UNIVERSITY OF BAMBERG

International Software Systems Science Degree Program in the
Faculty of Information Systems and Applied Computer Sciences

Master's Thesis

Leveraging AI in Data-Sensitive Domains: A Framework for Developing Hybrid AI Systems with a Focus on Ensuring Information Integrity

BY

Paul Sigloch

Supervisors:
Prof. Dr. Christoph Benzmüller
David Fuenmayor

Chair for AI Systems Engineering (AISE)

Bamberg 2025

URN: urn:nbn:de:bvb:473-irb-112146x

DOI: https://doi.org/10.20378/irb-112146

Abstract

This thesis investigates the development of hybrid Artificial Intelligence (AI) systems for data-sensitive domains, where information integrity and confidentiality are essential. While modern AI offers impressive performance, it often lacks the reliability required in critical applications, particularly when handling sensitive data where hallucinations, inconsistencies, or privacy breaches can have severe consequences. To address this challenge, a project management-inspired framework is proposed to guide the development of hybrid systems that strategically combine symbolic and sub-symbolic methods, leveraging the complementary strengths of rule-based reasoning and neural network capabilities.

The framework provides structured guidance across five interconnected phases: assessment and planning, design and implementation preparation, implementation and integration, evaluation and refinement, and deployment and continuous improvement. It emphasizes data sensitivity as a core principle throughout the development lifecycle and includes a supporting toolkit of reusable components. The framework was validated through Action Design Research methodology via the iterative development of a medical device damage assessment application in collaboration with a domain expert. This paradigm implementation demonstrates how symbolic reasoning can be effectively integrated with large language models to create systems that balance generative power with verifiable accuracy.

The resulting application proved suitable for real-world deployment, achieving a 30% reduction in report creation time while maintaining the information integrity required for professional use. The system successfully detected over 83% of hallucinated content and 90% of missing critical information in controlled tests, with user-centered evaluation confirming its practical utility and usability in actual workflows. By facilitating knowledge transfer between theoretical advances and real-world implementation, this work enables the creation of AI systems that combine innovation with robust information guarantees. The resulting methodology offers a sustainable, privacy-aware approach for deploying AI in high-stakes domains while addressing critical concerns regarding automation bias and meaningful human oversight in AI-assisted professional workflows.

Acknowledgements

I would like to express my profound gratitude to everyone who supported me throughout this research journey.

I am deeply indebted to Professor Christoph Benzmüller, whose mentorship, critical insights, and academic guidance shaped not only this thesis but also my approach to research. His expertise in formal methods and AI provided a strong foundation for this work.

My sincere appreciation goes to David Fuenmayor for his meticulous attention to detail and rigorous feedback, which significantly improved the methodological precision and theoretical depth of this thesis. His support with thesis formalities was also invaluable.

I am grateful to Fabian Kropfhamer, Marc Nickert, and Ibrahim Khalil for their technical expertise in fine-tuning strategies and hybrid AI architectures. Their insights were instrumental in translating theoretical concepts into practical solutions, particularly during complex development phases.

Special thanks to Walter Sigloch, whose extensive domain knowledge in medical device assessment grounded this research in real-world needs. His willingness to share professional experiences, define practical requirements, and participate in prototype evaluations was crucial to the development and validation of the medical device assessment application built alongside this thesis.

I also wish to thank my sister, Lisa Sigloch, and Michael Pries for their thoughtful reviews and editorial suggestions, which greatly enhanced the clarity and accessibility of this document.

Finally, my heartfelt thanks go to my family and friends for their encouragement, patience, and unwavering support throughout this challenging but rewarding journey.

List of Abbreviations

ADR	Action Design Research
AI	Artificial Intelligence
API	Application Programming Interface
BEAM	Bogdan/Björn's Erlang Abstract Machine
BERT	Bidirectional Encoder Representations from Transformers
BF16	16-bit Brain Floating Point
BIE	Building, Intervention, and Evaluation
BRMS	Business Rule Management Systems
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DAPT	Domain-Adaptive Pre-Training
DOCM	Microsoft Word Macro-Enabled Document
DOCX	Microsoft Word Open XML Document
FBM	Feedback Module
FDA	Food and Drug Administration
FP16	16-bit Floating Point
GAM	Gateway API Module
GDPR	General Data Protection Regulation
GGUF	GPT-Generated Unified Format
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HAIMEDA	Hybrid AI for Medical Device Assessment
HATR	Hybrid AI Task Router
HTML	Hypertext Markup Language
IIVM	Information Integrity Verification Module
INT8	8-bit Integer
IT (Technology)	Information Technology
IT (AI)	Instruction Tuning
JS	JavaScript
JSON	JavaScript Object Notation
JSONL	JavaScript Object Notation Lines
LLM	Large Language Model
LoRA	Low-Rank Adaptation
LSM	LLM Service Module
MD	Markdown
MDB	Microsoft Database
ML	Machine Learning
NLP	Natural Language Processing
OS	Operating System

OSCAR	Open Super-large Crawled Aggregated coRpus
OTP	Open Telecom Platform
PDF	Portable Document Format
PEFT	Parameter-Efficient Fine-Tuning
PII	Personally Identifiable Information
RAG	Retrieval-Augmented Generation
RAM (Computing)	Random Access Memory
RAM (Module)	Report Authoring Module
RIM	Research and Investigation Module
Regex	Regular Expression
SFT	Supervised Fine-Tuning
SLERP	Spherical Linear Interpolation
SMT	Satisfiability Modulo Theories
SQL	Structured Query Language
SUS	System Usability Scale
TF-IDF	Term Frequency-Inverse Document Frequency
UI	User Interface
VRAM	Video Random Access Memory
WSL	Windows Subsystem for Linux
XAI	Explainable Artificial Intelligence
YAML	Yet Another Markup Language

List of Figures

List of Tables

Contents

1 Introduction

1.1 Challenges in Modern AI Applications

By 2023, AI had established itself as a major topic of discussion, spurring a wide range of debates about its potential uses, limitations, and future developments [Tho23], [Pre22]. While for many it represented a breakthrough in efficiency and innovation across the entire industry, for others it raised concerns about ethical implications such as job displacement or the transparency of decisions made by AI. Ultimately, the initial excitement regarding AI, particularly the advances in Large Language Models (LLMs) such as ChatGPT (see Figure 1.1), led to significant interest among both experts and the general public [FT23], [Mas+24, pp. 456f].

However, after the initial hype subsided, the limitations of publicly available and easily accessible AI systems such as ChatGPT[1] and Claude[2] became more apparent, also to non-experts. Although these models can generate impressive responses, they suffer from several limitations, such as hallucinations, i.e., the production of false or nonsensical information (see Figure 1.2), the difficulty of keeping up with real-time information, and the lack of domain-specific expertise required for more advanced tasks [Hua+24, pp. 1f], [Ji+23, pp. 3, 6]. Large-scale language models, like those developed for general use, are primarily designed to perform well on a wide range of common tasks. While this makes them versatile and accessible to a wider audience, they are not specifically optimized for highly specialized or niche applications [Bom+22, pp. 5-7], [Ben+21, pp. 610f, 617].

1: https://chatgpt.com/
2: https://claude.ai/

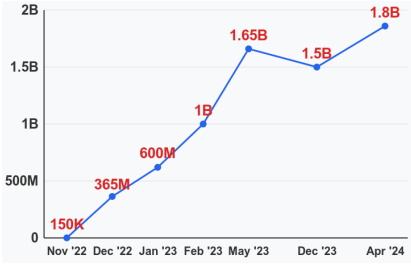


Figure 1.1: ChatGPT website visitors since launch (Nov 2022–Apr 2024) (Adapted from: https://www.tooltester.com/en/blog/chatgpt-statistics/).

Thus, both the scientific community and industry seek to integrate AI systems that perform well in their respective tasks and address shortcomings that compromise information reliability. In particular, professionals in data-sensitive environments, such as the medical field, are eager to find automation solutions that prioritize information integrity and minimize the risk of generating inaccurate content [WP24, pp. 6-8].

To address these limitations, a more sophisticated approach is required, one that combines the strengths of different AI paradigms while mitigating their individual weaknesses. This approach, known as hybrid AI, integrates symbolic AI, which encodes knowledge as explicit rules and logical structures, with sub-symbolic AI, which derives patterns and representations automatically from data through techniques such as Machine Learning (ML) and other statistical methods [Hau85, pp. 112ff], [Kel03, pp. 849f]. By uniting these complementary paradigms (see Figure 1.3), hybrid

AI systems offer a pathway to architectures that can handle uncertainty while maintaining formal guarantees about their behavior [Gar+19, pp. 611f], [MD19, pp. 177f].

1.2 Hybrid AI Systems as a Solution Approach

While traditional AI approaches tend to excel in specific domains but struggle with others, hybrid AI systems offer a more comprehensive solution by establishing complementary relationships between different AI paradigms. Beyond the already mentioned benefits of reducing hallucinations and maintaining formal guarantees, hybrid architectures provide several additional advantages crucial for data-sensitive applications.

Hybrid AI systems demonstrate greater adaptability to novel situations and changing requirements, a critical feature for domains with evolving knowledge bases. They encode symbolic background knowledge as logical constraints within neural architectures, yielding robust, data-driven learning coupled with symbolic interpretability and explainability. Furthermore, hybrid architectures facilitate more efficient knowledge transfer across domains by separating general reasoning capabilities from domain-specific knowledge [Gar+19, pp. 621-623]. Moreover, these systems enable interactive refinement processes where domain experts can directly correct reasoning errors and provide contextual knowledge without requiring complete system retraining, creating a collaborative learning environment [Del+19, pp. 639-641]. Finally, these architectures offer a principled methodology for integrated machine learning and reasoning, providing insights into the increasingly prominent need for interpretable and accountable AI systems that can robustly operate in real-world applications while maintaining human-understandable decision processes [MD19, pp. 126-130], [Gar+19, pp. 611f].

Despite these considerable advantages, developing effective hybrid AI systems presents significant design and implementation challenges. These include determining optimal integration patterns between symbolic and statistical components [Sar+21], managing knowledge representation across heterogeneous systems [GL23, pp. 12392-12397], and establishing verification methodologies that span both paradigms [Leo+18]. While existing research has made significant contributions to individual aspects of hybrid AI development, comprehensive methodological guidance that systematically addresses the integration challenges across symbolic and statistical components remains limited, particularly for data-sensitive domains that require careful consideration of privacy, reliability, and domain-specific constraints [Rud19, pp. 206ff].

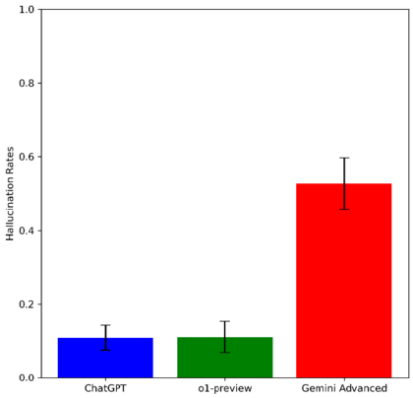


Figure 1.2: Hallucination rates of ChatGPT-4o, o1-preview, and Gemini Advanced on financial literature references, measured using the graded (Scale 2) metric. Error bars represent standard errors (Source: [EHE25, Fig. 2]).

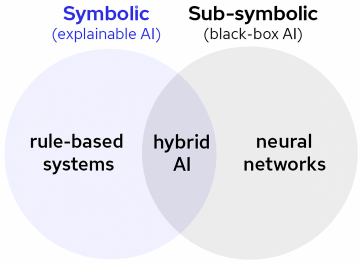


Figure 1.3: Venn diagram showing hybrid AI as the convergence of symbolic and sub-symbolic methods (Adapted from: https://www.pharmafocus.com/articles/don-t-settle-for-black-box-why-only-explainable-ai-is-built-for).

1.3 Research Objectives and Contribution

Despite significant research and specialized implementations in hybrid AI, there remains limited general guidance for developing such systems in data-sensitive domains. This work proposes a practical framework for practitioners implementing applications that integrate multiple AI paradigms for real-world use cases beyond scientific prototypes.

The framework guides practitioners through the entire development lifecycle, providing guidelines, key activities, best practices, and tools to mitigate common implementation pitfalls and reduce risks. This comprehensive approach makes the development process more efficient, allowing practitioners to focus on core logic and implementation tasks. Following project management principles, it emphasizes phases that enable agile, iterative development while incorporating privacy and data sensitivity considerations throughout.

To demonstrate the practical value of the framework, this research presents the *Hybrid AI for Medical Device Assessment (HAIMEDA)*[3] application, a comprehensive system implementation that enables damage assessment reporting for medical devices in practical settings. The medical device assessment domain provides a compelling testbed for hybrid AI through its convergence of high-stakes decision-making, regulatory complexity, and nuanced human judgment.

3: https://github.com/penthooose/HAIMEDA

Medical device surveyors operate at the intersection of law, medicine, and engineering, producing reports that influence legal outcomes, insurance settlements, and patient safety. Each assessment synthesizes fragmented information from technical manuals, device manufacturers, witness statements, and prior cases. The workflow demands continuous adaptation to new findings while maintaining strict compliance with data protection laws such as the General Data Protection Regulation (GDPR) and medical device regulations [Sig].

In this environment, hybrid AI offers unique advantages through the complementary strengths of its components. Symbolic reasoning modules enforce explicit rules and regulatory constraints, while sub-symbolic models extract patterns from unstructured data, support research-intensive tasks, and assist with report formulation. HAIMEDA combines these approaches to help surveyors manage complexity, ensure compliance, and deliver reliable assessments as a practical tool for daily assessment tasks.

Moreover, this application fulfills broader research purposes by rigorously demonstrating the framework's principles in action while providing practitioners with a tangible reference implementation. It illuminates the integration of symbolic and sub-symbolic AI components in a real-world context. By examining this implementation, practitioners across various domains can extract valuable insights and implementation patterns adaptable to their own specialized contexts, bridging the gap between theoretical hybrid AI concepts and practical deployment requirements.

1.4 Structure of the Work

Following this introduction, Chapter 2 analyzes related work to highlight the research gap, then provides an overview of AI paradigms and technical concepts employed throughout the framework and paradigm application. This foundation supports Chapter 3, which outlines the research objectives and methodological approach. Chapter 4 presents the proposed framework, detailing both the conceptual approach to hybrid AI system development and practical tools designed to streamline the development process. Following this in Chapter 5, the paradigm application, i.e., HAIMEDA, demonstrates the framework's implementation, including system architecture and module specifications. Quantitative and qualitative evaluation techniques are applied to both the framework and paradigm application in Chapter 6, while Chapter 7 discusses their strengths and limitations. Chapter 8 concludes the research by providing a final summary of the main findings and presenting an outlook on future directions.

2 Background and Motivation

This chapter establishes a foundation for understanding hybrid AI development in data-sensitive domains. Section 2.1 presents an overview of established work in the targeted domains, while Sections 2.2 and 2.3 formally define key terms, concepts, technologies, and AI paradigms referenced throughout this work, providing necessary background knowledge prior to the presentation of the methodology in Chapter 3.

2.1 Related Work

Related literature concerning hybrid AI development can be organized into four core areas that intersect with data-sensitive domain requirements relevant to this thesis. The review begins with development methodologies from both general software engineering and AI-specific approaches that share structural similarities with the proposed framework. Following this, fundamental approaches to hybrid AI architectures are explored, alongside implementations of domain-specific AI systems, with particular emphasis on healthcare applications relevant to the HAIMEDA paradigm application. The examination then turns to privacy and regulatory frameworks for data-sensitive domains, as these inform critical aspects of the development process. A synthesis of these findings concludes the section, identifying the research gap that motivates this work.

2.1.1 Development Methodologies for Software Engineering

Although generalized development methodologies for hybrid AI are largely unexplored, several existing frameworks offer partial foundations relevant to this field. These works can be grouped into privacy-focused processes, human-centered approaches, and AI-specific engineering methodologies.

Privacy-oriented development processes, such as Wijesundara et al.'s [WWA25] systematization of privacy-preserving software engineering, propose structured activities and design techniques for embedding privacy controls throughout the software lifecycle. Although not targeted at AI, they offer transferable principles useful for data-sensitive system design.

Human-centered frameworks contribute ethical and user-aligned perspectives. Xu et al.'s [XGD25] Human-Centered AI Methodological Framework outlines stakeholder engagement strategies, development phases, and evaluation dimensions that support responsible AI practices. However, it does not address the integration of symbolic and sub-symbolic components central to hybrid AI.

For general AI system development, lifecycle-based methodologies and design pattern catalogues offer more technical guidance. Lu et al.'s [Lu+23] pattern-based framework compiles governance and engineering best practices for AI systems, while Gill's [Gil25] agile decision architecture adapts standard project management processes to AI-specific challenges. Both address the broader AI engineering lifecycle, but neither provides systematic support for the design, integration, or compliance aspects of hybrid AI systems.

2.1.2 Technical Implementation Frameworks for Hybrid AI

Understanding existing technical frameworks for hybrid AI implementation is essential for positioning the proposed development methodology, as these frameworks represent the current state-of-the-art in combining symbolic and sub-symbolic reasoning capabilities. Among the most influential approaches, Badreddine et al.'s [Bad+22] Logic Tensor Networks embed first-order logic constraints directly into neural architectures through a differentiable framework, enabling end-to-end learning while preserving logical consistency. Taking a different approach, Manhaeve et al.'s [Man+21] DeepProbLog extends probabilistic logic programming by integrating neural predicates into Prolog, facilitating joint training of perception and inference modules. More recently, Li et al.'s [LHN23] Scallop advances the integration by proposing a differentiable Datalog engine that compiles logic programs into tensor computations, supporting scalable, provenance-aware reasoning. This capability is particularly relevant for data-sensitive domains requiring explainable decisions.

2.1.3 Domain-Specific Hybrid AI Systems

Examining domain-specific hybrid AI implementations reveals critical design patterns and architectural decisions that inform the development of systems for data-sensitive environments, particularly regarding the balance between performance, explainability, and regulatory compliance. In healthcare settings, several systems demonstrate this trend. Chudasama et al.'s [Chu+25] TrustKG exemplifies this by integrating neural models with symbolic reasoning over knowledge graphs to support transparent and accountable clinical decision-making, combining counterfactual inference with rule-guided machine learning to provide interpretable predictions and causal insights. Similarly, Amato and Branco's [AB25] SemFedXAI presents a semantic framework for explainable federated learning in healthcare, combining federated architectures with symbolic ontologies to preserve privacy while enhancing interpretability across distributed clinical data. García-Barragán et al.'s [Gar+24] Neuro-Symbolic System for Cancer further embodies this paradigm by enhancing named

entity recognition and linking in oncologic clinical records through a hybrid architecture that fuses deep learning with ontology-driven disambiguation, showcasing how domain-specific knowledge can improve accuracy in specialized contexts.

Beyond healthcare, hybrid approaches show versatility across other sensitive domains. In robotics, Capitanelli and Mastrogiovanni's [CM24] Teriyaki framework demonstrates hybrid AI's applicability by combining LLMs with symbolic planning languages to generate interpretable action plans for human-robot collaboration scenarios, enabling improved planning flexibility and reduced execution latency while maintaining verifiable outputs in dynamic environments. In industrial settings, Cao et al.'s [Cao+22] Knowledge-based System for Predictive Maintenance in Industry 4.0 illustrates the utility of neuro-symbolic methods by integrating predictive modeling with domain knowledge bases to facilitate transparent and adaptive maintenance decisions.

2.1.4 Privacy and Regulatory Frameworks for Data-Sensitive Systems

A number of conceptual frameworks and principles have been developed to address privacy concerns in digital systems more broadly, including but not limited to AI. These frameworks primarily target general software systems rather than hybrid AI specifically. Cavoukian's [Cav10] Privacy by Design framework advocates for embedding privacy-preserving mechanisms throughout the entire lifecycle of a system, from initial design to deployment and maintenance, emphasizing default privacy, end-to-end security, and user-centric control. Similarly, the National Institute of Standards and Technology Privacy Framework 1.1 [Nat23] provides a structured methodology for assessing and managing privacy risks across diverse technological contexts. Herwanto et al.'s [Her+24] Holistic Privacy Requirements Engineering approach further emphasizes the integration of privacy requirements early in software development processes, aiming to build systems that are compliant and accountable by design.

Complementing these conceptual frameworks, various regulatory frameworks establish formal requirements and legal boundaries for privacy in data-centric systems. The Health Level Seven Fast Healthcare Interoperability Resources standard, as described by Bender and Sartipi [BS13], defines interoperability and data exchange protocols for electronic health information. The GDPR [Eur16] sets comprehensive privacy rights and obligations within the European Union, mandating transparency, informed consent, data minimization, and user access rights.

Furthermore, a range of technical toolkits and software frameworks have been introduced to address privacy risks more directly in the context of AI development. Federated learning systems such as TensorFlow Federated [Ten24a] allow decentralized model training across distributed nodes, mitigating the need to transfer raw data and thus enhancing data locality and privacy, which are particularly beneficial in data-sensitive domains like healthcare [Rie+20, pp. 1-3]. Other privacy-preserving machine learning toolkits such as TensorFlow Privacy [Ten24b] and PyTorch

Opacus [PyT24] incorporate differential privacy mechanisms that add carefully calibrated noise to ensure individual-level data protection during training. Additionally, Microsoft Simple Encrypted Arithmetic Library [Mic24] provides homomorphic encryption capabilities that allow computations to be performed on encrypted inputs, maintaining data confidentiality throughout the process.

2.1.5 Summary and Research Gaps

This review reveals critical gaps in methodologies for hybrid AI development in data-sensitive domains. Despite advances in individual techniques, the field lacks a structured approach that integrates privacy protection and regulatory compliance with hybrid AI engineering principles. While privacy-focused software engineering frameworks exist, they don't address the unique challenges of combining symbolic and sub-symbolic AI components in sensitive contexts.

Current research demonstrates several key limitations. First, technical approaches disproportionately focus on neuro-symbolic integration while neglecting other promising hybrid paradigms. Second, there's insufficient integration between theoretical constructs and practical tooling, constraining immediate applicability. Third, current approaches show limited integration with functional programming paradigms, contraining architectural options for hybrid AI development. Fourth, implementation examples typically present specialized point solutions without extracting generalizable development patterns. Fifth, domain coverage skews heavily toward healthcare applications, leaving equally sensitive sectors like finance, legal services, and public governance underexplored, which impedes comprehensive understanding of cross-domain requirements.

These limitations are compounded by the absence of standardized evaluation frameworks that assess hybrid AI systems across dimensions of performance, privacy preservation, interpretability, and regulatory compliance. Without such metrics, meaningful comparison between approaches remains challenging. Collectively, these findings underscore the need for a comprehensive development framework that bridges technical requirements of hybrid architectures with socio-technical constraints in sensitive domains, providing practitioners with structured guidance currently absent from literature.

2.2 AI Paradigm Classifications

This section provides formal definitions of the existing AI paradigms, examining their strengths, limitations, and applications.

2.2.1 Symbolic AI

Symbolic AI, also known as Good Old-Fashioned AI or rule-based AI, represents an approach that relies on explicit representation of knowledge using symbols and rules for reasoning [Hau85, pp. 112ff]. This paradigm

models intelligence as the manipulation of symbolic representations through formal logic and rule-based systems, enabling machines to perform logical deductions and problem-solving tasks in a transparent, interpretable manner [RN09, pp. 24-28].

At its core, symbolic AI operates on predefined rules, logical structures, and explicit knowledge representation [NS76, pp. 115-117] (see Figure 2.1). This approach focuses on encoding domain expertise and knowledge in formal systems that machines can interpret and manipulate. The knowledge is typically represented as symbols, i.e., concepts, entities, and relations, that are manipulated according to well-defined rules of inference [Bee22, pp. 419f].

Figure 2.1: Schematic representation of symbolic AI: explicit programming combines data and rules to produce answers (Adapted from: Fig. 2, https://machinemindscape.com/artificial-intelligence-to-deep-learning-history-concepts/).

Strengths

Symbolic AI offers several significant benefits that make it valuable for specific applications. Its operations are highly transparent and interpretable, as reasoning steps can be traced through explicit rule applications. This enables formal verification of results and reliable consistency for well-structured problems. Symbolic systems excel at tasks requiring logical precision and can provide explanations for their reasoning processes. Additionally, they do not require large training datasets to function effectively [Bee22, pp. 415, 428], [MD19, pp. 7f, 14f, 38f].

Limitations

Despite its strengths, symbolic AI faces fundamental limitations. Such systems struggle with uncertainty and ambiguity, often demonstrating brittleness when encountering novel situations outside their predefined rules. A significant challenge is the knowledge acquisition bottleneck, i.e., the difficulty of manually encoding all required domain knowledge into rules [Dre92, pp. 197ff], [Len95, pp. 33f]. Symbolic approaches also perform poorly in pattern recognition tasks and lack the adaptability to learn from experience without explicit reprogramming [Len95, pp. 33ff].

Applications

Notable examples of symbolic AI include expert systems like MYCIN and DENDRAL, medical diagnosis and chemical analysis, respectively, which encode domain expertise as production rules [BS84, pp. 3f, 8f]. Other implementations include automated theorem provers like Isabelle/HOL and Coq, which verify mathematical proofs [Pau89, p. 363], and planning systems like STRIPS that represent actions and states symbolically to achieve goals [FN71, pp. 189f]. Business Rule Management Systems (BRMS), such as IBM ILog BRMS and Drools, use symbolic reasoning for enterprise decision automation [VKL16, pp. 6-8].

2.2.2 Sub-Symbolic AI

Sub-symbolic AI, also known as data-driven AI, refers to approaches that rely on statistical patterns, numerical representations, and distributed computations rather than explicit symbolic representations and logical rules [Kel03, pp. 849f]. Unlike symbolic AI, where knowledge is explicitly encoded, these approaches derive patterns and representations implicitly from data, with neural networks and ML being the predominant methodologies [Mit97, pp. 81ff] (see Figure 2.2).

Fundamentally, such systems operate by extracting regularities from large datasets, gradually adjusting internal parameters to minimize errors or maximize rewards [RHW86, pp. 533f]. This creates computational models that develop their own internal representations through exposure to data, with knowledge encoded in numerical weights and distributed patterns instead of discrete symbolic structures [BCV13, pp. 1798-1801].

Figure 2.2: Schematic representation of sub-symbolic AI: data-driven models infer rules from data and answers through statistical learning (Adapted from: Fig. 2, https://machinemindscape.com/artificial-intelligence-to-deep-learning-history-concepts/).

Strengths

Sub-symbolic AI approaches, particularly neural networks, offer significant strengths that have driven their recent prominence. These systems excel at pattern recognition tasks such as image classification, speech recognition, and natural language processing where rules are difficult to formulate explicitly. They demonstrate remarkable adaptability, learning from experience and improving with more data without requiring manual reprogramming. Sub-symbolic approaches can handle noisy, incomplete, or ambiguous data effectively, showing robustness in real-world applications [KS23, pp. 3302, 3304f]. Additionally, neural network approaches have been shown to scale much better with model size and dataset size, suggesting potential for improvement as computational resources increase [BCV13, pp. 1798ff].

Limitations

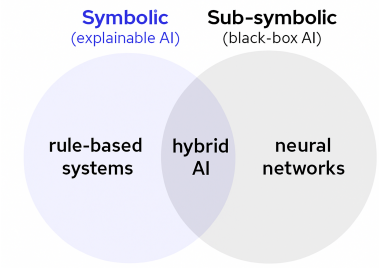
Despite their strengths, sub-symbolic systems face notable limitations. Their "black box" nature makes decision processes opaque and difficult to interpret, creating challenges for applications requiring transparency or explanations. These approaches typically require large amounts of training data and substantial computational resources, limiting their applicability in resource-constrained environments or domains with limited available data [Mar20, pp. 4-7, 12, 26f]. Additionally, they frequently generate hallucinations when faced with uncertainty and struggle with informational gaps that humans easily fill with common sense, leading to unreliable outputs in critical scenarios [MD19, pp. 26-28, 32f, 151f]. Sub-symbolic systems often struggle with tasks requiring logical reasoning, causal inference, and abstract thinking. They may also face challenges with generalization beyond their training distribution and can be vulnerable to adversarial examples that exploit their pattern-matching nature [Mar18, pp. 6, 12ff], [Sze+14, pp. 4-10].

Applications

Sub-symbolic AI has produced numerous breakthrough applications across diverse domains. Convolutional neural networks revolutionized computer vision through unprecedented image recognition capabilities [KSH12, pp. 1097, 1104f], while transformer architectures fundamentally improved natural language processing, establishing state-of-the-art performance in machine translation and other text processing tasks [Vas+17, pp. 6000f, 6008]. These transformer principles enabled LLMs like BERT that demonstrate remarkable text understanding and representation abilities [Dev+19, pp. 4171, 4175f]. Beyond perception tasks, reinforcement learning systems have mastered strategic decision-making in complex environments, notably defeating world champions in games like Go, an ancient strategy game with virtually unlimited move possibilities, that were previously considered too intuitive for machine mastery [Sil+16, pp. 484f, 488f].

2.2.3 Hybrid AI

Hybrid AI refers to approaches that combine symbolic and sub-symbolic AI paradigms (see Figure 1.3), leveraging their complementary strengths to overcome the limitations inherent in each individual approach [GL23, pp. 12390, 12397f, 12402]. This integration creates systems capable of both statistical pattern recognition and explicit logical reasoning, enabling more robust, transparent, and adaptable artificial intelligence solutions [Hit+22]. At its core, hybrid AI aims to bridge the gap between the interpretability and reasoning capabilities of symbolic AI and the pattern recognition and learning abilities of sub-symbolic approaches [GL23, pp. 12397f], [Bes+21, pp. 35f].



(cf. Figure 1.3)

2.2.3.1 Hybrid AI Integration Patterns

Hybrid AI systems can be categorized according to how and when symbolic and sub-symbolic components interact within the system architecture. Three primary integration patterns have emerged in contemporary research and applications:

1. Pre-processing integration (before inference): In this pattern, symbolic components influence the system before the sub-symbolic inference process begins [Rue+23, pp. 618ff]. This might involve rule-based prompt construction, knowledge graph-guided input formulation, or symbolic constraint setting that shapes how inputs are processed. For example, ontology-based query expansion can enhance information retrieval by incorporating domain knowledge before neural processing occurs [NC10, pp. 48ff]. This approach represents a lightweight hybrid method that preserves the efficiency of neural inference while incorporating symbolic knowledge [ZS24, pp. 20ff] (see Figure 2.3).

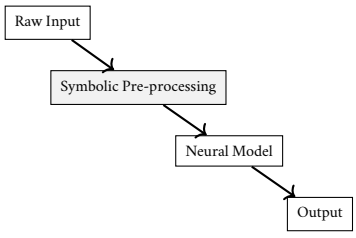


Figure 2.3: Pre-processing integration: symbolic components (e.g., rules, ontologies) enrich or transform the input before neural inference.

2. Integrated processing (during inference): More sophisticated hybrid systems enable symbolic and sub-symbolic components to interact

dynamically during the inference process itself [Sar+21, pp. 198f] (see Figure 2.4). This can take several forms:

- *Constrained decoding*, where symbolic constraints directly guide the neural generation process by modifying token probabilities or enforcing grammar rules during text generation [Gen+23, pp. 10932f].
- *Neural-symbolic reasoning*, where neural networks and symbolic reasoners work iteratively, with intermediate results passed between components [Lam+21, p. 4878].
- *Tool-augmented generation*, where neural models are equipped to call external symbolic tools during inference, e.g., calculators, knowledge bases, or logical verifiers [Sch+23, pp. 68539ff].
- *Hierarchical planning approaches*, where symbolic planning modules guide high-level task decomposition while neural reinforcement learning handles low-level policy execution for specific subtasks [Yu+23, pp. 119f].

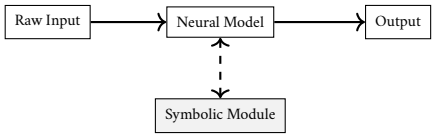


Figure 2.4: Integrated processing: symbolic and neural components interact during inference (e.g., constrained decoding, tool calls, or iterative reasoning).

3. Post-processing integration (after inference): In this pattern, outputs from sub-symbolic systems are verified, corrected, or enhanced by symbolic components [Sar+21, pp. 198ff]. Common techniques include logical consistency checking, rule-based output filtering, and formal verification of generated content. This approach is particularly valuable in domains requiring high precision or regulatory compliance, as symbolic processing can enforce hard constraints that neural systems might occasionally violate [SWS22, pp. 1f, 7f], [XKN22, pp. 3ff] (see Figure 2.5).

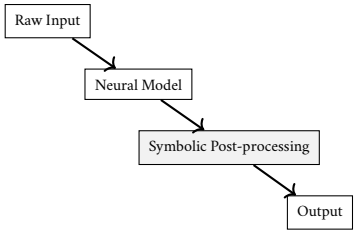


Figure 2.5: Post-processing integration: symbolic components verify, filter, or enhance neural outputs after inference.

2.2.3.2 Types of Hybrid AI Architectures

Although integration patterns describe how symbolic and sub-symbolic components interact within the processing pipeline, their structural organization results in distinct architectural configurations. These affect system behavior, scalability, and development complexities. The following architectural types encompass the predominant approaches observed in hybrid AI systems, recognizing that practical implementations may demonstrate variations, combinations, or novel arrangements beyond this taxonomy:

- *Sequential hybrid systems* arrange symbolic and sub-symbolic components in a pipeline, where the output of one becomes the input to the next. This approach maintains clear separation between components but may limit the system's ability to leverage complementary strengths during complex reasoning [Bek+21, pp. 6533ff].
- *Parallel hybrid systems* deploy symbolic and sub-symbolic components simultaneously, with their outputs combined through integration mechanisms. These architectures often employ voting or confidence-weighted aggregation to combine predictions from different components [Tra+22, pp. 7-10].

- *Embedded hybrid systems* incorporate symbolic elements directly within sub-symbolic architectures, such as neural networks with explicit symbolic representations in hidden layers or attention mechanisms guided by symbolic knowledge. Neuro-symbolic networks that encode logical constraints within their structure represent a prominent example of this approach [WYW25, pp. 882-887].
- *Multi-agent hybrid systems* organize different AI paradigms as specialized agents that collaborate through coordination protocols. These systems typically include a meta-reasoning layer that determines which agent is most suitable for handling specific tasks or subproblems [MMB21, pp. 3-7].

Strengths

Hybrid AI offers substantial benefits over purely symbolic or sub-symbolic approaches. By combining complementary paradigms, these systems achieve enhanced performance across various dimensions, demonstrating improved transparency and explainability compared to black-box neural models, as symbolic components can provide insights into reasoning processes [GL23, pp. 12395ff] (see Figure 2.6). They also typically demonstrate better generalizability when representative training data is limited, as symbolic knowledge components can compensate for underrepresented classes in imbalanced datasets [Hoo+25, pp. 2ff].

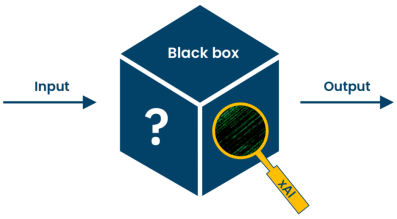


Figure 2.6: Schematic illustration of Explainable AI (XAI): a black-box model transforms input into output, while XAI techniques provide interpretability by revealing internal decision processes (Source: Fig. 1, https://papermaker.ai/explainable-ai/).

These systems excel at complex reasoning tasks that involve both pattern recognition and logical deduction, making them well-suited for domains like medical diagnosis, legal reasoning, and scientific purposes. Additionally, hybrid AI can enforce safety constraints and regulatory compliance more reliably than pure neural approaches, addressing a critical concern for deployment in sensitive domains [Hit+22], [Mar20, pp. 5f, 18, 54].

Limitations

Despite their advantages, hybrid AI systems face significant challenges. Integration complexity increases development effort and requires expertise in both symbolic and sub-symbolic approaches. Maintaining consistency between different reasoning paradigms can be challenging, especially when they produce ambiguities in the decision-making process [Chr23, pp. 3ff].

The computational overhead of combining multiple AI paradigms may impact system performance, particularly for real-time applications [Bes+21, p. 10], [Wan+24, pp. 54f]. Additionally, hybrid systems often require domain-specific engineering, limiting their generalizability across different problem domains without significant adaptation [BHC22, pp. 418, 424].

Applications

Hybrid AI has found successful applications across numerous domains. In healthcare, hybrid systems combine statistical pattern recognition with

medical knowledge bases for more accurate and explainable diagnosis systems [Vid+25, pp. 1ff]. Financial institutions employ hybrid AI approaches for fraud detection, combining rule-based logic with AI predictions to balance detection accuracy with interpretability and regulatory compliance requirements [Shi25, pp. 27f].

In autonomous vehicles, symbolic components enforce safety constraints and traffic rules while neural networks handle pattern recognition and perception tasks [MP25, pp. 5f] (see Figure 2.6). Natural language processing applications can benefit from hybrid deep learning approaches for text analysis, while knowledge graphs provide complementary capabilities by extracting organized knowledge to aid in semantic understanding [Mee+23, pp. 1f, 8f].

The convergence of symbolic and sub-symbolic approaches continues to evolve, with increasing recognition that their integration represents a promising direction for achieving more robust, trustworthy, and capable AI systems that can address the complex challenges of real-world applications [Mar20, pp. 17f, 49].

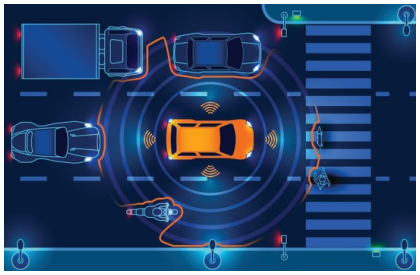


Figure 2.7: Schematic illustration of hybrid AI in autonomous vehicles (Source: Fig. 1, https://engineering.stanford.edu/news/new-ai-camera-recognizes-objects-faster-and-more-efficiently).

2.3 Technical Concepts Utilized in This Work

This section outlines and explains the specific AI paradigms and techniques utilized throughout this work, particularly in the HAIMEDA application. Additionally, it introduces the key concepts of data-sensitivity and information integrity, which are central to both the framework development and the HAIMEDA paradigm implementation.

2.3.1 Symbolic Methods

2.3.1.1 Pattern Matching and Regular Expressions

Pattern matching is a fundamental symbolic method for text processing and information extraction, enabling the identification of specific structures within unstructured content based on formal definitions. Regular expressions (regexes) are a compact, formal language for specifying such search patterns, theoretically grounded in finite-state automata [Reg09, pp. 17ff, 93ff], [GH14, pp. 25ff]. They enable deterministic string matching with predictable computational bounds through efficient implementations, as demonstrated in Table 2.1. In symbolic AI systems, Regexes serve as a key building block for rule-based entity extraction, allowing explicit encoding of domain expertise to identify elements such as dates, numerical values, identifiers, or semantic constructs [Sim+18, p. 5924]. Unlike statistical approaches that learn patterns implicitly from data, regexes offer complete transparency and precise control over extraction logic, ensuring predictable behavior, formal verification, and adaptability through configurable pattern hierarchies [Mit21, pp. 72ff], [Sim+18, pp. 5924ff].

Table 2.1: Regex for University of Bamberg ID numbers (BA numbers): case-insensitive BA prefix with 4–6 alphanumeric characters.

Pattern	Matches
`[Bb][Aa][A-Za-z0-9]{4,6}`	BA058123 ba4zt3

2.3.1.2 Formal Logic and Reasoning Systems

Formal logic systems symbolically represent knowledge using well-defined languages with precise syntax and semantics, enabling rigorous reasoning through manipulation of symbolic structures. Propositional logic forms the foundation, combining atomic statements with logical operators such as conjunction, disjunction, implication, and negation; predicate logic extends this with quantifiers and variables to express complex relationships and generalizations. Tableaux methods provide a systematic proof technique, decomposing complex formulas into simpler subformulae and constructing proof trees by refuting the negation of a statement, thus enabling transparent, traceable reasoning steps valuable for explainable formal reasoning tasks [Smu95, pp. 15ff, 60ff, 105ff].

Formal verification leverages these logical systems to mathematically prove the correctness of algorithms, protocols, or systems against specified properties. Techniques such as model checking and deductive verification allow symbolic reasoning systems to exhaustively explore possible states or construct rigorous proofs that a system meets critical safety, security, or functional requirements [SUM99, pp. 49ff, 67]. The deterministic nature of these approaches ensures conclusions necessarily follow from premises, providing mathematical certainty, which is an essential feature in domains requiring strict behavioral guarantees, where errors can have severe consequences [DP01, pp. 250f, 258f], [Kul+22, pp. 5:2f].

2.3.1.3 Knowledge Representation

Knowledge representation provides a structured means of encoding domain knowledge, enabling intelligent reasoning and efficient computation [DSS93, pp. 17f]. Common formats include semantic networks, where nodes represent concepts and edges define relationships, and frames, which encapsulate object attributes and behaviors with inheritance properties [Sow92, pp. 1493ff]. Concept hierarchies establish taxonomic relationships through specialization and generalization, allowing child concepts to inherit and extend properties from parent concepts [Bra83, pp. 30-32].

More advanced representations, such as ontologies, define comprehensive domain models with classes, relationships, attributes, and axioms specified in formal languages, supporting sophisticated reasoning about entities and constraints [Gru93, pp. 199f], [GOS09, pp. 13f]. These structures enable key symbolic AI operations, including subsumption checking, consistency verification, inference chain construction, and explanation generation, while providing human-readable access to encoded knowledge bases [Baa+03, pp. 13, 50ff, 65–68].

2.3.1.4 Rule-Based Systems

Rule-based systems are a fundamental symbolic AI approach that encode domain knowledge as explicit "if condition then action" statements for reasoning and decision-making [Hay85, pp. 921-923] (see Figure 2.8). These

systems use an inference engine to systematically apply production rules, which are formal representations of conditions and actions, to a working memory of current facts [RN09, pp. 546ff]. Expert systems extend this concept by organizing comprehensive rule sets from human experts into knowledge bases, enabling automated reasoning in fields such as medicine, engineering, and finance [Jac99, pp. 5-10, 28f].

Production rule systems employ pattern-matching algorithms to identify applicable rules based on current facts, executing actions that may update the fact base, trigger further rules, or generate conclusions [For82, pp. 18-20]. Their explicit rule tracing ensures transparency and explainability, making them valuable in domains requiring clear decision paths and supporting incremental knowledge refinement [Lug08, pp. 22-24], [Leo02, pp. 25-27].

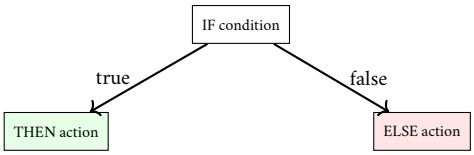


Figure 2.8: Basic structure of a rule in rule-based systems: IF-THEN-ELSE conditional logic.

2.3.2 Sub-Symbolic Methods

2.3.2.1 Large Language Models

LLMs are neural architectures based on the transformer design, which process and generate text using self-attention mechanisms. Multi-headed attention enables these models to capture relationships between all tokens in a sequence, allowing them to maintain coherence across long contexts. Learned linguistic patterns are encoded in billions of parameters distributed across multiple layers, capturing information from phonetics to advanced semantics [Bro+20, pp. 3-5], [Vas+17, pp. 6001-6004].

LLM Processing Stages and Key Concepts

Pre-Training Pre-training develops general language understanding by predicting the next token on massive text corpora, a process requiring substantial computational resources. Tokenization divides text into basic units, i.e., characters, subwords, or words, depending on the strategy [Bro+20, pp. 3-5, 6f], [Vas+17, pp. 6001-6004].

Fine-Tuning Fine-tuning adapts pre-trained models to specific tasks or domains using smaller, targeted datasets, updating only a subset of parameters to enhance task-specific performance while preserving general capabilities [EW25, p. 255:2f]. This process is further detailed in Section 2.3.2.2.

Few-Shot Learning Few-shot learning enables pre-trained models to generalize to new tasks from just a few examples provided in the context window, without parameter updates [Bro+20, pp. 6f].

Inference Inference is the deployment phase, where the model generates outputs from inputs without further training, typically requiring less computation than training but still demanding latency and throughput optimizations [EW25, pp. 255:2ff]. Several parameters can be configured

Table 2.2: Key inference parameters for LLMs (Adapted from: [McL25]).

Parameter	Description
Temperature	Controls randomness/diversity
Top_k	Limits sampling to top k tokens
Top_p	Nucleus sampling, cumulative prob.
Repetition Penalty	Penalizes repeated tokens
Max Tokens	Maximum output length
Frequency Penalty	Reduces repetition of common tokens
Presence Penalty	Encourages topic diversity
Stop Sequences	Strings that halt generation

to control the model's operation and output behavior, with the most commonly used parameters summarized in Table 2.2.

Prompt Engineering Prompt engineering shapes LLM behavior by providing instructions, examples, and context. Effective prompts use clear instructions and relevant context, often structured with templates that include placeholders for dynamic content, enabling systematic and predictable model behavior across tasks [Sah+25, pp. 1ff], [Wei+22, pp. 28f].

2.3.2.2 Fine-Tuning Language Models

While the basic concept involves adapting pre-trained models through continued training, the practical implementation of fine-tuning encompasses several strategic considerations crucial for specialized AI systems. Figure 2.9 illustrates how this process leverages transfer learning principles, enabling practitioners to customize general-purpose models for domain-specific applications without the computational costs of training from scratch [HR18, pp. 2f]. The following examines fine-tuning approaches utilized in the HAIMEDA application, alongside the benefits and limitations of creating specialized models through LLM fine-tuning.

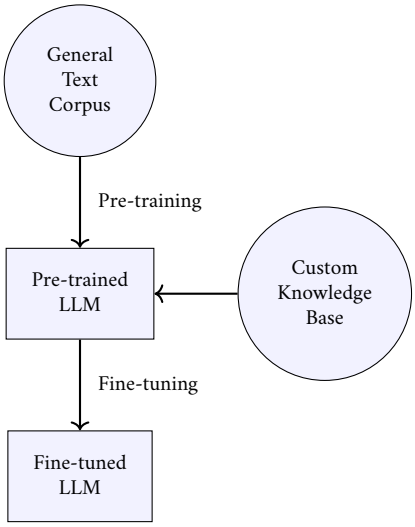


Figure 2.9: Fine-tuning workflow: pre-training on general, large-scale text data, then specialization with custom knowledge.

Fine-Tuning Approaches

Unsupervised Fine-Tuning Unsupervised fine-tuning, often called Domain-Adaptive Pre-Training (DAPT), involves continuing model training on domain-specific corpora without explicit task labels or human feedback. This approach uses self-supervised objectives similar to initial pre-training, allowing models to adapt to specialized language patterns, terminology, and knowledge through exposure to representative texts. DAPT proves particularly valuable when domain language differs substantially from general text in vocabulary and discourse patterns, as demonstrated by improvements in both high- and low-resource settings [Gur+20, pp. 8343ff].

Supervised Fine-Tuning Supervised Fine-Tuning (SFT) incorporates explicit human guidance through labeled examples that demonstrate desired outputs for specific inputs, creating a direct path to task-specific optimization. This approach typically employs datasets consisting of input-output pairs that exemplify target behavior, often constructed through human annotation or curation of existing domain resources [Ouy+22, pp. 4-6]. Within SFT, Instruction Tuning (IT) represents a powerful technique where models learn to follow explicit directions through examples of instruction-response pairs, enabling them to understand and execute tasks described in natural language [Wei+22, pp. 1f]. Direct preference optimization further extends this paradigm by incorporating comparative judgments between alternative outputs, allowing models to learn from relative quality assessments rather than single 'gold standard' examples [Ouy+22, pp. 2f].

Parameter-Efficient Fine-Tuning Parameter-Efficient Fine-Tuning (PEFT) techniques address the computational and storage challenges of full model fine-tuning by modifying only a small subset of model parameters while keeping most weights frozen. This approach substantially reduces computational requirements while maintaining comparable performance [Hu+21, pp. 1f, 5f]. Key methods include Low-Rank Adaptation (LoRA), which injects trainable rank decomposition matrices into transformer layers to modify usually less than 1 % of parameters while enabling efficient domain adaptation, and adapter modules, which insert small bottleneck networks between transformer layers. Another approach is selective layer fine-tuning, where only some specific layers of the network are updated. These techniques offer different trade-offs between efficiency and performance, with LoRA and adapters achieving results comparable to full fine-tuning while requiring orders of magnitude fewer trainable parameters, and enabling specialized model versions to be easily stored, shared, and swapped [Hu+21, pp. 1-3, 5], [Hou+19, pp. 2790-2792].

Benefits

Fine-tuning creates specialized models that offer several advantages for specific use cases. Domain-specific models demonstrate enhanced performance on tasks within their target domain, achieving higher accuracy and relevance than general models, as continued pre-training on in-domain data leads to consistent performance gains [Gur+20, pp. 8342f]. Additionally, specialized models can be optimized for task-specific objectives beyond general language modeling, such as enhancing factual accuracy in educational contexts or maintaining stylistic consistency in content generation. They can also be aligned with organizational values, policies, and ethical guidelines, reducing inappropriate outputs in sensitive domains [Ouy+22, pp. 7f].

Limitations

Despite these benefits, fine-tuned models face important limitations. Most notably, specialization often comes at the cost of generalization ability, i.e., models optimized for specific domains may perform poorly on tasks outside their training focus [Dod+20, pp. 1f]. This creates a trade-off between specialized excellence and versatile applicability. Fine-tuned models may also amplify existing biases in domain-specific data, potentially reinforcing problematic patterns present in specialized corpora [Ron+21, pp. 1ff]. Moreover, continuous fine-tuning on task-specific datasets risks catastrophic forgetting, where previously acquired general knowledge deteriorates as the model overspecializes [HR18, pp. 1f, 9].

2.3.2.3 Model Optimization and Deployment Techniques

Quantization

Quantization reduces the numerical precision of model parameters and computations, allowing models to operate with lower memory and computational requirements in resource-constrained environments [Det+22, pp. 1f, 10]. Post-training quantization applies this precision reduction after fine-tuning, converting 32-bit floating-point parameters to lower precision formats like 16-bit (FP16/BF16), 8-bit integers (INT8), or even 4-bit representations [Zaf+19, pp. 36, 38], while quantization-aware training incorporates these effects during the fine-tuning process itself to minimize performance degradation [Che+25, pp. 1f] (see Figure 2.10). This technique often complements other efficiency methods like knowledge distillation, where smaller models learn from larger teacher models, and pruning, which removes redundant parameters, collectively enabling the practical deployment of specialized language models across diverse computing environments [Mov+22, pp. 2861f].

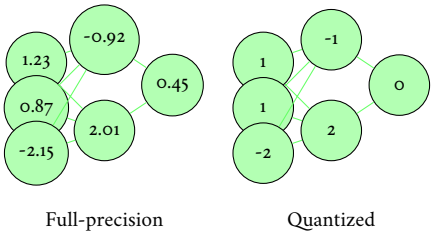


Figure 2.10: Illustration of quantization: floating-point weights (left) are mapped to lower-precision integer values (right), reducing memory and computation requirements.

Model Weight Interpolation and Merging

Model weight merging techniques combine parameters from multiple neural networks or adapters into a single consolidated model, preserving capabilities while improving inference efficiency. More broadly, model merging combines multiple trained neural networks at the parameter level, unlike ensemble methods that run models in parallel, with tools like MergeKit facilitating this process [God+24, pp. 477-479]. In contrast, adapter merging integrates specialized fine-tuning adaptations directly into a base model's parameter space, particularly valuable for parameter-efficient methods where adaptations exist as distinct components [He+23, pp. 1ff].

Several merging approaches exist: linear weight averaging combines parameters through weighted sums, while more sophisticated methods like Spherical Linear Interpolation (SLERP) treat weight spaces as hyperspheres and perform geometrically-aware interpolation along geodesic paths, often yielding better results than naive averaging. The resulting consolidated models preserve both foundation capabilities and domain-specific knowledge without the computational overhead of adapter mechanisms or ensemble approaches during inference [PJA24, pp. 7, 13].

2.3.2.4 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) enhances language model capabilities by integrating external knowledge retrieval with generative text processing. Unlike traditional models that rely solely on parametric knowledge, RAG systems dynamically retrieve relevant information from external sources before generating responses, improving accuracy and reducing hallucinations [Lew+20, pp. 1, 5f, 9] (see Figure 2.11). The following subsections explore RAG's core components, advanced retrieval strategies, and specific benefits for data-sensitive applications.

Core Components

RAG systems usually comprise three key components. First, an embedding model transforms text documents into high-dimensional vector representations capturing semantic meaning. This involves text normalization and strategic chunking, segmenting documents into manageable units before embedding with pre-trained models. Fixed-size token chunking is common, with smaller chunks (64–128 tokens) optimal for concise, fact-based answers, and larger chunks (512–1024 tokens) better for content requiring broader contextual understanding [Bha+25, pp. 1f, 4f].

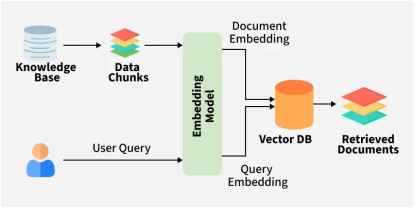


Figure 2.11: Schematic overview of a RAG system: user queries and knowledge base documents are embedded into vectors, enabling relevant documents to be retrieved from a vector database for context-aware generation (Source: https://www.geeksforgeeks.org/nlp/what-is-retrieval-augmented-generation-rag/).

Second, these vectors are stored in specialized vector databases optimized for efficient similarity search operations [Gao+24, p. 3]. Finally, during inference, a retriever component identifies relevant documents by comparing embedded query vectors with stored document vectors, providing this context to the generator component, typically a language model, that produces responses based on both the query and retrieved information [Lew+20, pp. 2-4].

Advanced Retrieval Strategies

During retrieval, user queries undergo the same embedding process for vector comparison, primarily using cosine similarity to identify relevant content [Bha+25, p. 3]. Hybrid approaches combine multiple retrieval strategies, as demonstrated by HybridRAG, which integrates vector similarity with knowledge graph-based retrieval and metadata filtering to improve faithfulness, answer relevance, and context recall for complex domain-specific documents [Sar+24, pp. 608-610, 614]. Retrieved documents are then concatenated with the input query as additional context for the language model's generation process [Shu+21, p. 3786].

Benefits for Data-Sensitive Applications

RAG architectures maintain clear separation between the language model and domain-specific content, allowing document updates without retraining while supporting information governance through source tracking [Lew+20, pp. 1f, 9]. Additionally, RAG systems can process sensitive information locally without external transmission [WC24, pp. 2765f]. This combination of retrieval with generation provides detailed, context-specific responses while maintaining the flexibility and natural language capabilities of modern language models [Lew+20, pp. 1f, 5f].

2.3.2.5 Neural Text Embeddings and Semantic Similarity

Neural text embeddings transform words and sentences into dense vectors, where similar meanings are positioned close together in high-dimensional space. Unlike simple keyword matching, these representations enable

mathematical processing of meaning, where distance metrics quantify semantic relatedness [Mik+13, pp. 1f, 5f, 10] (see Figure 2.12). Cosine similarity serves as the primary metric, measuring angular similarity between vectors to highlight semantic relationships preserved in the embedding space. Alternative metrics offer complementary insights: Euclidean distance captures absolute magnitude differences, Manhattan distance measures cumulative dimensional differences, Mahalanobis distance accounts for feature correlations in anisotropic distributions, while Term Frequency-Inverse Document Frequency (TF-IDF) weighting emphasizes rare but informative terms [WD20, pp. 2f], [FK03, p. 118]. This vector-based approach enables models to detect paraphrase equivalence and semantic similarity despite varied phrasing, capabilities that literal string matching cannot achieve [Gio+21, pp. 879-881].

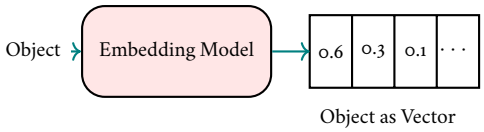


Figure 2.12: A single object is transformed by an embedding model into a vector representation.

2.3.3 Cross-Cutting Concerns in Data-Sensitive Domains

Data-sensitivity and information integrity are foundational concepts central to this work, underpinning both the proposed framework and the HAIMEDA application. The following subsections examine these concepts in detail.

2.3.3.1 Data Sensitivity Handling

Data sensitivity handling encompasses methodologies for processing confidential information while preserving privacy, security, and regulatory compliance. Local processing approaches prioritize on-device or on-premise operations, minimizing exposure risks while working within local hardware constraints. This local-first paradigm designs systems to operate primarily within secure, controlled environments [Kle+19, pp. 160f].

When distributed processing becomes necessary, federated learning enables model training across decentralized data sources without centralizing sensitive information. These architectures often incorporate differential privacy mechanisms that mathematically guarantee individual privacy by introducing calibrated noise into computations while preserving statistical utility [McM+17, pp. 1273ff].

2.3.3.2 Information Integrity Verification

Information integrity verification encompasses methodologies for ensuring factual accuracy, logical consistency, and trustworthiness across AI system interactions, critical in domains where misinformation could have significant consequences [Kas+21, pp. 8849f]. Verification strategies operate at various processing stages: pre-processing with rule-based systems and formal constraints [Rue+23, pp. 618ff], during inference through neuro-symbolic reasoning networks [Lam+21, p. 4878], and post-processing by comparing outputs against reference knowledge [Sar+21, pp. 199f].

These approaches span a methodological spectrum, with formal verification techniques providing mathematical certainty through tableaux-based logical proof systems and constraint satisfaction [Smu95, pp. 105ff], [SUM99, p. 67], while statistical methods deliver probabilistic confidence via similarity metrics and entity recognition [Lew+20, pp. 3f]. Effective systems typically employ multi-stage verification pipelines that first identify key entities, then verify relationships between them, and finally assess overall semantic consistency using both symbolic rules and neural models [Sar+21, pp. 198ff], [Rue+23, pp. 618ff].

This hybrid approach leverages the strengths of different AI paradigms, with symbolic systems verifying precise factual elements and logical consistency, and neural approaches handling semantic equivalence across varied expressions [GL23, pp. 12397f]. These verification methodologies form a foundation that ensures the integrity of information in AI systems where accuracy and reliability are paramount [Kas+21, pp. 8849f], [SWS22, pp. 1ff].

3 Methodology

This chapter presents the methodological foundation for developing both the framework and its paradigm application, HAIMEDA. Section 3.1 introduces Action Design Research (ADR) as the research method, establishing why this approach is particularly suited for simultaneously addressing theoretical framework development and practical implementation. Section 3.2 then details the application of ADR throughout the research process, including research questions derived from the gaps identified in Section 2.1.5, iterative building and evaluation cycles, and the formalization of knowledge gained from this process.

3.1 Action Design Research as Methodological Framework

This research requires a methodological approach that integrates theoretical framework development with practical artifact implementation, where artifacts are concrete design outputs such as frameworks, prototypes, or systems. This integration enables mutual refinement throughout the research process.

Developing a comprehensive hybrid AI framework alongside a functional medical device assessment system demands structured guidance for artifact creation while validating design activities as knowledge production. This positions the work within the design paradigm, where theoretical understanding and practical solutions evolve together through creating and evaluating artifacts in their intended settings [Hev+04, pp. 79, 81, 88f]. As illustrated in Figure 3.1, this methodology follows an iterative cycle that alternates between planning, implementation, analysis, and refinement.

Figure 3.1: The action research cycle, with iterative stages of planning, action, analysis, and conclusion, provides a conceptual foundation for many design-oriented research methodologies (Adapted from: https://www.scribbr.com/methodology/action-research/).

3.1.1 Design-Oriented Research Background

While design science research in Information Systems, established through seminal works by Simon [Sim96], Walls et al. [WWE92], and Hevner et al. [Hev+04], provides a foundation for creating and evaluating Information Technology (IT) artifacts to solve organizational problems, its traditional approaches often artificially separate artifact building from evaluation in authentic contexts [Sei+11, p. 39]. Traditional survey-based or experimental methods cannot adequately support the dual objectives of knowledge creation and artifact development that this research requires [Sei+11, pp.

38-40]. Given this project's fundamentally interconnected goals where framework and implementation must evolve in tandem, a more integrated approach is required, one that recognizes knowledge emergence through active intervention in real-world settings [Col+05, pp. 332-334].

3.1.2 Overview of Action Design Research

ADR provides a methodological approach for developing prescriptive design knowledge through the simultaneous building and evaluation of integrated IT artifacts within organizational environments [Sei+11, pp. 38, 41f]. Unlike approaches that treat design and deployment as separate phases, ADR recognizes them as interrelated concurrent activities [Pur+13, pp. 76f]. Central to ADR is the concept of "ensemble artifacts," where IT systems emerge through dynamic interaction between technological components and organizational contexts [Sei+11, p. 39]. In the context of this research, this perspective is especially valuable when developing hybrid AI systems that bridge technical implementation and organizational processes.

ADR consists of four interrelated stages as defined by Sein et al. [Sei+11, pp. 41ff]:

1. **Problem Formulation**: Anchors research on a practice-inspired problem while establishing scope, identifying stakeholders, and ensuring theoretical grounding.
2. **Building, Intervention, and Evaluation (BIE)**: Interweaves artifact building, organizational intervention, and evaluation activities in cycles rather than sequential phases.
3. **Reflection and Learning**: Creates knowledge through continuous reflection on the problem, theories, and emerging artifact throughout the process.
4. **Formalization of Learning**: Abstracts the specific solution into generalizable concepts and design principles.

What distinguishes ADR from other design science methodologies is its emphasis on iterative cycles of building, intervention, and evaluation that operate concurrently rather than sequentially [Ma19, pp. 8, 14]. These iterations allow the researchers to continuously refine both the artifact and the underlying design principles based on ongoing learning [Pur+13, pp. 76f].

ADR is guided by seven interrelated principles established by Sein et al. [Sei+11, pp. 40-44], which provide methodological guidance while maintaining flexibility, as illustrated in Figure 3.2:

1. **Practice-Inspired Research**: Views field problems as knowledge-creation opportunities, ensuring research addresses real organizational challenges rather than hypothetical scenarios.
2. **Theory-Ingrained Artifact**: Ensures artifacts embody theoretical precursors, leveraging prior research to inform design decisions rather than building solely from practice.

3. **Reciprocal Shaping**: Recognizes the mutual influence between the emerging IT artifact and its organizational context, where each shapes the other during the development process.
4. **Mutually Influential Roles**: Emphasizes cross-learning between researchers and practitioners, acknowledging the value of different expertise in shaping the artifact.
5. **Authentic and Concurrent Evaluation**: Intertwines evaluation with building rather than treating them as separate activities, enabling continuous refinement based on real-world feedback.
6. **Guided Emergence**: Allows the artifact's final form to emerge through interaction between design and use rather than being fully predetermined, balancing initial design with adaptation.
7. **Generalized Outcomes**: Focuses on abstracting learning beyond the specific instance to address broader classes of problems through design principles and artifacts.

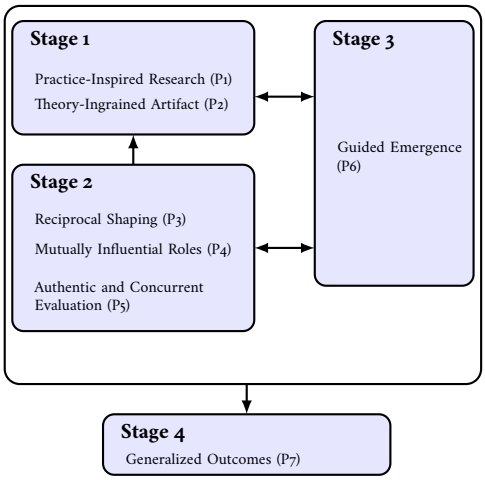


Figure 3.2: The four-stage Action Design Research process and its guiding principles (Adapted from [Sei+11, Fig. 1]).

The BIE stage can follow two distinct forms depending on the primary source of innovation. In IT-dominant BIE, the initial design is created by researchers based on theoretical premises and then refined through organizational intervention cycles. Conversely, organization-dominant BIE begins with organizational intervention where the artifact emerges through practitioner-led processes [Sei+11, pp. 42f]. For hybrid AI systems development, elements of both forms may be necessary, with technical innovations requiring IT-dominant phases while organizational integration demands organization-dominant considerations.

3.1.3 Methodological Fit of ADR

ADR aligns particularly well with this research due to the reciprocal nature of the two primary objectives, framework development and system implementation, which require concurrent theoretical development and practical application. ADR's emphasis on mutual shaping of theory and practice enables continuous refinement of the framework based on implementation insights, while implementation decisions are guided by evolving theoretical understanding [Sei+11, pp. 42-44].

The development of hybrid AI systems represents a complex socio-technical challenge where technical components must integrate with organizational processes. ADR's concept of ensemble artifacts addresses this by recognizing that effective AI systems emerge through integration with organizational practices and human expertise [OI01, pp. 126-128]. Additionally, the medical device assessment domain's requirements for data sensitivity and information integrity benefit from ADR's authentic evaluation principle, which incorporates these constraints throughout development rather than only in final evaluation phases [Sei+11, p. 39].

Finally, ADR's focus on generating generalizable design knowledge through formalization of learning directly supports the objective of creating a

broadly applicable framework, enabling design principles that extend beyond specific implementations to inform hybrid AI development across various contexts [Sei+11, p. 44].

3.2 ADR Process and Application in This Study

The following subsections describe how this research applies the four stages of the ADR methodology. Section 3.2.1 begins with problem formulation, formalizing the research objectives introduced in Chapter 1 and detailing requirements specific to the medical device assessment domain. Section 3.2.2 describes the building, intervention, and evaluation activities conducted to develop the framework, its toolkit, and the paradigm. Section 3.2.3 briefly outlines the reflection and formalization stages, which are explored in detail in Chapters 6 and 7.

3.2.1 Stage 1: Problem Formulation

Problem Formulation, the first stage of ADR methodology, focuses on identifying and framing the research challenge as a class of problems amenable to intervention. In alignment with Sein et al.'s [Sei+11, pp. 40f] principles of practice-inspired research and theory-ingrained artifacts, this stage anchors the study in both practical relevance and theoretical foundations by identifying research problems and objectives, establishing technical boundaries for software artifacts, and examining stakeholders and contextual factors within the medical device assessment domain.

3.2.1.1 Research Problem, Objectives, and Questions

As identified in Section 2.1.5, there exists a critical gap in the current landscape of AI development methodologies, specifically, the absence of a comprehensive framework that integrates hybrid AI approaches within data-sensitive domains. From the five gaps identified in Section 2.1.5, this research addresses three critical dimensions that represent addressable limitations through framework development and practical implementation, whereas the remaining gaps concerning literature focus on neuro-symbolic approaches and limited cross-domain coverage require broader community-level changes and extensive multi-sector validation beyond this research's scope:

- Absence of lifecycle-spanning, project-oriented development guidance for hybrid AI systems
- Insufficient integration between theoretical constructs and practical tooling
- Limited support for functional programming paradigms in the hybrid AI development domain

In response to these identified gaps, this research pursues two primary objectives that evolve through mutual development:

Framework Development: The first objective is to create a comprehensive framework for hybrid AI applications in data-sensitive domains, providing structured development guidance, practical tools, and implementation examples for integrating symbolic and sub-symbolic AI approaches.

Paradigm Implementation: The second objective is to develop the HAIMEDA application, a functional hybrid AI system that serves as both proof of concept and refinement mechanism for the framework, addressing domain-specific requirements for data confidentiality and information integrity.

These objectives are inherently interconnected, with the framework benefiting from empirical insights gained through practical implementation, while implementation is guided by the emerging framework principles. This bidirectional relationship exemplifies ADR's reciprocal shaping principle, allowing both artifacts to evolve simultaneously rather than sequentially. From these objectives, the following research questions are derived that guide this study:

Primary Research Question 1:

RQ1 *How can a comprehensive development framework be designed to effectively guide the creation of hybrid AI systems?*

Secondary Questions regarding RQ1:

RQ1.1 *What specific guidelines, tools, and best practices are essential for developing hybrid AI systems in data-sensitive domains?*

RQ1.2 *How can the framework balance flexibility across different application domains with sufficient structure to guide implementation?*

RQ1.3 *What is the relationship between theory development and practical implementation in the evolution of a hybrid AI framework?*

RQ1.4 *To what extent can a framework developed alongside a specific implementation be generalized to diverse hybrid AI applications?*

Primary Research Question 2:

RQ2 *How can a hybrid AI system be implemented to support medical device damage assessment while ensuring data integrity and confidentiality?*

Secondary Questions regarding RQ2:

RQ2.1 *Which architectural design strategies enable effective integration of different AI techniques to maximize their complementary strengths and reduce their individual limitations?*

RQ2.2 *What measures must be taken during the development and deployment of a hybrid AI system to ensure data confidentiality and maintain information integrity?*

RQ 2.3 *To what extent can a hybrid AI system for medical device assessment achieve accuracy levels comparable to human experts?*

These research questions directly inform both the framework development and the HAIMEDA application implementation, with a comprehensive analysis of their fulfillment presented in Chapter 7.

3.2.1.2 Technical Solution Space and Architectural Boundaries

This research employs `Elixir` as the primary implementation language with targeted `Python` integration for specialized AI components. This architectural approach directly addresses RQ 2.1 by leveraging Elixir's functional programming paradigm to create verifiable, transparent data processing pipelines while maintaining access to Python's mature AI ecosystem.

Elixir's functional nature, with immutable data structures and pattern matching capabilities, provides an ideal foundation for implementing hybrid AI systems where explainability and traceability are essential [Efe24], [GB85, p. 438]. The language's strong fault tolerance through the Bogdan/Björn's Erlang Abstract Machine (BEAM) virtual machine and supervision trees offers reliability guarantees critical for systems operating in data sensitive domains. Through integration mechanisms such as the `:erlport` library [Dmi16], Python-based ML models are selectively invoked from Elixir when needed, creating a cohesive architecture that connects different AI paradigms.

This technical solution space applies ADR's theory-ingrained artifact principle [Sei+11, p. 40] by embedding diverse theoretical foundations into the research's technological foundation, enabling coherent progression of both framework development and paradigm implementation. Beyond functional programming, additional foundations include Beck's agile methodologies [Bec+01], Parnas' modularization principles [Par72], Jorgensen's testing methodologies [Jor14], and Sculley's work on technical debt in ML systems [Scu+15], all collectively informing the framework principles presented in Chapter 4.

3.2.1.3 Context and Stakeholder Definition in the Medical Device Assessment Domain

The medical device assessment domain provided a practice-inspired context for developing and testing the hybrid AI framework, exemplifying ADR's principle of practice-inspired research [Sei+11, p. 40]. Through multiple interviews with medical device expert Walter Sigloch [Sig], several key domain characteristics were identified that directly informed the problem formulation:

- **Data Sensitivity**: Assessment reports contain confidential information including patient data, device specifications, and potential liability details, requiring strict data protection measures.

- **Information Integrity**: Reports must maintain factual accuracy and logical coherence as they may be used in legal or insurance contexts, with consequences for errors or inconsistencies.
- **Complex Stakeholder Environment**: Multiple parties are involved including contracting entities (authorities, insurances), defending parties, information-providing parties (manufacturers, witnesses), and expert consultants.
- **Research-Intensive Process**: Approximately 30 % of the assessment work involves research activities including reviewing technical documentation, interviewing witnesses, and examining prior similar cases.

These domain characteristics established specific constraints for both the framework development and system implementation, particularly emphasizing local data processing, verification mechanisms, and information consistency. A complete description of the medical device assessment domain is provided in Appendix A.1, including survey types, stakeholder roles, data sensitivity considerations, and the typical survey lifecycle.

Initial implementation attempts using primarily sub-symbolic approaches with LLMs quickly revealed practical limitations that directly shaped the framework's evolution. The medical device assessment domain presented complex decision logic with numerous conditional factors affecting how context information should be provided to LLM inference tasks, which is logic that needed to be explicitly represented through rule-based systems rather than implicitly learned. Additionally, the limited availability of domain-specific reports proved insufficient for adequately training ML models to capture nuanced reasoning patterns, with full reports being too long for local fine-tuning and too few in number, necessitating division into chapters. These practical challenges prompted the shift toward a hybrid architecture capable of both structured rule-based reasoning and flexible pattern recognition, with each iteration of implementation providing insights that refined the framework's design.

While practice-inspired insights guided the framework's overall direction, its design was equally shaped by established theoretical foundations, embodying ADR's theory-ingrained artifact principle. The framework draws on Hughes' functional programming theory [Hug89] and Hewitt's actor model [HBS73] as implemented through Elixir and the BEAM virtual machine [LE17, p. 16f]. For data security, both Kleppmann's "local-first software" principles [Kle+19] and Cavoukian's "privacy by design" framework [Cav09] informed architectural decisions for handling sensitive data. Information consistency approaches drew from explainable AI research by Arrieta et al. [Arr+20], formal verification methods [Rij01], and Kautz's neuro-symbolic integration work [Kau22].

This bidirectional influence between practical domain challenges and theoretical foundations exemplifies how ADR's practice-inspired research and theory-ingrained artifact principles worked together to shape a framework capable of addressing the complex requirements of data-sensitive domains while maintaining theoretical rigor.

3.2.2 Stage 2: Building, Intervention, and Evaluation

This section describes the iterative BIE cycles of Sein et al. [Sei+11, pp. 42f] conducted to develop the framework, toolkit, and paradigm artifacts. It begins by introducing these artifacts as an integrated ecosystem, then details their development cycles, and finally examines the context of intervention with domain stakeholders.

3.2.2.1 Overview of the ADR Artifacts

This research developed three interrelated artifacts following ADR's ensemble view: the framework, toolkit, and paradigm. Rather than functioning independently, these components form an integrated ecosystem with distinct levels of abstraction and mutual interdependence, as illustrated in Figure 3.3. The framework provides the conceptual foundation and design principles, the toolkit offers concrete implementation resources that underpin these concepts, and the paradigm applies these tools to develop the hybrid AI application for medical device assessment. This multi-level structure aligns with ADR's dual objectives of solving specific problems while developing generalizable knowledge.

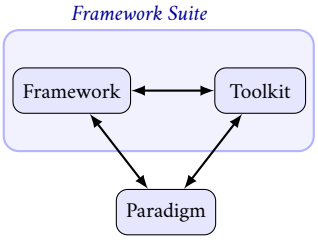


Figure 3.3: Relationship between framework, toolkit, and paradigm artifacts.

Collectively, these artifacts address the research questions established in Section 3.2.1.1. The framework and toolkit primarily address RQ 1 by providing generalizable design principles for hybrid AI development, while the paradigm implementation addresses RQ 2 by demonstrating these principles in the medical device assessment domain.

Framework Artifact

The framework artifact serves as a methodological guide for developing hybrid AI systems in data sensitive domains, applying ADR's theory ingrained principle to create generalizable knowledge beyond the specific implementation. Conceived as an abstract architecture and process model, it guides practitioners through the complex integration of symbolic and sub-symbolic AI techniques.

Following ADR's guided emergence principle, the framework evolved through iterative BIE cycles rather than being fully predetermined. Initial boundaries included conceptual components for data preparation, hybrid system integration, information integrity verification, and secure deployment strategies. These components were refined as development progressed, based on practical challenges encountered during paradigm implementation, with necessary tools incorporated into the framework's toolkit according to implementation needs.

The framework deliberately maintained sufficient abstraction to enable application across various domains beyond medical device assessment, supporting ADR's generalized outcomes principle by addressing a class of problems rather than a single instance. Its structure balances prescriptive guidance with flexibility, allowing practitioners to adapt components to specific contexts while preserving core principles of hybrid AI integration and data sensitivity handling.

Toolkit Artifact

The toolkit artifact comprises concrete tools, libraries, and implementation examples that operationalize the framework's conceptual guidance. It emerged through practical implementation needs encountered during paradigm development, addressing the gap between abstract principles and practical implementation. Developed iteratively alongside the paradigm application following ADR's authentic and concurrent evaluation principle, these tools demonstrate how the framework's concepts translate into working code.

The toolkit features a modular architecture centered around six key functional areas:

- **Data preparation** tools for fine-tuning language models and creating structured knowledge bases
- **Data sanitization** utilities for handling sensitive information in compliance with privacy requirements
- **Information extraction** techniques combining rule-based systems with statistical approaches
- **Dataset building** components preparing instructions for both supervised and unsupervised learning
- **Model fine-tuning** tools for fine-tuning and merging
- **Model optimization** tools for quantization and integration to enhance accessibility

As a result, the toolkit provides a foundation that can be readily extended or adapted for future hybrid AI projects in other domains.

Paradigm Artifact

The paradigm artifact represents the complete medical device assessment application that instantiates both the framework and toolkit in a real-world context, serving as the primary vehicle for intervention and evaluation within the ADR process. It implements a modular ecosystem of interconnected components organized around a functional data pipeline architecture, with a central `MainController` module at its core orchestrating specialized components including pre-processing validation, LLM integration, and post-processing verification. This architectural approach creates clear boundaries between components while maintaining system-level coherence, allowing individual elements to evolve independently through BIE cycles.

The paradigm demonstrated how framework principles could be applied in practice while addressing RQ 2 through multiple aspects of its implementation. It fulfilled RQ 2.1 through strategic integration of symbolic and sub-symbolic components, satisfied data confidentiality requirements in RQ 2.2 through its local-first deployment approach, and provided a concrete testbed for RQ 2.3 by enabling comparison between

system-generated and human-expert assessments. This bidirectional relationship between theoretical framework, practical toolkit, and domain implementation exemplifies ADR's mutually influential roles principle.

3.2.2.2 Development Iterations and Design Decisions

The development process followed a structured iterative approach aligned with ADR's BIE principles from November 2024 to June 2025. The process encompassed three primary iteration categories: framework conceptualization, toolkit development, and paradigm implementation, as detailed in Appendix A.2. Each category progressed through multiple cycles, supporting ADR's guided emergence principle by allowing artifacts to evolve through practical implementation insights.

Technical constraints significantly shaped design decisions throughout development, as illustrated in Table 3.1. The requirement for local data processing necessitated aligning the development environment with the medical device expert's operational context, ensuring immediate deployability. Hardware specifications constrained model selection to 8B parameter models with quantization and limited fine-tuning to LoRA adapters due to Video Random Access Memory (VRAM) constraints. Specifically, development was conducted on a Windows 11 Pro system equipped with an AMD Ryzen 9 5900X CPU, 64 GB DDR4 RAM, and an NVIDIA RTX 4090 GPU with 24 GB VRAM. Windows was chosen as the Operating System (OS) to develop on the same platform that the surveyor would use, mitigating potential compatibility issues during deployment.

Development began with a prototype system *Hybrid AI Task Router (HATR)* that validated key architectural concepts before full implementation. HATR implemented the initial `MainController` orchestration approach using the LangChain framework for prompt engineering and incorporated rule-based validation using tableaux logic. This prototyping aligned with ADR's risk mitigation approach, validating core patterns before committing resources to full implementation.

Model selection followed its own iterative process through three distinct BIE cycles. Initial efforts utilized models from the LeoLM team, which specializes in German language model development, particularly *Mistral 7B*[1] and *Llama 2 13B*[2] variants, before ultimately selecting *Llama 3 8B IT*[3] from DiscoResearch, a research collective focused on open-source German AI models, as the foundation due to its superior performance on domain-specific tasks, better context handling, and improved German language capabilities.

Throughout all cycles, decision-making followed a consistent methodology of requirements formulation, solution implementation, effectiveness testing, and refinement. Documentation through version-controlled repositories, structured decision logs, and expert feedback notes ensured traceability between implementation decisions and requirements, supporting concurrent evaluation and later knowledge formalization.

Table 3.1: Overview of the hardware, software, and model constraints guiding system development.

Component	Spec.
Dev. Environment	Windows 11 Pro
CPU	AMD Ryzen 9 5900X
	(12-Core, 24 Threads)
RAM	64 GB (DDR4)
GPU	NVIDIA RTX 4090
	(24 GB VRAM)
LLM Constraints	8B Models with
	Quantization
Fine-Tuning	LoRA Adapters
Models	LeoLM Mistral 7B
	LeoLM Llama 2 13B
	DiscoLeo Llama 3 8B IT
Integration	`:erlport` [Dmi16]
	for Elixir-Python Bridge

Note: '*X*B' denotes *X* billion model parameters (e.g., 8B = 8 billion). The model in bold was chosen as the standard for all subsequent development.

1: https://huggingface.co/jphme/em_german_leo_mistral
2: https://huggingface.co/jphme/em_german_13b_leo_gguf
3: https://huggingface.co/DiscoResearch/Llama3-DiscoLeo-Instruct-8B-v0.1

3.2.2.3 Expert Involvement

This research incorporated continuous expert feedback throughout the development process, with Walter Sigloch, the medical device surveyor described in Appendix A.1, serving as the primary expert stakeholder. His involvement followed a structured pattern, beginning with the identification of core system requirements such as information integrity, data sensitivity, and local processing needs. These requirements directly shaped both framework design principles and implementation decisions. Following each major module implementation, structured evaluation sessions assessed functionality and workflow alignment, with expert feedback directly informing refinements, as exemplified by improvements to rule-based verification logic following preprocessing validation system review.

What distinguishes this collaboration from typical AI development projects is the expert's sustained engagement in both formative and summative evaluation, including hands-on testing with actual survey report drafts. This approach ensured that the evolving framework and application remained grounded in authentic domain practices and professional standards. By integrating expert insights at every stage, the research achieved a level of practical relevance and adaptability uncommon in purely technical AI projects, resulting in a solution that is not only technically robust but also operationally viable and aligned with legal, ethical, and stakeholder requirements.

3.2.3 Stages 3 & 4: Reflection, Learning and Formalization

Following ADR methodology, Stages 3 and 4 transform practical implementation experiences into formalized, generalizable knowledge. While detailed outcomes appear in Chapters 6 and 7, this section outlines the methodological approach used to structure reflection, learning, and knowledge formalization throughout the research process as well as providing an overview of the data collection and evaluation techniques.

3.2.3.1 Reflection and Learning Methodology

In accordance with Sein et al.'s [Sei+11, p. 44] reflection and learning stage, the research implemented a systematic approach to capture insights emerging from BIE cycles. This involved:

- **Structured Documentation**: Maintaining development logs that recorded not only technical decisions but also their underlying rationales and connection to framework principles
- **Critical Incident Analysis**: Identifying key challenges, unexpected outcomes, and successful implementations that provided significant learning opportunities
- **Comparative Assessment**: Evaluating differences between theoretical expectations and practical outcomes to highlight areas where framework refinement was necessary

These reflection activities were not isolated to a distinct phase after implementation but occurred continuously throughout the development process, allowing theoretical understanding and practical solutions to evolve simultaneously.

3.2.3.2 Formalization of Learning Approach

The formalization process followed a systematic methodology to transform specific implementation findings into generalizable design principles:

- **Pattern Recognition**: Identifying recurring successful approaches and challenges across different modules and development phases
- **Cross-Domain Abstraction**: Separating domain-specific implementation details from generalizable hybrid AI patterns
- **Principle Extraction**: Deriving explicit design guidelines that transcend the specific medical device assessment context
- **Theoretical Integration**: Connecting empirical findings with established theories to strengthen conceptual foundations

This methodological approach ensured that knowledge formalization followed a rigorous process rather than relying on post-hoc rationalization.

3.2.3.3 Data Collection and Analysis

This research employed a multi-method approach combining quantitative and qualitative techniques to comprehensively evaluate the developed artifacts. Quantitative methods included performance measurements for system resource utilization, model inference and verification tasks, training/validation loss curves across fine-tuning epochs, and standardized usability metrics via Brooke's System Usability Scale (SUS) [Bro95]. Productivity was assessed through comparative task analysis using Sauro & Dumas' framework [SD09], which enabled time-to-completion comparisons between traditional and AI-assisted workflows. Qualitative data collection utilized expert think-aloud protocols as described by Boren & Ramey [BR00], where the medical device surveyor verbalized thoughts during system interaction, providing contextual insights into usability challenges and workflow integration. Content quality was evaluated using structured frameworks with multi-dimensional rating scales for factual accuracy, structural coherence, and language appropriateness. Triangulating these quantitative metrics with qualitative expert assessments produced a more complete understanding of both technical effectiveness and practical utility, directly informing the critical assessment sections in Chapter 7.

4 Framework for Developing Hybrid AI Systems in Data-Sensitive Domains

This chapter presents the comprehensive development framework, beginning with foundational principles and architectural patterns in Section 4.1 that shape the entire framework. Section 4.2 provides a systematic approach to AI paradigm selection across different implementation scenarios. Building upon this paradigm selection, Section 4.3 details the development lifecycle, where the core principles from Section 4.1 are particularly reflected in the implementation guidance organized into clearly defined phases. The chapter concludes with Section 4.4, which presents toolkit components developed alongside the paradigm application, providing practitioners with concrete resources for hybrid AI system development.

4.1 Core Principles and Architectural Patterns

The framework is built upon five foundational principles that guide the development of hybrid AI systems in data-sensitive domains and are detailed below.

The primary principle is modularization, advocating for standardized and reusable data pipelines that interconnect different modules in a unified manner. While modules can be internally heterogeneous, using various programming languages, libraries, APIs, and frameworks, the orchestration layer must use a consistent programming language with unified interfaces across all modules. This approach, inspired by Parnas [Par72] and Heineman et al. [HC01], provides flexibility within modules while maintaining system-wide coherence.

Central to this methodology is a strong emphasis on agile development strategies with iterative and incremental artifacts, particularly valuable when working with specialized sub-symbolic AI methods like fine-tuned LLMs. Since model performance can only be evaluated after completing a fine-tuning cycle with no guarantee of intended behavior, this iterative

approach, derived from Beck et al. [Bec+01], accommodates uncertainty through systematic adaptation.

For implementation, the framework recommends functional programming languages for both AI and non-AI components. As established by Hughes [Hug89] and Wadler [Wad92], functional paradigms promote immutability, composable functions, and declarative logic that enhance system reliability and maintainability. For hybrid AI systems specifically, these characteristics facilitate transparent data transformations and clean separation of concerns. The framework particularly recommends Elixir, drawing from Cesarini & Thompson's [CT16] work demonstrating how its implementation of Erlang's actor model provides superior fault-tolerance and concurrency management, enabling developers to decompose complex problems while ensuring consistent data handling across the system architecture.

Another fundamental aspect is the endorsement of the Privacy-by-Design principles, which integrate privacy protections proactively during initial design rather than reactively after development. As established by Cavoukian [Cav09], this framework embeds privacy as a default setting within systems, shifting from remedial to preventative protection. For hybrid AI systems in data-sensitive domains, this manifests through architectural decisions that minimize data exposure, including prioritizing local processing over remote services when handling sensitive information, implementing a *local-first* approach as proposed by Kleppmann et al. [Kle+19], applying comprehensive anonymization before remote processing, and designing clear boundaries between components accessing different sensitivity levels of information.

Thorough evaluation and testing of each completed module is a critical principle, especially for AI systems where components may exhibit emergent behaviors. This approach draws from established software engineering practices that emphasize module verification before system integration [Jor14]. For AI systems specifically, this principle becomes even more crucial as untested ML components can introduce what Sculley et al. term "hidden technical debt" [Scu+15, pp. 1, 7f], where errors compound silently until causing significant downstream failures. By validating each module both independently and within the integrated system, developers can ensure components operate as intended while verifying their interactions within the broader application context, substantially reducing deployment risks in data-sensitive domains.

4.2 AI Architecture Patterns and Selection

This guide presents an incremental methodology for AI system design that enables practitioners to align architectures with specific application requirements. The approach progresses through four distinct phases. First, design criteria and constraints are analyzed at component, module, and system levels. Second, appropriate AI paradigms are selected for individual components based on these criteria. Third, the potential

benefits of hybrid AI approaches are evaluated at module or system levels. Finally, the optimal architectural pattern for integrating symbolic and sub-symbolic components is determined.

The system design criteria and architectural patterns build upon the AI paradigms and integration mechanisms introduced in Section 2.2. These recommendations derive from established research in hybrid AI architectures [GLG08; Kam20; Sar21; Sar22], additional literature discussed in Chapter 2, and practical insights from the HAIMEDA application development. Following this structured approach enables informed decisions about when to employ symbolic AI, sub-symbolic AI, or hybrid approaches, and how to effectively integrate them.

4.2.1 Design Criteria and Constraints Across Architectural Levels

Before selecting AI paradigms, practitioners must systematically evaluate the specific requirements that will shape their architectural decisions. The following three categories of criteria should be assessed for each component, module, and the overall system.

4.2.1.1 Data Characteristics

Understanding the nature of input and output data is crucial for paradigm selection. Symbolic AI excels with structured, rule-based data, while sub-symbolic approaches handle ambiguous, high-dimensional data more effectively. These characteristics directly determine which AI paradigm can process the information most efficiently:

- Input data type (e.g., mathematical symbols, structured forms, natural language)
- Output data type requirements (e.g., binary decisions, structured records, free-form text)
- Data variety (homogeneous vs. heterogeneous)
- Data length variation (consistent vs. highly variable)
- Data ambiguity level (well-defined vs. requiring interpretation)

4.2.1.2 Performance Requirements

System performance constraints often dictate paradigm choice. Symbolic systems typically offer faster inference and deterministic behavior, while sub-symbolic systems may require more computational resources but provide greater flexibility. These requirements must be balanced against available resources:

- Time constraints (milliseconds vs. seconds vs. minutes)
- Precision requirements (exact vs. approximate acceptable)
- Consistency demands (deterministic vs. probabilistic acceptable)
- Resource limitations (memory, CPU, or GPU constraints)

4.2.1.3 Domain Constraints

Domain-specific constraints often dictate which architectural options are viable. Symbolic approaches excel when explainability and privacy compliance are mandatory because they offer transparent, deterministic data handling. Operational requirements, such as on-device processing of sensitive data, further narrow the available choices. These non-functional requirements often take precedence over technical considerations when selecting a paradigm:

- Privacy considerations (Personally Identifiable Information (PII) handling, confidentiality requirements)
- Regulatory compliance needs (auditability, explainability)
- Data locality restrictions (on-premises vs. cloud processing)
- Fault tolerance requirements (error handling, recovery mechanisms)

4.2.2 AI Paradigm Selection

4.2.2.1 Component-Level Criteria

Careful consideration is required when selecting an appropriate AI paradigm for each individual component or submodule within a larger system. The criteria presented in Table 4.1 are primarily intended for component-level decisions, where each criterion addresses the specific requirements of discrete functional units. While these criteria are formulated for individual component selection, they may subsequently inform broader architectural considerations when aggregated across multiple components or when evaluating system-wide paradigm choices. The table delineates eleven selection criteria for each AI paradigm to facilitate decision-making at various levels of system design.

Table 4.1: Selection criteria for choosing AI paradigms at the component level.

Symbolic AI	Sub-Symbolic AI
► Well-defined, constrained problem space with clear rules	► Processing natural language with complex linguistic nuances
► Complete explainability and traceability of decisions essential	► Pattern recognition in high-dimensional data (images, audio, sensors)
► Deterministic outputs with guaranteed consistency required	► Generating creative or novel content based on patterns
► Formal verification needed (safety-critical domains)	► Handling ambiguous requests requiring contextual understanding
► Operations involve mathematical calculations or formal logic	► Generalizing across similar but previously unseen examples
► Structured data with explicit rule application	► Dealing with highly variable or unstructured inputs
► Fast processing with minimal latency critical	► Adapting to evolving patterns in data over time
► Working with exact matches or strict patterns	► Processing inputs where rules are difficult to encode
► System must function with minimal training data	► Extracting implicit relationships from data
► Reference data stable and encodable as rules	► Abstracting high-level concepts from low-level features
► Privacy compliance through deterministic data handling	► Acceptable privacy trade-offs for learning benefits

4.2.2.2 Module and System-Level Hybrid Criteria

Following the determination of appropriate AI paradigms for individual components, consideration must be given to the architectural decisions at the module and system levels. The question arises whether maintaining consistency with a single AI paradigm throughout a module or system would be preferable, or whether the adoption of a hybrid AI solution would yield superior outcomes. Such considerations are of particular importance, as the implementation of hybrid AI systems may not be warranted in all contexts. As previously described in 2.2.3, these systems introduce greater architectural complexity and require additional resource allocation. Consequently, the decision to employ hybrid approaches requires thorough evaluation to ensure that the anticipated benefits sufficiently justify the associated trade-offs.

The following criteria may be employed to inform decisions regarding the adoption of hybrid AI paradigms at the module or system level:

- The applicability of more than three criteria from both AI paradigms, as delineated in Table 4.1
- The requirement for cohesive processing of both structured and unstructured data within the system

- The necessity for capabilities encompassing both creative generation and rigorous verification processes
- The need to establish an equilibrium between operational flexibility and formal guarantees
- The presence of processing pipelines that encompass both well-defined procedures and stages characterized by ambiguity
- Domain requirements that necessitate both natural language understanding and formal reasoning capabilities
- The existence of complex workflows exhibiting varying degrees of certainty across different processing stages
- The criticality of maintaining information integrity and factual accuracy whilst preserving naturalness in system outputs

4.2.3 Hybrid AI Strategy Selection

After determining that a hybrid AI approach offers advantages for one or more modules, practitioners must select appropriate integration strategies. This phase addresses how symbolic and sub-symbolic components should be combined to achieve optimal system behavior. The choice of integration strategy significantly impacts system architecture, performance characteristics, and maintenance requirements.

Integration strategies can be applied at various architectural levels: within a single module, across multiple cooperating modules, or throughout the entire system. Furthermore, these strategies are not mutually exclusive. Sophisticated systems may benefit from combining multiple integration approaches, such as using pre-processing in one module while employing parallel processing in another, or layering strategies to create multi-stage verification pipelines.

Table 4.2 presents three fundamental integration strategies, each suited to different operational requirements and system objectives. These strategies build upon the integration patterns introduced in Section 2.2.3 and provide concrete guidance for architectural implementation. Additional implementation techniques and domain-specific examples are detailed in Appendix B.1.

Table 4.2: Integration strategies for hybrid AI architectures and their selection criteria.

Strategy	Primary Use Cases
1: First Symbolic, Then Sub-Symbolic (Pre-Processing Integration)	► Enhancing generation quality through structured inputs ► Optimizing performance by filtering irrelevant data ► Ensuring security by sanitizing sensitive information ► Standardizing inputs for consistent processing
2: First Sub-Symbolic, Then Symbolic (Post-Processing Integration)	► Content generation with verification requirements ► Extracting structured data from unstructured sources ► Creative solutions requiring validation against constraints ► Natural language outputs that must adhere to specific formats
3: Symbolic and Sub-Symbolic in Parallel (Integrated Processing)	► Complex decision processes requiring multiple reasoning modes ► Interactive systems with dynamic adjustment needs ► Workflows combining creativity and formal constraints ► Problems requiring continuous verification during solution development

4.3 Development Lifecycle

The following sections present the phases of the hybrid AI system development lifecycle in a project management oriented manner, which adopt an agile paradigm by supporting iterative repetition of multiple phases (see Figure 4.1). This lifecycle framework emerged from the development of the HAIMEDA application and is fundamentally shaped by the core principles established in Section 4.1. The recommended approach is to proceed from phase one through phase four, then repeat phases two to four as needed until the system is ready for deployment in phase five. After deployment, practitioners should periodically revisit phases two to five to address new or evolving requirements.

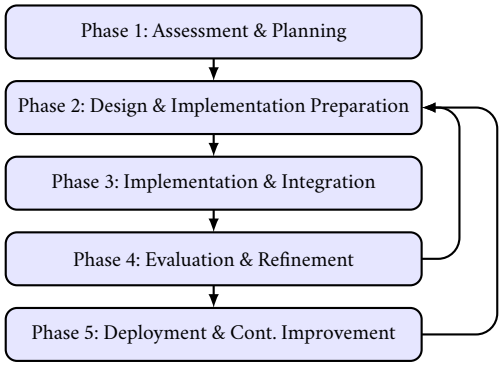


Figure 4.1: The five development phases and their iterative flow in the conceptual framework.

Each phase is structured into framework elements designated by alphanumeric identifiers following the convention E*X.Y*, where X denotes the phase number and Y the element number within that phase. These elements consist of principles and guidelines derived from the practical experience of implementing HAIMEDA, formulated in an intentionally direct, actionable textbook style to facilitate practical application. Principles provide brief foundational statements explained in a few sentences,

while guidelines elaborate on these principles with concrete implementation advice. Key activities that operationalize the principles and guidelines of each element are comprehensively documented in Appendix B.2, rather than within this section. This separation maintains readability by focusing here on conceptual foundations while providing complete implementation checklists in the appendix.

Phase 1: Assessment & Planning

This initial phase requires high rigor as it forms the foundation for all subsequent phases. It establishes key variables that shape the organizational environment and determines which tools, libraries, APIs, and frameworks will guide development.

E 1.1 Domain & Requirements Analysis

E 1.1 Principles:

- **Organize Requirements by Priority**

  Create a structured requirements list that identifies core functionalities and establishes clear data handling boundaries based on domain sensitivity. This ensures the system meets functional needs while adapting to evolving requirements and data protection needs.

- **Expect Requirements to Change**

  Establish protocols for capturing evolving stakeholder needs throughout the project. Requirements in hybrid AI systems frequently shift as stakeholders better understand AI capabilities and limitations, and as new possibilities emerge during implementation.

E 1.1 Guidelines:

During stakeholder interviews, focus on understanding data privacy constraints, integration requirements, and AI capability expectations. Provide clear explanations of AI techniques and their limitations to ensure realistic understanding. Document both explicit requirements and implicit assumptions, particularly regarding AI performance.

For data-sensitive domains, create a formal classification matrix identifying information that requires special handling. Map workflows where AI could augment human decisions, noting where verification or explanation is needed. This mapping reveals where symbolic approaches provide necessary transparency or where sub-symbolic methods handle complex patterns.

Requirements will evolve throughout development, whether from changing stakeholder needs or when AI components underperform expectations. These changes can cascade rapidly across system architecture. Document trade-offs between privacy, performance, and explainability early to prepare for necessary compromises. Pay special attention to integration points between AI techniques, as modifications to data formats,

syntax, or semantics in one module typically require adjustments in connected components. Establishing flexible adaptation protocols minimizes disruption when requirements shift.

E 1.2 Data Analysis

E 1.2 Principles:

- **Examine Data Before Designing**

  Analyzing available data reveals requirements and insights not initially considered by stakeholders, such as insufficient data volume or structures incompatible with language model fine-tuning.

- **Keep Options Open**

  Consider multiple approaches for data preparation, information extraction, and AI processing. Maintaining flexibility prevents premature commitment to suboptimal solutions.

- **Identify Hybrid Integration Points Early**

  Search for opportunities where symbolic and sub-symbolic approaches together overcome individual limitations. Focus on verification points, knowledge enhancement possibilities, and format enforcement needs.

E 1.2 Guidelines:

Begin by profiling data characteristics including volume, structure, quality, and sensitivity levels. This profile guides AI technique selection: structured data with explicit rules suits symbolic approaches, while large unstructured datasets benefit from sub-symbolic methods, as detailed in Section 4.2.

For data-sensitive domains, evaluate privacy implications of different AI techniques. Remote sub-symbolic processing, whether through GPU clusters or cloud services like *Fly.io*, may require data anonymization or synthetic data generation.

E 1.3 Infrastructure & Resource Planning

E 1.3 Principles:

- **Start Small, Scale When Necessary**

  Begin with smaller, locally deployable models before committing to resource-intensive options. Proving concepts with 8B parameter models before scaling to 70B models prevents wasted resources while maintaining upgrade paths.

- **Plan for Multiple Deployment Scenarios**

  Develop fallback options for each processing tier. Data-sensitive domains require contingency plans when primary approaches prove insufficient or deployment environments change unexpectedly.

E 1.3 Guidelines:

Begin infrastructure planning by inventorying available hardware, particularly GPU memory, CPU specifications, and network connectivity. Match these resources with appropriate AI models using quantization strategies that balance performance and constraints.

Training language models requires significantly more computational resources than inference. While quantization efficiently reduces inference costs, avoid it during fine-tuning as it compromises training stability and result quality. For deployment, select the largest practical model for your hardware, typically using 4 to 8 bit quantization to maintain output quality and reasoning capabilities.

For data-sensitive domains, design a tiered processing strategy where each tier represents an option. Tier 1 uses smaller local models for highly-sensitive data, Tier 2 enables anonymized processing with larger remote models when local processing is insufficient, and Tier 3 provides cloud-based services for non-sensitive operations. Document these alternatives to ensure operational continuity.

Test target operating systems and runtime environments early, as they significantly impact technology choices. Verify compatibility of key frameworks such as Ollama[1], LangChain[2], and llama.cpp[3] across potential deployment platforms before committing to specific technologies.

1: https://ollama.com/
2: https://www.langchain.com/
3: https://github.com/ggml-org/llama.cpp

E 1.4 System Outline

E 1.4 Principles:

- **Build Modular, Not Monolithic**

  Create self-contained modules for each core functionality rather than one interconnected system. This enables independent development, testing, and maintenance while allowing modules to evolve at different rates.

- **Keep Module Boundaries Flexible**

  Design adaptive boundaries between components that accommodate changing requirements throughout development. This allows the hybrid AI system to evolve without major architectural redesigns.

- **Coordinate Through the Orchestration Layer**

  Create a consistent orchestration layer with standardized data pipelines for module communication. Modules can use specialized internal implementations while maintaining system coherence through unified data flows.

E 1.4 Guidelines:

Design modules based on functional domains or data processing flows rather than technical implementation. This ensures modules align with business requirements and can evolve as technologies change.

Create standardized interfaces and unified data pipelines for all modules regardless of internal implementation. Each module should expose consistent APIs for inputs and outputs while internally using the most appropriate technologies, whether symbolic AI, LLMs, or rule engines. Define common data formats, transformation patterns, and exchange protocols to ensure reliable communication between diverse AI components.

For data-sensitive domains, implement strict data compartmentalization with explicit handling boundaries between modules. Document where sensitive data enters, processes, and exits the system. Apply the principle of least privilege and need-to-know access, minimizing modules with access to sensitive information. When using remote functionalities like cloud processing, add safeguards including data anonymization and encryption for third-party connections. Plan for both vertical scaling, improving individual module capabilities, and horizontal scaling, adding new modules. Document extension points for future modules to enable system growth without architectural overhaul.

Phase 2: Design & Implementation Preparation

Phase 2 begins the iterative cycle through Phases 2 through 4 or 5. Each iteration can focus on a single module or modify multiple modules simultaneously when system refinements affect shared data formats, syntax, or semantics. This phase provides guidance for data preparation, whether for language model fine-tuning or symbolic AI construction.

E 2.1 Module Selection & Planning

E 2.1 Principles:

- **Build Core Features First**

  Select modules delivering core requirements before peripheral features. This approach front-loads value delivery and establishes critical functionality early while creating a foundation for iterative enhancement.

- **Define Clear Module Boundaries**

  Specify module responsibilities according to the system outline with explicit input/output definitions. Maintain strict functional boundaries to prevent scope creep during planning.

- **Prototype Before Committing**

  Create experimental implementations for uncertain components before finalizing architecture. Prototypes validate technical approaches and reveal integration challenges early.

E 2.1 Guidelines:

Use dependency graphs to identify logical development sequences. Prioritize modules providing core services or fundamental data transformations before dependent modules.

Expand the system outline into detailed technical specifications defining how modules accomplish their tasks. For high-risk or uncertain components, develop low-fidelity prototypes to validate assumptions. These prototypes both confirm technical feasibility and inform contingency planning. Document alternative approaches for risky components to ensure development continues if initial strategies fail.

Test APIs, libraries, and third-party services with representative data before integration. Verify parameter handling, response formatting, and module compatibility to prevent integration surprises and confirm external dependencies function as promised.

E 2.2 Data Preparation & Enrichment

E 2.2 Principles:

- **Prioritize Data Quality Over Quantity**

  Focus on data accuracy, consistency, and relevance rather than volume. Clean, well-structured data yields better results for both symbolic rule extraction and sub-symbolic fine-tuning than larger amounts of noisy information.

- **Build In Privacy From the Start**

  Integrate privacy protection into data workflows from the beginning. For data-sensitive domains, this proactive approach ensures compliance while maintaining data utility for AI development.

E 2.2 Guidelines:

Begin data preparation by implementing the sanitization procedures for PII and sensitive content identified in Phase 1. When using remote technologies like cloud GPUs or third-party services, apply planned anonymization or synthetic data generation to minimize sensitive data exposure.

Transform stakeholder data into machine-processable formats, including plain text (TXT) files or structured Markdown (MD) files. Apply systematic normalization to eliminate noise and redundancies while preserving contextual meaning. Address conversion artifacts such as character encoding issues, structural corruption, or metadata loss. Based on Phase 1 model specifications, establish context window constraints for fine-tuning datasets. Limit individual files to 80 % of the model's maximum token capacity, allocating 10 % for task templates and instructions, with a 10 % safety margin for tokenizer variations. This conservative approach ensures processing stability and prevents truncation during training and inference.

Extract high-value information through targeted techniques that leverage domain expertise. Generate derivative datasets highlighting domain-specific patterns, edge cases, and specialized contexts relevant to module development. This extracted information enhances module functionality, enriches fine-tuning data, and creates comprehensive evaluation datasets.

For sensitive domains, ensure extraction processes maintain Phase 1 privacy constraints.

Document all transformation decisions, filtering criteria, and extraction rules for reproducibility and transparency. For complex datasets, create hierarchical pipelines progressing from general cleaning through domain-specific transformations to module-specific optimizations. Develop validation procedures confirming semantic equivalence between original and transformed data to prevent information loss or distortion.

E 2.3 AI Component Configuration

E 2.3 Principles:

- **Choose AI Techniques Based on Module Needs**

  Select AI approaches based on defined module requirements rather than technological preferences. This ensures each module uses techniques best suited for its specific tasks.

- **Confirm AI Capabilities Early**

  Test AI components with representative data before full implementation to prevent significant rework later in development.

E 2.3 Guidelines:

Before implementing modules from the *Module Selection & Planning* stage, validate candidate AI approaches through focused experiments with domain-specific data. For language models, test performance across different parameter settings, including temperature and context utilization, using prompts resembling production use cases with data from the *Data Preparation & Enrichment* stage. This experimentation identifies optimal configurations and limitations while informing parameters for subsequent fine-tuning.

For hybrid AI modules, test integration between symbolic and sub-symbolic components using representative data samples. This early validation prevents compatibility issues and ensures smooth transitions between different AI approaches.

Phase 3: Implementation & Integration

This phase transforms the outlined modules and tested AI techniques into working implementations. Modules requiring fine-tuned models begin with instruction dataset building and the fine-tuning process. These fine-tuned models then serve as foundations for remaining module components, whether standalone or combined with symbolic AI techniques.

E 3.1 Dataset Building & Fine-Tuning

E 3.1 Principles:

- **Match Training to Real Use**

  Design instruction templates that mirror the prototype prompts from AI Component Configuration. This alignment ensures models learn to respond appropriately to the exact interaction patterns they will encounter in deployment.

- **Keep General Skills While Specializing**

  Construct datasets emphasizing domain-specific knowledge while preserving general task capabilities. This balance prevents catastrophic forgetting while enabling meaningful specialization.

E 3.1 Guidelines:

Fine-tuning datasets require strict consistency between training instructions and inference prompts to ensure effective knowledge transfer to production. Structure datasets according to evaluation metrics from stage *Module Selection & Planning*, allocating 70–80 % for training, 10–20 % for validation, and 5 15 % for testing. This distribution provides sufficient learning data while reserving samples for validation and evaluation. Format each instruction using conventions that tested well during component configuration, such as `[INST]`/`[/INST]` for Llama models or ### for Mistral models.

Incorporate enriched data from stage *Data Preparation & Enrichment*, using extracted entity relationships and domain patterns to create contextually rich instructions. Apply the same data processing pipelines developed during preparation.

Domain-specific fine-tuning should leverage successful task prompts from component testing to create instructions addressing domain knowledge retrieval, format adherence, and reasoning within domain constraints, including specific rule compliance.

For hybrid AI systems, create specialized instructions teaching models to recognize boundaries between symbolic and sub-symbolic components. Include examples where models acknowledge uncertainty or identify cases requiring symbolic verification.

E 3.2 Implementation of Symbolic AI Components

E 3.2 Principles:

- **Combine Symbolic with Sub-symbolic Strengths**

  Design symbolic components to address limitations found in sub-symbolic approaches during testing. Focus on explicit rules, formal verification, or logical reasoning where sub-symbolic methods struggle, such as strict format enforcement or complex conditional logic.

- **Make Symbolic Processing Transparent**

  Implement symbolic components with explicit rule tracing and decision explanation capabilities. This transparency enables behavior verification, facilitates debugging, and provides accountability in data-sensitive domains.

E 3.2 Guidelines:

Symbolic AI implementation begins with analyzing module requirements from Phase 2 and framework elements from Phase 1 to select appropriate approaches. These components function either as standalone solutions or as complements to sub-symbolic methods.

Standalone symbolic implementations focus on capturing domain expertise through formal knowledge representation and explicit rules. In hybrid systems, performance metrics and limitations from sub-symbolic testing reveal enhancement opportunities. Common patterns include input validation and preprocessing, rule-based output generation or verification, grammar-based format enforcement, and formal reasoning for complex conditional logic.

Rule-based systems should start with minimal core rules addressing primary objectives and documented model limitations. Prioritize rules directly impacting primary objectives rather than encoding entire domain knowledge.

Knowledge representation formalism depends on specific verification tasks from component testing. Frame-based representations suit structured domain concepts, production rules handle conditional logic and business processes, while formal logic provides rigorous verification for critical decisions. Document representation choices with explicit rationale linking them to addressed sub-symbolic limitations.

For remote processing in data-sensitive domains, implement symbolic verification components that validate sub-symbolic outputs against privacy and compliance constraints without accessing original sensitive data. These components operate on abstracted or anonymized representations while providing meaningful verification. In hybrid modules, testing sub-symbolic components before implementing symbolic components identifies output patterns and edge cases early. This sequence allows symbolic components to be designed with concrete understanding of their processing requirements.

E 3.3 Unit and Integration Testing of Components

E 3.3 Principles:

- **Test Individual Components Before Integration**

  Validate each AI component independently before combining them in hybrid configurations. This isolates issues to specific components and establishes baseline performance expectations.

- **Include Edge Case Testing**

  Focus test cases on boundary conditions where components are most likely to fail, particularly at transition points between symbolic and sub-symbolic processing. Edge cases often reveal integration issues not apparent during unit testing.

E 3.3 Guidelines:

Sub-symbolic component testing requires consistent application of parameters identified during AI Component Configuration, including temperature, top_p, and other sampling settings. This consistency ensures performance differences reflect actual model behavior rather than parameter variations. Test cases should cover the full input range, emphasizing edge cases that might trigger undesired behaviors. Fine-tuned models require both automated evaluation using established metrics and manual evaluation of format adherence, reasoning quality, and domain alignment.

Symbolic component testing involves test suites verifying rule execution correctness, logical consistency, and boundary condition handling. Tests must confirm that symbolic components implement formal specifications from earlier phases. Document edge cases where symbolic reasoning produces unexpected results, particularly in complex rule interactions or ambiguous inputs.

Hybrid AI modules require integration tests targeting handoff points between symbolic and sub-symbolic components. These tests verify output-input compatibility, semantic consistency, and error handling across components. Create scenarios triggering fallback mechanisms to ensure proper execution when primary approaches fail. Use logging to capture internal component states, facilitating debugging of complex interactions.

Document all limitations and edge cases producing failures or unexpected results. These documented limitations guide safeguard implementation and system refinement in subsequent iterations. Data-sensitive domains require specific tests verifying privacy constraint maintenance throughout processing stages.

E 3.4 Interface & Pipeline Integration

E 3.4 Principles:

- **Verify Module Connections**

  Test all communication channels between new modules and existing system components before deployment. Focus on data format compatibility, event handling, and error propagation across module boundaries.

- **Keep Module Boundaries Intact**

  Maintain established module boundaries during integration while

making incremental orchestration layer changes that minimize disruption to existing functionality. This ensures system stability while accommodating new capabilities.

E 3.4 Guidelines:

Following successful component implementation and testing, integrate the complete module into the system's orchestration layer according to interface specifications from the *System Outline* stage in Phase 1. For hybrid AI implementations, verify that symbolic and sub-symbolic handoff points operate correctly within the overall system context, particularly when triggered by external events or within larger processing pipelines.

Multi-user systems and remote resource utilization require verification that data protections remain intact during inter-module information flow. Confirm that sanitization procedures, access controls, and privacy safeguards function correctly when modules operate within the full system rather than in isolation. This verification becomes critical when sensitive data crosses user contexts or transmits to external processing environments.

Document all integration points, data exchange formats, and event handling protocols implemented during this phase. This documentation provides essential reference for future maintenance and system extensions.

Phase 4: Evaluation & Refinement

This final phase of each development iteration encompasses testing and evaluating the complete system after module additions or updates. After implementing all projected modules, the process continues to Phase 5. When development remains incomplete, the next iteration begins from Phase 2.

E 4.1 System-Level Evaluation

E 4.1 Principles:

- **Test in Real-World Conditions**

  Test the system with realistic scenarios mirroring actual stakeholder workflows rather than isolated functional tests. This reveals integration issues and user experience challenges that unit testing cannot identify.

- **Verify Requirements Are Met**

  Evaluate system performance against primary objectives and success criteria from Phase 1's *Domain & Requirements Analysis*. This comparison ensures implementation addresses stakeholder needs beyond technical functionality.

E 4.1 Guidelines:

System evaluation requires assessing both technical performance and stakeholder requirement alignment. Evaluation scenarios should correspond to use cases and workflows from Phase 1, incorporating the full input range including boundary cases and problematic patterns from component testing.

System-level testing focuses on cross-module interactions rather than isolated functionality. Evaluation examines data and control flow across module boundaries, emphasizing transformation accuracy, error propagation, and recovery mechanisms. Hybrid AI systems require special attention at symbolic and sub-symbolic handoff points, where emergent behaviors often arise.

Both automated and manual evaluations should be performed. Automated testing provides consistent quantitative metrics, while manual evaluation captures qualitative aspects difficult to measure programmatically, particularly for sub-symbolic AI-generated outputs. Data-sensitive domains require verification that privacy, security measures, and user access controls function correctly at system level, with specific tests identifying potential information leakage across module boundaries.

Documentation of system-level issues requires clear categorization: critical flaws needing immediate attention, performance limitations requiring architectural changes, and minor issues addressable through parameter tuning or small adjustments. This prioritized documentation guides subsequent refinement planning.

E 4.2 Refinement Planning

E 4.2 Principles:

- **Address High-Impact Issues First**

  Prioritize fixes based on objective impact on primary system objectives rather than addressing all limitations equally. Focus on issues that affect core functionalities before tackling peripheral concerns.

- **Requirements Drive Refinements**

  Each planned refinement must directly address specific stakeholder requirements or system limitations from the *System-Level Evaluation*. This connection maintains focus on business value over technical perfection.

E 4.2 Guidelines:

Refinement planning begins by revisiting primary and secondary objectives from Phase 1's *Domain & Requirements Analysis*. Issues from *System-Level Evaluation* then populate a structured prioritization matrix mirroring the initial requirements hierarchy. Critical issues impacting

primary objectives or core functionality receive highest priority, followed by those affecting secondary objectives or performance, with minor enhancements deferred when resources are limited. This approach maintains alignment with original stakeholder needs.

Each high-priority issue requires a specific remediation plan detailing technical approach and estimated effort. Issues may need localized adjustments such as parameter tuning and rule modification, or significant architectural changes including component redesign and integration reconfiguration. Minor issues involving only configuration parameters or localized code receive immediate attention rather than deferral, preventing technical debt accumulation while maintaining momentum.

Iteration planning identifies the next module or objective for implementation. Remaining requirements from Phases 1 and 2 guide selection based on business value, existing component dependencies, and implementation complexity. Dependencies between refinement tasks and new module development establish logical implementation sequences, preventing wasted effort and ensuring systematic improvement.

The cumulative scope of refinements and next-module implementation determines whether to initiate a new development iteration returning to Phase 2, or proceed to Phase 5 deployment. Significant multi-component redesigns require thorough review of original Phase 1 requirements and system specifications before beginning the next iteration.

Phase 5: Deployment & Continuous Improvement

This final phase begins after achieving all primary objectives and system readiness for deployment. The phase transitions the system from development to production while establishing continuous improvement mechanisms. Production environment testing generates feedback revealing previously unidentified requirements and refinement opportunities. Deployment continues until sufficient new requirements and enhancements accumulate, triggering a new development iteration from Phase 2. This cyclical approach ensures ongoing system evolution responding to real-world usage patterns and emerging stakeholder needs.

E 5.1 Production Environment Setup & Deployment

E 5.1 Principles:

- **Implement Comprehensive Monitoring**

  Integrate robust monitoring and logging systems tracking both technical performance metrics and AI-specific indicators such as inference latency, model confidence scores, and usage patterns.

- **Use Least Privilege Access**

  Configure production systems with minimum necessary access rights to data resources and external services. This principle ensures data-sensitive domain security while maintaining functional system operation.

E 5.1 Guidelines:

Production deployment establishes the operational environment for serving end users. This requires configuring all dependencies, libraries, and runtime requirements from development within production infrastructure. Data-sensitive domains need careful attention to production-specific security configurations, including access controls, encryption settings, and sanitization procedures that often differ from development due to scale, user patterns, or regulatory requirements.

Deployment documentation maps development configurations to production settings, explaining intentional differences. Document version specifications, model quantization levels, and fixed AI component parameters. Verification procedures should test the complete system in controlled environments mirroring production conditions, using virtual machines or staging environments before actual deployment.

Monitoring systems provide ongoing detection of performance degradation and unexpected behaviors. This visibility transforms stakeholder-reported issues into specific technical limitations, generating actionable data for systematic improvements.

E 5.2 User Testing & Feedback Collection

E 5.2 Principles:

- **Observe Actions, Not Just Opinions**

  Prioritize direct observation of user interactions over self-reported experiences. This reveals actual usage patterns and challenges users may not articulate in formal feedback.

- **Focus Feedback on Core Objectives**

  Structure feedback collection to assess primary objective fulfillment rather than gathering generalized impressions. This approach delivers specific directions for meaningful improvements.

E 5.2 Guidelines:

User testing employs a two-phase approach combining structured assessment with open exploration. Task-oriented scenarios mapped to primary objectives from stage *Domain & Requirements Analysis* verify that core functionalities align with stakeholder expectations under realistic conditions.

Feedback collection uses complementary methods: structured evaluation forms for quantitative performance assessment, semi-structured interviews capturing qualitative experiences, and direct observation revealing actual usage patterns. Hybrid AI systems require particular attention to user perceptions of reliability, transparency, and output appropriateness. Instances of user surprise or confusion often reveal misalignments between mental models and system functionality.

Consistent metrics for feedback categorization enable quantitative tracking across test sessions. These metrics link directly to primary objectives and technical decisions, facilitating precise refinement planning. Unexpected usage patterns where users employ the system beyond design intentions often reveal valuable enhancement opportunities for subsequent iterations.

E 5.3 Continuous Monitoring & Improvement

E 5.3 Principles:

- **Evolve Based on Usage Patterns**

  Systematically capture user interactions and transform them into structured datasets for improving both symbolic and sub-symbolic components. This continuous feedback loop enables system evolution based on authentic usage patterns and resilience against concept drift.

- **Maintain Version Control for AI Components**

  Track and manage different versions of models, rules, and configurations in production to enable safe rollbacks and A/B testing. This versioning strategy allows performance comparison over time and provides safety nets when deploying updates.

E 5.3 Guidelines:

Analysis workflows aggregate usage statistics across key dimensions including response accuracy, processing time, resource utilization, and user satisfaction. Patterns diverging from development expectations often highlight valuable refinement opportunities. Resource consumption patterns deserve particular attention for identifying optimizations that reduce operational costs while maintaining or improving performance.

Hybrid AI systems benefit from reusing data transformation pipelines to convert user data into training materials for both component types. Fine-tuned models require datasets from successful interactions exemplifying desired behavior, particularly in previously challenging scenarios. Symbolic components need analysis of manual overrides or unexpected results to refine logical rules and constraints.

Parallel monitoring of system usage and emerging technologies involves regularly evaluating new AI models, libraries, and frameworks against existing implementations. Technical horizon scanning combined with periodic stakeholder interviews identifies evolving requirements beyond usage data. Clear thresholds trigger refinement planning when sufficient improvements accumulate or transformative technologies emerge. These thresholds consider both refinement volume and potential impact on primary objectives. Meeting these conditions initiates structured transition to Phase 2, ensuring production insights inform the next development iteration.

4.4 Framework Toolkit

The framework includes practical tools designed to streamline hybrid AI system development and reduce implementation complexity. These tools primarily support sub-symbolic AI module development, particularly fine-tuning pre-trained language models, since symbolic AI development typically requires domain-specific approaches rather than standardized toolkits. While neural approaches benefit from broadly applicable frameworks like PyTorch[4] or TensorFlow[5], symbolic systems demand custom knowledge representations and reasoning engines tailored to specific domains [DM15, pp. 97, 99, 102f], including specialized ontologies and inference mechanisms reflecting domain-specific constraints [Har+07, pp. 396f, 408f, 932ff].

4: https://pytorch.org
5: https://www.tensorflow.org

Specialized Satisfiability Modulo Theories (SMT) solvers such as Z3 [6] or CVC5 [7] provide formal verification capabilities, while rule engines like Drools[8] or CLIPS[9] enable knowledge-based reasoning. However, these typically require custom interfaces between symbolic and sub-symbolic elements. Chapter 5 presents practical symbolic AI implementations within the medical device assessment application, with concrete implementation code accessible via the HAIMEDA project repository[10].

6: https://www.microsoft.com/en-us/research/project/z3-3/
7: https://cvc5.github.io/
8: https://www.drools.org
9: http://www.clipsrules.net

10: https://github.com/penthooose/HAIMEDA

The toolkit components align with the framework's development cycles and were developed primarily to support custom LLM fine-tuning for specific tasks within the paradigm application. Components include *Data Preparation Tools*, including the *PII Sanitization Tool*, as well as *Information Extraction Tools*, *Dataset Building Tools*, *Model Fine-Tuning Tools*, and *Model Integration Tools*. Each component description covers its purpose, potential applications, and key functionalities, with detailed technical specifications in Appendix C.1. The complete source code is available in the framework toolkit repository[11].

11: https://github.com/penthooose/framework_toolkit

4.4.1 Data Preparation Tools

4.4.1.1 PII Sanitization Tool

The PII Sanitization Tool securely prepares confidential documents for AI training by identifying, extracting, and sanitizing PII to ensure data privacy compliance when preparing training datasets for LLM fine-tuning. Built on Microsoft's Presidio[12] framework with domain-specific extensions, the tool combines pattern-based entity recognition with context-aware validation to identify over 25 entity types including names, addresses, and identification numbers, optimized for German and English content.

12: https://microsoft.github.io/presidio/

Implemented as a standalone Elixir Phoenix[13] web application with a web-based interface, the tool supports both interactive document processing and batch operations for PDF, DOCX, and TXT formats. The system performs multi-stage entity detection while preserving structural integrity during content extraction. Sanitization strategies range from complete removal to categorized anonymization, preserving semantic distinctions

13: https://www.phoenixframework.org/

between entity types while eliminating specific identifiers. The tool generates outputs in various formats, and all processing activities are logged with verification reports for compliance documentation. This preprocessing capability proves particularly valuable in high-confidentiality domains like medical and legal sectors, enabling practitioners to leverage proprietary documents for LLM fine-tuning while maintaining regulatory compliance and ethical data handling standards. Full technical specifications appear in Appendix C.1.1.1.

4.4.1.2 Data Transformation Submodules

This subcollection of Data Preparation Tools contains modular components that function independently or in combination. Created specifically for preparing fine-tuning data for LLMs in the medical device assessment application, these tools serve primarily as implementation examples for dataset preparation workflows. The toolkit provides comprehensive data transformation capabilities for converting documents into formats suitable for instruction fine-tuning datasets. A central *Data Preparation Controller* orchestrates the modular pipeline via a unified Application Programming Interface (API), managing the transformation processes before instruction-based training data creation.

Key components include *File Extraction* for recursive document retrieval from structured directories, *MDB Data Extraction* leveraging Microsoft Database (MDB) Tools[14] through Windows Subsystem for Linux (WSL) for relational database integration, and *File Conversion* using Pandoc to transform proprietary formats into standardized MD. Additionally, the *Data Partitioning* module handles intelligent segmentation based on heading structures, overlength document processing, Q&A pair formatting, and contextual chapter relationships. Notable features encompass configurable token counting adaptable to different languages and technical vocabularies, intelligent document hierarchy handling, and comprehensive statistical analysis of processed document characteristics. While developed for medical device report preparation, some modules generalize across domains, enabling transformation of proprietary documents into structured formats suitable for instruction-based fine-tuning dataset creation. Complete technical specifications and functional descriptions appear in Appendix C.1.1.2.

14: https://mdbtools.github.io/

4.4.2 Information Extraction Tools

The Information Extraction Tools provide separate symbolic and sub-symbolic approaches for extracting and enriching document content. Symbolic extraction uses regex patterns to capture recurring information, while sub-symbolic extraction employs LLMs to generate new data such as chapter summaries or answers to chapter-specific questions.

This toolset extracts and restructures information that can later be used for dataset creation, enabling development of specialized models for different purposes from the same base data. While primarily designed to prepare data for fine-tuning LLMs in sub-symbolic AI modules, the

extracted information could theoretically support symbolic AI development through rules, ontologies, or other symbolic representations. Both modules offer generalizable designs that, with minor modifications, can process various user data types. They include implementation examples with regex patterns and variables from surveyor report data extraction for reference.

The *Information Extractor* component coordinates these independent extraction methods via a unified workflow management system employing two complementary approaches. Here, the *Symbolic Extractor* component features a customizable extraction framework where practitioners define domain-specific regex patterns. As a point of reference, the regexes used for information extraction when fine-tuning LLMs for HAIMEDA remain inside the component, demonstrating this approach with specialized patterns. However, domain experts must create patterns for their specific documents. This architecture supports automatic value normalization and hierarchical data extraction while maintaining extraction provenance.

In contrast, the *Sub-Symbolic Extractor* uses LLMs via configurable prompt templates to extract implicit information, generate summaries, and produce question-answer pairs. This approach captures contextual understanding that regex patterns cannot achieve. Both extraction methods generate structured JSON outputs that preserve semantic relationships within the extracted information. This dual approach combines symbolic precision with sub-symbolic contextual understanding for comprehensive data preparation. Appendix C.1.2 provides more detailed technical specifications.

4.4.3 Dataset Building Tools

These tools enable the creation of structured instruction datasets for fine-tuning LLMs by transforming prepared content into standardized JSON Lines (JSONL) files with configurable training, validation, and testing splits. These tools support two primary dataset types: unsupervised datasets processing document chapters and database records into self-supervised learning formats, and supervised datasets combining content with explicit instructions for tasks like document formatting and question-answering.

The *Instruction Utilities* components provide critical optimization functions. `MergeInstructions` combines data from multiple sources with balanced representation and proper statistical distribution, while `Instruction Preparation` analyzes and filters datasets based on token length estimations to ensure model context window compatibility. All dataset builders implement consistent text normalization, intelligent content validation, and produce outputs compatible with PyTorch[15] and HuggingFace[16] training pipelines.

15: https://pytorch.org/

16: https://huggingface.co/

An important innovation for fine-tuning merged models is introduced by the *Coherence Datasets* component. Drawing from multiple configurable

sources, this dataset creation tool selects either unsupervised or supervised instructions exclusively from validation and test sets that the model has not encountered during training. This approach helps align merged model layers, ensuring coherent behavior when combining knowledge domains. Although developed for medical device assessment, these components demonstrate effective dataset building patterns adaptable to other domains. Complete technical specifications appear in Appendix C.1.3.

4.4.4 Model Fine-Tuning Tools

The Model Fine-Tuning Tools bridge Elixir and PyTorch for efficient language model adaptation through a layered architecture emphasizing parameter-efficient training techniques. A core `FineTuningController` communicates with Python via ErlPort, prioritizing memory-efficient fine-tuning through LoRA adapters while supporting comprehensive model merging capabilities.

The system implements unsupervised and supervised training approaches with specialized dataset handling, real-time progress reporting, and checkpoint management for training resilience. Model knowledge integration supports both adapter-to-model merging and sophisticated model-to-model combining using the SLERP technique, effectively preserving knowledge from multiple sources while maintaining coherent behavior.

Structured JSON configuration files define hyperparameters, optimization strategies, and hardware acceleration options, creating a flexible yet consistent adaptation approach. This toolset significantly reduces technical barriers to creating domain-specialized language models on consumer hardware while maintaining compatibility with downstream deployment systems. Complete technical specifications and implementation guidance appear in Appendix C.1.4.

4.4.5 Model Integration Tools

The Model Integration Tools provide essential post-training capabilities for optimizing model deployment across resource-constrained environments through a consistent two-tier architecture pattern. The toolkit converts fine-tuned models from safetensors to GPT-Generated Unified Format (GGUF) for deployment in environments like Ollama and LMStudio[17], while implementing quantization techniques to optimize memory usage and execution speed.

17: https://lmstudio.ai/

Built on a layered architecture, Elixir controllers manage workflows while Python scripts interface with the llama.cpp library through ErlPort, maintaining clear separation between high-level process management and technical implementation. The *Model Converter* component handles automatic script location detection across environments and standardized parameter handling. The *Model Quantizer* supports configurable precision-speed trade-offs through various quantization options.

Structured JSON configuration files control both components, defining conversion parameters, quantization levels, and output specifications. This integration toolkit enables efficient deployment of specialized models on hardware with varying resource constraints while maintaining compatibility with downstream inference systems. Complete technical specifications appear in Appendix C.1.5.

5 HAIMEDA: A Paradigm Implementation of the Hybrid AI Development Framework

This chapter presents the HAIMEDA application (see Figure 5.1), demonstrating the framework in practice. Section 5.1 documents essential modules and components, highlighting applications of AI techniques. Section 5.2 presents the complete approach following the development cycle structure introduced in Section 4.3, incorporating the surveyor's objectives, individual tasks, and workflow from Sections 3.2.1.1 and 3.2.1.3. Finally, Section 5.3 documents the various fine-tuning cycles and individual stages, offering insights into the empirical fine-tuning process.

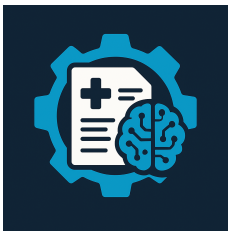


Figure 5.1: Logo of the HAIMEDA application.

5.1 System Architecture and Modules of HAIMEDA

This section focuses on modules and components that are either essential to HAIMEDA or contain AI paradigms. The presentation follows a high-level approach, highlighting architectural design, AI components, and functionalities without exhaustively detailing specific functions or implementation details. Additional technical details and modules are documented in Appendix C.2, while the README file and application code are available in the HAIMEDA project repository[1].

1: https://github.com/penthooose/HAIMEDA

5.1.1 System Architecture Overview

HAIMEDA implements a sophisticated hybrid AI system built on Elixir's Open Telecom Platform (OTP) foundations, utilizing Elixir Desktop[2] to deliver a native desktop application with Phoenix LiveView for real-time web interfaces. The architecture combines Erlang virtual machine fault-tolerance with modern web paradigms while maintaining a native desktop user experience essential for professional report authoring. WxWidgets integration enables the web-based Phoenix application to render within native desktop windows, providing OS-level features like

2: https://github.com/elixir-desktop/desktop

system menus and file dialogs while preserving the familiar desktop application feel.

Central to the design is a process-based architecture using Elixir's actor model, coordinated through the `MainController` orchestration pattern that manages all AI modules via standardized communication protocols. The system leverages OTP supervisor trees for automatic process restart capabilities, ensuring robust operation in production environments where report integrity is paramount. Phoenix LiveView enables server-side rendering with WebSocket-based real-time interactivity, eliminating the need for complex JavaScript frontend frameworks while maintaining responsive user interactions through asynchronous Task-based operations and process ID communication between LiveView processes and background AI tasks. This approach enables sophisticated collaborative editing features while ensuring proper resource management and error isolation. Content persistence through MongoDB[3] integration with intelligent caching ensures automatic saving of all user interactions without explicit save actions, optimizing database operations while protecting against data loss during intensive AI operations.

3: https://www.mongodb.com/

5.1.2 AI Modules and Integration Patterns

MainController and Orchestration Layer The `MainController` serves as the central orchestration hub, coordinating interactions between all core modules through standardized communication protocols while managing five primary workflows: chapter creation pipeline with pre- and post-processing validation, content verification with configurable parameters, text processing operations for optimization and revision, research query processing using RAG patterns, and LLM integration management with dynamic model configuration. The controller facilitates bidirectional data flow using process IDs for asynchronous communication, ensuring User Interface (UI) responsiveness during intensive AI processing with comprehensive error handling and centralized configuration management through `application_properties.yaml`. The central role of the `MainController` orchestration layer and its interactions with all major modules and interfaces in HAIMEDA are illustrated in Figure 5.2. More technical specifications on the configuration management are provided in Appendix C.2.1.

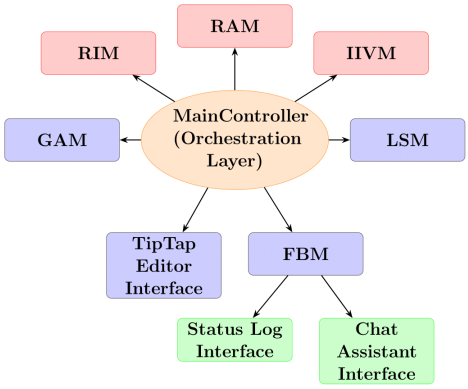


Figure 5.2: Overview of the MainController orchestration layer in HAIMEDA: central coordination of all core modules and feedback components, illustrating the system's modular architecture and data flow.

Core Modules HAIMEDA comprises of several core and submodules that implement distinct aspects of the hybrid AI workflow. This section focuses on the three primary AI modules and their integration patterns: the *Report Authoring Module (RAM)* for LLM-based content generation, the *Information Integrity Verification Module (IIVM)* for hybrid verification combining symbolic and sub-symbolic techniques, and the *Research and Investigation Module (RIM)* for RAG-based information retrieval with symbolic query preprocessing. The functionalities of those core modules regarding symbolic, sub-symbolic, and hybrid AI mechanisms are shown in Table 5.1.

Table 5.1: Comparison of AI techniques across the core modules of HAIMEDA.

Module	Symbolic AI Techniques	Sub-symbolic AI Techniques	Hybrid Integration Patterns
RAM	▸ Template-based document generation ▸ Rule-based context selection ▸ Structured mapping of report sections	▸ LLM inference with specialized domain adapters ▸ Context window management ▸ Content generation with style constraints	▸ Information verification loops ▸ Context preparation pipeline ▸ Tiered approval process with feedback integration
IIVM	▸ Rule-based verification engines ▸ Formal logic verification ▸ Pattern matching for entity validation ▸ Consistency checking through tableaux methodology	▸ ML-based similarity scoring ▸ Statistical anomaly detection ▸ Coherence validation through embedding comparison	▸ Cross-validation between symbolic rules and ML predictions ▸ Entity verification flows with fallback mechanisms ▸ Confidence scoring with multi-method consensus
RIM	▸ Keyword extraction and classification ▸ Symbolic word analysis ▸ Structural query transformation ▸ Taxonomy-based search classification	▸ Vector-based semantic search ▸ Embedding similarity ranking ▸ Query expansion through contextual understanding	▸ Two-tier query processing pipeline ▸ Hybrid search result ranking ▸ Context-enhanced document retrieval with symbolic pre-filtering

Secondary Modules and Components Supporting these core AI modules are additional modules and components: the *Gateway API Module (GAM)* for Elixir-Python integration, the *LLM Service Module (LSM)* for unified model access, the *Feedback Module (FBM)* for user communication, the user interface components including an implementation of the TipTap[4] editor for rich-text document editing, and various utility modules for session management and configuration. Further technical specifications for these secondary modules can be found in Appendix C.2.2.

4: https://tiptap.dev/

5.1.3 Interworkings of the Modules

HAIMEDA's workflow begins with user-provided inputs in the editor environment, which flow through the verification and content generation pipeline following the orchestration patterns established by the `MainController`. When generating report content, the workflow begins with user-provided inputs in the editor environment, which `MainController` passes to the IIVM Pre-Processor for symbolic validation using tableaux logic. If validated, these inputs flow to RAM where JSON-based prompt templates transform them into LLM queries executed through the LSM, which maintains standardized access to language models via the Ollama API. Generated content then undergoes hybrid verification by the IIVM

Post-Processor, which employs both symbolic entity detection and ML-based similarity scoring, leveraging Gateway API for Python integration, to identify potential inconsistencies. Feedback from both verification stages is delivered to users through FBM's status logs and visual indicators in the TipTap editor, which renders entity-level corrections as interactive elements. The workflow for the report chapter creation in HAIMEDA is illustrated in Figure 5.3.

For research tasks, user questions from the chat interface are routed to RIM, which employs symbolic word processing to extract entities before performing targeted database queries and vector-based searches across historical reports, with results returned through FBM's chat interface. This interconnected workflow enables continuous verification loops where each module's outputs become inputs for subsequent modules, creating a comprehensive system that combines the precision of symbolic approaches with the flexible generation capabilities of sub-symbolic AI while maintaining information integrity throughout.

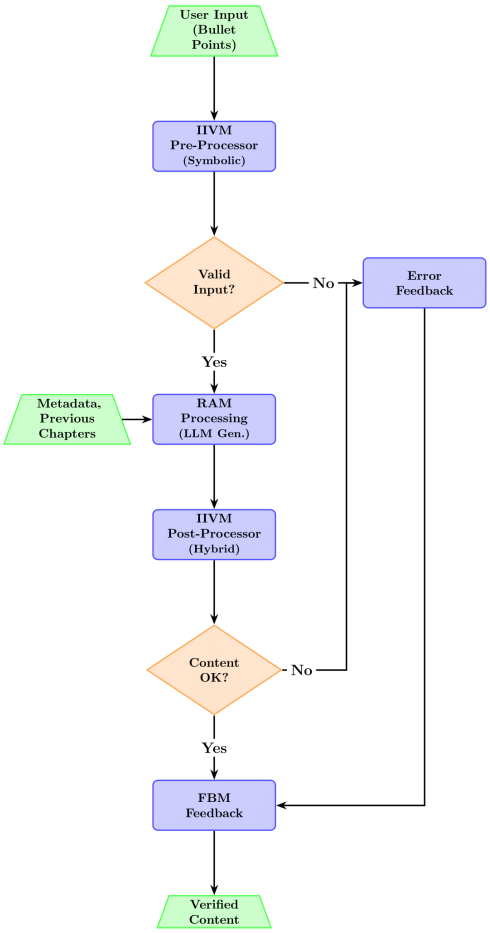


Figure 5.3: Workflow for report chapter creation in HAIMEDA: from user input and symbolic validation, through LLM-based generation and hybrid content verification, to interactive feedback and final content approval in the editor.

5.1.4 Core Modules of HAIMEDA

5.1.4.1 Report Authoring Module

The RAM serves as HAIMEDA's primary content generation component for medical device assessment reports, implementing sophisticated prompt engineering techniques through a multi-tiered template architecture that dynamically composes context-aware prompts from hierarchically organized components. The module employs context-dependent prompt generation strategies that adapt based on available previous content types, party statements, and metadata, either using full chapters or condensed summaries as context sources, with intelligent variable substitution that preserves semantic relationships between document sections. RAM integrates complex document structure awareness through its templating system, where JSON-defined prompt fragments maintain strict separation between engineering logic and runtime content, enabling dynamic adaptation to different chapter types without modifying core processing code.

The system implements template-driven prompt construction with explicit component hierarchy, i.e., base templates, task descriptions, examples, and context snippets, that maintains consistent prompt structure across different authoring tasks including chapter creation, text optimization, content revision, text summarization, and prompt creation for the RIM module. Prompt engineering patterns include systematized few-shot learning with examples embedded directly in templates, contextual boundary markers that clearly delineate different information sources, and dynamic context window management that preserves semantic coherence while respecting token limitations. For LLM inference through the LSM, where model specifics like hyperparameters and model names are configured in the `application_properties.yaml` file, the module employs quality verification mechanisms that detect and reject malformed responses through pattern recognition, with automatic retry strategies

and response sanitization pipelines that remove artifacts like instruction markers while preserving semantic content.

5.1.4.2 Research and Investigation Module

The RIM serves as HAIMEDA's specialized research assistant connected directly with the chat assistant interface, providing hybrid information retrieval capabilities that combine vector-based semantic searches with symbolic pattern matching to support natural language queries about regulatory requirements, similar assessment cases, and technical specifications. The system employs a two-tier retrieval architecture that demonstrates advanced hybrid AI integration by leveraging both symbolic AI precision and vector-based semantic capabilities (see Figure 5.4).

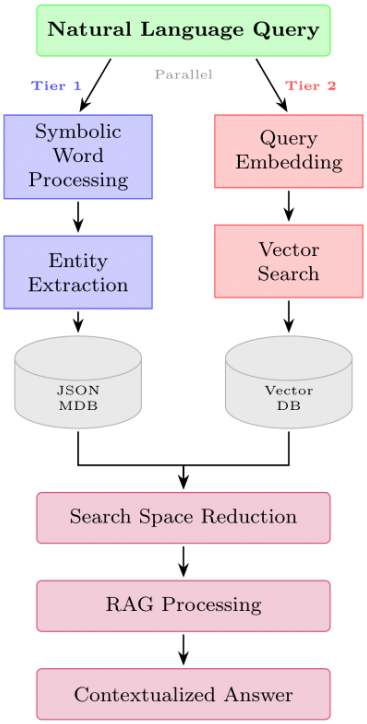


Figure 5.4: Query processing workflow in the Research and Investigation Module: parallel symbolic and vector-based retrieval pipelines reduce the search space for RAG processing, resulting in contextualized answers to natural language queries.

The system's query processing begins with the `SymbolicWordProcessor` component, which performs entity-aware query decomposition using over 40 specialized regex patterns that extract structured entities including dates, identifiers, quantities, and stakeholder information from natural language questions. This symbolic pre-processing phase strategically narrows search spaces by mapping extracted entities to database schema locations through rule-based pattern matching and fuzzy matching with configurable similarity thresholds, targeting JSON files containing data previously stored in tables from the surveyor's Microsoft Database. For database searches, the `QueryProcessor` executes dynamically constructed rule-priority queries that integrate high-priority table columns before low-priority ones, supporting comparison operators including exact match, contains, and range queries with type-specific handling. The system implements intelligent record joining across multiple tables based on common fields while filtering and normalizing query fragments to remove noise from datatype inconsistencies, enabling retrieval of information such as contact information for stakeholders or statistics about prior damage cases.

The vector search component leverages the `VectorPersistence` component to maintain document embeddings through a comprehensive pipeline that processes multiple document formats and implements context-preserving chunking strategies, e.g., one chapter per chunk, that respect sentence boundaries, position-aware chunk management that tracks document hierarchies, and sophisticated similarity measurement using cosine distance with character count-based filtering. The `VectorMaintenance` subsystem synchronizes document changes with the vector database. It automatically detects and reprocesses modified documents to ensure the embeddings are up to date. During execution, parallel tasks simultaneously extract symbolic entities and generate embeddings through the embedding model. When report IDs are identified through the symbolic analysis phase, the system triggers RAG processing on corresponding report chapters, with results transformed through the `ResponseGenerator` into contextualized answers that integrate both structured database findings and semantically similar document passages, providing a comprehensive demonstration of how symbolic precision can enhance vector search effectiveness in domain-specific RAG implementations.

5.1.4.3 Information Integrity Verification Module

The IIVM implements a two-stage verification architecture that exemplifies advanced hybrid AI integration through complementary symbolic and sub-symbolic approaches (see Figure 5.5). This architectural design addresses the fundamental challenge of ensuring information integrity in AI-generated medical device assessment reports through rigorous mathematical verification combined with neural semantic understanding.

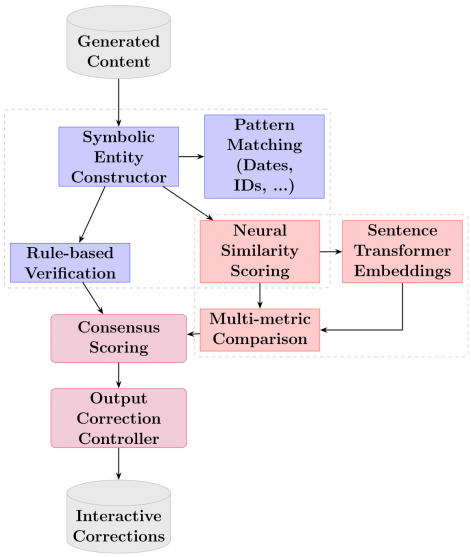


Figure 5.5: Hybrid verification workflow in the Information Integrity Verification Module: symbolic entity extraction and rule-based verification are combined with neural similarity scoring and multi-metric comparison to enable output corrections for LLM-generated content.

Pre-Processor: Symbolic Input Validation

The *Pre-Processor* submodule establishes a purely symbolic foundation using formal tableaux logic to validate input data before LLM processing. Operating through a multi-stage symbolic reasoning pipeline, it implements a theorem-proving system with rigorous set-theoretic foundations where conditions are represented as elements within carefully defined universes including satisfied conditions, evaluated conditions, and categorized condition collections. The implementation transforms JSON ruleset definitions into formal logical expressions that detect complex interdependent violations through a hierarchical condition taxonomy spanning core conditions for basic data validation, meta-conditions for logical relationships between conditions, and aggregate conditions for statistical properties across condition groups.

This formal verification engine employs distributive laws and logical simplification rules to construct dynamic proof trees through systematic decomposition, evaluating complex logical constructs including quantified statements, attribute-based calculations, and triggered actions. Unlike traditional rule engines, the tableaux-based verification implements context-sensitive validation through chapter-type awareness, automatically applying different validation rules for specialized report sections while maintaining logical consistency across verified elements. This rigorous symbolic foundation establishes mathematical certainty regarding input validity, providing a complementary approach to the neural language generation components while specifically targeting data integrity concerns before LLM processing occurs.

Post-Processor: Hybrid Content Verification

The *Post-Processor* submodule extends beyond purely symbolic approaches by integrating neural language models for content verification. Operating after LLM generation, this component implements a multi-stage entity detection and verification pipeline through the `HybridVerificationEngine`, creating a sophisticated bridge between deterministic symbolic processing and probabilistic neural understanding.

Entity Extraction and Pattern Recognition The `SymbolicEntityConstructor` systematically extracts structured entities from both input and generated content through specialized pattern matching algorithms. The system supports comprehensive date recognition across more than fifteen formats,

including German and English variations, and generates derivatives for improved matching accuracy. Number patterns handle financial amounts, quantities, and percentages with locale-specific formatting, while identifier patterns cover legal references, technical codes, and international standards. The classification distinguishes between phrase entities containing fewer than four words and statement entities comprising longer sentences and clauses requiring semantic verification.

Neural-Symbolic Verification Strategy For definitive entities such as dates and identifiers, verification employs rule-based symbolic comparison with cross-referencing between input and output representations. Complex content verification, particularly for statements, integrates Python-based neural language models via the Gateway API. The system utilizes *SentenceTransformer*[5] with the `paraphrase-multilingual-MiniLM-L12-V2`[6] model for generating multilingual embeddings optimized for German and English technical content. Multi-dimensional similarity scoring implements four complementary algorithms: cosine similarity with sentence embeddings for primary semantic measurement, TF-IDF vector comparison for lexical overlap detection, Manhattan and Euclidean distance metrics converted to percentage scales, and domain-sensitive keyword weighting specialized for technical language patterns.

5: https://sbert.net/

6: https://huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2

Parallel Processing Architecture The `StatementWorkerPool` implements an intelligent three-phase parallel processing pipeline that dynamically scales vector processing based on available GPU resources. The architecture begins with batch preparation pre-computing embeddings for all unique statements, proceeds through parallel comparison execution using `ThreadPoolExecutor` with adaptive thread allocation, and concludes with result aggregation maintaining original comparison pair ordering. Performance optimizations include singleton model management for efficient resource utilization, incremental garbage collection to prevent memory accumulation, and CUDA cache management for GPU-accelerated processing stability.

Interactive Correction Framework The `OutputCorrectionController` generates interactive correction interfaces through systematic visual feedback mechanisms. Green highlighting indicates correct entities while offering alternative options, violet marking denotes automatically replaced entities with system-applied corrections, and red emphasis identifies potentially false entities requiring user verification with replacement alternatives. The system produces TipTap editor-compatible JSON output maintaining complete entity provenance tracking, preserving both original and replacement content with associated confidence scores and source attribution. State management through Elixir Agents ensures verification context persistence across the entire pipeline without requiring explicit parameter passing between components.

5.2 Application of the Development Lifecycle Phases

This section documents the framework phases and elements applied in HAIMEDA development. Rather than presenting all implemented modules and iterations, which would exceed scope and create confusion, only essential project-related elements are presented, including development iteration summaries for key application modules.

Phase 1: Assessment & Planning

E 1.1 Domain & Requirements Analysis

Following the framework's principle of hierarchical requirements structuring, multiple structured interviews with medical device surveyor Walter Sigloch systematically identified and prioritized system requirements. These interviews explored the daily assessment workflows, practical challenges, and data sensitivity considerations introduced in Section 3.2.1.3. The following sections outline the mutually assessed primary and secondary objectives, followed by essential details on workflow, domain specifics, implementation considerations, and data sensitivity aspects.

Primary Objectives

Through stakeholder analysis, the following critical requirements were established:

- Create an intuitive editor environment supporting chapter-by-chapter report development for efficient medical device assessment report creation
- Implement AI-assisted content generation that expands minimal surveyor input into comprehensive report sections while maintaining expert control
- Ensure all processing operates locally without external data transmission, maintaining confidentiality of sensitive medical device information
- Integrate verification mechanisms ensuring factual accuracy and integrity of AI-generated content

Secondary Objectives

Additional requirements supporting primary functionality include:

- Provide structured mechanisms for documenting stakeholder statements from manufacturers, notified bodies, and regulatory authorities
- Implement a conversational interface enabling general inquiries and domain-specific research capabilities
- Support exploration of historical cases and identification of similar assessment scenarios

- Create systems for maintaining and integrating device metadata, including case details and technical specifications
- Develop a specialized utility for calculating device time values

The surveyor's domain-specific workflow, detailed in Section 3.1.3, revealed key AI integration points through workflow mapping. These include AI-assisted report creation using LLMs for generation with symbolic or hybrid AI components for information integrity verification, a RAG-based research assistance system enhanced by symbolic AI for efficient search space reduction, and integration with the surveyor's Access database for retrieving information potentially absent from existing reports. Data sensitivity assessment established a single high-sensitivity classification for all application data, including current and historical reports. This classification mandates local processing for both development and production environments, eliminating complex sanitization protocols while ensuring confidentiality.

The application design recognizes that sub-symbolic AI systems, particularly Natural Language Processing (NLP) technologies, excel at text refinement and optimization over generating complete documents from minimal input [Shi+23, pp. 508ff]. The interface therefore supports collaborative interaction where surveyors provide structured input and context, while AI assists in expanding, refining, and organizing this information into professional report chapters.

Current technological and local hardware constraints, examined in Section 3.2.2.2, necessitate chapter-by-chapter report creation rather than single-operation complete report generation. This approach addresses context size restrictions and reduces inaccuracy risks when processing extensive documents with language models [Ji+23, pp. 248:15, 248:21]. Sequential chapter creation aligns naturally with the surveyor's workflow while maintaining manageable scope for each AI-assisted task.

E 1.2 Data Analysis

Analysis of the available report corpus confirmed the appropriateness of using individual chapters rather than full reports for fine-tuning. This approach addresses token length limitations while yielding approximately 10 000 chapters and subchapters from 899 reports, providing a moderate dataset size suitable for effective model training. The consistent layout patterns and recurring chapter structures across reports further support fine-tuning as a viable adaptation strategy.

For information integrity, verification efforts focus on key structured elements such as dates, numbers, and identifiers, where errors would indicate significant accuracy problems. This verification strategy enables two distinct symbolic AI integration opportunities: pre-processing to ensure sub-symbolic component inputs contain essential reference data, and post-processing to verify key factual elements in model outputs.

The human expert performs final validation of the partially AI-corrected content.

E 1.3 Infrastructure & Resource Planning

As detailed in Section 3.2.1.2, the expert provided an NVIDIA RTX 4090 GPU for model fine-tuning and application execution. This hardware constrains viable model sizes to approximately 8-14B parameters when using FP16 precision for model loading, with fine-tuning output adapters limited to 8-bit quantization or lower due to memory constraints. Since the expert's environment consists primarily of Windows machines, with the final application requiring native Windows functionality, all development occurred within the Windows OS.

E 1.4 System Outline

Dedicated modules address each functionality specified in the primary and secondary objectives, as detailed in Section 5.1: RAM for LLM-based report generation, IIVM for ensuring content accuracy, RIM for querying prior reports and MDB content, a comprehensive editor environment with rich text capabilities, and metadata management components for documenting stakeholder, device, and report-specific information.

Summary of Iterative Development Cycles

Following the framework structure, application development proceeded through iterative cycles of Phases 2 through 4, each focusing on specific modules and functionalities. The following section presents selected highlights that emphasize key development milestones, significant challenges, and architectural evolution across HAIMEDA's iteration cycles, as illustrated in Figure 5.6. Thereby, representative examples from critical development stages demonstrate how framework principles guided implementation decisions and system refinements.

Cycle 1: Testing Techniques and Preparation

The initial cycle implemented the HATR prototype introduced in Section 3.2.2.2 to validate key technologies for the final application. Critical frameworks and libraries including LangChain, Ollama, and ErlPort were tested to confirm functionality in the production environment. An early version of the symbolic AI verification system was developed, later evolving into the Pre-Processor submodule of the IIVM module. This prototype phase examined techniques for orchestrating multiple sub-symbolic AI components through a central controller while implementing comprehensive logging and request versioning. A resource management component was developed to intelligently detect available hardware resources,

select appropriate models based on system constraints, and implement retry mechanisms for Ollama LLM requests. Testing revealed a significant LangChain library issue where runtime parameters such as temperature, max_*p*, and max_*k* were not reliably transmitted to the Ollama client. Additionally, experiments on selected pre-trained LLMs were conducted, examined in detail in Section 5.3.1.

Cycle 2: Data Preparation and First Fine-Tuning

The second cycle focused on preparing and partitioning the surveyor's historical report corpus to create structured instructions for fine-tuning. This process necessitated the development of specialized tools for data preparation, information extraction, and dataset building, that were later formalized as the framework toolkit components. The comprehensive first exploratory fine-tuning methodology developed during this iteration is detailed in Section 5.3.2.

Cycle 3: MainController and Editor Environment

With the initial fine-tuned model completed, development shifted to establishing the application's architectural foundation. This included implementing the editor environment, creating GUI components for event triggering, and developing the `MainController` module as the orchestration layer coordinating all module interactions. To fulfill the secondary objective of documenting stakeholder statements and case data, a tabbed interface was implemented featuring a left-side navigation panel providing access to stakeholder information, device metadata, case data, and report chapters through an efficient tab management system.

Cycle 4: Refined Fine-Tuning and RAM Implementation

A second fine-tuning iteration significantly improved upon the initial approach, addressing previous limitations and enhancing output quality. The RAM module and its prompt template engine were developed in close alignment with updated fine-tuning instruction datasets to ensure consistency between production and training environments. During this phase, the framework toolkit's Model Fine-Tuning and Model Integration Tools were implemented. The complete refined fine-tuning approach with outcomes is detailed in Section 5.3.3.

Cycle 5: LSM and FSM Implementation

Following RAM module completion, the LLM request functionality was extracted into a dedicated LSM, enabling standardized language model access for all application components. Concurrently, the FBM was implemented to provide structured communication channels for displaying error messages, results, and AI responses in both the status log and chat assistant interface.

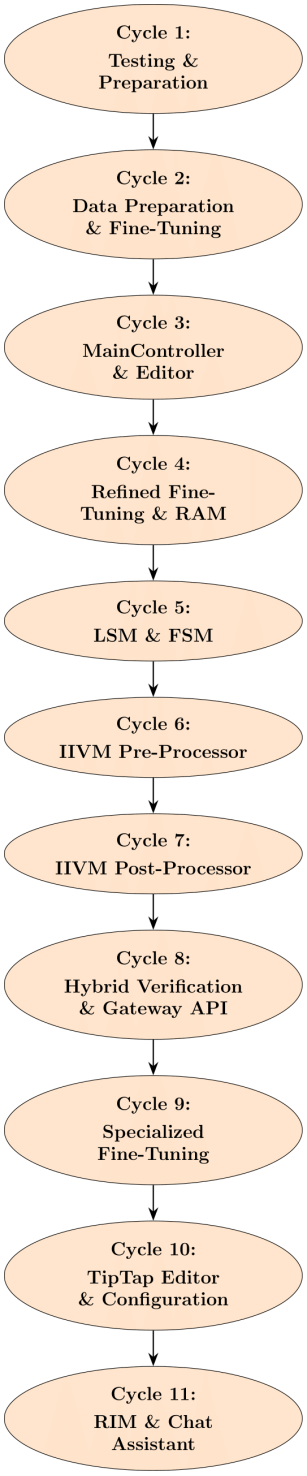


Figure 5.6: Overview of the eleven iterative development cycles in HAIMEDA: each cycle completed the lifecycle Phases 2 through 4.

Cycles 6-8: IIVM and Gateway API Implementation

Cycle 6 integrated the Pre-Processor submodule of the IIVM by adapting the symbolic AI system from the HATR prototype. This implementation required modifications to enhance input quality verification before LLM processing, improving the likelihood of effective LLM outputs.

Cycle 7 initially implemented the Post-Processor submodule using the SMT solvers Z3 [MB08] and CVC5 [Bar+22] via Python modules through ErlPort. This approach aimed to validate consistency between LLM inputs and outputs by checking entities such as dates using *SMT-libV2*[7] statements. However, significant limitations emerged: the SMT provers could only process limited input lengths, necessitating complex batch processing. The situation worsened when implementing comprehensive entity detection rules for format variations, as the maximum batch size decreased significantly, requiring numerous processing batches and producing complex SMT-LibV2 statements that became difficult to read and debug.

7: https://smt-lib.org/language.shtml

Given these limitations, Cycle 8 replaced the SMT solver approach with a custom hybrid verification system. This implementation uses regex patterns and symbolic rules to detect entities such as dates, numbers, IDs, or phrases in both input and output content, collecting and comparing them systematically. For complex content, particularly statements containing more than four words, the system employs sub-symbolic similarity measurements via Python modules to generate comparison scores, effectively combining symbolic rule-based detection with neural language model capabilities. For this purpose, the GAM was created as a generalized Elixir-to-Python interface, enabling any Python module function calls while providing error handling and environment validation capabilities.

Cycle 9: Further Refined Fine-Tuning

Cycle 9 implemented a sophisticated fine-tuning strategy using two parallel instances of the same LLM architecture. The first instance trained exclusively on knowledge acquisition from historical reports, while the second focused on structural formatting and chapter organization. These specialized models were merged using MergeKit's SLERP method to create a unified model combining both capabilities. This cycle also addressed accumulated minor bugs and implemented cost optimization improvements across the application. The complete methodology and outcomes of this specialized fine-tuning approach are documented in Section 5.3.4.

Cycle 10: TipTap Editor and Configuration System Implementation

The simple text field editor was replaced with a TipTap editor implementation via JavaScript to enable advanced content verification feedback capabilities. This implementation supports colored text formatting and drop-down menus with alternative selections and replacement options

for entities identified by the `OutputCorrectionController` component of the Post-Processor submodule. The editor also displays missing entities, those present in input data but absent from generated output text, providing comprehensive verification feedback and simplified content editing actions for surveyors.

Development faced significant technical challenges in ensuring proper rendering of content containing rich text embeddings and achieving seamless integration between JavaScript components and the Elixir backend. These obstacles required substantial architectural adjustments to ensure cohesive functioning of interface components within the application's rendering pipeline.

A comprehensive properties configuration system using the `application_properties.yaml` file was implemented, allowing users to preset critical application parameters including preferred models for specific AI interactions, model runtime settings, and various other application configurations. These settings load automatically when opening reports and entering the editor environment. The configuration system enhanced development efficiency by enabling rapid testing through configuration toggles that could disable time-consuming operations.

Cycle 11: RIM Implementation and Specialized Chat Assistant

The final development cycle incorporated the RIM implementation to fulfill one of the project's secondary objectives. The process began by further fine-tuning the knowledge-focused model variant from Cycle 9 using a specialized supervised instruction dataset. This dataset contained Question-Answer (QA) pairs constructed from shuffled report contents to simulate RAG system interactions with actual surveyor queries, as described in detail in Section 5.3.4.

The module processes questions through a hybrid approach. Here, a symbolic AI engine first divides queries into individual words searched across the surveyor's Access database, previously extracted to JSON files and loaded into memory for efficient runtime querying. Using rule-based pattern matching and regex processing, the system identifies relevant report IDs or specific database entities such as client contact information. These report IDs strategically narrow the search scope for the RAG system, significantly improving response accuracy while reducing context size requirements, a critical factor when using smaller language models with limited context windows. Although the RIM implementation is functional, it requires further refinement on symbolic extraction accuracy.

Phase 5: Deployment & Continuous Improvement

E 5.1 Production Environment Setup & Deployment

Following Cycle 11 completion, minor adjustments were implemented before deploying the application to a dedicated test PC equipped with

an AMD Radeon RX Vega 64 GPU. The GPU's 8GB VRAM limitation required quantizing the fine-tuned models to 8-bit precision using the framework's Model Integration tools. All necessary dependencies were installed, including Elixir 1.18.4 on Erlang 27, required Elixir libraries from the Mix project, and the Ollama runtime environment. Functionality verification confirmed similar performance to the development environment.

This configuration serves as a temporary testing platform for collecting surveyor feedback and identifying new requirements. Final deployment will utilize the surveyor's Windows virtual machine in VMware Workstation Pro, where report creation was previously conducted, with the RTX 4090 GPU currently used for development attached to enable running fine-tuned models in full FP16 precision without quantization. Outstanding refinements for the next development iteration include implementing the comprehensive monitoring and logging system recommended by the framework to collect technical metrics and usage statistics.

E 5.2 User Testing & Feedback Collection

Two testing sessions have been conducted with the medical device surveyor to date. The initial session demonstrated the application's functionality and intended workflow to the surveyor. After becoming familiar with HAIMEDA, the surveyor participated in a feedback interview to collect refinement suggestions and improvement opportunities, which is further detailed in Chapter 6.

E 5.3 Continuous Monitoring & Improvement

While the application is functional, several planned improvements remain outstanding. These include implementing a specialized module for calculating medical device time values using statistical calculations and metadata like purchase price and acquisition year. New requirements emerged during development and user testing, including the ability to export reports to MD files and potentially connecting to a template-based document generation model that produces reports with identical formatting to the surveyor's standard in Microsoft Word Macro-Enabled Document (DOCM) format. Performance limitations of the fine-tuned Llama 3 8B models have prompted plans for additional fine-tuning with larger models, contingent on surveyor consent to utilize remote GPU resources.

5.3 LLM Fine-tuning Methodology for HAIMEDA

This section details the fine-tuning methodology for creating task-specific LLMs as major sub-symbolic components of the HAIMEDA application

(see Figure 5.7). The main objective was to enable efficient chapter-wise content generation from minimal input, with additional context such as previous chapters and device metadata improving accuracy. A secondary objective involved training a chat assistant LLM for research-related queries and information retrieval from prior reports and the surveyor's MDB. The following sections describe the iterative fine-tuning phases, approaches, and toolkit tools used, highlighting differences in data preparation, information extraction, dataset building, fine-tuning strategies, and evaluation results.

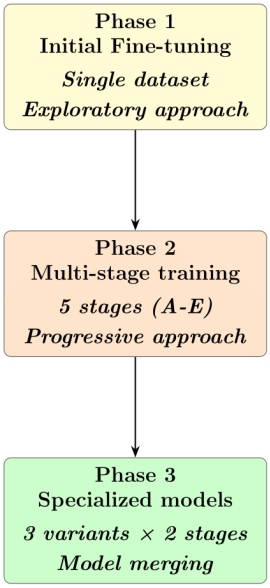


Figure 5.7: Overview of the three-phase fine-tuning methodology.

5.3.1 Preliminary Considerations

Since the final application is delivered in German, with all clients and prior reports using this language exclusively, selecting an LLM with existing German language capabilities was essential. The choice focused on smaller models that could run on the surveyor's local hardware and be fine-tuned on an RTX 4090 GPU without extensive quantization. Several German pre-trained LLMs from the HuggingFace repository were considered: Mistral 7B and Llama 2 13B models pretrained by the LeoLM team, and a Llama 3 8B IT variant by DiscoResearch.

After extensive testing on general and medical device survey-related tasks including question-answering, summarization, and text generation, the Llama 3 8B IT model was selected as the primary base model for fine-tuning and for use in the *Sub-Symbolic Extractor* introduced in Section 4.4.2. Full hardware specifications appear in Table 3.1 in Section 3.2.2.2.

Initial tests examined the models' capability to process full reports, but their extensive size exceeded the context length limitations of 8B models. This led to using individual chapters for fine-tuning instead of complete reports, which additionally created significantly more fine-tuning instructions than would have been available using full reports alone.

5.3.2 Phase 1: Initial Fine-Tuning Approach

The first phase served as an exploratory test to assess the capabilities of a single small model trained on the reports. During this phase, most framework toolkit tools were developed, with many subsequently refined and completed in later phases. The key stages of this initial phase are outlined in the following subsections.

Data Processing Stages

Data Preparation Using the toolkit tools described in Section 4.4.1, the surveyor's 916 previously written reports were initially prepared. These reports were stored in client-categorized folders with varying structure, sometimes nested, and containing different timeline versions. The *File Extraction* tool first identified, extracted, and isolated all final reports for further processing. The *File Filtering* tool then removed 17 outlier reports that were either statements to reports or contained minimal text without

chapters, leaving 899 reports. These reports were in DOCM format, which is unsuitable for both programmatic processing and creating fine-tuning instructions. The *File Conversion* tool converted these files to MD format. The *Data Partitioning* tool then separated title pages, individual chapters, and references from each report into separate files stored in dedicated folders per report. During this process, chapter texts were normalized and cleaned of conversion fragments from the DOCM to MD conversion.

Information Extraction The *Information Extraction Tools* presented in Section 4.4.2 extracted and generated additional data. The *Symbolic Extractor* extracted relevant report-specific metadata and stored this information in dedicated `extracted_meta_info.json` files within each report's directory. The *Sub-Symbolic Extractor* created summaries for each chapter and generated QA pairs specific to chapter types. The *Statistics* tool extracted all possible chapter names, categorizing each chapter into one or multiple categories, with tailored questions created for each category. Summaries, questions, and answers for each report's chapters were stored in a `extracted_summary.json` within the respective report directory.

Dataset Building A single supervised instruction dataset was constructed for the initial phase, containing one instruction per chapter. Each instruction used previously extracted metadata, questions and answers, and summaries as input values, combined with different task description variants instructing the LLM to produce chapter text. Normalized chapter text with titles served as output values. While the *Dataset Building Tools* were later removed from the official framework, they remain available for reference in the `Idea & Testing` directory within the Dataset Building Tools on the framework's GitHub repository.

Fine-Tuning Strategy

Phase 1 implemented a PEFT approach using LoRA to maximize GPU memory while preserving model capabilities. The strategy created comprehensive instruction templates blending multiple information sources tailored to chapter types and requirements. Each template systematically combined chapter metadata selected by category, chapter summaries, relevant QA pairs, and contextual information from related sections to establish content continuity. Different prompt formats were developed for regular chapters versus technical specification sections, with formatting appropriate to each content type.

To prevent instruction memorization, the strategy employed seven distinct task description variants maintaining consistent objectives while varying phrasing and structure. Training parameters were deliberately conservative with $r = 16$, $= 32$, and learning rate 2×10^{5}, using a maximum sequence length of 3 000 tokens to accommodate complete chapter examples with rich contextual information. Complete hyperparameters appear in Appendix Table D.4.

This experimental approach tested how much context the model could effectively utilize and which information types most improved output quality. The strategy intentionally provided more information than strictly necessary to identify elements most significantly impacting generation quality, establishing a baseline for targeted approaches in subsequent phases while revealing insights about context handling and response formatting.

Results and Limitations

Despite achieving substantial evaluation loss reduction of 75.6 % (from 1.626 to 0.396), as shown in Table 5.2, Phase 1 fine-tuning revealed a critical disconnect between quantitative metrics and practical utility. Inference testing demonstrated that the model predominantly generated only chapter titles without corresponding content, particularly at temperature settings below 0.7, a key technical challenge documented in Appendix D.2.1.3.

Table 5.2: Summary of fine-tuning evaluation metrics for Phase 1.

Metric	Value
Initial Loss	1.626
Final Loss	0.396
Loss Reduction	75.6 %

Root cause analysis identified the instruction structure as the primary issue. Output examples began with chapter titles followed by newlines before main content, creating an artificial prediction boundary that bifurcated the training objective. The model optimized for the simpler task of title prediction rather than generating coherent chapter content, highlighting the opaque nature of language model training dynamics [Lip18, pp. 36f, 39f]. Temperature sensitivity testing revealed that settings between 0.7 and 1.0 produced complete chapters only about 15 % of the time, suggesting the model had learned a strong probabilistic boundary at newline characters.

These findings directly informed subsequent fine-tuning phases by demonstrating how seemingly minor formatting decisions in training data can significantly impact model behavior in ways traditional metrics fail to detect [RSG16, pp. 1136ff]. Comprehensive technical analysis and evaluation metrics of Phase 1 appear in Appendix D.2.1.

The insights from Phase 1 shaped Phase 2 development, particularly the need to redesign instruction formatting to eliminate artificial stopping points and implement explicit prompting techniques that discourage optimization for simpler prediction tasks.

5.3.3 Phase 2: Multi-Stage Fine-Tuning and Optimization

After Phase 1 revealed that minor instruction design differences significantly impact model output, the instruction templates were adjusted and a new multi-stage strategy was developed. This approach began with unsupervised fine-tuning to strengthen the model's domain knowledge and its ability to generate chapter content using professional medical device surveyor terminology. The expert's MDB served as a foundation, containing keywords distributed across reports such as manufacturer names and device information. The strategy aimed to enhance the model's contextual understanding of chapter sequencing and intelligently utilize the maximum token length for instructions, with some cases including

multiple chapters within single instructions. The following sections detail this approach.

Data Processing Stages

Data Preparation The Access database was prepared using the *MDB Data Extraction* tool, which extracted, formatted, and converted all tables to JSON and JSONL files. During conversion, some values were unrecoverable and certain data types were mixed. Report chapters underwent the same preparation process documented in Phase 1 (Subsection 5.3.2), with the addition of splitting chapters into subchapters at nested objects. This enabled more granular chapter production and reduced model confusion about when to produce nested sections. The *Following Chapters Preparation* tool was used to prepare multiple sequential chapters.

Information Extraction Metadata extraction using the *Symbolic Extractor* tool was enhanced with additional patterns to capture more data, and the strategy for adding metadata to datasets was revised. The *Metadata Processor* tool determined exactly which extracted metadata appeared in specific chapters, specifying this information in templates for dataset building tools. The *Sub-Symbolic Extractor* created summaries for unnested subchapters, while Statistics submodule functions identified potentially malformed LLM outputs. These outputs either underwent repeated summary creation or were excluded entirely from instruction datasets if they contained less than 10 characters of chapter content.

Dataset Building Five specialized fine-tuning stages were implemented using three unsupervised and two supervised datasets. The *Unsupervised Datasets* toolset created the first three datasets: MDB table data prepared as instructions, individual chapters, and a smaller dataset containing multiple sequential chapters. The *Supervised Datasets* tools then created two structured datasets with consistent formatting. Input values included task variations for chapter writing, specific metadata extracted per chapter, chapter numbers and titles, chapter summaries, and contextual information from preceding chapters. The first supervised stage used full chapters as context, while the second used summaries to enhance the model's ability to produce coherent content from summarized information and limited details. Questions and answers were excluded to focus solely on report chapter generation. Output values contained plain chapter content without titles or chapter numbers, correcting the formatting issue from Phase 1.

Fine-Tuning Strategy

Phase 2 implemented a progressive multi-stage approach with deliberately planned parameter variations across five distinct stages, as presented in Table D.7. The strategy began with three unsupervised stages (A–C) focused on domain knowledge acquisition before transitioning to supervised instruction following in Stages D–E, effectively separating factual knowledge from structural formatting.

Each stage selectively modified key parameters based on observations from previous stages. Sequence length gradually increased from 1 100 tokens in Stage A to 3 200 in Stages D and E to progressively expand context handling capabilities. Content complexity evolved from basic unsupervised formats like MDB data to more sophisticated supervised structures. Targeted hyperparameter adjustments included reducing learning rates from 2×10^{-5} to 1.5×10^{-5} and increasing LoRA dimensions from $r = 16$, $= 32$ to $r = 24$, $= 48$ for specialized instruction formats.

Stage D introduced a novel technique where training began with mixed instructions containing both summaries of previous chapters and full previous chapters as context for one epoch before switching exclusively to full-chapter formats, balancing content diversity with structural consistency. After each stage, the resulting LoRA adapter was merged with the base model to create the foundation for subsequent stages, except for Stage E which used Stage D's checkpoint directly. The supervised stages implemented progressively increasing learning rate warm-up ratios to prevent catastrophic forgetting of previously acquired knowledge during specialized format training.

Results and Limitations

Phase 2's progressive multi-stage approach produced substantial quantitative improvements, with each stage showing distinct performance characteristics detailed in Table 5.3. Stage A using unsupervised MDB data achieved 71.7 % loss reduction, Stage B with single chapters showed 62.5 % improvement, while Stage C with multiple chapters demonstrated minimal 4.0 % additional loss reduction, suggesting diminishing returns for unsupervised training. The supervised stages showed marked improvements, with Stage E achieving the best final loss of 0.322, representing 74.4 % reduction.

Table 5.3: Summary of fine-tuning evaluation metrics for Phase 2.

Stage	Initial Loss	Final Loss	Loss Reduct.
A	2.433	0.688	71.7 %
B	2.039	0.765	62.5 %
C	0.666	0.639	4.0 %
D	1.309	0.478	63.5 %
E	1.258	0.322	74.4 %

Note: Stages A–C unsupervised, Stages D–E supervised training. Bold values show best results.

Testing revealed significant functional improvements over Phase 1, with the model reliably generating complete chapter content rather than only titles. However, several limitations persisted, particularly inconsistent structural coherence across different chapter types. As detailed in Appendix D.2.2.3, these issues stemmed from multiple factors: catastrophic forgetting during sequential fine-tuning [Kir+17, pp. 3521f], multi-format training inconsistencies from tabular data inclusion [Che+23, pp. 1, 4f], and competing optimization goals between content accuracy and structural consistency [MGS23, pp. 1f].

The minimal improvement in Stage C provided a key insight that diminishing returns for unsupervised training after sufficient domain knowledge acquisition suggest supervised fine-tuning offers more efficient improvement pathways. Comprehensive technical analysis and evaluation metrics appear in Appendix D.2.2.

5.3.4 Phase 3: Specialized Model Training and Merging

To address the limitations discovered in Phase 2, a new strategy was introduced to create a model capable of generating chapter content with

appropriate medical device terminology while conforming to proper chapter structure and length. Two specialized variants were developed: variant V1 focused on medical device domain knowledge acquisition, while variant V2 concentrated on producing reports with correct structure and appropriate length. These models were merged into variant V3 using MergeKit's SLERP method, aiming to combine the strengths of both variants.

Using the V1 variant as a base, a further fine-tuned model V1-B was created using previously gathered QA data for individual chapters. This specialized model serves as a chat assistant within the RIM module to answer surveyor questions. The following sections detail this merged model approach and V1 variant specialization.

5.3.4.1 Data Processing Stages

Data Preparation and Information Extraction Besides minor adjustments to chapter content formatting and normalization, no additional data was extracted. The QA data per chapter from Phase 1 was extracted from each `extracted_summary.json` file, formatted, and saved in a new `finetuning_layout.json` file per report directory.

Dataset Building The unsupervised datasets from Phase 2 containing single and multiple report chapters were merged into a single mixed instruction dataset using the *Mixed Chapters Datasets* tools. The supervised instruction datasets from Phase 2 were reproduced with refinements such as improved outlier exclusion using the same tools.

A new supervised instruction dataset was created using the *Q&A Datasets* tools with the prepared QA data. This dataset included different task variants for answering user questions alongside chapter contents as input values. Chapter contents were separated by paragraphs and shuffled to simulate a real-world RAG scenario where relevant chapter parts are ordered by embedding similarity rather than sequential appearance. The dataset explicitly included unanswerable questions to train the model to recognize when available references cannot address a query. Output values consisted of answers to posed questions, with varying statements indicating when questions could not be answered using the given material.

Finally, two coherence datasets were created using the *Coherence Datasets* tools to harmonize potentially discontinuous knowledge representations from model merging. These datasets extracted previously unseen instructions from validation and test sets, with one dataset for unsupervised instructions and another for supervised instructions.

5.3.4.2 Fine-Tuning Strategy

Phase 3 implemented a specialized approach using parallel model variants, each optimized for different report generation aspects: variant V1 focused on domain knowledge acquisition through unsupervised training, while variant V2 concentrated on document structure and formatting

via supervised training with sophisticated instruction templates and complementary hyperparameter configurations, as detailed in Table D.10.

These models were merged using SLERP through MergeKit to create variant V3, which combined domain expertise with strong formatting capabilities. The merged model then underwent coherence-focused fine-tuning with selective layer unfreezing of layers 8 through 14 and 20 through 25, targeting transitional boundaries between transformer layer hierarchies to facilitate seamless knowledge integration across the architecture [Jia+25, pp. 2f, 6f].

After completing V3 for chapter generation, the original V1 model was specialized into V1-B for the RIM module through supervised question-answering instruction tuning on QA datasets constructed from report fragments. This parallel development approach with subsequent model merging represented a significant advance over previous sequential training strategies, reducing catastrophic forgetting [God+24, pp. 477f].

5.3.4.3 Results and Limitations

Phase 3's specialized model approach demonstrated varied quantitative results across variants, as illustrated in Table 5.4. The most substantial improvements occurred in V1-B with 86.9 % loss reduction for question-answering tasks, while moderate improvements were achieved in domain knowledge with V1-A showing 66.1 % reduction and formatting capabilities with V2-A achieving 75.9 % reduction. The merged V3 model successfully generated appropriate chapter length and structure, addressing key limitations from Phase 2, but introduced new challenges documented in Appendix D.2.3.3. Most notably, an increased tendency toward hallucinations emerged when using temperatures below 0.2, potentially stemming from the model merging strategy or selective layer unfreezing approach. The specialized V1-B variant exhibited format-specific performance limitations, performing significantly better with textual content than tabular MDB data, reflecting cross-format generalization challenges despite strong overall metrics [Che+23, pp. 1ff]. While the parallel development approach with specialized optimization objectives advanced beyond sequential training, these findings highlight persistent trade-offs between generative capability and factual precision requiring attention in future fine-tuning efforts. Comprehensive technical analysis and evaluation metrics appear in Appendix D.2.3.

Table 5.4: Summary of fine-tuning evaluation metrics for Phase 3.

Model	Initial Loss	Final Loss	Loss Reduct.
V1-A	1.865	0.633	66.1 %
V1-B	1.467	0.192	86.9 %
V2-A	1.786	0.431	75.9 %
V2-B	0.579	0.328	43.4 %
V3-A	0.749	0.694	7.3 %
V3-B	0.726	0.449	38.1 %

Note: V1 models focused on general training, V2 on specialized formats, and V3 on coherence. Bold values show best results.

6 Evaluation of Framework and Paradigm

Formative evaluation activities were integrated throughout each BIE cycle to ensure ongoing assessment and refinement, as detailed in Sections 5.2 and 5.3. These intermediate evaluations informed framework and toolkit decisions as well as guided improvements after each fine-tuning phase. Moving beyond these iterative assessments, this chapter presents the final comprehensive evaluation of both the framework and its paradigm application: Section 6.1 provides a gap analysis comparing the framework with established literature, while Section 6.2 evaluates the HAIMEDA application through system performance metrics and usability assessment with the medical device surveyor.

6.1 Evaluation of the Framework

This section presents a gap analysis that maps the framework's principles to established theories while identifying specific gaps addressed and novel contributions made in existing literature. While comprehensive mapping of all 15 framework principles appears in Table D.1 in the Appendix, Table 6.1 presents the most significant contributions, which the following subsections examine in detail, highlighting both novel approaches and important extensions to established theories.

6.1.1 Novel Contributions

The framework introduces several novel contributions addressing critical gaps in hybrid AI development. The *Identify Hybrid Integration Points Early* principle establishes a proactive methodology for identifying complementary AI technique combinations during initial planning phases, rather than as reactive solutions to implementation challenges. The *Coordinate Through the Orchestration Layer* principle provides structured management of communication between heterogeneous AI components, preserving internal implementation flexibility while ensuring system-wide coherence. This addresses a significant difference in existing microservices patterns when applied to hybrid AI systems.

Table 6.1: Gap analysis summary: key framework principles and their theoretical foundations.

Framework Principle	**Established Theory**	**Key Gap Addressed/Extension**
Build Modular, Not Monolithic	Modular Programming [Par72]	Extends modular programming theory to accommodate heterogeneous internal module designs across symbolic and sub-symbolic AI components
Identify Hybrid Integration Points Early	Hybrid AI Systems [Del+19]	Novel contribution establishing proactive methodology for identifying complementary AI technique combinations during initial planning phases rather than as reactive solutions
Build In Privacy From the Start	Privacy Engineering [Cav09]	Extends privacy principles to accommodate specialized data flow requirements in hybrid AI training and inference while maintaining information confidentiality
Coordinate Through the Orchestration Layer	API Gateway Pattern [Ric18]	Novel contribution providing structured communication management between heterogeneous AI components while preserving internal implementation flexibility
Implement Comprehensive Monitoring	Observability Systems [Lig12]	Novel contribution creating AI-specific metrics and multi-level indicators tracking both technical performance and information integrity aspects
Evolve Based on Usage Patterns	Continuous Improvement [Dem86]	Novel approach establishing systematic mechanisms for converting real-world usage patterns into structured improvement datasets for both symbolic and sub-symbolic components

For operational aspects, the *Implement Comprehensive Monitoring* principle extends traditional observability approaches with AI-specific metrics and multi-level indicators tracking both technical performance and information integrity unique to hybrid systems. The *Evolve Based on Usage Patterns* principle establishes systematic mechanisms for converting real-world usage patterns into structured improvement datasets, enabling continuous adaptation of both symbolic and sub-symbolic components based on authentic usage rather than artificial test cases.

6.1.2 Strategic Extensions to Established Theories

The framework strategically extends established theories to address unique challenges of hybrid AI in data-sensitive domains. The *Build In Privacy*

From the Start principle adapts Privacy-by-Design concepts to accommodate specialized data flow requirements of hybrid AI training and inference, crucial for contexts where information integrity and confidentiality are paramount. The *Make Symbolic Processing Transparent* principle enhances traditional explainability approaches by specifically addressing verification methods that combine symbolic and sub-symbolic elements.

Several principles extend classical software engineering theories for AI contexts. The *Build Modular, Not Monolithic* principle advances modular programming by enabling heterogeneous internal module design, allowing symbolic and sub-symbolic components to coexist within unified architectures. The *Expect Requirements to Change* principle adapts agile methodologies to address AI development's unique uncertainties and iterative requirements, bridging traditional agile practices with model-centric workflows. The *Organize Requirements by Priority* extends goal-oriented requirements engineering by introducing data sensitivity classification as a core analysis aspect.

Data-centric principles provide crucial extensions. The *Examine Data Before Designing* principle integrates empirical data analysis as a prerequisite to system design, ensuring architectural decisions reflect actual data characteristics. The *Keep Options Open* principle adapts set-based concurrent engineering for parallel evaluation of diverse AI processing methodologies. The *Match Training to Real Use* principle refines ML theory on training/serving skew, focusing on instruction/inference consistency particularly relevant for LLMs.

Operational principles extend existing practices for hybrid AI systems. The *Coordinate Through the Orchestration Layer* principle advances the API gateway pattern to unify orchestration across heterogeneous AI components. The *Implement Comprehensive Monitoring* principle adapts observability practices with AI-specific metrics and multi-level indicators. The *Test in Real-World Conditions* principle extends contextual design with domain-specific applicability criteria, while the *Evolve Based on Usage Patterns* principle enables systematic model evolution through real-world feedback loops. These extensions collectively bridge gaps between software engineering practices, ML operations, and data privacy frameworks, providing an integrated approach tailored to hybrid AI development in data-sensitive domains.

6.2 Evaluation of the Paradigm

6.2.1 Technical Evaluation of HAIMEDA

6.2.1.1 System Performance Evaluation

System performance evaluation, conducted using the dedicated `Performance Monitor` module, measured response times and resource utilization across key HAIMEDA modules through 50 test runs per operation, with results presenting statistical averages. As detailed in Table D.2 in Appendix D.1.2,

performance metrics reveal distinct operational characteristics across the three primary modules.

The RAM module demonstrates the highest computational demands, averaging 14.16 seconds for chapter creation while consistently utilizing 94.0 % of GPU resources. This intensive GPU utilization persists across all RAM operations, reflecting the computational requirements of generative tasks using the fine-tuned V3 model. Conversely, RIM operations show variable resource profiles based on query type. RAG-based requests complete in 5.01 seconds, significantly faster than MDB data requests at 7.91 seconds, despite consuming higher CPU resources at 40.0 % versus 30.4 %. This difference highlights the efficiency of embedding-based retrieval compared to structured data querying.

IIVM operations exhibit marked efficiency variations between verification approaches. Symbolic verification processes execute remarkably quickly, completing pre-processing and basic post-processing tasks in under 20 milliseconds. However, hybrid statement verification with similarity comparison requires substantially more time, averaging 16.58 seconds. This performance differential illustrates the computational trade-off between pure symbolic verification, which focuses on structured entities like dates and IDs, and hybrid approaches performing deeper semantic analysis. System initialization requires 0.90 seconds with moderate resource consumption, establishing a baseline footprint of approximately 110 MB memory and 1 261 MB VRAM.

These metrics indicate that RAM module report generation constitutes the primary computational bottleneck, while symbolic verification adds minimal processing overhead. Hybrid verification processes, though slower than symbolic methods, remain within acceptable parameters for complex integrity checks. This performance profile supports the intended workflow design where computationally intensive generative tasks occur during initial content creation, while verification can be strategically applied using rapid symbolic checks for routine validation and comprehensive hybrid verification for critical content requiring semantic analysis.

6.2.1.2 Information Integrity Verification Module Evaluation

The IIVM underwent systematic evaluation through 100 verification runs using a test suite comprising correct statements, hallucinated content, and examples with missing information. This methodology enabled comprehensive assessment across different verification tasks and content types. A summary of the verification performance for all IIVM components is provided in Table 6.2.

The preprocessing component achieved perfect 100 % success rates across all test cases, confirming the robustness of the tableaux-based logical rules implementation. This deterministic performance reflects the formal verification methods employed, which ensure complete coverage of defined rule patterns.

Table 6.2: IIVM verification performance showing detection rates for hallucinated and missing content across all verification components based on 100 test runs per component.

Component	Hallucin. Detection	Missing Content
Preprocessor	–	100 %
Entity Verif.	83 %	90 %
Phrase Verif.	76 %	90 %
Statement Verif.	72 %	65 %

Entity verification in the postprocessing stage demonstrated variable performance characteristics. For structured entities such as numbers, dates, and identifiers, symbolic verification achieved 83 % detection of hallucinated content. This performance correlates directly with regex pattern comprehensiveness and could theoretically approach 100 % with additional pattern refinement. Phrase-level verification showed slightly lower accuracy at 76 %, reflecting the greater complexity of defining semantic rules compared to structured entity matching. Both entity and phrase verification components achieved higher accuracy of 90 % for missing information detection, indicating the system more effectively identifies omissions than insertions.

The hybrid statement verification engine employs similarity-based semantic comparison and achieved 72 % accuracy for hallucination detection and 65 % for missing statement identification using the "moderate match" threshold. This component provides configurable sensitivity through five classification grades: no match, weak match, moderate match, strong match, and exact match, enabling users to adjust verification stringency for specific use cases. Testing confirmed that moderate match settings provide optimal balance between false positives and false negatives for medical device report verification scenarios.

These results demonstrate successful integration of deterministic symbolic verification for structured information with flexible sub-symbolic techniques for statement-level integrity checking. Semantic verification of statement correctness remains challenging even with sentence transformer models, as the task requires deep contextual understanding and inference capabilities beyond current technologies [May+20, pp. 1913f]. This challenge would be significantly greater using purely symbolic methods [Dre92, pp. 197ff]. The modular verification architecture enables continuous improvement through expanded regex patterns and refined semantic rules without structural modification, providing a flexible foundation for incorporating future advances in both symbolic and sub-symbolic verification techniques.

6.2.2 User-Centered Assessment of HAIMEDA

6.2.2.1 Productivity and Task Completion Time Analysis

A controlled evaluation following the comparative task analysis methodology of Sauro & Dumas [SD09] measured time requirements for standardized report creation tasks with and without HAIMEDA assistance. The assessment excluded stakeholder contact activities, which remain constant in both workflows, while focusing on documentation, research, and quality assurance tasks where AI assistance could provide efficiency gains.

The comparative analysis, as shown in Table 6.3, revealed significant efficiency improvements in content creation tasks, with writing time reduced by 50 % when using HAIMEDA. Research activities decreased by 33.3 %, with the surveyor noting improved convenience for retrieving reference information, though basic contact detail gathering required

Table 6.3: Comparative task completion times for creating medical device assessment reports.

Task Category	Without HAIMEDA	With HAIMEDA	Time Reduction
Content Writing	6.0 h	3.0 h	50.0 %
Research Activities	3.0 h	2.0 h	33.3 %
Final Review	1.0 h	2.0 h	-100.0 %
Initial System Training	–	2.25h	–
Total Productive Time	**10.0 h**	**7.0 h**	**30.0 %**

Note: Total productive time excludes initial training time, as this represents one-time onboarding investment.

similar time. However, final review time doubled with HAIMEDA, reflecting the need for thorough verification of AI-generated content. These findings align with Van der Lee et al. [Lee+19] regarding the efficiency-accuracy tradeoff in AI-assisted content generation.

Overall, HAIMEDA reduced total productive time from 10 to 7 hours, a 30 % improvement after accounting for the one-time 2.25-hour system training investment. This efficiency gain supports current research showing that well-designed AI assistance systems improve productivity in content generation while requiring additional human oversight for verification [JA25, pp. 2f, 6f], [Hem+23, p. 461].

6.2.2.2 Content Quality Metrics

A structured quality assessment framework evaluated content generated through the HAIMEDA system, following methodology recommendations by Zhang et al. [Zha+20] for domain-specific AI output evaluation. Three key system components were assessed across multiple quality dimensions using a 5-point scale, where 1 indicated poor quality, 3 represented acceptable performance, and 5 denoted excellent results.

As shown in Table 6.4, the quality assessment revealed variable performance across system components. The RAM demonstrated consistent medium-quality output for standard chapters, with performance degradation for complex or lengthy inputs, aligning with findings from Levy et al. [LJG24, pp. 15339f]. The RIM performed well for basic reference retrieval but showed limitations with exhaustive searches requiring manual verification, a common challenge in retrieval-augmented generation systems [Pac+25, pp. 165f].

The IIVM module achieved excellent performance in structured data verification using symbolic AI, scoring 4.5 out of 5, while demonstrating limitations in complex statement verification using sub-symbolic AI at 2.5 out of 5. This performance asymmetry reflects expectations, as symbolic AI excels at structured entity alignment and deterministic matching, while sub-symbolic models remain error-prone in semantic verification

Table 6.4: Quality assessment of HAIMEDA component-generated content.

Component	Rating	Key Observations
Report Chapter Generation (RAM)		
Factual Accuracy	3.5/5	Generally accurate facts with occasional inconsistencies
Structural Coherence	3.0/5	Diminishing quality with increased input complexity
Language Appropriateness	3.5/5	Professional terminology with occasional awkward phrasing
Research Results (RIM)		
Reference Retrieval	3.5/5	Effective for routine queries but inconsistent for complex cases
Response Timeliness	2.5/5	Processing delays noted for certain query types
Comprehensiveness	3.0/5	Limited verification for exhaustive search results
Verification System (IIVM)		
Entity Detection	4.5/5	Excellent for structured data (dates, IDs, numbers)
Statement Verification	2.5/5	Limited coverage for complex phrases; requires manual review
Correction Efficiency	3.0/5	Effective suggestions but time-consuming implementation

tasks requiring nuanced contextual understanding, though they still outperform symbolic approaches in these areas [Pla25, pp. 168-171].

6.2.2.3 System Usability Assessment

The HAIMEDA application's usability was formally assessed using the SUS defined by Brooke [Bro95], with interpretation guidelines from Bangor et al. [BKM09]. The complete questionnaire and responses appear in Appendix Table D.3. The evaluation yielded a score of 62.5 out of 100, placing the system in Bangor's "Marginal/OK" usability range. This score falls slightly below the industry average of 68, indicating that while the application is functionally usable, clear opportunities for improvement remain. The assessment reflects evaluation by a single user, the medical device surveyor, as the system was specifically designed for this specialized role and use case.

6.2.2.4 Critical Assessment by the Surveyor

The critical assessment employed the think-aloud protocol described by Boren & Ramey [BR00], where the surveyor verbalized thoughts and reactions while interacting with HAIMEDA. This approach provided insights into both immediate usability impressions and deeper cognitive interactions with the system. The following surveyor quotations presented in the following sections have been translated from German.

Interface and Navigation Assessment

The surveyor found the initial start page "very overviewable and intuitively to use without guidance," noting immediate accessibility of report selection and creation functions. However, the editor environment with a freshly loaded report appeared "a bit overwhelming" due to feature density. This initial complexity diminished significantly after explanation of the structure and functionalities, suggesting a moderate learning curve for new users.

The surveyor particularly praised the side navigation panel for efficiently organizing report data and device metadata alongside the tab environment, commenting that "the tab-like style enables efficient working". The stakeholder statements section received "great acknowledgement" for its structured approach to documenting and analyzing party inputs. Status logs and chat messaging provided "good readable messages" that effectively communicated system status.

Feature Comprehension and Learning Requirements

While basic navigation proved intuitive, the surveyor required "extensive explaining of how the report creation works and IIVM module functions". This indicates that sophisticated verification features need additional onboarding or improved in-application guidance. The observations suggest that a brief tutorial or guided workflow would significantly improve initial user experience, particularly for complex system components.

Technical Performance Observations

The surveyor identified technical limitations when "input text gets too long or too many metadata parts are additionally input to the system", indicating potential optimization opportunities. The IIVM correction process was "sometimes taking long", affecting workflow continuity during report creation.

Regarding the research process using the RIM module, the surveyor noted that "results quality decreases" as queries become more complex or when more potential results are included in searches.

Despite these issues, the surveyor emphasized system stability during typical usage scenarios with no data loss occurring, attributing this to the application's automatic saving and recovery features. The feedback highlights the importance of optimizing verification algorithms and resource management, particularly when working with complex or large-scale report chapters.

7 Discussion of the Outcomes

This chapter discusses the research outcomes. Section 7.1 reflects on the chosen ADR method and reassesses how the research questions from Chapter 3 were fulfilled. Section 7.2 examines the framework, highlighting its strengths and limitations along with its interconnection to the paradigm application. Section 7.3 discusses the toolkit's strengths and limitations, while Section 7.4 provides similar analysis for the HAIMEDA application and identifies potential improvement directions. Section 7.5 concludes with ethical and societal implications of this work.

7.1 Methodological Reflection

7.1.1 Reflection on Action Design Research as Research Method

The selection of ADR as the primary methodological approach proved exceptionally well-suited for this work, as it explicitly recognizes the reciprocal relationship between theory development and practical implementation. By conceptualizing the framework, toolkit, and paradigm as distinct yet mutually informing artifacts, this methodology enabled bidirectional knowledge transfer that substantially enhanced both theoretical and practical outcomes. The framework's theoretical constructs, initially derived from literature and preliminary analysis, underwent continuous refinement through insights gained during paradigm implementation, transforming abstract principles into concrete, actionable guidance. Simultaneously, the paradigm application benefited from the evolving theoretical foundation, which provided structural coherence and methodological direction throughout development iterations.

This methodological synergy represents a significant departure from linear approaches where theory precedes implementation with minimal feedback between stages. Instead, the ADR process facilitated continuous validation and refinement of theoretical constructs through practical application while ensuring implementation decisions remained anchored in sound theoretical principles. The toolkit emerged organically from this process, bridging theoretical concepts with practical implementation requirements and providing tangible manifestations of the framework's

abstract guidance. The following sections systematically reassess the research questions that guided this mutually informing implementation process, demonstrating how each question was addressed through the development of these interconnected artifacts.

7.1.2 Reassessment of the Research Questions

This section systematically reassesses the research questions from Chapter 3 to evaluate how effectively this work addressed them. The first and central research question, which shaped the direction of this study, is:

Primary Research Question 1:

RQ1 *How can a comprehensive development framework be designed to effectively guide the creation of hybrid AI systems?*

The five-phase framework detailed in Chapter 4 comprehensively addressed RQ 1. This framework successfully balances high-level guidance with practical implementation support, emphasizing modularization and agile development principles particularly suited to hybrid AI systems. However, limitations remain in providing tactical implementation guidance for specific hybrid AI mechanisms, which are discussed in Section 7.2.2. While the HAIMEDA paradigm implementation and expert feedback validated the framework's effectiveness, domain-specific biases may limit transferability to substantially different contexts. The following secondary questions explored specific aspects of framework development.

Secondary Questions regarding RQ 1:

RQ1.1 *What specific guidelines, tools, and best practices are essential for developing hybrid AI systems in data-sensitive domains?*

RQ1.2 *How can the framework balance flexibility across different application domains with sufficient structure to guide implementation?*

RQ1.3 *What is the relationship between theory development and practical implementation in the evolution of a hybrid AI framework?*

RQ1.4 *To what extent can a framework developed alongside a specific implementation be generalized to diverse hybrid AI applications?*

The toolkit development in Section 4.4 responds to RQ 1.1 through 26 specialized tools across six toolsets, demonstrating practical implementation of theoretical guidance with best practices encoded throughout the framework's elements. The framework's architectural design tackles RQ 1.2 by separating core principles applicable to all hybrid AI development from optional elements addressing data sensitivity, with its effectiveness and limitations, which are discussed in Section 7.2.2. The theory-practice relationship explored in RQ 1.3 was demonstrated through the reciprocal development process described in Section 7.1 above, while framework generalizability from RQ 1.4 receives detailed analysis in Section 7.2.2.

The second primary research question focused on implementing the HAIMEDA application as a practical manifestation of hybrid AI techniques in medical device assessment:

Primary Research Question 2:

RQ 2 *How can a hybrid AI system be implemented to support medical device damage assessment while ensuring data integrity and confidentiality?*

Chapter 5 systematically addressed RQ 2 through the development and documentation of the HAIMEDA application. This question required concrete implementation strategies and architectural decisions demonstrating effective integration of hybrid AI techniques within highly regulated domains requiring stringent data controls.

Secondary Questions regarding RQ 2:

RQ 2.1 *Which architectural design strategies enable effective integration of different AI techniques to maximize their complementary strengths and reduce their individual limitations?*

RQ 2.2 *What measures must be taken during the development and deployment of a hybrid AI system to ensure data confidentiality and maintain information integrity?*

RQ 2.3 *To what extent can a hybrid AI system for medical device assessment achieve accuracy levels comparable to human experts?*

Architectural design strategies in Section 5.2 and system functionalities in Section 5.1 addressed RQ 2.1, particularly through the `MainController`'s orchestration of specialized modules and the IIVM's hybrid verification approach combining symbolic and neural techniques. Section 4.3 details the strict local-first approach that fulfills RQ 2.2 through pre-processing validation, post-processing verification, and localized deployment constraints preventing external data transmission. Quantitative and qualitative assessments in Section 6.2 answer RQ 2.3 by comparing system performance to human expert standards.

7.2 Critical Assessment of the Framework

7.2.1 Framework Strengths and Contributions

The framework's high-level design intentionally avoids prescribing specific implementation techniques, presenting the paradigm application as a separate, complementary component rather than an integral part. This approach prevents practitioners from developing unintended biases toward particular programming techniques or implementation standards. Such neutrality proves crucial for hybrid AI development, where innovation through experimentation and prototyping drives the creation of effective hybrid AI mechanisms. By emphasizing foundational principles

rather than specific implementations, the framework encourages original research instead of merely replicating existing solutions. Furthermore, it guides practitioners to systematically evaluate which AI approach best suits their specific tasks, whether symbolic, sub-symbolic, or hybrid, thereby reducing the risk of defaulting to trendy approaches without proper analysis.

Unlike frameworks focusing primarily on implementation phases, this framework provides comprehensive lifecycle coverage from initial assessment through deployment to continuous improvement. This holistic approach enables systems to evolve sustainably over extended periods, with individual modules being updated or replaced while maintaining overall system integrity. Modular design principles ensure components can be independently modified without affecting the functionality of the entire system.

Integration of agile development methodologies with structured feedback loops and iteration points makes the framework compatible with established software engineering practices. This compatibility facilitates integration with other theoretical frameworks that might better suit specific domain aspects. Through continuous stakeholder engagement, the framework ensures systems remain aligned with actual user needs rather than being driven solely by technical possibilities. The framework remains applicable even in non-business contexts by allowing practitioners to position themselves as primary stakeholders when third-party stakeholders are absent.

A distinctive feature emerges through the integration of privacy-by-design principles, where data sensitivity considerations are systematically embedded throughout the development cycle rather than retroactively added as compliance requirements. This approach raises awareness among practitioners who might not initially target data-sensitive domains but nonetheless benefit from incorporating privacy considerations. Developers creating multi-user systems in industries where privacy constitutes a fundamental design principle can leverage these built-in practices, even when not specifically developing for traditionally data-sensitive sectors.

7.2.2 Framework Limitations

7.2.2.1 Implementation Guidance Limitations

While the framework provides valuable high-level guidance on hybrid AI development processes, it deliberately avoids prescriptive implementation details. This abstraction encourages innovation but creates a significant gap between conceptual understanding and practical execution. The framework primarily addresses strategic questions, such as which pitfalls to avoid or which indicators suggest particular development strategies, rather than tactical implementation challenges.

The paradigm application partially bridges this gap by offering concrete implementation examples, yet these represent only a narrow subset of

possible hybrid AI architectures. The crucial question of constructing effective hybrid AI mechanisms remains largely unaddressed, as such solutions tend to be highly context-dependent and resist generalization. The HAIMEDA development experience demonstrated this clearly: while initial conceptual ideas formed during system design, the most effective hybrid AI mechanisms emerged organically during mid-development experimentation rather than from predetermined patterns.

This limitation presents a double-edged challenge for practitioners. The framework appropriately encourages prototyping and experimentation as essential for discovery, yet the absence of specific guidance may extend development timelines as teams navigate the vast solution space. Additionally, practitioners might disproportionately adopt familiar or trendy AI techniques rather than exploring the full spectrum of hybrid possibilities, especially given the rapidly evolving research landscape where many hybrid approaches remain underexplored.

7.2.2.2 Applicability and Domain Constraints

Despite efforts to create a domain-agnostic framework, the HAIMEDA development context has left a distinct imprint on the framework's structure and recommendations. The medical device surveyor application, developed with business-oriented methodologies for gathering requirements and workflows, has influenced fundamental aspects of the framework's design. This raises questions about transferability to substantially different contexts. Had the framework emerged alongside a paradigm application in manufacturing, pure healthcare delivery, or scientific research rather than a professionally-oriented surveyor tool, its core principles, phases, and recommended activities might have emphasized different aspects or structured workflows differently.

The framework's origins in data-sensitive domains appear throughout, with privacy considerations deeply embedded in the development lifecycle. While this integration represents a forward-thinking approach to system design, it creates potential barriers to adoption in domains where data sensitivity is less critical. Practitioners working in areas with minimal privacy concerns may perceive these embedded safeguards as unnecessary overhead rather than valuable guidance. Mandatory privacy assessment steps in the initial framework phase may deter adoption by teams focused on domains where other considerations, such as computational efficiency or algorithmic transparency, deserve greater emphasis.

Furthermore, the framework presupposes certain resource allocations and team structures that may not reflect reality across diverse development environments. Organizations with different resource constraints, team compositions, or institutional priorities might struggle to implement the framework's recommendations without significant adaptation, potentially undermining its effectiveness as a standardized approach to hybrid AI development.

7.2.2.3 Methodological and Technical Biases

The framework's development methodology bears the unmistakable signature of contemporary business software practices, reflecting its co-evolution with the HAIMEDA paradigm application. These practices, including iterative development cycles, sprint-based deliverables, and agile methodologies, have become embedded within the framework's recommended processes. While effective in commercial software development, these approaches represent just one philosophical tradition among many. Alternative methodologies such as formal verification-driven development, scientific computing approaches, or research-oriented frameworks might have yielded fundamentally different recommendations for hybrid AI system construction.

Technical implementation choices in the paradigm application have shaped the framework in subtle but significant ways. Though the framework documentation maintains technology neutrality, extensive use of Elixir in the paradigm application introduces an implicit bias toward functional programming concepts. Principles such as modularity, statelessness, and immutability, cornerstones of the Elixir ecosystem, manifest in the framework's architectural recommendations. This unconscious influence appears particularly in how the framework approaches system decomposition and component isolation.

These biases do not necessarily diminish the framework's value but do constrain its perspective. Practitioners from traditions outside business-oriented software engineering or those working primarily in object-oriented, procedural, or logic programming paradigms may need to translate the framework's guidance into terms aligning with their technical and methodological contexts. Thus, the framework might have evolved differently had the paradigm application been implemented using C++ with object-oriented design patterns, Python with its flexible multi-paradigm approach, or Prolog with its declarative logic programming model.

7.2.3 Framework-Paradigm Interdependency

Beyond the one-way influence already discussed, the framework and paradigm exist in a unique circular dependency creating distinct adoption challenges. While most frameworks include examples as optional supplements, here the paradigm serves as an essential interpretive lens without which abstract principles remain difficult to apply. This creates a higher barrier to entry, as practitioners must invest in understanding both components rather than just the framework documentation.

This interdependency raises unique sustainability concerns for future evolution. As hybrid AI techniques advance, maintaining coherence between abstract principles and concrete implementations will require simultaneous curation of both components. To demonstrate the framework's full adaptability across different technical contexts, future iterations may need multiple reference implementations spanning diverse architectural approaches and programming paradigms.

7.3 Critical Assessment of the Toolkit

7.3.1 Toolkit Strengths and Contributions

The toolkit embodies modularity as a foundational design principle, mirroring the framework's emphasis on component-based architecture. This modular approach extends throughout the implementation, where functions, submodules, and tools are named to explicitly reflect their purpose, creating inherent documentation through nomenclature. The inheritance structure demonstrates architectural thoughtfulness, with newer tools extending core tool functionality while maintaining compatibility. This carefully designed extensibility enables practitioners to easily export and repurpose specific tools, submodules, or individual functions without adopting the entire toolkit, fulfilling the primary intention of creating a flexible ecosystem of interoperable components.

Several high-value components demonstrate exceptional generalizability beyond architectural strengths. The *Model Fine-Tuning Tools* represent a particularly versatile subsystem that can be redeployed across diverse applications with minimal adaptation. Despite focusing on PEFT and LoRA adapters for efficiency reasons, these tools maintain flexibility through their controller-based design pattern. Similarly, the *Model Integration Tools* achieve broad compatibility with various model types and formats, functioning effectively across different fine-tuned variants and base models. The *PII Sanitization Tool* provides substantial value through sophisticated entity detection and replacement strategies, offering a robust solution to a common challenge in AI development for sensitive domains.

The strategic decision to implement Model Fine-Tuning Tools using PyTorch rather than native Elixir solutions like Bumblebee represents a pragmatic compromise that ultimately enhanced toolkit robustness. This approach leveraged PyTorch's mature ecosystem and widespread compatibility, circumventing specific technical limitations encountered with Bumblebee[1] and related libraries like Nx[2] on Windows or Linux systems with NVIDIA GPUs using the CUDA Toolkit[3]. Bumblebee currently supports only a limited set of models and remains under active development, with notable constraints in areas such as quantization support. While this choice introduced a departure from the otherwise Elixir-centric implementation, it demonstrates the framework's flexibility to prioritize functional requirements over programming paradigm purity when necessary. The resulting interoperability between Elixir components and PyTorch-based fine-tuning modules provides an instructive example of effective cross-language integration in complex AI systems.

1: https://hexdocs.pm/bumblebee/Bumblebee.html
2: https://hexdocs.pm/nx/Nx.html
3: https://developer.nvidia.com/cuda-toolkit

Even domain-specific tool implementations serve as valuable reference implementations and practical manifestations of framework principles. These concrete implementations provide practitioners with working examples of data privacy protection, information extraction patterns, and model integration techniques. By revealing common pitfalls and demonstrating effective approaches, even specialized components offer educational value beyond their direct utility. This exemplar function

helps bridge the gap between abstract framework guidelines and practical implementation challenges, providing a crucial middle layer of guidance through demonstrated patterns and solutions.

7.3.2 Toolkit Limitations and Constraints

The toolkit's implementation choices impose technical constraints that may limit adoption across different programming paradigms. Predominant use of Elixir narrows the practitioner base to those familiar with functional programming concepts or willing to invest in learning this paradigm. While functional programming offers advantages for concurrency and immutability, it creates barriers for developers primarily versed in object-oriented or procedural approaches. Additionally, structural issues exist within the codebase itself, particularly in the *Dataset Building* toolset where significant code duplication could be refactored into a common base class, representing an unrealized opportunity for improved internal modularity.

Domain-specificity presents another significant limitation. Several core components, particularly the *Data Preparation* and the Dataset Building toolset, and aspects of the *Information Extraction* tools, are tightly coupled to HAIMEDA's specific requirements. These tools operate on predetermined data structures and workflows reflecting the paradigm application rather than generalized solutions. While they demonstrate important techniques, practitioners would need substantial modifications to accommodate different data layouts or extraction requirements, reducing direct transferability across domains.

Technical dependencies further constrain the toolkit's versatility. Model Integration tools rely specifically on the Llama.cpp library for critical operations like GGUF format conversion, limiting compatibility with alternative model processing libraries. Integration capabilities remain restricted to the Ollama runtime environment with a focus on local execution, lacking support for distributed computing environments or cloud-based deployment strategies. The *Model Fine-Tuning Tools* subsystem, though flexible within its domain, emphasizes particular optimization approaches, specifically PEFT and LoRA, without comparable support for alternative fine-tuning methodologies.

Several components remain in an evolving state with incomplete features affecting their robustness. The PII Sanitization tool, while effective for typical use cases, lacks optimization for processing large volumes and would benefit from more sophisticated fake data generation capabilities. A notable gap across multiple components is the current absence of support for distributed computing, particularly integration of remote processing GPUs using frameworks like Slurm[4]. Similarly, the *Model Integration Tools* currently lack comprehensive format conversion options and remote cloud system integration, which limits deployment flexibility in enterprise environments. While support for distributed computing is actively being developed for the HAIMEDA application, it is expected to be incorporated into the toolkit modules in the near future. Addressing these gaps through future development will substantially enhance

4: https://slurm.schedmd.com/documentation.html

the toolkit's applicability across diverse computing environments and operational scales.

7.4 Critical Assessment of the Paradigm

7.4.1 Implementation Achievements and Strengths

The HAIMEDA application serves as a compelling proof of concept for the framework, demonstrating its practical viability while simultaneously enriching its development. This bidirectional relationship between theory and implementation created a reinforcing cycle where practical insights directly shaped framework refinement. The implementation successfully addressed all primary objectives identified in cooperation with the medical device surveyor, with only the time value calculation of medical devices remaining unimplemented at this stage. This high completion rate validates the framework's effectiveness in guiding complex hybrid AI development from requirements to deployment.

Beyond requirements fulfillment, HAIMEDA demonstrates sophisticated integration of diverse AI techniques across multiple architectural layers. At the application level, the `MainController` orchestrates interactions between symbolic verification systems, LLM-based content generation, and hybrid research capabilities. Within individual modules, more granular hybrid approaches emerged, particularly in the IIVM, where symbolic pattern recognition works alongside neural embedding-based similarity detection to ensure content accuracy. This multi-level integration of AI techniques demonstrates the framework's flexibility in supporting various hybridization strategies.

The implementation validated several key framework principles, particularly regarding modularization and iterative development. Clean separation between AI components allowed for independent evolution of each module while maintaining system cohesion. The progressive fine-tuning strategy, evolving from basic to specialized models through multiple iterations, demonstrates how the framework's emphasis on experimental prototyping directly contributes to implementation success. The application's ability to adapt to changing requirements throughout development, such as pivoting from SMT solvers to custom verification systems when limitations emerged, highlights how the framework effectively supports agile response to technical challenges in hybrid AI development.

7.4.2 Current Paradigm Limitations

Despite successfully implementing the primary objectives, the HAIMEDA paradigm exhibits several performance limitations requiring further refinement. The fine-tuned Llama 3 8B models currently produce inference output that falls short of optimal quality for production use, particularly for complex damage assessment scenarios. This limitation stems primarily from hardware resource constraints restricting both model

size and training capabilities, but is likely exacerbated by methodological decisions in the fine-tuning process.

The sequential merging of LoRA adapters with the base model between training stages potentially introduced knowledge degradation and parameter inefficiencies that could have been avoided with a checkpoint-based approach. By merging adapters before subsequent fine-tuning, each new training stage likely disrupted previously learned patterns, leading to suboptimal knowledge integration and potential catastrophic forgetting across the adaptation sequence. This approach also increased computational overhead by repeatedly fine-tuning complete merged models rather than maintaining the parameter-efficient LoRA advantage throughout the process. Current token length constraints during training further compromise model effectiveness, limiting the system's ability to process and learn from longer, more comprehensive examples.

The RIM represents another area with notable limitations. Its specialized V1-B model exhibits limited generalization when handling unfamiliar context structures, a constraint stemming from training dataset homogeneity. The symbolic user question processing component similarly demonstrates restricted semantic understanding capabilities, particularly when processing novel query formats or accessing complex database structures. Both issues impact the system's ability to respond appropriately to diverse user inquiries outside its directly trained domains.

Several functional limitations also exist in the current implementation. The secondary objective of calculating device time values remains unimplemented, while the system lacks export capabilities for written reports, requiring manual transfer of content to external document formats. These workflow constraints, though not affecting core assessment capabilities, impact practical utility in production environments. A more effective strategy would have involved maintaining a consistent base model while fine-tuning from checkpoints of previous stages, allowing better preservation of learned capabilities from each stage while enabling more controlled and interpretable adaptation of the model's behavior across the progressive training sequence.

7.4.3 Planned Enhancements and Future Development

To address model performance limitations, a comprehensive enhancement strategy has been devised. The initial phase involves improving the PII Sanitization tool to enable more effective dataset preparation with sophisticated fake data generation capabilities. This will be followed by implementing remote GPU techniques for training substantially larger models, specifically transitioning to a *Llama* 3.3 70*B*[5] model architecture. Since specialized German-language versions of this model don't currently exist, initial unsupervised pretraining on large German corpora such as OSCAR would significantly enhance domain-specific performance [CSM20, pp. 6789ff]. For production deployment, the model will be quantized to 4 or 5 bits with hybrid CPU-GPU loading techniques, maintaining performance-critical layers in GPU memory while offloading others to system RAM.

5: https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_3/

The integration of remote inference capabilities through Ollama on HPC infrastructure at FAU Erlangen[6] demonstrates promising feasibility for distributed model deployment. A prototype utilizing remote GPU resources via Slurm job scheduling is in the final stages of implementation, with initial tests validating the technical approach. This architecture will enable researchers to deploy custom GGUF models within the HAIMEDA framework, leveraging institutional computational resources while maintaining deployment flexibility. The remote inference infrastructure is nearing completion and will be available for external validation upon finalization.

6: https://hpc.fau.de/

RIM enhancements will focus on creating more diverse QA datasets incorporating varied content structures and topics beyond medical devices. Implementing a larger model for the RAG system will increase available context length, enabling more reliable processing of multiple reports simultaneously. The symbolic AI component will be enhanced for improved semantic understanding, database interaction, and data extraction capabilities. Additional functionality includes internet connectivity for online searches about unfamiliar devices and awareness of the current report being written to extract device information before searching, enabling more targeted results for similar cases.

Several new capabilities are planned to address current limitations and extend system functionality. Following the framework's guidance, a comprehensive monitoring and logging system will evaluate application effectiveness and user interaction patterns, providing valuable insights for continuous improvement while addressing the framework's emphasis on ongoing evaluation and quality assurance. Document workflow enhancements will streamline the report generation process through a templating-based system that converts created reports into DOCX documents, maintaining the surveyor's established formatting and structure. This functionality will include MD export of report chapters, eliminating the current manual transfer process and facilitating the transition from AI-assisted drafting to final documentation.

The most significant expansion involves extending HAIMEDA into multimodal analysis through image processing integration. This capability will enable surveyors to include device photographs directly within the application, while specialized multimodal LLMs analyze these images to suggest damage causes and categorization based on visual evidence. This enhancement requires fine-tuning on surveyor-provided images paired with professional analyses to develop domain-specific visual assessment capabilities complementing existing text-based assessment functions.

System intelligence will be optimized through dynamic LoRA adapter activation architecture. Rather than employing separate LLMs and prompt engineering for different tasks like text summarization and revision, this strategy enables a single base model to serve diverse functions through adapter switching, offering considerable efficiency advantages while maintaining specialized capabilities for each task type. Finally, a continuous learning architecture will enable the system to acknowledge and incorporate reports created through HAIMEDA. This feedback loop will

automatically insert newly written report chapters into the RAG system database and facilitate periodic fine-tuning through automated data preparation pipelines. The resulting self-improving system will evolve with continued use, progressively enhancing performance based on real-world application.

These enhancements collectively represent both targeted solutions to current limitations and strategic extensions of system capabilities, following the framework's emphasis on continuous improvement and adaptation to evolving requirements.

7.5 Ethical and Societal Implications

This research demonstrates that hybrid AI architectures represent not merely a technical preference but an ethical imperative for data-sensitive domains where purely statistical approaches prove fundamentally inadequate. Beyond the specific medical context, this work addresses broader ethical concerns inherent in all AI applications: the risk of automation bias where professionals might uncritically accept AI-generated content, the challenge of maintaining meaningful informed consent when processes become increasingly complex, and the subtle power shifts that occur when technological systems mediate professional judgment.

Even hybrid approaches must navigate persistent challenges of potential algorithmic bias, transparency for end-users, and preventing gradual professional deskilling through overreliance. The structural choices in the framework address these ethical dimensions directly by preserving professional autonomy while enhancing productivity. Verification pipelines and explicit boundaries between AI-assisted and human decision processes ensure that technology augments rather than replaces professional judgment.

As AI capabilities continue advancing, this hybrid framework provides an ethical foundation for systems demanding both innovation and unwavering information integrity, particularly in contexts where errors carry significant human consequences. The framework's emphasis on maintaining human control over critical decisions while leveraging AI for efficiency gains offers a balanced approach to the ethical deployment of advanced AI systems in sensitive domains.

8 Conclusion

8.1 Summary of Contributions

This research makes three interconnected contributions to the field of hybrid AI development. First, it introduces a comprehensive framework for developing hybrid AI systems specifically designed for data-sensitive domains. This framework provides a structured development approach with clearly defined phases, principles, and guidelines that address the unique challenges of combining symbolic and sub-symbolic AI techniques while maintaining data confidentiality and information integrity.

Second, the research delivers a practical toolkit that operationalizes the framework's theoretical guidance through concrete implementation resources. This toolkit offers direct utilities for critical development tasks, including PII sanitization, model fine-tuning, and information extraction, alongside reference implementations demonstrating effective patterns for hybrid AI integration. These resources bridge the gap between abstract framework principles and technical implementation, enabling practitioners to adapt and extend the approaches for their specific contexts.

Third, the HAIMEDA application for medical device damage assessment serves as a paradigm implementation validating the framework's effectiveness in a real-world context. This application demonstrates sophisticated hybrid AI integration through specialized modules combining symbolic verification systems, LLM-based content generation, and hybrid research capabilities. Through its successful implementation, HAIMEDA proves the practical viability of the framework while illustrating how different AI techniques can be effectively combined to address complex domain requirements.

Methodologically, this work exemplifies the value of ADR for developing interconnected theoretical and practical artifacts. The bidirectional knowledge transfer between framework development and practical implementation created a reinforcing cycle that significantly enhanced both components, resulting in contributions that are theoretically sound, practically relevant, and adaptable across diverse application contexts.

Beyond technical contributions, this research establishes hybrid AI as an ethical imperative for domains where errors carry significant consequences. The framework's structural choices directly address critical ethical concerns including automation bias, meaningful informed consent,

and professional autonomy while providing a foundation for responsible AI development.

8.2 Critical Reflection of the Outcomes

The framework's conceptual orientation enables broad applicability across various contexts, though this abstraction naturally creates distance from concrete implementation. While promoting flexibility, practitioners may seek more detailed guidance or code-level examples for specific hybrid AI integration techniques.

The toolkit reflects current priorities with its emphasis on sub-symbolic AI components, particularly LLM fine-tuning, integration, and deployment. Symbolic AI tools remain comparatively less developed, creating opportunities to enhance future versions by broadening support for rule-based systems and symbolic reasoning techniques essential for certain hybrid AI use cases.

HAIMEDA illustrates key hybrid AI patterns and validates the framework's practical relevance through its focus on sequential and verification-based hybridization strategies. While effective for the intended context, this approach leaves room for exploring alternative architectures such as symbolic-guided decoding, neural-symbolic reasoning, or differentiable logic programming. Expanding into these areas could enhance the framework's generalizability and offer deeper insights into hybrid AI integration's full potential.

These reflections point to promising directions for future work rather than fundamental shortcomings, highlighting natural areas for continued development within the evolving landscape of hybrid AI.

8.3 Related and Emerging Topics

Contemporary developments in hybrid AI complement and extend this research in several important directions. Neuro-symbolic AI initiatives like IBM's Neuro-Symbolic Concept Learner [Mao+19] demonstrate alternative hybridization patterns that more deeply integrate neural and symbolic components through differentiable programming. These approaches offer potential enhancements to the verification-focused patterns implemented in HAIMEDA.

Multi-modal hybrid systems have gained prominence in regulated domains, particularly where text, image, and structured data require joint processing [Aco+22]. Medical systems combining radiological imaging with clinical documentation exemplify how hybrid architectures address complex diagnostic challenges, aligning with planned HAIMEDA extensions while offering broader insights into multi-modal reasoning capabilities.

The regulatory landscape continues evolving rapidly, with frameworks like the EU AI Act [Eur24] and FDA guidelines for AI/ML-based medical devices [US 21] establishing new compliance requirements. These standards increasingly recognize the unique verification challenges of hybrid systems, where both explainability and performance standards must be satisfied simultaneously.

The expanding ecosystem of local-first AI deployment tools, including TensorFlow Lite [Ten23], ONNX Runtime [ONN24], and llama.cpp [Ger23], enables increasingly sophisticated on-device inference. These technologies complement the framework's emphasis on data-sensitive deployment while extending capabilities through hardware-specific optimizations that could enhance future hybrid AI applications in privacy-demanding contexts.

8.4 Impact of the Results and Future Work

This framework represents a significant advance in conceptualizing practical knowledge for hybrid AI system development. While originally directed toward data-sensitive domains, its principles extend to any field where protection of PII or confidentiality remains paramount, including legal services, cybersecurity, and financial services.

Future research opportunities include developing more specialized frameworks from this foundation. Promising directions involve deeper exploration of hybrid AI techniques across domains, investigating how different hybridization patterns perform in various environments. Research on sophisticated integration patterns between symbolic and sub-symbolic components could expand the framework's application scope beyond current implementations.

For HAIMEDA specifically, several enhancements merit implementation. An additional hybrid component for the RAM module could use symbolic AI to determine inquiry purposes and dynamically activate specialized LoRA adapters. Other improvements include fine-tuning on larger language models and implementing multimodal analysis capabilities. These innovations illustrate how practical development catalyzes innovation, generating both technical improvements and unexpected research opportunities.

Beyond technical contributions, HAIMEDA demonstrates the democratization of advanced AI capabilities. It showcases how individuals can create sophisticated applications for complex professional use cases using consumer hardware and limited resources. This fundamental shift from earlier paradigms, where such technologies required corporate-scale investments, is enabled by open-source libraries, APIs, and conceptual frameworks like the one presented here. Domain experts can now develop tailored AI solutions for specialized needs independently.

As hybrid AI systems continue evolving, the principles, patterns, and practices established in this work provide a foundation for responsible and effective development across increasingly diverse domains. By bridging theoretical guidance with practical implementation through a mutually informing development process, this research contributes to both academic understanding and practical advancement of hybrid AI in an era where such capabilities become increasingly integral to specialized professional practice.

A Domain Analysis and Iterative Development Cycles

This appendix provides comprehensive details on two key aspects of this research. Section A.1 examines the medical device assessment domain, including background information on the surveyor's role, typical workflows, and data sensitivity considerations. Section A.2 presents the iterative development process through a summary table of the BIE cycles that guided the development of the framework, toolkit, and paradigm.

A.1 Overview of the Medical Device Assessment Domain

The proposed application for medical device assessment is designed to support medical device expert Walter Sigloch in his daily work, particularly in writing and documenting reports delivered to contracting entities such as authorities, insurances, or individuals. The following sections detail the specific nature of this work based on multiple personal interviews with Walter Sigloch [Sig].

A.1.1 The Role of the Medical Device Surveyor

The medical device surveyor, represented in this research by Walter Sigloch, serves as an independent expert who conducts comprehensive assessments of medical devices in cases of damage, malfunction, or disputed functionality. The surveyor produces legally relevant reports that may be used in insurance claims, legal proceedings, or regulatory decisions.

A.1.2 Survey Types

Mr. Sigloch primarily distinguishes between two types of surveys: legal and insurance-related. Legal surveys can span multiple months and involve extensive communication between various stakeholders, demanding highly individual workflows and adaptive techniques. However, these appear relatively rarely. In contrast, insurance-related surveys constitute the majority of his daily work.

A.1.3 Relevant Stakeholders

The survey creation process involves several parties. *Contracting parties* typically include legal and police authorities for legal surveys, while insurance-related surveys involve insurances, healthcare facilities, doctors, patients, and other private clients. *Defending parties*, particularly present in insurance-related surveys, include insurances or other device-owning or using parties under investigation. *Information-giving parties* provide crucial details about damage causes or device specifics, including repair costs, part prices, and operational methods. These informants range from witnesses such as nurses, doctors, and patients who can describe damage circumstances, to device manufacturers and service providers who offer technical insights. Finally, the *expert party* includes the conducting surveyor, who may collaborate with other specialists such as medical device engineers, doctors, or legal experts to gather comprehensive information.

A.1.4 Research Work

Research constitutes approximately 30 % of the overall survey work. This involves questioning device manufacturers, studying device operation modes through technical and user manuals, and interviewing all involved parties. The surveyor also reviews previously conducted surveys on similar device types, comparable cases, or even the same device or stakeholders from earlier investigations.

A.1.5 Data Sensitivity

Survey reports invariably contain sensitive information ranging from device serial numbers and stakeholder contact details to patient medical histories and potential causes of device malfunctions that may have led to patient harm or death. Consequently, ensuring careful information handling and compliance with applicable data protection regulations is paramount. These regulations include the GDPR [Eur16] in the European Union, as well as medical device-specific regulations such as the Medical Device Regulation [Eur17] and its German national implementation through the Medizinprodukterecht-Durchführungsgesetz [Deu21]. Compliance measures include anonymizing sensitive data where possible and ensuring all parties understand their data usage rights.

A.1.6 Challenges and Pivotal Aspects

Creating effective survey reports requires coherence, logical implications, professional expression, and meaningful structure. The surveyor must present all information clearly and logically, avoiding contradictions or ambiguities while maintaining appropriate restraint to prevent legal or defamatory consequences. Information integrity therefore plays a vital role throughout the structuring, execution, and review of surveys.

A.1.7 Workflow of a Typical Survey Creation

The survey process begins when the surveyor receives the contract along with all available information from the contracting party. Research commences immediately, with the surveyor either receiving the damaged medical device with the initial contracting information or visiting its location. Simultaneously, the surveyor contacts involved parties to gather facts that may help exclude certain damage scenarios.

During device investigation, the surveyor specifically examines both acute damage and attrition damage, as insurances typically cover only acute damages. This examination includes photographing damaged device parts and damage-causing threats such as broken water pipes. Before beginning the written work, the surveyor contacts information providers, particularly device manufacturers, to obtain current price lists for individual parts. While listing and categorizing damaged parts, the surveyor calculates separate sums for acute and attrition damages.

The cause of loss requires particular attention, as violations of obligations by device owners or responsible persons can result in total loss of insurance coverage for acute damages. The surveyor completes the process by writing the final report incorporating all gathered information, personal notes, and photographs of damaged parts. Although writing theoretically occurs at the end, new information or details often emerge during the report composition phase, requiring adaptive documentation approaches.

A.2 Building, Intervention, and Evaluation Cycles

Table A.1: Summary of BIE cycles across framework, toolkit, and paradigm development.

Timeline	Framework Evolution	Toolkit Development	Paradigm Implementation
Nov 2024	Literature review and domain adaptation principles	Initial research on tools and frameworks for hybrid systems	Requirements gathering and surveyor workflow analysis
Dec 2024	Core principles definition; Domain theory integration	Research on data handling and privacy-preserving methods	Technical feasibility assessment; Access database exploration
Jan 2025	Development lifecycle phases formalized	File Extraction and Data Preparation tools prototyped	HATR prototype: Testing hybrid orchestration patterns
Feb 2025	AI component selection guidelines	File Conversion and Data Partitioning tools implemented	Data collection and preprocessing for fine-tuning Phase 1
Mar 2025	Implementation patterns refined based on HATR findings	Information Extraction tools; Symbolic Extractor developed	Fine-Tuning Phase 1: Initial approach with single model; Testing LLM capabilities
Apr 2025	Data-sensitive domain guidelines; Functional programming patterns	Model Fine-Tuning tools; Dataset Building tools with supervised instruction support	Fine-Tuning Phase 2 (Stages A–E); `MainController`, IIVM-Preprocessor and RAM implementation
May 2025	Hybrid AI integration patterns; Verification system designs	Model Integration tools with GGUF conversion; Enhanced Dataset Building for specialized models	Fine-Tuning Phase 3: Model variants V1/V2; LSM, IIVM-Postprocessor implementation; TipTap Editor development
Jun 2025	Framework consolidation; Documentation of generalization principles	SLERP merging tools; V1-B specialization for Q&A; Toolkit documentation completed	V3 merged model deployment; User testing; RIM implementation with specialized Q&A model; System evaluation

Note: The Framework Toolkit development aligned closely with the fine-tuning phases, with most components created during Phases 1 and 2 (March–April 2025). Phase 3 introduced specialized model variants: V1 (knowledge-focused), V2 (structure-focused), V3 (merged model), and V1-B (Q&A specialized).

B Framework Extension for Practical Application

This appendix presents the framework in a format designed for practitioners, mirroring the structure of the main framework chapter while emphasizing practical features. Section B.1 offers deeper insights into Hybrid AI Strategies than the main text, providing an overview of implementation techniques and example applications. Section B.2 follows the same approach as the framework's development lifecycle phases from Section 4.3, extending established concepts with practical checklists of key activities for each framework component across all development phases.

B.1 Hybrid AI Strategies

Strategy 1: First Symbolic, Then Sub-Symbolic (Pre-Processing Integration)

- **Primary Use Cases:**
  - Enhancing generation quality through structured inputs
  - Optimizing performance by filtering irrelevant data
  - Ensuring security by sanitizing sensitive information
  - Standardizing inputs for consistent processing

- **Implementation Techniques:**
  - Rule-based input preparation and structuring
  - Constraint-based query formulation
  - Symbolic reasoning to identify relevant context
  - Pre-filtering of data based on formal criteria
  - Input validation against domain schemas
  - Knowledge graph traversal for context enrichment
  - Template-based prompt construction with dynamic variables

- **Example Applications:**
  - Clinical decision support with validated patient data inputs
  - Data analysis pipelines with structurally verified inputs
  - Privacy-preserving NLP with PII detection and anonymization
  - Enhanced search systems with query optimization

Strategy 2: First Sub-Symbolic, Then Symbolic (Post-Processing Integration)

- **Primary Use Cases:**
  - Content generation with verification requirements
  - Extracting structured data from unstructured sources
  - Creative solutions requiring validation against constraints
  - Natural language outputs that must adhere to specific formats

- **Implementation Techniques:**
  - Sub-symbolic generation followed by symbolic verification
  - Pattern extraction via LLMs with rule-based validation
  - Fact-checking through knowledge base comparison
  - Format enforcement through grammar-based parsing
  - Entity recognition followed by logical consistency checking
  - Numerical value extraction with mathematical validation

- **Example Applications:**
  - Medical report generation with protocol compliance verification
  - Legal document drafting with regulatory constraint checking
  - Financial analysis with mathematical consistency validation
  - Technical documentation with terminology standardization

Strategy 3: Symbolic and Sub-Symbolic in Parallel (Integrated Processing)

- **Primary Use Cases:**
  - Complex decision processes requiring multiple reasoning modes
  - Interactive systems with dynamic adjustment needs
  - Workflows combining creativity and formal constraints
  - Problems requiring continuous verification during solution development

- **Implementation Techniques:**
  - Constrained decoding with symbolic guidance during generation
  - Tool-augmented generation with function calling
  - Neuro-symbolic reasoning networks
  - Iteration loops with symbolic evaluation and regeneration
  - Hybrid planning systems with symbolic goals and neural execution
  - Dynamic prompt construction based on symbolic state
  - Multi-agent systems with specialized symbolic and neural agents

- **Example Applications:**
  - Autonomous planning systems with safety constraints
  - Scientific discovery combining hypothesis generation and verification
  - Interactive programming assistants with code correctness checking
  - Complex diagnostic systems combining heuristics and pattern recognition

B.2 Framework Development Lifecycle: Key Activities Checklists

Phase 1: Assessment & Planning

E 1.1 Domain & Requirements Analysis

E 1.1 Key Activities:

- Conduct stakeholder interviews to identify core requirements and data privacy constraints
- Analyze existing systems and processes that will integrate with the new AI solution
- Document domain-specific workflows, identify possible AI integration points
- Identify regulatory compliance requirements applicable to the domain
- Classify data sensitivity levels and establish initial handling protocols
- Create user personas and scenarios to understand basic interaction needs
- Catalog domain expertise and rules that might inform later AI approaches

- Document initial performance expectations and stakeholder success criteria
- Define primary objectives (critical, must-have requirements) and secondary objectives (supplementary, nice-to-have features), and prioritize them
- Document stakeholder expectations regarding AI capabilities and limitations

E 1.2 Data Analysis

E 1.2 Key Activities:

- Perform statistical analysis of data volume, quality, distribution, and structure
- Identify data patterns, outliers, and anomalies that may influence AI technique selection
- Document data sensitivity levels and associated processing constraints
- Identify data extraction and transformation requirements
- Determine appropriate AI processing techniques for certain forms of data
- Evaluate whether data volume and quality are sufficient for fine-tuning, or if symbolic rule-based techniques or RAG approaches might be more suited
- Map domain-specific rules and constraints that could be encoded in symbolic components
- Determine scenarios where AI can produce information as part of the primary requirements
- Determine which parts of the system require high information integrity and verification
- Identify verification points where symbolic AI / human review should validate sub-symbolic AI outputs

Toolkit Support: The Statistics submodule (Appendix C.1.1.2) provides functions for quantitative data analysis, enabling pattern identification and occurrence frequency calculations. The Symbolic Extractor and Sub-Symbolic Extractor's Prompt Creation component (Appendix C.1.2) demonstrate different information extraction and enrichment methods, helping identify suitable AI techniques and data aspects requiring special attention during implementation.

E 1.3 Infrastructure & Resource Planning

E 1.3 Key Activities:

- Document available hardware resources, focusing on CPU, RAM, GPU specifications, storage capacity, and bandwidth and latency of internet connection
- Evaluate local deployment options for different model sizes with various quantization levels
- Assess network connectivity constraints that might impact remote AI service usage

- Identify target operating systems and compatibility requirements for deployment environment
- Document libraries, APIs, and frameworks compatible with target environment
- Develop data sanitization protocols for scenarios requiring remote processing of sensitive data
- Map stakeholder data sensitivity requirements to appropriate processing environments (local vs. remote)
- Establish fallback processing strategies when primary approaches prove insufficient
- Determine specialized hardware needs for specific AI techniques (e.g., requirements for fine-tuning)
- Estimate inference latency requirements and match with appropriate model configurations

E 1.4 System Outline

E 1.4 Key Activities:

- Map stakeholder requirements to specific system modules
- Create a high-level system diagram depicting core modules and their interactions
- Identify key decision points where symbolic and sub-symbolic approaches will be integrated
- Define the orchestration strategy for coordinating module activities
- Define clear interfaces between modules, focusing on data exchange formats and protocols
- Document data flow pathways between modules, highlighting where sensitive information is processed
- Design standardized and reusable data pipelines for cross-module communication and information exchange
- Develop a data transformation strategy for cross-module communication
- Create technical specifications for each module's responsibilities and constraints
- Document technology stack decisions for each module and the orchestration layer
- Establish error handling and fallback procedures between modules
- Identify potential bottlenecks or performance concerns at the architectural level

Phase 2: Design & Implementation Preparation

E 2.1 Module Selection & Planning

E 2.1 Key Activities:

- Create a module dependency graph to establish logical implementation sequence
- Prioritize modules based on combined primary objectives and technical risk assessment
- Define detailed module responsibilities, expanding on the system outline from Phase 1
- Identify specific integration points with existing system components
- Document specific symbolic and sub-symbolic components within each selected module
- Document integration points where control transfers between symbolic and sub-symbolic components
- Define specific data formats and transformation rules for intra-module communication
- Establish concrete success criteria and evaluation metrics for this iteration
- Document specific third-party libraries, APIs, and frameworks to be used in implementation
- Develop low-fidelity prototypes specifically targeting high-risk technical approaches
- Devise contingency plans for modules with high implementation uncertainty
- Create a detailed development roadmap with milestones for the current iteration

E 2.2 Data Preparation & Enrichment

E 2.2 Key Activities:

- Implement the PII detection and sanitization procedures defined during infrastructure planning
- Configure context window partitioning for fine-tuning datasets (80 % content)
- Develop document parsing workflows specific to the selected module's input requirements
- Create data normalization procedures that standardize formats while preserving semantic meaning

Toolkit Support: The PII Sanitization Tool (Appendix C.1.1.1) detects and anonymizes sensitive information while preserving document structure. The Data Transformation Submodules demonstrate format conversion and processing techniques for building fine-tuning datasets. The Symbolic and Sub-Symbolic Extractors (Appendix C.1.2) provide information extraction and enrichment capabilities.

- Develop data filtering mechanisms to remove noise and irrelevant content
- Create structured data schemas optimized for the selected module's processing needs
- Design hierarchical processing pipelines with general cleaning, domain-specific transformations, and module-specific optimizations
- Document specific transformation rules and preprocessing decisions for reproducibility
- Create data validation procedures that verify extraction and transformation accuracy
- Implement entity and relationship extraction procedures targeting domain-specific concepts
- Generate augmented datasets with contextual metadata to enhance training signal

E 2.3 AI Component Configuration

E 2.3 Key Activities:

- Test language models with task-specific prompts that align with the module's primary objectives
- Evaluate model performance with representative domain data at different parameter settings
- Document optimal configuration parameters for selected models (temperature, optimal context handling length, etc.)
- Test API behavior with sample data to identify limitations or discrepancies from documentation
- Create prototype prompt templates for language model components aligned with module requirements
- Update integration points between symbolic and sub-symbolic components
- Implement sample symbolic rules to evaluate their effectiveness for the module's tasks
- Establish fallback mechanisms for when primary AI approaches underperform
- Document specific limitations discovered during component testing

Phase 3: Implementation & Integration

E 3.1 Dataset Building & Fine-Tuning

E 3.1 Key Activities:

- Convert the prototype prompt templates into standardized instruction templates
- Design instruction templates that specifically address domain-specific formatting requirements
- Document rationale for instruction template design choices for reproducibility
- Incorporate the enriched data and extracted relationships from into training examples
- Include counter-examples that teach the model to recognize and acknowledge task limitations and unsolvable requests
- Create synthetic edge cases to improve model robustness on rare scenarios
- Create a balanced distribution of instruction types across domain knowledge, format compliance, and reasoning categories
- Segment dataset construction into test, validation, and training sets
- Calculate token distribution statistics to ensure compliance with the maximum token length limit
- Set optimal fine-tuning parameters based on model architecture, dataset characteristics, and hardware resource capabilities
- Establish multi-stage fine-tuning workflow if needed, based on instruction types
- Conduct fine-tuning and analyze resulting metrics

Toolkit Support: Dataset Building Tools (Appendix C.1.3) provide components for creating unsupervised and supervised instruction sets. Statistics submodule functions (Appendix C.1.1.2) verify instruction token limits. Model Fine-Tuning Tools (Appendix C.1.4) enable fine-tuning with constructed datasets and adapter merging. Model Integration Tools (Appendix C.1.5) support quantization, GGUF conversion, and Ollama integration.

E 3.2 Implementation of Symbolic AI Components

E 3.2 Key Activities:

- Identify primary objectives that symbolic components should address based on earlier phase requirements
- For hybrid implementations, analyze sub-symbolic model performance to identify specific limitations addressable by symbolic methods
- Develop a minimal set of core rules directly addressing primary objectives
- Implement rule-based validation for format consistency and structural requirements
- Create symbolic reasoners for handling complex conditional logic and multi-step inference

Implementation Examples: The HAIMEDA application presented in Chapter 5 demonstrates symbolic AI components including rule-based reasoning, pattern matching, and formal verification. These implementations showcase verification mechanisms for data-sensitive domains, with complete source code available in the HAIMEDA project repository.

- Develop pattern matching components for structured information extraction and normalization
- Implement domain-specific grammar enforcement for outputs requiring precise formatting
- Implement constraint satisfaction components for scenarios with multiple interdependent requirements
- Create audit logging mechanisms that track symbolic component decisions for accountability and traceability
- Implement symbolic AI logic as generalizable as possible and inside stand-alone components

E 3.3 Unit and Integration Testing of Components

E 3.3 Key Activities:

- Create comprehensive test cases covering expected input variations for each component
- Apply the same model runtime configurations identified during component testing when evaluating fine-tuned models
- Use verbose intermediate output logging that captures data state between each processing step, enabling verification that individual subcomponents are functioning correctly
- Conduct manual evaluation of outputs for reasoning quality and domain alignment
- Compare component performance against the baseline metrics established in Phase 2
- Develop test suites specifically targeting the integration points between symbolic and sub-symbolic components
- Create edge case test scenarios that deliberately push components to their limits
- Verify that privacy constraints remain intact throughout component processing chains
- Document specific cases where components produce unexpected or incorrect outputs
- Document specific integration limitations to guide subsequent refinement activities

E 3.4 Interface & Pipeline Integration

E 3.4 Key Activities:

- Implement or integrate into the standardized interfaces defined during Phases 1 and 2

- Update the orchestration layer to incorporate the new module into processing pipelines, but without modifying its original structure
- Configure event handlers to trigger appropriate module functionality
- Verify error propagation and handling across module boundaries
- Test module behavior when triggered by system events rather than direct invocation
- Document the module's implementation and all integration points
- Confirm that privacy and security constraints are maintained during cross-module data transfer

Phase 4: Evaluation & Refinement

E 4.1 System-Level Evaluation

E 4.1 Key Activities:

- Create evaluation scenarios directly mapped to original stakeholder requirements
- Conduct end-to-end testing across complete processing pipelines
- Measure system performance metrics including response time, throughput, and resource utilization
- Analyze processing bottlenecks across module boundaries
- Evaluate data transformation accuracy across the full processing chain
- Perform specific tests targeting system behavior under high load or stress conditions
- Conduct information integrity verification tests for hybrid AI scenarios targeting the minimization of hallucinations or missing information
- Compare actual performance against the success criteria established in Phase 1
- Document all identified limitations, categorized by severity and required effort to address
- If possible, collect feedback from stakeholders to validate alignment with expectations
- Analyze gaps between expected and actual performance to inform refinement planning

E 4.2 Refinement Planning

E 4.2 Key Activities:

- Create a prioritized issue matrix categorizing both system limitations and unfulfilled objectives by severity and impact on primary requirements
- Map identified issues to stakeholder requirements from Phase 1
- Implement immediate fixes for minor configuration or localized code issues
- Develop specific remediation plans for high-priority issues requiring significant changes
- Review remaining requirements and select the next module for implementation, if there are no severe issues
- Estimate effort and resources required for each planned refinement and new module
- Document dependencies between refinement tasks and new module implementation
- Identify opportunities to combine refinements with new functionality implementation
- Establish concrete success and evaluation criteria for the next iteration's objectives
- Conduct a requirements review if needed before beginning the next iteration

Phase 5: Deployment & Continuous Improvement

E 5.1 Production Environment Setup & Deployment

E 5.1 Key Activities:

- Create an environment specification document listing all dependencies and version requirements
- If required, implement access control mechanisms aligned with data sensitivity requirements
- Configure model serving infrastructure with appropriate quantization and runtime parameters
- Verify that all symbolic AI rules and constraints remain correctly implemented, and update if needed
- Document production-specific configuration settings and their rationale
- Implement comprehensive monitoring and logging for both technical metrics and AI-specific indicators

- Test system resource consumption under various load conditions to verify production stability, especially for AI components
- Conduct an initial health check verifying all components are operational
- Verify that users can access the system without any issues related to local access rights or firewall rules

E 5.2 User Testing & Feedback Collection

E 5.2 Key Activities:

- Create task-based scenarios that directly test alignment with primary objectives
- Develop structured evaluation forms that assess system performance against defined metrics
- Conduct observation sessions documenting user interaction patterns and pain points
- Implement feedback collection mechanisms directly within the production system
- Perform semi-structured interviews focusing on alignment with user workflows
- Document unexpected usage patterns and creative applications of system capabilities
- Analyze user-generated data to identify patterns and improvement opportunities
- Create systematic categorization of feedback items linked to system components
- Reassess primary objectives with stakeholders based on real-world usage insights
- Collect new requirements and refinement opportunities revealed through actual use
- Document regularly occurring errors or confusion points in user interactions
- Evaluate outputs quality on representative stakeholder tasks and expectations

E 5.3 Continuous Monitoring & Improvement

E 5.3 Key Activities:

- For multi-user systems, configure anonymized data collection for successful user interactions and error cases

- Implement statistical analysis processes for identifying patterns across user interactions
- Implement usage pattern analysis to identify potential refinements for symbolic AI components
- Monitor resource utilization patterns to identify optimization opportunities
- Create periodic review sessions with stakeholders to evaluate collected improvement opportunities
- Document emerging user workflows that differ from originally anticipated patterns
- Establish quantitative thresholds for initiating new development iterations
- Schedule regular technology assessments to evaluate emerging AI techniques and frameworks
- Conduct periodic stakeholder interviews to identify evolving requirements and expectations
- Update the primary requirements document with insights gained from production usage, technological advancements, and stakeholder feedback
- If applicable, reuse existing data transformation pipelines for converting user interactions into fine-tuning datasets
- Conduct a structured review verifying that Phase 1 requirements and architectural decisions remain valid before initiating a new development iteration from Phase 2

C Technical Overview of Toolkit and HAIMEDA

This appendix provides detailed technical specifications for the systems presented in the main text. Section C.1 examines the toolkit components extending the overview from Section 4.4, while Section C.2 documents HAIMEDA's system architecture, core modules, and user interface design with comprehensive implementation details, extending the description of the architecture and core modules from Section 5.1.

C.1 Toolkit Components and Implementation Details

The following subsections provide technical documentation for each toolset within the framework's toolkit: Data Preparation Tools in Section C.1.1, Information Extraction Tools in Section C.1.2, Dataset Building Tools in Section C.1.3, Model Fine-Tuning Tools in Section C.1.4, and Model Integration Tools in Section C.1.5. The complete source code is available in the framework toolkit repository[1].

1: https://github.com/penthooose/framework_toolkit

C.1.1 Data Preparation Tools

C.1.1.1 PII Sanitization Tool

Architecture and Integration The PII Sanitization Tool employs a polyglot architecture combining Elixir Phoenix for the web application layer with Python-based Microsoft Presidio for core NLP functionality. Communication between layers occurs through ErlPort, with the `AnalyzerServer` module maintaining persistent Python processes for optimal performance. The system implements automatic process recovery mechanisms and handles character encoding conversions between Elixir and Python environments.

Language Processing Pipeline Multi-language support utilizes specialized *spaCy* models (`en_core_web_lg` for English, `de_core_news_lg` for German, `fr_core_news_lg` for French, `es_core_news_lg` for Spanish) with automatic language detection on text fragments up to 1 000 characters. The system applies a confidence threshold of 0.8 for language identification, falling back to English processing when confidence is insufficient.

Language-specific tokenization and context rules are applied through a modular NLP configuration provider.

Custom Recognition System The tool implements dual recognition mechanisms through YAML-based configuration files. Pattern recognition utilizes regex patterns with configurable scoring thresholds (default 0.3), while deny-list recognition performs exact match validation. The `PatternRecognizer` module generates word-boundary-aware patterns from example inputs, handling special character escaping and pattern optimization. Custom recognizers support context-aware detection through word proximity analysis. Additionally, they implement Not-entity rules configured in `not_recognizers.yaml` to prevent false positives.

Entity Resolution and Scoring The system resolves overlapping entity detections through scoring-based prioritization, preserving nested entity relationships when appropriate. Each detected entity receives confidence scores based on pattern matching metrics, with position information (using start/end indices) maintained for verification. The tool implements containment rules for substring detection and maintains entity type classification throughout the processing pipeline.

Pseudonymization Implementation Pseudonymization mode leverages the Faker library to generate contextually appropriate replacements. Entity-specific strategies ensure realistic substitutions: person names maintain cultural consistency, phone numbers follow valid formatting patterns, and email addresses preserve domain structures. The system maintains formatting consistency with original entities while ensuring document coherence.

State Management and Persistence The `PII_State` module implements agent-based state synchronization with atomic updates for configuration changes. Entity type registries, label sets, and active configurations persist through serialization mechanisms. The system supports dynamic recognizer updates through registry management and provides self-recovery capabilities for service interruptions.

Batch Processing Architecture The `PII_Industrial` module enables efficient batch processing by handling ZIP archive uploads and processing multiple files in parallel while preserving directory structures. The system generates segment-level analysis with progress tracking for long-running operations. Processing reports include aggregate statistics, confidence distributions, and detailed entity position mappings for compliance documentation.

C.1.1.2 Data Transformation Submodules

This subcollection of Data Preparation Tools comprises various submodules that can operate independently or utilize functions from multiple components. Created specifically for fine-tuning data preparation for LLMs in the HAIMEDA application, these tools are closely tailored to

this use case and serve as implementation examples for initial data preparation.

Data Preparation Controller The *Data Preparation Controller* orchestrates all data preparation tools through a unified API interface. It supports both direct function calls and comprehensive module importing, enabling interactive and programmatic usage. The controller manages dependencies between extraction, conversion, filtering, and partitioning operations while maintaining consistent processing standards across document types. This centralized approach abstracts component integration complexity, allowing developers to define transformation sequences without managing individual tool interactions.

File Extraction The *File Extraction* submodule locates and extracts documents from structured database directories through recursive scanning. It identifies files matching specific patterns, such as *GA*01.docm, and copies them to processing directories while preserving organizational structure. The module provides extraction statistics including file counts, processing times, and directory coverage, with robust error handling and detailed logging for troubleshooting extraction failures in complex hierarchies.

MDB Data Extraction This submodule extracts structured information from Microsoft Access databases to integrate relational data with document content. Using MDB Tools [MDB] through WSL, it processes proprietary formats while handling multiple character encodings, including `cp1252` and `iso8859-1`, with automatic binary data detection. The module sanitizes fields for JSON compatibility, exports data in JSON and JSONL formats, and converts dates, currencies, and special characters appropriately, enabling comprehensive domain knowledge representation in training datasets.

File Conversion The *File Conversion* submodule transforms proprietary formats into standardized MD using Pandoc [Mac], particularly converting DOCM files while preserving document structure. It implements cross-platform execution strategies, handles Windows path conversions and character escaping, and provides detailed logging with error reporting. Intelligent fallback mechanisms ensure reliable processing across diverse operating systems and document complexities.

File Filtering This submodule identifies relevant documents through content analysis and keyword matching. It implements pattern matching tolerant of OCR errors and spacing issues, supports regular expressions, and organizes files into relevant and non-relevant categories while preserving originals. This targeted selection enables effective content filtering for downstream processing.

Data Partitioning The *Data Partitioning* submodule segments documents into logical sections for granular processing. It detects document structure through heading analysis, extracts metadata and references, and

splits content into chapters and subchapters while maintaining hierarchy. The module creates individual section files, handles formatting inconsistencies, performs content normalization, and preserves both sections and complete documents for organized model training.

Data Revision This submodule corrects and updates processed content to improve training data quality. It integrates revised summaries, merges multiple revision sources, and maintains document structure while tracking changes between versions. The module re-triggers downstream processing after revisions, ensuring consistent quality control throughout the pipeline.

Overlength File Removal The *Overlength File Removal* submodule manages documents exceeding LLM context window limits. Using configurable token counting optimized for German technical language, it filters oversized files and generates length statistics. The module offers three handling strategies: truncation with section preservation, document splitting with overlap management, or complete filtering, depending on training requirements. Comprehensive logs document length distributions and handling decisions, optimizing context window utilization while preventing truncation errors.

Training Data Preparation This submodule formats content for optimal model training by standardizing formatting across sources and normalizing text representations, including headings, quotes, and emphasis. It processes paragraph structures to preserve context, prepares MD and JSON formats, handles special characters and escape sequences, and maintains semantic structure while standardizing format across varied inputs.

Q&A Data Preparation The *Q&A Data Preparation* submodule processes question-answer pairs from JSON files for dataset building. It validates question formats, manages prefixes and identifiers, estimates token lengths, and detects related question subitems for merging. The module creates fine-tuning layouts with controlled token counts, prevents duplicate patterns, and filters malformed content, ensuring proper structure for chat assistant capabilities.

Following Chapters Preparation This submodule establishes contextual relationships between document sections to enhance model understanding of content flow. It identifies chapter sequences, determines optimal groupings based on content relationships, and combines related chapters within token constraints. The module creates structures for context-aware training, defining hierarchies and dependencies for both supervised and unsupervised formats. This preparation enables models to understand document progression, particularly beneficial for generating coherent multi-part documents.

Statistics The *Statistics* submodule provides comprehensive analytics on processed collections, generating metrics to evaluate dataset characteristics before fine-tuning. It calculates token distributions, identifying mean, median, and outlier lengths for context window requirements. The module analyzes content type frequencies, terminology prevalence, and formatting patterns, producing visualizations including length distribution histograms, content breakdowns, and processing success rates. These insights guide dataset balancing, identify potential biases, and inform model selection based on document collection characteristics.

C.1.2 Information Extraction Tools

This toolset comprises two major modules: symbolic extraction and sub-symbolic extraction. The symbolic extraction module uses Regex patterns to extract recurring information, while the sub-symbolic extraction module generates new data from existing content using LLMs, such as creating summaries for report chapters or answering chapter-specific questions.

The toolset's primary purpose is to add new information or restructure existing data to create diverse datasets, enabling the development of different specialized models from the same base data. While these extraction tools primarily create datasets for fine-tuning LLMs and sub-symbolic AI modules, they could theoretically support symbolic AI development through rules, ontologies, or other symbolic representations. Both modules are designed for generalizability and can be adapted for other data sources with minor modifications. They include implementation examples with specific regexes and variables used for surveyor report data extraction, retained for illustrative purposes.

Information Extractor

The *Information Extractor* orchestrates all information extraction processes, managing a hybrid pipeline that integrates symbolic and sub-symbolic techniques through a unified interface. It configures file selection using inclusion and exclusion patterns, preprocesses content for optimal extraction quality, and coordinates workflows across extraction stages. The module consolidates information across files while producing both structured JSON and human-readable MD outputs.

Supporting batch and targeted extraction modes, the Information Extractor maintains detailed logging that tracks uncaptured content to identify information gaps. It measures performance metrics throughout the process and ensures consistent transformation from raw documents to structured data, enabling efficient extraction across diverse document types and formats.

Symbolic Extraction

Symbolic Extractor The *Symbolic Extractor* provides a customizable rule-based system for extracting metadata using regular expressions from technical documents. Users can define and insert custom regex patterns

tailored to their specific extraction needs. The current implementation demonstrates this flexibility with over 90 specialized extraction rules for technical and document metadata, using context-aware patterns optimized for German-language technical documentation. These serve as practical examples from the paradigm application rather than fixed components.

The module applies automatic data transformations and normalization to captured values, handling text, dates, and monetary values with appropriate conversion routines. It implements comprehensive exception handling for edge cases and missing data while identifying uncaptured content to improve extraction rules continuously. The extractor architecture supports hierarchical data extraction across nested document structures, preserves extraction provenance for traceability, and offers configurable extraction tolerance with customizable transformation rules. This design enables adaptation to diverse document formats and extraction requirements across different domains and languages.

Metadata Processor The *Metadata Processor* analyzes the presence and distribution of extracted metadata across document subchapters to enrich chapters with contextual information. It maps extracted metadata to relevant document sections through fuzzy matching, accommodating content with varying formats. The module employs word-level detection with partial matching capabilities to identify metadata instances despite formatting inconsistencies.

The processor calculates confidence scores for metadata matches, updates document summaries with distribution analytics, and preserves original metadata values while tracking occurrences throughout the document structure. It validates cross-chapter metadata coherence and processes document trees hierarchically from root to leaf nodes, providing statistical measurements of metadata distribution patterns. In the paradigm application, this module maps data extracted by the Symbolic Extractor from complete reports to individual chapters, storing this information for extended dataset building.

Sub-Symbolic Extraction

Sub-Symbolic Extractor The *Sub-Symbolic Extractor* leverages LLMs to perform AI-powered information extraction through comprehensive content analysis on document collections. Using contextual language understanding, it extracts structured information via prompt-based question answering techniques across both full reports and individual chapters. The module categorizes and processes content according to document structure while preserving semantic relationships between extracted information.

The extractor supports two operational modes: single type extraction, which produces summaries only, and detailed information extraction, which enables question and answer functionality. It implements intelligent token usage limitation through content chunking strategies and includes robust error handling with automatic retry mechanisms for

LLM processing failures. The module generates flexible outputs in both JSON and MD formats.

By dynamically adapting extraction strategies to identified content categories, the extractor preserves document classification metadata for downstream processing. This approach ensures comprehensive information capture across diverse document types while maintaining processing efficiency.

Prompt Creation The *Prompt Creation* module generates context-aware prompts for domain-specific information extraction from technical documentation. It utilizes a JSON configuration file containing multiple prompt snippets and templates specifically designed for medical device assessment reports. While these templates are closely tailored to the paradigm application, they demonstrate effective prompt structure principles applicable to other domains.

Serving as the prompt engineering engine for the Sub-Symbolic Extractor, the module generates prompts based on document classification and employs few-shot learning approaches to improve extraction quality. It manages prompt complexity through content chunking, estimates token usage, and supports multiple extraction scenarios via template-based construction. This configurable approach enables adaptable information extraction while showcasing domain-specific prompt engineering techniques that can be customized for various applications.

C.1.3 Dataset Building Tools

The Dataset Building toolset provides a simple interface that utilizes prepared data from the Data Preparation tools and extracted information from the Information Extraction tools. These tools generate JSONL files containing instruction sets, with one instruction per row in each file. The tools create three dataset types with customizable ratios: training sets for fine-tuning the LLM, validation sets for model validation during fine-tuning, and test sets for post-training evaluation.

Similar to the Data Preparation tools, these modules are closely tailored to the paradigm application and serve primarily as implementation examples demonstrating dataset building techniques and important considerations for fine-tuning. The following subsections present modules corresponding to various fine-tuning phases and stages of the LLMs used in the medical device assessment application, with the concrete phases and stages introduced in Section 5.3.

Unsupervised Datasets

The *Unsupervised Datasets* tools create formatted datasets for self-supervised learning from diverse document sources through three complementary components. `SingleChaptersDS` processes individual document chapters into isolated training examples, preserving categorical structure while validating content through minimum word count thresholds and ensuring

proportional distribution across training splits. `MultipleChaptersDS` handles consolidated multi-chapter content, creating examples that preserve broader document context across chapter boundaries and enable models to learn cross-referential relationships between sections. `MDBDatasetDS` transforms extracted structured MDB records into natural language representations, converting data from multiple tables into consistent textual formats while preserving field-label relationships.

All three components implement consistent text normalization and category-aware sampling for balanced representation, producing standardized JSONL outputs compatible with PyTorch and HuggingFace training pipelines. This comprehensive approach generates diverse unsupervised learning datasets that capture both document-based knowledge and structured database information, establishing robust pretraining foundations for domain-specific language models.

Supervised Datasets

Creating supervised datasets differs from unsupervised dataset creation by requiring multiple elements for each instruction, including task templates, input and output templates, and templates for specialized extracted content such as question-answering pairs or summaries. Both dataset types presented below rotate through multiple task instruction variants to generate diverse training examples while maintaining consistent structure.

Mixed Chapters Datasets The *Mixed Chapters Datasets* components create supervised training data by combining multiple document chapters with structured instructions. `MixedChapters` selects and combines chapter content while preserving hierarchical relationships and maintaining semantic coherence between sections. `MixedInstructionCreation` generates format-specific instructions that teach models to interpret various document structures. Together, these components produce training examples that enable models to recognize and process different document formatting standards, sectioning conventions, and organizational patterns in complex structured content.

Q&A Datasets The *Q&A Datasets* components create specialized question-answering training data from structured content sources. `QADataset` extracts and organizes question-answer pairs from documents, grouping related questions with relevant context passages and implementing intelligent handling of unavailable answers through diverse replacement strategies. It structures content with consistent question formatting and balanced dataset distribution. `QAInstructionCreation` transforms these pairs into standardized instruction-following formats optimized for fine-tuning, using template-based formatting with clearly defined sections for context, questions, and answers.

Coherence Datasets

The *Coherence Datasets* component creates specialized training data to ensure proper alignment and coherent behavior in merged models. `MixedCoherenceDS` integrates multiple dataset sources, balancing supervised and unsupervised content while incorporating instructions from validation and test sets of other datasets that the model has not previously encountered. It implements sampling algorithms ensuring representative distribution across diverse instruction types and automatically processes different instruction formats within a unified framework. The module filters instructions based on configurable token limits, creates statistically balanced training splits, and ensures equal representation through proportional sampling.

Instruction Utilities

The *Instruction Utilities* components provide essential tools for managing and optimizing instruction datasets throughout the fine-tuning pipeline. `MergeInstructions` combines instruction data from multiple source directories into unified datasets with balanced representation. It creates statistically distributed training, validation, and test splits while shuffling instructions for random distribution. The module verifies data integrity during merging, produces detailed distribution reports, and efficiently handles large-scale dataset operations using consistent file naming conventions.

`InstructionPreparation` analyzes and optimizes datasets through token length estimation and filtering to ensure compatibility with model context windows. It processes both supervised and unsupervised formats, estimates token lengths using configurable ratio approximations, and generates comprehensive statistics on token distribution. Together, these utilities form the final processing stage before fine-tuning, ensuring dataset quality, proper distribution, and technical compatibility with model training requirements.

C.1.4 Model Fine-Tuning Tools

The Model Fine-Tuning Tools utilize the PyTorch framework, employing Elixir over ErlPort to initiate processes. Fine-tuning focuses on working with PEFT and creating LoRA adapters that can subsequently merge with base models. Beyond fine-tuning, the tools provide functionality for merging base models with adapters and combining two models using MergeKit [God+24] and SLERP.

The `FineTuningController` serves as the central Elixir module orchestrating the entire fine-tuning and merging process, providing a consistent API for both interactive and programmatic usage. It communicates with the Python environment through `GatewayAPI`, which establishes a reliable bridge between Elixir and Python for executing ML operations with proper error handling and progress reporting. Configuration relies on structured JSON files for both fine-tuning and merging operations,

defining hyperparameters, dataset paths, optimization strategies, and hardware acceleration options.

Fine-Tuning using PyTorch

The *PyTorch Fine-Tuning* module implements parameter-efficient training using LoRA to optimize memory usage while effectively adapting models to specific domains. The system supports multiple training modes, including unsupervised, supervised, and mixed approaches, with specialized handling for different dataset types. During training, it provides real-time progress updates and manages checkpoint creation, enabling training resumption and model evaluation throughout the process. This integrated approach simplifies the complex task of fine-tuning language models while maintaining the flexibility needed for creating domain-specialized AI capabilities.

Merging Models and Adapters

The *Model Merging* module implements sophisticated techniques, including SLERP, for effectively combining knowledge from different models while maintaining coherent behavior. The system supports both direct adapter integration, where LoRA weights merge into the base model, and model-to-model merging, which combines separate model instances. Throughout the merging process, the system manages precise data typing requirements and provides detailed progress reporting. The resulting merged models maintain compatibility with deployment pipelines while incorporating specialized capabilities from multiple training stages, effectively combining general language understanding with domain-specific expertise.

C.1.5 Model Integration Tools

The Model Integration Tools provide functionalities for quantizing models to enable execution in resource-limited GPU environments or to run multiple models simultaneously. These tools include functionality for integrating models into deployment environments such as Ollama or LMStudio, which utilize the GGUF file format. The toolkit includes conversion from safetensors format to GGUF format using the llama.cpp API. Similar to the Model Fine-Tuning Tools, an Elixir controller manages the conversion, integration, and quantization processes from a high-level perspective, with specific parameters configured through JSON files.

Model Conversion and Integration

The *Model Conversion* module transforms models from safetensors to GGUF format using a two-tier architecture. `ModelConverter` coordinates the workflow while `ConvertWrapper` interfaces directly with llama.cpp utilities. The Python wrapper provides robust error reporting, automatic script location detection across environments, and consistent parameter handling. This design separates high-level process management from

technical implementation, creating a reliable pipeline for model format conversion.

Model Quantization

The *Model Quantization* module applies compression techniques through a similar two-tier architecture. `ModelQuantizer` manages the high-level process while `QuantizeWrapper` executes the compression operations. This approach leverages Python's specialized libraries while maintaining consistent process control within the Elixir framework. The module supports various precision-speed trade-offs and ensures compatibility with downstream deployment systems, enabling efficient model execution across hardware with varying resource constraints.

C.2 HAIMEDA System: Design, Architecture, and Components

This section provides additional technical details on the HAIMEDA application introduced in Section 5.1. Further information on configuration management in the `MainController` is presented in Section C.2.1, while secondary modules and user interface components are described in Section C.2.2. The full implementation is available in the HAIMEDA project repository[2].

2: https://github.com/penthooose/HAIMEDA

C.2.1 Configuration Management in the MainController

Central to `MainController` operations is the application properties configuration system, managed through the `application_properties.yaml` file. This centralized configuration enables dynamic parameter adjustment without code modifications and ensures consistent behavior across all modules. The controller extracts configuration parameters through specialized helper functions and passes these variables as input parameters to module calls.

The YAML file allows users to configure a wide range of critical components. Key features include:

- **General settings:** Control console verbosity and performance output.
- **IIVM options:** Enable or disable hybrid post-processing for information integrity verification.
- **LLM infrastructure:** Specify paths to local GGUF models, toggle remote model usage, set Ollama server URLs, and manage model overwriting behavior.
- **RAG configuration:** Define vector database usage, parent folder paths, subcollection mappings, chunking strategies, embedding model selection, and maximum context size for retrieval-augmented generation.

- **Model parameters:** Set global and task-specific model names and runtime parameters (e.g., temperature, top_*p*, top_*k*, max_tokens, repeat_penalty) for general, RAG, and specialized tasks such as text optimization, revision, and summarization.
- **Module-specific settings:** Fine-tune validation rules for IIVM, select prompt templates for RAM, and configure response formatting for RIM.

The configuration system implements a hierarchical structure where global settings can be overridden by task-specific parameters, allowing granular control over system behavior. Default configurations ensure system operability even when custom parameters are not specified, while validation mechanisms prevent invalid configurations from disrupting system operations. This architectural approach enables seamless addition of new AI modules while maintaining consistent communication protocols through centralized configuration management.

C.2.2 Secondary Modules and Components of HAIMEDA

C.2.2.1 Gateway API Module

The GAM serves as a communication bridge enabling seamless integration between Elixir-based processing and Python-based components across the HAIMEDA application. It abstracts the complexities of cross-language communication while providing reliable data exchange for advanced operations across multiple modules.

The module implements comprehensive ErlPort management for Python process lifecycle handling, including process spawning, communication, and termination with proper resource cleanup. It supports multiple concurrent operations through connection pooling, ensuring efficient resource utilization during intensive processing tasks. GAM handles complex data structure serialization between Elixir and Python, enabling sophisticated information exchange while maintaining type consistency and data integrity across language boundaries.

This integration layer serves as a universal connector for any Python functionality needed throughout the application, providing a standardized interface that can be utilized by any module requiring Python capabilities. Through robust error handling with automatic retry mechanisms and graceful degradation strategies, the Gateway API ensures reliable Python integration even during temporary resource constraints or processing failures.

C.2.2.2 LLM Service Module

The LSM serves as the central API layer for all LLM interactions within HAIMEDA, providing a unified interface for specialized modules to interact with various language models through the Ollama platform. It abstracts LLM operations and manages the complete lifecycle of local

model integration while providing standardized communication protocols across the entire application ecosystem.

LSM implements comprehensive model management capabilities including automatic GGUF model integration, lifecycle management, and dynamic model fallback using similarity matching when specified models are unavailable. The module handles structured message processing with comprehensive error handling incorporating automatic retry mechanisms, graceful degradation strategies, and detailed logging for debugging and monitoring purposes.

Through its integration with the Ollama platform, LSM provides seamless access to both custom or fine-tuned language models and general-purpose embedding models optimized for RAG operations. The module supports template-based prompt construction with variable substitution, enabling consistent prompt engineering across different AI tasks. It implements German-optimized token counting and estimation algorithms, providing accurate context size management for various data formats. For Ollama integration, the module reuses the same singleton Ollama client to the Ollama REST API available over localhost for all inference tasks, efficiently handling multiple requests and responses without creating new instances for each interaction.

C.2.2.3 Feedback Module

The FBM establishes bidirectional communication between the surveyor and the AI system, providing real-time status updates and interactive functionality throughout HAIMEDA. It processes outputs from various AI components through two primary channels: a status log for system-wide notifications and a chat interface for AI assistant interactions. For verification results, FBM implements sophisticated formatting with percentage calculations, color-coding, and context-appropriate displays for both Hypertext Markup Language (HTML) and text formats.

Through Phoenix LiveView integration, the module delivers updates without requiring page refreshes, maintaining interface responsiveness even during intensive AI processing tasks. This real-time communication system ensures users receive immediate feedback about operations while enabling continuous monitoring of complex AI workflows during report creation or research work.

C.2.2.4 User Interface and Interaction Design

The HAIMEDA application implements a sophisticated web-based interface using Phoenix LiveView technology, providing real-time interactive report editing capabilities with integrated AI assistance. The interface design prioritizes workflow efficiency, focusing on the specialized needs of medical device assessment while maintaining intuitive navigation and responsive feedback mechanisms.

Editor Environment

The editor environment centers around a core Editor module that orchestrates all GUI operations and content management. It features a comprehensive tab management system with a left-side navigation panel providing access to stakeholder information, device metadata, case data, and report chapters. This system supports multiple document sections with lazy loading and session persistence, enabling surveyors to navigate efficiently between report components. The interface provides specialized sections for metadata management, party statement documentation, and chapter content editing, with collapsible panels optimizing screen real estate.

The environment implements session-based workflow management, enabling users to maintain context across application restarts by preserving editor states, user preferences, and active content. This persistent session architecture supports multiple concurrent reports with isolated workspace management, crucial for surveyors handling multiple assessment cases simultaneously. The editor coordinates bidirectional communication between UI components and AI modules through the `MainController`, ensuring responsive feedback during intensive processing operations while maintaining UI synchronization across concurrent tasks.

TipTap Editor

The TipTap editor, embedded within the output textfield area for each tab, provides advanced rich text editing capabilities with specialized reporting features. It implements bidirectional transformation between plain text and formatted content with entity preservation, enabling sophisticated content manipulation while maintaining document structure integrity. The editor features interactive entity highlighting with color-coded visual feedback for different entity types and their states, including verification states of identified missing entities displayed in a list below the generated text.

For AI-assisted content verification, the editor integrates with the PostProcessor module's correction system, rendering AI suggestions as interactive entities with replacement options. This allows surveyors to efficiently review and incorporate AI-generated corrections through click-to-replace functionality or removal with dropdown selection for alternatives. The system supports multiple content versions with intuitive navigation, simple version creation, and deletion capabilities. Through JS integration with Elixir components, the editor maintains real-time synchronization between user edits and backend processing, ensuring content integrity throughout the verification workflow while directly saving content changes to the database.

Status Logs and Chat Assistant

The Status Log component provides real-time monitoring of system operations with categorized, timestamped entries for different message types including success confirmations, warnings, errors, and AI processing notifications. This enables surveyors to track AI operations and system events with proper context while maintaining a comprehensive activity history. The log implements automatic scrolling with icon indicators and color-coding for clear visual differentiation between message categories.

The Live Chat assistant component offers bidirectional communication with the RIM module, supporting natural language queries about regulatory requirements, similar assessment cases, and technical specifications. The interface implements distinct visual styling for different message sources, including user, system, symbolic AI, subsymbolic AI, and hybrid AI, clarifying the origin of each response.

D Extended Evaluation Results and Technical Analyses

This appendix provides comprehensive extensions to the evaluation presented in Chapter 6. Section D.1 expands upon the framework and HAIMEDA evaluation, while Section D.2 offers detailed technical analyses and comprehensive parameter listings that complement the results and limitations discussed in the fine-tuning phases from Section 5.3.

D.1 Detailed Results for Framework and Paradigm Evaluation

This section presents the complete evaluation results in three parts. Section D.1.1 contains the full gap analysis of framework principles in Table D.1, discussed in Section 6.1. Section D.1.2 provides extended performance evaluation details for HAIMEDA, expanding on Section 6.2.1. Finally, Section D.1.3 presents the comprehensive SUS assessment results for HAIMEDA, which were summarized in Section 6.2.2.

D.1.1 Framework Principles Analysis

Table D.1: Comprehensive gap analysis of framework principles, established theories, literature references, and addressed extensions throughout the development lifecycle phases.

Framework Principle	Established Theory	Literature Reference	Gap Addressed/Extension
Modularization	Modular Programming	Parnas, D.L. (1972) "On the Criteria to Be Used in Decomposing Systems into Modules" [Par72]	Extends theory to hybrid AI context with heterogeneous internal module design
Agile Development for AI	Agile Methodology	Beck et al. (2001) "Manifesto for Agile Software Development" [Bec+01]	Addresses gap between traditional agile and AI development uncertainty
Organize Requirements by Priority	Goal-Oriented Requirements Engineering	Van Lamsweerde, A. (2001) "Goal-Oriented Requirements Engineering: a guided tour" [Lam01]	Extends with specific focus on data sensitivity classification
Examine Data Before Designing	Data-Driven Design	Patil, D. (2012) "Data Jujitsu: The Art of Turning Data into Product" [Pat12]	Addresses gap between data science and system architecture processes
Keep Options Open	Set-Based Concurrent Engineering	Sobek et al. (1999) "Toyota's Principles of Set-Based Concurrent Engineering" [SWL99]	Extends to AI-specific selection of processing methodologies
Identify Hybrid Integration Points Early	Hybrid AI Systems	Dellermann et al. (2019) "Hybrid Intelligence" [Del+19]	**Novel contribution**: proactive approach to complementary AI technique selection
Build In Privacy From the Start	Privacy Engineering	Cavoukian, A. (2009) "Privacy by Design: The 7 Foundational Principles" [Cav09]	Extensions for hybrid AI data flows and model training with sensitive data
Build Modular, Not Monolithic	Component-Based Software Engineering	Heineman & Councill (2001) "Component-Based Software Engineering: putting the pieces together" [HC01]	Extends with AI-specific component boundaries and interaction patterns
Coordinate Through the Orchestration Layer	API Gateway Pattern	Richardson, C. (2018) "Microservices Patterns" [Ric18]	**Novel contribution**: unified orchestration specifically for heterogeneous AI components
Match Training to Real Use	ML Training/Serving Skew	Sculley et al. (2015) "Hidden Technical Debt in Machine Learning Systems" [Scu+15]	Extends with specific focus on LLM instruction/inference consistency
Combine Symbolic with Sub-symbolic Strengths	Neuro-Symbolic AI	Besold et al. (2021) "Neural-Symbolic Learning and Reasoning: A Survey and Interpretation" [Bes+21]	**Novel application** of complementary strengths in domain-specific contexts
Make Symbolic Processing Transparent	Explainable AI	Gunning, D. (2019) "XAI — Explainable artificial intelligence" [Gun+19]	Extensions for hybrid verification approaches combining symbolic and sub-symbolic elements
Test in Real-World Conditions	Contextual Design	Beyer & Holtzblatt (1997) "Contextual Design: Defining Customer-Centered Systems" [BH97]	Extends with AI-specific evaluation criteria for domain applicability
Implement Comprehensive Monitoring	Observability in Distributed Systems	Ligus, S. (2012) "Effective Monitoring and Alerting" [Lig12]	**Novel contribution** for AI-specific metrics and multi-level indicators
Evolve Based on Usage Patterns	Continuous Improvement	Deming, W.E. (1986) "Out of the Crisis" [Dem86]	**Novel approach** to AI model evolution based on usage patterns and feedback loops

D.1.2 HAIMEDA System Performance Evaluation

Table D.2: Detailed performance metrics for HAIMEDA modules and operations, including duration, memory usage, and peak CPU/GPU/VRAM consumption for key system tasks.

Module	Operation	Duration (sec)	Process Memory (MB)	Peak CPU (%)	Peak GPU (%)	Peak VRAM (MB)
System	Initialization	0.90	110.3	23.0	15.0	1,261
IIVM	Pre-processing Input	0.01	100.4	13.4	15.4	19,174
IIVM	Post-processing Output (Symbolic)	0.02	100.7	12.4	15.4	19,150
IIVM	Post-processing Output (Hybrid)	16.58	117.6	26.0	27.0	2,661
RAM	Create Chapter	14.16	102.9	22.6	94.0	19,163
RAM	Revise Chapter Text	6.13	105.2	29.0	94.0	19,190
RAM	Optimize Text	8.75	100.4	30.2	93.8	19,170
RAM	Process User Request (RIM)	19.6	118.5	25.0	93.8	18,350
RIM	Process User Request (RAG)	5.01	116.9	40.0	32.0	18,471
RIM	Process User Request (MDB)	7.91	134.8	30.4	17.0	18,355

Note: Values represent averages across multiple test runs ($n = 50$). System specifications: AMD Ryzen 9 5900X (12 Cores), 64GB RAM (DDR4), Nvidia RTX 4090 with 24GB VRAM.

D.1.3 Usability Assessment of HAIMEDA

Table D.3: System Usability Scale (SUS) results for HAIMEDA, showing the expert's responses to ten usability statements and the calculated raw SUS score.

Statement	Strongly Disagree 1	Disagree 2	Neutral 3	Agree 4	Strongly Agree 5
1. I think that I would like to use this system frequently.					X
2. I found the system unnecessarily complex.			X		
3. I thought the system was easy to use.			X		
4. I think that I would need technical support to use this system.			X		
5. I found the various functions in this system were well integrated.				X	
6. I thought there was too much inconsistency in this system.		X			
7. I would imagine that most people would learn to use this system very quickly.			X		
8. I found the system very cumbersome to use.		X			
9. I felt very confident using the system.				X	
10. I needed to learn a lot of things before I could get going with this system.				X	

Note: SUS assessment conducted with the surveyor after testing HAIMEDA during regular surveyor activities. Raw SUS score: [62.5] /100.

D.2 Fine-Tuning Implementation Details and Outcomes

This section provides a comprehensive technical analysis of HAIMEDA's three fine-tuning phases described in Section 5.3. For each phase, it presents detailed configuration parameters, evaluation metrics, outcome analyses, and identified limitations. The analysis progresses through Section D.2.1 for the first phase, Section D.2.2 for the second phase, and Section D.2.3 for the third phase.

D.2.1 Phase 1: Initial Fine-Tuning Approach

D.2.1.1 Hyperparameters and Configuration

Table D.4: Fine-tuning parameters for Phase 1.

Parameter	Value
Training Configuration	
Mode	Supervised
Base Model	Llama3-DiscoLeo-Instruct-8B
Max Sequence Length	3000
Dataset Focus	Mixed instruction data
Use Checkpoint	No
Training Hyperparameters	
Epochs	3
Learning Rate	2×10^{-5}
Batch Size	1×8
Warmup Steps	100
LR Scheduler	Cosine
Max Gradient Norm	1.0
Eval Steps	301
Save Steps	100
Save Total Limit	3
Gradient Checkpointing	Yes
Weight Decay	0.01
LoRA Configuration	
Rank (r)	16
Alpha	32
Dropout	0.05
Target Modules	7 modules

D.2.1.2 Training Metrics and Loss Curves

Table D.5: Training duration, throughput, and learning rate metrics for Phase 1 fine-tuning.

Metric	Value
Training Duration	5.75 hours
Samples per Second	0.422
Steps per Second	0.053
Initial Learning Rate	2.0×10^{-6}
Peak Learning Rate	2.0×10^{-5}
Final Learning Rate	2.0×10^{-10}

Note: Learning rate followed a cosine schedule with warmup, peaking at epoch 0.27 and decaying to near-zero by completion.

Table D.6: Evaluation loss at key checkpoints during Phase 1 fine-tuning, showing model performance progression across epochs.

Checkpoint	Epoch	Eval Loss
Pre-Training	0.00	1.6263
Validation 1	0.83	0.4513
Validation 2	1.65	0.3902
Validation 3	2.48	0.3707
Final Test	3.00	0.3962

Note: Metrics collected during fine-tuning of the Llama 3 8B IT model. The decreasing evaluation loss indicates improving model performance through the training process.

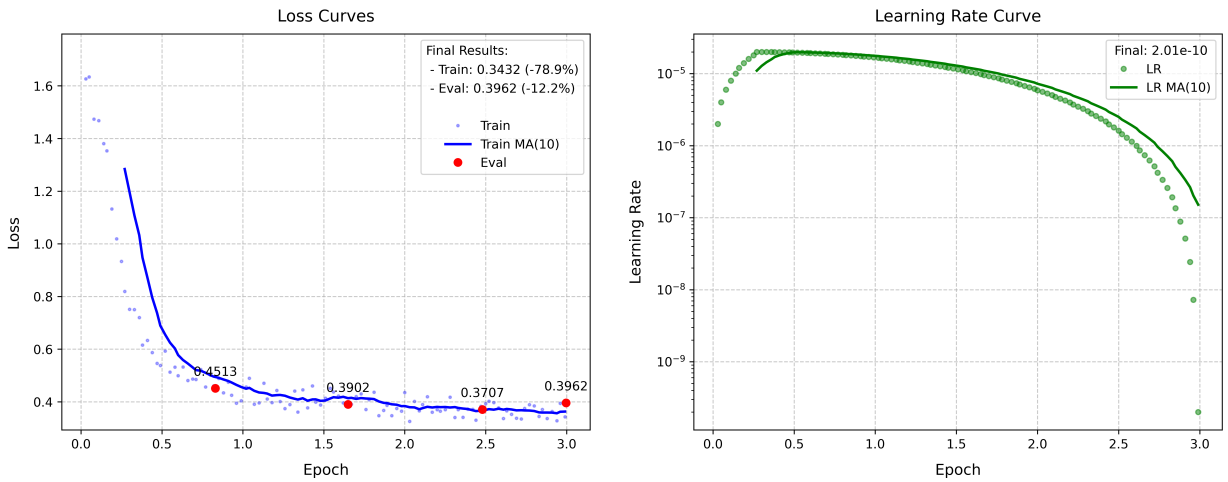


Figure D.1: Training dynamics during Phase 1 fine-tuning: evolution of evaluation loss and learning rate across training epochs.

D.2.1.3 Technical Analysis and Challenges of Fine-Tuning Phase 1

The initial fine-tuning phase revealed a critical disconnect between promising quantitative metrics and practical performance, highlighting fundamental challenges in language model adaptation. Despite achieving a 75.6 % loss reduction from 1.626 to 0.396 across three epochs, as shown in Table D.6, the model failed to generate usable content in practice.

The fine-tuning configuration, detailed in Table D.4, employed the `Llama3-DiscoLeo-Instruct-8B` base model with PEFT using LoRA configured with $r = 16$ and $= 32$. This approach optimized memory utilization while maintaining adaptation capacity. Training dynamics revealed slow processing at 0.422 samples per second with high memory demands, despite implementing gradient checkpointing and 8-bit quantization, as documented in Table D.5.

The learning rate followed a carefully designed cosine decay pattern with warm-up, starting at 2.0×10^{-6}, peaking at 2.0×10^{-5} approximately 27 % through the first epoch, and decaying to 2.0×10^{-10} by completion. This schedule enabled aggressive initial adaptation while preventing catastrophic forgetting in later epochs. The loss curve illustrated in Figure D.1 exhibited characteristic power-law behavior, with 54 % of total loss reduction occurring during the first third of epoch 1, followed by diminishing returns. This pattern aligns with established neural network convergence behavior where early training captures broad statistical patterns before refining specific relationships [GBC16, pp. 245f].

However, systematic inference testing exposed a fundamental failure mode: the model consistently generated only chapter titles without corresponding content. At temperature settings below 0.7, outputs terminated immediately after producing accurate titles. Only at higher temperatures exceeding 0.7 did the model occasionally continue generating full chapters, though with inconsistent quality. This behavior occurred despite validation loss improving from 0.4513 at epoch 0.83 to 0.3707 at epoch 2.48, demonstrating how standard metrics can mask critical behavioral issues.

Root cause analysis identified the training data formatting as the primary culprit. Each example began with a chapter title followed by two consecutive newlines before the main content, creating an artificial prediction boundary. The model optimized for the simpler task of title prediction based on input metadata rather than generating coherent chapter content. This limitation exemplifies the opacity of language model training dynamics and the insufficiency of standard loss metrics in detecting behavioral pathologies [Lip18, pp. 36f, 39f].

Temperature sensitivity testing provided additional insights into the model's learned behavior. Temperatures below 0.7 resulted in virtually all outputs terminating after titles, temperatures between 0.7 and 1.0 produced full chapters approximately 15 % of the time, and temperatures above 1.0 generated longer but increasingly incoherent content. This pattern indicated the model had learned a strong probabilistic boundary at

the newline characters, with temperature essentially controlling whether sampling would overcome this artificial stopping point.

These findings directly informed subsequent fine-tuning phases by establishing three critical requirements: redesigning instruction templates to eliminate artificial stopping points, implementing explicit prompting to delineate expected output structure, and developing specialized training objectives that balance title accuracy with comprehensive content generation. The results underscore how seemingly minor formatting decisions can significantly alter model behavior in ways traditional metrics fail to capture [RSG16, pp. 1136ff], necessitating the iterative, experimental approach with frequent intermediate testing that is fundamental to the ADR methodology employed throughout this research.

D.2.2 Phase 2: Multi-Stage Fine-Tuning and Optimization

D.2.2.1 Hyperparameters and Configuration

Table D.7: Fine-tuning parameters for Phase 2. Bold values indicate stage-specific changes or optimizations.

Parameter	Stage A	Stage B	Stage C	Stage D	Stage E
Training Configuration					
Mode	Unsupervised	Unsupervised	Unsupervised	**Supervised**	**Supervised**
Base Model	Llama3-DiscoLeo-8B	Stage A output	Stage B output	Stage C output	Stage C output
Max Sequence Length	1100	**3000**	3100	3200	3200
Dataset Focus	Basic MDB texts	Single chapters	Multiple chapters	Chapters & summaries	Chapters
Use Checkpoint	No	**Yes**	No	No	**Yes**
Training Hyperparameters					
Epochs	3	3	4	4	5
Learning Rate	2×10^{-5}	2×10^{-5}	2×10^{-5}	$\mathbf{1.5 \times 10^{-5}}$	$\mathbf{1.5 \times 10^{-5}}$
Batch Size	1 × 8	1 × 8	1 × 8	1 × 8	1 × 8
Warmup Ratio	0.03	0.03	0.03	**0.05**	**0.08**
LR Scheduler	Cosine	Cosine	Cosine	Cosine	**Cosine w/restarts**
Max Gradient Norm	0.3	0.3	0.3	0.3	0.3
Eval Steps	200	350	45	210	210
Save Steps	100	200	40	200	200
Save Total Limit	—	—	—	3	5
Gradient Checkpointing	Yes	Yes	Yes	Yes	Yes
LoRA Configuration					
Rank (r)	16	16	16	**24**	**24**
Alpha	32	32	32	**48**	**48**
Dropout	0.10	0.10	**0.15**	0.10	0.10
Target Modules	All (10)	**Reduced (8)**	All (10)	All (10)	All (10)

D.2.2.2 Training Metrics and Loss Curves

Table D.8: Training duration, throughput, and learning rate metrics for each fine-tuning stage (A–E) in Phase 2.

Metric	Stage A	Stage B	Stage C	Stage D	Stage E
Training Duration	2.05 hours	11.45 hours	1.13 hours	6.38 hours	12.9 hours
Samples per Second	0.921	2.727	0.392	1.769	0.893
Steps per Second	0.115	0.341	0.048	0.048	0.112
Initial Learning Rate	2.0×10^{-6}	2.0×10^{-6}	2.0×10^{-6}	1.5×10^{-6}	1.2×10^{-6}
Peak Learning Rate	2.0×10^{-5}	2.0×10^{-5}	2.0×10^{-5}	1.5×10^{-5}	1.5×10^{-5}
Final Learning Rate	4.5×10^{-9}	5.37×10^{-6}	1.9×10^{-7}	1.33×10^{-5}	3.1×10^{-6}

Note: Stages A–D used cosine learning rate schedules, while Stage E employed a cosine schedule with restarts.

Table D.9: Evaluation metrics during fine-tuning Phase 2, showing loss values at key checkpoints for each training stage (A–E).

Checkpoint	Stage A		Stage B		Stage C		Stage D		Stage E	
	Epoch	Loss	Epoch	Loss	Epoch	Loss	Epoch	Loss	Epoch	Loss
Pre-Training	0.00	2.4333	0.00	2.0385	0.00	0.6657	0.00	1.3087	1.00	1.2581
Validation 1	0.72	0.8018	0.35	0.9032	0.90	0.7292	0.20	0.5698	1.01	0.4521
Validation 2	1.43	0.7261	0.69	0.8257	1.81	0.7204	0.39	0.5297	1.21	0.4340
Validation 3	2.15	0.6949	1.04	0.7961	2.71	0.7164	0.59	0.5087	1.42	0.4223
Validation 4	2.86	0.6873	1.39	0.7730	3.62	0.7142	0.79	0.4922	1.62	0.4122
Validation 5	–	–	1.74	0.7594	–	–	–	–	1.82	0.3846
Validation 6	–	–	2.08	0.7525	–	–	–	–	2.02	0.3908
Validation 7	–	–	2.43	0.7450	–	–	–	–	2.23	0.3764
Validation 8	–	–	2.78	0.7432	–	–	–	–	2.43	0.3689
Validation 9	–	–	–	–	–	–	–	–	2.63	0.3622
Validation 10	–	–	–	–	–	–	–	–	2.83	0.3564
Validation 11	–	–	–	–	–	–	–	–	3.04	0.3528
Validation 12	–	–	–	–	–	–	–	–	3.24	0.3475
Validation 13	–	–	–	–	–	–	–	–	3.44	0.3440
Validation 14	–	–	–	–	–	–	–	–	3.65	0.3414
Validation 15	–	–	–	–	–	–	–	–	3.85	0.3392
Validation 16	–	–	–	–	–	–	–	–	4.05	0.3376
Validation 17	–	–	–	–	–	–	–	–	4.25	0.3368
Validation 18	–	–	–	–	–	–	–	–	4.45	0.3363
Validation 19	–	–	–	–	–	–	–	–	4.66	0.3360
Validation 20	–	–	–	–	–	–	–	–	4.86	0.3360
Final Test	3.00	0.6880	3.00	0.7653	3.94	0.6389	1.00	0.4779	5.00	0.3224

Note: Stages A–C (unsupervised training) used progressively more complex corpus data, while Stages D–E (supervised training) demonstrated significant improvements with instruction fine-tuning.

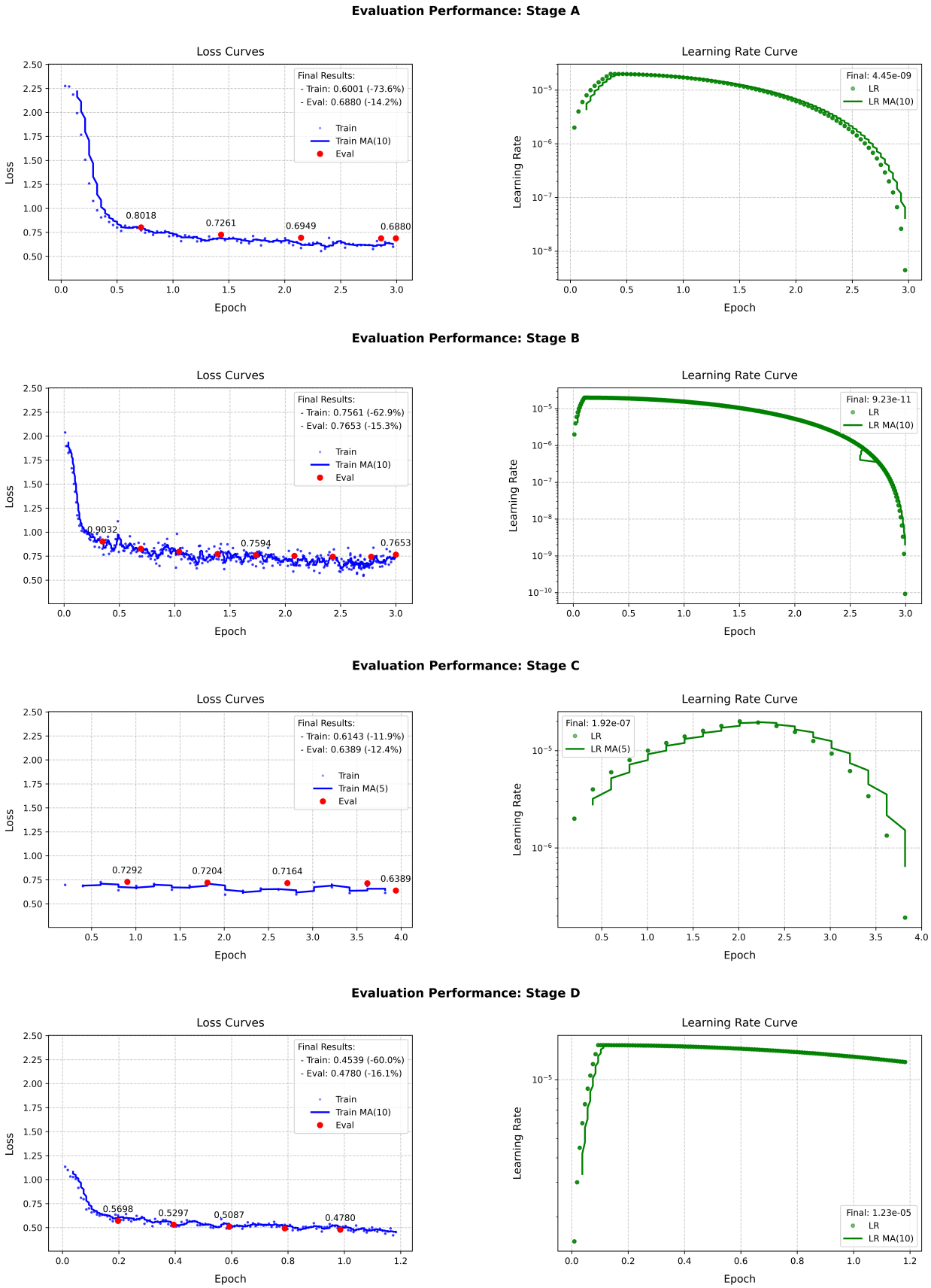


Figure D.2: Training dynamics during Phase 2 fine-tuning: evolution of evaluation loss and learning rate across training epochs for Stages A through D.

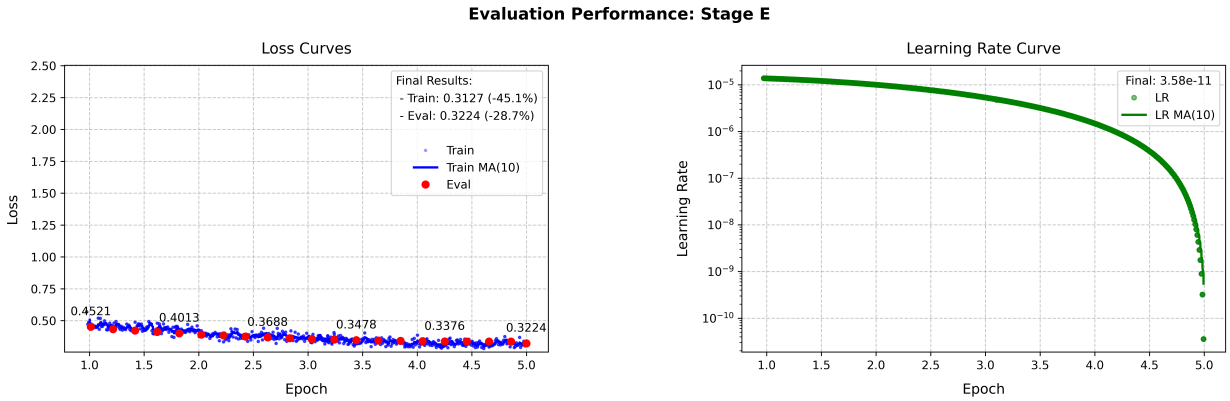


Figure D.3: Training dynamics during Phase 2 fine-tuning: evolution of evaluation loss and learning rate across training epochs for Stage E.

D.2.2.3 Technical Analysis and Challenges of Fine-Tuning Phase 2

Phase 2 implemented a sophisticated multi-stage approach to address the limitations discovered in Phase 1, employing five sequential stages. The strategy consisted of three unsupervised stages focusing on domain knowledge acquisition, followed by two supervised stages targeting specific instruction formats, with comprehensive hyperparameters documented in Table D.7.

The training progression methodically built domain expertise before transitioning to instruction-following capabilities. Stage A established foundational knowledge using MDB texts with a moderate sequence length of 1 100 tokens, achieving a 71.7 % loss reduction from 2.433 to 0.688. Stage B extended this foundation with single chapters and increased the sequence length to 3 000 tokens, resulting in a 62.5 % loss reduction. Stage C implemented multiple chapter contexts but yielded minimal improvement at 4.0 %, suggesting diminishing returns for unsupervised learning once robust domain knowledge was established.

Training dynamics varied significantly across stages, as shown in Table D.8. Stage B achieved notably higher throughput at 2.727 samples per second compared to Stage C's 0.392. Learning rate strategies evolved progressively, culminating in Stage E's sophisticated cosine scheduler with restarts that maintained optimization effectiveness through five epochs.

Validation metrics in Table D.9 along with Figures D.2 and D.3 reveal distinct learning patterns. Stages A and B exhibited rapid initial convergence followed by gradual improvement, while Stage C's minimal loss reduction from 0.666 to 0.639 indicated approaching optimal boundaries for unsupervised domain learning. The supervised stages demonstrated more consistent improvement, with Stage E's extended training showing continuous gains across 20 validation checkpoints and achieving the best final loss of 0.322, representing a 74.4 % reduction.

Architectural refinements in the supervised stages included increased LoRA dimensions from $r = 16$ and $= 32$ to $r = 24$ and $= 48$, alongside

progressively tuned warm-up ratios from 0.03 to 0.08. These modifications created more robust adaptation capacity while preventing catastrophic forgetting of previously acquired knowledge.

Testing revealed significant improvements over Phase 1, with the model reliably generating complete chapter content rather than only titles. However, critical limitations emerged that impacted practical deployment. The model successfully produced text with appropriate medical terminology but frequently struggled with structural consistency across different chapter types and maintaining coherence when processing multiple context chapters.

These limitations stemmed from several interconnected factors identified through systematic analysis. The sequential five-stage process likely induced catastrophic forgetting, causing the model to gradually lose document structure understanding while optimizing for domain-specific content [Kir+17, pp. 3521f]. Additionally, unstructured database tables from MDB conversion introduced format inconsistencies that confused the model's understanding of document organization [Che+23, pp. 1, 4f]. The multi-objective training created competing optimization goals, potentially prioritizing content accuracy over structural consistency [MGS23, pp. 1f]. Distribution shifts between training stages contributed to inconsistent structural capabilities, while extended context lengths may have fragmented the model's ability to maintain coherent document structure across longer sequences [Liu+24, pp. 12252ff].

The minimal improvement in Stage C validated a crucial insight: after sufficient domain knowledge acquisition, unsupervised training yields diminishing returns, indicating that supervised fine-tuning offers more efficient improvement pathways for specialized document generation. These findings directly influenced the transition to Phase 3, which adopted a specialized approach with parallel model variants focusing on different capabilities rather than sequential training. This strategic shift addressed the observed pattern where competing optimization goals in Phase 2 limited simultaneous performance in both knowledge representation and structural formatting.

D.2.3 Phase 3: Specialized Model Training and Merging

D.2.3.1 Hyperparameters and Configuration

Table D.10: Fine-tuning parameters for Phase 3.

Parameter	V1		V2		V3 (Coherence)	
	Stage A	Stage B	Stage A	Stage B	Stage A	Stage B
Training Configuration						
Mode	Unsupervised	Supervised	Supervised	Supervised	Unsupervised	Supervised
Base Model	Llama3-DiscoLeo-8B	V1-A output	Llama3-DiscoLeo-8B	V2-A output	Merged V1+V2 Model	V3-A output
Dataset Focus	Mixed chapters	Questions & answers	Full chapters	Chapters & Summaries	Unsupervised coherence	Supervised coherence
Max Sequence Length	2900	3200	3200	3200	2900	3100
Model Freezing Strategy						
Freeze Strategy	Partial freeze	Partial freeze	Partial freeze	Partial freeze	Selective unfreeze	Selective unfreeze
Frozen Layers	22 of 32	20 of 32	16 of 32	20 of 32	All except 8-14, 20-25	All except 8-14, 20-25
Training Hyperparameters						
Epochs	7	5	4	3	1	1
Learning Rate	2.5×10^{-5}	2.2×10^{-5}	1.2×10^{-5}	1.0×10^{-5}	5.0×10^{-6}	5.0×10^{-6}
Gradient Acc. Steps	6	8	12	12	8	8
Effective Batch Size	6	8	12	12	8	8
Warmup Ratio	0.06	0.06	0.08	0.08	0.15	0.15
LR Scheduler	Linear	Linear	Cosine w/restarts	Cosine w/restarts	Constant w/warmup	Constant w/warmup
Weight Decay	0.01	0.01	0.01	0.02	0.005	0.005
LoRA Configuration						
Rank (r)	32	32	32	32	8	8
Alpha	64	64	72	72	16	16
Dropout	0.10	0.10	0.08	0.12	0.05	0.05
Target Modules	All (10)	All (10)	All (10)	All (10)	All (10)	All (10)

D.2.3.2 Training Metrics and Loss Curves

Table D.11: Training duration, throughput, and learning rate metrics for each model variant (V1–V3) and stage (A, B) in Phase 3 fine-tuning.

Metric	V1-A	V1-B	V2-A	V2-B	V3-A	V3-B
Model	Unsupervised	Supervised	Supervised	Supervised	Merged Model	Supervised
Training Duration	45.93 hours	46.62 hours	27.26 hours	18.75 hours	1.27 hours	1.86 hours
Samples per Second	0.397	0.341	0.348	0.347	0.359	0.314
Steps per Second	0.066	0.043	0.029	0.029	0.045	0.039
Initial Learning Rate	2.5×10^{-6}	2.2×10^{-6}	1.2×10^{-6}	1.0×10^{-6}	5.0×10^{-7}	5.0×10^{-7}
Peak Learning Rate	2.5×10^{-5}	2.2×10^{-5}	1.2×10^{-5}	1.0×10^{-5}	5.0×10^{-6}	5.0×10^{-6}
Final Learning Rate	1.79×10^{-5}	1.33×10^{-5}	3.94×10^{-10}	7.21×10^{-10}	5.0×10^{-6}	5.0×10^{-6}

Table D.12: Evaluation metrics during fine-tuning Phase 3, showing loss values at key checkpoints for each model variant (V1–V3) and stage (A, B).

Checkpoint	V1				V2				V3			
	Stage A		Stage B		Stage A		Stage B		Stage A		Stage B	
	Epoch	**Loss**	**Epoch**	**Loss**	**Epoch**	**Loss**	**Epoch**	**Loss**	**Epoch**	**Loss**	**Epoch**	**Loss**
Pre-Training	0.00	1.8649	0.00	1.4673	0.00	1.7858	0.00	0.5788	0.00	0.7485	0.00	0.7260
Validation 1	0.32	0.8565	0.21	0.2565	0.25	0.6997	0.27	0.4437	0.24	0.7262	0.19	0.6742
Validation 2	0.64	0.7925	0.42	0.2435	0.49	0.6105	0.54	0.4156	0.49	0.7032	0.38	0.5361
Validation 3	0.96	0.7592	0.63	0.2347	0.74	0.5692	0.81	0.3956	0.73	0.6862	0.57	0.4806
Validation 4	1.28	0.7458	0.84	0.2288	0.98	0.5407	1.08	0.3771	0.97	0.6761	0.76	0.4606
Validation 5	1.60	0.7252	1.05	0.2246	1.23	0.5194	1.35	0.3606	–	–	0.95	0.4498
Validation 6	1.92	0.7089	1.26	0.2200	1.48	0.5006	1.62	0.3473	–	–	–	–
Validation 7	2.24	0.7039	1.47	0.2151	1.72	0.4838	1.88	0.3368	–	–	–	–
Validation 8	2.56	0.6929	1.68	0.2104	1.97	0.4692	2.15	0.3304	–	–	–	–
Validation 9	2.88	0.6815	1.89	0.2058	2.22	0.4578	2.42	0.3258	–	–	–	–
Validation 10	3.20	0.6777	2.10	0.2002	2.46	0.4475	2.69	0.3236	–	–	–	–
Validation 11	3.52	0.6663	2.31	0.1963	2.71	0.4389	2.96	0.3232	–	–	–	–
Validation 12	3.84	0.6520	2.52	0.1941	2.95	0.4325	–	–	–	–	–	–
Validation 13	4.16	0.6585	2.73	0.1891	3.20	0.4286	–	–	–	–	–	–
Validation 14	4.48	0.6510	2.94	0.1894	3.45	0.4257	–	–	–	–	–	–
Validation 15	4.80	0.6443	3.15	0.1869	3.69	0.4246	–	–	–	–	–	–
Validation 16	5.12	0.6540	3.36	0.1844	3.94	0.4244	–	–	–	–	–	–
Validation 17	5.44	0.6499	3.56	0.1821	–	–	–	–	–	–	–	–
Validation 18	5.76	0.6426	3.77	0.1799	–	–	–	–	–	–	–	–
Validation 19	6.08	0.6477	3.98	0.1777	–	–	–	–	–	–	–	–
Validation 20	6.40	0.6454	4.19	0.1785	–	–	–	–	–	–	–	–
Validation 21	6.71	0.6444	4.40	0.1792	–	–	–	–	–	–	–	–
Validation 22	–	–	4.61	0.1785	–	–	–	–	–	–	–	–
Validation 23	–	–	4.82	0.1775	–	–	–	–	–	–	–	–
Final Test	7.00	0.6325	5.00	0.1916	4.00	0.4307	3.00	0.3275	1.00	0.6937	1.00	0.4494

Note: V1 stages represent the first model version with unsupervised (Stage A) and supervised (Stage B) training. V2 stages show specialized format training. V3 represents coherence fine-tuning with unsupervised (Stage A) and supervised (Stage B) training.

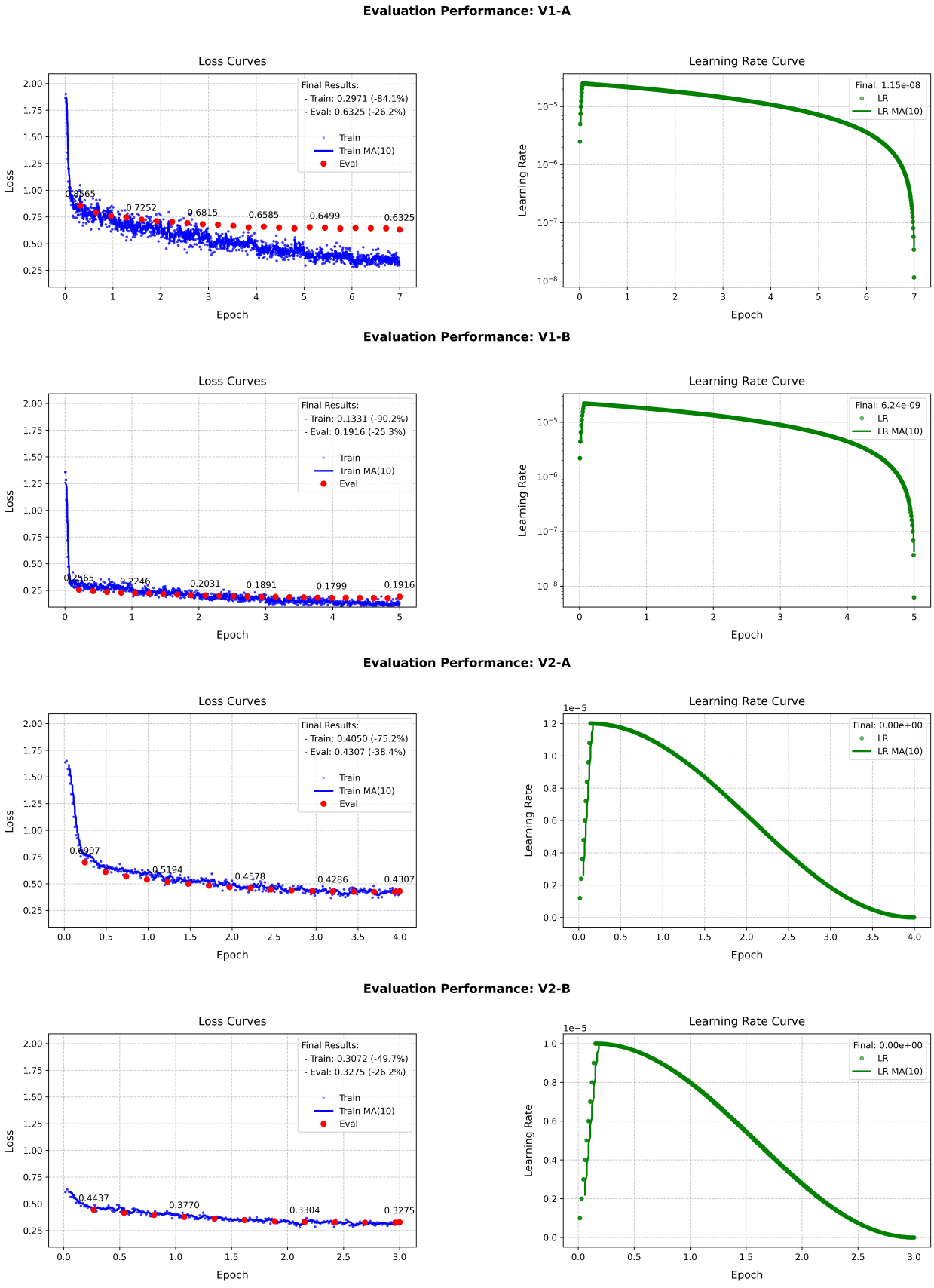


Figure D.4: Training dynamics during Phase 3 fine-tuning: evolution of evaluation loss and learning rate across training epochs for model variants V1 and V2.

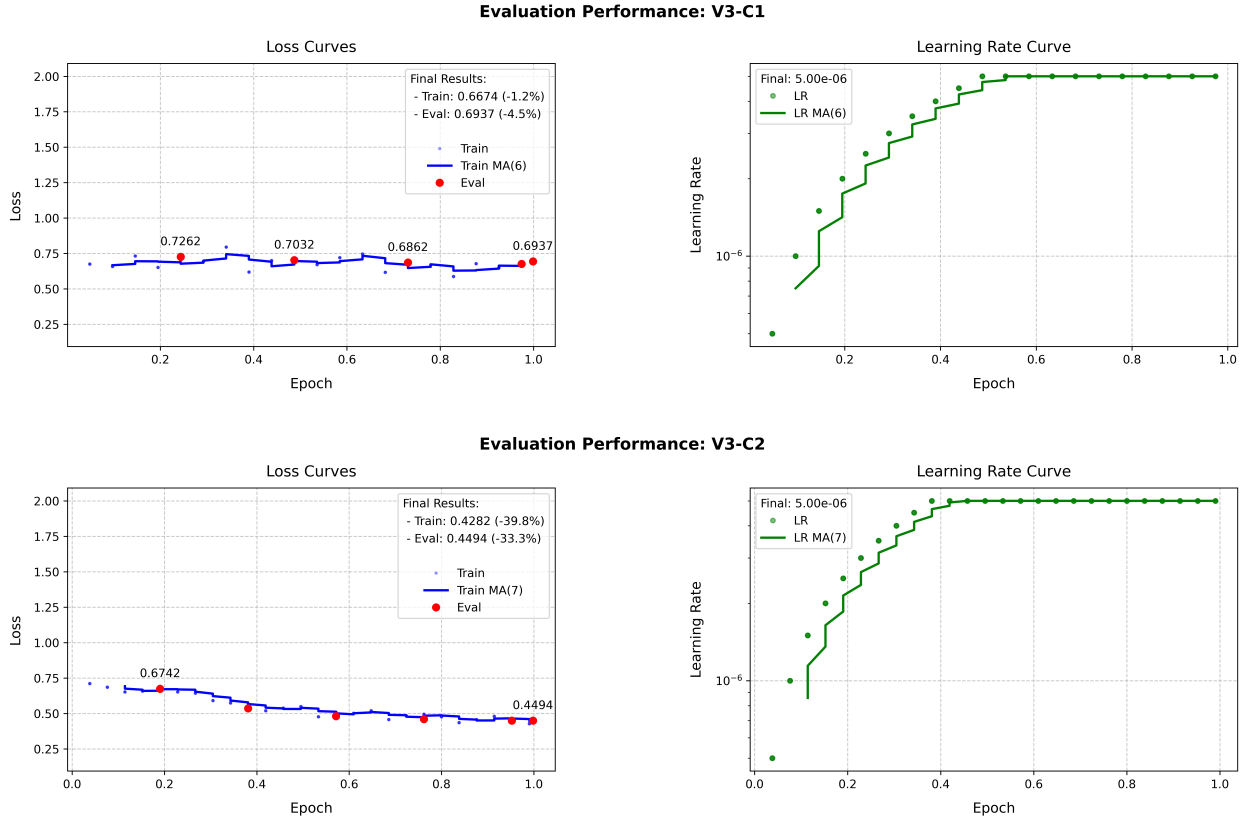


Figure D.5: Training dynamics during Phase 3 fine-tuning: Evolution of evaluation loss and learning rate across training epochs for model variant V3.

D.2.3.3 Technical Analysis and Challenges of Fine-Tuning Phase 3

Phase 3 fundamentally departed from previous approaches by implementing parallel training paths rather than sequential stages, developing three distinct model variants as detailed in Table D.10. V1 focused on domain knowledge acquisition, V2 specialized in document formatting and structure, and V3 emerged from SLERP-based model merging followed by coherence fine-tuning. This approach incorporated sophisticated parameter configurations including partial layer freezing strategies and specialized learning rate scheduling tailored to each variant's objectives.

The V1 variant demonstrated effective knowledge-centric fine-tuning, achieving 66.1 % loss reduction from 1.865 to 0.633 across seven epochs of unsupervised training. As shown in Table D.11, this variant employed an aggressive linear learning rate schedule at 2.5×10^{-5} that maintained consistent learning throughout extended training, optimizing for domain knowledge retention rather than rapid convergence. In contrast, V2 achieved 75.9 % loss reduction using a more conservative learning rate of 1.2×10^{-5} with cosine scheduling and restarts, which stabilized performance during later epochs as evidenced in Figures D.4 and D.5.

Validation metrics in Table D.12 reveal distinct learning patterns across variants. V1-B achieved the most dramatic loss reduction at 86.9 % during supervised Q&A fine-tuning, indicating that question-answering represents a more constrained optimization task than general content

generation. The V3 variant showed more modest loss reduction during coherence fine-tuning at 38.1 % in stage B, yet demonstrated superior qualitative performance in testing. This discrepancy illustrates how standard loss metrics may inadequately capture improvements in coherence and structural consistency.

The partial unfreezing strategy for V3's coherence fine-tuning targeted specific layer ranges, unfreezing layers 8 through 14 and 20 through 25, based on transformer architecture principles where middle layers handle contextual relationships and higher layers process document-level structure [Jia+25, pp. 2f, 6f]. This selective approach created a frozen bridge comprising layers 15 through 19 that preserved core capabilities from both parent models while allowing targeted adaptation of coherence-specific parameters.

Testing revealed improved performance over previous phases, with V3 consistently generating appropriate chapter length and structure. However, critical limitations emerged that impact practical deployment. The V3 model exhibited increased hallucination tendencies when using temperatures below 0.2, introducing details absent from the provided context. This likely stems from the increased number of unfrozen layers during coherence fine-tuning, which enhanced generative capabilities at the cost of factual precision, or potentially from the model merging strategy itself.

The specialized V1-B variant demonstrated format-dependent performance when tested across diverse question types. While excelling with textual chapter content queries, it showed diminished capability when processing tabular MDB data. This format-specific degradation likely results from the specialized training corpus that focused predominantly on textual chapter content rather than diverse data representations [Che+23, pp. 1ff]. The performance pattern suggests that cross-format generalization remains challenging for specialized fine-tuned models, particularly when inference and fine-tuning data distributions differ significantly in structure.

These findings highlight the dual nature of specialized training approaches. While parallel development of distinct models followed by merging created more robust chapter generation capabilities, it introduced challenges in maintaining strict factual precision. Similarly, specialized Q&A fine-tuning dramatically improved performance on similar-format tasks but potentially reduced generalization across different data formats. Future optimization efforts could address these limitations by incorporating explicit anti-hallucination training objectives for generation models and implementing format-diverse Q&A training data to enhance cross-format generalization capabilities.

References

[Aco+22] JN Acosta et al. Multimodal biomedical AI. In: *Nature Medicine* 28(9):(Sept. 2022), 1773–1784. https://doi.org/10.1038/s41591-022-01981-2 (page 103).

[AB25] A Amato and D Branco. SemFedXAI: A Semantic Framework for Explainable Federated Learning in Healthcare. In: *Information* 16(6):(2025). https://www.mdpi.com/2078-2489/16/6/435 (page 6).

[Arr+20] AB Arrieta et al. Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI. In: *Information Fusion* 58:(2020), 82–115 (page 29).

[Baa+03] F Baader et al., eds. *The description logic handbook: theory, implementation, and applications*. USA: Cambridge University Press, 2003 (page 15).

[Bad+22] S Badreddine et al. Logic Tensor Networks. In: *Artificial Intelligence* 303:(2022), 103649. https://www.sciencedirect.com/science/article/pii/S0004370221002009 (page 6).

[BKM09] A Bangor, P Kortum, and J Miller. Determining what individual SUS scores mean: adding an adjective rating scale. In: *J. Usability Studies* 4(3):(May 2009), 114–123 (page 88).

[Bar+22] H Barbosa et al. cvc5: A Versatile and Industrial-Strength SMT Solver. In: *Tools and Algorithms for the Construction and Analysis of Systems*. Ed. by D Fisman and G Rosu. Cham: Springer International Publishing, 2022, pp. 415–442 (page 72).

[Bec+01] K Beck et al. Manifesto for Agile Software Development. 2001. http://www.agilemanifesto.org/ (pages 28, 36, 139).

[Bee22] M Beetz. Knowledge Representation and Reasoning. In: May 2022, pp. 413–432 (page 9).

[Bek+21] M van Bekkum et al. Modular design patterns for hybrid learning and reasoning systems. In: *Applied Intelligence* 51(9):(Sept. 2021), 6528–6546. https://doi.org/10.1007/s10489-021-02394-3 (page 12).

[BS13] D Bender and K Sartipi. HL7 FHIR: An Agile and RESTful approach to healthcare information exchange. In: *Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems*. 2013, pp. 326–331 (page 7).

[Ben+21] EM Bender et al. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. FAccT '21. Virtual Event, Canada: Association for Computing Machinery, 2021, pp. 610–623. https://doi.org/10.1145/3442188.3445922 (page 1).

[BCV13] Y Bengio, A Courville, and P Vincent. Representation Learning: A Review and New Perspectives. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 35(8):(Aug. 2013), 1798–1828. https://doi.org/10.1109/TPAMI.2013.50 (page 10).

[Bes+21] TR Besold et al. Neural-Symbolic Learning and Reasoning: A Survey and Interpretation. In: *Neuro-Symbolic Artificial Intelligence: The State of the Art*. Ed. by P Hitzler and MK Sarker. Vol. 342. Frontiers in Artificial Intelligence and Applications. IOS Press, 2021, pp. 1–51 (pages 11, 13, 139).

[BH97] H Beyer and K Holtzblatt. *Contextual Design: Defining Customer-Centered Systems*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997 (page 139).

[Bha+25] SR Bhat et al. Rethinking Chunk Size For Long-Document Retrieval: A Multi-Dataset Analysis. 2025. arXiv: 2505.21700 [cs.IR] (page 20).

[Bom+22] R Bommasani et al. On the Opportunities and Risks of Foundation Models. 2022. arXiv: 2108.07258 [cs.LG] (page 1).

[BR00] T Boren and J Ramey. Thinking aloud: reconciling theory and practice. In: *IEEE Transactions on Professional Communication* 43(3):(2000), 261–278 (pages 34, 88).

[BHC22] J Bosch, H Holmström Olsson, and I Crnkovic. Chapter 13 Engineering AI Systems. In: *Accelerating Digital Transformation: 10 Years of Software Center*. Ed. by J Bosch et al. Cham: Springer International Publishing, 2022, pp. 407–425. https://doi.org/10.1007/978-3-031-10873-0_18 (page 13).

[Bra83] Brachman. What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks. In: *Computer* 16(10):(1983), 30–36 (page 15).

[Bro95] J Brooke. SUS: A quick and dirty usability scale. In: *Usability Eval. Ind.* 189:(Nov. 1995) (pages 34, 88).

[Bro+20] TB Brown et al. Language models are few-shot learners. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS '20. Vancouver, BC, Canada: Curran Associates Inc., 2020 (page 16).

[BS84] BG Buchanan and EH Shortliffe. Rule Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project (The Addison-Wesley series in artificial intelligence). In: USA: Addison-Wesley Longman Publishing Co., Inc., 1984 (page 9).

[Cao+22] Q Cao et al. KSPMI: A Knowledge-based System for Predictive Maintenance in Industry 4.0. In: *Robotics and Computer-Integrated Manufacturing* 74:(2022), 102281. https://www.sciencedirect.com/science/article/pii/S0736584521001617 (page 7).

[CM24] A Capitanelli and F Mastrogiovanni. A framework for neurosymbolic robot action planning using large language models. In: *Frontiers in Neurorobotics* 18:(2024). https://www.frontiersin.org/journals/neurorobotics/articles/10.3389/fnbot.2024.1342786 (page 7).

[Cav09] A Cavoukian. Privacy by Design: The 7 Foundational Principles. In: *Information and Privacy Commissioner of Ontario, Canada*:(2009). https://www.ipc.on.ca/wp-content/uploads/resources/7foundationalprinciples.pdf (pages 29, 36, 83, 139).

[Cav10] A Cavoukian. Privacy by Design: The 7 Foundational Principles. Revised: Oktober 2010. May 2010 (page 7).

[CT16] F Cesarini and S Thompson. *Designing for Scalability with Erlang/OTP: Implement Robust, Fault-Tolerant Systems*. O'Reilly Media, 2016 (page 36).

[CSM20] B Chan, S Schweter, and T Möller. German's Next Language Model. In: *Proceedings of the 28th International Conference on Computational Linguistics*. Ed. by D Scott, N Bel, and C Zong. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 6788–6796. https://aclanthology.org/2020.coling-main.598/ (page 99).

[Che+25] M Chen et al. EfficientQAT: Efficient Quantization-Aware Training for Large Language Models. 2025. arXiv: 2407.11062 [cs.LG] (page 19).

[Che+23] N Chen et al. Bridge the Gap between Language models and Tabular Understanding. 2023. arXiv: 2302.09302 [cs.CL] (pages 79, 81, 150, 155).

[Chr23] D Chroni. *Symbolic Reasoning Meets Deep Learning: A Hybrid AI Model Approach to Complex Decision-Making and Autonomous Adaptation*. Dec. 2023 (page 13).

[Chu+25] Y Chudasama et al. Toward Interpretable Hybrid AI: Integrating Knowledge Graphs and Symbolic Reasoning in Medicine. In: *IEEE Access* 13:(2025), 39489–39509 (page 6).

[Col+05] R Cole et al. Being Proactive: Where Action Research Meets Design Research. In:(2005) (page 24).

[DM15] E Davis and G Marcus. Commonsense reasoning and commonsense knowledge in artificial intelligence. In: *Commun. ACM* 58(9):(Aug. 2015), 92–103. https://doi.org/10.1145/2701413 (page 56).

[DSS93] R Davis, H Shrobe, and P Szolovits. What Is a Knowledge Representation? In: *AI Magazine* 14(1):(1993), 17–33 (page 15).

[Del+19] D Dellermann et al. Hybrid Intelligence. In: *Business & Information Systems Engineering* 61(5):(Oct. 2019), 637–643. https://doi.org/10.1007/s12599-019-00595-2 (pages 2, 83, 139).

[DP01] G Delzanno and A Podelski. Constraint-based deductive model checking. In: *International Journal on Software Tools for Technology Transfer* 3(3):(Aug. 2001), 250–270. https://doi.org/10.1007/s100090100049 (page 15).

[Dem86] WE Deming. *Out of the Crisis*. MIT Press, 1986 (pages 83, 139).

[Det+22] T Dettmers et al. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In: NIPS '22:(2022) (page 19).

[Deu21] Deutscher Bundestag. Medizinprodukterecht-Durchführungsgesetz (MPDG) vom 28. April 2020. BGBl. I S. 833, das zuletzt durch Artikel 15 des Gesetzes vom 20. Dezember 2023 (BGBl. 2023 I Nr. 408) geändert

worden ist. 2021. https://www.gesetze-im-internet.de/mpdg/ (page 107).

[Dev+19] J Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In:(June 2019). Ed. by J Burstein, C Doran, and T Solorio, 4171–4186. https://aclanthology.org/N19-1423/ (page 11).

[Dmi16] Dmitry Vasiliev. ErlPort. http://erlport.org/. Accessed on June 16, 2025. 2016 (pages 28, 32).

[Dod+20] J Dodge et al. Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping. In:(2020). arXiv: 2002.06305 [cs.CL] (page 18).

[Dre92] HL Dreyfus. *What computers still can't do: a critique of artificial reason*. Cambridge, MA, USA: MIT Press, 1992 (pages 9, 86).

[EW25] K Edemacu and X Wu. Privacy Preserving Prompt Engineering: A Survey. In: *ACM Comput. Surv.* 57(10):(May 2025). https://doi.org/10.1145/3729219 (page 16).

[Efe24] F Efendi. *Patterns in Elixir*. Accessed on May 25, 2025. Nov. 25, 2024. https://softwarepatternslexicon.com/patterns-elixir/8/1/?utm_source=chatgpt.com (page 28).

[EHE25] O Erdem, K Hassett, and F Egriboyun. Hallucination in AI-generated financial literature reviews: evaluating bibliographic accuracy. In: *International Journal of Data Science and Analytics*:(Feb. 2025). http://dx.doi.org/10.1007/s41060-025-00731-0 (page 2).

[Eur16] European Parliament and Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance). Consolidated version. 2016. http://data.europa.eu/eli/reg/2016/679/2016-05-04 (pages 7, 107).

[Eur17] European Parliament and Council of the European Union. Regulation (EU) 2017/745 of the European Parliament and of the Council of 5 April 2017 on medical devices, amending Directive 2001/83/EC, Regulation (EC) No 178/2002 and Regulation (EC) No 1223/2009 and repealing Council Directives 90/385/EEC and 93/42/EEC (Text with EEA relevance). Consolidated version. 2017. http://data.europa.eu/eli/reg/2017/745/2017-05-05 (page 107).

[Eur24] European Parliament and Council of the European Union. Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence and amending Regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and Directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act). PE/24/2024/REV/1. July 2024. http://data.europa.eu/eli/reg/2024/1689/oj (page 104).

[FK03] O Farber and R Kadmon. Assessment of alternative approaches for bioclimatic modeling with special emphasis on the Mahalanobis distance. In: vol. 160. (1). 2003, pp. 115–130. https://www.sciencedirect.com/science/article/pii/S0304380002003277 (page 21).

[FT23] M Faverio and A Tyson. *What the data says about Americans' views of artificial intelligence*. Accessed on October 24, 2024. 2023. https://www.pewresearch.org/short-reads/2023/11/21/what-the-data-says-about-americans-views-of-artificial-intelligence/ (page 1).

[FN71] RE Fikes and NJ Nilsson. Strips: A new approach to the application of theorem proving to problem solving. In: vol. 2. (3). 1971, pp. 189–208. https://www.sciencedirect.com/science/article/pii/0004370271900105 (page 9).

[For82] CL Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. In: *Artificial Intelligence* 19(1):(1982), 17–37. https://www.sciencedirect.com/science/article/pii/0004370282900200 (page 16).

[Gao+24] Y Gao et al. Retrieval-Augmented Generation for Large Language Models: A Survey. In:(2024). arXiv: 2312.10997 [cs.CL] (page 20).

[Gar+19] A Garcez et al. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. In: vol. 6. (4). 2019, pp. 611–631. https://www.collegepublications.co.uk/ifcolog/?00033 (page 2).

[GL23] Ad Garcez and LC Lamb. Neurosymbolic AI: the 3rd wave. In: vol. 56. (11). USA: Kluwer Academic Publishers, Mar. 2023, pp. 12387–12406. https://doi.org/10.1007/s10462-023-10448-w (pages 2, 11, 13, 22).

[GLG08] ASd Garcez, LC Lamb, and DM Gabbay. Neural-Symbolic Cognitive Reasoning. In:(2008) (page 37).

[Gar+24] Á García-Barragán et al. NSSC: a neuro-symbolic AI system for enhancing accuracy of named entity recognition and linking from oncologic clinical notes. In: *Med Biol Eng Comput* 63(3):(Nov. 2024), 749–772 (page 6).

[Gen+23] S Geng et al. Grammar-Constrained Decoding for Structured NLP Tasks without Finetuning. In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by H Bouamor, J Pino, and K Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 10932–10952. https://aclanthology.org/2023.emnlp-main.674/ (page 12).

[Ger23] G Gerganov. llama.cpp: Port of Facebook's LLaMA model in C/C++. https://github.com/ggerganov/llama.cpp. Accessed on June 10, 2025. 2023 (page 104).

[Gil25] AQ Gill. Agile System Development Lifecycle for AI Systems: Decision Architecture. 2025. arXiv: 2501.09434 [cs.SE] (page 6).

[Gio+21] J Giorgi et al. DeCLUTR: Deep Contrastive Learning for Unsupervised Textual Representations. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Ed. by C Zong et al. Online: Association for Computational Linguistics, Aug. 2021, pp. 879–895. https://aclanthology.org/2021.acl-long.72/ (page 21).

[GB85] J Glasgow and R Browse. Programming languages for artificial intelligence. In: *Computers & Mathematics with Applications* 11(5):(1985), 431–448 (page 28).

[God+24] C Goddard et al. Arcee's MergeKit: A Toolkit for Merging Large Language Models. In: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*. Ed. by F Dernoncourt, D Preoţiuc-Pietro, and A Shimorina. Miami, Florida, US: Association for Computational Linguistics, Nov. 2024, pp. 477–485. https://aclanthology.org/2024.emnlp-industry.36/ (pages 19, 81, 131).

[GBC16] I Goodfellow, Y Bengio, and A Courville. *Deep Learning*. Book in preparation for MIT Press. MIT Press, 2016. http://www.deeplearningbook.org (page 144).

[GH14] H Gruber and M Holzer. From Finite Automata to Regular Expressions and Back—A Summary on Descriptional Complexity. In: vol. 151. Open Publishing Association, May 2014, pp. 25–48. http://dx.doi.org/10.4204/EPTCS.151.2 (page 14).

[Gru93] TR Gruber. A translation approach to portable ontology specifications. In: *Knowledge Acquisition* 5(2):(1993), 199–220. https://www.sciencedirect.com/science/article/pii/S1042814383710083 (page 15).

[GOS09] N Guarino, D Oberle, and S Staab. What Is an Ontology? In: May 2009, pp. 1–17 (page 15).

[Gun+19] D Gunning et al. *XAI—Explainable artificial intelligence*. Tech. rep. Dec. 2019, pp. 1–12 (page 139).

[Gur+20] S Gururangan et al. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by D Jurafsky et al. Online: Association for Computational Linguistics, July 2020, pp. 8342–8360. https://aclanthology.org/2020.acl-main.740/ (pages 17, 18).

[Har+07] F van Harmelen et al. *Handbook of Knowledge Representation*. San Diego, CA, USA: Elsevier Science, 2007 (page 56).

[Hau85] J Haugeland. *Artificial intelligence: the very idea*. USA: Massachusetts Institute of Technology, 1985 (pages 1, 8).

[Hay85] F Hayes-Roth. Rule-based systems. In: *Commun. ACM* 28(9):(Sept. 1985), 921–932. https://doi.org/10.1145/4284.4286 (page 15).

[He+23] S He et al. MerA: Merging Pretrained Adapters For Few-Shot Learning. In:(2023). arXiv: 2308.15982 [cs.CL] (page 19).

[HC01] GT Heineman and WT Councill, eds. *Component-based software engineering: putting the pieces together*. USA: Addison-Wesley Longman Publishing Co., Inc., 2001 (pages 35, 139).

[Hem+23] P Hemmer et al. Human-AI Collaboration: The Effect of AI Delegation on Human Task Performance and Task Satisfaction. In: *Proceedings of the 28th International Conference on Intelligent User Interfaces*. ACM, Mar. 2023, pp. 453–463. http://dx.doi.org/10.1145/3581641.3584052 (page 87).

[Her+24] GB Herwanto et al. Toward a Holistic Privacy Requirements Engineering Process: Insights From a Systematic Literature Review. In: *IEEE Access* 12:(2024), 47518–47542 (page 7).

[Hev+04] AR Hevner et al. Design Science in Information Systems Research. In: *MIS Quarterly* 28(1):(2004), 75–105. http://www.jstor.org/stable/25148625 (visited on 05/20/2025) (page 23).

[HBS73] C Hewitt, P Bishop, and R Steiger. A Universal Modular Actor Formalism for Artificial Intelligence. In: *Proceedings of the 3rd International Joint Conference on Artificial Intelligence (IJCAI)*:(1973), 235–245 (page 29).

[Hit+22] P Hitzler et al. Neuro-symbolic approaches in artificial intelligence. In: *National Science Review* 9(6):(Mar. 2022), 1–3 (pages 11, 13).

[Hoo+25] D Hooshyar et al. Towards Responsible and Trustworthy Educational Data Mining: Comparing Symbolic, Sub-Symbolic, and Neural-Symbolic AI Methods. 2025. arXiv: 2504.00615 [cs.AI] (page 13).

[Hou+19] N Houlsby et al. Parameter-Efficient Transfer Learning for NLP. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by K Chaudhuri and R Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 2790–2799. https://proceedings.mlr.press/v97/houlsby19a.html (page 18).

[HR18] J Howard and S Ruder. Universal Language Model Fine-tuning for Text Classification. 2018. arXiv: 1801.06146 [cs.CL] (pages 17, 18).

[Hu+21] EJ Hu et al. LoRA: Low-Rank Adaptation of Large Language Models. 2021. arXiv: 2106.09685 [cs.CL] (page 18).

[Hua+24] B Huang et al. Can Knowledge Editing Really Correct Hallucinations? 2024. arXiv: 2410.16251 [cs.CL] (page 1).

[Hug89] J Hughes. Why Functional Programming Matters. In: *The Computer Journal* 32(2):(Jan. 1989), 98–107 (pages 29, 36).

[Jac99] P Jackson. *Introduction to Expert Systems*. International computer science series. Addison-Wesley, 1999. https://books.google.de/books?id=9rJQAAAAMAAJ (page 16).

[Ji+23] Z Ji et al. Survey of Hallucination in Natural Language Generation. In: *ACM Computing Surveys* 55(12):(Mar. 2023). https://doi.org/10.1145/3571730 (pages 1, 69).

[Jia+25] J Jiang et al. From Compression to Expansion: A Layerwise Analysis of In-Context Learning. 2025. arXiv: 2505.17322 [cs.CL] (pages 81, 155).

[Jor14] PC Jorgensen. *Software Testing: A Craftsman's Approach*. 4th. USA: Auerbach Publications, 2014, p. 494 (pages 28, 36).

[JA25] H Ju and S Aral. Collaborating with AI Agents: Field Experiments on Teamwork, Productivity, and Performance. 2025. arXiv: 2503.18238 [cs.CY] (page 87).

[Kam20] S Kambhampati. Challenges of Human-Aware AI Systems. In: *AI Mag.* 41(3):(Sept. 2020), 3–17. https://doi.org/10.1609/aimag.v41i3.5257 (page 37).

[Kas+21] N Kassner et al. BeliefBank: Adding Memory to a Pre-Trained Language Model for a Systematic Notion of Belief. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Ed. by MF Moens et al. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 8849–8861. https://aclanthology.org/2021.emnlp-main.697/ (pages 21, 22).

[Kau22] H Kautz. The Third Wave of AI: Neurosymbolic AI. In: *Artificial Intelligence Review* 55(3):(2022), 1–15 (page 29).

[Kel03] TD Kelley. Symbolic and Sub-Symbolic Representations in Computational Models of Human Cognition: What Can be Learned from Biology? In: *Theory & Psychology* 13(6):(2003), 847–860 (pages 1, 10).

[Kir+17] J Kirkpatrick et al. Overcoming catastrophic forgetting in neural networks. In: *Proceedings of the National Academy of Sciences* 114(13):(2017), 3521–3526 (pages 79, 150).

[Kle+19] M Kleppmann et al. Local-first software: you own your data, in spite of the cloud. In: *Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*. Onward! 2019. Athens, Greece: Association for Computing Machinery, 2019, pp. 154–178. https://doi.org/10.1145/3359591.3359737 (pages 21, 29, 36).

[KSH12] A Krizhevsky, I Sutskever, and GE Hinton. ImageNet classification with deep convolutional neural networks. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 1097–1105 (page 11).

[Kul+22] T Kulik et al. A Survey of Practical Formal Methods for Security. In: *Form. Asp. Comput.* 34(1):(July 2022). https://doi.org/10.1145/3522582 (page 15).

[KS23] S Kumar and PK Sharma. Machine Learning: Trends, Perspectives, and Prospects. In: *Tuijin Jishu / Journal of Propulsion Technology* 44(6):(2023), 3302–3311. https://doi.org/10.52783/tjjpt.v44.i6.3899 (page 10).

[Lam+21] LC Lamb et al. Graph neural networks meet neural-symbolic computing: a survey and perspective. In: IJCAI'20:(2021) (pages 12, 21).

[Lam01] A van Lamsweerde. Goal-oriented requirements engineering: a guided tour. In: *Proceedings Fifth IEEE International Symposium on Requirements Engineering*. 2001, pp. 249–262 (page 139).

[LE17] SS Laurent and JD Eisenberg. *Introducing Elixir: Getting Started in Functional Programming*. 2nd. O'Reilly Media, Inc., 2017 (page 29).

[Lee+19] C van der Lee et al. Best practices for the human evaluation of automatically generated text. In: *Proceedings of the 12th International Conference on Natural Language Generation*. Ed. by K van Deemter, C Lin, and H Takamura. Tokyo, Japan: Association for Computational Linguistics, Nov. 2019, pp. 355–368. https://aclanthology.org/W19-8643/ (page 87).

[Len95] DB Lenat. CYC: a large-scale investment in knowledge infrastructure. In: *Commun. ACM* 38(11):(Nov. 1995), 33–38. https://doi.org/10.1145/219717.219745 (page 9).

[Leo+18] F Leofante et al. Automated Verification of Neural Networks: Advances, Challenges and Perspectives. In: 2018. arXiv: 1805.09938 [cs.AI] (page 2).

[Leo02] C Leondes. *Expert Systems: The Technology of Knowledge Management and Decision Making for the 21st Century*. Expert Systems: The Technology of Knowledge Management and Decision Making for the 21st Century Vol. 1. Elsevier Science, 2002. https://books.google.de/books?id=4LBQAAAAMAAJ (page 16).

[LJG24] M Levy, A Jacoby, and Y Goldberg. Same Task, More Tokens: the Impact of Input Length on the Reasoning Performance of Large Language Models. In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by LW Ku, A Martins, and V Srikumar. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 15339–15353. https://aclanthology.org/2024.acl-long.818/ (page 87).

[Lew+20] P Lewis et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS '20. Vancouver, BC, Canada: Curran Associates Inc., 2020 (pages 19, 20, 22).

[LHN23] Z Li, J Huang, and M Naik. Scallop: A Language for Neurosymbolic Programming. 2023. arXiv: 2304.04812 [cs.PL] (page 6).

[Lig12] S Ligus. *Effective Monitoring and Alerting*. O'Reilly Media, 2012 (pages 83, 139).

[Lip18] ZC Lipton. The Mythos of Model Interpretability: In machine learning, the concept of interpretability is both important and slippery. In: *Queue* 16(3):(June 2018), 31–57. https://doi.org/10.1145/3236386.3241340 (pages 77, 144).

[Liu+24] NF Liu et al. Lost in the Middle: How Language Models Use Long Contexts. In: *Transactions of the Association for Computational Linguistics* 12:(2024), 157–173. https://aclanthology.org/2024.tacl-1.9/ (page 150).

[Lu+23] Q Lu et al. Responsible AI Pattern Catalogue: A Collection of Best Practices for AI Governance and Engineering. 2023. arXiv: 2209.04963 [cs.AI] (page 6).

[Lug08] GF Luger. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. 6th. USA: Addison-Wesley Publishing Company, 2008 (page 16).

[Mac] J MacFarlane. Pandoc: A Universal Document Converter. https://pandoc.org/. Open-source document converter supporting multiple markup formats. (Visited on 05/21/2025) (page 125).

[MP25] K Manas and A Paschke. Knowledge Integration Strategies in Autonomous Vehicle Prediction and Planning: A Comprehensive Survey. 2025. arXiv: 2502.10477 [cs.AI] (page 14).

[Man+21] R Manhaeve et al. Neural probabilistic logic programming in DeepProbLog. In: *Artificial Intelligence* 298:(2021), 103504. https://www.sciencedirect.com/science/article/pii/S0004370221000552 (page 6).

[Mao+19] J Mao et al. The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision. In: 2019. arXiv: 1904.12584 [cs.CV] (page 103).

[Mar18] G Marcus. Deep Learning: A Critical Appraisal. In:(2018). arXiv: 1801.00631 [cs.AI] (page 10).

[Mar20] G Marcus. The Next Decade in AI: Four Steps Towards Robust Artificial Intelligence.

In:(2020). arXiv: 2002.06177 [cs.AI] (pages 10, 13, 14).

[MD19] G Marcus and E Davis. *Rebooting AI: Building Artificial Intelligence We Can Trust*. USA: Pantheon Books, 2019 (pages 2, 9, 10).

[Mas+24] N Maslej et al. The AI Index 2024 Annual Report. Report. Stanford University, Stanford, CA, April 2024. Licensed under Attribution-NoDerivatives 4.0 International. 2024. https://aiindex.stanford.edu/wp-content/uploads/2024/04/HAI_AI-Index-Report-2024.pdf (page 1).

[May+20] J Maynez et al. On Faithfulness and Factuality in Abstractive Summarization. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by D Jurafsky et al. Online: Association for Computational Linguistics, July 2020, pp. 1906–1919. https://aclanthology.org/2020.acl-main.173/ (page 86).

[McL25] S McLeod. *Comprehensive Guide to LLM Sampling Parameters*. Accessed on June 10, 2025. Apr. 2025. https://smcleod.net/2025/04/comprehensive-guide-to-llm-sampling-parameters/ (page 16).

[McM+17] B McMahan et al. Communication-Efficient Learning of Deep Networks from Decentralized Data. In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Ed. by A Singh and J Zhu. Vol. 54. Proceedings of Machine Learning Research. PMLR, Apr. 2017, pp. 1273–1282. https://proceedings.mlr.press/v54/mcmahan17a.html (page 21).

[MDB] MDB Tools Team. MDB Tools. https://mdbtools.github.io/. Open-source project for accessing Microsoft Access database files on Unix-like systems. (Visited on 05/25/2025) (page 125).

[Mee+23] G Meena et al. A hybrid deep learning approach for detecting sentiment polarities and knowledge graph representation on monkeypox tweets. In: *Decision Analytics Journal* 7:(2023), 100243. https://www.sciencedirect.com/science/article/pii/S2772662223000838 (page 14).

[MMB21] A Meyer-Vitali, W Mulder, and MHT de Boer. Modular Design Patterns for Hybrid Actors. 2021. arXiv: 2109.09331 [cs.AI] (page 13).

[Mic24] Microsoft Research. Microsoft SEAL (Simple Encrypted Arithmetic Library). https://www.microsoft.com/en-us/research/project/microsoft-seal/. Accessed on June 6, 2025. 2024 (page 8).

[Mik+13] T Mikolov et al. Efficient Estimation of Word Representations in Vector Space. In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. 2013. http://arxiv.org/abs/1301.3781 (page 21).

[Mit97] TM Mitchell. *Machine Learning*. 1st ed. USA: McGraw-Hill, Inc., 1997 (page 10).

[Mit21] S Mitra. Regular expressions: A detailed study for the understanding of their role and methods for efficient application. In: *International Journal of Research in Circuits, Devices and Systems* 2(2):(2021), 71–76 (page 14).

[MGS23] Y Moukafih, M Ghogho, and K Smaili. Supervised Contrastive Learning as Multi-Objective Optimization for Fine-Tuning Large Pre-Trained Language Models. In: *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2023, pp. 1–5 (pages 79, 150).

[MB08] L de Moura and N Bjørner. Z3: An Efficient SMT Solver. In: *Tools and Algorithms for the Construction and Analysis of Systems*. Ed. by CR Ramakrishnan and J Rehof. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 337–340 (page 72).

[Mov+22] R Movva et al. Combining Compressions for Multiplicative Size Scaling on Natural Language Tasks. In: *Proceedings of the 29th International Conference on Computational Linguistics*. Ed. by N Calzolari et al. Gyeongju, Republic of Korea: International Committee on Computational Linguistics, Oct. 2022, pp. 2861–2872. https://aclanthology.org/2022.coling-1.252/ (page 19).

[Ma19] MT Mullarkey and ARH and. An elaborated action design research process model. In: *European Journal of Information Systems* 28(1):(2019). Ed. by PÅ and, 6–20. https://doi.org/10.1080/0960085X.2018.1451811 (page 24).

[Nat23] National Institute of Standards and Technology. The NIST Privacy Framework (PF), Version 1.1 IPD. Accessed on June 16, 2025. 2023. https://www.nist.gov/privacy-framework (page 7).

[NS76] A Newell and HA Simon. Computer science as empirical inquiry: symbols and search. In: *Commun. ACM* 19(3):(Mar. 1976), 113–126. https://doi.org/10.1145/360018.360022 (page 9).

[NC10] VM Ngo and TH Cao. Ontology-Based Query Expansion with Latently Related Named Entities for Semantic Text Search. In: *Advances in Intelligent Information and Database Systems*. Ed. by NT Nguyen, R Katarzyniak, and SM Chen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 41–

52. https://doi.org/10.1007/978-3-642-12090-9_4 (page 11).

[ONN24] ONNX Runtime Team. ONNX Runtime: Accelerated Cross-Platform Machine Learning. https://onnxruntime.ai/. Accessed on June 10, 2025. 2024 (page 104).

[OIo1] WJ Orlikowski and CS Iacono. Research Commentary: Desperately Seeking the "IT" in IT Research—A Call to Theorizing the IT Artifact. In: *Information Systems Research* 12(2):(2001), 121–134 (page 25).

[Ouy+22] L Ouyang et al. Training language models to follow instructions with human feedback. 2022. arXiv: 2203.02155 [cs.CL] (pages 17, 18).

[Pac+25] S Packowski et al. Optimizing and Evaluating Enterprise Retrieval-Augmented Generation (RAG): A Content Design Perspective. In: *Proceedings of the 2024 8th International Conference on Advances in Artificial Intelligence*. ICAAI '24. Association for Computing Machinery, 2025, pp. 162–167. https://doi.org/10.1145/3704137.3704181 (page 87).

[PJA24] J Pari, S Jelassi, and P Agrawal. Collective Model Intelligence Requires Compatible Specialization. In:(2024). arXiv: 2411.02207 [cs.LG] (page 19).

[Par72] DL Parnas. On the criteria to be used in decomposing systems into modules. In: *Commun. ACM* 15(12):(Dec. 1972), 1053–1058. https://doi.org/10.1145/361598.361623 (pages 28, 35, 83, 139).

[Pat12] D Patil. Data Jujitsu: The Art of Turning Data into Product. In: *O'Reilly Radar*:(2012), 26. https://www.oreilly.com/library/view/data-jujitsu-the/9781449342692/ (page 139).

[Pau89] LC Paulson. The foundation of a generic theorem prover. In: *Journal of Automated Reasoning* 5(3):(Sept. 1989), 363–397. https://doi.org/10.1007/BF00248324 (page 9).

[Pla25] A Platzer. Intersymbolic AI. In: *Leveraging Applications of Formal Methods, Verification and Validation. Software Engineering Methodologies*. Ed. by T Margaria and B Steffen. Cham: Springer Nature Switzerland, 2025, pp. 162–180 (page 88).

[Pre22] G Press. *What Happened To AI In 2022?* Updated Dec 30, 2022, 10:07 AM EST. 2022. https://www.forbes.com/sites/gilpress/2022/12/30/what-happened-to-ai-in-2022/ (page 1).

[Pur+13] S Purao et al. Ensemble Artifacts: From Viewing to Designing in Action Design Research, Systems, Signs & Actions, An International Journal on Information Technology. In: *Action, Communication and Workpractices* 7:(Jan. 2013), 73–81 (page 24).

[PyT24] PyTorch Opacus Team. Opacus: A Library for Training Differentially Private Models in PyTorch. https://opacus.ai/. Accessed on June 6, 2025. 2024 (page 8).

[Reg09] SC Reghizzi. Finite Automata as Regular Language Recognizers. In: *Formal Languages and Compilation*. London: Springer London, 2009, pp. 1–53. https://doi.org/10.1007/978-1-84882-050-0_3 (page 14).

[RSG16] MT Ribeiro, S Singh, and C Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 1135–1144. https://doi.org/10.1145/2939672.2939778 (pages 77, 145).

[Ric18] C Richardson. *Microservices Patterns*. Manning Publications, 2018 (pages 83, 139).

[Rie+20] N Rieke et al. The future of digital health with federated learning. In: *npj Digital Medicine* 3(1):(Oct. 2020), 119. https://doi.org/10.1038/s41746-020-00323-1 (page 7).

[Rij01] M de Rijke. Handbook of Tableau Methods, Marcello D'Agostino, Dov M. Gabbay, Reiner Hähnle, and Joachim Posegga, eds. In: *Journal of Logic, Language and Information* 10(4):(Dec. 2001), 518–523. https://doi.org/10.1023/A:1017520120752 (page 29).

[Ron+21] S Rongali et al. Continual Domain-Tuning for Pretrained Language Models. 2021. arXiv: 2004.02288 [cs.CL] (page 18).

[Rud19] C Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. In: *Nature Machine Intelligence* 1(5):(May 2019), 206–215. https://doi.org/10.1038/s42256-019-0048-x (page 2).

[Rue+23] L von Rueden et al. Informed Machine Learning – A Taxonomy and Survey of Integrating Prior Knowledge into Learning Systems. In: *IEEE Transactions on Knowledge and Data Engineering* 35(1):(2023), 614–633 (pages 11, 21, 22).

[RHW86] DE Rumelhart, GE Hinton, and RJ Williams. Learning representations by back-propagating errors. In: *nature* 323(6088):(1986), 533–536 (page 10).

[RNo9] S Russell and P Norvig. *Artificial Intelligence: A Modern Approach*. 3rd. USA: Prentice Hall Press, 2009 (pages 9, 16).

[Sah+25] P Sahoo et al. A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications. In:(2025). arXiv: 2402.07927 [cs.AI] (page 17).

[Sar21] IH Sarker. Deep Cybersecurity: A Comprehensive Overview from Neural Network and Deep Learning Perspective. In: *SN Comput. Sci.* 2(3):(Mar. 2021). https://doi.org/10.1007/s42979-021-00535-6 (page 37).

[Sar22] IH Sarker. AI-Based Modeling: Techniques, Applications and Research Issues Towards Automation, Intelligent and Smart Systems. In: *SN Computer Science* 3(2):(Feb. 2022), 158. https://doi.org/10.1007/s42979-022-01043-x (page 37).

[Sar+21] MK Sarker et al. Neuro-symbolic artificial intelligence: Current trends. In: *AI Communications* 34(3):(2021), 197–209. eprint: https://journals.sagepub.com/doi/pdf/10.3233/AIC-210084 (pages 2, 12, 21, 22).

[Sar+24] B Sarmah et al. HybridRAG: Integrating Knowledge Graphs and Vector Retrieval Augmented Generation for Efficient Information Extraction. In: ICAIF '24. Brooklyn, NY, USA: Association for Computing Machinery, 2024, pp. 608–616. https://doi.org/10.1145/3677052.3698671 (page 20).

[SD09] J Sauro and JS Dumas. Comparison of three one-question, post-task usability questionnaires. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '09. Boston, MA, USA: Association for Computing Machinery, 2009, pp. 1599–1608. https://doi.org/10.1145/1518701.1518946 (pages 34, 86).

[Sch+23] T Schick et al. Toolformer: language models can teach themselves to use tools. In: NIPS '23:(2023), 68539–68551 (page 12).

[SWS22] G Schwalbe, C Wirth, and U Schmid. Enabling Verification of Deep Neural Networks in Perception Tasks Using Fuzzy Logic and Concept Embeddings. 2022. arXiv: 2201.00572 [cs.CV] (pages 12, 22).

[Scu+15] D Sculley et al. Hidden technical debt in Machine learning systems. In: *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 2503–2511 (pages 28, 36, 139).

[Sei+11] MK Sein et al. Action Design Research. In: *MIS Quarterly* 35(1):(2011), 37–56. http://www.jstor.org/stable/23043488 (visited on 05/19/2025) (pages 23–26, 28, 30, 33).

[Shi+23] S Shi et al. Effidit: An Assistant for Improving Writing Efficiency. In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*. Ed. by D Bollegala, R Huang, and A Ritter. Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 508–515. https://aclanthology.org/2023.acl-demo.49/ (page 69).

[Shi25] A Shirvanporzour. Artificial Intelligence in Banking Risk Management and Anti-Money Laundering: A Comprehensive Review. In:(Mar. 2025) (page 14).

[Shu+21] K Shuster et al. Retrieval Augmentation Reduces Hallucination in Conversation. In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Ed. by MF Moens et al. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 3784–3803. https://aclanthology.org/2021.findings-emnlp.320/ (page 20).

[Sig] W Sigloch. Expert interviews on medical device assessment processes and requirements. Personal communication. Series of structured interviews and feedback sessions conducted between November 2024 and June 2025, covering workflow requirements, information integrity needs, and system evaluation (pages 3, 28, 106).

[Sil+16] D Silver et al. Mastering the game of Go with deep neural networks and tree search. In: *Nature* 529(7587):(Jan. 2016), 484–489. https://doi.org/10.1038/nature16961 (page 11).

[Sim+18] S Simoes et al. Content and context: two-pronged bootstrapped learning for regex-formatted entity extraction. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI'18/IAAI'18/EAAI'18. New Orleans, Louisiana, USA: AAAI Press, 2018 (page 14).

[Sim96] HA Simon. *The sciences of the artificial (3rd ed.)* Cambridge, MA, USA: MIT Press, 1996 (page 23).

[SUM99] HB Sipma, TE Uribe, and Z Manna. Deductive Model Checking. In: *Formal Methods in System Design* 15(1):(July 1999), 49–74. https://doi.org/10.1023/A:1008791913551 (pages 15, 22).

[Smu95] RM Smullyan. *First-Order Logic*. 1st ed. Dover, 1995, pp. 35–80 (pages 15, 22).

[SWL99] DK Sobek, AC Ward, and JK Liker. Toyota's Principles of Set-Based Concurrent Engineering. In: *MIT Sloan Management Review* 40(2):(1999), 67–83 (page 139).

[Sow92] JF Sowa. Semantic networks. In: *Encyclopedia of Artificial Intelligence*. Ed. by SC Shapiro. John Wiley & Sons, 1992, pp. 1493–1511 (page 15).

[Sze+14] C Szegedy et al. Intriguing properties of neural networks. In:(2014). arXiv: 1312.6199 [cs.CV] (page 10).

[Ten24a] TensorFlow Federated Team. TensorFlow Federated: Machine Learning on Decentralized Data. https://www.tensorflow.org/federated. Accessed on June 6, 2025. 2024 (page 7).

[Ten24b] TensorFlow Privacy Team. TensorFlow Privacy: A Library for Training Machine Learning Models with Privacy for Training Data. https://github.com/tensorflow/privacy. Accessed on June 6, 2025. 2024 (page 7).

[Ten23] TensorFlow Team. TensorFlow Lite: Deploy machine learning models on mobile and edge devices. https://www.tensorflow.org/lite. Accessed on June 10, 2025. 2023 (page 104).

[Tho23] C Thorbecke. *A year after ChatGPT's release, the AI revolution is just beginning*. Updated 10:32 AM EST, Thu November 30, 2023. 2023. https://edition.cnn.com/2023/11/30/tech/chatgpt-openai-revolution-one-year/index.html (page 1).

[Tra+22] M Trabelsi et al. An Effective Hybrid Symbolic Regression–Deep Multilayer Perceptron Technique for PV Power Forecasting. In: *Energies* 15(23):(2022). https://www.mdpi.com/1996-1073/15/23/9008 (page 12).

[US 21] U.S. Food and Drug Administration. *Artificial Intelligence/Machine Learning (AI/ML)-Based Software as a Medical Device (SaMD) Action Plan*. Tech. rep. FDA, Jan. 2021. https://www.fda.gov/media/145022/download (page 104).

[VKL16] O Vasilecas, D Kalibatiene, and D Lavbič. Rule- and context-based dynamic business process modelling and simulation. In: *Journal of Systems and Software* 122:(2016), 1–15. https://www.sciencedirect.com/science/article/pii/S0164121216301509 (page 9).

[Vas+17] A Vaswani et al. Attention is all you need. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010 (pages 11, 16).

[Vid+25] ME Vidal et al. Integrating Knowledge Graphs with Symbolic AI: The Path to Interpretable Hybrid AI Systems in Medicine. In: *Journal of Web Semantics* 84:(2025), 100856. https://www.sciencedirect.com/science/article/pii/S1570826824000428 (page 14).

[Wad92] P Wadler. The essence of functional programming. In: *Proceedings of the 19th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. ACM, 1992, pp. 1–14 (page 36).

[WWE92] JG Walls, GR Widmeyer, and OA El Sawy. Building an Information System Design Theory for Vigilant EIS. In: *Information Systems Research* 3(1):(1992), 36–59 (page 23).

[Wan+24] Z Wan et al. Towards Efficient Neuro-Symbolic AI: From Workload Characterization to Hardware Architecture. In: *IEEE Transactions on Circuits and Systems for Artificial Intelligence* 1(1):(2024), 53–68 (page 13).

[WD20] J Wang and Y Dong. Measurement of Text Similarity: A Survey. In: *Information* 11(9):(2020). https://www.mdpi.com/2078-2489/11/9/421 (page 21).

[WYW25] W Wang, Y Yang, and F Wu. Towards Data-And Knowledge-Driven AI: A Survey on Neuro-Symbolic Computing. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 47(2):(Feb. 2025), 878–899. https://doi.org/10.1109/TPAMI.2024.3483273 (page 13).

[WC24] ZJ Wang and DH Chau. MeMemo: On-device Retrieval Augmentation for Private and Personalized Text Generation. In: *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '24. Washington DC, USA: Association for Computing Machinery, 2024, pp. 2765–2770. https://doi.org/10.1145/3626772.3657662 (page 20).

[Wei+22] J Wei et al. Finetuned Language Models Are Zero-Shot Learners. In: 2022. arXiv: 2109.01652 [cs.CL] (page 17).

[WWA25] T Wijesundara, M Warren, and NAG Arachchilage. SoK: Enhancing Privacy-Preserving Software Development from a Developers' Perspective. 2025. arXiv: 2504.20350 [cs.SE] (page 5).

[WP24] SM Williamson and V Prybutok. Balancing Privacy and Progress: A Review of Privacy Challenges, Systemic Oversight, and Patient Perceptions in AI-Driven Healthcare. In: *Applied Sciences* 14(2):(2024). https://www.mdpi.com/2076-3417/14/2/675 (page 1).

[XKN22] X Xie, K Kersting, and D Neider. Neuro-Symbolic Verification of Deep Neural Networks. 2022. arXiv: 2203.00938 [cs.AI] (page 12).

[XGD25] W Xu, Z Gao, and M Dainoff. An HCAI Methodological Framework (HCAI-MF): Putting It Into Action to Enable Human-Centered AI. 2025. arXiv: 2311.16027 [cs.HC] (page 6).

[Yu+23] D Yu et al. A survey on neural-symbolic learning systems. In: *Neural Networks* 166:(2023), 105–126. https://www.sciencedirect.com/science/article/pii/S0893608023003398 (page 12).

[Zaf+19] O Zafrir et al. Q8BERT: Quantized 8Bit BERT. In: *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-NIPS)*. IEEE, Dec. 2019, pp. 36–39. http://dx.doi.org/10.1109/EMC2-NIPS53020.2019.00016 (page 19).

[Zha+20] T Zhang et al. BERTScore: Evaluating Text Generation with BERT. In: *CoRR*:(2020). arXiv: 1904.09675 [cs.CL] (page 87).

[ZS24] X Zhang and VS Sheng. Bridging the Gap: Representation Spaces in Neuro-Symbolic AI. 2024. arXiv: 2411.04393 [cs.AI] (page 11).