Proceedings of the Design Society, Volume 6: DESIGN 2026
https://doi.org/10.1017/pds.2026.10643

Enabling AI-supported requirements engineering through model-based systems engineering and characteristics-properties modeling

Fabian Rahn✉, Anubhab Majumder and Eckhard Kirchner

Technical University of Darmstadt, Germany

✉ fabian.rahn@tu-darmstadt.de

ABSTRACT: High-quality requirements are essential for successful product development. This work proposes a model-based requirements engineering framework and AI-supported tool. The framework links design characteristics and measured properties via an OPM-based system model. This enables the implementation of a tool for systematic verification and validation of requirements in early product development stages, supporting the transition from experience- to data-/evidence-driven decision making and industry 4.0 paradigms. A hydraulic-press case-study demonstrates feasibility of the end-to-end workflow.

KEYWORDS: characteristics and properties, model-based systems engineering (MBSE), requirements engineering (RE), knowledge graph, large language model (LLM)

1. Introduction

Requirements are the foundation of product development (PD) as they define assignable and measurable development tasks and provide information about the development process (Bender & Gericke, 2021). In PD, requirements derived from stakeholder needs are typically expressed as functional requirements or desired properties of a system. To fulfil these, the development team defines characteristics or design parameters that are expected to result in the desired properties (Weber, 2008). However, the actual properties of a system can only be measured during its real-world usage, often long after development decisions are made.

This creates a significant ***gap***: While operational data may be available through digital twins or closed-loop engineering approaches, this feedback is rarely represented in a structured, model-based form that supports early PD (Kiritsis, 2011; Kritzinger et al., 2018). If measured property data from existing systems can systematically be linked to their corresponding design characteristics in a suitable data structure, this can be leveraged to support requirement validation in early design phases.

The ***challenge*** mainly lies in creating this link and enabling its systematic use. This enables a shift from experience-driven to data-driven requirement validation and refinement, improving innovation outcomes while reducing cost and time.

The ***goal*** of this contribution is to develop a framework for an artificial intelligence (AI)-supported model-based requirements engineering tool allowing systematic linking of design characteristics and measured properties of the product in use, ultimately enabling data-driven requirement validation and refinement of requirements during the early PD phase. The scope of this work aims at product line extensions and incremental product improvements rather than disruptive innovation.

To achieve this goal, the following ***research questions*** will be answered:

1. How can data generated or measured in the usage phase of a system or product be captured and structured efficiently with characteristics-properties modeling (CPM) in mind?
2. How can the structured data be leveraged to identify and relate characteristics and properties?

https://doi.org/10.1017/pds.2026.10643 Published online by Cambridge University Press

3. How can these relations support automated verification, validation, and refinement of requirements?

The following sections introduce relevant foundations before outlining the proposed framework and a case study.

2. Fundamentals and related work

2.1. Requirements engineering and management

Requirements Engineering and Management (RE & RM) form the foundation of a systematic PD process, ensuring that the final product meets all stakeholder expectations. The following subsections define requirement verification and validation, list quality characteristics of requirements, and introduce the concept of characteristics and properties, which is key for formulating high-quality requirements. In this work, we mainly focus on qualitative product requirements.

2.1.1. Verification and validation

Requirements *verification* is the process of confirmation, by examination, that the requirements are well-formed. During this process, requirements are compared to a defined set of quality characteristics (see Subsection 2.1.2), which serve as guidelines for writing high-quality requirements. Requirements *validation*, on the other hand, is the confirmation that the requirements define the correct system as intended by the stakeholders, ensuring that the specified needs and expectations can be achieved. (ISO 29148, 2018)

2.1.2. Requirement quality characteristics

In RE, several standards and publications define quality characteristics. Mavin et al. (2009) from Rolls-Royce PLC developed and published an 'Easy Approach to Requirements Syntax' (EARS), adapted by many organizations. They state that requirements are typically written in unconstrained natural language (NL), resulting in imprecise requirements, propagating problems, eventually leading to higher development cost. Hence, a basic syntax was proposed, and a small number of specific EARS patterns were introduced, providing structured guidance for writing high-quality textual requirements.
While EARS suggests specific patterns for requirement syntax, there are other sources explicitly listing quality characteristics. For example, the INCOSE guide for writing requirements (INCOSE Requirements Working Group, 2023) and the ISO/IEC 29148 standard (ISO 29148, 2018) agree on 9 quality characteristics, listed in Table 1.

Table 1. Requirement quality characteristics summarized from INCOSE and ISO/IEC (INCOSE Requirements Working Group, 2023; ISO 29148, 2018)

Characteristic	Description
Appropriate	Reflects the intent and detail at the right level of abstraction, avoiding unnecessary constraints.
Complete	Fully describes the capability, characteristic, constraint, or quality factor without needing external information.
Conforming	Complies with approved standards, patterns, and style guides for writing and managing requirements.
Correct	Accurately represents the concept, source, or higher-level requirement from which it was derived.
Feasible	Can be realized within constraints (cost, schedule, technical, safety, etc.) with acceptable risk.
Necessary	Defines an essential capability, characteristic, constraint, or quality factor.
Singular	States a single capability, characteristic, constraint, or quality factor (though conditions may exist).
Unambiguous	Worded so it can be interpreted in only one way, clearly understood by all stakeholders.
Verifiable/ Validatable	Structured and worded so realization can be proven to the customer's or authority's satisfaction.

https://doi.org/10.1017/pds.2026.10643 Published online by Cambridge University Press

2.1.3. Characteristics and properties

For a clear distinction between characteristics and properties, the definition established by Weber et al. (2003) is used in this contribution, see Figure 1. Accordingly, characteristics describe the structure, shape and material consistency of a product. They can be directly influenced or determined by the designer (e.g. material, shape, dimensions, etc.). Properties, on the other hand, describe the product's behavior (e.g. weight, safety and reliability, aesthetic properties, but also things like manufacturability, assemblability, testability, environmental friendliness, cost of a product, etc.). They cannot be directly influenced by the designer; they are rather the result of the interplay between different characteristics. This is commonly referred to as characteristics-properties modeling (CPM) (Weber, 2008).

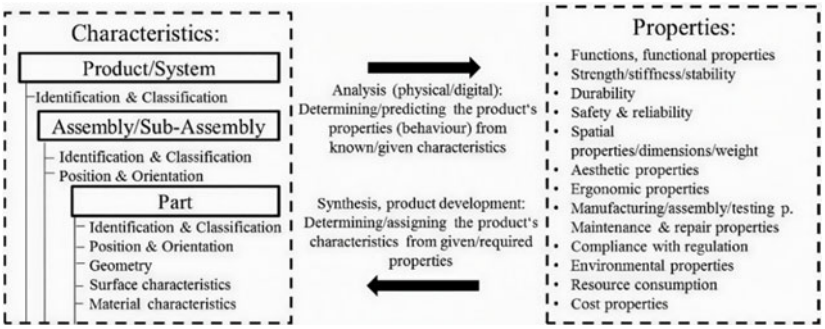


Figure 1. Characteristics-properties modeling reproduced from Buchert et al. (2016)

2.2. Model-based systems engineering (MBSE)

MBSE is the concept of using formalized models throughout the life cycle of a system, supporting requirements, design, analysis, verification, and validation activities (Friedenthal et al., 2007). It is one of the key enablers for managing complexity in PD. There are various modeling languages available for MBSE, such as object process methodology (OPM) or systems modeling language (SysML) as an extension of unified modeling language (UML).

2.2.1. Object-process methodology (OPM)

OPM is a unified and standardized (ISO 19450, 2024) conceptual modeling language used for describing the function, structure, and behavior of systems in a human-readable and machine-interpretable format. This unified approach supports transparent reasoning, improves communication, and enables systematic knowledge capture. It can be applied to both artificial and natural systems across various domains. A unique feature of OPM is its bimodality: It can be represented as graphical models using object-process diagrams (OPD) and as natural language using the object-process language (OPL).

2.2.2. OPM in requirements engineering

Several publications have demonstrated the added value of applying OPM in RE, allowing for enhanced traceability, early-stage verification, and the systematic refinement of requirements. This demonstrates the potential for data-driven, model-supported requirement management. Mordecai and Dori (2017) coined the term model-based requirement engineering (MBRE) as the integration of OPM within MBSE to capture stakeholder intent and system requirements in a unified model, making use of the bimodality of OPM. Kang et al. (2025) further introduce neuro-conceptual AI, demonstrating that OPM models can serve as interpretable knowledge structures to guide reasoning. Together, these works substantiate OPM's role as both a modeling and reasoning language in model-based, AI-supported RE.

3. Methods

3.1. Proposed mapping framework and RE tool

OPM is used to visualize the proposed CPM-framework and RE tool in Figure 2. The central mapping process requires measured properties obtained from the product in use. An expert maps these properties to design characteristics, yielding a characteristic-property knowledge base. Design characteristics are defined in a system model following CPM, realized using OPM and shown in Figure 3. The resulting

https://doi.org/10.1017/pds.2026.10643 Published online by Cambridge University Press

knowledge base is represented as a characteristic-property knowledge graph (CPKG), consisting of nodes and edges. Nodes represent characteristics and properties; edges encode the relationships between them. This addresses RQ1 by structuring usage-phase data as properties within the CPKG, following CPM. RQ2 is addressed through an explicit expert-driven mapping/modeling process and storage of the identified relations between characteristics and properties in the CPKG, enabling their identification and systematic retrieval.

The proposed RE tool accesses the CPKG to retrieve information not only to answer user questions about graph content or identify characteristics and properties, but also to generate, verify, validate, rephrase, and refine requirements as well as detect and convert unit-value pairs and comparators.

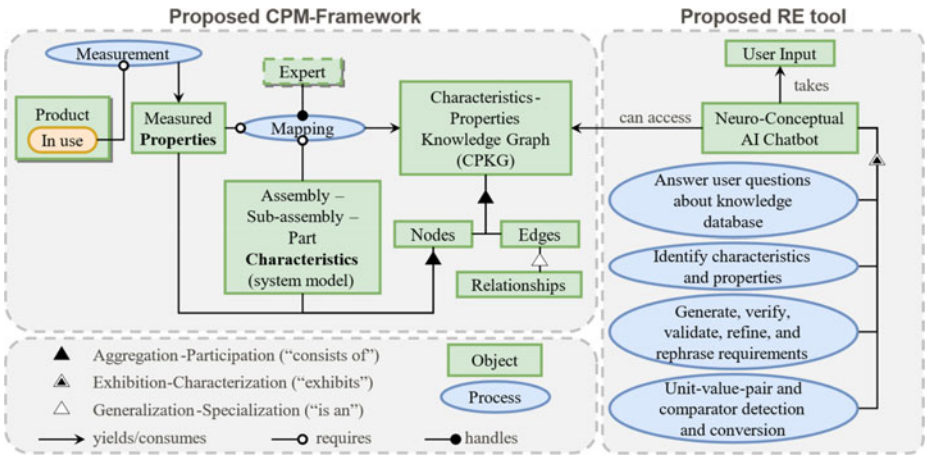


Figure 2. Proposed CPM-framework and AI-supported RE tool

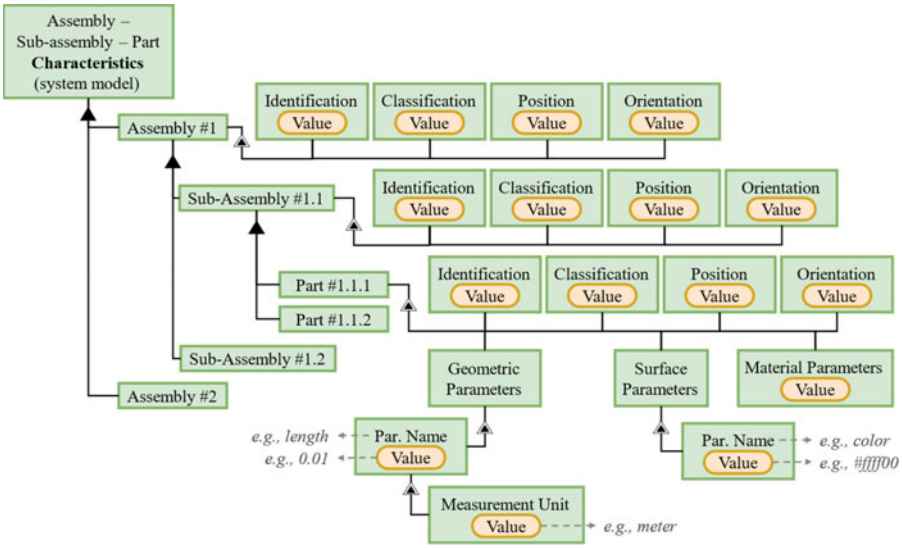


Figure 3. Zoom-in to the system model following CPM realized with OPM, providing the design characteristics of the product of interest for the central mapping process (see Figure 1)

3.2. Case study - hydraulic forming press

To demonstrate a proof-of-concept of the proposed framework and tool, a case study is conducted based on the example of a hydraulic forming press. Details of this case study are presented in the following.

3.2.1. Case study workflow

The case study is based on an online sourced datasheet of a hydraulic press by DAKE, (Metro Hydraulic Jack Co., 2025), more specifically, the model 'Elec-draulic I Hydraulic Press'. Statements and values from the datasheet were used as proxy properties, as no actual measurements were available.

https://doi.org/10.1017/pds.2026.10643 Published online by Cambridge University Press

The datasheet contains all necessary information to establish a system model containing characteristics, to map qualitative and quantitative properties and eventually build the CPKG. An example for a qualitative property is 'reliable pressing', quantitative properties include weight of the system and power of different models, and characteristics include the part dimensions. OPCAT (Dori et al., 2003), an open-source system modeling tool, was used to manually model the system and map the properties to its characteristics. From the created OPD, the OPCAT-generated OPL was taken and parsed into Cypher code using a custom rule- and regex-based Python script to subsequently use this Cypher code to build the knowledge graph in a local Neo4j instance. This end-to-end workflow is depicted in Figure 4.

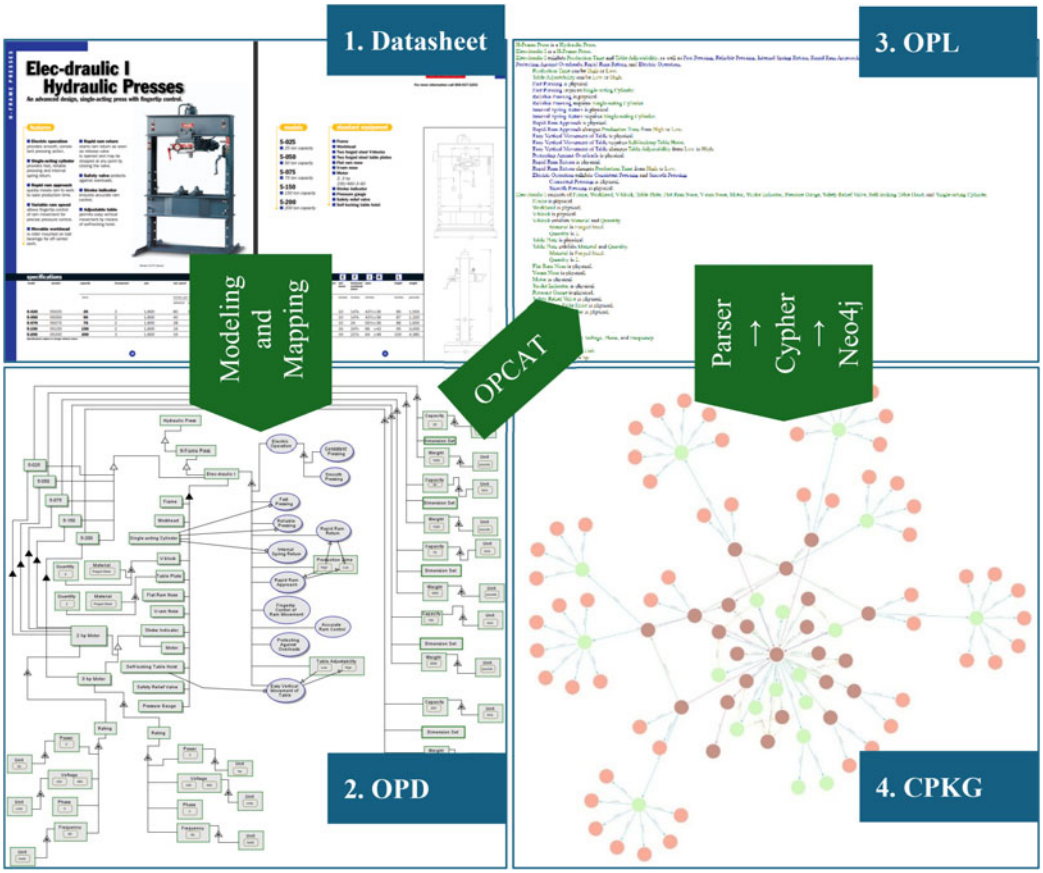


Figure 4. Case study workflow starting with manual generation (modeling and mapping) of an OPD from the datasheet, followed by the automatic translation into OPL using OPCAT and subsequent parsing into Cypher code using a custom Python parser to eventually build the CPKG in Neo4j

3.2.2. Modeling and mapping

The OPD was manually generated from the datasheet following consistent modeling rules: product hierarchy was modeled using generalization-specialization links ("is a"); structural elements were represented via aggregation-participation links ("consists of"); characteristics and properties were connected through exhibition relations ("exhibits"); quantitative attributes were modeled together with units according to the generic system model shown in Figure 3, also through exhibition relations ("exhibits"); behavioral dependencies and operational constraints were modeled using instrument links ("requires"); and state transitions or effects induced were modeled using input-output link pairs ("changes").

From the resulting OPD, OPL was automatically generated using OPCAT and subsequently parsed into Cypher code using a custom Python parser to build the CPKG.

3.2.3. Parsing into cypher and CPKG construction

The custom Python parser transforms OPL statements into Cypher code by extracting subject-predicate-object triplets, normalizing values and units, reconstructing hierarchy from exhibition relations, and

https://doi.org/10.1017/pds.2026.10643 Published online by Cambridge University Press

generating graph nodes and relationships while avoiding duplicates via consistent MERGE operations Using the generated Cypher code, the CPKG was built on a local Neo4j instance. The resulting graph contained 100 nodes and 113 relationships.

3.2.4. Tool implementation

A prototype AI-supported RE tool was implemented to answer NL user queries based on the CPKG content. Furthermore, it can verify, rephrase and validate formulated requirements as well as identify and convert unit-value-pairs and comparators, all demonstrated in Section 4.1. The user can choose from three modes before posting a request: 'Graph Query', 'Requirement Verification', and 'Requirement Validation'. Other tool functionalities shown in Figure 2 remain unimplemented in the current prototype. Certain content was extracted from the CPKG and embedded in 384 dimensions using the 'all-MiniLM-L6-v2' model (sentence-transformers, 2025). The embeddings were stored in a FAISS IndexFlatIP structure. At query time, lemmas from the user query were also embedded and matches were retrieved by inner-product similarity with no L2-normalization. For generating answers, a local large language model (LLM) was set up. Best results were achieved with a 4-bit quantized version (Q4_K_M) of the 'Mistral-7B-Instruct-v0.3' model (mistralai, 2025), served by a llama backend using CUDA (temperature = 0, fixed seed), running on an Ubuntu 24 workstation with an Intel Core i9-10900K and an NVIDIA GeForce RTX 5070 Ti.

3.2.5. RE tool algorithms

Independent of the chosen mode, the input is preprocessed first, i.e., the input text is normalized, split into individual sentences, tokenized and lemmatized using the Python package NLTK while considering multi-word phrases from the CPKG. Exact matches between input text and CPKG are already recorded during preprocessing. For the verification and validation mode, preprocessing also includes parameter extraction using the Python package spaCy, as well as the identification and conversion of comparator and unit-value-pair using the packages spaCy, quantulum3 and pint. Unit-value-pairs are converted into SI units, and the comparator terms to the standard comparator syntax '>', '>=', '<', '<=', and '=='.

In the *query mode*, after preprocessing, semantic retrieval is conducted between the CPKG embedding and the lemmas extracted from the user input, to not only rely on exact but also consider close matches. Depending on the number of exact and close matches, different information is extracted from the corresponding nodes and edges by getting the shortest connecting path (native Cypher function) between two or more nodes. From this information, relational sentences and property facts are built in NL, before being sent to the local LLM to compose an answer to the user input.

The *verification* algorithm requires exactly one parameter, one comparator, and one unit-value-pair, otherwise verification fails. With no comparator, implicit equality is used as a fallback. Subsequently, the requirement is rephrased into the EARS-inspired standard pattern '<Parameter> shall <Comparator> <SI-Number> <SI-Unit> (<Number> <Unit>)' and output with an answer sentence, as demonstrated in Table 2.

For the *validation*, first verification is run, and the process is stopped in case the requirement cannot be verified (see above). Subsequently, the above-described query mode is used to extract information required to validate the formulated requirement. The validation checks feasibility against values from the CPKG, the exact logic depends on the extracted comparator and assigns 'pass', 'near', 'fail', or 'ignored' to each of the found values. Then, a textual report is composed and provided to the user.

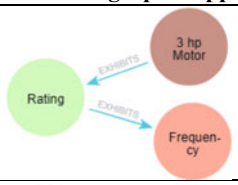
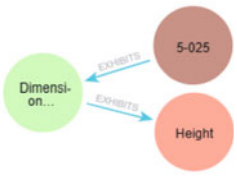
4. Results and discussion

4.1. Case study - query, verification and validation examples

Table 2 shows representative example inputs, retrieved graph snippets, and resulting outputs of the prototypical implementation of the proposed AI-supported RE tool. These examples demonstrate how the characteristic-property relations encoded in the CPKG support automated requirement verification, validation, and refinement, thereby addressing RQ3 at a proof-of-concept level.

https://doi.org/10.1017/pds.2026.10643 Published online by Cambridge University Press

Table 2. Representative example inputs, retrieved snippets from the property graph, and resulting outputs of the prototypical implementation of the proposed RE tool

Mode	Input	Retrieved graph snippet	Output
Answer user query	'What is the frequency of the 3hp motor?'		'The frequency of the 3 HP motor is 60.0 1 / s (60 hertz).'
Answer user query	'What is the range of weights of all the press models?'	 (5x, for each press model: 5-025, 5-050, 5-075, 5-150 and 5-200)	'The range of weights of all the press models is from 453.59 kg (1000 pounds) to 1986.73 kg (4380 pounds).'
Verify requirement	'The motor shall exhibit 3 HP power and operate at 60hz.'		'The formulated requirement cannot be verified. Too many values/units detected. Please rephrase to contain only one value–unit pair.'
Verify requirement	'For the overall height, not more than 90 inches are allowed.'		'The formulated requirement could be verified. This is what I understood: Overall height shall be <= 2.29 m (90 inches).'
Validate requirement	'For the overall height, not more than 90 inches are allowed.'	 (5x, for each press model: 5-025, 5-050, 5-075, 5-150 and 5-200)	'The formulated requirement could be verified. This is what I understood: Overall height shall be <= 2.29 m (90 inches). Found 5 relevant nodes in the KG. 5 of those contain information for the formulated requirement. A deviation of 10.0 % is accepted. Results are listed below: Height is 2.41 m (95 inches).→NEAR Height is 2.16 m (85 inches).→PASS Height is 2.54 m (100 inches).→FAIL Height is 2.21 m (87 inches).→PASS Height is 2.18 m (86 inches).→PASS'

Across the instantiated CPKG, all 100 nodes were retrievable via at least one NL formulation, indicating that the semantic retrieval mechanism systematically covers the modeled graph content rather than operating on isolated examples. The few examples provided in Table 2 showcase direct attribute lookup, aggregated range retrieval, singular requirement detection, comparator normalization, multi-node feasibility evaluation and robustness to variations in syntax.

The *query mode* examples demonstrate the flexibility of the tool as it either extracts a full path or multiple pairs automatically, depending on the query. It also shows direct attribute/property lookup, range aggregation and demonstrates the LLM's capability to formulate an answer in NL.

The *verification* examples demonstrate the guidance towards singular requirements, asking the user to rephrase as too many unit-value-pairs were identified. If verification succeeds, the requirement is automatically rephrased based on the single implemented pattern (see Section 3.2.5). A single pattern is sufficient for the presented prototype but can be extended to cover additional patterns. One last example demonstrates *validation* capability as a combination of verification and query mode. This allows checking the plausibility and feasibility of the formulated requirement. These results underline the added value of the proposed RE tool for requirements engineering.

4.1.1. Preliminary expert feedback

Preliminary expert feedback was collected from four product development PhD researchers. They were provided with the tool prototype, the datasheet containing proxy properties, and a short task sheet asking them to familiarize themselves with the tool for 10-15 minutes and subsequently evaluate predefined statements on a Likert scale based on their subjective judgment of practical utility. The feedback indicated a positive perception of the tool's usefulness for early-phase RE. While the distinction between verification and validation was perceived as partly unclear and would benefit from clearer guidance,

https://doi.org/10.1017/pds.2026.10643 Published online by Cambridge University Press

validation was considered meaningful and useful. Robustness to variations in NL formulations and the level of explanatory feedback were perceived as limited, reflecting the current maturity of the prototype. Overall, participants indicated that they could envision such a tool as part of an engineering workflow in RE, providing initial qualitative support for the practical relevance of the proposed CPM-based framework.

4.1.2. Challenges and limitation

In this case study, only proxy properties extracted from a manufacturer datasheet were used, as no usage-phase measurements were available. This limits empirical validity and prevents claims regarding closing the loop with actual operational performance. However, for RQ1 and RQ2, methodological validity depends on the structured representation of properties and their explicit linkage to design characteristics within the CPKG, rather than on the data source itself. Replacing proxy values with measured or simulated data would require no structural changes to the system model, mapping logic, or tool architecture.

The presented workflow (Figure 4) constitutes a case-specific instantiation of the general framework and may vary for other systems or domains. Transfer to different products would require domain-specific modeling and mapping decisions, while the underlying CPM- and OPM-based framework remains applicable. Further limitations include the manual expert-driven mapping process and the absence of automated physical plausibility checks, both identified as directions for future work.

4.2. Proposed mapping framework and RE tool

The aim of the proposed framework and RE tool is to combine an OPM-based system model capturing its characteristics with measurements from the usage phase of a product (properties) in a CPKG to eventually enable an AI-supported tool to support RE as efficiently and effectively as possible.

OPM plays a big role in this work, it is not only used to visualize the introduced framework and RE tool but also to model the system of interest, serving as a foundation for the created CPKG. The reasons for using OPM are its bimodality and minimal ontology, unlike other available system modeling standards or tools. Kang et al. (2025) further demonstrated its capability to guide answering and make reasoning more transparent. Additionally, OPM exhibits structural similarity to knowledge graphs as object-process links can be cleanly mapped to nodes and edges. This alignment provides an additional rationale for adopting a knowledge graph as the data model in the proposed framework. Other reasons being their ability to handle heterogeneous nodes, typed relationships and support for attributes on nodes and edges.

In the proposed framework, CPM is realized by linking or mapping characteristics and measured properties into a CPKG. This provides the evidence base needed for requirement validation, enabling comparison across system models, product generations, and product variants. It furthermore enables identification of characteristics and properties from requirements, eventually easing subsequent tasks like functional modeling. The unique combination of an OPM-based system model and the mapping process of measured product properties, i.e. the CPM, presents the novelty of this work.

4.2.1. Limitations and potentials

The proposed RE tool relies on measured product properties in the CPKG and therefore assumes the existence of comparable predecessors or variant products, which may constrain the solution space for product line extensions and incremental improvements. Although demonstrated on physical products, the framework is transferable to non-physical systems provided that qualitative and/or quantitative performance indicators can be captured as properties form the usage phase. Expert involvement in characteristic-property mapping constitutes an additional limitation due to potential subjectivity.

The framework is not inherently limited to small knowledge graphs. Structural scalability is supported by OPM-based modular decomposition, which localizes characteristic-property relations. Computational scalability is achieved by embedding-based matching and restricting graph operations to retrieved subgraphs rather than traversing the full graph. Scaling to industrial-scale knowledge graphs primarily affect modeling and maintenance effort and are identified as future work.

4.2.2. Support of requirement quality characteristics

The framework and RE tool support fulfilment of the requirement quality characteristics (see Section 2.1.2). The verification step strongly supports *'Conforming'*, *'Singular'*, and *'Unambiguous'* requirements: The

https://doi.org/10.1017/pds.2026.10643 Published online by Cambridge University Press

tool rephrases free text to *conform* with the defined standard; *Singularity* checks reject compound statements (see Table 2); unit-value normalization converts pairs into SI units, reducing *ambiguity*. The validation step supports the *'Feasible'* characteristic by comparing requirements with properties encoded in the knowledge graph. Together, requirement verification and validation steps support the quality characteristics *'Verifiable'* and *'Validatable'* by enforcing standardized phrasing and linking requirements to meaningful nodes. The framework also supports *'Appropriate'* requirements by anchoring statements to the right level of abstraction via OPM and *'Complete'* requirements by revealing gaps when no property nodes link to modeled characteristics. The quality characteristics *'Correct'* and *'Necessary'* remain subject to user judgement but can be supported by checks for contradictions and duplications.

5. Conclusions and future work

5.1. Conclusion

This work proposed a framework linking design characteristics and product properties through an OPM-based system model to create a characteristics-properties knowledge graph (CPKG). This knowledge graph forms the backbone of an AI-supported RE tool (see Figure 2), supporting data-driven requirement verification and validation. Using a hydraulic press case study, feasibility of the end-to-end workflow as well as structural and algorithmic feasibility of the framework were demonstrated, rather than statistical performance or industrial-scale validation. The prototypical AI-supported RE tool successfully answers user queries and verifies or validates formulated requirements based on the CPKG. This approach supports multiple requirement quality characteristics, as discussed in Section 4.2.2, and thereby enhances traceability and consistency management in RE.

The novelty of this work lies in the OPM-based combination of system and characteristics-properties modeling (CPM) as well as in the prototypically implemented AI-supported RE tool for early-phase RE. Limitations include the reliance on proxy data instead of measured data and on existing product variants. The latter limits the applicability to product line extensions and product improvements rather than disruptive innovation. Future potential lies in automating CPM and transferring the framework to other domains. To conclude, this framework has established the foundation for evidence-based and data-driven requirements validation and refinement, supporting transparent and traceable decision-making in RE. This is another step towards closing the loop between RE and the actual performance of products and systems.

5.2. Future work

Future work will replace proxy observations with measured data from physical products, for example using sensor-equipped production machines and sensor-integrating machine elements (Kirchner et al., 2024), to improve evidence quality. Further directions include extending the implemented requirement pattern set or the addition of an automated requirement assessment as presented by Kumar Sahu et al. (2024), assessing structural and computational scalability on larger knowledge graphs, and supporting automated requirement elicitation using LLM-based or agentic approaches as demonstrated by Korn et al. (2025) and reviewed by Cheligeer et al. (2022). A critical path is the partial automation of the characteristic-property mapping process, for example through rule- and pattern-based methods exploiting recurring CPM and OPM modeling motifs, as well as data-driven or LLM-assisted techniques that suggest mappings in an expert-in-the-loop manner. Finally, we aim to validate the framework's transferability to other domains and assess usability and scalability in industrial settings.

Acknowledgements

This research was funded by the Funding program Distr@l by the Hessian Ministry for Digitization and Innovation. Further, we thank the 'Future Talent Guest Stay' and 'Future Talent short-term Scholarship' programs of TU Darmstadt to allow Dr. Anubhab Majumder to visit and contribute to this research work at TU Darmstadt.

References

Bender, B., & Gericke, K. (2021). *Pahl/Beitz Konstruktionslehre*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-57303-7

Buchert, T., Pförtner, A., Bonvoisin, J., Lindow, K., & Stark, R. (2016). Model-based sustainable product development. *DS* 84: Proceedings of the DESIGN 2016 14th International Design Conference, 145–154. https://www.designsociety.org/publication/38824/model-based_sustainable_product_development

https://doi.org/10.1017/pds.2026.10643 Published online by Cambridge University Press

Cheligeer, C., Huang, J., Wu, G., Bhuiyan, N., Xu, Y., & Zeng, Y. (2022). Machine learning in requirements elicitation: a literature review. *AI EDAM*, 36, e32. https://doi.org/10.1017/S0890060422000166
Dori, D., Reinhartz-Berger, I., & Sturm, A. (2003). OPCAT-A Bimodal Case Tool for Object-Process Based System Development.
Friedenthal, S., Griego, R., & Sampson, M. (June 2007). *INCOSE model based systems engineering (MBSE) initiative*. INCOSE. INCOSE. INCOSE symposium 2007, San Diego.
INCOSE Requirements Working Group (2023). INCOSE Guide to Writing Requirements 1 Jul 2023.
ISO 19450 (2024). ISO 19450:2024: Automation systems and integration — Object-Process Methodology.
ISO 29148 (2018). ISO/IEC/IEEE 29148:2018: Systems and software engineering — Life cycle processes — Requirements engineering.
Kang, X., Shteingardt, V., Wang, Y., & Dori, D. (2025). Neuro-Conceptual Artificial Intelligence: Integrating OPM with Deep Learning to Enhance Question Answering Quality. International Conference on Computational Linguistics (COLING). Advance online publication. https://doi.org/10.48550/arXiv.2502.09658
Kirchner, E., … Stahl, K. (2024). A Review on Sensor-Integrating Machine Elements. *Advanced Sensor Research*, 3(4), Article 2300113, 2300113. https://doi.org/10.1002/adsr.202300113
Kiritsis, D. (2011). Closed-loop PLM for intelligent products in the era of the Internet of things. *Computer-Aided Design*, 43(5), 479–501. https://doi.org/10.1016/j.cad.2010.03.002
Korn, A., Gorsch, S., & Vogelsang, A. (2025, July 3). *LLMREI: Automating Requirements Elicitation Interviews with LLMs*. https://arxiv.org/pdf/2507.02564
Kritzinger, W., Karner, M., Traar, G., Henjes, J., & Sihn, W. (2018). Digital Twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, 51(11), 1016–1022. https://doi.org/10.1016/j.ifacol.2018.08.474
Kumar Sahu, C., Rai, V., & Roshan, V. (2024). Well-Formed Quality of System Requirements for Verifying to ISO 29148-2018: A Natural Language Processing (NLP) Based Framework and Quantitative Metric. ASME 2024 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. https://asmedigitalcollection.asme.org/IDETC-CIE/proceedings/IDETC-CIE2024/88353/V02BT02A013/1208775
Mavin, A., Wilkinson, P., Harwood, A., & Novak, M. (2009). Easy Approach to Requirements Syntax (EARS), 317–322. https://doi.org/10.1109/RE.2009.9
Metro Hydraulic Jack Co. (2025). *H-frame presses: Elec-draulic I Hydraulic Press*. https://www.metrohydraulic.com/pdf/dake-h-frame-presses.pdf
mistralai. (2025, Sep. 26). *Mistral-7B-Instruct-v0.3*. https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3
Mordecai, Y., & Dori, D. (2017). Model-based requirements engineering: Architecting for system requirements with stakeholders in mind, 1–8. https://doi.org/10.1109/syseng.2017.8088273
sentence-transformers. (2025, September 26). all-MiniLM-L6-v2. https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
Weber, C. (Ed.) (2008). *How to derive application-specific design methodologies*. https://www.designsociety.org/download-publication/26774/how_to_derive_application-specific_design_methodologies
Weber, C., Werner, H., & Deubel, T. (2003). A different view on Product Data Management/Product Life-Cycle Management and its future potentials. *Journal of Engineering Design*, 14(4), 447–464. https://doi.org/10.1080/0954482031000160687

https://doi.org/10.1017/pds.2026.10643 Published online by Cambridge University Press