CHALMERS
UNIVERSITY OF TECHNOLOGY

How to Build AI Agents by Augmenting LLMs with Codified Human Expert Domain Knowledge? A Software Engineering Framework

Downloaded from: https://research.chalmers.se, 2026-10-29 00:02 UTC

Citation for the original published paper (version of record):
Uulu, C., Kulyabin, M., Fuhrmann, I. et al (2026). How to Build AI Agents by Augmenting LLMs with Codified Human Expert Domain Knowledge? A
Software Engineering Framework. Proceedings 2026 IEEE ACM 5th International Conference on AI Engineering Software Engineering for AI Cain 2026: 271-281.
http://dx.doi.org/10.1145/3793653.3793776

N.B. When citing this work, cite the original published paper.

© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, or reuse of any copyrighted component of this work in other works.

(article starts on next page)

How to Build AI Agents by Augmenting LLMs with Codified Human Expert Domain Knowledge? A Software Engineering Framework

Choro Ulan uulu*
Siemens AG, Eindhoven University of Technology
München, Germany
choro.ulan-uulu@siemens.com

Mikhail Kulyabin
Siemens AG
München, Germany

Iris Fuhrmann
Siemens AG
München, Germany

Jan Joosten
Siemens AG
München, Germany

Nuno Miguel Martins Pacheco
Siemens AG
München, Germany

Filippos Petridis
Siemens AG
München, Germany

Rebecca Johnson
Siemens AG
München, Germany

Jan Bosch
Chalmers University of Technology, Eindhoven University of Technology
Göteborg, Sweden
j.bosch1@tue.nl

Helena Holmström Olsson
Malmö University
Malmo, Sweden
helena.holmstrom.olsson@mau.se

Abstract

Critical domain knowledge typically resides with few experts, creating organizational bottlenecks in scalability and decision-making. Non-experts struggle to create effective visualizations, leading to suboptimal insights and diverting expert time. This paper investigates how to capture and embed human domain knowledge into AI agent systems through an industrial case study. We propose a software engineering framework to capture human domain knowledge for engineering AI agents in simulation data visualization by augmenting a Large Language Model (LLM) with a request classifier, Retrieval-Augmented Generation (RAG) system for code generation, codified expert rules, and visualization design principles unified in an agent demonstrating autonomous, reactive, proactive, and social behavior. Evaluation across five scenarios spanning multiple engineering domains with 12 evaluators demonstrates 206% improvement in output quality, with our agent achieving expert-level ratings in all cases versus baseline's poor performance, while maintaining superior code quality with lower variance. Our contributions are: an automated agent-based system for visualization generation and a validated framework for systematically capturing human domain knowledge and codifying tacit expert knowledge into AI agents, demonstrating that non-experts can achieve expert-level outcomes in specialized domains.

ACM Reference Format:
Choro Ulan uulu, Mikhail Kulyabin, Iris Fuhrmann, Jan Joosten, Nuno Miguel Martins Pacheco, Filippos Petridis, Rebecca Johnson, Jan Bosch, and Helena Holmström Olsson. 2026. How to Build AI Agents by Augmenting LLMs with Codified Human Expert Domain Knowledge? A Software Engineering Framework. In *2026 IEEE/ACM 5th International Conference on AI Engineering - Software Engineering for AI (CAIN '26), April 12–13, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3793653.3793776

*Corresponding author

This work is licensed under a Creative Commons Attribution 4.0 International License.
CAIN '26, Rio de Janeiro, Brazil
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2475-6/26/04
https://doi.org/10.1145/3793653.3793776

1 Introduction

Organizations across industries face a critical scalability challenge: essential domain knowledge often resides with few experts, creating bottlenecks that limit productivity and decision-making quality [19]. When experts are unavailable, work either halts or proceeds with suboptimal outcomes, potentially leading to missed deadlines, increased costs, and catastrophic failures [19].

This challenge is particularly acute in data visualization, where creating effective charts requires both domain knowledge and visualization expertise. Non-experts typically default to familiar chart types because selecting appropriate techniques for complex data remains difficult [11]. Even when attempting sophisticated visualizations, results frequently require expert interpretation [5], while experts must balance mentorship against their primary responsibilities [21].

In simulation data visualization, these challenges intensify. Engineers need dual expertise in simulation analysis and data analytics to create visualizations revealing decision-critical insights. Without this expertise, users significantly underutilize available capabilities, missing opportunities to expose key trade-offs [5, 11]. Critical visualization design knowledge—such as which plot types reveal specific patterns—remains tacit within domain experts, necessitating continuous validation cycles that divert expert resources from high-value tasks.

We illustrate this through Simulation Analysis software, a design space exploration platform that optimizes parameters (e.g., minimizing weight while maximizing strength). While Simulation Analysis

software includes sophisticated post-result analysis capabilities that enable users to visualize complex data sets, this presents an opportunity to enhance user experience through automated guidance.

While it includes sophisticated visualization capabilities, users require multiple attempts to identify effective visualization types [9]. This trial-and-error approach is time-consuming and discourages full exploration of features that could accelerate design decisions.

Simulation Analysis software serves as an ideal test case for our framework because its extensive visualization capabilities and comprehensive post-processing features are representative of sophisticated engineering software where automated expert guidance can significantly enhance user productivity.

This paper addresses the research question (RQ): How can domain knowledge from human experts be captured, codified, and leveraged to construct Large Language Model (LLM) - based AI agents capable of autonomous expert-level performance?

Our results demonstrate how human expert domain knowledge can be captured to construct LLM-based AI agents to reduce expert bottlenecks. The resulting AI agent enables non-experts to generate expert-level visualizations that match expert-level quality in technical accuracy, visual clarity, and analytical insight, without requiring constant expert involvement.

The contributions of this paper are: (1) A systematic software engineering framework for capturing human domain knowledge and engineering an AI agent through complementary strategies: request classifier, RAG for domain-specific code generation, codified expert rules, and visualization design principles, implemented as a reference architecture demonstrating integration of heterogeneous AI techniques with clear separation of concerns.

(2) Empirical evidence from industrial evaluation with 12 evaluators across five scenarios spanning multiple engineering domains (electrochemical, electromagnetic, mechanical systems) demonstrating 206% improvement in output quality (mean: 2.60 vs. 0.85 on 0-3 scale), with our system achieving expert-level ratings (Mode=3) consistently versus baseline's poor performance (Mode=0 in 4/5 scenarios), and superior code quality with lower variance (SD: 0.29-0.58 vs. 0.39-1.11).

(3) Demonstration that the framework addresses organizational expert bottlenecks by enabling non-experts to generate expert-level visualizations through simple prompts, effectively democratizing domain knowledge and allowing domain experts to focus on specialized tasks requiring unique expertise.

Section 2 provides background and reviews related work, Section 3 describes our research methodology, Section 4 presents expert interview findings, Section 5 presents our solution, Section 6 validates effectiveness, and Section 7 concludes, and Section 8 discusses limitations.

2 Background and Related work

Poor visualizations cause fundamental misinterpretation of critical data [8]. Medical-risk visualizations can lead patients to fundamentally misunderstand the base rates or risk factors for diseases or medical procedures [3, 8]. Suboptimal visualizations in minimally invasive surgery have contributed to patient injury rates exceeding 50% [2]. Misleading visualizations can lead to spread of misinformation [4, 16]. In the context of simulation software, according to [14], misused visualization types are difficult to interpret and may lead to erroneous decision making.

The challenges in creating effective visualizations are established. Non-experts struggle with selecting appropriate visualization techniques for complex data [11], often defaulting to familiar but suboptimal chart types. Even when more sophisticated visualizations are attempted, they frequently require expert interpretation [5], creating dependencies on scarce expert resources [21].

Knowledge Codification and Rule-Based Systems The integration of structured knowledge into LLM systems represents an active research area. [25] conducted preliminary research on how LLMs can learn from rules. [24] developed a rule generation framework, creating 8000 primitive rules and 6000 compositional rules across five domains. [23] applied rule-based results to an LLM to generate insights. However, earlier work suggested limitations in LLMs' ability to follow rules [18], though newer LLM versions have demonstrated significantly improved rule-following capabilities. More recently, [28] showed that LLMs with rules perform 30% better than LLMs without rules, providing empirical evidence for the value of rule integration.

LLM based scientific visualization We define an AI agent following [7] as a system exhibiting four key properties: autonomy (operates independently after prompting), reactivity (responds to user requests), proactivity (applies expert rules), and social ability (natural language interaction). These characteristics manifest through classifier-based routing, autonomous rule application, and code generation.

Recent advances in Large Language Models have opened new possibilities for automated code generation and visualization creation. [22] evaluated whether LLMs are ready for generating code that creates visualizations, showing that a high number of charts were properly built and that LLMs are a promising approach to generate charts. [6] demonstrated that modern multimodal LLMs can surpass human performance in visualization literacy tasks when given the proper analytical framework. Several approaches have emerged for LLM-based visualization generation. [26] created a method for LLM Agents-based visualization of data, generating code for the Python library matplotlib and incorporating a feedback loop to validate results. [10] employed a multi-agent system to generate plots. [17] used example code snippets specific for ParaView's PvPython API to help the LLM generate scripts for 2D visualization creation. The application of AI in engineering software contexts continues to evolve [20].

3 Research Method and Case Context

Case Company Context This research was conducted at Siemens, focusing on their Simulation Analysis software. The first author's position as a Siemens employee provided privileged access to domain experts, proprietary data, and technical documentation, enabling deep understanding of real-world visualization challenges and facilitating the iterative framework development process.

3.1 Overall Research Process

Case study methodology was employed to investigate knowledge codification through a systematic four-step research process. The

research followed a sequential approach designed to extract, codify, and validate expert knowledge integration into LLMs.

3.1.1 Step 1: Expert Knowledge Extraction. We conducted semi-structured interviews with two domain experts to extract specialized knowledge: a simulation analysis software expert and a data visualization expert. The interviews aimed to understand current challenges, identify knowledge gaps, and extract actionable rules for improving visualization creation in Simulation Analysis software.

Interview Design and Conduct The interview protocol was structured around three main themes: (1) current visualization workflows and pain points, (2) expert decision-making processes when creating visualizations, and (3) specific rules and heuristics used in practice. Each interview lasted approximately 60-90 minutes and was conducted individually to avoid groupthink effects. The interviews were screen-recorded with participant consent and subsequently transcribed for analysis.

The interview guide included open-ended questions such as "Can you walk me through your typical process when creating a visualization for simulation data?" and "What are the most used and useful plots?" followed by probing questions to elicit specific rules and decision criteria. We used a combination of direct questioning and scenario-based discussions where experts were presented with example visualization tasks to reveal their tacit knowledge.

Rule Extraction and Implementation The experts directly provided explicit rules and guidelines during the interviews, which were captured and documented. Rather than requiring interpretive analysis, the experts articulated clear, actionable rules that could be directly implemented. These expert-provided rules were then systematically codified into Python functions and integrated into our system architecture (Section 5). This direct rule provision approach ensured high fidelity between expert knowledge and system implementation, forming the foundation for the framework development in Step 2.

Using two experts is methodologically appropriate for this study because: (i) we require comprehensive coverage of two distinct, specialized knowledge domains that rarely overlap in single individuals, (ii) our goal is systematic knowledge extraction rather than statistical generalization, and (iii) validation relies on objective technical performance metrics rather than expert consensus. Additionally, the knowledge extraction process was significantly enhanced by the first author's position as an employee within the case company, providing excellent access to internal data, company internal discussions, and a variety of proprietary data sources that complemented the formal expert interviews. This approach aligns with established knowledge engineering practices where small expert groups (1-3 experts) are sufficient to construct a coherent knowledge base [12].

3.1.2 Step 2: Framework Development. Following knowledge extraction, we analyzed the interview findings to derive a systematic framework for codifying expert knowledge into AI systems. This step involved identifying common patterns across expert insights, structuring the knowledge into implementable components, and designing a generalized framework that could be applied beyond the specific case study domain.

3.1.3 Step 3: System Implementation. We developed an automated system that generates Python code for visualization creation, incorporating the extracted expert knowledge through multiple augmentation mechanisms. The implementation integrated a RAG system for domain-specific code generation, codified expert rules as executable components, enhanced system prompts with visualization guidelines, and unified these elements through an intelligent classifier within a comprehensive architecture.

3.1.4 Step 4: Comprehensive Validation. To validate our system, we designed a comprehensive evaluation methodology that assessed both the quality of generated visualizations and the underlying code quality across engineering domains. The validation process involved two key evaluation components:

Quality of Final Visualization Assessment We recruited a non-expert user from within the case company—a mechanical engineer with 1 year of experience in simulation domain but limited visualization expertise—to test the system's practical effectiveness. The participant was introduced to the system through a 30-minute orientation session covering basic functionality without visualization training. The non-expert user generated visualizations using simple prompts, which were then assessed by a visualization expert—a data visualization specialist with 20 years of experience in simulation data analysis, currently working at the case company. The expert evaluated both analytical insight and visual effectiveness based on their professional expertise. This evaluation component assessed whether our system could enable non-experts to generate expert-level visualizations.

Quality of Generated Code Assessment We conducted a technical evaluation of the underlying Python code quality, comparing our proposed system (LLM based AI agent) against a baseline approach (LLM with RAG only) across multiple representative scenarios. We employed a triangulated assessment approach involving two domain experts—a software engineering expert and a simulation optimization expert—supplemented by an AI assessor (Claude 4.5 Sonnet) to enhance evaluation robustness and reduce potential bias. The evaluation focused on three critical dimensions: code validity, code correctness, and output quality.

4 Expert Knowledge Extraction and Analysis

We conducted semi-structured interviews with two distinct domain experts to extract specialized knowledge from different areas of expertise. The first interview focused on simulation analysis domain knowledge, while the second concentrated on visualization design principles.

4.1 Simulation Analysis Software Expert Insights

Our interview with the simulation analysis software domain expert revealed four primary challenges in simulation data visualization: (1) Expert bottleneck: Significant expert time is required to guide users through effective visualization creation, creating scalability constraints. (2) Advanced feature adoption: Maximizing the value of sophisticated visualization capabilities requires domain knowledge that can be systematically captured and transferred to users. (3)

Iterative complexity: Creating decision-enabling visualizations requires extensive refinement to identify critical insights and effective visual encodings.

4.1.1 Simulation Analysis software. **Domain expert rules for plot generation** During the interview, a set of rules was obtained that guide users through the post-processing workflow in Simulation Analysis software. An example of a rule is convergence of objectives. The study is not finished if the objective is not converged.

A key insight from the expert was that Simulation Analysis software operates as a physics-agnostic platform, meaning it can be applied across diverse engineering domains without being tied to specific physical phenomena or simulation types.

The expert emphasized that the rules should mirror this physics-agnostic nature of Simulation Analysis software. Consequently, these guidelines were deliberately formulated to be domain-independent, because the domain expert works with users that are for example working with: (i) Computational Fluid Dynamics (CFD) simulations. (ii) Pressure drop analyses. (iii) Stress analysis studies.

This universal applicability is crucial because it allows the same post-processing methodology to be consistently applied across different engineering disciplines, reducing the learning curve for users who work in multiple domains.

Example rule: Convergence Assessment Any analysis begins with a fundamental question: Have the objectives converged? This initial step is critical because: it validates the reliability of the optimization results, it determines whether additional iterations are needed and it provides confidence in subsequent analysis steps.

To answer this question, users are guided to create a history plot that visualizes the objective function values over the course of the optimization. This graphical representation allows users to clearly observe whether the objectives have reached a stable, converged state or if the optimization is still actively searching for better solutions.

4.2 Visualization Expert Insights

In a separate interview, we consulted with a visualization design expert to extract principles for creating effective and readable visualizations. This expert provided complementary knowledge focused on visual design and communication effectiveness.

4.2.1 History Plot Requirements. The visualization expert provided specific guidelines for history plots in simulation contexts. History plots should always display the best design for reference and limit displays to no more than 2 variables, objectives, or responses to maintain readability. Visual encoding should communicate convergence status through dashed lines for non-converged variables and solid lines for converged variables. Related objectives should be grouped on the same plot for comparison, establishing clear visual hierarchy with primary and secondary objectives. Both objectives must show their individual best design history while prioritizing readability over decorative styling.

4.2.2 General Visualization Principles: The expert emphasized that effective simulation visualizations must: clearly communicate convergence status through visual encoding, maintain visual clarity and avoid unnecessary styling that reduces readability, provide sufficient context for interpretation without expert guidance and follow consistent styling patterns across different plot types.

These insights revealed that technical correctness alone was insufficient - visualizations needed to follow established design principles to effectively communicate insights to non-expert users. Additionally, according to [15] beautiful visualizations indicate increased trust.

4.3 Summary of Key Findings

Based on the combined insights from both expert interviews and the first authors position as an employee within the case company, we derived a framework described in Section 4.4, which is validated in this paper by an implementation of the solution to the challenges.

4.4 Software Engineering Framework

4.4.1 Framework Purpose and Derivation. The expert bottleneck in data visualization represents a broader organizational challenge where critical domain knowledge remains concentrated among few specialists. Our framework systematically codifies tacit expert knowledge into AI systems, enabling automated replication of expert-level decision-making. The framework addresses three core challenges: (1) capturing diverse forms of expert knowledge (explicit rules vs. tacit guidelines), (2) implementing this knowledge in forms that AI systems can effectively utilize, and (3) ensuring the resulting systems maintain expert-level quality without requiring expert oversight.

This framework emerged directly from analyzing expert interview transcripts (Section 4.1). A key pattern emerged: experts described two fundamentally different types of knowledge:

Explicit procedural rules As described in Section 4.1, the simulation expert emphasized convergence checking as a prerequisite for analysis. This type of knowledge—clear if-then logic—could be directly translated into executable code (Python functions that check convergence and generate reports).

Tacit design principles The visualization expert's guidelines on visual encoding (e.g., using dashed lines for non-converged variables, Section 4.1) represent contextual, judgment-based knowledge that couldn't be reduced to simple rules. This required integration into the AI's reasoning process through prompt engineering or retrieval mechanisms.

This insight led to our two-pronged implementation strategy: (1) executable code for algorithmic rules, and (2) LLM augmentation for contextual principles. Initial testing revealed neither approach alone was sufficient—code lacked analytical insight while LLM-only solutions produced domain-inappropriate outputs. This drove the integrated approach where both strategies work complementarily.

4.4.2 Framework Overview. We developed a four-step framework (Fig. 1) to codify expert knowledge: (1) conduct expert interviews, (2) implement rules through complementary approaches—writing Python scripts for algorithmic rules and embedding guidelines directly into LLM system prompts, (3) test results, and (4) iterate based on findings.

This validated framework (Section 5) provides a systematic approach for transferring expert knowledge into an AI agent.

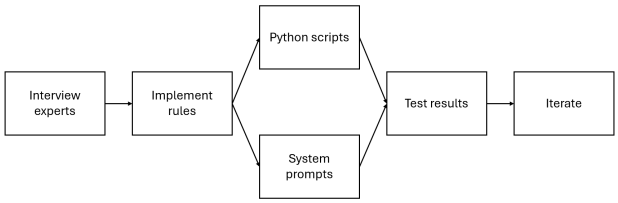

Figure 1: Software Engineering Framework for Knowledge Codification

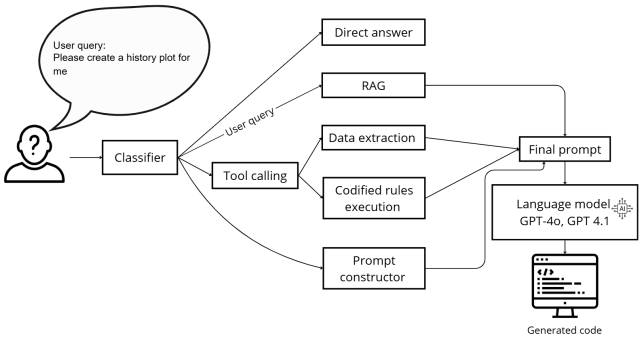

Figure 2: AI agent architecture. User query enters the agent's perception layer (classifier), which autonomously routes to appropriate processing components.

5 Solution

5.1 Framework Validation through Implementation

To validate our framework, we developed a comprehensive AI Agent (Fig. 2) that automates expert-level visualization generation by integrating a classifier using codified expert rules, prompt constructor (for visualization guidelines and code generation) from domain experts, and a RAG system.

The agent architecture begins with intelligent request classification, where a classifier-based system categorizes user requests and dynamically invokes appropriate processing scripts based on predefined expert rules, ensuring efficient routing of diverse query types. At the core, an LLM connected to a RAG system containing domain-specific knowledge—including code examples and technical manuals—generates Python code that produces targeted visualizations when executed within the simulation analysis environment.

Domain expert knowledge is codified into executable rules that generate analytical reports, which are automatically integrated into the system prompt to enhance the LLM's understanding of domain-specific requirements and industry best practices. When initial results revealed that generated plots lacked informativeness and contained visual flaws (Fig. 7), we extracted additional rules from visualization experts focusing on the most frequently used plot types: 2D relation plots, parallel plots, and history plots (Section 4.2). These expert-derived rules were directly implemented in the system prompt to guide visualization generation.

The final component combines all previous outputs into a comprehensive query incorporating user requirements, domain knowledge, and visualization best practices. Based on this enhanced query, the LLM generates optimized scripts that produce high-quality visualizations. This implementation addresses critical challenges through intelligent classification for real-time responsiveness, codified rules and RAG for domain expertise integration, expert-derived guidelines for visual quality assurance, and scalable architecture supporting diverse visualization requirements.

6 Technical validation

To validate our agent-based approach, we conducted a comparative analysis to demonstrate its effectiveness in mitigating the expert bottleneck. We assess our agent from two perspectives: **1. Quality of the final visualization** 1. We show that our AI agent enables a non-expert to generate visualizations that are as insightful as those created by a human expert, using only a simple prompt through autonomous agent operation. **2. Quality of the Generated Code** 2. We provide a technical evaluation of the underlying Python code, proving that our system generates objectively better code by correctly implementing codified expert knowledge.

Our validation compares a **Baseline System** (a LLM only connected to RAG) against our **Proposed System** (LLM based AI agent). We demonstrate that our agent produces expert-level results without requiring expert involvement from the user.

6.1 Quality of the Final Visualization

We present scenarios where non-expert users generate insightful plots from their Simulation Analysis software simulation data. A visualization expert—a data visualization specialist with 20 years of experience in simulation data analysis, currently working at the case company — evaluated the generated plots. The expert was provided with the original data, the user's request, and the generated visualization, and asked to evaluate both analytical insight and visual effectiveness based on their professional expertise and experience with simulation data visualization.

6.1.1 Scenario: Visualizing Convergence in a History Plot. **Goal** An engineer wants to demonstrate that simulation objectives have converged, a critical step for validating results.

User Input (provided to both systems) `"Please generate a history plot to check convergence."`

Human Result: The engineer used identical solid lines for both converged and non-converged objectives (Fig. 4), failing to communicate convergence status—the primary goal.

Baseline System Result: The LLM+RAG generated a plot without useful information (Fig. 3).

Proposed System Result: Agent correctly applied expert rules, using dashed lines for non-converged and solid lines for converged variables (Fig. 5), immediately communicating the status and fulfilling the request accurately.

6.1.2 Scenario: Creating an Informative 2D Relation Plot. **Goal** A junior engineer needs to understand the trade-off between "total mass" and "first torsion frequency" to identify optimal design choices.

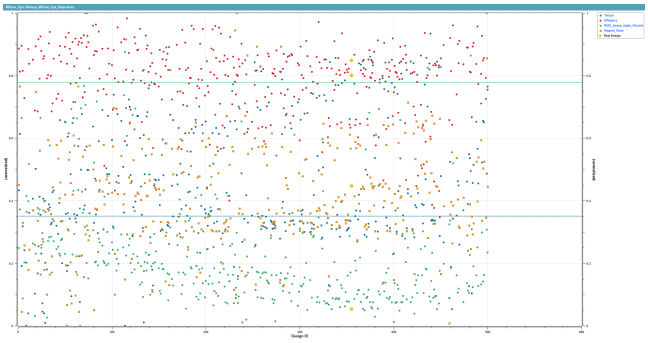

Figure 3: History plot generated by LLM+RAG

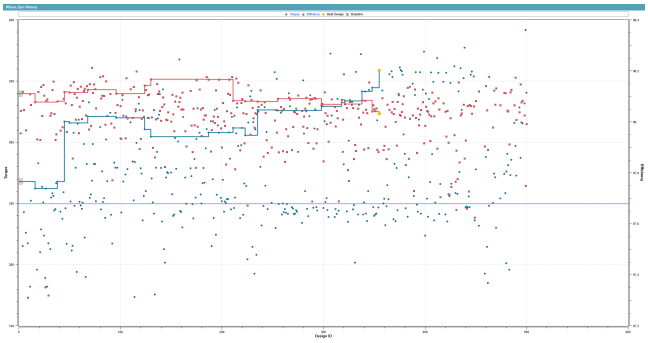

Figure 4: Human generated history plot

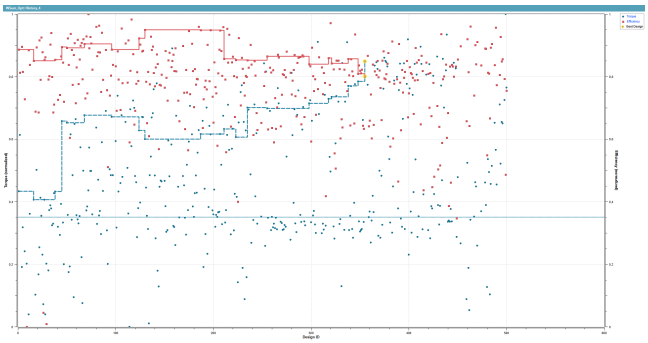

Figure 5: History plot generated by AI agent

User Input (provided to both systems) `please generate python code for 2d relation plot with variables total mass, first torsional frequency and total cost`

Human Result: The expert used color to represent material type and structured the plot to highlight the trade-off between weight and cost (Fig. 6). The plot guides users to identify a cost-effective steel option (blue dot) among lighter aluminum alternatives (red dots).

Baseline System Result: The baseline LLM produced a technically correct but uninformative plot (Fig. 7). It failed to use color effectively, placed incomparable variables on the same axis, and did not highlight the critical cost-effective outlier.

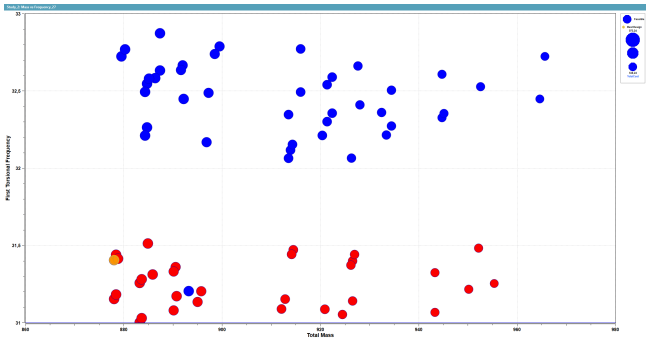

Figure 6: Human generated 2D relation plot

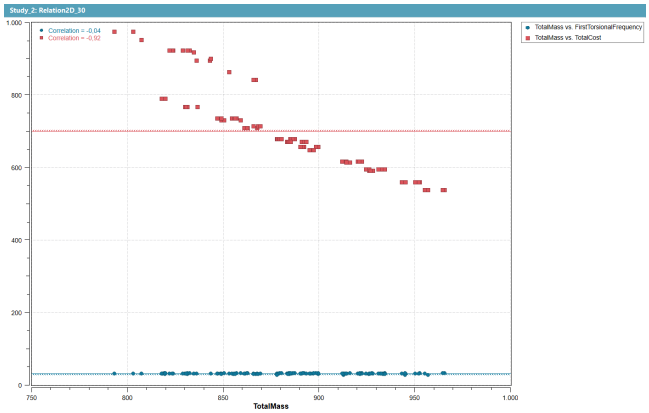

Figure 7: LLM+RAG generated 2D relation plot

Figure 8: AI agent generated 2D relation plot

Proposed System Result: Agent generated an improved visualization (Fig. 8) that highlights the optimal solution and enhances readability. While less insightful than expert plots in design set selection, it demonstrates clear improvements over the baseline through expert-derived visualization principles.

6.1.3 Scenario: Creating an Informative Parallel Plot. **Objective** A junior engineer requires a visualization to distinguish between the optimal design and outlier configurations in the dataset.

User Input (provided to all systems) `"Please generate a parallel plot."`

Human Expert Result: The expert visualization (Fig. 9) effectively reveals that the outlier uses aluminum for most components and lacks battery glue. While comprehensive, grouping range variables with categorical variables reduces readability.

Baseline System Result: The baseline system produced functional code that indiscriminately includes all variables (Fig. 10), lacking emphasis on the optimal design and outlier, making it difficult to extract actionable insights.

Proposed System Result: Agent generated a more informative visualization (Fig. 11) with enhanced readability through optimal design highlighting, improved legends, and better visual hierarchy. However, it currently lacks outlier detection capabilities—a targeted area for future development.

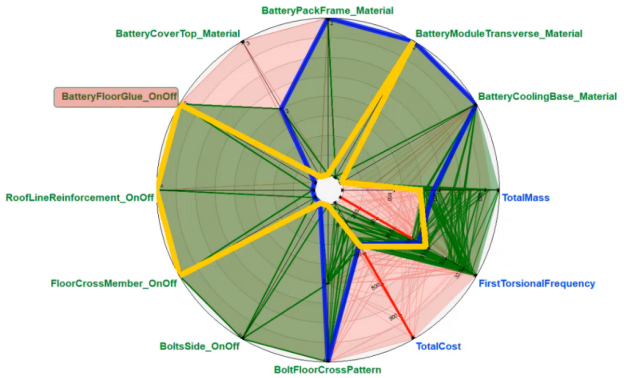

Figure 9: Human expert-generated parallel (radial) plot. Yellow: optimal design, Blue: outlier design, Green: feasible designs, Red: infeasible designs

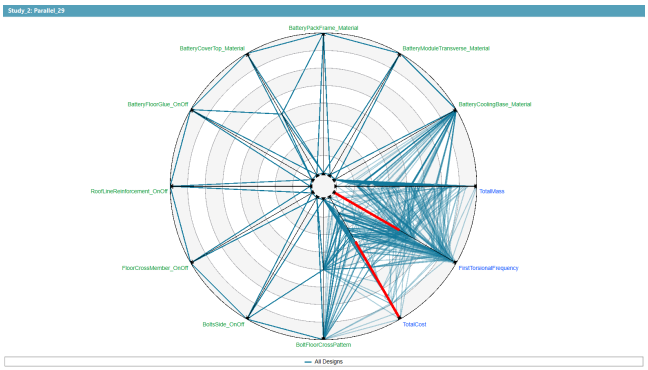

Figure 10: Baseline LLM+RAG generated parallel (radial) plot

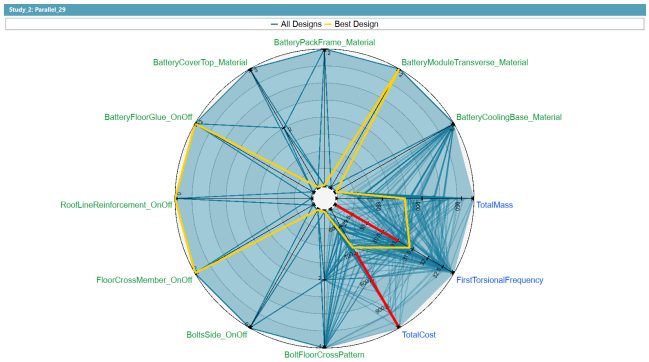

Figure 11: Proposed system-generated parallel (radial) plot with enhanced highlighting and legend

6.2 Quality of Generated Code

The enhanced visualization quality directly correlates with the system's ability to generate superior code. We evaluated code quality using the methodology established by [27], focusing on three critical dimensions: (1) Code validity - syntactic correctness and runtime error absence. (2) Code correctness - functional accuracy and adherence to requirements. (3) Output quality - effectiveness of resulting visualizations.

Scope of Evaluation We excluded code security from our assessment since the generated scripts serve solely visualization purposes and pose no security risks. Software reliability, as defined by [13], was also omitted because our scripts execute once for visualization generation rather than requiring sustained operation. Similarly, code maintainability was not evaluated since these scripts are designed for single-use generation, though they can be adapted for reuse with appropriate modifications.

To address framework generalizability, we validated our physics-agnostic rules across three distinct engineering domains representing fundamentally different physical phenomena:

6.2.1 Cross-Domain Validation: Project Descriptions. Our validation employs five projects spanning three distinct engineering domains:

Automotive Battery Design (Scripts 1-2): Electrochemical optimization of battery systems for electric vehicles, focusing on parameters such as total mass, energy density, and thermal management. The optimization objectives include minimizing weight while maximizing capacity and first torsional frequency.

Electric Motor Optimization (Script 4): Electromagnetic motor design balancing competing objectives of torque output and efficiency while managing thermal constraints. Variables include rotor geometry, winding configurations, and material selections.

Structural Control Arm Analysis (Script 5): Mechanical optimization of automotive suspension control arms, focusing on structural stress analysis and weight reduction. Objectives include minimizing mass while maintaining structural integrity under load.

These three domains represent fundamentally different physical phenomena—electrochemical, electromagnetic, and mechanical systems—providing a rigorous test of the framework's physics-agnostic design.

6.2.2 Evaluation Framework. Initial testing revealed that standalone LLMs consistently generated Python-specific solutions using libraries such as matplotlib and seaborn, which are incompatible with the Simulation Analysis software environment. To ensure fair comparison, we evaluated our proposed agent approach against LLM+RAG (augmented with code examples).

6.2.3 Expert Evaluation Process. We conducted comprehensive testing across numerous examples, presenting five representative cases here. 12 evaluators with diverse technical backgrounds independently assessed the results: a simulation expert with 20 years of experience in CAE software; a simulation optimization expert with multiple years of specialized domain knowledge; a software engineering manager with 12 years of experience in simulation software; 2 AI software engineering experts with extensive programming experience; a PhD in software engineering, leading a small data analytics and platform development team; a data scientist with 4 years of experience; an electrical engineer with 2 years of experience in CAD design; a computer scientist with 6 years of experience in coding; a business economist with 1 year of experience in python; a mechanical engineer with background in robotics and 4 years of experience in programming; an electrical engineer with 1.5 years of experience in LLM development.

Additionally, the systems were evaluated by members of the product software team, but these results are proprietary and cannot be disclosed in this paper.

To enhance the robustness of our assessment methodology, we employed Claude 4.5 Sonnet as a third assessor to provide triangulation of findings and reduce potential human bias in evaluation. Additionally, we evaluated Claude's alignment with expert human judgment to assess whether the AI assessor accurately captures what expert humans consider to be superior solutions.

6.2.4 Assessment Criteria. **Code Validity** Generated scripts were evaluated for syntactic correctness using binary scoring (0 = invalid, 1 = valid). Following [27], this measures compliance with programming language syntax rules and identifies potential runtime errors.

Code Correctness We assessed "the extent to which the generated code performs as intended" [27] across four dimensions using binary scoring based on [1]:

- *Code efficiency*: Optimization level without compromising functionality (0 = requires optimization, 1 = appropriately concise)
- *Documentation quality*: Adequacy of comments and documentation (0 = absent/inadequate, 1 = present/sufficient)
- *Exception handling*: Implementation of error handling mechanisms (0 = absent, 1 = present)
- *Code cleanliness*: Absence of unused variables or redundant elements (0 = contains unused elements, 1 = clean)

Output Quality We evaluated visualization effectiveness based on the script's ability to: (i) select appropriate data dimensions for analytical goals, (ii) use effective visual encoding for clear communication, and (iii) highlight critical information supporting decision-making.

6.2.5 Results. Tables 1 and 2 summarize the evaluation scores for both approaches. Both systems demonstrated equivalent performance in code validity and correctness, confirming LLMs' general capability for producing syntactically correct scripts. However, the agent approach significantly outperformed the baseline in output quality and instruction adherence across all evaluation criteria.

Metric	Scenario	Human (n=12)			Claude 4.5 Sonnet		
		Mean	SD	Mode	Mean	SD	Mode
Code Validity	S1	1.00	0.00	1	1.00	-	1
	S2	1.00	0.00	1	1.00	-	1
	S3	1.00	0.00	1	1.00	-	1
	S4	1.00	0.00	1	1.00	-	1
	S5	1.00	0.00	1	1.00	-	1
Code Correctness	S1	3.42	0.51	3	2.00	-	2
	S2	2.83	0.58	3	2.00	-	2
	S3	3.00	0.43	3	3.00	-	3
	S4	2.83	0.58	3	3.00	-	3
	S5	3.08	0.29	3	3.00	-	3
Output Quality	S1	2.75	0.62	3	3.00	-	3
	S2	2.00	1.21	3	3.00	-	3
	S3	2.75	0.62	3	3.00	-	3
	S4	3.00	0.00	3	3.00	-	3
	S5	2.50	1.00	3	3.00	-	3

Table 1: Code Quality Assessment - AI Agent: Per-Scenario Breakdown (Human: 12 evaluators per scenario; Claude: single AI evaluation per scenario)

Metric	Scenario	Human (n=12)			Claude 4.5 Sonnet		
		Mean	SD	Mode	Mean	SD	Mode
Code Validity	S1	1.00	0.00	1	1.00	-	1
	S2	1.00	0.00	1	1.00	-	1
	S3	1.00	0.00	1	1.00	-	1
	S4	1.00	0.00	1	1.00	-	1
	S5	1.00	0.00	1	1.00	-	1
Code Correctness	S1	2.33	0.65	2	2.00	-	2
	S2	2.83	0.39	3	2.00	-	2
	S3	1.83	1.11	3	2.00	-	2
	S4	2.08	0.79	2	2.00	-	2
	S5	2.67	0.49	3	2.00	-	2
Output Quality	S1	0.75	0.87	0	1.00	-	1
	S2	1.17	1.19	0	0.00	-	0
	S3	0.50	0.67	0	0.00	-	0
	S4	0.42	0.90	0	0.00	-	0
	S5	1.42	1.24	1	0.00	-	0

Table 2: Code Quality Assessment - LLM with RAG only: Per-Scenario Breakdown (Human: 12 evaluators per scenario; Claude: single AI evaluation per scenario)

Script 1 (Parallel/Radial Plots) Tables show Proposed (Mean = 2.75, SD = 0.62) vs. Baseline (Mean = 0.75, SD = 0.87) — a 267% improvement. The Mode difference (3 vs. 0) indicates most evaluators found baseline output analytically worthless, while proposed system outputs were consistently rated highest quality. The proposed system distinguished optimal/baseline designs and filtered for feasible solutions. Claude's divergent assessment (Baseline = 1.00 vs. human = 0.82) illustrates AI tendency to credit functional execution despite poor interpretability.

Script 2 (2D Relation Plot) Proposed system (Mean = 2.00, SD = 1.21, Mode = 3) vs. Baseline (Mean = 1.17, SD = 1.19, Mode = 0) demonstrates 71% improvement. The baseline's Mode = 0 indicates most evaluators found minimal value, while the proposed system's Mode = 3 shows most evaluators rated it highest quality. The proposed system's variance (SD = 1.21) suggests some evaluators penalized suboptimal dimension selection (Fig. 12) while others valued the correlation coefficients being displayed. Baseline's high variance (SD = 1.19) and Mode = 0 indicate fundamental disagreement on whether the output provided any value. Claude's assessment (Proposed = 3.00, Baseline = 0.00) aligned with human

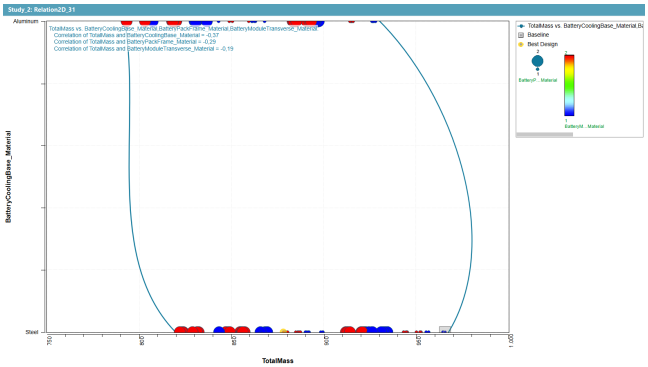

Figure 12: 2D relation plot - Car Batteries - Proposed system

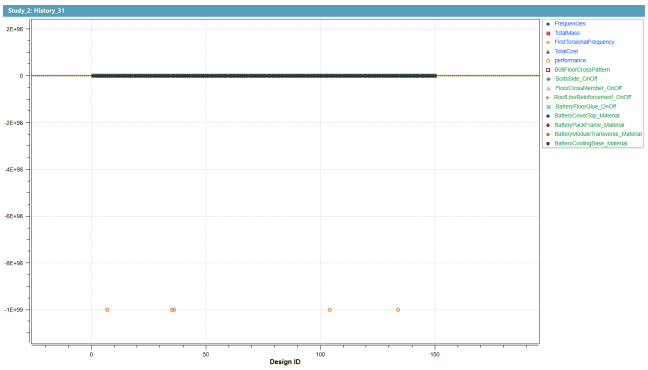

Figure 13: History plot - Car Batteries - Baseline system

Mode patterns, correctly identifying the baseline's categorical-vs-continuous plotting error.

Script 3 (History Plot) Strong evaluator consensus for proposed system (Mean = 2.75, SD = 0.62, Mode = 3) versus baseline (Mean = 0.50, SD = 0.67, Mode = 0) demonstrates 450% improvement. Baseline Mode = 0 (most evaluators rated it worthless) versus Proposed Mode = 3 (highest quality). The baseline (Fig. 13) plotted 14 parameters with catastrophic Y-axis scaling ($-1E{+}99$ to $2E{+}98$), while proposed system applied normalization and convergence indicators (dashed lines)—directly using visualization expert rules from Section 4.2. Claude's perfect score (3.00) for proposed system aligns with human consensus, validating this specific rule implementation.

Script 4 (History Plot - Electric Motors) Achieved perfect human consensus for proposed system (Mean=3.00, SD = 0.00, Mode=3)—the only scenario with zero evaluator disagreement in output quality — versus baseline's poor performance (Mean = 0.42, SD = 0.90, Mode = 0). This demonstrates that when codified rules align perfectly with scenario requirements, the system achieves unanimous evaluators approval. The baseline repeated the critical flaw from Script 3, plotting 14 parameters simultaneously with catastrophic Y-axis scaling ($-1E + 99$ to $2E + 99$), creating an analytically worthless flat line. The proposed system intelligently selected related parameters (Torque and Efficiency), co-plotting them with dual Y-axes and convergence indicators. Claude's perfect score (3.00) matched human consensus, while rating baseline = 0.00

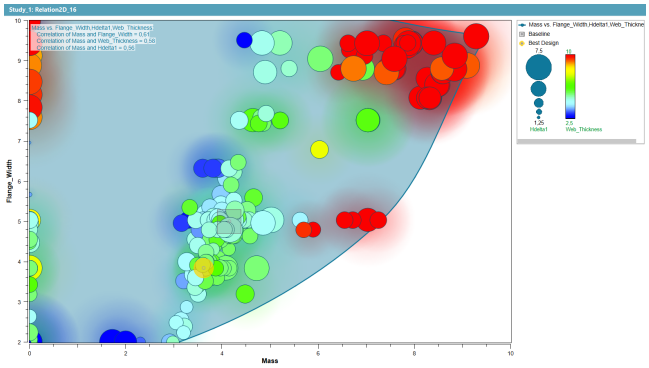

Figure 14: 2D relation - Control Arms - proposed system

— more harshly than humans (1.00), suggesting AI assessors apply stricter standards for extreme scaling errors.

Script 5 (2D Relation Plot - Control Arms) Baseline achieved its best Output Quality performance (Mean = 1.42, SD = 1.24, Mode = 1), yet still underperformed the proposed system (Mean = 2.50, SD = 1.00, Mode = 3), representing a 76% improvement. The simulation expert praised the proposed system: "The plot clearly shows the optimization process, normalization along the Y-axis is done correctly, colors are chosen appropriately, and it is also convenient that the line type reflects convergence." The proposed system (Fig. 14) plotted Mass objective vs. Flange_Width variable with multi-dimensional encoding (bubble size, color gradient, correlation coefficients: 0.61, 0.58, 0.55), while the baseline selected less informative variable-vs-response relationships. Claude rated the baseline = 0.00, more harshly than humans, noting it "creates a suboptimal visualization by plotting variable vs. response instead of the more analytically valuable objective vs. variable relationship, showing only a simple density cloud with limited optimization insights."

6.3 Synthesis of Evaluation Results

The technical evaluation demonstrates that our framework successfully codifies expert domain knowledge into an agent, addressing our three primary contributions: (1) systematic framework for capturing expert knowledge, (2) empirical validation, and (3) automated AI agent for addressing organizational bottlenecks.

Contribution 1: Framework Validation The proposed system achieved 206% higher Output Quality (mean: 2.60 vs. 0.85), with Mode = 3 across all scenarios versus baseline's Mode = 0 in 4/5 scenarios. This extends beyond [22]'s demonstration that LLMs can build charts, showing proper construction alone is insufficient without domain knowledge integration. In comparison to [26]'s system of LLMs with prompts, our approach builds a single complex agent integrating a classifier, RAG, and codified rules. This unified architecture architecture enables potentially domain-agnostic rule application, validated across diverse physics simulation scenarios in this work.

Lower variance in Code Correctness (SD: 0.29-0.58 vs. 0.67-1.24) and Output Quality (SD: 0.00-1.00 vs. 0.67-1.24) demonstrates predictable results. Script 4's perfect consensus (SD = 0.00, Mean =

3.00) validates our unified framework achieves more predictable results than [10]'s distributed multi-agent coordination.

Contribution 2: Empirical Validation Expert assessments confirmed framework-codified principles translate into observable characteristics. The simulation expert noted baseline failures: "no proper scales and no normalization, rendering plots unreadable," while proposed outputs "clearly show the optimization process, normalization done correctly, colors chosen appropriately, and line type reflects convergence." While [17] demonstrated LLM visualization using code snippets, our approach extends this by systematically codifying expert rules that address domain-specific analytical requirements beyond code examples.

Scripts 3 and 4 showed largest improvements (450%, and 614%) where baseline failed fundamental principles. Our 614% improvement significantly exceeds [28]'s 30% gain from rule integration, suggesting our comprehensive framework achieves synergistic effects beyond simple rule addition. Script 2's moderate improvement (71%, SD=1.21) demonstrates our framework handles both primitive rules (normalization) and compositional rules (multi-dimensional encoding) effectively, aligning with [24]'s observations about rule complexity.

AI assessor alignment (100% on Code Validity) addresses [18]'s earlier concerns about LLMs' rule-following—our results with newer versions demonstrate significantly improved adherence when rules are properly codified.

Contribution 3: Addressing Expert Bottlenecks The framework's consistent Mode=3 ratings across diverse contexts demonstrate our AI agent successfully addresses challenges identified by [11] and [5] regarding non-experts' visualization struggles and expert interpretation requirements. Our approach goes beyond [6]'s visualization literacy by autonomously generating expert-level outputs that non-experts create through simple prompts, enabling scalable production of high-quality visualizations without requiring domain expertise for each implementation.

7 Conclusion

This paper addresses the pervasive challenge of expert bottlenecks in organizations, where critical domain knowledge remains siloed with few specialists. We investigated how domain knowledge from human experts can be captured, codified, and leveraged to construct LLM-based AI agents capable of autonomous expert-level performance. We proposed and validated a systematic software engineering framework that empowers non-experts to achieve expert-level outcomes through AI agents.

We demonstrated our framework's efficacy through a rigorous case study in Simulation Analysis software simulation data visualization. We successfully engineered an AI agent by integrating a Retrieval-Augmented Generation (RAG) system for Simulation Analysis software-specific Python code generation, incorporating codified expert rules, and embedding visualization design principles directly into the Agent.

The key findings from our technical validation demonstrate the framework's effectiveness. Our agent achieved 206% improvement in output quality (mean: 2.60 versus 0.85) across five scenarios spanning three simulation domains, with Mode = 3 ratings in all cases. Evaluator assessments validated the practical impact: baseline outputs were deemed unreadable with no proper scales and no normalization, while proposed system outputs received praise for showing the optimization process clearly with normalization done correctly and appropriate color choices. The system enables non-experts using simple prompts to generate visualizations that correctly apply nuanced expert rules—such as dashed lines for non-converged variables—that even human engineers sometimes miss. By codifying and embedding expert knowledge in this manner, our framework effectively bridges the expertise gap and mitigates the expert bottleneck, allowing non-experts to produce high-quality, insightful visualizations while freeing domain experts to focus on more complex, specialized tasks that truly require their unique skills. This addresses a critical organizational scalability challenge where essential domain knowledge traditionally remained concentrated among few specialists. To support this, a Physics-Agnostic design pattern proved essential for scalability to other domains. By decoupling visualization rules from specific physical phenomena, we achieved zero-shot reuse across battery, motor, and structural domains without retraining. Finally, strong AI-Human alignment suggests that 'LLM-as-a-Judge' frameworks can serve as reliable proxies for rapid regression testing to reduce human evaluation bottlenecks.

This research contributes a robust AI agent for visualization generation and a systematic, validated framework for engineering AI agents with human expert domain knowledge. Our solution represents a significant step towards democratizing access to specialized expertise through an agent, enabling more efficient and effective data analysis across industries, and ultimately fostering greater productivity and innovation within organizations.

8 Limitations and Threats to Validity

8.0.1 Limitations. **Expert Selection and Internal Validity** Our framework development relied on knowledge from two experts within a single organization, potentially limiting the diversity of captured expertise and introducing selection bias. While methodologically appropriate for knowledge engineering [12], this constrains the generalizability of codified rules. **Evaluation Methodology** While code quality used standardized metrics [27], visualization effectiveness required expert judgment with clear criteria, mitigated through multi-assessor triangulation.

8.0.2 Scope and External Validity. **Domain Specificity** Validation spanned three physics domains but remained within simulation-based optimization contexts. Effectiveness in non-simulation domains (medical visualization, financial analytics) remains unvalidated, limiting external validity. **Organizational Context** The single-company case study within a large technology organization limits generalizability to different organizational contexts, user types, and expertise levels.

8.1 Future Work

Future research should address these limitations through: dynamic knowledge integration enabling learning from user feedback, enhanced user interaction capabilities for iterative refinement, cross-domain validation in multiple specialized fields, and large-scale

evaluation through comprehensive user studies and standardized benchmarks.

Acknowledgments

The authors gratefully acknowledge the support provided by the Software Center. The authors also thank their colleagues and supervisors within Siemens for valuable discussions and feedback during this research. Special thanks to Roberto d'Ippolito, Ranny Sidhu, Jason Dunn and Guillermo Zschaeck from the product software team for their contributions, support and evaluation.

References

[1] Sally Almanasra and Khaled Suwais. 2025. Analysis of ChatGPT-generated codes across multiple programming languages. *IEEE Access* (2025).
[2] Aysha Ameerah, Mansi Patel, Maansi Srinivasan, Juslyn Dhingra, Noah Beinart, Abraar Ahmed, Etse-Oghena Y Campbell, Christopher R Idelson, and John M Uecker. 2025. Blurred vision, clear concern: linking poor visualization with adverse events in minimally invasive surgery. *Surgical Endoscopy* (2025), 1–9.
[3] Jessica S Ancker, Yalini Senathirajah, Rita Kukafka, and Justin B Starren. 2006. Design features of graphs in health risk communication: a systematic review. *Journal of the American Medical Informatics Association* 13, 6 (2006), 608–618.
[4] Tom Biselli, Katrin Hartwig, Niklas Kneissl, Louis Pouliot, and Christian Reuter. 2025. ChartChecker: A User-Centred Approach to Support the Understanding of Misleading Charts. In *Proceedings of the 2025 ACM Designing Interactive Systems Conference*. 2075–2102.
[5] Kiroong Choe, Chaerin Lee, S. Lee, Jiwon Song, Aeri Cho, Nam Wook Kim, and Jinwook Seo. 2024. Enhancing Data Literacy On-demand: LLMs as Guides for Novices in Chart Interpretation. *IEEE transactions on visualization and computer graphics* PP (2024). doi:10.1109/TVCG.2024.3413195
[6] Amit Kumar Das, Mohammad Tarun, and Klaus Mueller. 2025. Charts-of-Thought: Enhancing LLM Visualization Literacy Through Structured Data Extraction. *arXiv preprint arXiv:2508.04842* (2025).
[7] Zehang Deng, Yongjian Guo, Changzhou Han, Wanlun Ma, Junwu Xiong, Sheng Wen, and Yang Xiang. 2024. AI Agents Under Threat: A Survey of Key Security Challenges and Future Pathways. *Comput. Surveys* 57 (2024), 1 – 36. doi:10.1145/3716628
[8] S. Franconeri, Lace M. K. Padilla, P. Shah, Jeffrey M. Zacks, and J. Hullman. 2021. The Science of Visual Data Communication: What Works. *Psychological Science in the Public Interest* 22 (2021), 110 – 161. doi:10.1177/15291006211051956
[9] Sivanagaraju Gadiparthi. 2024. Effective Visualization Techniques for Multidimensional Data: A Comparative Analysis. *Int. J. Sci. Res.(IJSR, no* (2024).
[10] Kanika Goswami, Puneet Mathur, Ryan Rossi, and Franck Dernoncourt. 2025. Plotgen: Multi-agent llm-based scientific data visualization via multimodal feedback. *arXiv preprint arXiv:2502.00988* (2025).
[11] Lars Grammel, Melanie Tory, and Margaret-Anne Storey. 2010. How information visualization novices construct visualizations. *IEEE transactions on visualization and computer graphics* 16, 6 (2010), 943–952.
[12] Robert R Hoffman, Nigel R Shadbolt, A Mike Burton, and Gary Klein. 1995. Eliciting knowledge from experts: A methodological analysis. *Organizational behavior and human decision processes* 62, 2 (1995), 129–158.
[13] Gurpreet Kaur and Kailash Bahl. 2014. Software reliability, metrics, reliability improvement using agile process. *International Journal of Innovative Science, Engineering & Technology* 1, 3 (2014), 143–147.
[14] Daiki Kido, Osamu Imazeki, Kahoru Nakayama, and Yasuhiro Nakano. 2023. Visualization of Computer-aided Engineering Simulations on Multi-User Virtual Reality for Consensus Building. *eCAADe proceedings* (2023). doi:10.52842/conf.ecaade.2023.2.843
[15] Chujun Lin and Mark Allen Thornton. 2021. Fooled by beautiful data: Visualization aesthetics bias trust in science, news, and social media. (2021).
[16] Maxim Lisnic, Cole Polychronis, Alexander Lex, and Marina Kogan. 2023. Misleading beyond visual tricks: How people actually lie with charts. In *Proceedings of the 2023 CHI conference on human factors in computing systems*. 1–21.
[17] Tanwi Mallick, Orcun Yildiz, David Lenz, and Tom Peterka. 2024. Chatvis: Automating scientific visualization with a large language model. In *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 49–55.
[18] Norman Mu, Sarah Chen, Zifan Wang, Sizhe Chen, David Karamardian, Lulwa Aljeraisy, Basel Alomair, Dan Hendrycks, and David Wagner. 2023. Can LLMs Follow Simple Rules? *arXiv preprint arXiv:2311.04235* (2023).
[19] Benson Rosen, Stacie Furst, and Richard Blackburn. 2007. Overcoming barriers to knowledge sharing in virtual teams. *Organizational dynamics* 36, 3 (2007), 259–273.
[20] Choro Ulan Uulu, Mikhail Kulyabin, Layan Etaiwi, Nuno Miguel Martins Pacheco, Jan Joosten, Kerstin Röse, Filippos Petridis, Jan Bosch, and Helena Holmström Olsson. 2026. AI for Better UX in Computer-Aided Engineering: Is Academia Catching Up with Industry Demands? A Multivocal Literature Review. In *Euromicro Conference on Software Engineering and Advanced Applications*. Springer, 298–312.
[21] Abhishek Vajpayee. 2024. Fostering Excellence through Mentorship: A Study of Professional Growth in Data Engineering. *International Journal for Research in Applied Science and Engineering Technology* (2024). doi:10.22214/ijraset.2024.64122
[22] Pere-Pau Vázquez. 2024. Are llms ready for visualization?. In *2024 IEEE 17th Pacific Visualization Conference (PacificVis)*. IEEE, 343–352.
[23] Aliaksei Vertsel and Mikhail Rumiantsau. 2024. Hybrid llm/rule-based approaches to business insights generation from structured data. *arXiv preprint arXiv:2404.15604* (2024).
[24] Siyuan Wang, Zhongyu Wei, Yejin Choi, and Xiang Ren. 2024. Can llms reason with rules? logic scaffolding for stress-testing and improving llms. *arXiv preprint arXiv:2402.11442* (2024).
[25] Wenkai Yang, Yankai Lin, Jie Zhou, and Ji-Rong Wen. 2025. Distilling rule-based knowledge into large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*. 913–932.
[26] Zhiyu Yang, Zihan Zhou, Shuo Wang, Xin Cong, Xu Han, Yukun Yan, Zhenghao Liu, Zhixing Tan, Pengyuan Liu, Dong Yu, et al. 2024. Matplotagent: Method and evaluation for llm-based agentic scientific data visualization. *arXiv preprint arXiv:2402.11453* (2024).
[27] Burak Yetiştiren, Işık Özsoy, Miray Ayerdem, and Eray Tüzün. 2023. Evaluating the code quality of ai-assisted code generation tools: An empirical study on github copilot, amazon codewhisperer, and chatgpt. *arXiv preprint arXiv:2304.10778* (2023).
[28] Zhaocheng Zhu, Yuan Xue, Xinyun Chen, Denny Zhou, Jian Tang, Dale Schuurmans, and Hanjun Dai. 2023. Large language models can learn rules. *arXiv preprint arXiv:2310.07064* (2023).