Enabling Model-Based RAMS Through LLM-Driven Legacy Data Transformation

You-Jung Jun, PHM Technology
Raphael Chagnoleau, PHM Technology
Yanek Stecki, PHM Technology
Navid Zaman, PHM Technology
Derek Kim, PHM Technology

Key Words: Model-Based system engineering, digital twin, generative AI, LLMs

ABSTRACT

The rapid digital transformation in engineering, coupled with the development of increasingly complex systems, is pushing industries to develop smarter and more efficient methods for system development. Major stakeholders/ industries are moving towards a model-based framework for systems engineering, RAM, and safety analysis to manage growing system complexity while maintaining data consistency and traceability.

The convergence and consolidation of previously document-based engineering approaches allows for the standardization and streamlined capture of knowledge across engineering disciplines. In this framework, data availability and interoperability can easily become a bottleneck without comparable innovation to tooling and processes. In more recent times Artificial Intelligence (AI) has been identified as a powerful enabler on this front. AI can assist engineers in developing RAMS models more efficiently by leveraging legacy data, such as historical FMECAs, and aligning it with standardized taxonomies to automatically and rapidly develop system models for downstream analysis requirements.

1 INTRODUCTION-CURRENT INDUSTRY LANDSCAPE

Modern engineering systems become increasingly complex as the ever-growing demand for more functionality perpetuates a software driven evolution of engineering process and design. This trend is prevalent in high velocity sectors such as aerospace, and defense where cutting edge consistently contends with strict safety, regulatory, and cost constraints; requiring increasingly efficient, necessarily integrated system analysis and decision making capabilities.

System development requires contributions from a wide range of disciplines, organizations, and geographic locations. This distributed development environment frequently results in siloed activities and workflows, where different stakeholders rely on disparate tools, data formats, and methodologies. The absence of centralized and standardized/ compatible data exacerbates challenges such as duplication of effort, data inconsistency, and poor change management.

Even with many software tooling options available to do the heavy lifting in engineering modelling and analyses, interoperability remains a persistent challenge. Outputs from different tools are often incompatible, and the lack of digital continuity limits the ability to effectively reuse prior work—leading to redundant efforts and increased development timelines.

A particularly pronounced symptom of this actuality is the vast amounts of legacy data in the form of historical FMEA/ FMECA reports which continue to remain underutilized. These documents contain critical engineering insights, yet they are often immediately stored unmanaged, unmaintained, and under-referenced, in formats that are not compatible with modern applications.

A Digital Twin approach grounded in model-based engineering offers a promising solution. By creating structured, centralized, and machine-readable representations of systems, organizations can improve traceability, enable cross-disciplinary collaboration, and ensure consistency throughout the system lifecycle. Combining this with AI-powered natural language processing (NLP) techniques, such as large

language models (LLMs), can enable the automated transformation and ingestion of legacy documents into structured and centralized digital models. This not only accelerates model creation and reduces manual workload but also helps preserve engineering intent across the system lifecycle.

2 MODEL-BASED APPROACH TO RAMS

A Digital Twin modelling solution such as MADE (Maintenance Aware Design Ecosystem) enables a model-based approach to RAMS (Reliability, Availability, Maintainability, and Safety) analysis by digitally mirroring aspects of a system's expected operational behaviour; simulating the interdependencies of its functions, flows, and failures. At its core, this system model serves as a dependency mapping of components and subsystems, offering a structured and scalable process for modelling and analyzing system-wide resilience against risk.

The model creation process in MADE is typically structured into three key steps:

- Functional Definition – Each system element (or item) is assigned a defined function, characterized by its input and output flows. This allows for a clear understanding of the system's intended behavior and interactions.
- Failure Concept Attribution – For each item, potential failure causes and modes are identified. These define how the item can fail.
- System Dependency Mapping – Components are then connected to one another, establishing a dependency that reflects the system's overall behavior and the cascading effects of failures.

Approaches such as Functional Causal Modelling (FCM) or Bond Graph simulation can be employed to model and analyze these interactions, enabling automated propagation of failure effects across the architecture.

The core modelling principle leverages standardized taxonomies for functions, flows, and failure concepts. Use of common language and labelling allows organization-wide collaboration, distribution and reuse of engineering knowledge, and frictionless integration across toolchains.

A critical enabler of this model-based approach is digital continuity, made possible through integration interfaces such as APIs and structured import formats (e.g., Excel). These ensure that data remains machine-readable and centrally accessible throughout the system lifecycle.

Notably, many legacy FMEA/FMECAs already include the foundational elements needed for model-based analysis (functional descriptions, failure modes, causes, and effects). However, this data is often buried in unstructured, non-standardized documents that are difficult to reuse manually. The advent of advanced LLM technology offers a powerful lifeline. By using LLMs to extract, interpret, and structure legacy RAMS data, organizations can accelerate the creation of Digital Twins and unlock valuable engineering insights that would otherwise remain siloed or inaccessible.

3 AI ASSISTED LEGACY DATA CONVERSION

Recent advances in machine and deep learning have propelled rapid progress in generative artificial intelligence (AI). Because of its versatility and the breadth of knowledge encoded in its models, generative AI is not applied across an ever-widening range of domains. In particular, LLMs have expanded their influence beyond natural language tasks to motion simulation, music generation, and robotic control because of their text generation capabilities.

3.1 *Potential of Generative AI Applications in Systems Engineering*

LLMs are rapidly reshaping systems engineering. Traditionally, engineers have depended on manual design, simulation and analysis to manage the escalating complexity of modern systems. Recent progress in LLM technology now enables AI tools to generate technical artefacts—from detailed diagrams and executable code to well-formed system requirements—with minimal human input [1]. By combining rich systems-engineering knowledge bases with advanced computational techniques, generative AI offers a more efficient and less error-prone approach to both conceptual and practical engineering tasks. Ongoing breakthroughs in multimodal processing and reasoning further broaden this potential, allowing LLM-based systems to ingest test, images, and audio concurrently; as a result, they promise to redefine the efficiency, creativity, and precision of engineering solutions [2].

A concrete use case is automated content generation. Engineers can instruct AI assistants to produce tailored documents for diverse stakeholders or to repurpose existing engineering information for adjacent objectives. For example, generative AI can partially automate the conversion of FMEA data into a formal system model—a task that has historically required labour-intensive, time-consuming manual work by systems engineers. By delegating the initial

model-creation step to AI, engineers can devote more time to comprehensive analysis and iteration, thereby accelerating the overall development cycle [3]. These capabilities herald not merely the automation of routine tasks, but a paradigm shift in how engineers conceive, validate, and iterate on design models.

3.2 *Use of LLMs Toward Model Generation*

Generative AI is permeating not only systems engineering but a growing array of other disciplines, and its use-cases are poised to multiply rapidly. This diversification is being matched by a wave of inventive technical approaches. Therefore, understanding key skills in LLMs are crucial since importance of skills are increasing and, they can't be used properly without knowing, so a few skills are described follows:

- Prompt engineering is the deliberate design of input instructions that steer a LLM toward a verifiable, use case specific result. In workflow presented in this paper, each prompt is partitioned into three explicitly labelled blocks: task —a concise statement of the exact work to be performed, criteria—bullet-point constraints that the model must observe, output format—a schema that the model's reply must replicate
- LLMs fall into two categories: proprietary frontier models —such as GPT o4-mini, GPT o3, GPT 4o, Gemini, and Claude—that lead in performance, scalability, and multimodal integration, and open-source alternatives like Llama and Phi, which can be deployed locally for greater data privacy and security. Factors such as model size, tokenization strategy, prompt complexity, and decoding method influence both response fidelity and runtime, so choosing the right model requires balancing these trade-offs. For example, when extracting insights from FMEA data, Llama 3 offered enhanced control and confidentiality but produced largely unsatisfactory outputs, whereas GPT o1and o3-mini generated acceptable results. In creating MADE models—where precise interpretation of complex inputs is vital—minimizing hallucinations and maximizing accuracy are paramount.
- Fine-tuning and retrieval-augmented generation (RAG) enhance LLMs by incorporating domain-specific data, enabling them to deliver more accurate, context-aware outputs. There are multiple ways to apply these methods. For fine-tuning, a dataset that correctly maps FMEA data to the MADE taxonomy can be used to adapt a pre-trained model. RAG is appropriate when engineering manuals or case-specific documentation (e.g., rocket-engine maintenance guides) are available. By integrating these resources, LLMs can produce substantially more precise and reliable responses.

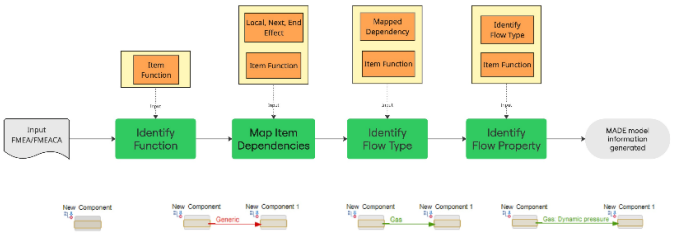
3.3 *Description of Solution Architecture*

To automate system-model creation with LLMs, the processing pipeline must be explicit, modular, and repeatable. The workflow below balances data-engineering pragmatism with LLM-centric best practices are listed in Table 1 below:

Table 1 – Solution Architecture for LLMs Application

Paragraph Stage	Purpose & Design Notes
Legacy-data ingestion	Collect requirements documents, FMEA tables, etc. Convert every source to a machine-readable format (CSV/JSON/markdown)
Domain-taxonomy injection	Supply a controlled vocabulary (flow properties, functions) as a constraint so the model can generate consistent outputs and support interoperability
Deterministic feature extraction	Pre-compute information that is obvious from the data e.g., item hierarchy
Structured prompting	Craft prompt the embed a clear task with ideal formats
Optional adaption layer	Fine tuning or RAG can be used for LLMs for better outputs
Human-in-the-loop validation	Present the draft outputs to engineers/users for verification and feedback
Result serialization & integration	Export final structure to the certain formats e.g., JSON

Figure 1 – Flowchart for a subsystem-level



4 CASE STUDY

There are significant challenges in creating RAMS models for the MADE digital-twinning process, primarily because of the high up-front time and resource requirements. A case study

was developed using an existing FMEA of a rocket engine, to demonstrate the potential advantages of an LLM seeded modelling approach. The FMEA dataset, sourced from a legacy system, includes text-based fields such as LCN, item nomenclature, item function, failure mode, local effect, next effect, and end effect. The dataset contains a wide range of failure modes that vary in specificity. Typical examples include failures related to fuel delivery (e.g., fails to supply fuel, fails to increase fuel flow) mechanical power transmission (e.g., fails to create rotational motion), filtration (e.g., fails to remove contaminants from air/fuel). This variability reflects common inconsistencies found in legacy FMEA sources and presents a realistic challenge for automated model generation, particularly when mapping unstructured text to structured MADE model concepts.

Following the process and workflow described in the previous section, our approach aimed to significantly reduce the time and manual effort required for transformation, making legacy-system integration more efficient and accessible while also synergizing with human expert knowledge. More specifically, the objective of this case study was to generate a fully serviceable MADE model using only FMEA data as input. A serviceable MADE model must include item dependencies, item functions, flows, and flow properties, as these are essential for the model-creation process. These outputs are visualized in Figure 2.

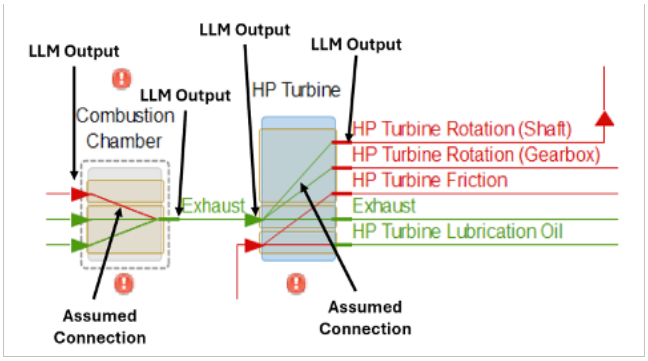
4.1 *Data Injection and Feature Extraction*

When the FMEA dataset is 'loaded' the first major step is to enrich the FMEA dataset with a predefined set of model concept definition taxonomies (i.e. functions, failures, flow properties). This is provided as a text file alongside the source FMEA, the taxonomical language therein acting as the targets for mapping natural language artefacts to. This ensures that all items, hierarchy, and failure definitions are fully recovered.

The emphasis on item hierarchy is essential for the top-down generation process: we begin by producing outputs at the system level and then use those results to guide the generation of subsystem-level outputs. Figure 1 illustrates the generation process, which is applicable to both system-level and all subsystem-level generation processes.

As an example, the case study dataset defines five system-level items—A1 (Fuel Tank), A2 (Diesel Engine), A3 (Coupling), A4 (Alternator Unit), and A5 (Control Unit)—and multiple subsystem items under A2, such as A20001 (Coupling 1), A20002 (Air Filter), A20003 (Engine), and so forth. By identifying and processing A2 at the top level, we can subsequently generate precise outputs for each of its subordinate items. This hierarchical strategy preserves structural consistency and enhances the accuracy and relevance of our feature-extraction pipeline.

Figure 2 – Sample Outputs from LLMs



4.2 *Structured Prompting*

In this phase, we employ LLM-based methodologies to guide and refine our outputs. First, we empirically optimize each prompt to elicit the desired format and context. Because even minor prompt variations can produce dramatically different responses, multiple iterations are required to converge on an ideal prompt structure. We evaluate leading GPT architectures—namely 4o, o1, and o3-mini—both for their cutting-edge performance and scalability. Each model is prompted identically, then its outputs are compared in terms of consistency, accuracy, and relevance to determine which delivers the best results.

Second, we integrate Chain-of-Thought (CoT) prompting [4] to decompose complex generation tasks into smaller, more manageable steps. Our application must produce three distinct outputs—item functions, inter-item flow names, and flow properties—and CoT enables the model to

generate intermediate reasoning for each subtask. By passing these intermediate outputs along the pipeline, we both improve overall accuracy and more closely mimic human problem-solving

4.3 *Human-in-the-loop Validation*

Once the model has generated its outputs, users may review each result and is given the opportunity to provide feedback on its acceptability. If any aspect of the generated information is incomplete or insufficiently detailed, users request additional content from the model. Through this iterative feedback loop—beginning with system-level outputs and cascading into subsystem-level validation—the MADE model gradually becomes both more comprehensive and more concise. Because the LLM occasionally misinterpreted or inconsistently mapped these failure modes, a simple human-in-the-loop correction step was used. Through the UI, engineers were shown the model's output along with a set of valid options for each element (function, flow or property). If the model's selection was incorrect, the user could quickly replace it with the correct option.

4.4 *Result Integration*

After the generation and validation phases are complete, all approved outputs are mapped into the MADE API's data schema. In this final integration step, item dependencies, item functions, flows, and flow properties are organized according to the API specification, producing a fully structured dataset that is ready for deployment in the digital-twinning process.

4.5 *System Model Diagrams Comparison*

Table 2 – Processing Times (s)

Model	Functions	System Boundaries	Dependencies	Flow Properties	Total
o3-mini	192	104	264	900	1460
o1	510	315	1056	1503	3384
4o	84	98	287	504	973

Table 3 – Task Completion

Task	Completion (%)
Generating Functional Failure Analysis Model (FMECA)	87.83
Constructing Baseline RAMS model	67.95
Identify Source Failure Modes	100

Table 4 – LLM & Engineer Modelling Times

Modelled by	Time to Complete (hr)
o3-mini	0.41
o1	0.94
4o	0.27
Engineer	20

In this study, accuracy (5) for each task was calculated as the number of correctly generated artefacts divided by the total number of generated artefacts for that task. As per the results, current LLM capabilities have the potential to reduce the amount of time required to construct a MADE model from legacy FMEA's by up to 95%, It must also be noted that this significant reduction in time does not introduce significant inaccuracies or errors into the model. Although early generation LLM's such as GPT-4o were prone to hallucinations and unsatisfactory outputs, the latest models employ advanced reasoning techniques that dramatically enhance the accuracy of outputs. When assessing the relevant accuracy scores for replicating a complex system model, from the latest model, GPT-o1, the results were as follows:

Table 5 – Accuracy Scores

Task	Accuracy (%)
Designating Function	90.00
System Boundaries	95.00
Mapping Dependencies	89.32
Designating Flows	80.00

These results indicate a good understanding of the system-level boundaries and functions, allowing the LLM to generate the lower-level system items with the relevant context required to generate an accurate and serviceable MADE model. Overall, these results reaffirm the potential for LLM's to drastically reduce modelling time, without sacrificing accuracy of a similar magnitude. The figures below illustrate expected output results, compared to the original MADE model:

Figure 3 – Original Subsystem

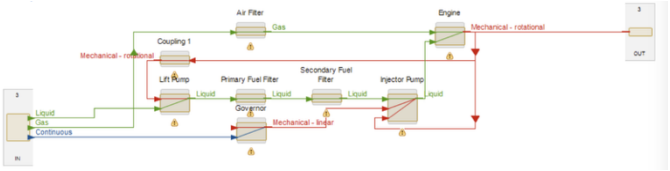
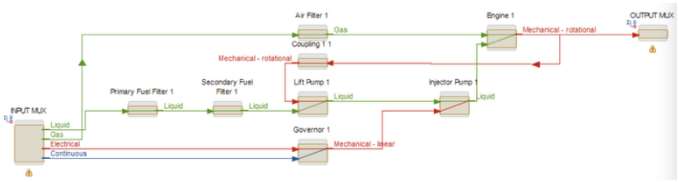


Figure 4 –Subsystem Generated by LLM



5 BENEFITS AND RISKS ASSESSMENT

5.1 *Benefits and Opportunities*

As illustrated in the case study above, leveraging an LLM integrated modelling workflow has the potential to significantly accelerate and enhance the engineering system-model development process—While the case study focused rather narrowly on the scenario of utilizing a relatively unpolished, 'freehand' data source to build scalable engineering system model, this method is easily augmented to enhance modelling beyond this use case. An AI-driven approach dramatically reduces development time and scales effortlessly: models can be generated rapidly and expanded on demand. Looking ahead, AI won't merely automate routine tasks but will also spur innovation by enabling creative and technical applications. Furthermore, ongoing advances in AI—along with the continual upgrade of pre-trained models such as GPT-3 and GPT-4-Mini—promise ever-better quality and more capable responses.

5.2 *Limitations and Risks*

Despite these benefits and rapid technical progress, significant challenges remain:

- Hallucination and Reliability
  LLMs can produce seemingly plausible but factually incorrect outputs. This makes human oversight essential: every model-generated artifact must be vetted, and one should never rely on the model's response without review.
- Implementation Complexity
  Effective use of LLMs requires prompt engineering and programming expertise. Involving subject-matter experts in prompt design is crucial to guide the model in the right direction and extract the needed features for the MADE model.
- Token Constraints
  Each model imposes a limit on the number of tokens (words or word-pieces) per request. Exceeding this limit causes early parts of the prompt or conversation to be truncated, hindering coherence. Techniques like chain-of-thought prompting or dynamic summarization can mitigate—but not eliminate—this issue.
- Lack of Deep Knowledge
  While LLMs excel at general language tasks, they often lack the specialized knowledge needed for certain engineering domains. This limitation is exacerbated when insufficient domain data is available for fine-tuning or retrieval-augmented generation (RAG).

6. CONCLUSION

Generative AI has demonstrated its capacity to transform model-based engineering by automating the extraction, structuring, and validation of complex, often 'freehand' system data—thereby allowing practitioners to redirect effort from routine processing to strategic design and trade-off analysis. In this paper we showcased a practical, end-to-end example: using large language models to convert legacy rocket-engine FMEA/FMECA data into a review-ready Digital Twin within the MADE framework. The case study illustrates how AI-driven generation of item functions, dependencies, flows, and flow properties can cut model-building time by orders of magnitude while preserving engineering intent and traceability. As these methodologies continue to evolve, their accuracy and robustness will steadily improve, addressing today's limitations in reliability, scalability, and domain specificity. Crucially, the ongoing maturation of generative techniques promises not only to enhance the precision of system models but also to simplify engineers' workflows, reducing cognitive load and streamlining collaboration. Future research will therefore focus on refining on-demand knowledge injection mechanisms and human-in-loop validation strategies to ensure that AI-generated outputs remain both accurate and actionable, further integrating generative AI into every phase of the systems engineering lifecycle.

REFERENCES

1. Alzoubi, Y. I., Mishra, A., Topcu, A., & Cibikdiken, A. O. (2024). Generative artificial intelligence technology for systems engineering research: Contribution and challenges. International Journal of Industrial Engineering and Management, 15(2), 169–179.
2. Decardi-Nelson, B., Alshehri, A. S., Ajagekar, A., &

You, F. (2024). Generative AI and process systems engineering: The next frontier. ArXiv
3. Grabill, N., Wang, S., Olayinka, H.A., De Alwis, T.P., Khalil, Y.F. and Zou, J., 2024. AI-augmented failure modes, effects, and criticality analysis (AI-FMECA) for industrial applications. Reliability Engineering & System Safety, 250, p.110308.
4. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V. and Zhou, D., 2022. Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems, 35, pp.24824-24837

BIOGRAPHIES

Jun, You-Jung, M.I.T.
Data Scientist
PHM Technology
9/120 Queens Pd.
North Fitzroy, VIC 3068, Australia
e-mail: youjung.jun@ phmtechnology.com
You-Jung Jun is a Master of Information Technology graduate from the University of Melbourne, Australia since 2022, where he focused data science, machine learning and data analysis. Prior to joining PHMT as a data scientist, he worked at Food Legal as a machine learning engineer.

Zaman, Navid T., MCEng
PHM Technology
9/120 Queens Pd.
North Fitzroy, VIC 3068, Australia
e-mail: navid.zaman@ phmtechnology.com
Navid Zaman is the Lead Data Scientist at PHM Technology, mainly working on causation-based AI and failure detection and isolation tool Syndrome Diagnostics. His main responsibilities are around R&D involving data-driven and AI technologies.

Derek Kim, BEng (MechEng) (Hons)
PHM Technology
9/120 Queens Parade
North Fitzroy, VIC 3068, Australia
e-mail: derek.kim@phmtechnology.com
Derek serves as the Head of Engineering at PHM Technology, accountable for the technical engineering integrity of their products and services. His primary focus is on the development and application of model-based engineering and digital RAMS solutions for the design and sustainment of complex, mission critical assets in aerospace, naval, land, and automotive industries.

Yanek Stecki,
PHM Technology
9/120 Queens Pd.
North Fitzroy, VIC 3068, Australia
e-mail: yanek.stecki@ phmtechnology.com
Yanek Stecki is pursuing a Bachelor of Aerospace Engineering from Royal Melbourne Institute of Technology (RMIT). He is currently working as a Junior Engineer at PHMT, where his primary focus is on the development of AI driven enhancements to the MADE software.

Raphaël Chagnoleau,
PHM Technology
9/120 Queens Pd.
North Fitzroy, VIC 3068, Australia
e-mail: raphael@phmtechnology.com
Raphaël Chagnoleau is the EU Engineering Manager at PHM Technology. He represents the company across Europe by supporting customers through consulting, training, and technical expertise, while also contributing to the ongoing development and improvement of the MADE software.