Accelerating Software Acquisition Using Generative AI for Regulatory Compliance

John Robert is a Principal Engineer at the Carnegie Mellon University Software Engineering Institute (SEI) and currently serves as the Deputy Director of the Software Solutions Division. In this role, Robert leads software engineering research and development (R&D) efforts in partnership with programs in the Department of Defense and industry. Robert is a co-author of Architecting the Future of Software Engineering: A National Agenda for Software Engineering Research & Development. Robert holds a Master of Software Engineering from CMU and a BS in Electrical Engineering from West Virginia University.

Carlos Olea is a Graduate Research Assistant at the Magnum Research lab, Vanderbilt University. His work focuses on interdisciplinary AI evaluation and utilization, with applications in fields from cyber-physical security to aerospace design to sports analytics.

Yash Hindka graduated with a B.S. in Computer Science from the University of Wisconsin-Madison in December 2021 with Highest Distinction. After graduating, he joined Raytheon as a software engineer, driven by the warfighters he grew up admiring from a young age. Yash currently works at the Software Engineering Institute (SEI) at Carnegie Mellon University (CMU). He joined the SEI to learn from and be a part of the strong academic tradition at CMU, all while continuing to support the warfighter. At the SEI, he analyzes embedded weapon system software for vulnerabilities and participates in research efforts to increase automation in the analysis process.

Nanette Brown is a Senior Member of the Technical Staff at the Software Engineering Institute. In that capacity she has worked as a researcher in the areas of agile and architecture and technical debt management. She has consulted with government agencies on Agile and Lean Practices, and Operational Test practices and policy development. Prior to joining the SEI, Nanette Brown spent over 20 years in product development, starting as a developer and continuing on to lead development organizations and change initiatives as the executive director of Architecture and Quality Management of a Fortune 500 company.

Douglas C. Schmidt is the Dean of Computing, Data Sciences & Physics at William & Mary. Dr. Schmidt has served the President-appointed and Senate-confirmed Director of Operational Test and Evaluation, where he was responsible for overseeing the evaluation of the operational effectiveness, suitability, survivability, and (when necessary) lethality of United States defense systems to defend the homeland and prevail in conflict. He also served as the Chief Technology Officer at the Carnegie Mellon University Software Engineering Institute (SEI).

Abstract

Detecting document incompleteness, inconsistencies, and discrepancies between regulatory documents and software artifacts is a common and people-intensive task for acquisition teams. Department of Defense (DoD) Acquisition environments have extensive documentation describing policies, guidance, and standards that must be repeatedly compared to delivered software artifacts for a DoD program to ensure regulatory conformance throughout a project's lifecycle. Acquisition professionals in these environments must learn the extensive and complex regulatory information, apply the knowledge to multiple projects, and identify document incompleteness, inconsistencies, and discrepancies (DIID) that could indicate non-compliance or high-risk areas. Currently, teams of people review multitudes of documents and data, reading and using general search on keywords to find relevant text to review and compare to regulatory documents. As the DoD continues moves toward DevSecOps with continuous integration and rapid capability deployment approaches, people-intensive approaches to ensure regulatory compliance are slow, do not scale, and delay mission capability.

This paper investigates the use of large language models (LLMs) to improve the efficiency and accuracy of DIID detection while enabling customization through prompt engineering. The proposed approach leverages LLMs to augment acquisition professionals by providing semi-automated and meaningful connections of software artifacts to regulatory documents. Testing approaches are proposed to assess the effectiveness of LLMs for DIID detection, and preliminary results are provided for detecting DIID with augmented LLMs. This paper also proposes prompt engineering approaches for DIID detection and suggests benefits for DIID detection in software acquisition activities.

Introduction

DoD software acquisition programs must adhere to a complex web of regulations, standards, and contractual requirements. Ensuring every requirement is addressed involves producing and reviewing extensive documentation, such as requirements, design specifications, and test plans. Currently, teams of analysts manually cross-check these artifacts against DoD policy checklists, derived from DoD directives or standards, to identify any section that is incomplete, inconsistent, or discrepant with respect to the official guidance. This manual process is tedious, error-prone, and people-intensive, so important omissions or contradictions can be overlooked due to sheer volume or reviewer fatigue. Moreover, as the DoD increasingly adopts DevSecOps practices to enable continuous integration and rapid deployment, human-centric compliance reviews struggle to keep pace, introducing delays in delivering mission capabilities.

Automating regulatory compliance checks has become an area of great interest to improve efficiency and reduce human error. A common regulatory compliance task is identifying where software acquisition or engineering artifacts are inconsistent, incomplete, or discrepant from regulatory or standards documents. Such document incompleteness, inconsistencies, and discrepancies (DIID) could indicate a regulatory requirement is not addressed or does not apply to a system, or it could be a simple omission. Software acquisition and engineering teams often begin regulatory or milestone reviews searching for DIID because they can be an indicator of potential risks or deviations from expected system performance. Currently, acquisition and engineering teams are using blunt word searches or lengthy reading and review to find DIID, which is both slow and error-prone because documents can use different words for similar topics.

Our goal is to augment acquisition professionals by automating the tedious detection of DIID issues, while human experts remain central to interpreting and resolving these issues. Determining the applicability generative AI to specific use cases, including software acquisition, requires assessing the opportunities and risks among multiple considerations for their contexts (Stephany Bellomo et al., 2023). The risks and opportunities of applying LLMs to DIID detection are discussed throughout this paper, along with some of the strengths of LLMs and understand the risks. This paper does not explore the question of AI-based recommended resolutions to detected DIID because automating DIID resolutions has higher risks given current AI capabilities.

Not every document discrepancy or omission is critical because some may be contextually justified. Identify all potential DIID issues is a necessary first step before human judgment can determine their severity. Automated support to augment humans in this task could significantly improve efficiency. However, any automation to detect DIID, particularly in mission-

or safety-critical environments, must keep humans as an integral part of the process and as the ultimate decision makers.

There is a pressing need for tools that can rapidly and reliably detect DIID across large collections of acquisition documents. This paper describes how we are assessing the use of LLMs to find DIID in software engineering artifacts efficiently by cross-referencing them with regulatory documents. We also outline prompt engineering techniques to tailor LLMs for this task and ensure they work effectively alongside humans. Finally, we describe gaps in testing and evaluating the LLM's performance on DIID detection and discuss preliminary findings. Our approach illustrates initial observations and proposes approaches to improve scalability and accuracy in compliance checking, ultimately accelerating software acquisition while maintaining confidence in regulatory conformance.

New Opportunities in AI-Augmented Regulatory Compliance

Recent advances in generative AI, including LLMs, offer a unique opportunity to transform software engineering and software acquisition (Robert et al., 2024). This transformation extends to automating and accelerating software regulatory compliance workflows. Unlike keyword-based search tools, LLMs can consider natural language context and make judgements about document content based on training and other data. The ability of LLMs to interpret semantics makes them suitable for text comparisons (Brown et al., 2020). Using LLMs for document analysis is a common topic of interest across many domains, including healthcare (Moilanen et al., 2022; Shokrollahi et al., 2023), financial or business applications (Cao et al., 2024; Shukla et al., 2023), or for analyzing legal documents (Prasad et al., 2024). In most of these domains, LLMs are used for document summaries and to provide insights into data for that domain.

LLMs like GPT-4, Gemini, and Ollama can analyze documents, respond to prompts, or generate natural language from unstructured text. LLM abilities to consider natural language context has generated interest in automation of document heavy tasks. For example, an LLM can interpret a requirement like “The system shall implement encryption in accordance with FIPS 140-2” and check that the design document specifies FIPS 140-2 compliant encryption beyond a simple word search of the document to include some semantic understanding.

By embedding the relevant regulations and policy documents into a vector database and using it to provide context, an LLM can be prompted to cross-reference an acquisition document against applicable rules. This approach, known as retrieval augmented generation (RAG), allows an LLM to find explicit regulatory clauses or past examples from the vector store to confirm its compliance checks (*RAG for LLMs*, n.d.). Using RAG to augment LLMs is currently part of many online services offered by many LLMs and can be implemented on stand-alone LLM solutions.

LLMs also offer the opportunity for people to interact with the documents and information in new ways. Prompt engineering (Liu et al., 2023) and prompt patterns (White et al., 2023) are techniques humans use to create and refine inputs to optimize responses from a generative AI model, but they also represent new opportunities for humans to interact with AI systems (Bozkurt, 2024). For software acquisition and engineering teams, prompt engineering and prompt patterns provide a flexible and natural way to customize and refine DIID detection.

Role of DIID in Regulatory Risk and Non-Compliance

DIID issues in DoD program documents are not just theoretical inconveniences. In practice they have been linked to project risks and delays. For example, (Brownsword, 2012) identified DoD program anti-patterns that contribute to cost overruns and schedule delays. Several of these patterns are examples of discrepancies or incompleteness in a DoD program's acquisition artifacts. A more recent GAO report identified that several programs failed to conduct or report cybersecurity testing phases despite a DoD policy requiring the tests (Weapon Systems Annual Assessment, 2022). These findings explain why the DoD mandates rigorous document reviews and checklist-driven inspections because even minor omissions can indicate compliance violations or additional risk if left undetected.

Despite these precautions, human reviewers struggle with information overload. Critical DIID issues can be missed simply because they are buried in hundreds of pages of specifications and plans. The more complex the program, the more likely an overwhelmed analyst might overlook a subtle inconsistency, which is why automated DIID detection assistants can prove invaluable. Such AI-powered assistants (e.g., an LLM scrutinizing the documents in parallel) act as a second pair of eyes, scanning at scale and flagging potential issues that warrant human attention. By catching contradictions and gaps that a single reviewer might miss, these tools can enhance overall compliance assurance.

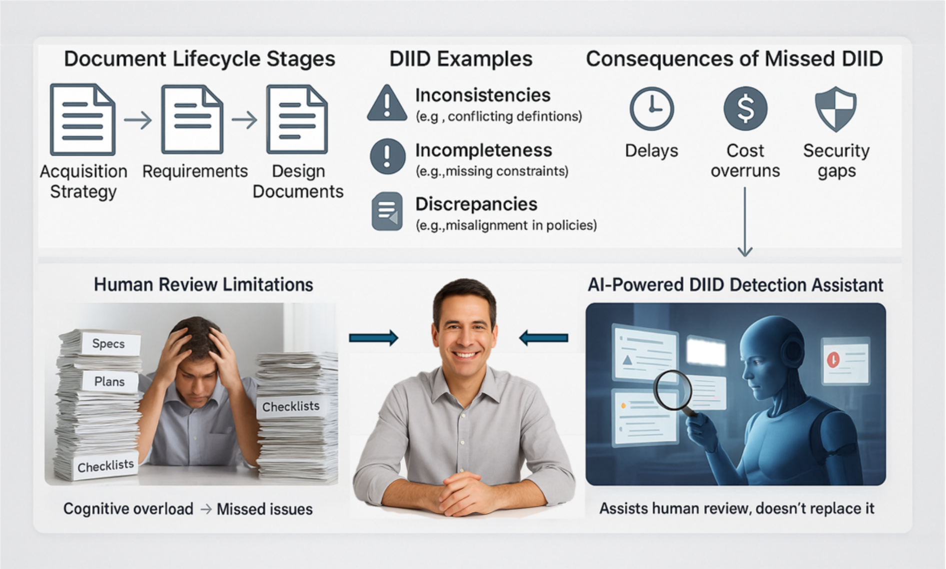
Document Lifecycle Stages
Acquisition Strategy
Requirements
Design Documents
DIID Examples
Inconsistencies
(e.g , conflicting defintions)
Incompleteness
(e.g.,missing constraints)
Discrepancies
(e.g.,misalignment in policies)
Consequences of Missed DIID
Delays
Cost overruns
Security gaps
Human Review Limitations
Specs
Plans
Checklists
Checklists
Cognitive overload → Missed issues
AI-Powered DIID Detection Assistant
Assists human review, doesn't replace it

Figure 1 Role of DIID in Software Acquisition

Figure 1 visualizes the role of DIID in software acquisition. Multiple documents are created and connected as part of a program lifecycle, and compliance with DoD regulations is both required and verified. However, DIID examples exist in many documents and remain unidentified due to the limitations in human reviewers and result in delays and other consequences. An AI-powered DIID detection assistant helps identify DIID and prevent delays or other issues.

It is important to note that the findings of AI tools would feed into the human review process, not replace it. The goal is to improve review effectiveness and reduce risk while allowing human experts to focus on judgment calls and solutions. In short, improving DIID detection through automation reduces risk in acquisition programs by increasing the likelihood that potential issues are found, and humans can review and analyze these issues.

DIID issues can surface at many stages of the software acquisition or software development lifecycle. DoD policies drive some requirements but also contribute to acquisition strategy, architecture, and testing. When DoD policies recommended or require processes or standards, these must be reflected in the relevant program documents. Some examples are provided in Table 1.

Table 1 DIID Examples by Lifecycle

Lifecycle Stage	DIID Examples
Acquisition Strategy	Goals conflicting with
Requirements	Missing safety constraints or security clauses from DoD standards or policies
Design Documents	Conflicting interface definitions
Test Plans/Reports	Incomplete traceability to requirements
Certification Artifacts	Discrepant terminology from regulatory standards

Defining DIID in the Acquisition Lifecycle

Document incompleteness, inconsistencies, and discrepancies (DIID) all indicate potential problems in software acquisition artifacts, but each term has a distinct meaning:

- *Incompleteness* - Important content is missing either within a document or across a set of documents. For example, if a safety policy mandates a “detailed analysis of safety-critical signals” in the architecture description, but the project’s Software Architecture Document only provides an analysis of “signals” without the safety-critical qualifier, this omission constitutes an incompleteness. Incompleteness typically means some requirement or context that should be present is absent, potentially necessitating additional review or rework to address the gap.
- *Inconsistency* - There are contradictions or a lack of uniformity in terminology or content either within one document or between documents. For instance, using “safety” and “security” interchangeably in a system description (when in some domains they imply different requirements) can confuse readers about the true requirements. Inconsistencies might not always signify an error because they can be acceptable in the context, but they often create confusion and may hint at deeper incompleteness or misunderstandings.
- *Discrepancy* - There is a direct conflict or divergence between facts or statements in one place versus another. Discrepancies can be factual (e.g., one document states a response time requirement as 0.01s while another cites 0.1s for the same event, a tenfold difference), policy-related (a process described in the project plan deviates from what a governing policy mandates), narrative (two documents describing the system’s behavior tell conflicting stories), or theoretical (project results or assumptions conflict with established theory or prior data). Discrepancies often indicate more serious issues because one of the conflicting statements could be wrong. If critical parameters differ, the impact can be severe.

These definitions are typical of what is found in dictionaries and literature on human interpretation or document analysis. Although there is no comprehensive list of different types of DIID, common types and examples are summarized in Table 2.

Table 2 Types of Incompleteness, Inconsistencies, and Discrepancies

Incompleteness	Inconsistency	Discrepancy
• **Incomplete**: Important context or terms are missing.	• **Terminology**: Using different terms interchangeably without clear definitions or consistency.	• **Factual discrepancies**: Conflicting factual information.

Incompleteness	Inconsistency	Discrepancy
	- **Structural**: Lack of uniform structure in presenting information.	- **Policy or procedural discrepancies**: Deviations from established protocols. - **Narrative discrepancies**: Different user stories fail to align. - **Theoretical discrepancies**: Actual results conflict with theoretical predictions.

Exploring DIID for Software Safety in DOD Regulatory Compliance

Exploring specific examples provides important insights into the opportunities and current limitations of automating DIID detection in DoD regulatory use cases. We have performed initial tests using currently available online LLMs, such as OpenAI ChatGPT 4o and Google Gemini 2.0 as well as locally hosted LLMs, such as Ollama, to perform simple DIID comparisons. Although testing is in progress, our initial results provide important insights into opportunities and risks to enable generative AI for regulatory compliance. Some of the testing insights are shared in blogs or webinars, to provide general observations while in progress (John Robert & Douglas Schmidt, 2024; Schmidt & Robert, 2024).

The initial testing has been performed on public (Distro-A) data comparing DoD and NASA software safety standards to text statements that represent different types of DIID. There are currently over 100 statements to help identify false positives, the majority of which do not contain DIID. There *are* examples of DIID, although not all sub types of discrepancies are represented at this time.

A common use case is when DoD programs must follow system or software safety standards to identify safety requirements or risks. For example, MIL-STD-882E is used for hazards analysis, and Safety Assurance Cases (*MIL-STD-882E*, 2023), which must align with and program-specific safety policies. The initial testing of DIID detecting used DoD standards, including or similar to MIL-STE-882E, to compare with different statements that do or do not contain DIID when compared to the standard. Inconsistencies could include mismatch in system functions between FHA and test procedures or missing causal chains in hazard analyses.

Using this DoD standard, we illustrate DIID examples in Table 3 that might be found in requirements documents, design documents, or testing documents. These are just a few of the many examples of possible DIID, which can be in any of the DIID types, across many documents, at different parts of the software development lifecycle, and for difference software releases over time.

Table 3 DIID Examples

DIID Type	DIID Example	DIID Justification
Incompleteness	The software test report includes results from functional and integration testing.	The DoD standard may require explicit evidence of software safety testing, including corner cases or hazardous conditions. The absence of test cases for safety-critical scenarios indicates incompleteness in test coverage.

Inconsistency	Cyber resiliency is ensured through encryption and secure boot mechanisms,” (in a System Description) vs. “Safety-critical software must handle fail-safe transitions during fault events for resiliency”	If the term “resiliency” is used in both contexts (cybersecurity and safety) but without consistent definitions, and without clarifying boundaries, it introduces inconsistency.
Discrepancy	The system shall complete data processing within 5 milliseconds,” (in Requirements Specification) vs. “The implemented architecture meets the 50 millisecond timing constraint for processing.” (in Design Document)	There is a factual discrepancy in performance in timing requirements with a 10x difference between what is required and what is implemented or claimed, possibly leading to a violation of real-time constraints mandated in mission-critical DoD systems.

The examples are a small sample of the many types of DIID tests. The statements are simple, but each statement must be reviewed by people, sometimes acquisition experts, to confirm that the tests are relevant DIID that human reviewers would find interesting or consider investigating in more detail. In addition, examples of statements without DIID, as perceived by human reviewers, are also needed to consider false positives. The number of tests needed to really exercise LLM solutions for DIID detection is extensive, particularly to consider all types of regulatory documents and software artifacts. Work is currently underway to further characterize DIID testing and gaps.

An important comment from software acquisition professionals and software engineers is that DIID could appear to be limited when compared to software artifacts, such as design or testing documents, but DIID could be extensive when compared to software code for that same system. It is common to have software plans or software artifacts claim performance or processes, only to review code or detailed testing to discover that the implementation is not consistent with the documentation. This known issue requires extension of DIID detection of regulatory documents beyond software artifacts to code. Commercial software development and analysis tools are already integrating LLMs into their workflows, and these solutions can be tested for DIID detection. In addition, the DIID detection for software artifacts remains important for software acquisition teams and this may be the earliest opportunities to automate some of the regulatory burden on acquisition teams. We strongly agree that DIID solutions must extend to code, but in practice today we frequently observe programs only reviewing software artifacts.

Figure 2 illustrates a notional DIID detection testing pipeline. One or more regulatory standards, DoD or other government standards, are used (top left) to create public DIID detection test data sets. The test data is created and/or reviewed by humans for accuracy and used to test multiple DIID detection architectures, including online LLMs, stand-alone (locally hosted) LLMs, or LLMs combined with RAG solutions. The test results are summarized to understand which solutions provide the best DIID detection, categorized by DIID type and other groupings.

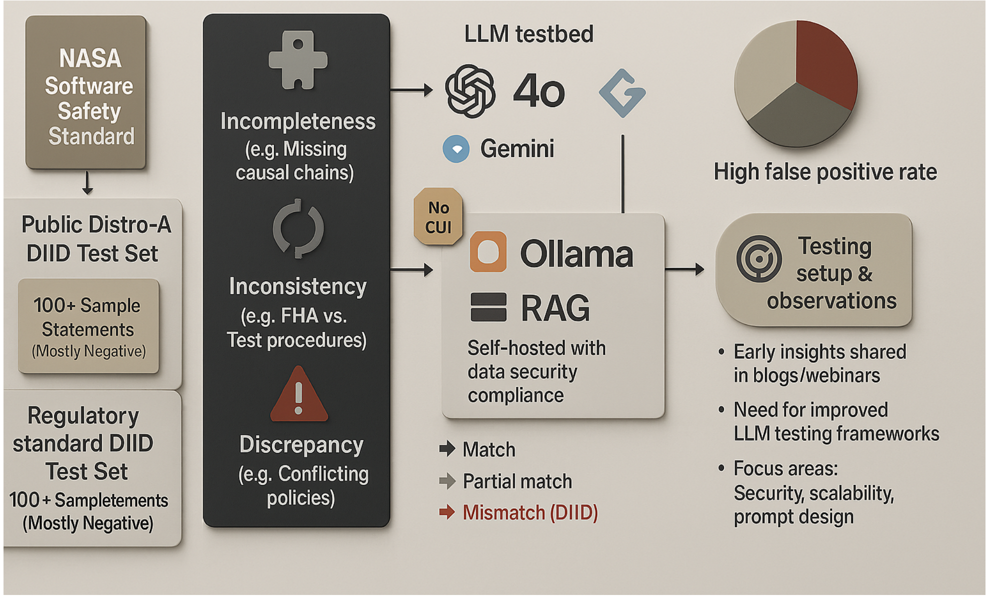
NASA Software Safety Standard
Public Distro-A DIID Test Set
100+ Sample Statements (Mostly Negative)
Regulatory standard DIID Test Set
100+Samplements (Mostly Negative)
Incompleteness (e.g. Missing causal chains)
Inconsistency (e.g. FHA vs. Test procedures)
Discrepancy (e.g. Conflicting policies)
LLM testbed
4o
Gemini
No CUI
Ollama
RAG
Self-hosted with data security compliance
Match
Partial match
Mismatch (DIID)
High false positive rate
Testing setup & observations
Early insights shared in blogs/webinars
Need for improved LLM testing frameworks
Focus areas: Security, scalability, prompt design

Figure 2 AI-Augmented DIID Detection Pipeline in DoD Safety Standards

Our initial testing is simplified from the actual needs for an operational system. For example, the following aspects are, for now, not the focus of our testing:

- *Data security* – Sensitive information must be protected. Some software acquisition activities may involve sensitive or proprietary information to LLMs, which raises concerns about data security and privacy. We never use CUI or secured information for these tests because the online LLM services do not support data security for CUI or proprietary data, which is a known concern. We therefore include self-hosted LLMs as part of our testing to observe similarities or differences in the responses. Organizations should always be aware of how to mitigate the data disclosure risks of LLMs and prevent access to private or protected data.
- *Scalability* – We initially uses LLMs with RAG for testing, but more online services have increased their support for scaling to support multiple documents. RAG is still needed for the self-hosted solutions.

These initial tests highlight a few important areas of discussion to achieve AI-augmented DIID detection for software acquisition use cases:

- *Prompt engineering*, which is important to ensure clear definition of DIID types and provide guidance on what documents to compare.
- *LLM testing frameworks* – As we assess current LLM testing frameworks and data sets, we found gaps for the DIID use case that point to the need for additional testing resources.

The Role of Prompt Engineering in DIID Detection

The flexibility of LLMs to support a wide range of uses effectively is supported in part by prompt engineering (Liu et al., 2023), which involves structured interactions with LLMs to optimize outputs via natural language interfaces. Prompt patterns (White et al., 2023) codify best practices for phrasing prompts to maximize extraction accuracy and provide knowledge transfer mechanisms to problem-solve with LLMs more effectively and accurately. Prompt patterns also enable more effective and repeatable performance of LLMs, and many patterns have been identified for a range of task objectives.

Prompt engineering is a key capability that provides new opportunities for DIID detection in software acquisition, e.g., the *Context Manager* pattern (White et al., 2023) directs an LLM to set the context. Prompt engineering, including identification and refinement of relevant prompt patterns, is thus central to our research. When DIID types are narrowly defined and examples are provided to LLMs through fine tuning or prompt engineering, DIID detection can be improved for specific use cases. In our initial experiments, failure to include DIID definitions in the prompt led to a high variance in the output and the increase confusion for the user. For instance, a user would have to read the output several times to understand is the output was consistent with the DIID types.

To detect DIID in software artifact analysis, an LLM must act like a reviewer, i.e., it should analyze the content, the context, and then apply reasoning to identify gaps. For incompleteness, this analysis means knowing what should be in the document (perhaps via a template or a checklist derived from policy) and checking if it is there. For inconsistencies, it entails comparing statements for logical alignment. For discrepancies, it requires cross-referencing external sources.

Figure 3 The Role of Prompt Engineering in DIID Detections a summary of prompt engineering for DIID detection. A DIID taxonomy must be provided to ensure consistency in the types of DIID. Specific context is sometimes needed to define how different documents are related.

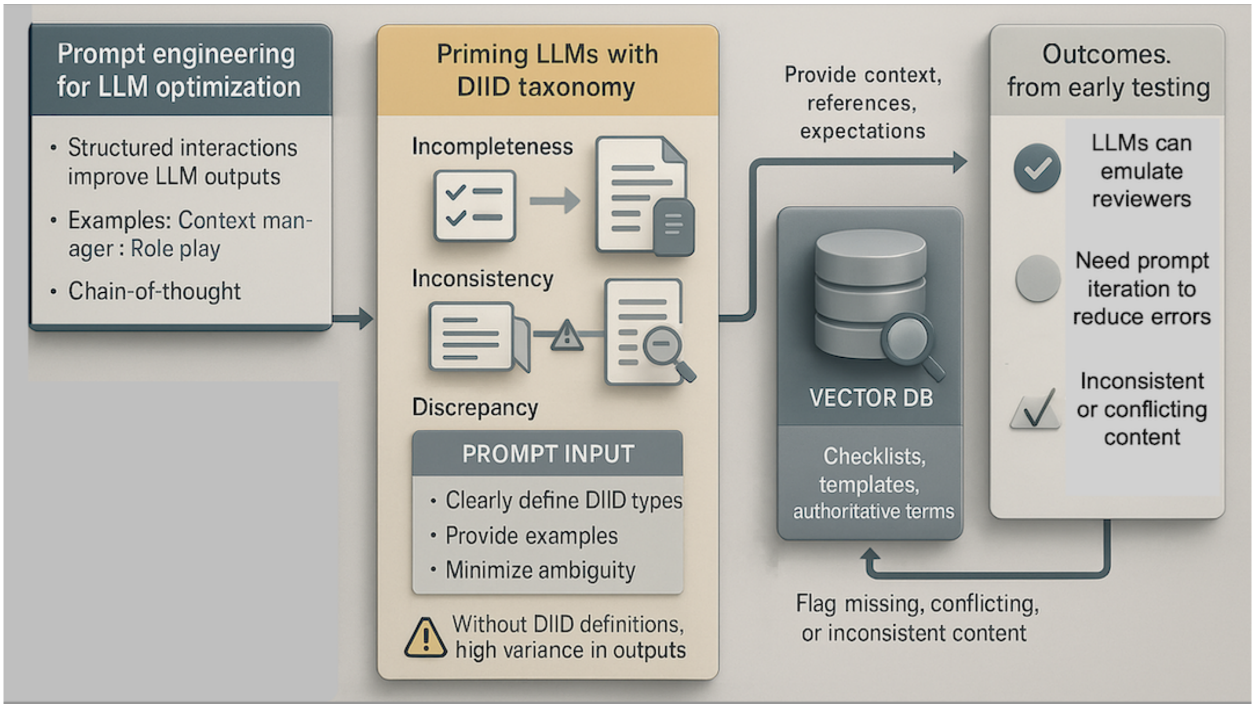
Prompt engineering for LLM optimization
• Structured interactions improve LLM outputs
• Examples: Context manager : Role play
• Chain-of-thought
Priming LLMs with DIID taxonomy
Incompleteness
Inconsistency
Discrepancy
PROMPT INPUT
• Clearly define DIID types
• Provide examples
• Minimize ambiguity
Without DIID definitions, high variance in outputs
Provide context, references, expectations
VECTOR DB
Checklists, templates, authoritative terms
Flag missing, conflicting, or inconsistent content
Outcomes. from early testing
LLMs can emulate reviewers
Need prompt iteration to reduce errors
Inconsistent or conflicting content

Figure 3 The Role of Prompt Engineering in DIID Detection

These complex tasks are where the combination of LLM + vector database shines. A vector database can supply the LLM with a checklist of expected content (for completeness checks) or the authoritative values/terms from reference documents (for discrepancy checks). An LLM, in turn, can combine this information and flag where the document deviates. By training or priming the LLM with plenty of examples of DIID issues (possibly from past project documentation annotated with findings), we can further improve its detection abilities. Early experimental setups indicate that LLMs can replicate the judgment of reviewers in identifying such issues, although careful evaluation and iteration on the prompts are required to minimize false positives and false negatives.

LLM Testing Frameworks for DIID Use Cases

Evaluating the effectiveness of LLMs on DIID detection requires specialized testing frameworks and data. Common LLM testing frameworks and benchmarks provide a starting point but often do not directly assess the kind of document analysis needed for regulatory compliance. For example, OpenAI's Evals toolkit allows users to create custom evaluations and has been used to measure GPT-4's performance on tasks ranging from math problems to code generation. For DIID detection on software acquisition, however, we need to design evaluations that mirror the task, e.g., given a set of documents and related regulations, does the LLM correctly identify all instances of DIID?

LLM testing protocols for detecting inconsistencies in document summaries could be extended because it provides tests specific to DIID and benchmarks for some of the latest LLMs, as described in (Laban et al., 2023). This work provides a relevant DIID detection test framework and dataset that considers multiple discrepancy examples, with multiple LLMs, and compares results of a few benchmarks. This current solution is limited to only a subset of the DIID detection needed for software acquisition, however, and without representation about levels of abstraction.

Figure 4 further describes the LLM testing framework for DIID. Using the test data set previously discussed, multiple LLMs are tested for DIID detection. Additional DIID tests with real systems are performed to ensure consistency of the DIID detection.

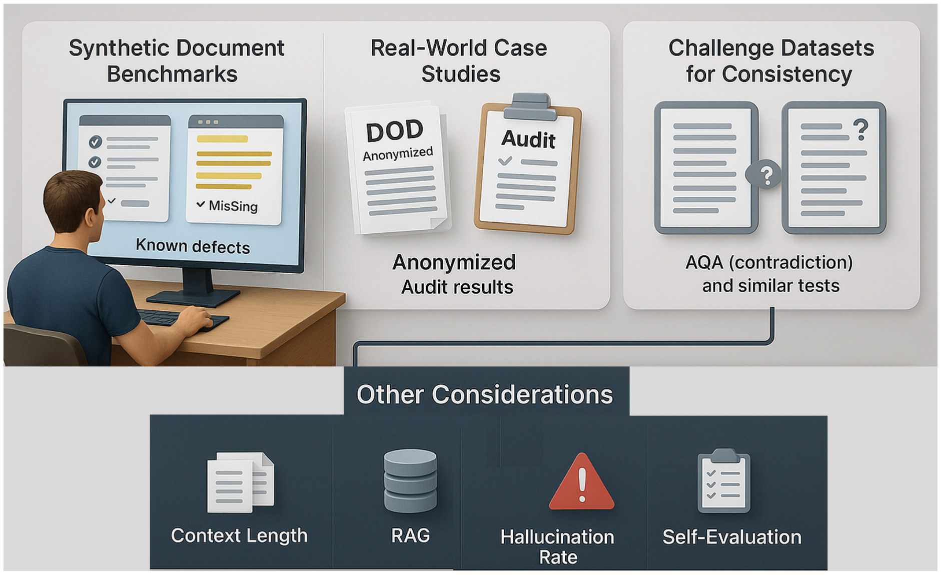

Figure 4 LLM Testing Frameworks for DIID Use Cases

Traditional natural language processing (NLP) benchmarks, such as MMLU (Massive Multitask Language Understanding) or BIG-bench, include reading comprehension and logical reasoning tasks, but they don't capture the multi-document, domain-specific nature of DIID detection. Similarly, academic benchmarks like Holistic Evaluation of Language Models (HELM) (Liang et al., 2023) provide broad assessments on accuracy, robustness, and fairness across tasks, but lack metrics for "document consistency checking" or "completeness verification" which are critical in our context. These gaps indicate that testing benchmarks should be extended to better consider DIID detection. Potential approaches to test LLMs for DIID include the following:

- *Synthetic document benchmarks* – Construct pairs of documents and regulatory checklists where we intentionally introduce known incompleteness or inconsistencies. For instance, create a requirement spec with one requirement missing a subsection, or two documents with slightly differing data for the same item. These synthetic cases (with ground-truth labels of where the issues are) can be used to evaluate LLMs quantitatively. Metrics like precision, recall, and F1-score for issue detection can be computed to indicate how many of the known missing sections did the model correctly flag (recall) versus how many flags were incorrect (precision).
- *Real-world case studies* – Use actual past project documents from DoD or industry , with appropriate anonymization, where known issues were found through audits. These documents serve as a gold standard to see if the LLM finds the same issues. This approach is more challenging because the model might find different issues than the human auditors. In these cases, additional analysis is needed to determine if the issues are valid or hallucinations. An evaluation framework must account for these differences by having experts review the AI-identified DIID issues.
- *Challenge datasets for consistency* – Leverage existing datasets focusing on contradiction detection or question answering consistency. For example, some academic tasks require a model to identify contradictions in text (similar to our inconsistency detection). Adapting such tasks (e.g., the BoolQ or contradiction questions from NLI benchmarks) could provide some insight into how well the LLM maintains consistency understanding. If an LLM struggles with basic contradiction identification in short texts, it will likely struggle with large documents too.

In testing current LLMs for DIID, limitations and challenges have been identified. One consideration is how to scale LLMs to support large documents or many documents, which is very common in DoD software acquisition and software engineering. RAG is a popular approach to scale LLMs for multiple documents and has proven both effective and relatively fast response but effectively implementing RAG systems requires some decision about how to chunk and store data (Godfrey, 2024). An alternate approach is to increase the context length and the trend has been for LLMs to increase context length to ~ 1M for current models. This larger context length simplifies application of LLMs to larger document sets because it does not require RAG. We are considering architectures with and without RAG, which complicates evaluation.

Another challenge is ensuring the model doesn't hallucinate. A powerful generative model might "imagine" a discrepancy that isn't there, especially if prompts are not tight. This type of hallucination is analogous to a false positive in testing. For reliable adoption, we prefer an LLM to miss a minor issue (false negative) rather than invent a problem (false positive) that sends teams on a wild goose chase. Testing frameworks must measure hallucination rates. One way is to include control documents that are error-free (or have no DIIDs) and then verify that the LLM mostly outputs "No issues found" for those. Any issues it does claim in a known-good document count as false alarms.

We also can examine the use of LLM self-evaluation and iterative refinement in testing. Some frameworks allow an LLM to evaluate its own answers or engage in a back-and-forth (e.g., asking the LLM to provide rationale and then verifying the rationale). For DIID, this self-evaluation could mean the LLM first lists what it thinks should be in the document, then checks off what's present. By evaluating how well this self-check correlates with actual document quality, we gauge the LLM's thoroughness.

Citing testing benchmarks and prior work, some LLMs have demonstrated human-level performance on various professional and academic benchmarks, suggesting its reasoning abilities are strong (Minaee et al., 2025). This finding gives optimism that LLMs can handle complex compliance tasks with the right prompts. DoD domain-specific testing with software artifacts is limited, however, and does not include specific testing of DIID. Early internal tests we constructed indicate that LLMs can catch subtle requirement inconsistencies (e.g., mismatched units or thresholds) that earlier models or simple scripts would miss. Conversely, some LLMs struggled unless the inconsistency was blatant, highlighting a performance gap. Such observations underscore the need for continued testing and prompt refinement, as well as possibly fine-tuning models on compliance data to improve their capabilities in DIID detection.

In summary, adapting LLM testing frameworks to DIID use cases involves creating realistic test scenarios and measuring detection performance carefully. These activities not only help quantify the effectiveness of current LLMs but also guide future improvements. For example, if we find that LLMs often miss certain types of discrepancies (e.g., terminology mismatches), we can then focus on training/prompt strategies to address those problems. In addition, if we find LLMs are flagging too many non-issues, we can adjust prompts so they are more conservative. Systematic testing and evaluation is essential for robust AI-augmented compliance in practice.

Quantifying the Benefits of LLM-Augmented DIID Detection

Beyond qualitative improvements, it is important to measure how LLM-augmented DIID detection impacts the software acquisition process. Prior research on humans detecting discrepancies in text provides some clues. For example, Schoor et al., (2023) found that when readers encountered conflicting information, it increased their "attention to sources," i.e., they spent extra effort cross-checking where the information came from. In the software acquisition DIID context, this suggests that an AI tool must not only flag an issue but also point analysts to the source evidence (e.g., the specific sections of the documents in conflict) to effectively aid and not hinder the human's workflow.

Figure 5 illustrates some of the potential benefits of DIID detection. Discussions with acquisition professionals and software engineers that review program artifacts for regulatory compliance have indicated that manual review is both exhausting and error prone, although quantifying the frequency of these issues is difficult. The benefits represented are based on discussions and will be used to define future testing.

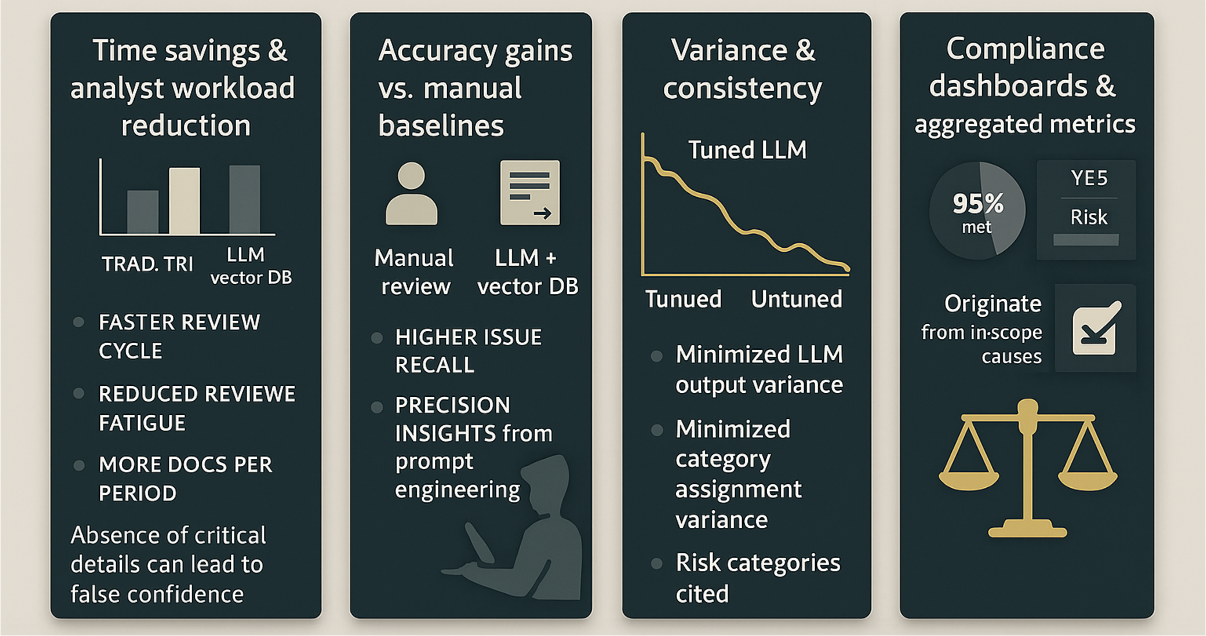
Time savings & analyst workload reduction
TRAD. TRI
LLM vector DB
FASTER REVIEW CYCLE
REDUCED REVIEWE FATIGUE
MORE DOCS PER PERIOD
Absence of critical details can lead to false confidence
Accuracy gains vs. manual baselines
Manual review
LLM + vector DB
HIGHER ISSUE RECALL
PRECISION INSIGHTS from prompt engineering
Variance & consistency
Tuned LLM
Tuned
Untuned
Minimized LLM output variance
Minimized category assignment variance
Risk categories cited
Compliance dashboards & aggregated metrics
95% met
YE5
Risk
Originate from in-scope causes

Figure 5 Measuring the Impact of AI-Augmented DIID Detection

We are incorporating the insights from prior work by designing our LLM's output to include references or quotes from the documents, so analysts can quickly validate the AI's findings. To evaluate the overall value proposition of LLM augmentation, we consider the following key questions and metrics:

- *Time savings & analyst workload reduction* – Does an LLM-assisted review significantly reduce the time and effort required by human analysts? We hypothesize that automating the tedious parts of the comparison will allow analysts to review more documents within the same time frame or to focus on deeper analysis of each document. Our hypothesis can be evaluated by controlled experiments where one group uses the LLM assistant and another follows traditional methods, measuring total person-hours spent. We are also exploring cognitive load indicators (perhaps via surveys) to see if LLM-based tools alleviate reviewer fatigue.
- *Accuracy gains compared to manual baselines* – Can the LLM + vector database approach match or exceed human performance in finding DIIDs, and with more consistency across different document types? We hypothesize that LLM-based tool will have higher recall (catch more true issues) than a typical manual review, especially for cross-document inconsistencies that humans might overlook. Precision (avoiding false flags) might initially be lower, but we expect the AI's precision to improve with effective prompt engineering and prompt patterns. We will measure this improvement by comparing the set of issues found by humans versus the AI in test scenarios.
- *Variance* – An interesting aspect is the variance of DIID detection in both human reviewers and LLMs. Human reviewers' thoroughness can vary widely and is not well quantified, but this issue is one of the reasons why regulatory compliance reviews include multiple reviewers. Given the same prompt and data, LLM output can vary, which is a common issue for generative AI solutions. We aim to demonstrate that the LLM output variance can be minimized, ensuring even challenging documents get a thorough check every time.
- *Enhanced traceability and new metrics* – A potential ancillary benefit of using an LLM for DIID detection is the traceability data it can generate. As the LLM links sections of policy to sections of project artifacts, it could produce a traceability matrix or graph as a by-product. For instance, if requirement *A* in a standard is fulfilled by section *X* of a design document, the LLM's analysis can record that link. Over time, this approach generates a graph of which policies map to which project artifacts. Such an output could be used to answer questions like "Have all policy clauses been addressed by the project's documents?" – providing an assurance metric for compliance coverage. We are exploring the extent to which an LLM's outputs can be aggregated into useful metrics, such as "percentage of requirements with at least one corresponding implementation evidence." This analysis goes beyond what manual reviews typically produce and could transform how compliance is reported by generating dynamic dashboards of compliance status.

In summary, our research will quantify the LLM's impact on efficiency (time/workload), effectiveness (issues detected vs missed), and augmented capabilities (like automatic traceability). Demonstrating concrete improvements in these areas is essential for gaining stakeholder buy-in, especially in DoD programs where any new tool must justify its adoption in terms of saved resources or improved outcomes

Future Research

Our initial results are promising, but we are investigating the following avenues to enhance and validate this approach further, as shown in Figure 6 and described below.

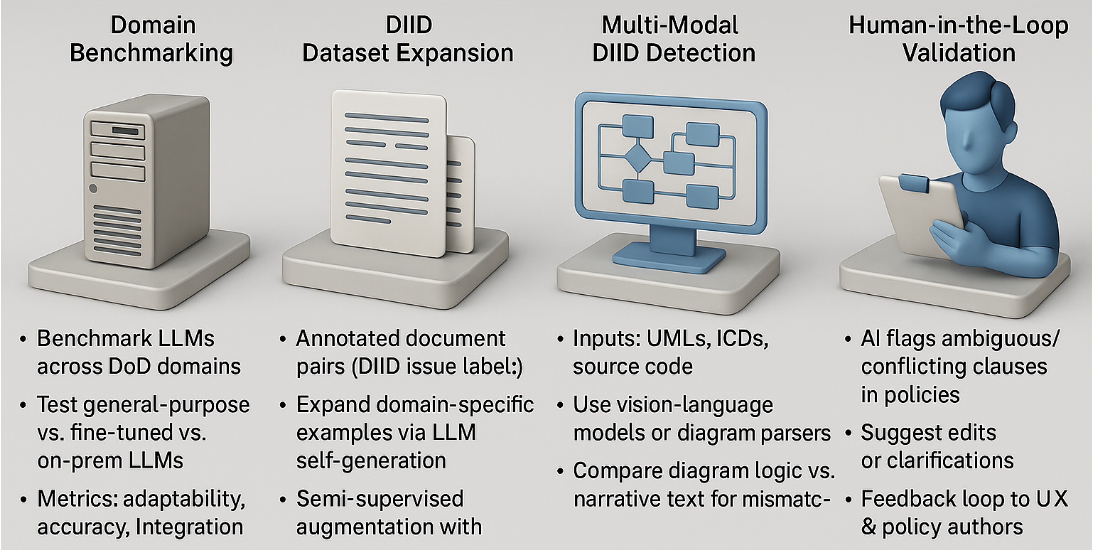
Domain Benchmarking
DIID Dataset Expansion
Multi-Modal DIID Detection
Human-in-the-Loop Validation
• Benchmark LLMs across DoD domains
• Test general-purpose vs. fine-tuned vs. on-prem LLMs
• Metrics: adaptability, accuracy, Integration
• Annotated document pairs (DIID issue label:)
• Expand domain-specific examples via LLM self-generation
• Semi-supervised augmentation with
• Inputs: UMLs, ICDs, source code
• Use vision-language models or diagram parsers
• Compare diagram logic vs. narrative text for mismatc-
• AI flags ambiguous/ conflicting clauses in policies
• Suggest edits or clarifications
• Feedback loop to UX & policy authors

Figure 6 Future Research Directions for AI-Augmented DIID Detection

1. *Benchmark LLMs across different software acquisition domains* – The DoD acquisition landscape is diverse, i.e., what works for a software-intensive weapons system might need adaptation for an IT business system or a healthcare system. We will test the DIID approach on documents from multiple domains (with their own terminologies and compliance standards) to evaluate the model's adaptability and identify if certain domains require specialized tuning or domain-specific training data. Likewise, we are experimenting with various LLMs (including newer models as they become available, or smaller fine-tuned models that could be deployed on-premises for security reasons) to see which offer the best balance of accuracy, context length, and ease of integration.

2. *Refine our DIID datasets and training resources* – A current limitation in our work is the lack of large, labeled datasets of DIID occurrences. We will therefore curate and release a dataset of annotated document pairs with known incompleteness, inconsistencies, and discrepancies. A DIID specific dataset will not only help us fine-tune models for better performance, but it can also benefit the research community by providing a benchmark for this problem. In addition, we might explore semi-supervised approaches: using the LLM itself to generate plausible DIID examples to augment training data (with careful validation to avoid reinforcing errors). We are currently extending existing software lifecycle data sets to include considerations of DIID, such as (Moreno et al., 2024).

3. *Incorporate multi-modal inputs like diagrams and code into the DIID detection process* – Many regulatory and technical documents include information that isn't pure text, e.g., architecture diagrams, interface control drawings, or source code snippets for critical algorithms. Our future work will examine multi-modal LLMs or pipelines to detect DIID issues across different media. For example, a UML diagram inconsistent with the written design description. Early exploration might involve feeding textual descriptions of diagrams into the LLM to see if it can relate them to the text.

4. *Rigorously quantify the benefits of LLM-augmented DIID detection in realistic settings* – We are conducting user studies where experienced acquisition professionals use the AI assistant on real tasks and provide feedback on workload reduction and confidence in the results. We will measure time saved, error rates reduced and collect qualitative feedback on how the tool affects their workflow. Such studies will be vital for refining the tool's usability and demonstrating ROI (return on investment) to decision-makers in the DoD.

5. *Drive changes in the regulatory documents themselves* – If AI tools struggle due to ambiguous or conflicting policy language, that signals an opportunity to improve the source materials. We foresee a feedback loop where the AI not only checks compliance but also suggests improvements to the policies and standards. For example, the LLM might frequently flag a particular paragraph in a policy as confusing or internally inconsistent; this could be reported to the policy authors, who can then clarify the text in the next revision. In fact, we have observed that with simple prompting, LLMs can recommend rewordings of requirements for clarity.

By adopting the approaches described above, organizations can create AI-friendly regulations that are easier for both machines and humans to parse. This co-evolution of policy and AI tooling – essentially writing requirements with automated checking in mind – is an intriguing area for future exploration. It aligns with the DoD's interest in more modular and unambiguous requirements and could further accelerate compliance activities. We will pursue research into how best to formulate guidance for policy authors, potentially in collaboration with DOD standards bodies.

In summary, our future work will scale up our current DIID research to cover more domains and data types, systematically evaluate its impact, and explore the bidirectional relationship between AI and policy (e.g., to determine how each can inform improvements in the other). We believe these steps are crucial to transition our research into a practical capability for the DoD and beyond.

Concluding Remarks

This paper discussed accelerating regulatory compliance checks in software acquisition by leveraging LLMs. We defined document incompleteness, inconsistencies, and discrepancies, and demonstrated how an LLM, guided by prompt engineering, can effectively assist in detecting DIID. Our approach has the potential to reduce the manual burden on acquisition professionals significantly, increase the consistency and thoroughness of compliance reviews, and ultimately help DoD programs deliver capability faster without sacrificing rigor.

Our preliminary exploration indicates that even with current LLM technology, there is improvement in DIID detection accuracy and efficiency when compared to traditional methods. There remain challenges to address, including creating test frameworks and data sets to ensure the LLM's outputs are trustworthy and consistent, to tailoring the solution for different domains and document types. Ongoing work is tackling these through targeted evaluation and iterative refinement.

Overall, the integration of human expertise with LLM capabilities is a powerful new paradigm. This human-AI collaboration can not only speed up the acquisition process but also improve its outcomes by catching issues early and iteratively with different software releases. As generative AI tools mature, and as organizations adapt their practices (and possibly their documentation standards) to better accommodate AI assistance, we anticipate that LLM-augmented compliance checking will become an invaluable part of software engineering researchers and practitioners. By continuing to evaluate and improve our approach in realistic settings, and by sharing insights with the community, we aim to move one step closer to a future where compliance is maintained more proactively and efficiently.

Acknowledgements

Copyright 2025 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

DM25-0544

References

Bozkurt, A. (2024). Tell Me Your Prompts and I Will Make Them True: The Alchemy of Prompt Engineering and Generative AI. *Open Praxis*, *16*(2). https://doi.org/10.55982/openpraxis.16.2.661

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020). Language models are few-shot learners. *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 1877–1901.

Cao, T., Raman, N., Dervovic, D., & Tan, C. (2024). *Characterizing Multimodal Long-form Summarization: A Case Study on Financial Reports* (No. arXiv:2404.06162). arXiv. https://doi.org/10.48550/arXiv.2404.06162

Godfrey, C. (2024, November 22). *RAG in the era of long-context LLMs*. Medium. https://medium.com/tr-labs-ml-engineering-blog/rag-in-the-era-of-long-context-llms-b8ecda2d5693

John Robert & Douglas Schmidt. (2024, January 22). *10 Benefits and 10 Challenges of Applying Large Language Models to DoD Software Acquisition*. https://insights.sei.cmu.edu/blog/10-benefits-and-10-challenges-of-applying-large-language-models-to-dod-software-acquisition/

Laban, P., Kryściński, W., Agarwal, D., Fabbri, A. R., Xiong, C., Joty, S., & Wu, C.-S. (2023). *LLMs as Factual Reasoners: Insights from Existing Benchmarks and Beyond*. https://doi.org/10.48550/ARXIV.2305.14540

Liang, P., Bommasani, R., Lee, T., Tsipras, D., Soylu, D., Yasunaga, M., Zhang, Y., Narayanan, D., Wu, Y., Kumar, A., Newman, B., Yuan, B., Yan, B., Zhang, C., Cosgrove, C. A., Manning, C. D., Re, C., Acosta-Navas, D., Hudson, D. A., … Koreeda, Y. (2023). Holistic Evaluation of Language Models. *Transactions on Machine Learning Research*. https://openreview.net/forum?id=iO4LZibEqW

Liu, P., Zhang, L., & Gulla, J. A. (2023). *Pre-train, Prompt and Recommendation: A Comprehensive Survey of Language Modelling Paradigm Adaptations in Recommender Systems* (No. arXiv:2302.03735). arXiv. https://doi.org/10.48550/arXiv.2302.03735

MIL-STD-882E. (2023). https://quicksearch.dla.mil/qsDocDetails.aspx?ident_number=36027

Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., & Gao, J. (2025). *Large Language Models: A Survey* (No. arXiv:2402.06196). arXiv. https://doi.org/10.48550/arXiv.2402.06196

Moilanen, T., Sivonen, M., Hipp, K., Kallio, H., Papinaho, O., Stolt, M., Turjamaa, R., Häggman-Laitila, A., & Kangasniemi, M. (2022). Developing a Feasible and Credible Method for Analyzing Healthcare Documents as Written Data. *Global Qualitative Nursing Research*, *9*, 23333936221108706. https://doi.org/10.1177/23333936221108706

Moreno, G., Hristozov, A., Robert, J., & Klein, M. (2024, July 23). *A Model Problem for Assurance Research: An Autonomous Humanitarian Mission Scenario*. https://insights.sei.cmu.edu/library/a-model-problem-for-assurance-research-an-autonomous-humanitarian-mission-scenario/

Prasad, N., Boughanem, M., & Dkaki, T. (2024). *Exploring Large Language Models and Hierarchical Frameworks for Classification of Large Unstructured Legal Documents* (No. arXiv:2403.06872). arXiv. https://doi.org/10.48550/arXiv.2403.06872

RAG for LLMs – Nextra. (n.d.). https://www.promptingguide.ai/research/rag#rag-evaluation

Robert, J., Ivers, J., Schmidt, D. C., Ozkaya, I., & Zhang, S. (2024). The Future of Software Engineering and Acquisition with Generative AI. *Crosstalk*, *AI Taming the Beast*, 26–43. https://community.apan.org/wg/crosstalk/m/documents/464157

Schmidt, D. C., & Robert, J. (2024, April 1). *Applying Large Language Models to DoD Software Acquisition: An Initial Experiment*. Carnegie Mellon University, Software Engineering Institute's Insights (Blog). https://doi.org/10.58012/s32x-gc46

Schoor, C., Rouet, J.-F., & Britt, M. A. (2023). Effects of context and discrepancy when reading multiple documents. *Reading and Writing*, *36*(5), 1111–1143. https://doi.org/10.1007/s11145-022-10321-2

Shokrollahi, Y., Yarmohammadtoosky, S., Nikahd, M. M., Dong, P., Li, X., & Gu, L. (2023). *A Comprehensive Review of Generative AI in Healthcare*. https://doi.org/10.48550/ARXIV.2310.00795

Shukla, N. K., Katikeri, R., Raja, M., Sivam, G., Yadav, S., Vaid, A., & Prabhakararao, S. (2023). Investigating Large Language Models for Financial Causality Detection in Multilingual Setup. *2023 IEEE International Conference on Big Data (BigData)*, 2866–2871. https://doi.org/10.1109/BigData59044.2023.10386558

Stephany Bellomo, Shen Zhang, James Ivers, Julie B. Cohen, & Ipek Ozkaya. (2023). *Assessing Opportunities for LLMs in Software Engineering and Acquisition*. Carnegie Mellon University Software Engineering Institute. https://doi.org/10.58012/m3hj-6w28

Weapon Systems Annual Assessment: Challenges to Fielding Capabilities Faster Persist | U.S. GAO. (2022). GAO. https://www.gao.gov/assets/gao-22-105230.pdf

White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., & Schmidt, D. C. (2023). *A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT* (No. arXiv:2302.11382). arXiv. https://doi.org/10.48550/arXiv.2302.11382