Proceedings of the Design Society, Volume 6: DESIGN 2026
https://doi.org/10.1017/pds.2026.10601

Evaluating large language models for automated design structure matrix extraction from unstructured documents: an empirical study

Anubhab Majumder✉, Sahana Parasuram, Kausik Bhattacharya and Amaresh Chakrabarti

Indian Institute of Science, Bangalore, India

✉ majumderanubhab@gmail.com

ABSTRACT: Design Structure Matrices (DSMs) capture dependencies between system entities and help analyze system complexity, but manually creating them from unstructured documents is time consuming. This work proposes an automated DSM extraction framework using LLMs and RAG with an explicit reasoning step before the LLM determines the presence of a dependency between two system entities. Using a hand-curated dataset, we evaluate three LLM models (GPT-4o-mini, GPT-3.5, and GPT-4o) across six performance metrics and cost.The findings show that reasoning length affects LLM's DSM extraction performance.

KEYWORDS: design structure matrix (DSM), large language model (LLM), retrieval-augmented generation (RAG), empirical studies

1. Introduction

A Design Structure Matrix (DSM) is an N×N square matrix that maps the relationships among a set of N system entities (Browning 2015). DSM are widely used across various domains including product development, project management, and systems engineering for their ability to visualize, analyze, and optimize system architecture (Eppinger & Browning 2012). Constructing a DSM early in the process can help identify subsystems and their interactions, enabling more structured decomposition and targeted analysis. DSMs can also be mapped to graph representations, making them a flexible and efficient tool for system modelling from text.

The use of DSMs in both research and industry has been extensively documented by Browning (Browning, 2002). Their utility extends across numerous industries, including construction, automotive, aerospace, telecommunications, and electronics, as well as in design synthesis, analysis, and process studies (Browning, 2002). There are four key types of DSM applications that are particularly valuable for product developers, project planners, systems engineers, and organizational designers (Dorf, 1999):

(a) Component-Based (Architecture) DSM that captures system architectures by representing components and/or subsystems and their interconnections, (b) Team-Based (Organization) DSM that represents organizational structures by modelling individuals, teams, and their interactions, (c) Activity-Based (Schedule) DSM that captures process flows and activity networks, including information flows and task dependencies, and (d) Parameter-Based (Low-Level Schedule) DSM which describes detailed relationships between design decisions and parameters, such as systems of equations or subroutine parameter exchanges.

This paper specifically focuses on the Component-Based symmetric DSMs, which capture interactions among elements within a system's architecture. To classify these interactions systematically, a well-defined taxonomy is essential. Pimmler and Eppinger (1994) propose a widely adopted framework that distinguishes these four types of interactions, as outlined in Table 1.

https://doi.org/10.1017/pds.2026.10601 Published online by Cambridge University Press

Table 1. Types of interaction in component-based DSM (Pimmler & Eppinger, 1994)

Type	Definition
Spatial	"A spatial-type interaction identifies needs for adjacency or orientation between two elements."
Energy	"An energy-type interaction identifies needs for energy transfer between two elements."
Information	"An information-type interaction identifies needs for information or signal exchange between two elements."
Material	"A material-type interaction identifies needs for materials exchange between two elements."

However, despite the numerous benefits of DSMs, their creation is a labor-intensive process. Traditional methods for DSM generation rely on surveys, structured interviews, and meetings with engineers, requiring a large amount of time and effort. Koh (2024) has compiled several examples that demonstrate the amount of time and effort involved in the creation of DSMs. These examples emphasize the significant time and cost required to manually construct a DSM. This motivates the exploration of automated approaches capable of generating DSMs directly from existing technical documentation.

Large Language Models (LLMs) are state-of-the-art Artificial Intelligence (AI) systems designed to process and generate human-like text. They can perform tasks such as content generation, question answering, and machine translation (Raiaan et al., 2024). Given these capabilities, LLMs offer a promising avenue for automating DSM generation. However, LLMs also have a few drawbacks. LLMs can sometimes face challenges such as hallucination (Huang et al., 2025), only having access to outdated knowledge and non-transparent reasoning processes, etc. Retrieval-Augmented Generation (RAG) addresses these limitations by integrating LLMs with external knowledge sources, enabling models to retrieve relevant information dynamically during inference (B´echard & Ayala, 2024).

1.1. Related work

The automation of DSM generation has evolved through several key contributions, with an early and notable advancement made by Dong and Whitney (2001). Their work introduced a method for systematically deriving a DSM from a Design Matrix (DM), a foundational construct in Axiomatic Design theory (Suh, 1998). Senescu et al. (2012) introduced an "Automatic Information Dependency Algorithm (AIDA)," a proof-of-concept system designed to infer networks of information dependencies by monitoring the sequence and frequency of professionals' interactions with files. The algorithm uses data logs, typically available in information management tools, and generates a matrix representing the information dependencies, offering insights into collaborative workflows. While this approach showed that dependencies could be identified, it also suffered from many false positives due to coincidental co-occurrences and concurrent work. To address this issue, further advancements were made by Gopsill et al. (2016), who explored whether a continuously evolving DSM could be automatically generated by monitoring changes in digital models (e.g., CAD, FEA, CFD, etc.) representing a product. Pons et al. (2021) proposed an automated method for generating DSMs from SysML models developed using open-source MBSE tools. The tool demonstrates strong potential for integration into MBSE workflows, although it currently depends on naming conventions for cross-diagram mapping and lacks full support for all SysML diagram types. Additionally, the approach does not fully address scalability concerns or the challenges posed by the high-fidelity and complex nature of SysML models.

Several studies have highlighted the growing potential of Natural Language Processing (NLP) techniques in supporting design system decomposition and analysis (Siddharth et al., 2022; Tosserams et al., 2010). NLP-based approaches for DSM creation have also been explored in various studies, as summarized by Koh (2024). However, the integration of LLMs into DSM creation remains relatively unexplored. Koh (2024) introduced Auto-DSM, a novel workflow that leverages an off-the-shelf LLM (GPT-3.5-turbo) to automate the creation of DSMs from unstructured textual data. The method employs a retrieval-augmented generation (RAG) approach, where input documents are split, embedded, and stored in a vector database. A simple predefined prompt is then used to extract pairwise dependencies between system elements, which are compiled into a DSM. The workflow was tested on a diesel engine case study using various data sources, including technical handbooks and YouTube comments. Auto-DSM achieved up to 77.3% overlap with a manually curated DSM, demonstrating promising feasibility. Compared to direct prompting of ChatGPT, Auto-DSM consistently yielded higher completeness (i.e., proportion of DSM entries with usable labels), though correctness (i.e., proportion of symmetrical entries

https://doi.org/10.1017/pds.2026.10601 Published online by Cambridge University Press

in DSM) varied depending on input data source quality. While the study did not explore advanced prompting strategies or LLM model comparisons, it represents a significant step toward DSM generation using LLMs.

1.2. Aim, objectives and research questions

The overarching aim of this research is to systematically evaluate the potential of LLMs, within a RAG framework, to automate the extraction of symmetric DSM entity-pair relations from unstructured technical documents. This involves two complementary objectives. The first is to develop a framework capable of retrieving relevant information from unstructured documents and generating DSM entries. The second is to empirically evaluate the performance of different LLMs within this framework in order to understand their strengths, limitations, and suitability for DSM extraction tasks.

Accordingly, this study addresses two specific research questions: (RQ1) to what extent a RAG-enabled LLM framework can reliably extract symmetric DSM relationships from unstructured technical documents? and (RQ2) how different closed-source OpenAI GPT models compare in terms of their extraction performance and cost-efficiency?

2. Method

To address RQ1, our work is built upon the foundational RAG-enabled LLM-based DSM extraction framework proposed by Koh (2024); however, it introduces several key advancements that differentiate it from prior work. First, we propose an enhanced prompting strategy that incorporates an explicit reasoning step before generating the final output. This addition is motivated by recent studies demonstrating that intermediate reasoning significantly improves the accuracy and reliability of LLM responses (Wei et al., 2022). Prompt engineering is crucial for optimizing LLM responses (Chen et al. 2025). In the proposed framework, two distinct prompts are employed: (a) a query expansion prompt, used to extract contextual information relevant to entity relations, and (b) a Chain-of-Thought (CoT) prompt, designed to guide the model's reasoning in generating decisions for DSM entries. The resulting extended framework is discussed in more detail in Section 3.

To address RQ2, we evaluate our methodology using a set of synthetically generated test cases, for which we have developed a human-annotated benchmark dataset to ensure objective evaluation. We then define a set of performance metrics to systematically benchmark the performance and cost-efficiency of different LLMs, analyzing the trade-offs between output quality and cost. In this study, we benchmark the DSM extraction performance of three closed-source OpenAI GPT models: GPT-4o-mini, GPT-3.5-Turbo, and GPT-4o, across seven evaluation dimensions. The first four metrics are Accuracy, Precision, Recall, and F1-score. The fifth and sixth metrics, correctness and self-consistency, have been adopted from Koh (2024) and measure how much the model agrees with itself. The final metric is cost, which is used to assess the trade-off between model performance and inference cost. The dataset preparation, models selected for experiments, and performance metrics are each described in the following subsection.

2.1. Experimental setup

2.1.1. Dataset preparation

To evaluate our proposed approach to extract DSMs from unstructured documents, a ground truth dataset is needed where a human expert analyzes the given textual description of a particular system and manually identifies the dependencies by reading the document. Since unstructured technical documents, such as patents, are complex and difficult to interpret, preparing ground truth and validating results for the same is time-consuming. In addition, patents or handbooks describing complex systems typically contain a large number of system entities (e.g., mechanical parts, electronic modules, etc.), leading to high experimental costs when testing with closed-source paid LLMs such as OpenAI during the iterative prompt designing and testing phase.

To mitigate these challenges, we conduct an empirical study using nine hand-crafted documents that simulate unstructured technical descriptions of familiar systems (e.g., ceiling fan, tricycle, cold-pressed juicer, etc.). These synthetic documents were manually authored following a consistent template derived from common patterns found in real technical texts. Each document includes: an abstract, providing a high-level system overview; a background section, describing historical context and purpose of the

https://doi.org/10.1017/pds.2026.10601 Published online by Cambridge University Press

system; a system summary, listing key components and their high-level roles; a multi-paragraph description section, where the structure, function, and interactions between components are explained in natural technical language; and a short conclusion, describing the overall behaviour of the system.
This structure was intentionally chosen to mimic the narrative and organizational features seen in patents, engineering manuals, and introductory product descriptions. The documents maintain realistic variation in phrasing and interaction descriptions (e.g., mechanical attachment, electrical connection, material/energy/information flow), ensuring that entity relationships appear in diverse linguistic forms similar to real-world documentation. For example, statements in the ceiling fan document include: *"once the motor is powered, it drives the rotation of the blades", "The switch is electrically connected to the motor through the internal wiring . . . ",* etc. Such variation ensures that DSM relationships are not trivially detectable and require semantic interpretation.
These documents are then given to two experts to hand-annotate the DSM entries to prepare the ground truth DSM dataset. Across the nine documents, word counts vary between 700 and 1300 words and the number of entities ranges from 5 to 13. These documents are used to iteratively refine our framework, i.e., tuning prompts, debugging, and experimenting with different query strategies, without incurring high computational costs. The level of agreement between the annotators was measured using Cohen's kappa (κ), a statistical metric that assesses inter-rater agreement beyond chance (McHugh, 2012). A κ score of 0.89 indicated a strong level of agreement, validating the reliability of the annotations.

2.1.2. LLMs used in experiments

To evaluate the application's performance across different LLMs, three OpenAI models were selected, namely, GPT-4o, GPT-4o-mini, and GPT-3.5-Turbo. GPT-4o is one of OpenAI's flagship chat models, at the time of conducting this study, with high intelligence. GPT-4o-mini is a lightweight, cost-optimized model trained on less data, and provides relatively moderate intelligence. GPT-3.5-Turbo is a legacy GPT model with comparatively lower intelligence than the other two. While running the experiments, we configured the LLMs with a temperature setting of 0 to minimize randomness and ensure deterministic outputs. (OpenAI, 2025)

2.1.3. Performance evaluation matrix

To assess the performance of different LLMs in automating DSM creation, we have used seven key evaluation metrics discussed as follows.

Accuracy, precision, recall, and F1-score

We have used four essential evaluation metrics to assess the performance of the LLMs - accuracy, precision, recall, and the F1-score. These are the standard metrics used to measure the performance of a prediction model and provide a well-rounded understanding of how well the model's predictions align with the ground truth annotations (Naidu et al., 2023).
Accuracy measures the overall proportion of correct predictions made by the model. Precision quantifies the proportion of predicted positive links that were correct. Recall captures the proportion of actual positive links that the model was able to identify. F1-score provides a balanced measure that combines both precision and recall.

Correctness and self-consistency

The metrics correctness and self-consistency were introduced by Koh (2024) and were used to evaluate the proposed framework. Correctness is a metric that measures the proportion of symmetrical links identified relative to the total number of links identified by the model (see Equation 1). It assesses whether the model maintains reciprocal link logic (i.e., if A is linked to B, then B should also be linked to A).

$$Correctness = \frac{Number\ of\ symmetrical\ links\ found}{Total\ number\ of\ links\ found} \tag{1}$$

Where 'Number of symmetrical links found' refers to instances where the model found both directions of a relationship to be present (e.g., A →B and B →A). 'Total number of links found' is the total number of 'linked' relationships found by the model.
In contrast, self-consistency checks how much the model applies its own logic uniformly without contradicting itself. In our analysis we examined whether the decision provided by the model is consistent in both directions (see Equation 2).

https://doi.org/10.1017/pds.2026.10601 Published online by Cambridge University Press

$$Self - consistency = \frac{Number\ of\ consistent\ decisions}{Total\ number\ of\ decisions} \quad (2)$$

Cost metric

To evaluate the computational cost of the DSM generation process, we estimate the monetary expense incurred per prompt based on token usage and model-specific pricing. For each query, the cost is computed using the number of input and completion tokens processed by the model, multiplied by the respective token pricing for input and completion, as published by OpenAI. The cost rates for each model are derived from OpenAI's official API pricing (https://openai.com/api/pricing/). Note that the cost metric does not account for cost related to computational resources such as processing time, memory usage, or hardware load.

3. Proposed application framework

The proposed LLM-RAG application framework is depicted in Figure 1. The steps are described in the following parts of this section.

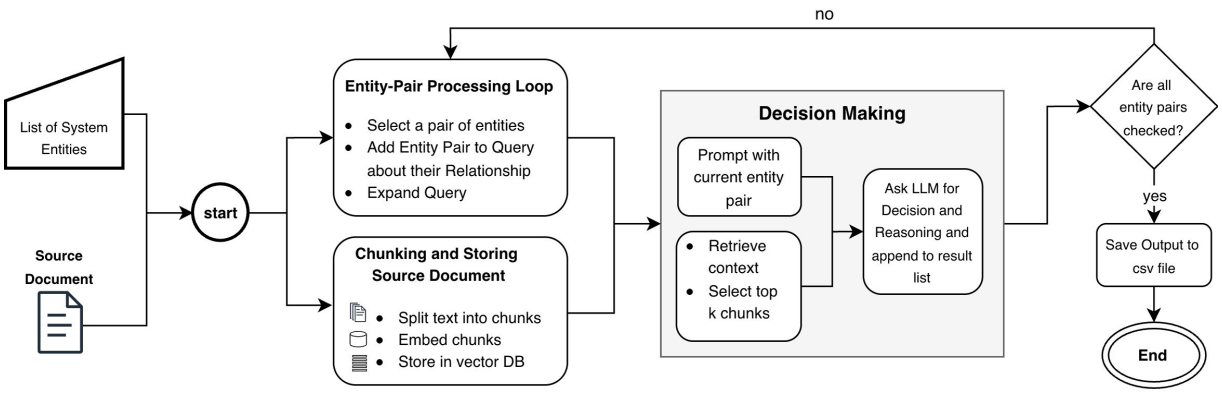


Figure 1. Workflow of the proposed LLM-RAG application

3.1. Chunking and storing source document embedding

First, the full document undergoes text chunking, where it is divided into segments of a predefined number of characters each, following fixed-length chunking practices that are widely used in RAG pipelines (Gao et al., 2023). A chunk size of 150 characters and a chunk overlap of 15% are set to improve contextual continuity across chunks. These parameters are determined through trial and error and based on document length. These chunks are then embedded into a vector database using *OpenAIEmbeddings* embedding model that converts textual content into numerical vectors based on semantic meaning.

3.2. Entity relationship context extraction

A nested loop iterates over all entity pairs, with a complexity of $O(n^2)$, ensuring that dependency search for each entity pair in both directions is covered. The entity pairs are then processed using a semantic similarity-based retriever. This queries the vector database extracting relevant chunks that should contain information about the interaction between two entities. The query undergoes a process called query expansion in each iteration. The two entities under consideration are dynamically replaced with a new entity pair after every iteration.

3.2.1. Query expansion prompt

As shown in Figure 2, the query expansion prompt guides the LLM to generate multiple semantically equivalent variations of a given query. This enhances information retrieval by broadening its lexical and syntactic range while preserving the original intent. Prior work has shown that such query expansion improves recall, reduces ambiguity, and supports better decision-making by ensuring broader semantic coverage (Jagerman et al. 2023).

https://doi.org/10.1017/pds.2026.10601 Published online by Cambridge University Press

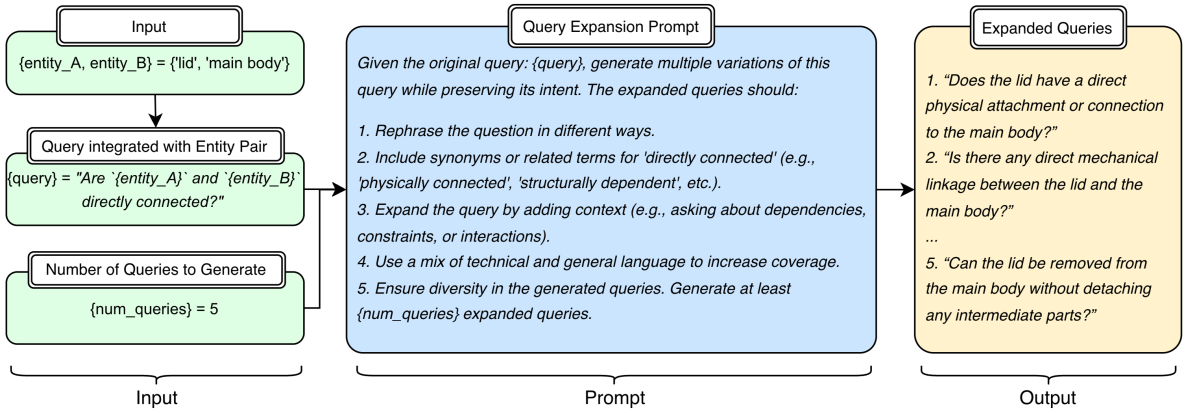


Figure 2. Query expansion prompt

3.3. Decision making for DSM entries via LLM

The retieved chunks are provided as context to the LLM along with a Chain-of-Thought (CoT) prompt. The goal is to check whether two entities are interacting with each other in terms of spatial, energy, information, or material means (see Table 1). The model analyzes the retrieved context and makes a binary decision (YES/NO) for each entity pair, storing the results along with its reasoning in a CSV file.

3.3.1. Decision making prompt

The decision-making prompt is shown in Figure 3. Here, we apply the Chain-of-Thought prompting technique, instructing the LLM to reason step-by-step before arriving at a decision. Studies show that this "think-then-decide" process leads to higher accuracy compared to direct answer prompting (Sahoo et al., 2024; Wei et al., 2022). The prompt is structured around a pre-defined taxonomy of interaction types - spatial, energy, information, and material as described in Table 1. These descriptions have not been elaborated in Figure 3 but have been included in the prompt. Additionally, it enforces a constrained JSON output format to ensure responses are structured and machine-readable for downstream DSM construction. For instance, as shown in Figure 3, the model receives the entity pair 'lid' and 'main body' and responds that they are directly connected, with reasoning based on the paragraph (context chunks) stating that the lid is attached to the top of the kettle's body.

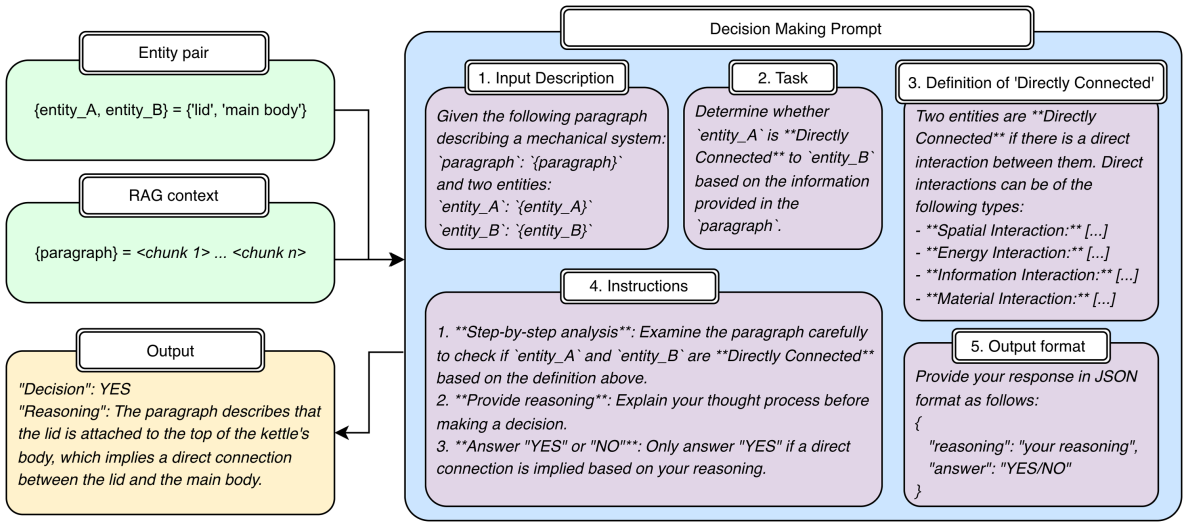


Figure 3. Decision making prompt

4. Evaluation of the proposed framework

The LLM-RAG pipeline developed (see Figure 1) was applied to the nine synthetic documents. Three different models were employed for this evaluation: GPT-3.5-turbo, GPT-4o-mini, and GPT-4o.

https://doi.org/10.1017/pds.2026.10601 Published online by Cambridge University Press

4.1. Model performance on hand-curated documents

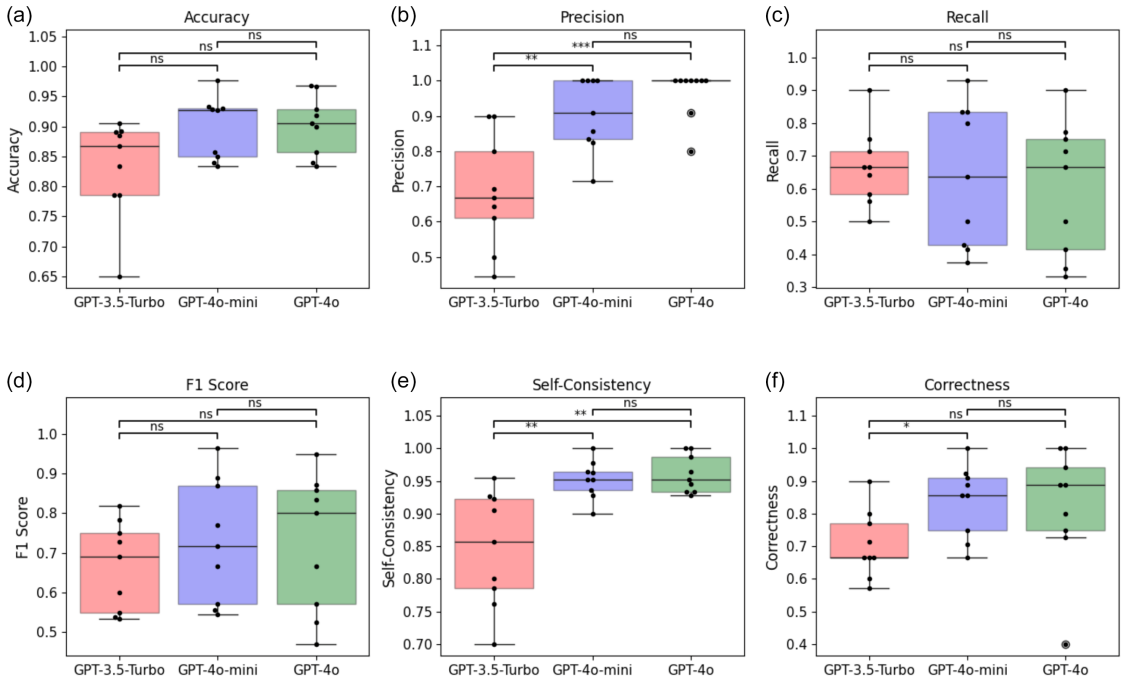
Across all metrics, the three models show broadly similar performance, with only a few statistically significant differences, as reported in Figure 4.

Accuracy, precision, recall, and F1-score

GPT-4o achieves the highest average accuracy (0.91, SD = 0.05) and precision (0.97, SD = 0.07), while GPT-3.5-Turbo records the lowest accuracy (0.83, SD = 0.082) and precision (0.68, SD = 0.16). Significant differences in precision were found between GPT-4o and GPT-3.5-Turbo ($t = 5.495$, $p < 0.05$) and between GPT-4o-mini and GPT-3.5-Turbo ($t = 4.545$, $p < 0.05$). For recall and F1-score, the models perform comparably, with no statistically significant differences despite small numerical variations.

Correctness and self-consistency

For self-consistency, GPT-4o (0.96) and GPT-4o-mini (0.95) achieve the strongest performance with low variability, while GPT-3.5-Turbo scores lower (0.85) with higher variance; statistically significant differences are observed between GPT-4o vs. GPT-3.5-Turbo and GPT-4o-mini vs. GPT-3.5-Turbo ($p < 0.05$). In terms of correctness, GPT-4o-mini performs best (0.84), followed by GPT-4o (0.82), with GPT-3.5-Turbo again lowest (0.706); only GPT-4o-mini vs. GPT-3.5-Turbo shows a significant difference, with no significance between GPT-4o and GPT-4o-mini.



*** p<0.001, ** p<0.01, * p<0.05, ns = not significant

Figure 4. Model performance comparison

Cost

To estimate the computational costs involved in our experiments, first, we calculated the average input tokens and completion tokens per prompt considering all hand-curated documents. The distribution of input tokens and completion tokens are shown in Figure 5(a) and Figure 5(b), respectively. Then, using the official token pricing provided by OpenAI, we computed the cost per query for different language models. GPT-4o-mini is the cheapest at ~$0.0002 per query, followed by GPT-3.5-Turbo at ~$0.0005, while GPT-4o is the most expensive at ~$0.0032 per query.

4.2. Correlation between completion tokens and performance metrics

First, we performed standard pairwise t-tests to compare the mean completion token counts across the three models. The results revealed statistically significant differences in the average completion tokens generated by each model ($p < 0.01$ for all pairs), indicating that the models produce different number of completion tokens. These findings suggest that the models differ in their token generation behaviour, a trend that is also visually evident in the distributions shown in Figure 5(b).

https://doi.org/10.1017/pds.2026.10601 Published online by Cambridge University Press

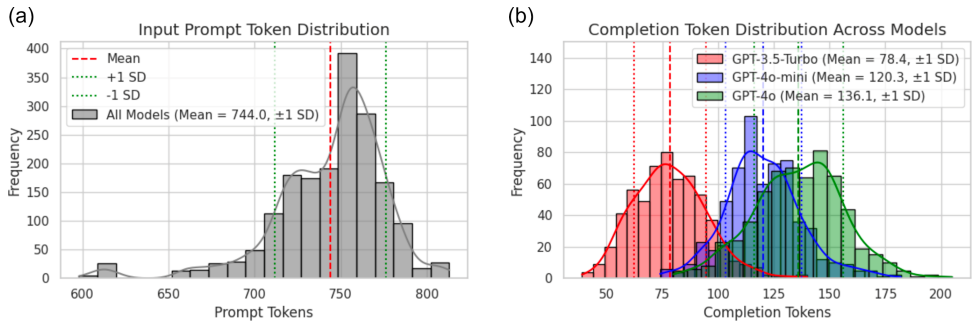


Figure 5. (a) Distribution of input prompt tokens aggregated across all three models; (b) Comparison of completion token distributions across the three models (GPT-3.5-Turbo, GPT-4o-mini, and GPT-4o)

Correlation analysis is conducted between the average completion tokens generated per document and the six-performance metrics (see Section 2.1.3) across three models (GPT-3.5-turbo, GPT-4o-mini, and GPT-4o) evaluated on nine hand-curated documents. The results can be seen in Figure 6. The analysis encompassed all 27 data points collectively.

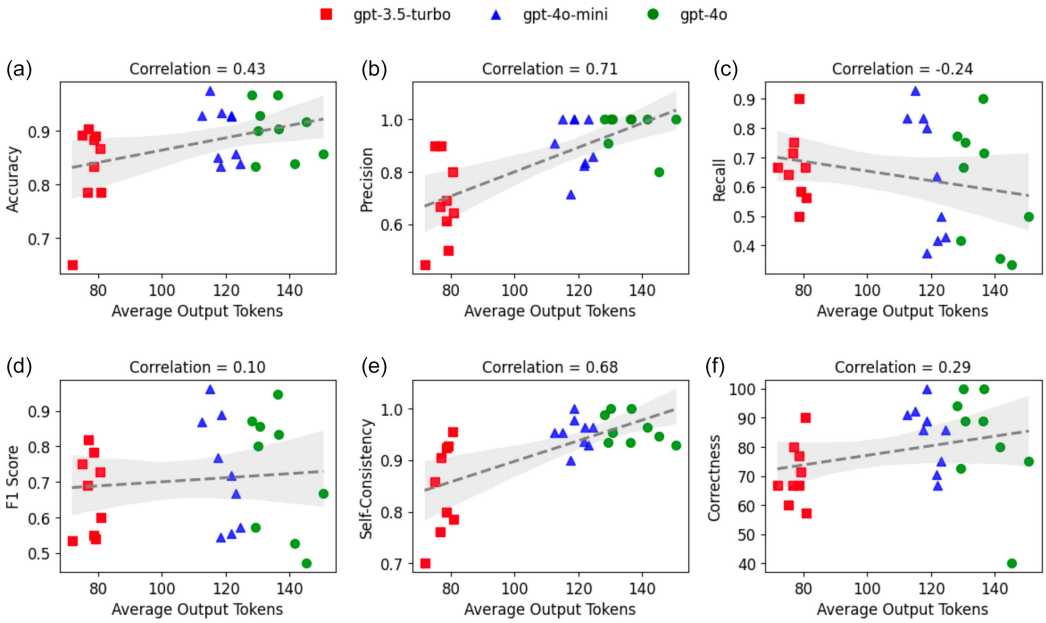


Figure 6. Correlation analysis between performance metrics and average completion or output tokens

The correlation coefficients indicate a moderate positive relationship between average completion tokens and both Accuracy ($r = 0.43$) and Self-Consistency ($r = 0.68$). Correctness demonstrates a weaker positive correlation ($r = 0.29$), while the F1-score exhibits the lowest correlation with the completion token count ($r = 0.10$). Precision exhibits the strongest positive correlation ($r = 0.71$), indicating that longer responses tend to increase the model's ability to correctly identify true positive links without including false positive ones. However, Recall showed a weak negative correlation ($r = -0.24$), implying that as completions become longer, the models might miss some relevant links. This trade-off between Precision and Recall contributes to the weak overall correlation between completion tokens and F1-score ($r = 0.10$), which balances both.

5. Discussion

The results obtained from the experiments demonstrate the effectiveness of our proposed framework for extracting Design Structure Matrices (DSMs) from unstructured documents using large language models (LLMs). For the hand-curated documents, across three tested models, GPT-3.5-turbo, GPT-4o-mini, and GPT-4o, all were able to perform the task reliably, underscoring the general robustness of the approach. Among the models, both GPT-4o and GPT-4o-mini demonstrated superior performance across key metrics, including self-consistency and precision, with statistically significant improvements over GPT-3.5-turbo.

https://doi.org/10.1017/pds.2026.10601 Published online by Cambridge University Press

These results highlight a practical trade-off: while GPT-4o offers the most balanced overall performance, GPT-4o-mini delivers competitive performance at significantly lower inference cost, making it viable for large scale deployments with large number of system entities.

The correlation analysis reveals a moderate to strong positive correlation between output reasoning length and both accuracy and self-consistency. These findings help explain how smaller models like GPT-4o-mini, through more detailed reasoning in their outputs, can match or surpass the performance of larger models in certain metrics.

A closer inspection of error profiles across models reveals additional practical considerations. GPT-4o-mini exhibited a higher number of false positives, tending to identify links that did not exist in the ground truth. Conversely, GPT-3.5-turbo and GPT-4o generated more false negatives, missing some valid links. These distinct behaviors suggest that model selection can be tailored based on application needs. If the goal is to capture as many potential dependencies as possible, GPT-4o-mini may be preferable. However, if precision is important and minimizing erroneous links is critical, GPT-4o or GPT-3.5-turbo may be better suited.

6. Conclusions, limitations and future work

This research presents a RAG-based framework that LLMs to automate the generation of symmetric DSMs from unstructured technical documents. By prompting models to explicitly reason about dependencies between system elements, our approach improves upon prior tools. An empirical evaluation using nine synthetic documents demonstrates that LLMs, particularly GPT-4o and GPT-4o-mini, can achieve promising performance in DSM extraction, with GPT-4o-mini offering an attractive trade-off between cost and performance. Our findings also show that the quality of model reasoning, as reflected in output length, contributes significantly to performance.

Despite these promising results, several limitations remain. First, our current implementation assumes that the list of system entities is manually provided. Future work will explore integrating entity extraction using RAG to identify relevant components directly from source documents. Second, while the synthetic dataset enabled controlled evaluation, the hand-curated nature and limited complexity of these documents do not fully represent the challenges posed by real-world engineering texts. Hence, broader validation using diverse, large-scale, and noisy technical documents is needed to evaluate generalizability. Cost constraints limit the scale of such testing in this study. Third, we observed occasional inconsistencies in model outputs, particularly predicting asymmetric or contradictory dependencies. Addressing such conflicts through an additional consistency resolution mechanism is a key next step to improve reliability and DSM coherence. Fourth, this study focuses specifically on systems with symmetric DSMs, and in the future, we aim to extend the framework to accurately predict entity-pair relations for asymmetric DSMs as well. Fifth, although the study focused exclusively on OpenAI models, future work should broaden evaluation to additional LLMs and strengthen the framework itself by improving prompting, refining the RAG pipeline, and assessing performance across different document types. Finally, this work focused on architectural DSMs that represent component-level interactions. However, DSMs can also represent organizational or procedural relationships. Extending the proposed framework to handle these diverse DSM types will enhance its applicability across a wider range of engineering and management contexts.

References

B´echard, P., & Ayala, O. M. (2024). Reducing hallucination in structured outputs via retrieval- augmented generation. arXiv preprint arXiv:2404.08189. https://doi.org/10.18653/v1/2024.naacl-industry.19

Browning, T. R. (2002). Applying the design structure matrix to system decomposition and integra- tion problems: A review and new directions. *IEEE Transactions on Engineering manage- ment*, 48(3), 292–306. https://doi.org/10.1109/17.946528

Browning, T. R. (2015). Design structure matrix extensions and innovations: A survey and new opportunities. *IEEE Transactions on engineering management*, 63(1), 27–52. https://doi.org/10.1109/TEM.2015.2491283

Chen, B., Zhang, Z., Langrené, N., & Zhu, S. (2025). Unleashing the potential of prompt engineering for large language models. *Patterns*, 6(6). https://doi.org/10.1016/j.patter.2025.101260

Dong, Q., & Whitney, D. E. (2001). Designing a requirement driven product development process. *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 80258, 11–20. https://doi.org/10.1115/DETC2001/DTM-21682

https://doi.org/10.1017/pds.2026.10601 Published online by Cambridge University Press

Dorf, R. C. (1999). The design structure matrix. In R. C. Dorf (Ed.), *Technology management handbook* (pp. 103–111). Chapman & Hall/CRCnetBASE.
Eppinger, S. D., & Browning, T. R. (2012). *Design structure matrix methods and applications*. MIT press.
Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Guo, Q., Wang, M., & Wang, H. (2023). Retrieval-Augmented Generation for Large Language Models: A Survey. ArXiv, abs/2312.10997.
Gopsill JA, Snider C, McMahon C, Hicks B. Automatic generation of design structure matrices through the evolution of product models. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*. 2016; 30(4): 424–445. https://doi.org/10.1017/S0890060416000391
Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., *et al.* (2025). A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2), 1–55. https://doi.org/10.1145/3703155
Jagerman, R., Zhuang, H., Qin, Z., Wang, X., & Bendersky, M. (2023). Query expansion by prompting large language models. arXiv preprint arXiv:2305.03653.https://doi.org/10.48550/arXiv.2305.03653
Koh, E. C. (2024). Auto-dsm: Using a large language model to generate a design structure matrix. *Natural Language Processing Journal*, 9, 100103. https://doi.org/10.1016/j.nlp.2024.100103
McHugh, M. L. (2012). Interrater reliability: the kappa statistic. *Biochemia medica*, 22(3), 276–282. doi: 10.11613/bm.2012.031
Naidu, G., Zuva, T., & Sibanda, E. M. (2023). A review of evaluation metrics in machine learning algorithms. *Computer science on-line conference*, 15–25. Cham: Springer International Publishing.
OpenAI. (2025). Models overview [Accessed: 2025-05-05]. https://platform.openai.com/docs/ models/compare
Pimmler, T. U., & Eppinger, S. D. (1994). Integration analysis of product decompositions. *Inter- national Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 12822, 343–351. https://doi.org/10.1115/DETC1994-0034
Pons, W., Cordero, S. K. S., & Vingerhoeds, R. A. (2021, September). Design structure matrix generation from open-source mbse tools. In *2021 IEEE International Symposium on Systems Engineering* (pp. 1–8). https://doi.org/10.1109/ISSE51541.2021.9582525
Raiaan, M. A. K., Mukta, M. S. H., Fatema, K., Fahad, N. M., Sakib, S., Mim, M. M. J., Ahmad, J., Ali, M. E., & Azam, S. (2024). A review on large language models: Architectures, applica- tions, taxonomies, open issues and challenges. *IEEE access*, 12, 26839–26874. https://doi.org/10.1109/ACCESS.2024.3365742
Sahoo, P., Singh, A. K., Saha, S., Jain, V., Mondal, S., & Chadha, A. (2024). A systematic survey of prompt engineering in large language models: Techniques and applications. arXiv preprint arXiv:2402.07927, 1. https://doi.org/10.48550/arXiv.2402.07927
Senescu, R. R., Head, A. W., Steinert, M., & Fischer, M. A. (2012, September). Generating a network of information dependencies automatically. In *Proc. 14th Int. Dependency and Structure Modelling Conf* (pp. 139–152). https://doi.org/10.3139/9783446434127.012
Siddharth, L., Blessing, L., & Luo, J. (2022). Natural language processing in-and-for design research. *Design Science*, 8, e21. https://doi.org/10.1017/dsj.2022.16
Suh, N. P. (1998). Axiomatic design theory for systems. *Research in engineering design*, 10(4), 189–209. https://doi.org/10.1007/s001639870001
Tosserams, S., Hofkamp, A. T., Etman, L. F. P., & Rooda, J. E. (2010). A specification language for problem partitioning in decomposition-based design optimization. *Structural and Multidisciplinary Optimization*, 42(5), 707–723. https://doi.org/10.1007/s00158-010-0512-z
Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., … & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35, 24824–24837. https://doi.org/10.48550/arXiv.2201.11903

https://doi.org/10.1017/pds.2026.10601 Published online by Cambridge University Press