

Hybrid Neuro-Symbolic Decision Support System in Product Planning and Design

Felix Engel¹, Nenad Krzdavac¹, Siamak Ghodsi², Julian Haller³, Antonia Markus³, Sven Munker³, Anas Abdelrazeq³, Robert H. Schmitt³

¹TIB – Leibniz Information Centre for Science and Technology, 30167 Hanover, Germany

²L3S Research Centre, Leibniz University Hannover, 30167 Hannover, Germany

³RWTH Aachen University, WZL-IQS, 52062 Aachen, Germany

*Corresponding author. E-Mail address: Felix.Engel@TIB.eu

Abstract: This article presents research work on a neuro-symbolic decision support system for product planning. It describes the motivation for the work and presents the conceptual framework and the status of implementation. The system to be developed is intended to support engineers in selecting and evaluating assembly and disassembly alternatives during product development. The chosen approach combines symbolic models, represented as Petri nets, ontologies and knowledge graphs, with learning-based, non-algorithmic graph transformation using small language models. Symbolic representations derived from CAD data are formally verified with a theorem prover to ensure feasibility, compliance with constraints and consistency. These verified models provide a reliable basis for decision-making for neural components that generate and evaluate AND/OR Graphs. Instead of programming fixed decision rules, the system learns transformation strategies from verified planning data and human feedback. This enables adaptive decision support for different product concepts. The resulting Decision Support System provides explainable, validated decision alternatives, thus supporting informed decisions in product development, particularly with regard to assembly and disassembly, taking into account circular economy goals

Keywords:

Small language models, neuro-symbolic, decision support system

1 Overview of Content

The process of product development and design is increasingly taking place against a backdrop of global socio-economic challenges. Given the limited availability of natural resources, it is essential to consider the principles of the circular economy (CE) from the earliest stages of design. Engineers and designers are therefore faced with complex decisions in which they must weigh up product functionality, material efficiency, ease of assembly and disassembly, and economic feasibility, while at the same time taking environmental and sustainability goals into account. Traditional decision support approaches, which are often based on static rules, deterministic optimisation or individual cost models, are generally insufficient in this context. They struggle to capture the dynamic interactions between product structures, life cycle strategies, material flows and operational constraints. They are also often unable to take into account multi-criteria trade-offs or the uncertainty inherent in early design phases. Furthermore, conventional tools can only integrate heterogeneous knowledge such as formalised design rules, sustainability guidelines, empirical process data and user feedback into a coherent decision-making framework to a limited extent. These limitations highlight the need for more advanced, adaptive decision support systems that can combine symbolic reasoning, knowledge representation, and learning-based methods to provide flexible, multi-criteria guidance and allow designers to iteratively explore, evaluate, and optimize design alternatives with CE objectives in mind.

To address this situation, interdisciplinary research at the intersection of engineering, artificial intelligence, and sustainability is being conducted. This research explores innovative approaches to the development of adaptive Hybrid Decision Support Systems (HDSS) that combine symbolic and neural methods to support consideration of CE principles in product engineering. In this context, particular emphasis is placed on the integration of explainable, knowledge-based representations with data-driven learning approaches. Combining symbolic and neural methods enables decision support systems to leverage structured engineering and sustainability knowledge while retaining the flexibility and adaptability of modern AI techniques. This hybrid paradigm provides the foundation for coupling human expertise with intelligent systems, as detailed in our architecture section. Therein, symbolic models, implemented as knowledge graphs, represent static domain knowledge and serve as a baseline for generating structured inputs (prompts) for subsequent neural processing steps. The neural models consume these inputs through the symbolic layer while simultaneously interacting with the user and continuously adapting their behavior based on user feedback. The human in the loop approach is chosen based on the assumption that, although we can explicitly model much of the necessary knowledge using symbolic approaches, there is far more implicit knowledge in the minds of engineers, which can be collected through constant interaction with our system. This knowledge is invaluable for future decision-making processes and for preserving knowledge within an organization.

2 Content Structure

This paper reports on research conducted within the Human-Centered & Neuro-Symbolic Decision Support in Circular Product Engineering (HUMAN DECISION) project. It comprises conceptual considerations, the selected research approach, the current state of implementation, and an outline of ongoing and future work. First, some thoughts on consideration of CE criteria are introduced. Next the chosen Neuro-Symbolic (NeSy) approach and research methodology are documented and substantiated by a system architecture that illustrates the interaction of symbolic models, formal verification components, and learning-based methods. Subsequently, the implementation section describes the current realization of the proposed architecture. Particular emphasis is placed on the phases of information extraction from CAD files, their structured representation using graphs and ontologies, and the ongoing development of a neural model. This model is intended to generate valid AND/OR Graphs (AOG) from the verified input in a non-algorithmic, learning-based manner. The paper concludes with a summary of the current research status and an outlook on further development steps within the HUMAN DECISION project.

3 Consideration of Circular Economy

At the current stage of our research, several CE criteria are discussed and identified as having potential for integration into the HDSS. As these criteria have not yet been implemented, this section reflects our present perspective which CE considerations could be incorporated. These include the minimization of disassembly steps required to access wear or core components, as it is assumed that this directly influences the effort associated with repair, refurbishing, and remanufacturing. The installation depth of critical components is regarded as a potentially suitable quantitative metric within the AOG, since it can be derived from structural product data and enables an assessment of component accessibility. Irreversible joining methods, such as adhesive bonding or welding, are considered limiting with respect to disassembly and reuse and are therefore either avoided or explicitly modeled, whereas reversible joining techniques (e.g., screwing or plugging) are viewed as beneficial for supporting repeated assembly and disassembly cycles. Furthermore, it is assumed that considering component accessibility can contribute to more efficient planning of maintenance and end-of-life processes. Product modularity with respect to the production environment is treated as another relevant aspect, as modular structures

are expected to facilitate the targeted replacement and reuse of subassemblies while accounting for existing production constraints. On the process level, the energy and time required for individual assembly and disassembly operations, as well as the number of required tool changes, are regarded as potentially suitable and generally measurable indicators, given their direct relation to resource consumption and environmental impacts across the product lifecycle.

4 Approach and Methodology

The suggested HDSS system architecture integrates NeSy methods with human-centered decision support. Figure 1 illustrates the activity diagram for generating an AOG and corresponding precedence graphs. The workflow (1) begins with the Graph Constructor, which preprocesses CAD data to generate graphs representing directly connected parts (adjacency, liaison, or part contact graphs) and geometric, structural information. Generated graphs in workflow (2) form an initial data structure and are passed to the Converter. Within workflow (3), the Converter transforms graphs into a Petri-Net structure. These Petri Net structures are used as a formal, visual, and mathematical framework to model complex assembly processes. They explicitly represent dependencies between actions, resource constraints, and completion conditions, making it easier to analyze system behavior. Mainly, in assembly sequence planning, Petri Nets are used to model parallel and concurrent activities, thereby enabling efficient planning of assemblies with multiple simultaneous operations. Moreover, Petri Nets enable deadlock detection, ensuring feasible assembly sequences, and hierarchical modeling, which helps manage complex systems by breaking them into manageable subassemblies. Overall, they offer clarity, rigor, and flexibility in assembly sequence planning. To validate the Petri Net-based models of the assembly process with respect to assembly operations and to detect deadlocks, within workflow (4), a formal theorem prover programming language is used (Lean [4]). The Converter transforms the Petri Net-based model into Lean code.

The Prover Agent in workflow (4) receives the Lean code generated by the Converter within workflow (3) and verifies whether the graphs comply with Petri-Net theory. In other words, if the output is valid, it means the theorem has been proved, and if it is invalid, it means the theorem has not been proved. Additionally, the Prover Agent verifies that the generated graphs satisfy predefined constraints, such as assembly rules, gravitational stability requirements, and other domain-specific conditions. The validation results, including any identified constraint violations, are passed within workflow (4) directly to the Graph Constructor in order to eliminate the violations. If the Lean prover proves that all generated graphs are valid, the Ontology Creator uses this information to generate structured ontologies representing these graphs as instances of the Petri-Nets ontology. The Ontology Creator also generates ontologies that describe the geometric and physical properties of the CAD model, and its components, such as wearing parts. Essentially, the Ontology Creator ensures that the data flowing into the ontologies is verified, forming a reliable foundation for symbolic reasoning.

The Reasoner Agent checks the consistency of these ontologies (semantic reasoning output (ontology consistency --> true, false)). If they are consistent, within workflow (5) they are stored in a knowledge graph. Within workflow (6), if the Reasoner Agent detects any inconsistencies, it reports them to the Ontology Creator, while simultaneously logging the inconsistencies and providing explanations for ontology's inconsistency. This is important so that the NeSy AI can retain experience and reuse it for learning, improving future ontology creation. If necessary, the Prover Agent also receives inconsistencies for iterative refinement. In workflow (7), the Query Engine facilitates retrieving specific information from the knowledge graph, such as part properties, constraints, and connectivity relationships.

The Pairwise Feature Calculator activity within workflow (7) processes the retrieved data to compute critical metrics, such as moments of inertia, centers of gravity, gravitational stability, optimal disassembly sequences, and incompatible part pairs. These metrics are then integrated into the knowledge graph as enriched data. The Fine-Tuning SLM Agent subsequently uses this enriched within workflow (8) and (9), validated data to generate training data and fine-tune the Small Language Model (SLM). This ensures that the SLM can effectively reason over the verified structural and constraint knowledge.

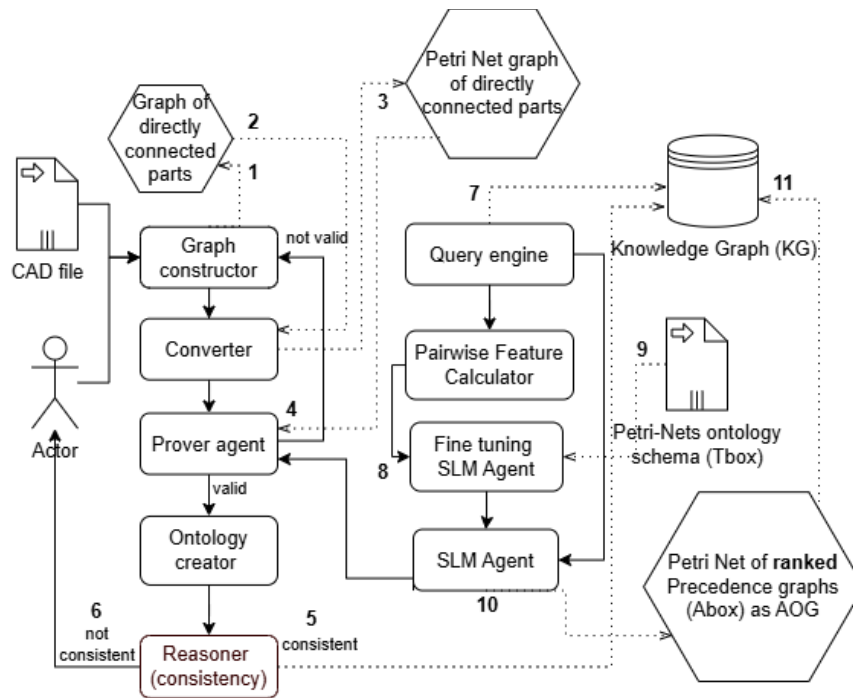


Figure 1. Activity diagram for assembly sequence planning

The SLM Agent integrates information from multiple sources, including a knowledge graph containing Petri-Net ontology schema, computed pairwise features, and constraint representations (e.g., wearing parts, gravitational stability). It also considers human feedback. Using this data, the agent then analyzes the generated graphs, which are serialized as instances of the Petri-Net ontology, and selects appropriate graph transformation actions. Each transformation is applied iteratively, while respecting constraints to optimize for resource-efficient assembly and disassembly. In this context, a transformation represents the task the neural model performs when iteratively generating an AOG, referring to the incremental conversion of one graph into the next: $\text{graph}_1 \rightarrow \text{graph}_2 \rightarrow \text{graph}_3 \rightarrow \dots \rightarrow \text{graph}_n \rightarrow \text{final AOG}$. Within workflow (10), the SLM Agent produces an AOG as its output. This graph is converted into ontologies, which constitute the final AOG that are instances of the Petri-Nets ontology schema. In the workflow (11), generated ontologies are validated for consistency by the Reasoner Agent. If consistent, these ontologies are stored in the knowledge graph. The AOG encodes an optimized sequence of assembly and disassembly operations and integrates ranking information that reflects priorities, constraints, and sustainability objectives. This makes the AOG directly usable for engineering decision support and downstream planning tasks.

This architecture highlights a continuous, structured flow of data and information from raw CAD input to validated, actionable assembly plans. It demonstrates how reasoning, theorem proving, knowledge representation, and adaptive learning converge to support circular product development.

5 Implementation

This section describes the current status of the HDSS implementation. The work can be divided into two main areas. On the one hand, there is the preparation of the input for the SLM agent (based on a CAD file). On the other hand, there is the provision of the SLM-Agent itself. Both aspects are discussed in the following chapters.

5.1 SLM Agent Input

All project materials are hosted in the *Human-Decision* GitLab project [1], which includes *data*, *model*, and *source-code* repositories. The data stored in these repositories are used to train neural networks for generating AOG and related precedence graphs.

The *data* GitLab repository consists of a Petri net ontology and data generated from STEP files. For example, the *step-knowledge-graph* GitLab repository contains the *der HundD30* and *AIMFREE_Batterypack* knowledge graphs generated from STP files, along with a set of SPARQL queries that can be used to query these knowledge graphs. The *Assembly Sequence Planning Knowledge Graph* GitLab repository, within the *data* repository, contains the Petri net ontology used in *Human-Decision* project as a schema for describing part contact graph, which represents directly connected parts (see Steps 1–3 in Figure 1), as well as AOG (see Step 10 in Figure 1).

The *source-code* GitLab repository contains a Python project that analyzes STEP (Standard for the Exchange of Product Data) files to create contact matrices indicating which parts touch or are connected to each other. It visualizes the results as interactive graphs using NetworkX, generates instances of the Petri net ontology representing these contact matrices (Step 1 in Figure 1), and exports the results in multiple formats (PNG, TTL, and Lean). In addition, the Python project generates Lean code that implements the part contact graph (Step 4 in Figure 1). The *source-code* repository also includes a Lean project that validates the generated Petri net of the part contact graph using the Lean proof assistant. All data generated in the aforementioned repositories are used for neural network fine-tuning. The *Human-Decision* GitLab project also includes a model GitLab repository that provides an implemented example demonstrating how an assembly process can be modeled using existing ontologies. Figure 2 presents a part contact graph generated from a Hund STEP file using software available in a GitLab repository [2]. This software automatically converts this graph into an ontology that is an instance of the Petri-Net ontology. A visual snapshot of the resulting ontology is shown in Figure 3.

As illustrated in Figure 3, part B shows a portion of the Hund CAD consisting of four distinct components, while part A presents a visual representation of the corresponding Petri-Net ontology that governs the part contact graph for this portion of the Hund CAD model. Each part of the CAD model shown in part B of Figure 3 is represented as a black node in part A. These black nodes correspond to instances of the Place class in the generated Petri-Net ontology. The double lines shown in part A represent transitions, which are instances of the Transition class, while the arrows denote instances of the Arc class in the Petri-Net ontology.

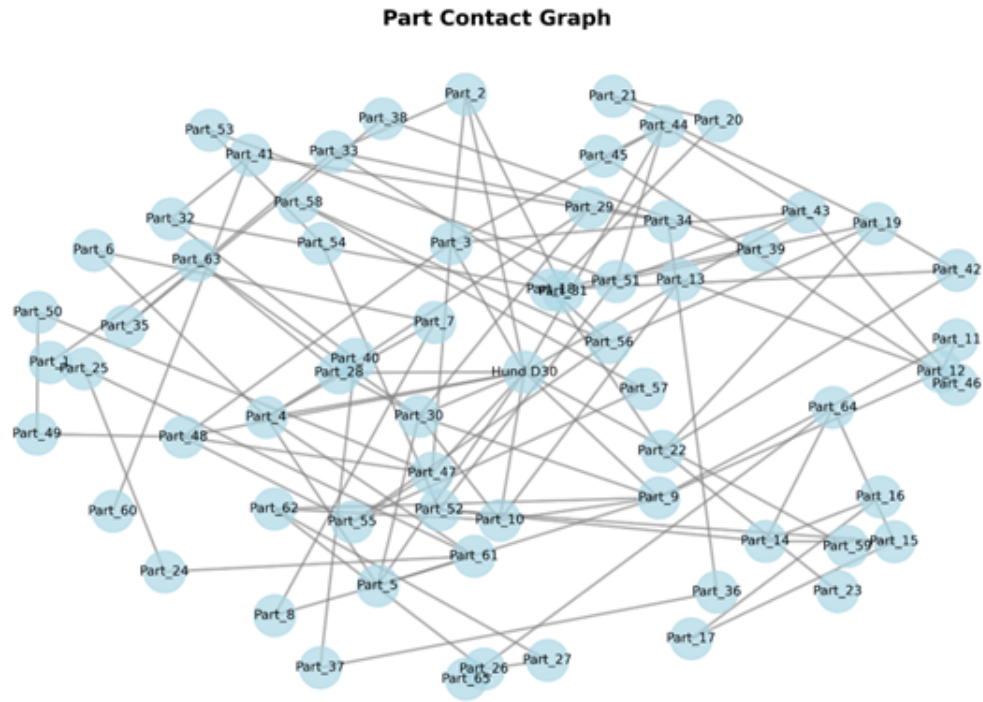


Figure 2. Part contact graph generated automatically from STEP file

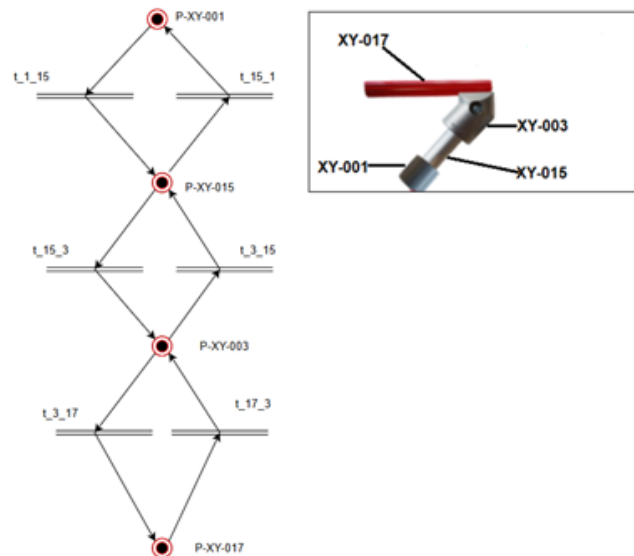


Figure 3. Part A visualizes a Petri-Net ontology of the part-contact graph for the Hund CAD model shown in Part B

The Petri-Net ontology derived from the Hund CAD model is available as a TTL file in the results folder of the GitLab repository [2] and has successfully passed a consistency check using the HermiT reasoner [3]. In addition to the TTL file, the software [2] generates Lean code that represents the part contact graph shown in Figure 2. This Lean code is also available in the same results folder of GitLab repository [4]. The correctness of the Petri-Net derived from the Hund CAD model is validated using the Lean proof assistant [17].

5.2 SLM Agent

Our approach is to learn AOG for assembly sequence planning by combining knowledge engineering, and neural models (see section 3, Fine Tuning SLM Agent). Multiple heterogeneous inputs are used,

including part contact graphs extracted from CAD models, product geometry, and topological constraints represented with the OntoSTEP ontology, and a Petri-net ontology schema that defines the formal structure of valid AOG; CE data are intended as an additional numerical input in a later stage of the project. All ontology-based inputs are transformed into representations suitable for machine learning. Recent advances in LLMs for CAD demonstrate their potential for automating design workflows through code generation and iterative self-improvement [5, 6]. We adapt these methodologies to assembly sequence planning, where the neural model learns to generate valid graph transformations grounded in ontological constraints. However, even this adaptation process necessitates a sufficient amount of training and test data. We propose the following combined training pipeline to overcome challenges of data scarcity, model limitation, data augmentation, and evaluation.

5.2.1 Training Pipeline

Our approach follows a four-phase pipeline designed to address the scarcity of annotated assembly planning data:

Phase 1: Gold-Standard Collection. We curate a reference dataset of validated assembly sequences from three sources: (i) expert-annotated sequences for existing assemblies including toy examples of knife, and dog assemblies, (ii) sequences generated and verified by the tools like ASAP or NeSy models like LLM4CAD [8, 9], and (iii) sequences validated through the Lean theorem prover pipeline. Each sequence should ideally include metadata on optimality status, constraint satisfaction, and circular economy annotations.

Phase 2: Zero-Shot Benchmarking. Using the gold-standard set, we evaluate existing LLMs on AOG generation without fine-tuning. Models receive a structured prompt containing the Petri-Net state (serialized as JSON-LD), enabled transitions, and task instructions. We benchmark models such as GPT-4, Claude-3.5, Qwen2.5-Coder, and Llama-3 to identify which architectures best handle structured graph reasoning. This phase establishes baselines and informs model selection.

Phase 3: Knowledge Distillation. The best-performing model undergoes knowledge transfer from ASAP's algorithmic planning. Training pairs take the form where is the serialized Petri-Net state, is the set of enabled transitions, and is ASAP's selection. This supervised fine-tuning teaches the model to replicate expert-level planning decisions.

Phase 4: Synthetic Data Augmentation. To address the challenge of missing data, we initially pursue an approach in which AOGs are generated for a selection of CAD files using existing software. Following BlenderLLM's self-improvement methodology [5], our training set can be expanded based on synthetic data augmentation as follows: (1) The model generates candidate sequences for held-out assemblies, (2) Sequences validated via Petri-Net simulation and Lean verification, (3) Valid sequences filtered by quality (length, stability), (4) High-quality generations augment training data. This cycle continues until validation performance plateaus.

5.2.2 Evaluation Framework

The evaluation focuses on criteria relevant to manufacturing process planning. First, the correctness of generated assembly sequences is verified by checking whether all operations are feasible and whether the final state represents a completed assembly. Second, process efficiency is assessed by comparing the number of operations in the generated sequences with known reference or optimal solutions. Additional measures evaluate the reliability of the method, such as the probability of finding a valid assembly plan within a limited number of planning attempts. The proposed approach is compared against baseline methods including heuristic planners, classical search-based planning techniques, and data-driven methods without formal verification. We adopt metrics from recent CAD-LLM benchmarks [5, 9, 10], adapted for assembly planning as follows:

Table 1: Evaluation metrics for assembly sequence generation

Metric	Definition	Verification
Validity	Sequence executes without error	PN Simulator
Completeness	Final marking is complete	
Optimality Ratio	$ S / S^* $ vs. reference	Gold-standard
Success@1	Valid on first attempt	Binary
CAD F1	balance between Entity Accuracy and Sketch Accuracy	[0,1]

This evaluation table shall be enriched with production and engineering KPIs in the future. After defining appropriate KPIs, we now move to baseline comparisons. Our intended baseline models set include: (i) random sampling from enabled transitions, (ii) greedy selection by part connectivity, (iii) zero-shot LLM prompting, and (iv) BFS-optimal search for small assemblies. All experiments will use equal random seeds like $N=5$ with paired t-tests for significance ($p<0.05$).

5.2.3 Expected Results and Failure Analysis Plan

The expected result is a planning system that consistently produces feasible assembly sequences while achieving near-optimal process lengths. By learning from structural similarities across different assemblies, the approach is expected to generalize to new products and scale to assemblies with a large number of components. The explicit integration of symbolic verification ensures that generated process plans comply with all assembly constraints, which is a key requirement for industrial applicability. To support systematic improvement and industrial deployment, failure cases are explicitly analyzed. Failures are categorized into infeasible operation selections, deadlock situations, valid but inefficient assembly sequences, and planning timeouts for complex assemblies. For each failure case, the corresponding assembly state and Petri Net configuration are examined to identify structural or procedural causes. This analysis provides insights into typical planning bottlenecks and guides further refinement of both the data and the planning model.

6 Outlook

Based on the current state of implementation, we can demonstrate that the information initially specified in the system architecture can be successfully extracted from CAD data and provided as structured input to the SLM agent. As the models responsible for computing AOGs from this input are still under development, it is not yet possible to draw definitive conclusions regarding the full functionality of the envisioned HDSS. In addition to addressing performance requirements related to input preprocessing and model execution, a key area for further development is the integration of additional contextual information (further CE relevant information) that can be considered during AOG computation. Potential extensions include the analysis of joining technologies and material choices, as well as the incorporation of production-related constraints, such as the manufacturing processes and infrastructural conditions of the target production facility (see section 3).

However, AOGs are not definitive production plans, but rather serve as validated templates for production systems. They encode assembly and disassembly sequences, but their practical implementation depends largely on variable characteristics of the production environment. These include, in particular, the production layout, the capacities of plants and workstations, the available

equipment, the degree of automation, the flexibility of processes and sequences, and the necessary retooling effort. Digital twins, for example in the form of scenario analysis systems [16], enable the systematic combination of a multitude of relevant configuration parameters and the identification of optimal characteristics. Based on the AOGs, corresponding production parameters can be configured. This allows alternative production scenarios to be generated and evaluated, for example with regard to optimal resource utilisation. However, the configuration space is characterised by inherent conflicts of interest: maximising throughput, for example, can conflict with changeover capability or accessibility for disassembly. A higher degree of automation increases process stability, but is often accompanied by rising changeover costs. In addition, local optimisations in line-free production systems can lead to increased global resource consumption.

Against this, we plan to expand our decision support system to include a multi-objective optimisation approach that can be used to evaluate these parameters. By incorporating user feedback, we ensure that the scenarios generated remain practical and feasible. The AOG herein acts as a central link between circular product development and adaptive production planning, ensuring the consistency of early design decisions throughout the entire product life cycle.

7 Conclusion

The NeSy approach, enhanced by the integration of theorem provers and inference engines as external reasoning components, enables a wide range of industrial applications. In manufacturing, and in particular in assembly and disassembly planning, the generation of AOGs represents a central task. Classical algorithmic approaches construct AOG that guarantee logical correctness for a given input configuration [11]. However, these methods are typically limited to rule-based transformations and lack the ability to learn recurring transformation patterns across different products, system configurations, or lifecycle scenarios, which restricts their scalability and adaptability [12].

The proposed NeSy framework addresses these limitations by tightly coupling symbolic reasoning and formal verification with learning-based graph transformation mechanisms. Verified structural representations, such as Petri Nets and precedence graphs, are embedded into a knowledge graph and enriched with semantic information, physical properties, and operational constraints. A set of interconnected, agentic SLMs [13] operates on these representations and is supported by external reasoning tools, including a theorem prover such as Lean [14] and description logic reasoners. This combination ensures that all generated graph transformations remain consistent with formally specified constraints, such as assembly feasibility, gravitational stability, and circularity-related requirements. Graph transformations are performed within a reinforcement learning framework [15], allowing the SLM agents to explore the space of feasible transformations and to learn transformation strategies across different product and production contexts. Rather than producing a single static solution, the system generates ranked and validated AOGs that reflect multiple, potentially conflicting objectives, such as efficiency, reconfigurability, and disassembly accessibility. Human feedback is integrated into this loop to guide exploration, resolve ambiguities, and ensure alignment with engineering intent.

While manufacturing and assembly planning represent a primary application domain, the underlying principles of the framework are not domain-specific. The combination of symbolic constraint enforcement, knowledge graph-based representations, and learning-driven graph transformations is applicable to other complex, graph-based decision problems, such as treatment planning in medicine or curriculum and competency modeling in education. Across domains, the NeSy approach enables flexible, adaptive, and formally grounded handling of tasks that are difficult to address with purely algorithmic or purely data-driven methods.

Acknowledgement: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – HUMAN-DECISION (543081196), part of SPP 2443: Hybride Entscheidungsunterstützung in der Produktentstehung (521148212)

References

- [1] Human-Decision GitLab repository: <https://gitlab.com/TIBHannover/human-decision>, Accessed January 17th, 2026.
- [2] Step to contact matrix extraction, Python project: <https://gitlab.com/TIBHannover/human-decision/source-code/STEP-contact-matrix-extraction/>, Accessed January 20th, 2026
- [3] Glimm, B., Horrocks, I., Motik, B., Stoilos, G. and Wang, Z., 2014. HermiT: an OWL 2 reasoner. *Journal of automated reasoning*, 53(3), pp.245-269.
- [4] Implementation Petri-Nets in Lean proof assistant. Lean project: <https://gitlab.com/TIBHannover/human-decision/source-code/theorem-proving>, Accessed January
- [5] Yuhao Du, Shunian Chen, Wenbo Zan, Peizhao Li, Mingxuan Wang, Dingjie Song, Bo Li, Yan Hu, Benyou Wang (Arxiv 2024). “BlenderLLM: Training Large Language Models for Computer-Aided Design with Self-improvement”, arXiv:2412.14203
- [6] Licheng Zhang, Bach Le, Naveed Akhtar, Siew-Kei Lam, Tuan Ngo (Arxiv 2025). “Large Language Models for Computer-Aided Design: A Survey.”, arXiv:2505.08137.
- [8] Yunsheng Tian, Karl D.D. Willis, Bassel Al Omari, Jieliang Luo, Pingchuan Ma, Yichen Li, Farhad Javid, Edward Gu, Joshua Jacob, Shinjiro Sueda, Hui Li, Sachin Chitta, Wojciech Matusik (MIT CSAIL) (ICRA 2024). “ASAP: Automated Sequence Planning for Complex Robotic Assembly with Physical Feasibility.”, 2024
- [9] Xingang Li, Yuewan Sun, Zhenghui Sha, *Journal of Computing and Information Science in Engineering* (JCSIE 2025). “LLM4CAD: Multimodal Large Language Models for Three-Dimensional Computer-Aided Design Generation.”, Feb 2025
- [10] Sifan Wu · Amir Khasahmadi · Mor Katz · Pradeep Kumar Jayaraman · Yewen Pu · Karl Willis · Bang Liu, NeurIPS 2023 Workshop on Machine Learning for Creativity and Design. “CAD-LLM: Large Language Model for CAD Generation.”, 2023.
- [11] De Mello LH, Sanderson AC. A correct and complete algorithm for the generation of mechanical assembly sequences. In: 1989 IEEE International Conference on Robotics and Automation 1989 Jan 1 (pp. 56-57). IEEE Computer Society. URL: <https://www.computer.org/csdl/proceedings-article/robot/1989/00099967/>
- [12] Moore NS, Cyr EC, Ohm P, Siefert CM, Tuminaro RS. Graph neural networks and applied linear algebra. *SIAM review*. 2025 Mar 31;67(1):141-75. URL: <https://epubs.siam.org/doi/10.1137/23M1609786>
- [13] Belcak P, Heinrich G, Diao S, Fu Y, Dong X, Muralidharan S, Lin YC, Molchanov P. Small language models are the future of agentic AI. arXiv preprint arXiv:2506.02153. 2025. URL: <https://arxiv.org/abs/2506.02153>
- [14] de Moura L, Kong S, Avigad J, van Doorn F, von Raumer J. The Lean theorem prover (system description). In: *Automated Deduction – CADE-25. Lecture Notes in Computer Science*, vol. 9195. Springer, 2015. pp. 378–388. DOI: https://doi.org/10.1007/978-3-319-21401-6_26.
- [15] He G, Parker S, Yoneki E. X-RLflow: Graph reinforcement learning for neural network subgraphs transformation. arXiv preprint arXiv:2305.12345. 2023. URL: <https://arxiv.org/abs/2305.12345>
- [16] Grunert, D., Göppert, A., Hort, S., Rachner, J., Grahn, L., & Schmitt, R. H. (2021). Modelling for Decision-making in Dynamical Line-less Assembly Systems. *Journal of Production Systems and Logistics* 1 (2021), 1. doi: <https://doi.org/10.15488/11327>
- [17] Implementation Petri-Net in Lean proof assistant: <https://gitlab.com/TIBHannover/human-decision/source-code/theorem-proving>, Accessed January 21st, 2026