Proceedings of the Design Society, Volume 6: DESIGN 2026
https://doi.org/10.1017/pds.2026.10575

Handling AI-generated knowledge artifacts in generative product engineering

Martin Becker[1,✉], Damun Mollahassani[1], Simon Schleifer[2], Stefan Goetz[2], Sandro J. Wartzack[2] and Jens C. Göbel[1]

[1] *RPTU University Kaiserslautern-Landau, Germany,*
[2] *Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany*

✉ martin.becker@mv.rptu.de

ABSTRACT: Development of complex interdisciplinary products increases engineering challenges, that AI supported engineering approaches attempt to reduce by increasing automation. The resulting AI generated engineering artifacts, however, need to be classified, verified and managed to enable traceability and auditability of engineering decisions. This paper presents a classification and management approach for these artifacts, allowing verification of AI generated engineering artifacts. A use case on the iterative development of an e-bike demonstrates the approach.

KEYWORDS: artificial intelligence (AI), knowledge management, model-based systems engineering (MBSE), product lifecycle management (PLM)

1. Introduction

The development of interdisciplinary products, such as e-bikes and Smart Products, increases complexity (Abramovici & Stark, 2013) and requires improved engineering methods (Tomiyama et al., 2019). Interdisciplinarity creates multi-stakeholder value networks (Khan et al., 2022), whose diverse disciplines, and terminologies pohinder communication in Smart Product development (Kober et al., 2024). Rising regulatory complexity forces requirement alignment to ensure legal compliance and marketability (O'Dwyer & Cormican, 2017).Therefore, Smart Product development involves many disciplines, like electrical for e-motors, chemical for batteries, and mechanical for drivetrains in e-bikes. Additionally, project turnover often causes loss of tacit, siloed knowledge (Joe et al., 2013). Further development challenges are presented by persistent document-centric practices, data heterogeneity, and unstructured data (Kamm et al., 2021), which create knowledge silos. If prior engineering knowledge is not accessible or interoperable, organizations risk costly rework, especially in iterative product development typical for many modern products (Albers et al., 2015). This is due to the reason that engineering knowledge assets are not being optimized for reuse (Stenholm, 2018/2018).

Model-Based Systems Engineering (MBSE) promises a coherent and analyzable representation of requirements, structure, behavior, and verification. However, adoption remains uneven (Vogelsang et al., 2017), partly because modeling effort is high and common authoring tools are only loosely integrated with everyday artifacts, such as specifications, standards, and code. Product Lifecycle Management (PLM) systems, in turn, provide configuration and change control and are the established solutions for document and asset management. However, fragmented knowledge repositories, ownership and searchability issues (Stenholm & Bergsjö, 2020), can result in PLM repositories devolving into disconnected knowledge stores. The complementarity between MBSE and PLM is evident, but further work is needed to integrate them effectively (Gerhard et al., 2023).

Recent advances in Artificial Intelligence (AI), particularly Large Language Models (LLMs), provide practical support for development engineers throughout multiple phases of the product life cycle.

https://doi.org/10.1017/pds.2026.10575 Published online by Cambridge University Press

While potential applications for AI assistance in engineering face challenges, they are increasingly adopted into various engineering disciplines (Kretzschmar et al., 2024). This AI support leads to partially AI-generated AI knowledge artifacts, such as AI-generated SysML model fragments or requirements, or chats between engineers and AI. AI knowledge artifacts in this context are managed, traceable engineering outputs generated by AI systems during product development. As AI supports modeling and knowledge discovery, traceability and auditability of AI-generated content becomes critical, especially for regulated high-risk AI-systems (Ullah Shah et al., 2025). This paper introduces an approach that keeps AI-generated engineering knowledge traceable and controllable in product development. Knowledge is collected in a graph, pre-checked against vocabulary and rules, then reviewed by engineers. Only admitted items can be retrieved for question answering or used to generate models. Additionally, every output is stored with its origin and generation context for auditability.

The approach follows the Design Research Methodology (DRM) (Blessing & Chakrabarti, 2009) and, therefore, in section 2, clarifies the need for change in PLM and Model-Based Systems Engineering (MBSE). This is done by giving an overview of current approaches to integrate AI and LLMs into MBSE and PLM (Descriptive Study 1). The Prescriptive Study follows in section 3, presenting a concept to classify and improve AI output in the context of MBSE, as well as handle AI artifacts in product development using established approaches. Section 4 presents the Descriptive Study 2 and validates the concept by using the incremental development of an electric bicycle based on established knowledge of a classic, non-electric bike. Section 5 concludes the papers and gives an outlook into future research to further improve the concept.

2. State of the art

As a result of the described challenges of interdisciplinary, multistakeholder development, various approaches have been developed to manage the traceability of engineering decisions, holistic product development over various disciplines and knowledge reuse. Past research has focused on decision traceability, development overview, knowledge reuse, and task automation to reduce cost and development time and relieve engineers. Especially the approaches of PLM, MBSE (Bougain & Gerhard, 2020) and Knowledge-Based Engineering (KBE) are also relevant in this context, but the increased use of AI to support engineers must also be taken into account.

PLM enables engineers to manage requirements, product data, and documents in various forms and file formats, while also facilitating collaboration between different stakeholders by assigning permissions to specific documents. PLM can be improved by the application of AI (Wang et al., 2021), but while the paper discusses the mapping of AI to PLM tasks, the proposed applications omit managing AI artifacts needed for verification and validation. The issue of data siloing between different disciplines and engineering knowledge becoming inaccessible (Jokinen & Leino, 2019) therefore, remains unsolved.

The KBE approach encodes knowledge in rules, constraints, templates, and parametric CAD to guide design automation (Verhagen et al., 2012), and support multidisciplinary and mechatronic product development through specialized software and interface-based models (Mcharek et al., 2018; Zheng et al., 2018). The KBE approach enables engineers to systematically reuse design engineering knowledge, which can shorten design lead times, improve design consistency and facilitate a broader exploration of the design. However, existing industrial applications are still dominated by mechanical design, and the substantial effort, expertise and maintenance overhead required to elicit knowledge and implement KBE solutions, together with the dependence on specific tools and skills, continue to hinder more widespread adoption (Mcharek et al., 2018; Verhagen et al., 2012).

MBSE provides holistic modeling of interdisciplinary systems but lacks document management, faces adoption challenges due to complexity, and struggles with organizational resistance (Chami & Bruel, 2018) (Akundi et al., 2022). This hinders the reuse of MBSE models in iterative development (Henderson & Salado, 2024). PLM and MBSE, therefore, complement each other and need to be integrated to support engineering processes fully (Gerhard et al., 2023).

Ariffud (2025) showed the broad adoption of LLMs across an increasing number of disciplines including engineering. However, ensuring the traceability of engineering decisions and trust in AI-generated engineering artifacts remain challenges, which can result in the blocking of developed products (Mohsen Rezayat, 2025). In product development, AI enables multiple use cases to support engineering from

https://doi.org/10.1017/pds.2026.10575 Published online by Cambridge University Press

question answering to automatic model (Kretzschmar et al., 2024), across multiple languages, with somewhat reduced quality, even when discipline-specific nomenclatures are used (Meddeb et al., 2024). Natural Language Processing (NLP) methods have also been used to help engineers find existing engineering knowledge by comparing requirements between existing and new projects (Brünnhäußer et al., 2021). These approaches, however, have not been integrated into MBSE and PLM, resulting in limited use in established product development. Gräßler et al. (2023) used AI to extract requirements from requirements specifications in natural language, allowing for the much faster gathering of the required regulatory constraints a product faces. The resulting requirements, however, are not integrated into MBSE workflows and are not explicitly marked as AI-generated, limiting traceability and auditability. LLMs can be used for the automatic use case generation from requirements to support engineers during the product development (Schleifer et al., 2025). Even the automatic modeling of MBSE models exists (Apvrille & Sultan, 2024; DeHart, 2024), supporting engineers with time-consuming modeling tasks. Neither the resulting use case diagrams, nor the MBSE models are marked as AI-generated content, which again makes traceability difficult. LLMs face additional challenges, such as the potential to hallucinate (Anh-Hoang et al., 2025) and produce syntactically valid, but logically and content-wise flawed code and models. Retrieval Augmented Generation (RAG) (Lewis et al., 2020) may reduce these problems, by embedding text snippets from documents into the user prompt before sending it to the LLM. Purely embedding based RAG approaches have the disadvantage of losing complex relations typical in engineering. Graph-based RAG approaches have been introduced to improve this, such as GraphRag and LightRag (Edge et al., 2024; Guo et al., 2024), by preserving the complex relationships in a graph structure, resulting in improved contextual retrieval. Rogers & Crisan (2023) discuss the management of the collaborative human machine process, but the resulting auditable and traceable artifacts are not used in further engineering support to relieve engineers. Frameworks guiding and defining the work for engineers using AI product development exist, including process modeling (Kourani et al., 2024), and MBSE modeling assistance using the SysML v2 modeling language (Mollahassani et al., 2025). While the latter discusses the integration of MBSE and PLM with AI-assisted modeling, the AI-generated engineering knowledge artifacts are not classified, and no handling of these artifacts is described.

Due to the need to maintain trust in auditable and traceable engineering decisions and engineering knowledge artifacts for AI-assisted product development, the following research questions arise:

- Which artifacts of engineering knowledge are generated during AI-assisted MBSE?
- How can these artifacts be classified?
- What are the implications of trust and the usage of AI-generated engineering artifacts?

3. Approach

This section proposes an approach to address the challenges of describing, classifying, and managing both AI-generated and human-originating engineering knowledge artifacts, enabling the traceability of automated engineering decisions and increasing trust in AI-assisted product development. The approach combines automated information and knowledge extraction with human input, pre-validation, and ontology alignment, as well as the management of the generated engineering knowledge. The approach treats AI and human-generated engineering knowledge artifacts as items with explicitly recorded origins, trust levels, and admission rules. It integrates them with configuration and changes processes already established in enterprises. This enables classification, validation, and retrieval for additional AI support. The proposed approach consists of three phases, as illustrated in Figure 1 below. The **first phase establishes an engineering knowledge base** that preserves relationships across requirements, functions, structure, and verification, and is based on available heterogeneous knowledge, such as knowledge stored in documents and models managed in a PLM system. The **second phase involves aligning engineering knowledge** with an explicit ontology, enabling pre-validation against machine-readable constraints. The **third phase involves the final evaluation** of the knowledge by human experts, who decide whether to accept or reject the pre-validated knowledge. After this step, engineering knowledge becomes usable, allowing AI to support engineers with question-answering or modeling assistance.The approach assumes the existence of engineering knowledge from previous engineering efforts. This knowledge, in its various forms, will be reused and made available to the approach, containing the ontology and pre-existing development knowledge.

https://doi.org/10.1017/pds.2026.10575 Published online by Cambridge University Press

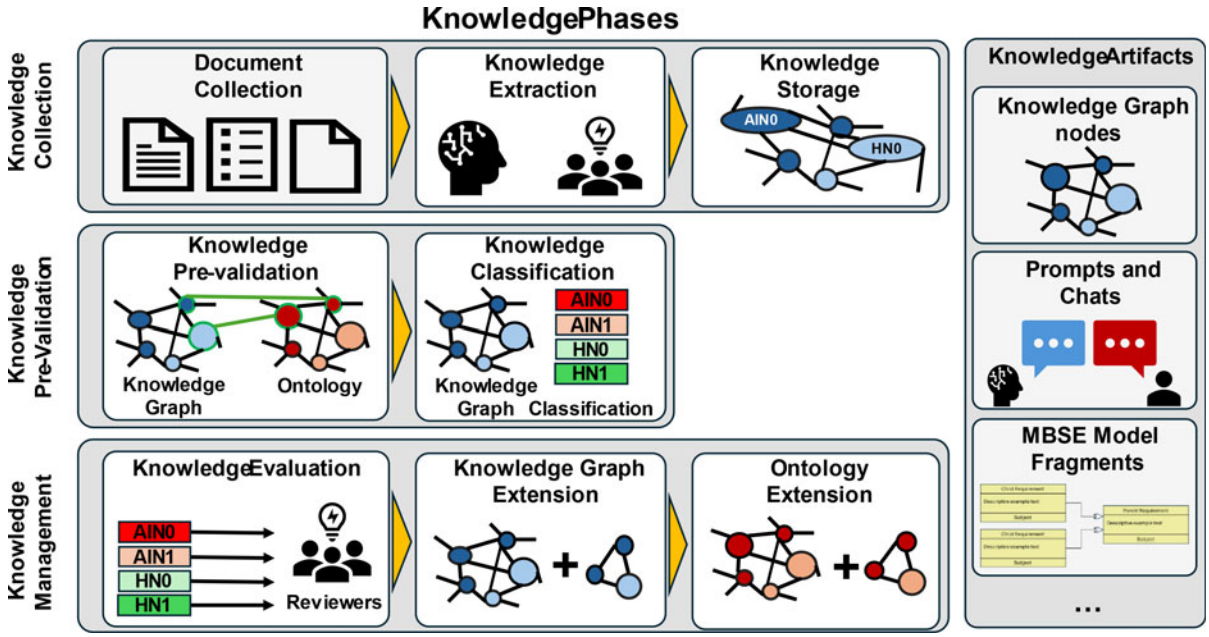


Figure 1. Process of classification and managing of AI engineering knowledge artifacts in MBSE

3.1. The knowledge collection phase

The approach begins with knowledge acquisition from sources already managed in PLM, such as documents, system models, test reports, parts lists, and change records, as well as system models from previous projects or ongoing product development, as shown in the upper phase of Figure 1. Automated extraction using a graph-oriented RAG Framework, such as LightRAG, identifies entities, relations, requirements, rationales, constraints, and verification evidence, and records them as semantic triples with attributes in a knowledge graph. This knowledge graph is then stored in a graph database such as Neo4j. As am example, BOMs and datasheets create *Battery* and *ElectricMotor* nodes and relations tagged AIN0. Engineers add their own domain knowledge, like recent process capabilities or supplier constraints, to enhance the engineering knowledge with implicit knowledge not contained in explicit knowledge and documents. At this stage, the graph, represented on the right side of the upper phase in Figure 1, is **intentionally flexible and open**, to capture what the organization **knows or suspects** rather than only what is already validated. However, this approach can and will generate false results. To preserve traceability and trust in the extracted knowledge, each node and edge created by the AI is tagged as "Artificial Intelligence Node 0" (AIN0), and each human assertion as "Human Node 0" or (HN0). These tags indicate a node's origin rather than its correctness and are a prerequisite for subsequent pre-validation, management, and use in further automated engineering tasks.

3.2. The knowledge pre-validation phase

In the second (middle) phase in Figure 1, pre-validation, aligns the unvalidated and flexible graph with an explicit project ontology to harmonize nomenclature and improve knowledge quality before the full review. The **alignment proceeds in three steps**. First, for each candidate node and relation, the system retrieves plausible ontology matches using lexical similarity, vector comparisons with manually created definitions, and neighborhood nodes in the local subgraph. Second, candidates are checked against Shape Constraint Language shapes that encode property constraints, data types, and additional specific rules expressed through the SPARQL Protocol and RDF Query Language for corner cases. Third, an optional ontology reasoner may verify the resulting assertions for logical consistency under an appropriate profile of the ontology language, detecting violations such as disjointedness or range conflicts. Nodes and relations that meet these criteria and **match the ontology are promoted in trust**. AI-originated items become Artificial Intelligence Node 1 (AIN1) artifacts. Human-originated items become Human Node 1 (HN1) artifacts. Items without an ontological equivalent remain at their current designation of AIN0 or HN0 and remain available for review as well. This requires extensive ontologies, which are rarely given. The following sub-section therefore provides an iteration step to

https://doi.org/10.1017/pds.2026.10575 Published online by Cambridge University Press

adapt incomplete ontologies. The mapping between graph and ontology is recorded explicitly using Simple Knowledge Organization System mapping relations, such as whether a node is an exactMatch, a closeMatch, broadMatch, relatedMatch or a narrowMatch help engineers assess alignment degree. A Battery as an example is an exactMatch, while alternative supplier modules are closeMatch to show alignment strength.

3.3. The knowledge management phase

Knowledge management places the pre-validated graph and its nodes under the control of human oversight. Each new or changed item becomes a managed artefact with a change request that follows established review workflows. Reviewers can accept an item, put it on hold, and request further evidence, or discard it while keeping the record for auditability and to prevent automatic regeneration of a knowledge artefact already marked as faulty, which would otherwise clutter the review process. Admission decisions determine eligibility for retrieval and model synthesis. **Only admitted items may be used** to ground question answering and to propose model changes. Discarded items remain accessible for investigation but are excluded from retrieval and generation. This policy ensures that AI assistance cannot circumvent the existing control points for design data.

Retrieval and model synthesis operate strictly on human-reviewed, admitted items. Question answering utilizes subgraph retrieval rather than isolated text chunks, ensuring responses reflect entities and typed relations that align with how systems engineers reason. Model proposals generate SysML v2 fragments that reference existing identifiers whenever possible and are constrained by the grammar to avoid invalid constructs. The generation of these models is facilitated by a combination of prompt engineering, grammar-constrained decoding, function calling, and parsing from other model structures.

Proposed changes are stored alongside the generative context bundle, a SysML v2 fragment for example is stored with the prompt, retrieved node IDs, model version, and decoding constraints, allowing reviewers to reconstruct the chain from the prompt and retrieved evidence to the output. If reviewers accept a proposal, the corresponding model files are updated, and the change is recorded as part of the legal record. The updated model, in turn, becomes an additional source in the subsequent acquisition cycle, so that the graph reflects the latest baseline. The result is a closed loop between PLM, knowledge graph, and system modelling. Sources under governance feed a living graph that is pre-validated against an ontology and that carries explicit provenance and trust. Retrieval and generation operate only on admitted items and produce outputs that are reviewable and reproducible. Review decisions update both the product lifecycle repository and the graph, which in turn improves future retrieval and reduces redundant development.

3.4. Generated knowledge artifacts

The approach generates different categories of artifacts that must be classified and managed throughout all phases of knowledge processing, as shown on the right side of Figure 1. These are primary engineering knowledge artifacts, secondary engineering artifacts that may be part of a primary artifact and provide further detail, and tertiary artifacts that are the result of AI-generated modeling. **The primary artifacts represent the nodes** of the knowledge graph and can be categorized into one of four, the previously mentioned sub-categories AIN0, AIN1, HN0, and HN1. These artifacts, although of different origins and varying levels of trust, all share the same set of values. While the primary artifacts remain during and after the human validation step, they change in state from non-validated to either accepted, awaiting further evidence, or discarded. **The second category of artifacts** are system prompts, user prompts, source document identifiers, mapped ontology concepts, reasoner logs, and further automatically generated data. **The third category includes generated documents and models**, such as SysML v2 models, as well as code and code snippets. AI Chats resulting from pure question-answering are also included in this category, as they represent the final product of AI assistance. All categories influence engineering outcomes and therefore require versioning, access control, and audit trails, like those for drawings and specifications.

To trace each artifact correctly, a minimum set of attributes must be associated with each artifact. Each primary artefact must carry a minimal metadata set to support traceability, quality control, and selective retrieval. These attributes comprise secondary artifacts and metadata. For primary knowledge items the

https://doi.org/10.1017/pds.2026.10575 Published online by Cambridge University Press

required fields are origin class as AI or human, validation status at level zero or level one, identifiers of source documents or models with version and location, the ontology concept or relation to which the item maps and the mapping type, confidence scores for extraction and alignment, timestamps, author or system identity, and links to related items in the graph. Furthermore, user and system prompts, retrieved passages, the identifier and version of the model, decoding constraints, shape validation reports, reasoner logs, mapping decisions and confidence scores, and reviewer comments and decisions. For generative context, the required fields are the model family and version, the exact prompt text, the identifiers of retrieved graph items and documents, decoding parameters or grammar constraints, and the output artefact identifier if a model fragment or text answer is produced. For validation evidence, the required fields are the shape profile used, pass or fail, and diagnostic messages, the reasoner profile and outcomes, reviewer identity and decision, and the change request identifier. This metadata enables fine-grained queries such as retrieving only human-originated and validated requirements with exact ontology matches from a particular project phase or excluding AI-originated items from safety-related change proposals. Since tertiary items represent mostly established files and documents, they are handled accordingly without further adaptation.

The product lifecycle workflow requires several adjustments to integrate these practices without disrupting established controls. First, the repository must introduce new item types for primary knowledge artifacts that contain a multitude of different attributes of variable length, in the form of relations, prompts, and retrieval bundles, all with proper lifecycle states and permissions. Second, change processes must be extended so that any new or modified knowledge item triggers an automatic change request with routing to the responsible role. Third, admission rules must be defined at the policy level. For example, only HN1 artifacts may influence safety-related configurations, while AI1 artifacts may be used for exploratory studies and for non-critical documentation drafts. Fourth, views must be added to the repository so that engineers can navigate from a requirement in the system model to the corresponding knowledge items, their sources, and their validation status. Fifth, the retrieval and generation services must enforce policy at the interface and provide the option to calibrate the trust level for retrieval and generation. They must only expose admitted items to generative models and must attach the complete context bundle to any generated output that proposes a model or document change. These changes embed AI assistance into the heart of configuration and change control rather than leaving it as an external tool. The curated knowledge graph and its engineering knowledge artifacts enable AI to answer questions in various languages related to specific problems that interdisciplinary and international engineering teams may face. Additionally, due to the retrieval of structured and interconnected information for the LLM, more complex modeling tasks can be fulfilled, and system engineers can be supported much better.

4. Instantiation

This section demonstrates the iterative development of an electric bike using the presented approach. The implementation assumes the availability of documents and models from the prior development cycle. It demonstrates how an enhanced ontology, a graph-based retrieval layer, and an AI-supported modeling approach transform past development data into reusable and actionable knowledge. Additionally, the created artifacts are managed across their lifecycle, and a SysML v2 system model is automatically updated.

In the first step, the specification sheets, test reports, bills of materials, discipline-specific models, and other documents are imported from the previous conventional bicycle development with their version identifiers and source locations. Simultaneously, new information regarding stakeholder requirements, legal requirements, customer requirements, and market requirements is added. Examples of this include the target price of approximately 2,500 euros, legal limits for motor assistance at 25 km/h, customer studies mandating a minimum range of 40 miles, and engineering targets requiring a mass of less than 25 kilograms. A *LightRAG*-style pipeline ingests the documents, as shown in the left panel of Figure 2, and produces an initial flexible knowledge graph that records candidate entities and relations. All automatically extracted nodes and edges are tagged as AIN0, and contributions from engineers not found in documents are tagged as HN0 to preserve origin and to create an explicit queue for pre-validation.

https://doi.org/10.1017/pds.2026.10575 Published online by Cambridge University Press

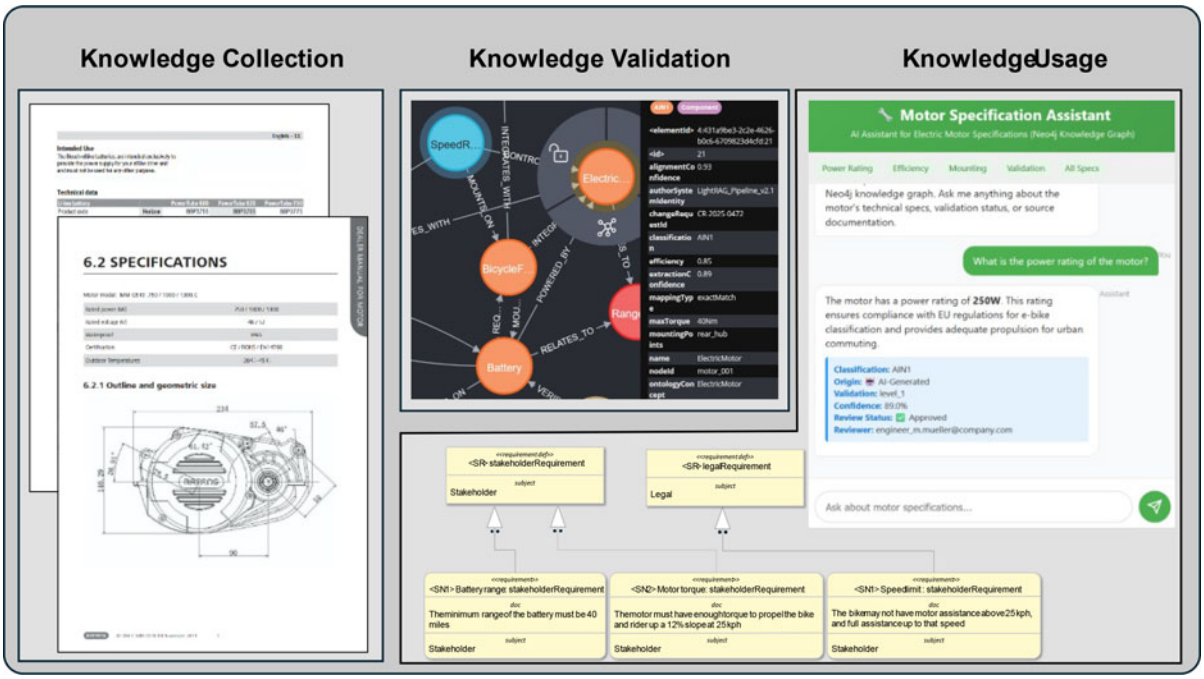


Figure 2. Results of the applied process at different stages

Examples for human-added knowledge are changes in tube hydroforming capabilities that affect down-tube geometry, and a shortlist of candidate battery modules and hub motors available in the next sourcing window. These HN0 assertions are linked to the affected structure and requirement nodes. At this stage, the flexible and unvalidated graph is not used to answer engineering queries or to drive model updates. The purpose is to capture what the organization knows, suspects, or intends, before any admission decisions. Additionally, the ontology is updated in the accompanying PLM-System after the required change request is approved. In this case, the battery, the electric motor, a power selector, as well as a control chip are added as features that an electric bicycle must have. This adapted ontology serves as the guideline for pre-validation in the new product development. The permissive graph is aligned with the project ontology, which defines the vocabulary for bicycle systems, subsystems, and relationships. Three steps are executed. First, candidate matches are computed between graph labels and ontology terms using lexical and vector similarity, while considering neighborhood signals such as component-to-requirement links. Second, Shape Constraint Language shapes verify datatypes and required properties for classes such as Battery, *ElectricMotor*, and *SpeedController*. Items that satisfy shapes and remain consistent are promoted in trust resulting in AIN0 becoming AIN1, and HN0 becoming HN1. Figure 2 shows the nodes with recorded mapping relations are recorded explicitly using *Simple Knowledge Organization System* so that exactMatch, closeMatch, broadMatch, or narrowMatch are visible during later reviews. Non-conforming items remain at level zero and cannot influence retrieval or generation. In this case, Battery and *ElectricMotor* nodes align as exact matches, several supplier modules as close matches, while the engineer-added *SpeedRegulator* lacks an ontology counterpart and remains HN0 pending ontology extension or rejection. All items retain source pointers and validation artifacts like shape reports and reasoner logs. After the pre-validation the battery node contains its id, AIN1 classification, exact match alignment scores, and document source, while the speed regulator carries the employee ID, enabling traceability during review cycles.

Every new or changed knowledge item creates a change request in PLM. Reviewers examine the pre-validation bundle, which includes extracted text spans, ontology mappings, Shape Constraint Language diagnostics, and reasoner outcomes. Approved items transition to admitted state, capturing who approved what, when, and on what evidence. Rejected items persist marked as discarded, preventing duplicate proposals in the knowledge extraction phase. Admitted AIN1 and HN1 items become eligible for retrieval and for substantiating subsequent model proposals. The *SpeedRegulator* and the separate ontology change request are declined, so the item is discarded and excluded from retrieval, question answering, and model generation. With accepted items in place, engineers begin to interact with the knowledge base through RAG during chats or requests for modeling assistance, as shown on the right side of Figure 2. A frame engineer requests motor power constraints using LLM chat functionalities.

https://doi.org/10.1017/pds.2026.10575 Published online by Cambridge University Press

Subgraph retrieval returns the information as well as its origin, reviewer and approval status. Because only admitted items are exposed, no non-validated concept, such as the *speedRegulator*, can leak into the answer. A powertrain engineer queries the legal top speed and assistance cut-off logic. When he asks the AI for help, the retrieval works on the knowledge graphs' approved items by looking for synonyms and connected nodes of *topSpeed* before supplying the found information to the LLM. This returns the admitted requirement for motor assistance up to 25 kilometers per hour. The response includes links back to the legal documents and in case of the motor to the technical data sheets of the purchasable models. If necessary, the engineer can access the related documents in PLM. These answers guide the definition of electrical and mechanical architecture, while keeping the chain of evidence intact.

The same retrieval context is used to generate SysML v2 fragments, also shown in Figure 2 at the bottom right side. User prompts reference the admitted *Battery* and *ElectricMotor* nodes, and grammar-constrained decoding produces proposals for the requested requirements. Identifiers of existing model elements are reused where possible to avoid duplication. Each proposal is stored together with its generative context, including the exact prompt, retrieved node identifiers, model version, and decoding constraints, so that reviewers can reconstruct how the output was produced. Proposals do not modify authoritative models until accepted through review. In this iteration, reviewers accept the Battery block with its parameter set and the allocation of the range requirement to the energy store and motor efficiency parameters. Accepted changes update the SysML v2 repository, enriching the knowledge graph in the next ingestion cycle, so that future retrieval reflects the current baseline. Throughout the iteration, all artifacts are handled, and primary knowledge items in the graph carry origin, validation level, ontology mapping type, extraction and alignment confidence, timestamps, authorship document identifiers to PLM sources. Generative context bundles record the model family and version, prompt text, retrieved node identifiers, and decoding constraints. Validation evidence stores shape profiles, pass or fail status, reasoner outcomes, and reviewer decisions.

The engineering knowledge artifacts therefore have been described and classified. The origin of AI generated engineering knowledge can also be traced, resulting in reduced ambiguity and increased trust in AI support during product development, as engineers can now manage the artifacts and control the knowledge base on which AI operates on a finer level. These metadata allow selective retrieval, for example, excluding AIN1 items from safety-related proposals or limiting queries to HN1 items with exact ontology matches during compliance documentation. Secondary engineering artifacts, such as documents and SysML v2 models, remain in their established PLM lifecycles but are linked to the knowledge items that influenced them.

5. Conclusion and outlook

AI-generated engineering artifacts in the context of product development under increasing regulatory and interdisciplinary pressures have been defined and classified and the management of such artifacts has been shown. The approach defined engineering knowledge artifacts with explicit origins and validation levels, captured the generative context and validation evidence required for AI assistance, and enforced admission into the lifecycle system, ensuring that only items reviewed and accepted by human experts are improved retrieval and model synthesis. The instantiation of electrifying a bicycle demonstrated that the approach reduces redundant development, improves traceability under regulatory pressure, and updates the SysML v2 baseline without bypassing configuration or change control.

Critically, the scientific contribution provides an explicit framework for managing AI-generated artifacts to support trustworthy and traceable AI integration into daily engineering work, by making the origin, validation, and generative context of artifacts auditable.

While the process may appear complex, most steps, including the majority of the extraction, networking, and pre-validation of engineering knowledge, are automated, thereby minimizing extra workload for engineers. Networking knowledge in RDF triples enables the codification of complex relationships between product aspects, improving reusability for future iterations. Over time, this can reduce development costs and duration by reducing miscommunication and avoiding redundant parallel efforts.

Future research should discuss and evaluate various approaches to generating models based on networked knowledge. Multiple approaches are theoretically usable to generate valid SysMLv2 code in its textual notation, such as prompt engineering, grammar-constrained decoding, function calling, or parsing SysML v2 code from a more established notation in the LLM training data. Which approach produces the best models is the most efficient, and how to evaluate the different approaches must be discussed.

https://doi.org/10.1017/pds.2026.10575 Published online by Cambridge University Press

References

Abramovici, M., & Stark, R. (2013). *Smart Product Engineering*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-30817-8

Akundi, A., Ankobiah, W., Mondragon, O., & Luna, S. (2022). Perceptions and the extent of Model-Based Systems Engineering (MBSE) use – An industry survey. In *2022 IEEE International Systems Conference (SysCon)* (pp. 1–7). IEEE. https://doi.org/10.1109/SysCon53536.2022.9773894

Albers, A., Bursac, N., & Wintergerst, E. (2015). Produktentgenerationsentwicklung - Bedeutung und Herausforderungen aus einer entwicklungsmethodischen Perspektive. In *Stuttgarter Symposium für Produktentwicklung 2015*.

Anh-Hoang, D., Tran, V., & Nguyen, L.-M. (2025). Survey and analysis of hallucinations in large language models: Attribution to prompting strategies or model behavior. *Frontiers in Artificial Intelligence*, 8, 1622292. https://doi.org/10.3389/frai.2025.1622292

Apvrille, L., & Sultan, B. (2024). System Architects Are not Alone Anymore: Automatic System Modeling with AI. In *Proceedings of the 12th International Conference on Model-Based Software and Systems Engineering* (pp. 27–38). SCITEPRESS - Science and Technology Publications. https://doi.org/10.5220/0012320100003645

Ariffud, M. (2025). *LLM statistics 2025: Comprehensive insights into market trends and integration*. https://www.hostinger.com/tutorials/llm-statistics

Blessing, L. T., & Chakrabarti, A. (2009). *DRM, a Design Research Methodology*. Springer London. https://doi.org/10.1007/978-1-84882-587-1

Bougain, S., & Gerhard, D. (2020). Supporting product design decision with a SysML design history Assistant. *Procedia CIRP*, 91, 255–260. https://doi.org/10.1016/j.procir.2020.02.174

Brünnhäußer, J., Lindow, K., Lünnemann, P., Kirsch, L., & Wrasse, K. (2021). KI in PLM: Automatisches Vergleichen von Anforderungen in natürlicher Sprache. In *ProduktDatenJournal* (pp. 58–65).

Chami, M., & Bruel, J.-M. (Eds.) (2018). *A Survey on MBSE Adoption Challenges*.

DeHart, J. K. (2024). Leveraging Large Language Models for Direct Interaction with SysML v2. *INCOSE International Symposium*, 34(1), 2168–2185. https://doi.org/10.1002/iis2.13262

Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., Metropolitansky, D., Ness, R. O., & Larson, J. (2024). *From Local to Global: A Graph RAG Approach to Query-Focused Summarization*. https://doi.org/10.48550/arXiv.2404.16130

Gerhard, D., Salas Cordero, S., Vingerhoeds, R., Sullivan, B. P., Rossi, M., Brovar, Y., Menshenin, Y., Fortin, C., & Eynard, B. (2023). MBSE-PLM Integration: Initiatives and Future Outlook. In F. Noël, F. Nyffenegger, L. Rivest, & A. Bouras (Eds.), *IFIP Advances in Information and Communication Technology. Product Lifecycle Management. PLM in Transition Times: The Place of Humans and Transformative Technologies* (Vol. 667, pp. 165–175). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-25182-5_17

Gräßler, I., Özcan, D., & Preuß, D. (2023). AI-based extraction of requirements from regulations for automotive engineering. In *DS 125: Proceedings September 2023* (pp. 163–172). https://doi.org/10.35199/dfx2023.17

Guo, Z., Xia, L., Yu, Y., Ao, T., & Huang, C. (2024). *LightRAG: Simple and Fast Retrieval-Augmented Generation*. http://arxiv.org/pdf/2410.05779v3

Henderson, K., & Salado, A. (2024). The Effects of Organizational Structure on MBSE Adoption in Industry: Insights from Practitioners. *Engineering Management Journal*, 36(1), 117–143. https://doi.org/10.1080/10429247.2023.2210494

Joe, C., Yoong, P., & Patel, K. (2013). Knowledge loss when older experts leave knowledge-intensive organisations. *Journal of Knowledge Management*, 17(6), 913–927. https://doi.org/10.1108/JKM-04-2013-0137

Jokinen, L., & Leino, S.-P. (2019). Hidden product knowledge: problems and potential solutions. *Procedia Manufacturing*, 38, 735–744. https://doi.org/10.1016/j.promfg.2020.01.099

Kamm, S., Jazdi, N., & Weyrich, M. (2021). Knowledge Discovery in Heterogeneous and Unstructured Data of Industry 4.0 Systems: Challenges and Approaches. *Procedia CIRP*, 104, 975–980. https://doi.org/10.1016/j.procir.2021.11.164

Khan, I. S., Kauppila, O., Fatima, N., & Majava, J. (2022). Stakeholder interdependencies in a collaborative innovation project. *Journal of Innovation and Entrepreneurship*, 11(1). https://doi.org/10.1186/s13731-022-00229-0

Kober, C., Medina, F. G., Benfer, M., Wulfsberg, J. P., Martinez, V., & Lanza, G. (2024). Digital Twin Stakeholder Communication: Characteristics, Challenges, and Best Practices. *Computers in Industry*, 161, 104135. https://doi.org/10.1016/j.compind.2024.104135

Kourani, H., Berti, A., Schuster, D., & van der Aalst, W. M. P. (2024). Process Modeling with Large Language Models. In H. van der Aa, D. Bork, R. Schmidt, & A. Sturm (Eds.), *Lecture Notes in Business Information Processing. Enterprise, Business-Process and Information Systems Modeling* (Vol. 511, pp. 229–244). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-61007-3_18

Kretzschmar, M., Dammann, M., Schwoch, S., Braun, F., Saske, B., & Paetzold-Byhain, K. (2024). Evaluating the Role of Generative AI in Product Development and Design - A Systematic Review. In *Proceedings of NordDesign 2024 2024* (pp. 21–30). https://doi.org/10.35199/NORDDESIGN2024.3

https://doi.org/10.1017/pds.2026.10575 Published online by Cambridge University Press

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020, May 22). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. https://arxiv.org/pdf/2005.11401v4

Mcharek, M., Azib, T., Hammadi, M., Choley, J.-Y., & Larouci, C. (2018). Knowledge sharing for mechatronic systems design and optimization. *IFAC-PapersOnLine*, 51(11), 1365–1370. https://doi.org/10.1016/j.ifacol.2018.08.338

Meddeb, A., Lüken, S., Busch, F., Adams, L., Ugga, L., Koltsakis, E., Tzortzakakis, A., Jelassi, S., Dkhil, I., Klontzas, M. E., Triantafyllou, M., Kocak, B., Yüzkan, S., Zhang, L., Hu, B., Andreychenko, A., Yurievich, E. A., Logunova, T., Morakote, W., … Bressem, K. (2024). Large Language Model Ability to Translate CT and MRI Free-Text Radiology Reports Into Multiple Languages. *Radiology*, 313(3), e241736. https://doi.org/10.1148/radiol.241736

Mohsen Rezayat. (2025). *The importance of considering ethical challenges and balancing speed vs. trustworthiness during AI implementation for digital industries: Part 1*. https://blogs.sw.siemens.com/thought-leadership/the-importance-of-considering-ethical-challenges-and-balancing-speed-vs-trustworthiness-during-ai-implementation-for-digital-industries-part-1

Mollahassani, D., Becker, M., Eickhoff, T., Pickel, J., Goetz, S., Wartzack, S., & Göbel, J. C. (2025). Framework for circular AI-driven model-based systems engineering. *Proceedings of the Design Society*, 5, 1883–1892. https://doi.org/10.1017/pds.2025.10202

O'Dwyer, C., & Cormican, K. (2017). Regulation - Do or Die: An Analysis of Factors Critical to New Product Development in a Regulatory Context. *Journal of Technology Management & Innovation*, 12(1), 26–38. https://doi.org/10.4067/S0718-27242017000100004

Rogers, J., & Crisan, A. (2023). *Tracing and Visualizing Human-ML/AI Collaborative Processes through Artifacts of Data Work.* https://doi.org/10.48550/arXiv.2304.02699

Schleifer, S., Lungu, A., Kruse, B., Goetz, S., & Wartzack, S. (2025). Large Language Model-based Generation of Use Case Diagrams from Requirement Specifications. *INCOSE International Symposium*, 35(1), 261–275. https://doi.org/10.1002/iis2.70015

Stenholm, D. (2018). *Reuse of engineering knowledge* [Dissertation, Chalmers tekniska högskola]. 978-91-7597-814-7.

Stenholm, D., & Bergsjö, D. (2020). Barriers to reuse of codified engineering knowledge in product development - a literature review. *International Journal of Product Lifecycle Management*, 12(4), Article 112773, 307. https://doi.org/10.1504/IJPLM.2020.112773

Tomiyama, T., Lutters, E., Stark, R., & Abramovici, M. (2019). Development capabilities for smart products. *CIRP Annals*, 68(2), 727–750. https://doi.org/10.1016/j.cirp.2019.05.010

Ullah Shah, S. T., Hussein, M., Barcomb, A., & Moshirpour, M. (2025). Explainability as a Compliance Requirement: What Regulated Industries Need from AI Tools for Design Artifact Generation. In *2025 IEEE 33rd International Requirements Engineering Conference Workshops (REW)* (pp. 13–22). IEEE. https://doi.org/10.1109/REW66121.2025.00007

Verhagen, W. J., Bermell-Garcia, P., van Dijk, R. E., & Curran, R. (2012). A critical review of Knowledge-Based Engineering: An identification of research challenges. *Advanced Engineering Informatics*, 26(1), 5–15. https://doi.org/10.1016/j.aei.2011.06.004

Vogelsang, A., Amorim, T., Pudlitz, F., Gersing, P., & Philipps, J. (2017). *Should I Stay or Should I Go? On Forces that Drive and Prevent MBSE Adoption in the Embedded Systems Industry.* https://doi.org/10.48550/arXiv.1709.00266

Wang, L., Liu, Z., Liu, A., & Tao, F. (2021). Artificial intelligence in product lifecycle management. *The International Journal of Advanced Manufacturing Technology*, 114(3–4), 771–796. https://doi.org/10.1007/s00170-021-06882-1

Zheng, C., Bricogne, M., Le Duigou, J., Hehenberger, P., & Eynard, B. (2018). Knowledge-based engineering for multidisciplinary systems: Integrated design based on interface model. *Concurrent Engineering*, 26(2), 157–170. https://doi.org/10.1177/1063293X17734591

https://doi.org/10.1017/pds.2026.10575 Published online by Cambridge University Press