

An AI Development Platform Utilizing HITL Verification: From Source Data to Tailored Domain Assistants

*Leonard V. Lopes and Martha C. Brown
Langley Research Center, Hampton, Virginia*

*James J. Deucher, Shane D. Moxley, and Lauren E. Steele
Analytical Mechanics Associates, Hampton, Virginia*

*Gregory L. Mekkes
Amentum, Hampton, Virginia*

NASA STI Program Report Series

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI Program provides access to the NTRS Registered and its public interface, the NASA Technical Report Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

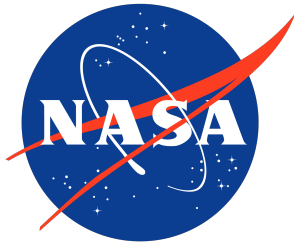
- **TECHNICAL PUBLICATION.** Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.
- **TECHNICAL MEMORANDUM.** Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.
- **CONTRACTOR REPORT.** Scientific and technical findings by NASA-sponsored contractors and grantees.

- **CONFERENCE PUBLICATION.** Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.
- **SPECIAL PUBLICATION.** Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.
- **TECHNICAL TRANSLATION.** English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI Program, see the following:

- Access the NASA STI program home page at <http://www.sti.nasa.gov>
- Help desk contact information: <https://www.sti.nasa.gov/sti-contact-form/> and select the "General" help request type.



An AI Development Platform Utilizing HITL Verification: From Source Data to Tailored Domain Assistants

*Leonard V. Lopes and Martha C. Brown
Langley Research Center, Hampton, Virginia*

*James J. Deucher, Shane D. Moxley, and Lauren E. Steele
Analytical Mechanics Associates, Hampton, Virginia*

*Gregory L. Mekkes
Amentum, Hampton, Virginia*

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

June 2026

Acknowledgments

The authors would like to acknowledge the Revolutionary Vertical Lift Technology (RVLТ) project and the Subsonic Vehicle Technologies and Tools (SVTT) project under the Advanced Air Vehicles Program (AAVP), as well as the Transformational Tools and Technologies (TTT) project under the Transformative Aeronautics Concepts Program (TACP). The authors also acknowledge the Artificial Intelligence Training and Hands-on Experience at NASA (AiTHENA) Initiative, founded at National Aeronautics and Space Administration (NASA) Langley Research Center, that supports the development of mission-relevant artificial intelligence (AI) capabilities through applied training, collaborative technical engagement, and hands-on project experience. In addition, the authors would like to acknowledge the Flight Vehicle Acoustics Branch (D314) at NASA Langley Research Center and Analytical Mechanics Associates, Inc. for their support.

This document was created with text generation assistance from the NASA generative AI tool ChatGSFC. The content has been thoroughly reviewed, heavily modified, and verified by Leonard V. Lopes and the authors.

The use of trademarks or names of manufacturers in this report is for accurate reporting and does not constitute an official endorsement, either expressed or implied, of such products or manufacturers by the National Aeronautics and Space Administration.

Available from:

NASA STI Program / Mail Stop 150
NASA Langley Research Center
Hampton, VA 23681-2199

Abstract

This paper introduces the Human-Enabled Learning & Evaluation Network for Assistants (HELENA), an end-to-end, locally deployable artificial intelligence (AI) development platform designed to transform proprietary or non-proprietary source data into specialized, tailored AI assistants. The HELENA framework integrates retrieval-augmented generation (RAG) to ground models and low-rank adaptation (LoRA) to instill response characteristics and formatting. The platform employs reinforcement supervised fine tuning (SFT) through a slow-burn human verification process that converts AI-generated synthetic data into verified training sets to ensure the highest level of accuracy and alignment. A critical advantage of the HELENA architecture is its portability; the system is designed to be deployed entirely behind secure firewalls, ensuring stringent data protection and eliminating external vendor vulnerabilities. Finally, this paper reviews the preliminary training progress of current assistants and details future capabilities, including knowledge graph integration, active knowledge elicitation, multi-agent collaboration, and the development of tool-using tailored assistants. Ultimately, HELENA serves as a secure and scalable blueprint for capturing institutional knowledge and deploying specialized AI.

1 Introduction

Knowledge capture is a fundamental pillar of institutional operational continuity and long-term organizational success. As organizations scale and evolve, they inevitably face the challenges of workforce turnover, often resulting in a critical knowledge attrition as subject matter experts (SMEs) retire or transition. This is a particularly critical issue in large engineering organizations, where a lack of accessible knowledge resources often impacts engineering efforts with practitioners unaware that past solutions exist. Without robust processes for the transfer of expertise, organizations inevitably suffer from prolonged onboarding times, duplicated engineering efforts, and the permanent loss of specialized technical insights. Consequently, effectively capturing, synthesizing, and retrieving this wealth of complex, nebulous information has evolved from a routine administrative function into an ongoing operational challenge for modern enterprises. National Aeronautics and Space Administration (NASA) has long recognized the critical importance of systematically preserving and transferring institutional knowledge to support mission success across decades-long programs. As early as 1987, researchers highlighted the need for robust design knowledge capture approaches, particularly for the Space Station era, to go beyond surface-level documentation and preserve deeper rationale for training, fault management, and automation [1, 2]. Building on foundational work in knowledge-capture engineering and expert systems from the early 1980s, these efforts established knowledge capture as essential for intelligent systems architecture and long-term engineering excellence at NASA.

Historically, organizations have relied on fragmented document-centric repositories along with code documentation and embedded comments to mitigate this knowledge attrition. Traditional knowledge-capture methods demand significant

time and expensive technical expertise to generate standard operating procedures (SOPs), technical manuals, internal wikis, published documents (such as conference or journal papers), and decentralized databases. While these systems provide necessary foundational documentation, they lack contextual synthesis and often omit the nuanced details required to leverage easily the capabilities they document. This becomes a critical failure point when the original SME is no longer available to fill in the gaps. Consequently, this unwritten expertise is typically preserved only through direct mentorship, informal SME consultations, and localized team experience, rather than within formalized, accessible knowledge bases.

Beyond the initial investment required to generate static documentation, the practical reality of fast-paced engineering environments is that documentation is frequently neglected [3]. The prioritization of active development over retrospective recording leaves immediate knowledge gaps and accelerates system rot, as capabilities evolve without corresponding historical records. Furthermore, when documentation is produced, it is often remarkably difficult to digest. Readers face a needle-in-a-haystack search that discourages them from utilizing the very capabilities the documentation is meant to make accessible. Finally, even in idealized scenarios where comprehensive and well-structured documentation exists and is accessible to a reader, human behavioral factors introduce a significant barrier to its utility: engineers frequently bypass lengthy static documents in favor of seeking direct answers from colleagues [4]. This inherent reluctance to synthesize static text manually renders even the most meticulously maintained repositories inefficient for rapid problem resolution.

The advent of artificial intelligence (AI), specifically the rapid evolution of large language models (LLMs) [5, 6], represents a fundamental paradigm shift in how institutional knowledge is captured and accessed. Instead of requiring users to passively search static documents, LLMs provide conversational synthesis that mirrors natural peer-to-peer inquiry. To achieve this within specialized engineering environments, modern AI frameworks leverage retrieval-augmented generation (RAG) [7] to dynamically ground the model’s responses in factual, proprietary documentation. This effectively isolates the metaphorical needle within the organizational haystack. Furthermore, parameter-efficient fine-tuning techniques, such as low-rank adaptation (LoRA) [8], allow these models to be tailored efficiently to exhibit domain-specific behaviors, terminology, and formatting. Together, these technologies transform rigid, document-centric repositories into dynamic, interactive assistants capable of synthesizing complex, nebulous data on demand.

Despite the transformative potential of these AI technologies, relying on generic, publicly hosted foundation models (such as Google’s NotebookLM or OpenAI’s customized chatbots) presents significant challenges for enterprise integration. Commercial off-the-shelf models introduce unacceptable data privacy vulnerabilities for proprietary engineering data, and they inherently lack the precise alignment required for technical workflows. To bridge this gap, organizations require a secure, end-to-end development pipeline capable of generating AI assistants on the fly while operating entirely behind security firewalls. These assistants are not only specialized in their knowledge but tailored in their behavior. To address these critical constraints, this paper introduces the Human-Enabled Learning & Evaluation Network for Assistants

(HELENA). By combining RAG for content specialization, LoRA for behavioral tailoring, and a structured human-in-the-loop (HITL) verification pipeline in the form of supervised fine tuning (SFT) [9], HELENA provides a locally deployable incubator for customized AI. The subsequent sections of this paper detail the constraints driving this architecture, outline the HELENA framework and data flow, discuss secure deployment strategies, and explore future capabilities such as active knowledge elicitation and the development of tool-using agentic assistants.

2 System Constraints

While the potential applications of LLMs are vast, trusting and deploying these models within secure, enterprise-level engineering environments using appropriate hardware introduces a rigid set of operational and architectural boundaries. The HELENA framework was explicitly designed to satisfy these constraints. This ensures that all resulting tailored assistants are not only capable but also secure, verifiable, and deployable within restricted infrastructure. To ensure a structured, lifecycle-driven approach to AI development, the ten primary constraints driving the HELENA architecture are organized into three sequential operational categories: 1) Foundational Infrastructure and Security, 2) System Incubation and Governance, and 3) Output Integrity and Operational Trust.

Category 1: Foundational Infrastructure and Security

Before an AI assistant can be developed, the underlying physical and digital environment must be absolutely secured. The following constraints define the non-negotiable boundaries regarding data isolation, user access segmentation, and localized computational limits.

Constraint 1: Data Sovereignty and Security

The most critical constraint governing the deployment of enterprise AI is data sovereignty [10]. Specialized engineering organizations generate vast quantities of proprietary, sensitive, and export-controlled data that cannot be exposed to external networks [11]. Consequently, any viable knowledge capture system must operate with zero reliance on third-party application programming interfaces (APIs) or commercial cloud-based foundation models (e.g., OpenAI, Anthropic). This isolation is required not only to mitigate the inherent data leakage risks associated with external model hosting [12], but also to eliminate operational vulnerabilities tied to vendor lock-in and recurring subscription models. Furthermore, reliance on external cloud entities exposes the enterprise to sudden service disruptions driven by regulatory volatility, geopolitical shifts, or executive mandates that can abruptly restrict access to specific commercial AI providers. Therefore, all phases of the AI life cycle, including data ingestion, document embedding, vectorization, model fine-tuning, and inference, must execute entirely on-premises behind secure firewalls. This firewall-restricted approach ensures operational continuity and guarantees that proprietary source data remain strictly within the organization’s governance boundary.

Constraint 2: Access Segmentation and Authorization

While operation behind secure firewalls is necessary to preserve data sovereignty (as stated in Constraint 1), enterprise deployment also requires strict internal control over who can retrieve and interact with specific knowledge domains. Within technical organizations, proprietary or sensitive information is often segmented by project, program, export-control restrictions, contractual boundaries, or role-based need-to-know policies. Therefore, the system is constrained not only by external security requirements, but also by internal access segmentation and authorization. This is well aligned with established access-control and auditing practices in federal information systems guidance [10].

A viable architecture should ensure that the retrieval pipeline respects user privileges at every stage of assistant interaction. This includes controlling which documents are indexed for a given assistant and which source materials may be cited in generated responses. In practice, this implies role-based or project-based access controls, partitioned knowledge stores, and auditable retrieval behavior. It is insufficient for the platform to keep data inside the firewall if it cannot also prevent internal overexposure of restricted information [10].

Constraint 3: Hardware Portability and Compute Limitations

Operating entirely on-premises behind secure firewalls, as dictated by Constraint 1, inherently limits an enterprise AI platform to localized computational resources during both development and deployment phases. Unlike cloud-based solutions that leverage massive, distributed graphics processing unit (GPU) clusters, an enterprise-deployed system must be lightweight and hardware-agnostic.

During the incubation phase, performing full-parameter fine-tuning on large foundation models is computationally prohibitive. Consequently, the architecture must utilize parameter-efficient fine-tuning (PEFT) methods [13], such as LoRA [8], and rely on RAG [7] to store domain knowledge externally. This maximizes computational efficiency during assistant creation.

Furthermore, the resulting domain assistants must be operationally viable outside of this centralized incubation environment. The generated assistants cannot rely on massive foundation models that require dedicated enterprise server racks for daily inference. Instead, specialized assistants must be built upon optimized, smaller-parameter LLMs (e.g., 7 billion (7B) to 13 billion (13B) parameters) [14] paired with their respective LoRA adapters. This ensures the fully trained assistant can be seamlessly exported and executed locally on standard, consumer-grade GPUs or advanced engineering workstations, allowing individual teams to utilize their tailored assistants in disconnected or edge-computing environments.

Category 2: System Incubation and Governance

Once a secure environment is established, strict protocols must govern how SMEs interact with and build the AI. The following constraints dictate the rules for collaborative development, parameter reproducibility, and the continuous lifecycle management of the foundational training data.

Constraint 4: Rapid, Collaborative Assistant Creation

To be operationally viable, the system cannot require a dedicated team of machine learning engineers to spend months developing each new domain-specific assistant. Instead, a primary constraint is the necessity for rapid, user-driven assistant creation. The system must function as an automated incubator that empowers teams of SMEs to collaborate concurrently on a single assistant. Users must be able to collectively upload unstructured source data, after which the framework must autonomously orchestrate the underlying data pipelines, including document chunking, embedding, vector database population, and LoRA initialization. This automated orchestration generates a functional, specialized assistant on the fly. Ultimately, this constraint ensures that the platform remains scalable, collaborative, and accessible to non-machine-learning professionals across various technical discipline domains.

Constraint 5: Configuration Control and Reproducibility

In engineering environments, trust in an AI assistant must include reproducibility of the assistant itself, not solely traceability of an individual answer. A deployed assistant may depend on an evolving combination of foundation model versions, LoRA adapters, embedding models, retrieval indexes, prompt templates, and curated training data. Consequently, the system is constrained by the requirement for strict configuration control across the full assistant lifecycle. This requirement is consistent with established configuration-management and documentation expectations in secure system engineering and AI risk management [10].

This means that every assistant instance should be reproducible from a fully defined configuration state, including the base model version, adapter version, embedding model, vector database snapshot, seed-data revision, verified training-set revision, and system-level prompt or behavioral template used during incubation and deployment. Without such configuration traceability, even correctly cited answers become difficult to audit, compare, or certify over time. Therefore, the architecture should support versioned assistant builds, controlled promotion of updates, and rollback to prior validated states [10].

Constraint 6: Knowledge Freshness and Source Lifecycle Management

Institutional knowledge is not static. Engineering procedures, software interfaces, validated assumptions, and operational guidance evolve continuously as projects mature. As a result, an enterprise AI assistant is constrained not only by the quality of its initial data, but also by the need to distinguish current, authoritative knowledge from outdated or superseded information. AI risk guidance emphasizes provenance, documentation, monitoring, and ongoing management of AI-system risks across the lifecycle [10, 15].

Consequently, the system should support source lifecycle management throughout the retrieval and training pipeline. The architecture should account for document aging, revision history, supersession, and conflicts among sources that may reflect different points in time. This includes mechanisms for reingesting updated data, flagging deprecated content, and prioritizing authoritative sources when multiple

documents address the same topic. The assistant should also expose temporal context when appropriate so that users can determine whether cited guidance reflects current practice or archived reference material [10, 15].

Category 3: Output Integrity and Operational Trust

For resulting AI assistants to be operationally viable in high-consequence engineering workflows, human operators must be able to securely trust, audit, and control their outputs. The following constraints establish the mandates for traceability, the handling of model uncertainty, rigorous deployment qualification, and the strict boundaries placed on agentic tool use.

Constraint 7: Verifiability, Traceability, and Alignment

While LLMs are capable of generating synthetic data and responses, they are inherently susceptible to hallucinations. Furthermore, recursively training an AI model on unchecked, AI-generated synthetic data leads to rapid model degradation, a phenomenon often referred to as model collapse [16]. Given the zero-tolerance for critical errors in advanced engineering environments, any viable AI platform is strictly constrained by the need for verifiable accuracy during its training phase [10]. This necessitates a robust HITL architecture. The system must enforce a slow-burn verification process where synthetic data and assistant responses are reviewed and validated by human experts before being incorporated into the formal training pipeline. This constraint not only ensures the integrity of the knowledge base but directly establishes the foundation for the system’s SFT methodology [9].

Beyond the training phase, a fundamental barrier to AI adoption in engineering environments is the black-box nature of LLM generation during inference. Operators cannot blindly trust unverified outputs, nor can they safely act on technical advice without knowing its origin. Therefore, the system is strictly constrained by the requirement for human-verifiable content and traceability. It is not sufficient for the assistant to merely provide a correct answer; it must explicitly demonstrate how that answer was derived. The architecture must ensure that every response generated via the RAG pipeline includes clear, accurate citations and references back to the original proprietary seed data. This constraint guarantees that users can easily audit the assistant’s logic, cross-reference the output against authorized documentation, and confidently apply the extracted knowledge without relying solely on the model’s parametric memory.

Constraint 8: Uncertainty Handling and Human Escalation

In specialized technical environments, a trustworthy assistant must do more than produce fluent responses; it must also recognize when the available evidence is insufficient, conflicting, or outside the scope of its validated knowledge. Therefore, the system is constrained by the requirement for explicit uncertainty handling and appropriate escalation behavior. This is consistent with recent emphasis on valid and reliable AI, human oversight, and management of confabulation risks, and it is

also supported by the selective-classification literature, which shows that abstaining under uncertainty can reduce errors in high-stakes settings [10, 15, 17].

To satisfy this constraint, the architecture should support bounded response behavior when confidence is inadequate. This may include abstaining from unsupported conclusions, explicitly identifying missing evidence, surfacing source conflicts, or prompting the user to consult a human SME. Rather than treating non-response as failure, the framework should treat calibrated uncertainty and escalation as core features of safe deployment [10, 15, 17].

Constraint 9: Operational Readiness and Deployment Qualification

An assistant that is technically functional is not necessarily ready for operational use in an engineering environment. Because the consequences of incorrect or poorly grounded outputs may be significant, deployment should not rely solely on qualitative impressions or anecdotal user success. Instead, the system is constrained by the requirement for formal operational readiness criteria that determines whether a specialized assistant is sufficiently accurate, aligned, and trustworthy for release. This is consistent with recent work on AI evaluation practices and with the measurement and management functions of the AI [10, 18].

This requires the architecture to support measurable evaluation before an assistant is promoted from incubation to operational deployment. Relevant criteria may include retrieval accuracy, citation correctness, hallucination rate, task-completion success on representative benchmark problems, human preference win-rates, and SME approval scores. The precise thresholds may vary across domains, but the underlying requirement remains the same: assistants should satisfy explicit acceptance standards before they are distributed to end users [10, 18].

Constraint 10: Bounded Tool Use and Action Control

As AI systems evolve beyond conversational retrieval toward tool use, code execution, and workflow orchestration, the risks associated with assistant autonomy increase substantially. In secure engineering environments, an assistant cannot be permitted to execute arbitrary actions simply because it can generate plausible instructions or code. Consequently, the system is constrained by the requirement that all tool use and action-taking occur within tightly bounded, policy-controlled execution environments. Recent work on AI agents highlights that reliable interaction with external systems and internal data are central to safe agent deployment, while established security controls emphasize least privilege, separation of duties, and controlled execution [10, 15].

This constraint applies to any capability involving code execution, workflow triggering, data transformation, or access to external services. The architecture must ensure that assistants operate strictly within approved action spaces, utilizing sandboxing, permission controls, logging, and, where necessary, explicit human authorization prior to executing consequential actions. Even internally developed tools should be treated as controlled interfaces rather than unrestricted capabilities [10, 15].

3 Incubation Framework

The primary goal of the HELENA platform is to provide an intuitive mechanism for training specialized AI assistants for onboarding and knowledge retention, all without requiring the user to possess prior machine-learning expertise. At its core, HELENA provides two fundamental services: the structured organization of domain knowledge, and the orchestration of intelligent search and synthesis over that knowledge. Both of these services are iteratively improved over time within the HELENA environment until they are finally packaged together as a deployable product.

The following sections detail the end-to-end architecture of this incubation pipeline. It begins by defining the target output (the tailored assistant), followed by the classifications of the ingested data (source, seed, and synthetic data) and the specific roles of the actors involved (teacher and human verifier). Finally, the core operational loops that drive the platform are detailed. This includes the verification cycle, training process, retrieval-augmented generation mechanics, and ultimate distribution of the content.

3.1 Tailored Assistant

Before detailing the data and training pipelines, it is necessary to define the product of the HELENA incubation framework: the tailored assistant. An assistant is essentially a specialized snapshot of knowledge packaged alongside structural code. This structural code provides the necessary inference orchestration for the assistant, ensuring it remains portable across different hardware environments. The underlying implementation and specific domain knowledge may vary but the core operational functionality remains straightforward: a user submits a complex query, and the assistant searches for supporting information before synthesizing a domain-informed response.

As illustrated in Fig. 1, a tailored assistant derives its knowledge from an interconnected composite architecture. At its core is a foundational base model, providing general capabilities. This is extended by a fine-tuned LoRA module. Like an LLM, this LoRA is a trained matrix of weights; because it is explicitly fine-tuned on domain data, it contains its own knowledge, and because it is separate from the base model, it builds on the underlying implicit knowledge of the base model.

Finally, the assistant’s non-parametric knowledge is stored within a vector database designed for RAG. This collection verified data are dynamically queried to ground the assistant’s responses and provide explicit source citations relevant to a user’s query.

The subsequent sections will describe the breakdown of data used in HELENA, the process of generating synthetic data using a Teacher LLM, the HITL verification system shown in Fig. 2, and the process for how the verified data are ultimately used to create a tailored assistant as illustrated in Fig. 1.

3.2 Source Data

As with any AI endeavor, establishing a robust, high-quality dataset is the critical first step in developing a capable solution. In the HELENA framework, the initial

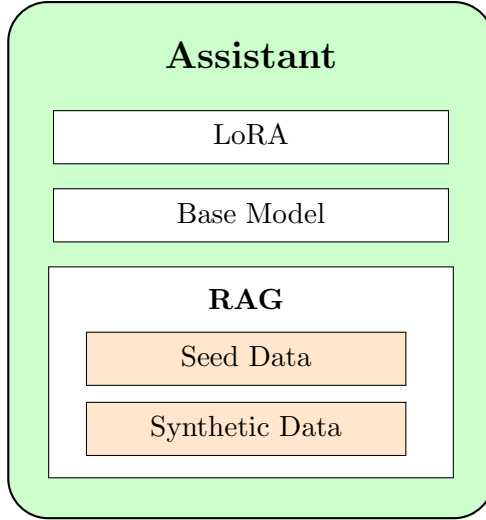


Figure 1: Component view of the AI Assistant.

collection of information is referred to as source data. Source data encompass all raw, unmodified materials generated by processes external to the HELENA platform. The framework is explicitly designed to refine and expand its intelligence over time through HITL verification and ongoing user interactions. Providing a moderately-sized initial dataset of referential materials is instrumental in rapidly bootstrapping a functional assistant.

To facilitate this initialization phase, HELENA is engineered to ingest a wide variety of standard engineering and documentation formats. The platform currently supports the direct upload of portable document format (PDF) files (such as reference manuals, research papers, and frequently asked questions), raw source code and unit tests, presentation slide decks, and custom conversational data logs. In scenarios where a formal repository of documentation does not yet exist for a specific domain, the HELENA interface provides a fallback data-capture mechanism. Users can record and export initial project conversations directly within the system to organically begin constructing the knowledge base from a zero-data starting point.

Once this heterogeneous collection of raw files is uploaded to the platform, it undergoes a series of transformations. These include parsing, filtering noise, augmenting information, composing new information, and chunking into semantically similar blocks. These critical data engineering steps convert the source data into one of the system’s normalized internal formats, formally establishing the initial batch of seed data.

3.3 Seed Data

Seed data represent the collection of fundamental, logically-segmented units of information, such as extracted paragraphs or tabular data, used within HELENA. These normalized data are derived from two primary pipelines: the data engineering of raw source data and the curation of verified synthetic data. As described in

engineering emails, or enterprise messaging platforms, creators can synthesize conversational seed data that accurately reflect localized knowledge and terminology. This process allows teams to explicitly teach the assistant the specific “voice” and behavioral tone it should adopt. Furthermore, chaining context-dependent pairs together tightly couples semantically related information, providing the assistant with robust training samples for managing long-form, multi-step technical inquiries.

3.4 Synthetic Data

Within the HELENA incubation framework, the rapid synthesis of high-quality training material is a critical component in developing assistants capable of answering complex user inquiries. To augment the foundational seed data, the HELENA framework employs a generative pipeline that presents existing knowledge in novel ways, implicitly and explicitly linking technical concepts to generate novel ideas. These synthetic data are created through an automated cycle of sampling, retrieval, and generation using a teacher LLM.

3.4.1 Teacher LLM

While the specialized domain assistants (the “student” models) are optimized for localized, computationally-efficient inference, the teacher LLM is a larger-parameter model deployed specifically to handle complex data collation and synthesis at scale.

The primary function of the teacher LLM is to analyze a seed datum and autonomously generate high-quality synthetic instruction-response pairs. When evaluating a specific chunk of foundational text—such as an engineering standard or a software manual—the teacher LLM is prompted to predict the types of technical questions a user would ask regarding that content. Afterward, the teacher LLM generates its best attempt at the corresponding answers based on the provided data and its parametric knowledge. This automated synthesis using a teacher LLM allows the HELENA platform to rapidly expand sparse documentation into a robust dataset suitable for training the student model’s LoRA.

3.4.2 The Generative Pipeline

The pipeline executes the following sequential steps to produce the unverified synthetic data:

1. *Sample*: A discrete seed datum (e.g., a specific paragraph or code snippet) is sampled from the established seed data.
2. *Retrieve Context*: Supporting information that is semantically relevant to the sampled seed datum is retrieved via the vector database and concatenated to provide a broader context window. Additionally, any synthetic data that have previously been verified are retrieved to provide guiding examples (few-shot prompting) and ensure the teacher LLM does not recreate any previously verified synthetic data.

3. *Generate*: The teacher LLM synthesizes instruction-response pairs—such as anticipated user questions and detailed answers—based strictly on the provided context and its own parametric knowledge.

Because an LLM autonomously executes this generative step, it inherently introduces the risk of hallucination. To ensure the knowledge base is not polluted with factually incorrect or ungrounded information, a HITL is strictly required. HELENA provides an intuitive interface for users to evaluate, correct, and verify this synthetically generated data prior to its inclusion in the training set (detailed in the subsequent verification section).

Incorporating verified synthetic data into the training pipeline provides several distinct training advantages:

- *Linguistic Diversity*: Synthetic generation expands the verbosity and phrasing of the dataset, enabling the student model to recognize and understand technical concepts when queried in novel or unexpected ways [19].
- *Conceptual Distillation*: Given the potential size and complexity of raw engineering documents, synthetic generation automatically breaks extensive texts into smaller, targeted question-and-answer pairs. This distillation grounds the model in foundational concepts and improves overall accuracy [20].
- *Scalability*: High-quality training pairs can be generated rapidly and in massive quantities, overcoming the limitations of relying solely on scarce human-generated data.
- *Adaptability*: The synthetic generation pipeline and teacher LLM can be continuously updated to address emerging technologies and the evolving needs of the engineering community.

Ultimately, while seed data provide the definitive factual baseline, verified synthetic data expand the conceptual surface area of the training set. This methodology allows the HELENA platform to efficiently enhance the quality and responsiveness of a tailored assistant while minimizing the manual data-entry burden placed on human domain experts.

3.5 Verification Cycle

Verification is the foundational quality-assurance mechanism within the HELENA architecture. It serves as the critical filter ensuring the accuracy, relevance, and safety of the data ultimately used to train the domain assistants. Fundamentally, verification is the procedural review, validation, and correction of data to ensure that the ground truth of the engineering discipline is accurately reflected. This collaborative, slow-burn process relies on a team of SMEs and developers who iteratively audit the platform’s synthetic outputs, gradually accruing an accurate and comprehensive operational knowledge base.

3.5.1 Human Verifier

The human verifier represents the indispensable HITL component of the HELENA data flow, as previously illustrated in Fig. 2. Given the high-consequence nature of engineering environments, the verifier acts as the ultimate arbiter of truth. Verifiers are explicitly responsible for assessing the factual accuracy, technical formatting, and contextual relevance of the synthetically generated data.

The primary objective of this role is to exert strict human oversight of the underlying knowledge base without requiring manual data entry for every token. By rigorously reviewing and correcting a targeted subset of the generated question-and-answer pairs, the human verifier guides the assistant’s subsequent training. This curated, high-signal data dictate the precise behavioral alignment of the LoRA adapter and enforces the factual integrity of the non-parametric memory storage used in RAG.

3.5.2 Process

In the context of the HELENA framework, the verification cycle is the operationalization of the HITL mandate. By actively reviewing the synthetically generated data against its parent seed datum, users directly control the quality of the model’s training pipeline.

The automated orchestration of this cycle involves the following five sequential steps:

1. *Sample*: The system retrieves a parent seed datum and its associated synthetically generated question-and-answer pairs from the database and routes them to a user.
2. *Review*: The user evaluates the synthetic data, considering its factual accuracy and contextual relevance to the parent seed datum.
3. *Verify*: The user officially adjudicates the synthetic data by accepting it, rejecting it as incorrect, or directly modifying the text to resolve inconsistencies or inject missing context.
4. *Update*: The system commits the verified data to the curated database, propagating any user modifications to the associated seed or synthetic records.
5. *Evaluate*: The system checks if the parent seed datum has met its designated verification threshold. If the threshold is unmet, the system autonomously generates new synthetic pairs for future review. If the verification threshold is met, the system marks the seed datum as fully explained and stores it in long-term memory until the source data indicate it is no longer correct.

While the platform automates the sampling, updating, and evaluating phases in the background, human verifiers dedicate their cognitive effort exclusively to the review and verification steps. During the review phase, the SME must evaluate the synthetic datum as an independent training artifact. First, the synthetic pair must be entirely self-contained; it should not contain dangling pronouns or references to

the parent seed datum, ensuring it is fully answerable based solely on the provided prompt. Second, the synthetic datum must be conceptually meaningful, capturing a complex fact or procedure rather than stating an easily inferred triviality. Finally, the synthetic text must mimic realistic user inquiries and exhibit the desired behavioral tone of the assistant, avoiding overly terse, excessively verbose, or irrelevant phrasing.

During the verification phase, the verifier makes a final operational judgment on the viability of the data. Because computational synthetic data generation is exceptionally cheap and human engineering time is expensive, users are explicitly encouraged to be conservative in their approvals. Rather than spending significant time heavily editing poor synthetic outputs, users are instructed to reject them, forcing the system to generate new samples. However, in specific scenarios—such as custom conversational data types or software source code linked via version control systems (e.g., Git or Subversion)—HELENA possesses the bidirectional capability to push user corrections made during verification directly back to the original source material. This ensures that the verification process not only trains the AI assistant but also simultaneously improves the quality of the organization’s underlying engineering documentation.

3.5.3 Metrics

Manual data verification is an inherently time-intensive process. Without clear visibility into systemic progress, the repetitive nature of the task can induce user fatigue and discourage consistent SME participation. To mitigate this friction and optimize resource allocation, the HELENA platform employs automated metrics that quantify exactly which knowledge areas have been sufficiently verified and which require further human attention.

To achieve this, the system uses a term frequency-inverse document frequency (TF-IDF) methodology [21,22] to assign a dynamic complexity metric (scaled from 1 to 10) to every seed datum upon upload. For example, a complex procedural text detailing how to configure a computational fluid dynamics (CFD) environment may receive a complexity score of 5. For natural language data, this metric is calculated using a two-step algorithmic process inspired by term TF-IDF methodologies:

1. *Extract Entities*: A foundational LLM analyzes the seed datum to extract key conceptual entities and map the relational edges between them.
2. *Evaluate Complexity*: The system applies min-max normalization, utilizing the volume and density of the extracted relational edges as the primary weighting term to calculate the final complexity score.

This complexity metric serves as a strict verification threshold for the system. It mathematically dictates the specific quota of synthetic question-and-answer pairs that must be human-verified before the parent seed datum is considered fully saturated or “explained”. Each time a user verifies a synthetic pair, a counter is incremented against this threshold, providing administrative clarity on the platform’s progress toward establishing a comprehensive, fully-verified knowledge base.

3.6 Challenges

Despite the robust technical architecture of the HELENA framework, the primary operational challenge in training specialized domain assistants remains the human knowledge acquisition bottleneck. Because the slow-burn verification process relies heavily on highly-compensated SMEs, the continuous curation of synthetic data inherently competes with their primary engineering responsibilities. Empirical studies in AI workflows demonstrate that data verification and curation are frequently undervalued by organizational incentive structures, leading to annotator fatigue and a reluctance among experts to participate in continuous data work [23, 24]. Consequently, the long-term viability of the incubation framework depends not only on algorithmic efficiency but also on minimizing the user-interface friction for verifiers and establishing formal administrative incentives that recognize knowledge capture as a critical, first-class engineering activity [25].

Beyond motivational barriers, the scalability of the HITL architecture presents an operational challenge. The continuous, automated generation of synthetic data by the teacher LLM may outpace available human bandwidth. SMEs risk becoming overwhelmed by the volume of required verifications, transforming the quality-assurance pipeline into a severe throughput bottleneck. While the current slow-burn methodology is effective for initial assistant incubation, mitigating this scalability limit necessitates the integration of automated pre-filtering mechanisms, such as LLM-as-a-Judge, detailed in Sec. 6.3.2.

3.7 Training Pipeline

Beyond knowledge capture and verification, the core operational function of the HELENA incubator is the cyclical training of the domain assistants. The framework employs a recurring, batch-oriented training pipeline to continuously adapt the models. Operating on a scheduled cadence, the system aggregates newly uploaded seed data alongside the curated synthetic data accrued during the human verification cycle. To strictly adhere to the localized compute limitations of Constraint 3 and the security mandates of Constraint 1, the platform avoids full-parameter fine-tuning. Instead, this aggregated dataset is used to optimize a LoRA module [8]. This PEFT approach ensures rapid, memory-efficient adaptation to domain-specific tasks while preserving the generalized representational capacity of the frozen base model.

3.7.1 Algorithmic Optimization and Implementation

To execute this training locally, the underlying optimization pipeline is implemented as a custom training loop designed to provide fine-grained control over memory allocation, numerical stability, and hardware utilization. The architecture integrates several critical mathematical and programmatic techniques:

- *Dynamic Sequence Batching*: Tokenized training samples are dynamically padded to the maximum sequence length within a specific batch rather than a global maximum, dramatically minimizing wasted computational cycles.

- *Gradient Accumulation and Precision Scaling:* To simulate the larger batch sizes required for stable training without exceeding local GPU memory limits, the algorithm accrues gradients across multiple forward passes. These passes utilize automatic mixed precision (AMP) [26] and 8-bit quantization [27] to reduce the memory footprint while maintaining numerical stability via dynamic loss scaling.
- *Algorithmic Stability Mechanisms:* To prevent gradient explosion and early training instability, the loop enforces strict gradient clipping and employs a linear learning rate warmup phase prior to achieving constant-rate optimization via the AdamW optimizer [28].
- *Stateful Check-pointing and Bounded Execution:* The training process is constrained by a maximum runtime to ensure compatibility with shared or scheduled compute environments. Periodic check-pointing captures both model weights and optimizer states, enabling restarts in the case of failure. Furthermore, loss is closely monitored to evaluate convergence for early stopping.

3.7.2 Data Engineering and Governance

Data engineering and governance are central to HELENA’s design, ensuring that the resulting assistants are both capable and securely deployable. The data pipeline balances three competing operational priorities: data quality, iterative algorithmic improvement, and strict data isolation.

To maintain data quality, the system relies on a bifurcated trust model. Foundational source data (e.g., reference manuals and code documentation) are ingested into the verification process through data engineering as seed data, as those are manually uploaded by cleared SMEs. Conversely, all synthetic data must pass through the strict HITL verification cycle prior to entering the training pipeline.

To ensure continuous iterative improvement without losing systemic control, the training cycle implicitly versions the data. Each training run records the exact metadata, composition, and version identifiers of the datasets utilized. This rigorous provenance tracking satisfies the configuration control and reproducibility mandates of Constraint 5, allowing system administrators to recreate specific training conditions, compare generational LoRA performance, and facilitate rollbacks if an assistant experiences behavioral degradation.

Finally, from a security perspective, HELENA establishes a strict architectural boundary between centralized incubation and decentralized inference. Training is performed on controlled infrastructure using only pre-cleared institutional knowledge. The resulting LoRA adapters contain only mathematical weights, rather than raw text. When these lightweight adapters are distributed to end-users for local inference, users can query the assistant with sensitive or export-controlled project data without the risk of their prompts leaking back to a centralized server. This localized execution ensures strict adherence to the data sovereignty mandates of Constraint 1.

3.8 Low-Rank Adaptation (LoRA)

As previously illustrated in Figure 1, the architecture of a tailored assistant relies on a LoRA [8] module to dictate its domain-specific behavior. The LoRA serves as the repository for the assistant’s implicit knowledge and communication style. This module is trained on the high-signal, curated datasets produced during the HITL verification cycle. By utilizing the verified synthetic data as its training corpus, the LoRA mathematically encodes the technical layout and formatting preferences of the SMEs, and can be formatted to also include strict structure such as input file formats.

To achieve this behavioral alignment without violating the compute limitations outlined in Constraint 3, the HELENA framework strictly avoids full-parameter fine-tuning. Instead, LoRA freezes the pre-trained weights $W_0 \in \mathbb{R}^{d \times k}$ of the base model, where d and k represent the original input and output dimensions of the specific transformer layer being modified, and then injects trainable rank decomposition matrices into the transformer architecture. The necessary weight update ΔW is constrained by approximating it as the product of two significantly smaller matrices, $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$, where the rank r represents the “rank” (the hyperparameter chosen to control the dimensionality of the A and B matrices) and is much smaller than the model dimensions ($r \ll \min(d, k)$).

For a given input vector x , the modified forward pass is expressed as:

$$h = W_0x + \Delta Wx = W_0x + BAx. \quad (1)$$

During inference, the outputs of the frozen base weights (W_0x) and the newly trained low-rank matrices (BAx) are simply summed together.

This parameter-efficient architectural choice provides three operational advantages for the HELENA platform:

- *Computational Efficiency:* By reducing the number of trainable parameters by several orders of magnitude, LoRA enables the incubation of domain assistants entirely on enterprise hardware, significantly reducing hardware overhead and training times.
- *Modular Portability:* Because the original base model remains frozen and unmodified, the resulting LoRA adapter is a significantly smaller file compared to the base model. This allows the specialized engineering structure and format to be easily exported, version-controlled, and distributed to end-users without duplicating the underlying foundation model.
- *Dynamic Swapping:* The separation of the base intelligence from the adapted behavior allows the system to seamlessly load or unload different LoRA matrices into the base model’s active memory, enabling dynamic, multi-disciplinary routing.

Ultimately, the LoRA adapter bridges the gap between a generic foundation model and a specialized engineering colleague. However, while the LoRA governs the assistant’s behavioral alignment and implicit reasoning capabilities, it is not

designed to memorize large quantities of recallable facts, such as those embedded in the corpus of verified synthetic data. To ground the assistant’s outputs in verifiable truth, the LoRA must operate in tandem with the RAG pipeline detailed in the following section.

3.9 Retrieval and Prompt Processing Pipelines

The HELENA framework relies on sophisticated vector retrieval and dynamic prompt construction to drive model inference. This architecture serves as the primary operational engine for the tailored assistant during user interactions, while a parallel implementation of this retrieval logic is leveraged by the teacher LLM during the incubation phase to generate synthetic data. The following subsections detail how these prompt processing and retrieval pipelines are orchestrated across both use cases.

3.9.1 Tailored Assistant Inference (RAG Pipeline)

During deployment, explicit factual knowledge is supplied to the tailored assistant via a multi-stage RAG pipeline. This workflow, illustrated in Fig. 3, combines vector similarity searching with post-retrieval filtering and dynamic query disambiguation to maximize contextual relevance.

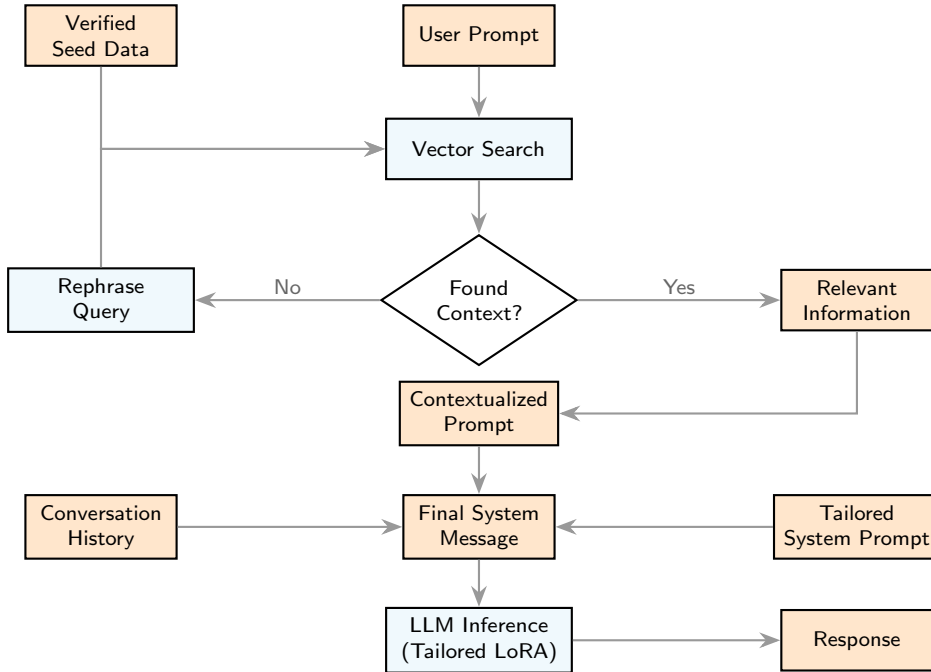


Figure 3: The multi-stage RAG inference and query disambiguation workflow for the tailored assistant.

The retrieval process is initiated using the user’s most recent message. The query is embedded and passed to the vector database, executing a nearest-neighbor search

to retrieve candidate documents. To ensure precision, these candidates are subjected to distance-based filtering. Any record exceeding a predefined maximum semantic distance is discarded, preventing weak matches from introducing irrelevant context into the model’s active memory.

In conversational environments, user queries frequently utilize ambiguous pronouns (e.g., *"How do I configure that module?"*). If the initial retrieval yields zero relevant records, the system autonomously intercepts the workflow. A secondary language model analyzes the conversation history to resolve co-references and clarify the user’s intent. This disambiguated query is submitted for a secondary vector search, significantly improving recall in multi-turn interactions. Once contextually valid records are established, metadata extraction occurs for every vetted document; metadata includes such information as to the lineage of the record. The retrieved text and positional identifiers are concatenated into a structured final system message, satisfying the verifiability mandates of Constraint 7 by allowing the interface to render precise source citations. The final system message is then passed to the LLM inference, which includes the tailored LoRA, explained in Sec. 3.8, which then generates a response.

Finally, to adhere to the uncertainty handling protocols outlined in Constraint 8, the system enforces a strict fallback behavior. If no relevant records pass the filters, the pipeline injects a fallback instruction, forcing the assistant to explicitly acknowledge that no supporting documentation was found, thereby preventing unsupported hallucinations.

3.9.2 Prompt Orchestration

To ensure the teacher produces accurate synthetic data during the incubation phase, the platform adapts the aforementioned retrieval architecture to dynamically construct generative prompts, as illustrated in Fig. 4.

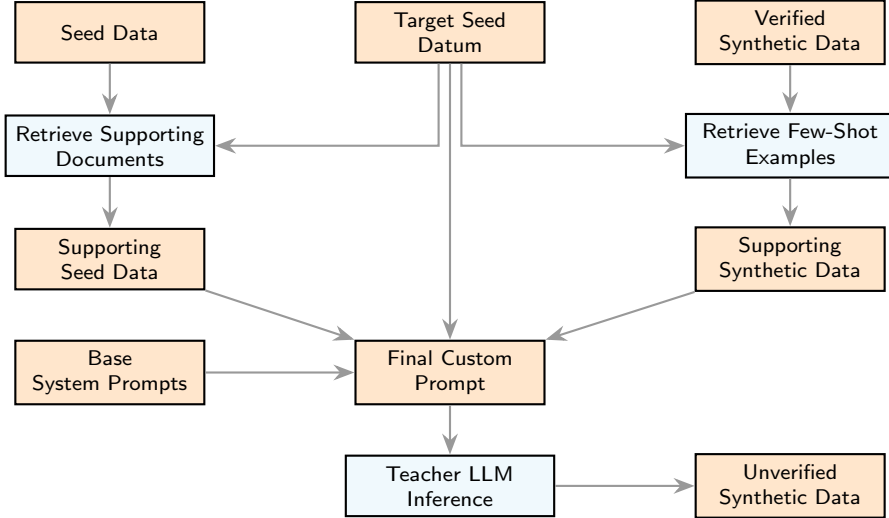


Figure 4: Prompt orchestration workflow for generating synthetic training data via the teacher LLM.

When a specific target seed datum is selected for expansion, the system queries the primary vector database to retrieve supporting seed data, providing the teacher LLM with adjacent engineering context. Simultaneously, the system queries the repository of previously verified synthetic data to retrieve semantically similar few-shot examples. These examples mathematically demonstrate the exact formatting, tone, and complexity expected by the human verifiers. By concatenating the target datum, the supporting context, the few-shot examples, and the base system instructions into a single unified prompt, the teacher model is constrained. The resulting output is high-fidelity, unverified synthetic data that are conceptually rich and structurally prepared for the HITL verification cycle.

3.10 Distributed Content and Portability

The ultimate objective of the HELENA incubation framework is the seamless deployment of specialized knowledge to end-users. Content distributed to these external users, whether they are internal engineers or third-party aerospace analysts, is packaged as a standalone software deployment alongside the fully trained Tailored Assistant.

To satisfy the strict hardware portability mandates of Constraint 3, the distributed software package relies heavily on containerization. By encapsulating the foundational base model, the specialized LoRA weights, the vector database, and the required inference orchestration code into a lightweight Docker image, the assistant achieves near-universal compatibility. This containerized approach ensures the assistant can be executed reliably across a wide array of localized hardware environments, from standard corporate laptops to advanced, air-gapped engineering workstations.

To maximize accessibility for non-machine-learning professionals, the distributed package includes a localized, browser-based graphical user interface (GUI). This simple, intuitive chat interface seamlessly integrates with any exported HELENA assistant, abstracting the complex underlying RAG and inference pipelines away from the end-user.

Crucially, the distribution of a tailored assistant must be governed by the exact same strict security protocols that apply to traditional software releases. It is a fundamental operational reality that all information embedded within the assistant—both the implicit knowledge baked into the LoRA weights and the explicit non-parametric knowledge stored in the RAG database—is fully accessible to anyone possessing the distributed package. Therefore, to uphold the data sovereignty and access authorization boundaries established in Constraint 1 and Constraint 2, the seed data utilized during the incubation phase must be rigorously vetted. System administrators must ensure that an assistant is trained exclusively on information already cleared and approved for the specific clearance level of its intended user base prior to packaging and deployment.

4 Deployment and Security

To satisfy the strict data sovereignty mandates outlined in Constraint 1, the HELENA incubator is designed to operate entirely independent of third-party cloud service

providers. Unlike conventional AI development platforms that require persistent internet connectivity or rely on proprietary APIs from external vendors such as Microsoft, Google, or OpenAI, HELENA is deployed directly within the host organization’s restricted network infrastructure. To achieve this, the platform’s backend services, vector databases, and machine-learning pipelines are packaged into modular containers. This containerized approach ensures the incubator can be seamlessly transferred across network air gaps and securely hosted behind security firewalls, guaranteeing zero external data leakage during the training and inference phases.

A fundamental advantage of this localized deployment strategy is the elimination of cloud computing dependencies and their associated costs. The platform is explicitly optimized to operate on low-cost, commercially available hardware, such as standard engineering workstations equipped with GPUs capable of handling the size of the teacher LLM. To achieve high-fidelity AI performance on this localized hardware without relying on external LLMs, the system utilizes publicly available open-weights models, such as the LLaMA architecture [14]. By establishing these public models as the foundational base and hosting them on a local inference server, all proprietary document embedding, RAG orchestration, and LoRA fine-tuning execute strictly on the organization’s own localized hardware.

Finally, deploying the incubation system entirely on-premises allows HELENA to integrate securely with the organization’s existing identity management protocols. Rather than trusting an external vendor with access control, the platform can tie directly into the institution’s internal active directory. This enables strict role-based access control (RBAC) [29], ensuring that only authorized SMEs are granted the privileges necessary to upload restricted seed data, participate as verifiers in the slow-burn SFT pipeline, or export completed assistants. By merging physical infrastructure isolation, low-cost hardware optimization, and public open-weights models, HELENA provides a secure, turnkey environment for institutional knowledge capture.

5 Preliminary Results and Case Studies

To validate the HELENA framework, the platform is currently incubating four tailored assistants. While the deliberate pacing of the slow-burn human verification cycle means these models remain in the early stages of their development, preliminary evaluations confirm that this approach successfully yields progressively more knowledgeable and accurate domain experts. The following sections detail the status of the tailored assistants currently being incubated at the time of this publication, demonstrating the platform’s proven ability to transform raw source data into aligned, highly capable tools over time.

5.1 ALANA: ANOPP2 LLM Assistant for Noise Applications

The first assistant that is currently being incubated within the HELENA framework is the ANOPP2 LLM Assistant for Noise Applications (ALANA). Designed to support users of the Second-Generation Aircraft NOise Prediction Program (ANOPP2) [30], ALANA serves as a specialized domain expert for acoustic analysis and software

implementation. ANOPP2 is a comprehensive, modular framework developed to predict and analyze noise for both fixed-wing and rotary-wing aircraft. The software is versatile, providing mixed-fidelity noise predictions for multiple noise sources by seamlessly integrating source noise prediction with atmospheric propagation. Furthermore, ANOPP2 facilitates fundamental noise prediction based on a mixed-fidelity framework spanning the analytical spectrum: from high-fidelity CFD output to low-fidelity design tools, and from physics-based theoretical acoustic formulations (such as Farassat’s Formulation 1A (F1A) [31]) to empirical-based noise models. Because this software is heavily utilized for a wide variety of aerospace applications, including aircraft design, environmental impact assessments, and regulatory noise certification, navigating its complex architecture of core frameworks, functional modules, and specialized tools requires significant technical expertise.

To lower this barrier to entry, the foundational source data for ALANA was compiled from a diverse array of unstructured and semi-structured sources. These sources include formal ANOPP2 documentation, such as published technical documents [32] alongside historical conversational data derived from previous user inquiries. Furthermore, SMEs proactively generated custom content, documenting specific edge cases and complex workflows they knew would be valuable to end users but were absent from traditional manuals.

To tailor the behavior of ALANA and ensure the highest degree of technical accuracy, the assistant is currently undergoing the platform’s slow-burn verification process. A panel of ANOPP2 SMEs periodically reviews and grades the synthetic responses generated by the assistant. During these review cycles, the SMEs correct formatting, inject missing contextual nuances regarding specific ANOPP2 modules, and validate the technical logic of the outputs. This iterative, HITL review continuously generates verified training sets, which are subsequently used to train the specific LoRA adapter of ALANA. This methodical approach gradually instills complex operational knowledge into the assistant, effectively transitioning undocumented, tacit engineering expertise into a reliable, accessible conversational interface.

5.2 ARIANA: ANOPP2 Research & Integration Assistant for Noise Applications

The second assistant currently incubating within the HELENA framework is the ANOPP2 Research & Integration Assistant for Noise Applications (ARIANA). In contrast to ALANA, which is designed to aid end users, ARIANA is strictly an internal tool developed exclusively for the core ANOPP2 engineering team. Its primary objective is to support developers in researching new acoustic methodologies, integrating functional modules, and maintaining the software’s extensive underlying architecture. Because ARIANA handles sensitive, proprietary development environments and raw source code, it is not distributed to the general user base. This strict isolation directly leverages the secure, on-premises deployment capabilities and data sovereignty mandates defined in Constraint 1.

To function as an expert-level software co-developer, ARIANA requires a fundamentally different knowledge base than a user-facing assistant. The source data driving ARIANA consist of the complete ANOPP2 source code, as well as the

extensive internal codebases utilized for testing, continuous integration, and preparing software distributions. Furthermore, the repository is enriched with technical SME write-ups and historical responses to queries posed by other developers. This combination of raw codebase ingestion and SME-to-SME communication allows the assistant to capture underlying design philosophies and undocumented architectural decisions that govern the ANOPP2 framework.

The slow-burn verification process for ARIANA focuses on enforcing coding standards and accurate architectural synthesis. Multiple developer SMEs are working concurrently within the HELENA incubator to evaluate and refine the output of the assistants. These developers grade and correct the assistant’s ability to generate syntactically accurate code snippets, explain complex integration testing procedures, and troubleshoot software distribution pipelines. This continuous, collective review serves a dual purpose: it populates a curated RAG database with verified architectural solutions, and it simultaneously trains a LoRA adapter tailored to the communication style expected in advanced software engineering.

5.3 CAMILLA: CAMRAD II Aerodynamic Modeling Intelligent Learning and Linguistic Assistant

The third assistant currently being incubated within the platform is the CAMRAD II Aerodynamic Modeling Intelligent Learning and Linguistic Assistant (CAMILLA). CAMILLA is designed to support engineers utilizing Second-Generation Comprehensive Analytical Model of Rotorcraft Aerodynamics and Dynamics (CAMRAD II), an industry-standard software tool developed for the aeromechanical analysis of rotorcraft. Built upon foundational rotorcraft aeromechanics [33], the software calculates performance, structural loads, complex aerodynamics (such as blade-vortex interaction (BVI)), and aeroelastic stability for traditional helicopters, tiltrotors, and advanced air mobility (AAM) aircraft. Due to the deep theoretical physics and mathematics underlying the software, users face a steep learning curve, making it an ideal candidate for a specialized AI assistant.

Unlike ALANA and ARIANA, the development of CAMILLA is serving as a critical usability test for the HELENA framework. The assistant is being intentionally incubated by a group of rotorcraft SMEs who are entirely separate from the core ANOPP2 development team and possess no prior AI or machine learning experience. Testing the system’s accessibility, these SMEs independently utilize the HELENA interface to upload comprehensive CAMRAD II documentation as foundational source data. They further enrich this repository by generating conversational data, directly answering user queries, and documenting internal back-and-forth technical discussions regarding complex VTOL modeling scenarios.

Through the platform’s intuitive interface, this independent group of SMEs is successfully executing the slow-burn verification process. By grading and refining the assistant’s responses to aerodynamic and structural dynamic queries, the SMEs are simultaneously populating a curated RAG database and training a specialized LoRA adapter. The successful ongoing incubation of CAMILLA not only yields a capable rotorcraft analysis assistant, but it also validates a core operational objective of the HELENA framework: the ability to empower non-AI domain experts to

autonomously capture, verify, and deploy localized institutional knowledge.

5.4 HELENA: Human-Enabled Learning & Evaluation Network for Assistants

The final assistant currently being incubated within the platform is a recursive application of the framework itself: an assistant dedicated to documenting and supporting the HELENA system. As the platform is deployed to various isolated engineering departments, new users—acting as source data providers, slow-burn verifiers, and system administrators—require onboarding to understand the incubator’s architecture. To facilitate this without requiring continuous manual training from the core development team, the HELENA framework is being used to create a specialized meta-assistant. This assistant serves as an interactive, localized user manual and troubleshooting guide for the incubation platform itself.

The source data powering the HELENA meta-assistant encompass the entirety of the platform’s proprietary documentation. This repository includes the system’s underlying source code, deployment configurations, GUI navigation guides, and the foundational systems engineering documents (including the drafts of this very Technical Memorandum). By vectorizing this architecture-level data, the RAG pipeline allows a new user to ask specific questions, such as how to adjust LoRA hyperparameters in the administrative dashboard or how to format their initial source data uploads for optimal chunking.

The verification of this meta-assistant is being conducted directly by the core creators of the HELENA system. By grading the assistant’s responses to simulated user queries and deployment troubleshooting scenarios, the developers are training a LoRA adapter specifically tailored to provide clear, instructional, and accurate onboarding support. The successful deployment of this meta-assistant not only accelerates the adoption of the framework across new organizational departments but also serves as the proof-of-concept: the HELENA system is effective at capturing and preserving its own foundational engineering knowledge.

6 Future Capabilities

The current implementation of the HELENA framework establishes a secure, locally deployable foundation for mitigating institutional knowledge attrition through specialized AI assistants. While the integration of RAG, LoRA, and slow-burn human verification satisfies the strict operational constraints outlined in this memorandum, the rapid advancement of AI presents opportunities to significantly expand the platform’s utility. To fully realize this potential, future iterations of the system must evolve from passive, text-centric retrieval repositories into active, autonomous, and multimodal digital collaborators.

Rather than representing a strict, chronological development pipeline, the following subsections detail a parallel strategic road map for HELENA’s future capabilities. These enhancements are grouped into four core thematic areas of concurrent development: advanced knowledge representation and reasoning, autonomous agency and

collaboration, scalable alignment and verification methodologies, and enterprise-wide infrastructure scaling.

6.1 Advanced Knowledge Representation and Reasoning

The foundation of any specialized domain assistant is the depth, structure, and traceability of its underlying data. While the current RAG architecture successfully grounds the models in static text, institutional engineering knowledge is inherently multidimensional and continuously evolving. To maximize the utility of the HELENA framework, future iterations must expand beyond semantic text search to capture complex visual data, map the structural relationships between engineering concepts, and force the AI to transparently articulate its logical deductions. The following capabilities detail the transition toward active, multimodal knowledge synthesis.

6.1.1 Multimodal Knowledge Capture and Vision-Language Models

While the current HELENA architecture is capable of synthesizing text-based documentation, a significant portion of institutional engineering knowledge is inherently visual. Critical information is frequently embedded within system schematics, computer-aided design (CAD) diagrams, plotted data, and technical documents such as journal articles. Currently, text-only RAG pipelines fail to capture or contextualize this visual data, leaving gaps in the knowledge base. To address this, a future capability for the platform is the integration of Vision-Language Models (VLMs) to enable true multimodal knowledge capture [34].

By incorporating visual encoders into the architecture, future iterations of the platform will allow users to upload multimodal seed data. The framework will process both the text and visual elements, embedding them into a unified multimodal vector space. This advancement will allow an engineer to provide the assistant with an image representing complex technical data and ask for specific insights based on that visual context. Furthermore, to maintain compliance with the hardware portability mandates outlined in Constraint 3, this multi-modal integration will rely on optimized, open-weights VLMs, such as the Large Language-and-Vision Assistant (LLaVA) architecture, that utilize adapter layers to project visual features into the language model’s text space [34]. Crucially, processing this visual data entirely on-premises ensures that proprietary schematics and export-controlled diagrams remain secure and properly segmented among authorized users, satisfying both the data sovereignty requirements of Constraint 1 and the access authorization mandates of Constraint 2.

6.1.2 Knowledge Graph Integration and Gap Identification

Currently, the RAG pipeline relies on standard vector databases that retrieve information based on semantic similarity. While effective for localized queries, vector databases lack structural awareness of the broader dataset. To enhance data traceability and source life cycle management (as mandated by Constraints 6 and 7), a critical future capability is the integration of knowledge graphs [35].

By mapping institutional knowledge as a network of interconnected entities (nodes) and relationships (edges), a knowledge graph provides creators with a transparent, visual topology of exactly what information is embedded within the assistant. More importantly, this structural mapping allows the system to identify what is missing. By algorithmically scanning the graph for sparse nodes, disconnected clusters, or conceptual dead ends, the platform can highlight specific areas that lack sufficient depth. This capability shifts the burden of discovering undocumented knowledge from the creator to the AI platform, explicitly highlighting what information needs to be injected to improve the knowledge base. Furthermore, autonomously identifying these gaps and prompting targeted knowledge injection inherently reduces model uncertainty by providing a more robust, comprehensive dataset, directly addressing the uncertainty handling and human escalation requirements outlined in Constraint 8.

6.1.3 Active Knowledge Elicitation

Once the knowledge graph identifies a critical gap in the embedded data, the platform must possess a mechanism to capture that missing information efficiently. Currently, AI assistants operate as passive entities that wait for users to initiate queries. Future iterations of the platform will invert this paradigm through the implementation of active knowledge elicitation [36].

When a sparse node is detected in the knowledge graph, the assistant will autonomously reach out to designated SMEs or assistant creators. Operating as an active interviewer, the assistant will generate targeted, context-aware questions designed to extract the undocumented details required to flesh out that specific conceptual node. For example, if the graph detects a documented software tool but lacks its corresponding troubleshooting heuristics, the assistant will actively prompt the developer to define those edge cases. By resolving this missing context, the AI systematically reduces its own internal uncertainty through direct human escalation, satisfying the mandates of Constraint 8. The SME’s conversational responses are then processed, verified, and automatically injected back into the knowledge graph and vector database. This capability transforms the AI from a passive repository into an active participant in institutional knowledge management, continuously updating the dataset to meet the freshness and lifecycle requirements established in Constraint 6.

6.1.4 Explicit Reasoning and Chain-of-Thought Generation

In complex, physics-based engineering domains, the utility of an AI assistant relies heavily on the transparency of its logical deductions. While the current RAG pipeline successfully traces factual claims back to their source documentation, it does not inherently expose the model’s internal reasoning process. A critical future enhancement for the HELENA platform is the systemic integration of chain-of-thought (CoT) generation [37].

By explicitly training the assistant to break down complex, multi-step engineering problems into transparent, sequential logical steps before outputting a final conclusion,

the platform dramatically increases both accuracy and human trust. Future iterations of the slow-burn verification process will require human SMEs to grade and correct not only the final answer but the intermediate reasoning steps. This curated reasoning data will be embedded directly into the LoRA training pipeline, forcing the resultant domain assistants to consistently “think out loud.” Exposing these intermediate steps provides a clear mechanism for uncertainty handling and human escalation, as mandated by Constraint 8, because engineers can instantly intervene the moment a logical fallacy or uncertain deduction occurs. Furthermore, this capability perfectly aligns with the strict verifiability mandates of Constraint 7, ensuring that users can audit the exact procedural logic the assistant utilized to solve a complex problem.

6.2 Autonomous Agency and Collaboration

Currently, the HELENA platform incubates assistants that are read-only, conversational interfaces that rely entirely on human initiation. However, resolving complex, cross-disciplinary engineering bottlenecks frequently requires active problem-solving and the execution of specialized software tools. To transition the framework from a passive repository into a system of active digital collaborators, future developments will focus on granting assistants functional agency and the ability to interact with one another. The following capabilities outline the secure implementation of tool-executing agents and adversarial multi-agent ecosystems within restricted enterprise environments.

6.2.1 Agentic Assistants (Agents) and Tool Execution

As the HELENA framework matures, a primary avenue for future development is the transition from passive knowledge retrieval to active problem-solving through the implementation of agentic assistants. Dictated by the bounded tool use requirements of Constraint 10, these agents will be granted the ability to execute specific, localized software commands. Currently, the system functions primarily as a read-only interface, synthesizing information from static data via the RAG pipeline. Future iterations will grant these assistants the ability to execute the tools safely made available to them locally. By equipping assistants with functional agency, users could instruct the assistant to perform complex, multi-step workflows, such as querying live engineering databases, running Python scripts for data analysis, or directly interfacing with institutional software environments.

However, empowering LLMs with autonomous execution capabilities introduces profound security and governance challenges, particularly within the restricted architectures detailed in Constraint 1. Allowing an AI assistant to execute code or manipulate internal databases creates significant vulnerabilities, including unauthorized data modification and susceptibility to prompt injection attacks. To mitigate these risks, future agentic capabilities must be strictly constrained by robust security guardrails. Code execution must occur within isolated sandboxes to prevent systemic compromise. Furthermore, to satisfy the access segmentation mandates of Constraint 2, the system must enforce RBAC [29], ensuring the assistant inherits only the exact network permissions of the authenticated user. Finally, all state-changing or

write-access operations will require mandatory HITL cryptographic verification, ensuring that the assistant remains a capable, verifiable tool rather than an unchecked autonomous actor.

6.2.2 Multi-Agent Collaboration and Adversarial Review

Currently, the HELENA framework is designed to produce isolated, specialized assistants tailored to a specific domain, dataset, or discipline. However, just as human engineering challenges require cross-disciplinary teamwork—a principle recognized by the collaborative creation mandates of Constraint 4—complex AI problem-solving requires similar synthesis. A future capability involves expanding the platform, or developing a parallel platform, to support multi-agent system (MAS), enabling these specialized models to collaborate, exchange context, and synthesize cross-disciplinary solutions [38].

In a multi-agent architecture, a user query could be routed by a meta-orchestrator to the appropriate domain experts. Furthermore, this multi-agent framework can be configured adversarially to enhance verification. By employing an actor-critic methodology, one assistant can generate a proposed solution or code snippet, while a secondary assistant—specifically trained on cybersecurity, for example—acts as an automated reviewer tasked with finding flaws in the proposal.

Crucially, facilitating this multi-agent interaction does not require loading multiple massive foundation models simultaneously into memory, as the HELENA architecture relies on PEFT. Instead, a single base student model can dynamically load lightweight LoRA adapters into its active context, simulating a diverse panel of experts while strictly adhering to the localized compute limitations established in Constraint 3. Furthermore, because the overarching behavior and capabilities of the agents are largely dictated by the surrounding orchestration and routing frameworks, scaling this multi-agent ecosystem is modular and computationally efficient.

6.3 Scalable Alignment and Verification

The integrity of the HELENA framework relies on the slow-burn human verification pipeline, which ensures that all synthetic data are accurate, aligned, and traceable before they influence the model’s weights. However, as the platform scales to support hundreds of unique domain assistants, relying exclusively on manual human review introduces a critical operational challenge. To accelerate the incubation life cycle without compromising the rigorous standards of enterprise engineering, the system must adopt advanced algorithmic alignment techniques. The following capabilities detail methods for automating quality control, demystifying parameter selection, and mathematically optimizing behavioral alignment.

6.3.1 Assistant Response Tailoring and Parameter Control

While the current architecture successfully grounds models in domain knowledge, a future capability is the implementation of assistant response tailoring. To be effective in varied engineering disciplines, an AI assistant must adopt the specific communication style required by its users and creators, strictly adhering to the

behavioral alignment mandates of Constraint 7. Future iterations of the platform will provide creators of the assistant with granular control over the behavioral characteristics of the generated responses, such as tone, verbosity, and formatting, through an administrative dashboard. For example, a creator could mandate that an assistant always outputs troubleshooting steps in a concise, bulleted format, or require that code reviews are strictly returned as structured JavaScript Object Notation (JSON). By embedding these response characteristics directly into the training data during the verification process, the resulting LoRA module becomes uniquely tailored to the creator’s operational preferences.

Furthermore, to uphold the principle that enterprise AI must not operate as an unmanageable black box—consistent with the configuration control and reproducibility requirements of Constraint 5—future developments will expose the systemic algorithmic configurations to the creator. Creators will be granted the authority to dictate the exact parameters used during the incubation process. This capability includes selecting the student model that best aligns with their specific requirements, as well as tuning the foundational LoRA hyperparameters, such as rank, alpha scaling, and dropout rates. By demystifying the training and inference pipelines, the platform ensures that creators retain absolute authority over both the knowledge the assistant retrieves and the mathematical mechanics of how that intelligence is synthesized, fully satisfying the traceability mandates of Constraint 4.

6.3.2 AI-Assisted Verification (LLM-as-a-Judge)

While the slow-burn human verification process guarantees a high standard of accuracy and alignment, relying exclusively on human SMEs introduces a significant scaling challenge. Human verification is inherently time-intensive, and as the platform generates vast quantities of synthetic data for varied assistants, manual review risks becoming a primary operational bottleneck. To prevent this friction and accelerate the incubation life cycle, future development pipelines will integrate larger, constrained teacher models to execute AI-assisted verification.

Operating under a paradigm commonly referred to as LLM-as-a-Judge [39], these secondary teacher models will act as an automated preliminary filter for the synthetic data. Before a human verifier ever interacts with the data, the judge model evaluates the assistant’s outputs against a strict structural rubric, automatically filtering out explicit hallucinations, formatting errors, or contextually ungrounded responses. Handling the rote evaluation of baseline accuracy, the system ensures that human experts only expend cognitive effort reviewing high-quality edge cases and nuanced technical logic. This dramatically accelerates the transition from SFT to formalized reinforced learning through human feedback (RLHF) [9] while ensuring the final, verified dataset strictly adheres to the alignment mandates of Constraint 7, maintaining the HITL integrity required by advanced engineering environments.

6.3.3 Advanced Alignment via Reward Modeling and DPO

Constraints 6 and 9 outline the critical need for HELENA to address continuous knowledge lifecycle management and rigorous operational readiness before deploy-

ment. Currently, the framework addresses these mandates by relying on SFT [9, 40] through the slow-burn verification of synthetic data. While effective for initial specialization and deployment qualification, scaling this alignment process over the assistant’s lifecycle requires transitioning to formalized RLHF [41, 42]. Future iterations of the platform will utilize human verification data not just as static text, but as preference rankings (e.g., indicating that one specific response format is strictly preferred over another).

Traditionally, these preference data are used to train a separate reward model that scores the assistant’s outputs to guide optimization via algorithms like proximal policy optimization (PPO) [9]. However, executing both a primary assistant and a secondary reward model simultaneously is computationally intensive, presenting a direct challenge to the hardware limitations outlined in Constraint 3. Consequently, the HELENA platform will bypass this traditional architecture and actively explore the implementation of direct preference optimization (DPO) [43]. DPO achieves the alignment goals of traditional RLHF by mathematically optimizing the language model directly on the human preference data, completely eliminating the need to train and host a separate reward model. By streamlining this process, DPO provides a computationally-efficient pathway to achieve behavioral alignment, ensuring the system remains viable for localized, consumer-grade hardware deployments.

6.4 Enterprise Infrastructure and Deployment

As specialized AI assistants are successfully incubated and distributed across disparate engineering teams, the platform must dynamically scale to meet enterprise-wide demand. Operating entirely on-premises behind corporate or government firewalls imposes limitations on computational resources and data sharing across isolated departments. To ensure the HELENA ecosystem remains both hardware-efficient and secure at scale, the underlying infrastructure must evolve. The following capabilities explore advanced routing architectures and decentralized training methodologies designed to support a unified, enterprise-level intelligence.

6.4.1 Dynamic LoRA Routing

As the repository of specialized assistants grows, the operational logistics of interacting with multiple domain experts simultaneously become increasingly complex. If an engineer encounters a cross-disciplinary problem that requires insights from both an acoustic modeling assistant and an aerodynamic modeling assistant, manually switching interfaces or loading multiple foundation models into memory contradicts the localized compute limitations established in Constraint 3. To provide a seamless user experience, the platform will ultimately require intelligent, dynamic routing mechanisms.

To achieve this, future deployments will implement an architecture similar to a mixture-of-experts (MOE) tailored specifically for parameter-efficient adapters [44]. When a user submits a query to a unified enterprise interface, a lightweight classifier model instantly analyzes the semantic context of the prompt to determine the required domain expertise. The system then automatically swaps the appropriate

domain-specific LoRA adapter into the base student model’s active memory on the fly. This dynamic composition allows a single, hardware-efficient base model to act as a gateway to dozens of distinct engineering disciplines, providing rapid, seamless access to the entire HELENA ecosystem without overwhelming local compute resources.

6.4.2 Federated Learning of LoRA Adapters

As the HELENA platform is adopted across disparate engineering departments, an inherent tension arises between the need for cross-disciplinary collaboration (Constraint 4) and the strict data isolation requirements of organizational firewalls (Constraint 1). For instance, a propulsion team and an acoustics team may be prohibited from sharing their classified, domain-specific source data with one another. However, preventing these groups from collaborating limits the potential for a unified, enterprise-wide intelligence. To resolve this tension, future iterations of the system will explore the implementation of federated learning architectures.

Instead of pooling sensitive source data into a centralized repository, federated learning allows isolated departments to train their respective LoRA adapters locally on their own classified infrastructure. Once the localized training is complete, the departments only share the updated mathematical weights (the low-rank matrices) with a central enterprise server. This central server aggregates the weights using secure averaging algorithms to create a globally-capable assistant, which is then redistributed to the edge teams [45]. This cryptographic separation guarantees that the underlying proprietary text never crosses a firewall, merging capabilities securely while strictly adhering to data sovereignty mandates.

7 Conclusions

This paper introduced the Human-Enabled Learning & Evaluation Network for Assistants (HELENA), a locally-deployable artificial intelligence (AI) development platform designed to transform proprietary or nonproprietary seed data into specialized domain assistants. By combining retrieval-augmented generation (RAG) for grounded knowledge access, low-rank adaptation (LoRA) for behavioral and formatting control, and supervised fine tuning (SFT) through a structured verification process, HELENA provides a practical framework for developing tailored assistants that can operate within secure engineering environments. In contrast to externally hosted, general-purpose AI systems, HELENA is designed to function entirely behind organizational firewalls, preserving data sovereignty while enabling the creation of useful, specialized assistants from institutionally-relevant knowledge sources.

The HELENA architecture is motivated by a set of real operational constraints: secure handling of sensitive data, portability to limited compute environments, rapid assistant creation by non-artificial intelligence/machine learning (AI/ML) specialists, and the need for verifiability, traceability, and alignment. Within these constraints, the platform is intended not merely as a model-training workflow, but as an extensible knowledge infrastructure that helps organizations preserve expertise, reduce duplicated effort, and improve access to complex technical information. In this sense, HELENA represents a shift from static, document-centered knowledge

repositories toward interactive, human-centered knowledge systems that better reflect how engineers seek and apply information in practice.

The future capabilities discussed in this work define a clear path for expanding HELENA beyond its initial implementation. Knowledge graph integration offers a mechanism for identifying gaps, relationships, and structural weaknesses in captured knowledge. Reward-model development and preference optimization provide a path toward more refined response quality and stronger behavioral alignment. Active knowledge elicitation creates the possibility of moving from passive retrieval to deliberate knowledge expansion, allowing the system to identify what it does not yet know and guide subject matter experts (SMEs) toward high-value contributions. Similarly, response tailoring, multi-agent collaboration, vision-language integration, and bounded tool use suggest that future assistants may become increasingly adaptive, multimodal, and capable of supporting more complex technical workflows.

Taken together, these future directions do not alter the core purpose of HELENA; rather, they extend it. The long-term value of the framework lies in its ability to serve as a secure incubator for specialized assistants that are not only informed by institutional knowledge, but improved over time through structured human oversight and targeted technical enhancement. As these capabilities mature, HELENA has the potential to evolve from a knowledge-access platform into a broader ecosystem for technical onboarding, knowledge retention, workflow assistance, and expert-guided AI deployment.

Ultimately, HELENA provides a scalable blueprint for organizations seeking to capture, protect, and operationalize specialized knowledge through AI without sacrificing security, transparency, or human trust. Its future development will depend on continued refinement of its verification pipelines, deployment controls, and assistant capabilities, but the underlying premise remains clear: domain-specific AI can be both powerful and responsible when it is grounded in trusted data, developed within secure boundaries, and shaped through sustained human involvement.

List of Acronyms

AAM	advanced air mobility. 23
AAVP	Advanced Air Vehicles Program. 0
AI	artificial intelligence. 0–8, 14, 15, 21, 23–32
AI/ML	artificial intelligence/machine learning. 31
AiTHENA	Artificial Intelligence Training and Hands-on Experience at NASA. 0
ALANA	ANOPP2 LLM Assistant for Noise Applications. 21, 23
AMP	automatic mixed precision. 16
ANOPP2	Second-Generation Aircraft NOise Prediction Program. 21–23
API	application programming interface. 3
ARIANA	ANOPP2 Research & Integration Assistant for Noise Applications. 22, 23
BVI	blade-vortex interaction. 23
CAMILLA	CAMRAD II Aerodynamic Modeling Intelligent Learning and Lin- guistic Assistant. 23
CAMRAD II	Second-Generation Comprehensive Analytical Model of Rotorcraft Aerodynamics and Dynamics. 23
CFD	computational fluid dynamics. 14
DPO	direct preference optimization. 30
GPU	graphics processing unit. 4
GUI	graphical user interface. 20, 24
HELENA	Human-Enabled Learning & Evaluation Network for Assistants. 1–3, 8–18, 20–32
HITL	human-in-the-loop. 3, 6, 8, 9, 12, 13, 15–17, 20, 22, 28, 29
LLaVA	Large Language-and-Vision Assistant. 25
LLM	large language model. 2–4, 6, 8, 10–12, 14, 15, 18–21, 27
LoRA	low-rank adaptation. 1–5, 8, 11, 13, 15–24, 27–29, 31
MAS	multi-agent system. 28
MOE	mixture-of-experts. 30
NASA	National Aeronautics and Space Administration. 0, 1
PDF	portable document format. 9
PEFT	parameter-efficient fine-tuning. 4, 15, 28
PPO	proximal policy optimization. 30

RAG	retrieval-augmented generation. 1–4, 6, 8, 13, 18, 20, 21, 23–27, 31
RBAC	role-based access control. 21
RLHF	reinforced learning through human feedback. 29, 30
RVLT	Revolutionary Vertical Lift Technology. 0
SFT	supervised fine tuning. 1, 3, 6, 21, 29–31
SME	subject matter expert. 1, 2, 4, 5, 7, 12–17, 21–23, 26, 27, 29, 32
SOP	standard operating procedure. 2
SVTT	Subsonic Vehicle Technologies and Tools. 0
TACP	Transformative Aeronautics Concepts Program. 0
TF-IDF	term frequency-inverse document frequency. 14
TTT	Transformational Tools and Technologies. 0
VLM	Vision-Language Model. 25

References

1. D. B. Wechsler and K. R. Crouse, “An approach to design knowledge capture for the space station,” in *Space Station Automation II*. Cambridge, MA, United States: NASA Technical Reports Server, 1987. [Online]. Available: <https://ntrs.nasa.gov/citations/19880034415>
2. T. Grant and S. Colombano, “Methodology requirements for intelligent systems architecture,” in *AIAA Computers in Aerospace Conference*. Wakefield, MA, United States: NASA Technical Reports Server, 1987. [Online]. Available: <https://ntrs.nasa.gov/citations/19880025335>
3. A. Forward and T. C. Lethbridge, “The relevance of software documentation, tools and technologies: a survey,” in *Proceedings of the 2002 ACM symposium on Document engineering*, 2002, pp. 26–33, <https://doi.org/10.1145/585058.585065>.
4. A. J. Ko, R. DeLine, and G. Venolia, “Information needs in colocated software development teams,” in *29th International Conference on Software Engineering (ICSE’07)*. IEEE, 2007, pp. 344–353, <https://doi.org/10.1109/ICSE.2007.45>.
5. T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan *et al.*, “Language Models are Few-Shot Learners,” in *Advances in Neural Information Processing Systems*, vol. 33, Virtual Event, December 2020, pp. 1877–1901, <https://doi.org/10.48550/arXiv.2005.14165>.
6. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention Is All You Need,” in *Advances in Neural Information Processing Systems*, vol. 30, Long Beach, CA, USA, December 2017, pp. 5998–6008, <https://doi.org/10.48550/arXiv.1706.03762>.
7. P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” in *Advances in Neural Information Processing Systems*, vol. 33, Virtual Event, December 2020, pp. 9459–9474, <https://doi.org/10.48550/arXiv.2005.11401>.
8. E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “LoRA: Low-Rank Adaptation of Large Language Models,” in *International Conference on Learning Representations*, Virtual Event, April 2022, <https://doi.org/10.48550/arXiv.2106.09685>.
9. L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, “Training language models to follow instructions with human feedback,” in *Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 27 730–27 744, <https://doi.org/10.48550/arXiv.2203.02155>.
10. Joint Task Force, “Security and privacy controls for information systems and organizations,” National Institute of Standards and Technology, Tech. Rep. NIST

- Special Publication (SP) 800-53, Revision 5, 2020, <https://doi.org/10.6028/NIST.SP.800-53r5>.
11. National Institute of Standards and Technology (NIST), “Artificial intelligence risk management framework (ai rmf 1.0),” U.S. Department of Commerce, Tech. Rep., 2023, <https://doi.org/10.6028/NIST.AI.100-1>.
 12. N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson *et al.*, “Extracting training data from large language models,” in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2633–2650, <https://doi.org/10.48550/arXiv.2012.07805>.
 13. N. Houlsby, A. Giurghi, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, “Parameter-efficient transfer learning for nlp,” in *International Conference on Machine Learning (ICML)*. PMLR, 2019, pp. 2790–2799, <https://doi.org/10.48550/arXiv.1902.00751>.
 14. H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, “Llama: Open and efficient foundation language models,” *arXiv preprint arXiv:2302.13971*, 2023, <https://doi.org/10.48550/arXiv.2302.13971>.
 15. National Institute of Standards and Technology, “Artificial intelligence risk management framework: Generative artificial intelligence profile,” U.S. Department of Commerce, Tech. Rep. NIST AI 600-1, 2024, <https://doi.org/10.6028/NIST.AI.600-1>.
 16. I. Shumailov, Z. Shumaylov, Y. Zhao, Y. Gal, N. Papernot, and R. Anderson, “The curse of recursion: Training on generated data makes models forget,” *Nature*, vol. 631, pp. 590–595, 2024, <https://doi.org/10.1038/s41586-024-07566-y>.
 17. Y. Geifman and R. El-Yaniv, “Selective classification for deep neural networks,” in *Advances in Neural Information Processing Systems*, vol. 30, 2017, <https://doi.org/10.48550/arXiv.1705.08500>.
 18. National Institute of Standards and Technology, “Practices for automated benchmark evaluations of language models,” U.S. Department of Commerce, Tech. Rep. NIST AI 800-2, Initial Public Draft, 2026, <https://doi.org/10.6028/NIST.AI.800-2.ipd>.
 19. Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi, “Self-instruct: Aligning language models with self-generated instructions,” *arXiv preprint arXiv:2212.10560*, 2022, <https://doi.org/10.48550/arXiv.2212.10560>.
 20. S. Gunasekar, Y. Zhang, J. Aneja, C. C. T. Mendes, A. D. Del Giorno, S. Gopi, M. Javaheripi, P. Kauffmann, G. de Rosa, O. Saarikivi *et al.*, “Textbooks are all you need,” *arXiv preprint arXiv:2306.11644*, 2023, <https://doi.org/10.48550/arXiv.2306.11644>.

21. K. Spärck Jones, “A statistical interpretation of term specificity and its application in retrieval,” *Journal of Documentation*, vol. 28, no. 1, pp. 11–21, 1972, <https://doi.org/10.1108/eb026526>.
22. G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988, [https://doi.org/10.1016/0306-4573\(88\)90021-0](https://doi.org/10.1016/0306-4573(88)90021-0).
23. N. Sambasivan, S. Kapania, H. Highfill, D. Akrong, P. Paritosh, and L. Aroyo, ““everyone wants to do the model work, not the data work”: Data cascades in high-stakes ai,” in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, <https://doi.org/10.1145/3411764.3445518>.
24. X. Liu and Y. Li, “Examining the double-edged sword effect of ai usage on work engagement: The moderating role of core task characteristics substitution,” *Behavioral Sciences*, vol. 15, no. 2, p. 206, 2025, <https://doi.org/10.3390/bs15020206>.
25. S. J. Condly, R. E. Clark, and H. D. Stolovitch, “The effects of incentives on workplace performance: A meta-analytic review of research studies 1,” *Performance Improvement Quarterly*, vol. 16, no. 3, pp. 46–63, 2008, <https://doi.org/10.1111/j.1937-8327.2003.tb00287.x>.
26. P. Micikevicius, S. Narang, J. Aliseti, J. Damos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, “Mixed precision training,” in *International Conference on Learning Representations (ICLR)*, 2018, <https://doi.org/10.48550/arXiv.1710.03740>.
27. T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, “LLM.int8(): 8-bit matrix multiplication for transformers at scale,” in *Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 30 318–30 332, <https://doi.org/10.48550/arXiv.2208.07339>.
28. I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in *International Conference on Learning Representations (ICLR)*, 2019, <https://doi.org/10.48550/arXiv.1711.05101>.
29. D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, “Proposed nist standard for role-based access control,” *ACM Transactions on Information and System Security (TISSEC)*, vol. 4, no. 3, pp. 224–274, 2001, <https://doi.org/10.1145/501978.501980>.
30. L. V. Lopes and C. L. Burley, “ANOPP2’s user’s manual,” National Aeronautics and Space Administration, Tech. Rep. NASA/TM-2016219342, October 2016, <https://ntrs.nasa.gov/citations/20160014858>.
31. F. Farassat, “Derivation of formulations 1 and 1A of Farassat,” National Aeronautics and Space Administration, Tech. Rep. NASA/TM-2007214853, March 2007, <https://ntrs.nasa.gov/citations/20070010579>.

32. L. V. Lopes, “ANOPP2’s farassat formulations internal functional module (affim) reference manual,” National Aeronautics and Space Administration, Langley Research Center, Hampton, Virginia, Technical Memorandum NASA/TM-20210021111, December 2021. [Online]. Available: <https://ntrs.nasa.gov/citations/20210021111>
33. W. Johnson, *Rotorcraft Aeromechanics*. Cambridge University Press, 2013, <https://doi.org/10.1017/CBO9781139177116>.
34. H. Liu, C. Li, Q. Wu, and Y. J. Lee, “Visual instruction tuning,” in *Advances in Neural Information Processing Systems*, vol. 36, 2024, <https://doi.org/10.48550/arXiv.2304.08485>.
35. D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, and J. Larson, “From local to global: A graph rag approach to query-focused summarization,” *arXiv preprint arXiv:2404.16130*, 2024, <https://doi.org/10.48550/arXiv.2404.16130>.
36. M. Aliannejadi, H. Zamani, F. Crestani, and W. B. Croft, “Asking clarifying questions in open-domain information-seeking conversations,” in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019, pp. 475–484, <https://doi.org/10.1145/3331184.3331265>.
37. J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, “Chain-of-thought prompting elicits reasoning in large language models,” in *Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 24 824–24 837, <https://doi.org/10.48550/arXiv.2201.11903>.
38. Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch, “Improving factuality and reasoning in language models through multiagent debate,” *arXiv preprint arXiv:2305.14325*, 2023, <https://doi.org/10.48550/arXiv.2305.14325>.
39. L. Zheng, W.-L. Chiang, Y. Sheng, S. Hao, Z. Wu, S. Ba, E. Zhuang, Y. Lin, H. Zhuang, D. Li *et al.*, “Judging llm-as-a-judge with mt-bench and chatbot arena,” in *Advances in Neural Information Processing Systems*, vol. 36, 2024, <https://doi.org/10.48550/arXiv.2306.05685>.
40. J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, “Finetuned language models are zero-shot learners,” in *International Conference on Learning Representations (ICLR)*, 2022, <https://doi.org/10.48550/arXiv.2109.01652>.
41. P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, “Deep reinforcement learning from human preferences,” in *Advances in Neural Information Processing Systems*, vol. 30, 2017, <https://doi.org/10.48550/arXiv.1706.03741>.

42. N. Stiennon, L. Ouyang, J. Wu, D. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. F. Christiano, “Learning to summarize from human feedback,” in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 3008–3021, <https://doi.org/10.48550/arXiv.2009.01325>.
43. R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn, “Direct preference optimization: Your language model is secretly a reward model,” in *Advances in Neural Information Processing Systems*, vol. 36, 2024, <https://doi.org/10.48550/arXiv.2305.18290>.
44. C. Huang, Q. Liu, B. Wright, P. Carew, L. Gui, and H. Guan, “Lorahub: Efficient cross-task generalization via dynamic lora composition,” *arXiv preprint arXiv:2307.13269*, 2023, <https://doi.org/10.48550/arXiv.2307.13269>.
45. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2017, pp. 1273–1282, <https://doi.org/10.48550/arXiv.1602.05629>.