexts, security concerns, the diverse and evolving information sources, and the high-risk constraints of space missions. This work presents an Incident Investigation AI system that assists operators in analysing, understanding, and resolving ground station incidents threatening space missions. Traditional incident investigation relies on subjective expert judgment, manual log searches, and fragmented document access. This leads to data duplication, inconsistent formats, and knowledge loss. This paper addresses these limitations through dedicated data-processing pipelines that continuously extract, ingest, and harmonise information. The resulting knowledge powers agentic systems that assist operators throughout the incident lifecycle, including similarity identification, correlation analysis, root cause analysis, and automated conclusion generation. These components create an extensible framework that, beyond immediate operational gains, establishes a foundation for future collaborative agentic ecosystems capable of reasoning over complex operational data and supporting more autonomous, data-driven decision-making in space operations.**

Keywords-artificial intelligence; generative AI; agentic systems; space operations; incident investigation; data pipelines; vector databases; root cause analysis; ground stations; ground segment.

I. Introduction

Space systems generate a large amount of daily data. These systems can sometimes run into issues classified as incidents. These need to be investigated by operational personnel to take the relevant action in a timely manner. To perform this task, operators rely on experience, technical documentation, and reports on previous incident investigations. Accessing these sources is not a trivial task because of the data being scattered across siloed systems, being inconvenient to access, or becoming unavailable due to staff turnover and organizational memory loss. The investigation itself is a daunting manual process due to the overload of information to corroborate.

We designed a solution to support this investigation work by providing a suite of AI tools to empower operators with insights and sources they could hardly access before. All these tools are accessible through a User Interface built in React [1] where they can access past and current investigations, previous incident similarity graphical insights, chat with Engi (i.e., an AI agent with access to all the knowledge gathered over years of previous investigations), or review initial conclusions drafted by an autonomous AI assistant upon investigation creation.

The implementation of these agents presented technical challenges due to context length limitations of Large Language Models (LLMs) [2],[3] and available hardware constraints. Loading all available data into the model was therefore not viable, while external proprietary models were precluded by confidentiality concerns [4].

Recent advances in the AI landscape, such as Retrieval Augmented Generation (RAG) systems [5] and vector databases [6] enabled us to encode our data sources as vectors. Using similarity search methods (keyword, vector, or hybrid), relevant documents can be retrieved at run-time [7]. The adoption of this technology circumvented part of the software and hardware limitations described previously. It also provided opportunities to lay out the similarities generated in an interactive graph users can navigate to discover new insights based on investigations closely related to the one at hand. To improve user adoption, this approach was combined with the existing metadata of the documents, comprising manual correlations created by operators in previous investigations. The result is a knowledge graph visualisation [8] where the information is structured as nodes and relationships as edges between documents.

As a whole, the system is based on well-defined and well-scoped agents providing different features. These agents are powered by state-of-the-art methods to provide quality responses despite the small model size able to fit on the available hardware (i.e., 8 Billion parameter dense models). The designed network of expert agents was tailored to different scenarios with access to distinct but sometimes overlapping data sources. Space operations being a sensitive field where accuracy is paramount, multiple quality control techniques were deployed, such as re-ranking, hallucination detection, and relevance checks, to ensure integrity and transparency in the results generated by our system. These controls are distributed throughout the agentic workflow, from ensuring the relevance of the retrieved documents, to providing safeguards against potential model hallucinations [9] and checking the quality of the final generated answer.

While RAG-based systems have been explored for industrial troubleshooting in manufacturing [10] or aerospace-specific RAG evaluation (e.g., NASA's VALOR [11]), no deployed system currently addresses the specific challenges of ground station incident investigation, namely the combination of heterogeneous operational data sources, security-constrained on-premise deployment, and the need for multi-expert reasoning over years of accumulated institutional knowledge.

The main contributions of this work are:

- A unified data pipeline architecture that continuously harmonizes heterogeneous operational data sources (logs, reports, communications) into queryable knowledge bases.
- A multi-expert agentic system combining retrieval-augmented generation with quality control mechanisms (re-ranking, hallucination detection) tailored for mission-critical environments.
- An "API-in-the-loop" protocol enabling autonomous agents to operate within network-restricted secure environments.
- A deployed, operational system validated through real-world use at ESOC ground stations.

This paper is structured as follows: Section II discusses the design of the data pipelines to continuously extract and structure operational data from diverse sources while touching on the construction of the similarity graphs. Section III presents the agentic AI system architecture, detailing the agent-assisted incident investigation workflow with its quality control mechanisms, as well as the automated conclusion generation process. Finally, conclusions and future work are drawn in Sections IV and V, respectively.

II. Knowledge Base Construction

Building an effective knowledge base relies on two essential components: properly processing incoming data and structuring it for practical use. The following subsections outline the technical architecture and methodologies employed to achieve both objectives.

A. Pipeline Architecture and Processing

Ground stations generate large amounts of data daily, stored in different but isolated environments. At the European Space Operations Centre (ESOC), these include Station Terminal Computer logs (STCs), Anomaly Report Traces (ARTs), technical documents, and communication records. To retrieve all this information and make it available to process through our pipelines, scheduled cron jobs execute daily retrieval operations, extracting data produced within the last 24 hours, or immediately, depending on the source. The purpose of the pipeline described in Figure 1 is to unify the data by extracting structured information and generating embeddings from their content and storing it in either a relational or vector database.

The pipeline first categorizes files by type, as each category requires a distinct extraction methodology, and subsequently stores the processed data in separate vector store collections. Source documents undergo deduplication at two levels: file-level checks using hash-based algorithms [12], and record-level checks for CSV files where individual rows may duplicate or update previous entries.

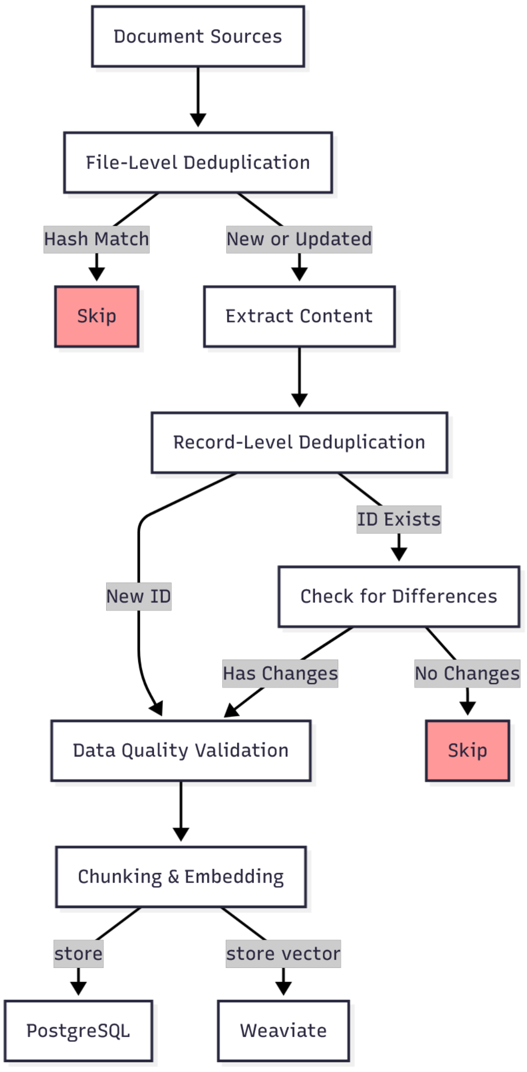

Figure 1. Pipeline architecture and workflow.

Each entry is analysed for missing information, as the data have mostly been created by humans over multiple years. Inconsistencies in notation, schema changes or mistakes are common and need to be handled. Next, a rule-based approach discards entries missing critical fields (e.g., main description) while retaining those with missing non-critical fields, such as cross-references or metadata.

To store the vector representation of each entry, the first step is chunking the information to ensure better retrieval performance [13]. Ensuring temporal and context coherence through these fragments is key to enabling transparent sourcing of the information used for each query. Chunks are obtained by overlapping them and enriching their metadata with timestamps, source identifiers, and their position in the overall text. After segmentation and embedding, the vector representation are stored in the chosen vector database, Weaviate [14], along with the positional metadata described above, while the structured information and remaining metadata are stored in a relational database, PostgreSQL [15]. The information from one database can be traced back to the other thanks to a unique identifier. The simplified pipeline workflow is described in Figure 1.

Additionally, users can manually upload new data (e.g., files of similar structure to the one described above, or images and PDFs that may contain illustrations) to the system. As the essence of this information, where content and structure are unknown, is not efficiently captured by traditional methods, a multimodal approach using ColPali [16] was adopted, which avoids lossy text-only segmentation and allows images, diagrams, and tables to be retrieved alongside text passages. The open-source Gemma3 [17] vision Large Language Model (vLLM) [18] was selected to understand the information and create embeddings from them, which are then stored in Weaviate.

B. *Structuring Data for Practical Use*

The vector database is segregated into different collections for each document type, enabling similarity searches on specific entries. The results are enriched using the metadata link to the relational database described previously. The resulting information is leveraged to produce a similarity graph using t-Distributed Stochastic Neighbors Embedding (t-SNE) [19], a dimensionality reduction technique that visualizes high-dimensional data by projecting each data point onto a 2D map. This processing also clusters related ideas, boosting the insights of the similarity graph by grouping similar concepts together.

In order to offer more context to the users, the solution also generates a second type of plot similar to a knowledge graph. It is built using metadata on related incidents as links to other files to enrich the graphic, as demonstrated in Figure 2.

This view is fully browsable and interactive; clicking on each node provides more information about the incident and empowers users with new ways to investigate by traversing the knowledge graph, supporting rapid and evidence-driven investigations across missions. Additional metadata related to dates or ground station origin are also present, which the AI assistant will leverage to further filter the results to investigate. Figure 3 summarises the post-processing data storage architecture and the flow of information from these sources to the visualisation services and AI agents.

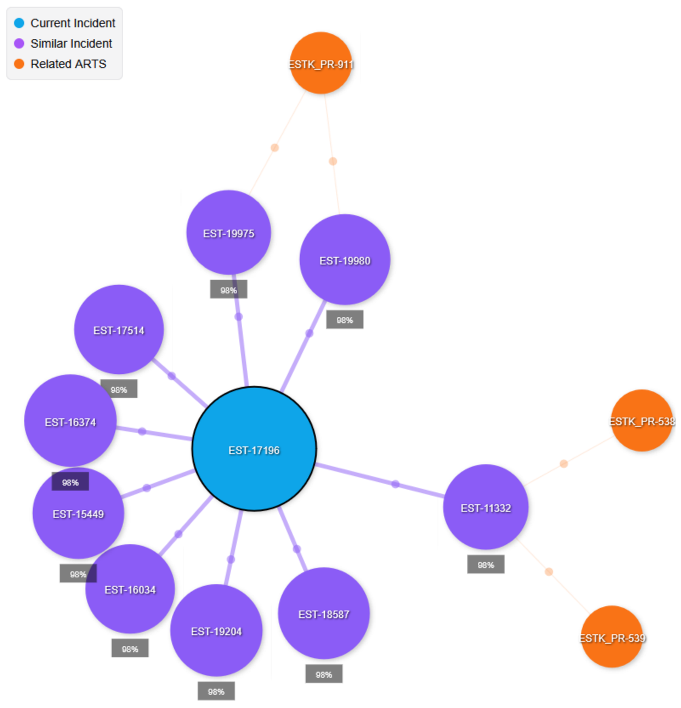

Figure 2. Knowledge Graph Visualisation.

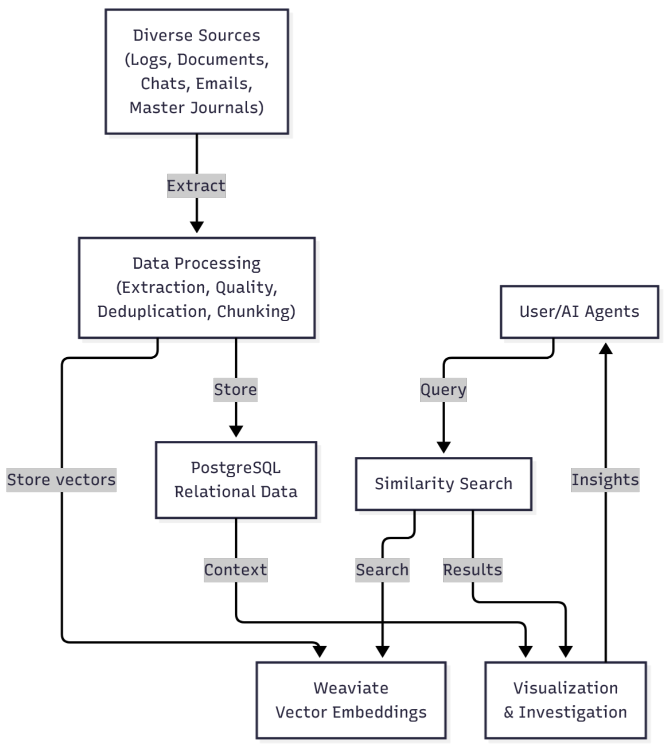
Diverse Sources (Logs, Documents, Chats, Emails, Master Journals)
Extract
Data Processing (Extraction, Quality, Deduplication, Chunking)
User/AI Agents
Store
Query
Store vectors
PostgreSQL Relational Data
Similarity Search
Insights
Context
Search
Results
Weaviate Vector Embeddings
Visualization & Investigation

Figure 3. Data Lifecycle.

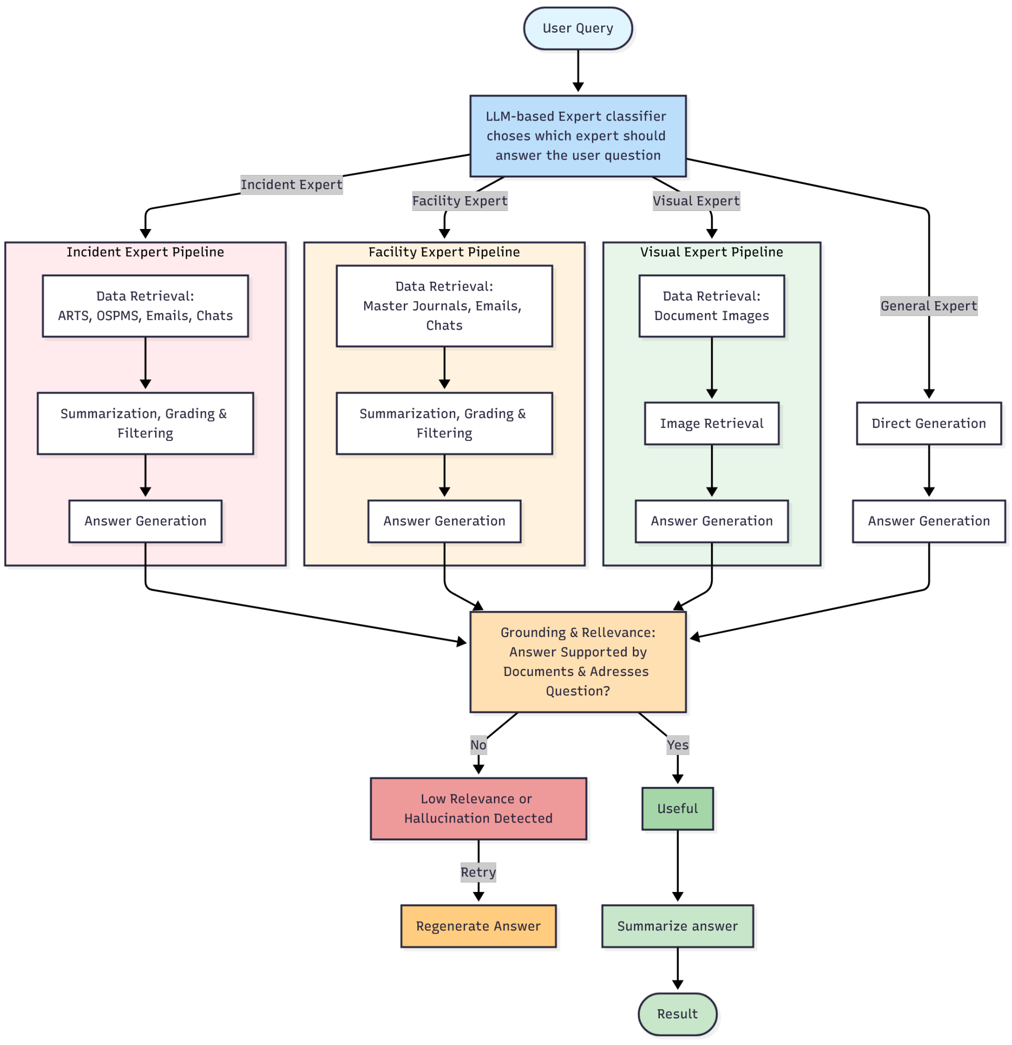
User Query
LLM-based Expert classifier choses which expert should answer the user question
Incident Expert
Facility Expert
Visual Expert
General Expert
Incident Expert Pipeline
Data Retrieval: ARTS, OSPMS, Emails, Chats
Summarization, Grading & Filtering
Answer Generation
Facility Expert Pipeline
Data Retrieval: Master Journals, Emails, Chats
Summarization, Grading & Filtering
Answer Generation
Visual Expert Pipeline
Data Retrieval: Document Images
Image Retrieval
Answer Generation
Direct Generation
Answer Generation
Grounding & Rellevance: Answer Supported by Documents & Adresses Question?
No
Yes
Low Relevance or Hallucination Detected
Retry
Regenerate Answer
Useful
Summarize answer
Result

Figure 4. AI Architecture diagram.

III. Agentic AI for Incident Investigation

The assistants are designed using open-source software and models, with LangGraph [20] orchestrating the agent steps and qwen3:8b [21] serving as an expert model. Model selection was driven by practical constraints: most open-source LLMs support fewer than 250K tokens, compared to 1M tokens in some closed-source alternatives [22],[23]. Furthermore, open-source models with larger context windows require multiple GPUs to store model weights and the Key-Value (KV) cache [24], which were unavailable during development. These hardware, architectural, and licensing constraints made qwen3:8b the optimal choice, balancing performance with size.

A. *Agent-Assisted Incident Investigation*

Our approach employs the multi-expert workflow architecture shown in Figure 4. The first step identifies and expands acronyms in the user query using a predefined dictionary. This step is essential because domain-specific terminology can be misinterpreted by general-purpose LLMs unfamiliar with the specialized lexicon. The updated query is routed by a model which selects the most appropriate experts from a pool of four specialists with access to different data sources and able to run in parallel.

- **Incident Expert**: Focused on incident reports and communications for anomaly-centric analyses. It can answer questions about similar past incidents and anomalies.
- **Ground Station Expert**: Specialized in ground station logs and communications for timeline reconstruction and subsystem-level investigations.
- **Visual Document Expert**: Interprets visual content including diagrams, charts, scanned pages, and embedded images in documents.
- **General Expert**: Handles general questions outside the scope of other specialists.

The answers from the specialists are then gathered and graded for quality before being summarised and returned to the user.

The retrieval part of the workflow is run against the vector database described in previous sections, and uses hybrid search [25] and automatic filtering based on the user query.

Hybrid search is used for textual sources. It is a combination of semantic similarity (vector search) with keyword-based scoring, specifically Best Match 25 Model with Extension to Multiple Weighted Fields (BM25F) [26]. This method strikes a balance between natural language queries and technical identifiers, such as error codes or timestamps. Filters are created autonomously by the agent using dspy [27] to provide consistent formatting of the filtering arguments. Filters enable agents to execute focused searches based on the relevant metadata (e.g., dates, stations, etc.) to improve the relevance of the retrieved sources.

The visual document expert is a special case as it needs to search through multi-modal documents, such as PDF manuals, schematics, or diagrams. Retrieval is performed using the ColPali framework. This allows the assistant to ground its reasoning on figures and tables, as well as textual evidence.

Once the relevant chunks are identified, their original document is retrieved to put each chunk in context before a summarisation step condenses each entry into concise representations preserving key information, such as timestamps, subsystem identifiers, and error codes. These representations are then re-ranked using another LLM to grade them against the user query, this method has been shown to improve answers by refining the quality of the subset [28]. If not enough documents are deemed useful, the filters are adapted and the user query is rewritten for performance inside a retry loop.

After retrieval, each expert generates a response to the user query. The response then undergoes two validation steps using LLM-as-a-judge methods [29]: a hallucination check to ensure the answer is grounded in the provided documents, and a quality check to verify that it addresses the user question. If either validation fails, query parameters are adjusted, and the retrieval is retried.

Once the system's answers pass all quality checks, it is returned to the user with direct links to the documents used for the generation in the User Interface (UI) thanks to the metadata. All these steps ensure that the quality meets the standards of this mission-critical domain while providing clear and transparent answers within seconds.

B. Automated Conclusion Generation

The last developed agent is focused on generating the investigation closure. Its objectives are to propose conclusions, a root cause, the impact on systems, and mitigation actions for each incident. However, the implementation contrasts with previous agents. This one is a fully autonomous agentic loop with tool calling [30], thus, it mimics how issues would be investigated by real users. The reason for the different implementation was simply because users started to trust the system and felt more comfortable with a pure agentic loop. Its architecture is described in Figure 5.

The main agent node represented in purple in Figure 5 uses qwen3:8b in thinking mode. By enabling this feature, the model is allowed more time to reflect on its actions and current state of its internal investigation; great improvements have been observed in the way the model handles the tools at its disposal to steer the investigation in a meaningful direction at the cost of increased inference time.

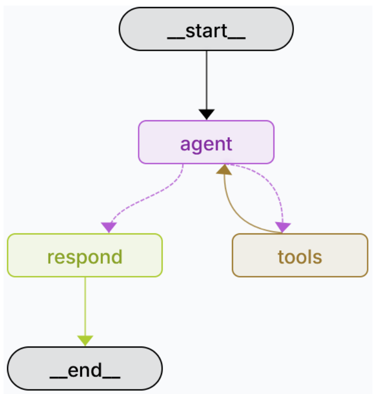

Figure 5. Conclusion Agent Architecture.

A single search tool is made available for the model to use in autonomy. The agent decides how to use it, with vector, keyword, or hybrid searches being performed independently based on what is thought to be the relevant step to perform. The output of the tool is post-processed and stored in the agent state [31] in a way that filters out duplicate results in subsequent queries. This state is ingested into the model prompt at each loop to prevent context expansion and information dilution after a few loops, maintaining the agent performance in long-running sessions; this is called context engineering [32].

With the agent being deployed on a secure server limiting outgoing requests, a networking issue was encountered, preventing the model from enriching the data retrieved from the vector database (running in the same environment) with the metadata of the relational database (running on another server). The nominal flow would have been for a GET request to be made against the relational database for the documents retrieved so far to enrich them with added context. The Human-In-The-Loop (HITL) protocol [33],[34] was used to address this issue by redirecting the call to the API endpoint of our database instead of asking for user feedback. The attempt was successful, as the HITL protocol does not rely on emitting a request from the server, but instead our request is embedded in the response body of the initial call to the model. By processing the response body on the other end of the system, which has unrestricted access to PostgreSQL, the requested data can be sent back to the agent by resuming the conversation where it was left off with the protocol. Hence, "API-in-the-loop" was implemented to ensure the agent could access the relevant data sources for the investigation.

At the start of each loop, the agent's partial findings are reflected upon, and a decision is made whether to investigate further in a set direction or if enough information has been gathered to draw conclusions. In the latter case, the proposed conclusions, root cause, impact, and mitigation strategies derived from the investigation are written by the main agent, making sure all sources used to generate the response are cited. Additionally, the model is prompted to score its confidence in the provided answers based on how well sources correlate. This feature builds trust and focuses the user's attention on low scoring answers to encourage further manual or assisted investigation. However, this free-form text cannot be used as-is by the system as structured JSON output is expected to populate the UI fields with this information. A response agent is subsequently called after conclusions are drawn and is set up to answer in structured JSON format following the expected schema of the calling service. It is instructed to simply reformulate the main agent findings without changing its content, separating the content of the answer from the confidence score it self-evaluated. This approach provides flawless accuracy in the generated JSON schema, ensuring the system does not fail due to a formatting mistake.

The resulting assistant is able to adapt the amount of time spent investigating issues based on their complexity and to reach documents that would not have been considered in the investigation agent described in Section III-A, thanks to the ability to investigate step by step, delving deeper into the analysis of relevant documents uncovered at each loop. This design implements a human-in-the-loop paradigm, as the generated conclusions are presented to users for review before being added to the incident database, creating a powerful hybrid approach where the system conducts comprehensive, autonomous investigations while human review ensures that only verified findings enter the organizational knowledge base.

IV. Conclusion

This paper presents an already-in-use AI-driven incident investigation framework designed to mitigate the fragmentation of operational data and the loss of institutional knowledge in ground station environments. The framework integrates automated data processing pipelines that continuously aggregate and standardize information across heterogeneous sources, interactive visualization interfaces revealing relationships and similarities among data sources, a multi-expert agentic system for supporting user investigations, and an autonomous reasoning agent formulating preliminary conclusions for each incident. The system incorporates rigorous validation mechanisms and transparency protocols to maintain accuracy and explainability while operating within hardware and security constraints inherent to these critical environments.

The practical deployment revealed innovative solutions including the "API-in-the-loop" protocol for network-restricted environments and context engineering techniques for managing open-source LLM constraints. Initial results have been promising, with users expressing satisfaction with the system's capabilities. Ongoing work consists in further refinement of the quality of the answers and mitigation of hallucinations inherent in LLM-based systems that require continuous monitoring and improvement.

Beyond immediate operational gains, this work establishes a foundation for future collaborative agentic ecosystems capable of reasoning over complex data and supporting more autonomous, data-driven decision-making in space operations.

V. Future Work

Future work will expand the existing solution, where two main directions are already identified and ongoing:

Unified Agentic Architecture: The multi-expert agent (Section III-A) and the conclusion agent (Section III-B) currently use different architectures. Migrating the multi-expert system to a fully autonomous tool-calling architecture, as implemented in the conclusion agent, would enable deeper investigation capabilities and more flexible reasoning across all user interactions.

Real-Time Collaborative Investigation: The current system provides answers after investigation completes. Future iterations will surface the agent's reasoning process in real-time, allowing operators to observe which documents are being consulted and why. Implementing human-in-the-loop interaction would enable operators to guide, correct, or redirect the investigation mid-analysis. This transparency and control would transform the system from a tool that delivers answers into a "co-pilot" experience that builds more trust with the user and could define the future vision for autonomous space operations.

References

[1] Meta, *React: A JavaScript library for building user interfaces*, 2025. Accessed: 2025-01-29. [Online]. Available: https://react.dev/.

[2] A. Vaswani *et al.*, *Attention is all you need*, 2023. arXiv: 1706.03762 `[cs.CL]`. Accessed: 2025-01-29. [Online]. Available: https://arxiv.org/abs/1706.03762.

[3] W. X. Zhao *et al.*, *A survey of large language models*, 2025. arXiv: 2303.18223 `[cs.CL]`. Accessed: 2025-01-29. [Online]. Available: https://arxiv.org/abs/2303.18223.

[4] S. Chen, S. Wong, L. Chen, and Y. Tian, *Extending context window of large language models via positional interpolation*, 2023. arXiv: 2306.15595 `[cs.CL]`. Accessed: 2025-01-29. [Online]. Available: https://arxiv.org/abs/2306.15595.

[5] P. Lewis *et al.*, *Retrieval-augmented generation for knowledge-intensive nlp tasks*, 2021. arXiv: 2005.11401 `[cs.CL]`. Accessed: 2025-01-29. [Online]. Available: https://arxiv.org/abs/2005.11401.

[6] L. Ma *et al.*, *A comprehensive survey on vector database: Storage and retrieval technique, challenge*, 2025. arXiv: 2310.11703 `[cs.DB]`. Accessed: 2025-01-29. [Online]. Available: https://arxiv.org/abs/2310.11703.

[7] W. Li *et al.*, *Approximate nearest neighbor search on high dimensional data — experiments, analyses, and improvement (v1.0)*, 2016. arXiv: 1610.02455 `[cs.DB]`. Accessed: 2025-01-29. [Online]. Available: https://arxiv.org/abs/1610.02455.

[8] A. Hogan *et al.*, “Knowledge graphs,” *ACM Computing Surveys*, vol. 54, no. 4, pp. 1–37, Jul. 2021, ISSN: 1557-7341. DOI: 10.1145/3447772. [Online]. Available: http://dx.doi.org/10.1145/3447772.

[9] Z. Ji *et al.*, “Survey of hallucination in natural language generation,” *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, Mar. 2023, ISSN: 1557-7341. DOI: 10.1145/3571730. [Online]. Available: http://dx.doi.org/10.1145/3571730.

[10] A. Narimani and S. Klarmann, "Integration of Large Language Models for Real-Time Troubleshooting in Industrial Environments based on Retrieval-Augmented Generation (RAG)," in *Proceedings of the 7th European Industrial Engineering and Operations Management Conference*, Augsburg, Germany: IEOM Society International, 2024, ISBN: 979-8-3507-1737-2. DOI: 10.46254/EU07.20240085. [Online]. Available: https://ieomsociety.org/proceedings/2024germany/85.pdf.

[11] K. S. Prakash *et al.*, "VALOR: Validation for Aerospace LLM Output and Reasoning," National Aeronautics and Space Administration, Ames Research Center, Moffett Field, California, NASA Technical Memorandum NASA/TM–20260000076, 2026. [Online]. Available: https://ntrs.nasa.gov/citations/20260000076.

[12] J. Alakuijala, B. Cox, and J. Wassenberg, *Fast keyed hash/pseudo-random function using simd multiply and permute*, 2017. arXiv: 1612.06257 [cs.CR]. Accessed: 2025-01-29. [Online]. Available: https://arxiv.org/abs/1612.06257.

[13] R. Schwaber-Cohen and A. Patel, *Chunking Strategies for LLM Applications*, 2025. Accessed: 2025-01-29. [Online]. Available: https://www.pinecone.io/learn/chunking-strategies/.

[14] Weaviate, *Weaviate: Vector Database*, 2025. Accessed: 2025-01-29. [Online]. Available: https://weaviate.io/.

[15] PostgreSQL Global Development Group, *PostgreSQL: The World's Most Advanced Open Source Relational Database*, 2025. Accessed: 2025-01-29. [Online]. Available: https://www.postgresql.org/.

[16] M. Faysse *et al.*, *Colpali: Efficient document retrieval with vision language models*, 2025. arXiv: 2407.01449 [cs.IR]. Accessed: 2025-01-29. [Online]. Available: https://arxiv.org/abs/2407.01449.

[17] G. Team *et al.*, *Gemma 3 technical report*, 2025. arXiv: 2503.19786 [cs.CL]. Accessed: 2025-01-29. [Online]. Available: https://arxiv.org/abs/2503.19786.

[18] F. Lin, *Vision language models: A survey of 26k papers*, 2025. arXiv: 2510.09586 [cs.CV]. Accessed: 2025-01-29. [Online]. Available: https://arxiv.org/abs/2510.09586.

[19] T. T. Cai and R. Ma, *Theoretical foundations of t-sne for visualizing high-dimensional clustered data*, 2022. arXiv: 2105.07536 [stat.ML]. Accessed: 2025-01-29. [Online]. Available: https://arxiv.org/abs/2105.07536.

[20] LangChain, *LangGraph: Build resilient language agents as graphs*, 2025. Accessed: 2025-01-29. [Online]. Available: https://langchain-ai.github.io/langgraph/.

[21] A. Yang *et al.*, *Qwen3 technical report*, 2025. arXiv: 2505.09388 [cs.CL]. Accessed: 2025-01-29. [Online]. Available: https://arxiv.org/abs/2505.09388.

[22] OpenAI, *OpenAI: Introducing GPT 4.1*, 2025. Accessed: 2025-01-29. [Online]. Available: https://platform.openai.com/docs/models/gpt-4.1.

[23] Anthropic, *Claude Sonnet 4 now supports 1M tokens of context*, 2025. Accessed: 2025-01-29. [Online]. Available: https://claude.com/blog/1m-context.

[24] R. Pope *et al.*, *Efficiently scaling transformer inference*, 2022. arXiv: 2211.05102 [cs.LG]. Accessed: 2025-01-29. [Online]. Available: https://arxiv.org/abs/2211.05102.

[25] Weaviate, *Weaviate: Hybrid search*, 2025. Accessed: 2025-01-29. [Online]. Available: https://docs.weaviate.io/weaviate/search/hybrid.

[26] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: Bm25 and beyond," *Foundations and Trends in Information Retrieval*, vol. 3, pp. 333–389, Jan. 2009. DOI: 10.1561/1500000019.

[27] O. Khattab *et al.*, *Dspy: Compiling declarative language model calls into self-improving pipelines*, 2023. arXiv: 2310.03714 [cs.CL]. Accessed: 2025-01-29. [Online]. Available: https://arxiv.org/abs/2310.03714.

[28] X. Ma, Y. Gong, P. He, H. Zhao, and N. Duan, *Query rewriting for retrieval-augmented large language models*, 2023. arXiv: 2305.14283 [cs.CL]. Accessed: 2025-01-29. [Online]. Available: https://arxiv.org/abs/2305.14283.

[29] J. Gu *et al.*, *A survey on llm-as-a-judge*, 2025. arXiv: 2411.15594 [cs.CL]. Accessed: 2025-01-29. [Online]. Available: https://arxiv.org/abs/2411.15594.

[30] LangChain, *LangChain: Tool calling documentation*, 2025. Accessed: 2025-01-29. [Online]. Available: https://docs.langchain.com/oss/python/langchain/tools.

[31] LangChain, *LangChain: State documentation*, 2025. Accessed: 2025-01-29. [Online]. Available: https://docs.langchain.com/oss/python/langgraph/use-graph-api.

[32] L. Mei *et al.*, *A survey of context engineering for large language models*, 2025. arXiv: 2507.13334 [cs.CL]. Accessed: 2025-01-29. [Online]. Available: https://arxiv.org/abs/2507.13334.

[33] X. Wu *et al.*, "A survey of human-in-the-loop for machine learning," *Future Generation Computer Systems*, vol. 135, pp. 364–381, Oct. 2022, ISSN: 0167-739X. DOI: 10.1016/j.future.2022.05.014. [Online]. Available: http://dx.doi.org/10.1016/j.future.2022.05.014.

[34] LangChain, *LangChain: Human in the loop protocol*, 2025. Accessed: 2025-01-29. [Online]. Available: https://docs.langchain.com/oss/python/langchain/human-in-the-loop.