

Skill-Governed Product Management Workflows for Constraining Generative Documentation Output and Their Transferability

Navtanay Sinha^{a,*}

a: Principal Product Manager, Santa Clara, CA 95054, USA

* Corresponding author: navtanaysinha@gmail.com

Accepted: 8 August 2026 | Published: 9 October 2026

Abstract

Product documentation carries the judgment of its author, including which claims may stand without supporting evidence and which priorities outrank others. Generative language models supply fluent prose, while the standards that make such documents defensible have stayed in the memory of the writer. This study examines a working product management system in which those standards exist as rule files that the model reads at drafting time. The purpose is to describe how a written rule turns into a constraint on generated output and what must hold for the arrangement to serve another practitioner. The method is a descriptive single-case review of fifteen artifacts from one system, including seven rule files currently in use, six earlier versions of the same material, a development journal, and an interface screenshot. Analysis combined structural decomposition of the corpus, paired comparison of earlier and current versions, and chronological extraction of decisions, checks, and one documented defect. Four findings emerge. Composition of several rule files into one instruction converts editorial standards into constraints that hold over every document produced afterward, provenance becomes a stored relation, the share of judgment withheld from the model is written down and stays stable, and transfer preserves document structure and evidence constraints while discarding domain context, organizational apparatus, and invocation metadata. The account states what a practitioner must write down before a documentation workflow can be governed by its author.

Keywords: document generation; generative artificial intelligence; human oversight; knowledge externalization; large language models; product management; prompt composition; workflow governance.

1. Introduction

A product requirements document, a one-page proposal, and a press release paired with frequently asked questions all serve a purpose beyond communication. Each one records decisions about which customer problem deserves engineering time, which assertions may appear without a source, and how a team should sequence commitments it cannot reverse. Fluency contributes little to that purpose. Whether a document survives scrutiny depends on the standard applied while it was written, and that standard has lived with the practitioner.

Before the arrangement described here, every document began from a blank page or from a copy of an earlier one, and consistency depended on the author recalling personal standards at each sitting. Customer signals sat in meeting notes, in the customer relationship system, and in a stack of product documents, with no link to the roadmap. Writing a requirements document meant returning to the meetings and deriving priorities again. Provenance, where it existed at all, was reconstructed after the fact. The judgment was real and unavailable to anyone else, including any tool.

Language models have compressed the distance between an idea and a finished draft. A preregistered experiment on mid-level professional writing found that access to a chatbot reduced completion time by 40% and increased quality ratings by 18% [1]. The same class of system produces assertions with no grounding in any source, a failure mode detailed for information systems [2]. In documents that direct

capital and engineering effort, an invented customer name or a fabricated revenue figure does damage that outlives the draft.

Practice has answered with prompting. Surveys of prompting methods describe how an instruction shapes what a model produces [3], and prompt patterns have been gathered into reusable catalogs for software engineering tasks [4]. Such catalogs are written by researchers for the general case. The standard that a senior practitioner applies to a requirements document is narrower, tied to one market, one product line, and a set of habits accumulated over years of review.

Research in knowledge management treats generative systems as instruments that change how tacit understanding turns explicit [5], and interview work with software developers documents shifts in knowledge creation and sharing once these tools enter daily practice [6]. Adjacent literature in requirements engineering has surveyed how generative models support elicitation, classification, and specification [7], and one study has reformulated models to score requirements against predefined quality criteria [8]. Those criteria originate outside the practitioner. Less attention has been paid to the arrangement in which the criteria belong to the practitioner, sit in files that the practitioner edits, and constrain the draft while the model writes.

Three questions organize this study. The first asks what form a written rule must take to serve as a constraint on generated output. The second asks what follows from composing several such rules into a single instruction during drafting. The third asks which conditions must hold for the same arrangement to serve another practitioner working in another domain. To the knowledge of the author, no comparable description of a practitioner-authored rule corpus and its passage into a second environment has appeared in the available literature.

The system under study was built by its author, a product manager working in enterprise infrastructure. Its rule corpus grew out of documents that already existed in the author's practice before any tool read them, which makes the case suitable for examining what survives when private standards are made machine-readable.

2. Materials and Methods

2.1. Study Design

The design is a descriptive single-case review. The author of the system is also the analyst, a position that carries the limits treated in Section 4.4. No experiment was run, no control condition existed, and no measurement of document quality was done by offline review with peers. Figures appearing in the development journal, including token counts and per-document cost, are entered into this account as observations recorded during construction and do not constitute a claim of measured efficacy.

2.2. Corpus

Fifteen artifacts collected on August 20 and August 25 of 2026 form the corpus. Seven rule files are currently in use within the application, covering an operating manual, an author profile, three document generators, a prose filter, and a statement of prioritization philosophy. Six are earlier versions of the same material as it stood before preparation for reuse. One is a development journal of about 4,200

words recording locked decisions, a capability check, a gap analysis, and sprint-level execution logs. One is a screenshot of the running interface.

The pairing across the six earlier versions is uneven. Four files entered the application with their bodies unchanged, one was rewritten to strip domain specifics, and one was replaced by a document assembled from a different template. Section 3.5 treats that unevenness as a result of the study.

2.3. Analytical Procedure

Three lines of analysis were applied. The first decomposed each rule file by the type of constraint it carries, separating document structure, prose style, admissibility of evidence, prioritization, and the stance the model takes toward its author. The second step placed each earlier version alongside its counterpart in the application and coded each passage as carried over, removed, or added. The third read the development journal in chronological order and extracted decisions with their stated reasoning, one capability check with an explicit confidence rating, and one defect together with its correction and verification.

2.4. Disclosure and Sanitization

Names of employers, products, internal programs, and staff were removed from the material at the request of the artifact owner. Structure was preserved throughout the removal. The sanitization limits what an outside reader can verify, and Section 4.4 treats that limit alongside the others.

3. Results

3.1. The Anatomy of a Written Rule

A rule in this system consists of a name and a body that the model reads. Nothing began as a configuration. Each file started as a document the author had already written for human readers, and each was imported into the application, where it can be edited in place.

Five types of constraints appear across the corpus. Structural rules fix the order and content of sections, as in the eight-section arc of the requirements generator. Style rules govern prose. The prose filter, an adopted component authored by Hardik Pandya and released under a permissive license, forbids the default use of the em dash, flags three consecutive sentences of matching length, and closes with a five-dimensional score and a threshold below which text is returned for revision. Evidence rules govern what may be asserted. The operating manual forbids inventing customer names, company names, metric values, anecdotes, and quotations, and directs the model to mark a gap where customer evidence is missing. Prioritization rules record how the author weighs value against effort, with irreversibility and strategic alignment layered above that comparison, and they name the frameworks the author declines to treat as decision procedures. Stance rules govern how the model addresses its author. One principle obliges the model to identify structural weakness in a plan and to withhold soft validation, with review sharpening ahead of high-stakes situations.

Prohibition carries more of the corpus than instruction. Rules phrased as prohibitions outnumber those phrased as directions, and they concentrate in the two areas where the author has the most exposure, namely, evidence and prose. Only one file in the corpus states a numeric acceptance threshold for its own output.

3.2. Composition at Drafting Time

The application never asks the model to supply expertise. At drafting time, it gathers the rule files the author has enabled, stitches them into one system instruction, and the model writes inside that

instruction. Figure 1 shows the two paths by which material reaches the instruction and what the resulting document carries.

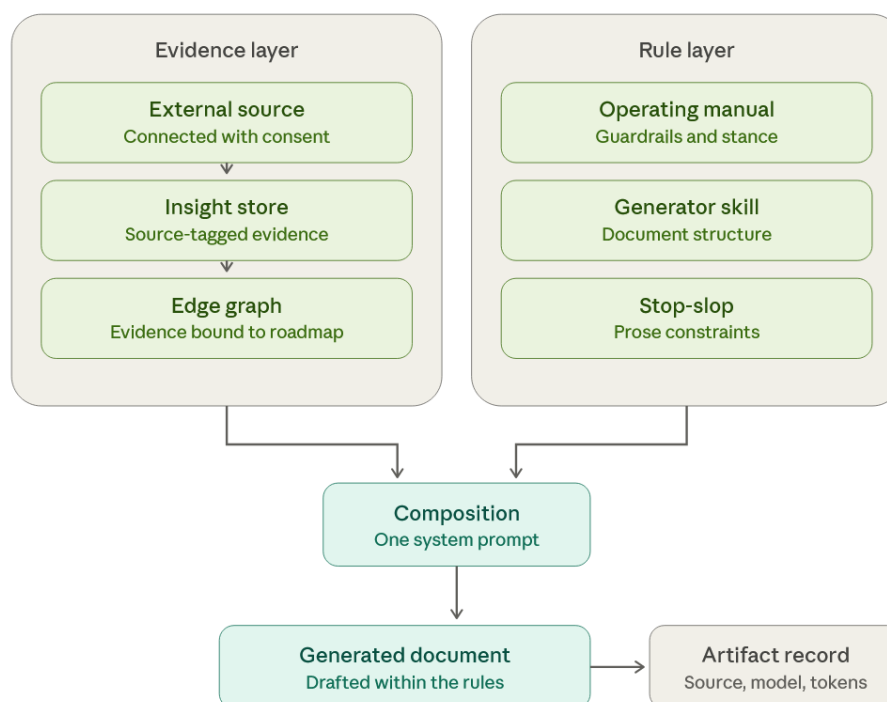


Figure 1. Evidence layer and rule layer converging at drafting time

Evidence passes through three transformations before it reaches the prompt. It enters from an external source authorized by the author, is stored as a record tagged with the tool that produced it, and is bound to a roadmap item through a typed relation. Rules travel a shorter path and arrive in parallel, with no hierarchy among them beyond the order of assembly. The resulting document contains a link back to the roadmap item it references, and its record stores the model identifier and the token counts consumed.

Two consequences follow. The first is editability. A single line change in the prose filter alters every document drafted afterward, converting a standard from something recalled into something applied. The journal marks the point at which the corpus began to drive the prompt: input tokens for one document increased from 787 to 3,700 once the imported rule files replaced a fixed instruction. The same entry places the cost of that document at nearly 0.6 cents on the application's cheapest model tier, a figure the author recorded during construction. The second consequence is cost. Governance is paid for in input tokens, and the price scales with the size of the active corpus, which places a ceiling on how many rules a practitioner can keep enabled at once. Prompting research has treated instruction as an object that an engineer designs for a task [3], and the arrangement here shows what happens when the same object is assembled at runtime from documents the practitioner maintains.

3.3. Evidence and Its Boundary with Instruction

Every claim in a generated document points back to a stored record. The press release generator carries the single exemption. Two quotations, one attributed to a company leader and one to a customer, may be synthesized from the evidence on hand, and each must be marked as illustrative so that no reader takes it for a statement anyone made. Everywhere else, the prohibition holds without qualification. The design addresses the failure mode identified in information systems, where plausible text lacking a factual basis is the dominant risk [2]. The running interface reports the state of that store, showing eleven

stored records at the moment of capture, of which four carried a link to a roadmap item and seven awaited triage.

External material entering a prompt occupies an uneasy position between data and instruction, and that ambiguity has been shown to open a real attack surface in applications that retrieve content at inference time [9]. The journal records that the connected meeting source wraps every note it returns in an explicit warning to treat the content as data, and the application inherits that posture.

The same boundary produced the one defect that the journal documents in full. Evidence reached the model, with its source label in brackets before the text. The model read the bracketed label as the name of a customer company and wrote it into the output as such. The correction replaced the label with an explicit phrase naming the source as a set of meeting notes, and a later run confirmed that the source no longer surfaced as a customer in either of the two document types reusing the generation path. The rule forbidding invented customers was correct throughout. What failed was the presentation of context around it, which locates the boundary between evidence and instruction as an open design surface.

A related constraint appeared during the capability check, which the author closed with a stated confidence of 7.5 on a ten-point scale. The meeting source returns structured summaries at its free tier and reserves verbatim transcripts for a paid tier. Summary-level evidence supports extraction and linking. Exact quotation requires the higher tier. The prohibition on invented quotations and the tier of data access therefore describe one limit approached from two directions, since a system without transcript-level evidence has no legitimate path to a real customer quotation.

3.4. The Boundary of Automation

A share of the work was intentionally withheld from the model, and the withholding is documented. Table 1 sets out six decision points, what the system does at each, what the author retained, and the file in which the rule is recorded.

Table 1. The boundary of automation in the observed system

Decision point	What the system does	What the author retained	Where the rule is recorded
Standard of quality	Drafts inside the written standard supplied to it	Definition of what a good document is	Operating manual
Evidence	Reuses stored records and names the source tool	Prohibition on invented customers, metrics, and quotations	Operating manual and every generator
Illustrative quotations	Produces two, each marked as illustrative	Scope of the single exemption	Press release generator
Prioritization	Proposes a value against effort score	The ranking decision	Prioritization philosophy
Data source access	Requests authorization and waits	Consent, granted per source	Connection flow and locked build decisions
Irreversible actions	Runs none without instruction	Sending and publishing	Operating principle

Two properties of this boundary separate it from a review step appended after generation. The first is timing. Each rule takes effect while the draft is produced. Analysis of oversight requirements imposed on public-sector algorithms has shown that a person placed in the loop often cannot perform the function the policy assumes, and that formal oversight can lend legitimacy to a system while providing little protection [10]. Written constraints applied at generation time do not dissolve that concern. They move where the burden falls, since the author states the standard before any output exists, and cannot be carried by fluent text into approving a claim with no source behind it.

The second property is asymmetry. Autonomy is granted over form and withheld over consequence. The operating manual directs the model to act on its own when a request is clear and to skip clarifying questions. The same corpus requires explicit authorization before connecting to a data source and forbids

unattended execution of anything the author cannot take back. Independence is wide inside the draft and closes at the point where an action leaves the workspace.

One asymmetry inside the corpus supports the same reading. The prioritization file is the least-finished document among the seven. It carries bracketed placeholders where worked examples belong, and it closes with a list of what still needs to be written. It also governs the one decision the author declined to delegate. The degree of formalization tracks the willingness to hand a decision to a model.

3.5. What Transfer Preserves and What It Discards

The six earlier versions supply the material for the third question. Figure 2 summarizes what became of a rule file as it moved from the author's working environment into the application.

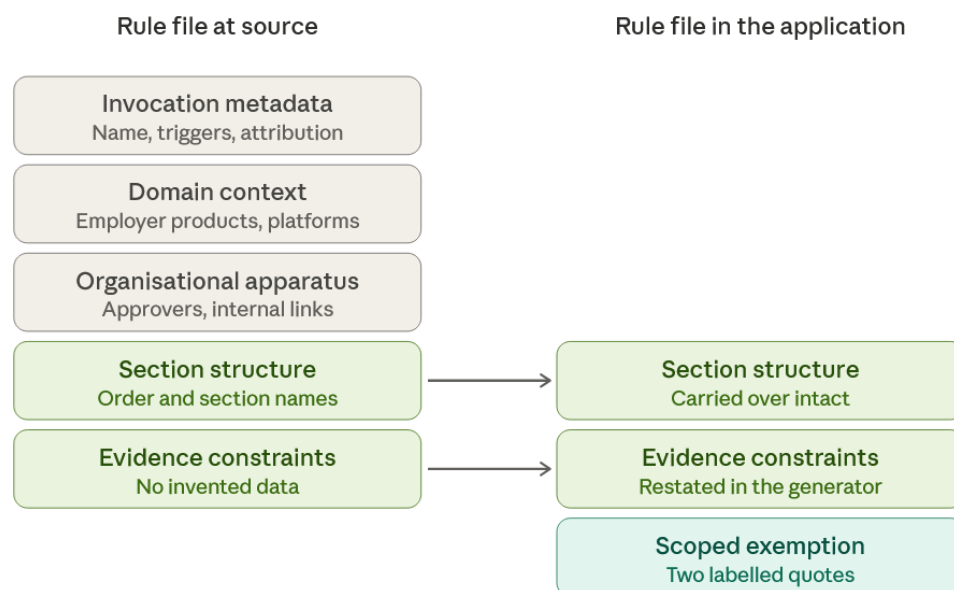


Figure 2. Layers of a rule file that survive transfer into the application

Four of the six files entered with their bodies unchanged. The operating manual needed no editing at all, since it named no employer or product, and its counterpart in the application is longer than the original, having gained a section on interview preparation. That added section directs the model toward skills the application never received, leaving the governing document pointing to capabilities absent from the product and showing that a rule corpus carries a maintenance burden of its own. One file, the requirements generator, was rewritten to remove domain specifics. Its word count fell from 849 to 421, and the order and the names of its eight sections survived intact. One file, the press release generator, was replaced outright. The original encoded a phase-gate procedure belonging to one employer, complete with a table of approvers and a program scorecard. Nothing of the practitioner's judgment remained once the organization was stripped from it, so a document built on the public working backward form took its place. Three further files in the author's collection were never ported, having been judged personal career tools with no place in a product.

Removals from the rewritten generator fall into three classes. Domain context went first, covering product names, platform behavior, protocol coverage, and sector regulation. Organizational apparatus followed, covering approver roles, links to internal trackers, and a list of competitive losses. Invocation metadata came last, covering the name, description, trigger phrases, and non-trigger phrases that allowed an agent to select the file on its own. The third class vanished from every imported file without exception

because the application asks the author to choose a document type via an interface, leaving the selection logic with no work to do.

That deletion carried a side effect worth recording. Authorship of the adopted prose filter, the link to its public repository, and its license all reside within the metadata block that the importer strips. A system whose purpose includes provenance lost the provenance of its only borrowed component at the moment of import. The attribution was recovered from the original file for this account.

Transfer also added. The requirements generator gained an explicit prohibition on invented revenue figures, customer names, and metrics, a rule that had lived only in the operating manual before. The exemption permitting two labeled quotations appears for the first time in the version inside the application. Preparing the corpus for a second reader obliged the author to state a boundary that had been held by habit.

4. Discussion

The four results answer the three questions posed in the introduction and carry a meaning that extends past the single system examined. Composition at drafting time answers the first question by showing that a rule becomes operative when it exists as text the model reads before it writes, which places the standard ahead of the output in time and removes the need for the author to recall it. The stored provenance relation answers the second question by showing what a composed instruction leaves behind, since a document that records the roadmap item it serves and the evidence bound to that item can be audited without reconstructing its origin. The stability of the withheld share and the pattern of what survives transfer together answer the third question, because a boundary written into files travels with those files, while the domain knowledge around it does not. Read together, the four results describe a division of labor in which the model supplies fluency and the practitioner supplies the standard, and they locate the cost of that division in the input tokens consumed at every draft.

4.1. Conditions for Transfer

The pattern in Figure 2 identifies what is portable. Section order, the separation of business justification from requirements, the demand that metrics measure outcomes, and the placeholder convention standing in place of an invented number all crossed without modification. Product knowledge and organizational procedure did not cross. A practitioner adopting this arrangement inherits a shape and supplies the content.

Four conditions follow. The first is that the practitioner either holds written standards or can articulate them, since a rule that cannot be stated cannot be formulated as an instruction. The second is that an evidence store exists to which claims can be bound, absent which the prohibition on invention has nothing to point at. The third is a willingness to keep prioritization in human hands, because the file governing it is the least formalized in the corpus and would fail as an automated scorer. The fourth is a habit of editing, since a rule corpus left untouched drifts away from the practice it claims to describe.

4.2. Relation to Prior Work

Prompting research describes the instruction as an object shaped by an engineer for a task [3], and pattern catalogs generalize such instructions for reuse across projects [4]. The corpus examined here inverts the authorship. The practitioner writes the rule, owns it, and revises it; the model has no standing to override it. Work on requirements engineering with generative models has assessed output against externally defined quality criteria [8]. The criteria in this case belong to one person and travel with that person.

Knowledge management theory frames generative systems as instruments altering the passage from tacit to explicit understanding [5], and empirical work with software developers reports changes in how

knowledge is created and shared once such systems enter daily use [6]. The case adds a mechanism to that account. Externalization was never an aim in itself here. It became a condition of using the tool at all, since a model can apply only a standard that exists as text.

Three bodies of prior work bound the contribution more precisely when their methods are placed beside the one used here. The systematic review of generative models in requirements engineering covers elicitation, classification, and specification, and its unit of analysis is the task performed on a requirement, with quality judged against criteria an evaluator brings to the artifact after it exists [7]. The catalog of prompt patterns for software engineering abstracts recurring instruction shapes into reusable solutions, and its unit of analysis is the pattern detached from any author [4]. Work on indirect prompt injection treats retrieved content as a channel through which instruction enters a system without the consent of its operator, and its unit of analysis is the attack path [9]. The present case shares the artifact common to all three, namely a textual instruction placed ahead of generation, and differs from them in ownership and timing. The instruction is authored by the practitioner whose judgment it encodes, it is versioned as a document and holds no status as a tuned parameter, and it constrains the draft at the moment of writing, which is the point these three literatures leave unoccupied. The survey of hallucination supplies the reason the arrangement matters, since a taxonomy of ungrounded output establishes that the prohibition carried in these files addresses a failure mode the literature has documented [2].

4.3. Deliberate Non-Automation

Debate about human oversight has centered on whether a person can perform the checking that a policy assigns [10]. The arrangement here shifts the question. The author writes the standard first, and the model operates inside it. Responsibility for the standard stays with the author, and the tool absorbs the friction of applying it. Whether the shift produces better documents remains untested. What can be stated is that the division of labor is explicit and inspectable, since every constraint sits in a file the author can open.

4.4. Limitations

The study rests on a single case, and the system's author produced the analysis. The comparison is against manually authored product documents, which takes weeks to achieve similar quality instead of days, which is the single most valuable experiment the arrangement makes available. Document quality was assessed against no instrument. Controlled work on generative assistance in professional writing shows what such an assessment demands, including randomization, task incentives, and blind quality rating [1], and nothing of that kind was attempted here.

The corpus was sanitized before analysis, which limits what a reader can verify. Observations were gathered over a short period on one price tier of one model family. Coverage of the system is partial. One external source was connected and in use at the time of the account, while two more remained planned. Semantic search over the evidence store was implemented and left inactive for want of a provider key, and multi-user access with commenting was not built. Figures the author reports about token consumption and per-document cost are entered as statements of the author, with no measurement protocol behind them.

The constraints on this account fall into four classes that bound what a reader may take from it. Scope is the first. The study describes one system, built and operated by one practitioner in enterprise infrastructure, and the corpus spans fifteen artifacts gathered on two dates, so nothing here establishes frequency, typicality, or distribution across practitioners. Construct validity is the second. Token counts and per-document cost measure consumption of a model service and stand at a distance from document quality, which no instrument in this study assessed. Internal validity is the third. The author designed

the system, selected the artifacts, and performed the coding, and no second coder verified the assignment of passages to the categories reported in Section 3.5, which leaves the classification open to the expectations of the person who built the system under classification. External validity is the fourth. The portable layer identified in Section 4.1 was derived from one corpus in one professional domain, and the four conditions stated there await a test and hold no status as findings. The account therefore supports a description of a mechanism and its boundary, while the question of how far that mechanism generalizes belongs to the studies named in Section 4.5.

4.5. Future Work

Three studies follow from this account. A second practitioner in an unrelated domain could take the portable layer and report what had to be rewritten. A within-subject comparison could hold the roadmap item constant, vary the corpus, and submit both drafts for blind review. Longitudinal observation could establish whether a rule corpus remains current across a change of employer, which is where the line between portable structure and local context would be tested most sharply.

5. Conclusion

A written standard becomes an operative constraint when it exists as a file that the model reads at the moment of drafting. The composition of several such files into a single instruction carries editorial judgment into every document produced afterward, and a single edited line propagates without further effort by the author. The price is paid in input tokens, which bounds how large an active corpus can grow.

Provenance in this system is a stored relation among a document, the roadmap item it serves, the evidence bound to that item, and a record of the model and the tokens that produced it. The prohibition on invented data is the one rule replicated across every file in the corpus, and its single exemption is narrow and labeled. The boundary between evidence and instruction proved to be the place where a real defect appeared and was corrected, which marks that boundary as a design surface for anyone building comparable workflows.

Transfer preserves the shape of a document and the constraints on its evidence. Transfer discards the domain context, organizational procedure, and the metadata, allowing an agent to select a file on its own. The degree to which a rule has been formalized reflects the author's willingness to let a model act on it, and the least-finished file in the corpus governs the decision the author declined to delegate. What a practitioner must write down before a documentation workflow can be governed by its author is the shape of the document, the limits on its claims, and the list of decisions that stay in human hands.

Declarations

Funding. This research received no external funding.

Competing interests. The author designed, built, and operates the system described in this study and also conducted the analysis reported in the manuscript. This dual role is disclosed; the author declares no other competing interests.

Author contributions. N.S. conceived the study, designed and built the system, curated and analyzed the artifact corpus, and wrote and approved the final manuscript.

Data availability. The study is based on the fifteen artifacts described in Section 2.2. The materials were sanitized at the artifact owner's request; consequently, the underlying artifact corpus is not publicly available. Code availability. Not applicable to the reported descriptive artifact analysis.

References

- [1] S. Noy and W. Zhang. “Experimental Evidence on the Productivity Effects of Generative Artificial Intelligence.” *Science*, vol. 381, no. 6654, pp. 187-192, Jul. 2023. doi: 10.1126/science.adh2586.
- [2] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, and T. Liu. “A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions.” *ACM Transactions on Information Systems*, vol. 43, no. 2, art. 42, pp. 1-55, Mar. 2025. doi: 10.1145/3703155.
- [3] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig. “Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing.” *ACM Computing Surveys*, vol. 55, no. 9, art. 195, pp. 1-35, Sep. 2023. doi: 10.1145/3560815.
- [4] Y. Sasaki, H. Washizaki, J. Li, D. Sander, N. Yoshioka, and Y. Fukazawa. “Systematic Literature Review of Prompt Engineering Patterns in Software Engineering,” in *Proc. IEEE 48th Annual Computers, Software, and Applications Conference*, Osaka, Japan, 2024, pp. 670-675. doi: 10.1109/COMPSAC61105.2024.00096.
- [5] M. Alavi, D. E. Leidner, and R. Mousavi. “A Knowledge Management Perspective of Generative Artificial Intelligence.” *Journal of the Association for Information Systems*, vol. 25, no. 1, pp. 1-12, 2024. doi: 10.17705/1jais.00859.
- [6] K. Kirchner, E. Bolisani, T. C. Kassaneh, E. Scarso, and N. Taraghi. “Generative AI Meets Knowledge Management: Insights From Software Development Practices.” *Knowledge and Process Management*, vol. 32, no. 4, pp. 223-235, 2025. doi: 10.1002/kpm.70004.
- [7] P. Vasudevan and S. Reddivari. “The Role of Generative AI Models in Requirements Engineering: A Systematic Literature Review,” in *Proc. 2025 ACM Southeast Conference*, Cape Girardeau, MO, USA, 2025, pp. 188-194. doi: 10.1145/3696673.3723053.
- [8] C. Ellsel and R. Stark. “Advancing Requirements Engineering with Large Language Models.” *Procedia CIRP*, vol. 136, pp. 701-706, 2025. doi: 10.1016/j.procir.2025.08.120.
- [9] S. Abdelnabi, K. Greshake, S. Mishra, C. Endres, T. Holz, and M. Fritz. “Not What You’ve Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection,” in *Proc. 16th ACM Workshop on Artificial Intelligence and Security*, Copenhagen, Denmark, 2023, pp. 79-90. doi: 10.1145/3605764.3623985.
- [10] B. Green. “The Flaws of Policies Requiring Human Oversight of Government Algorithms.” *Computer Law & Security Review*, vol. 45, art. 105681, Jul. 2022. doi: 10.1016/j.clsr.2022.105681.