Journal Title
XX(X):2–41

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Integrating Neurosymbolic Systems in Advanced Product Design: A Comprehensive Review

Siamak Ghodsi[1,2], Gollam Rabby[1,2], Felix Engel[2], Sven Münker[3] and Sören Auer[1,2]

Abstract

Neurosymbolic artificial intelligence combines neural network adaptability with symbolic reasoning, offering a framework for addressing multi-objective optimization, constraint satisfaction, and interpretability challenges in advanced product design. This survey reviews 70 studies spanning generative CAD, topology optimization, manufacturing and assembly planning, quality assurance, and sustainability-driven design, and classifies 38 representative systems by integration type, symbolic substrate, neural function, and evaluation domain. The analysis reveals a fragmented landscape dominated by loosely coupled architectures, with tighter integrations still concentrated largely in general benchmarks rather than engineering workflows. Evaluation practice is similarly fragmented: reported protocols are heterogeneous, and we did not identify a study that explicitly isolates the contribution of its symbolic component. We therefore propose an operational evaluation protocol separating task performance, symbolic contribution, and system credibility, and illustrate its relevance through a deployed industrial case study. By linking these techniques to the operational demands of product design, this work provides a roadmap for researchers and practitioners developing intelligent design automation systems.

Keywords

Neurosymbolic AI, Product Design, Manufacturing, CAD, Knowledge Graphs, Constraint Satisfaction, Survey

Prepared using sagej.cls *[Version: 2017/01/17 v1.20]*

1 Introduction

Product design has evolved from manual drafting and rule-based Computer-Aided Design (CAD) to data-driven generative methods powered by neural networks. While deep learning has revolutionized design space exploration through automatic feature extraction, generative synthesis, and performance prediction, purely neural approaches lack interpretability and cannot guarantee constraint satisfaction or formal verification of physical and manufacturing requirements such as structural integrity, thermal limits, or manufacturability. This gap has catalyzed interest in neurosymbolic (NeSy) systems that combine neural generalization with verifiable symbolic reasoning, representing the third wave of intelligent design automation d'Avila Garcez and Lamb (2023).

Modern product design across aerospace, electronics, automotive, and industrial domains requires simultaneously optimizing conflicting objectives, minimizing weight while maximizing strength, reducing cost while improving performance, subject to hard constraints: safety standards, manufacturing tolerances, material availability, and regulatory compliance Roijers et al. (2013). Neural networks excel at learning design-performance mappings but cannot inherently enforce constraints or explain decisions. Symbolic systems rigorously verify requirements but lack the capacity to discover novel patterns in high-dimensional data. Neurosymbolic Artificial Intelligence integrates neural pattern recognition with symbolic constraint reasoning, enabling design systems that are both data-informed and provably correct.

Neurosymbolic models offer the potential to learn from large-scale design databases (e.g., CAD repositories, simulation results, and manufacturing records) while enforcing domain-specific rules through symbolic layers. This approach supports the entire design lifecycle, from conceptual generation and topology optimization to engineering analysis and manufacturing planning. It balances multiple objectives on the Pareto front while satisfying hard constraints. Encoding geometric constraints and process capabilities symbolically alongside adaptive neural policies allows refinement to continue under formal guarantees. Yet developing robust architectures for high-dimensional design spaces, real-time constraint verification, multi-stakeholder preferences, and seamless CAD-to-manufacturing workflows remain active research challenges.

While neurosymbolic integration and multi-objective reinforcement learning (MORL) have each been studied extensively in isolation Besold et al. (2021); Hayes et al.

[1]L3S Research Center, Leibniz University of Hannover, Germany
[2]TIB – Leibniz Information Centre for Science and Technology, Hannover, Germany
[3]Laboratory for Machine Tools and Production Engineering (WZL), RWTH Aachen University, Germany

Corresponding author:
Siamak Ghodsi, L3S Research Center, Leibniz University of Hannover, Appelstraße 9a, 30167 Hannover, Germany.
Email: ghodsi@l3s.de

(2022), their integration for product design remains fragmented. Existing surveys primarily address general NeSy architectures Yu et al. (2023); d'Avila Garcez and Lamb (2023); Colelough and Regli (2024), abstract multi-objective reinforcement learning formulations Roijers et al. (2013), or domain-agnostic NeSy-RL frameworks Acharya et al. (2024), without systematically examining how symbolic constraints, multi-objective trade-offs, and verification requirements interact within end-to-end product design workflows. In particular, prior work concentrates either on single-objective or safety-constrained control, or on multi-objective methods without explicit symbolic reasoning and design-rule verification. Both are critical for engineering applications.

This review addresses this gap by synthesizing neurosymbolic integration patterns specifically in the context of product design, from generative CAD and structural optimization to manufacturing and assembly, with an emphasis on verifiability, interpretability, and Pareto-efficient decision-making.

Abbreviations. Table 1 summarizes abbreviations and notation used throughout this survey. All acronyms are defined at first use in the main text.

2 Survey Methodology

We conducted a structured literature review (SLR) to identify and synthesize NeSy methods for product design, spanning generative CAD, manufacturing and process optimization, assembly/disassembly planning, and sustainability-driven design. The protocol follows a PRISMA-inspired workflow adapted to engineering and AI literature; the study selection process is summarized in Figure 1.

Living resources (ORKG and repository). To improve transparency and facilitate community reuse, we maintain a queryable, structured comparison of the reviewed methods in the Open Research Knowledge Graph *(ORKG)*[*]. An accompanying *GitHub*[†] repository provides a living bibliography (Awesome-style list), metadata artifacts, and supporting material. These resources may extend beyond the static snapshot discussed in the manuscript, while the SLR corpus analyzed in this paper consists of **70** included papers.

Research scope and questions. The review targets methods that integrate *symbolic substrates* (e.g., logic, rules, constraints, ontologies, knowledge graphs, planners, solvers) with *neural components* (e.g., perception, generation, optimization, surrogate modeling) in product design pipelines. Concretely, we ask: (i) which NeSy integration patterns are used across design subdomains, (ii) what symbolic substrates and neural functions are combined, (iii) which objectives and constraints are supported (e.g., manufacturability, safety, sustainability, cost/time), and (iv) how systems are evaluated (datasets, simulators, benchmarks, case studies).

[*] orkg.org/comparisons/R1569149
[†] github.com/SiamakGhodsi/NS-Product-Design-Survey

Table 1. Abbreviations and notation used in this survey.

Abbrev.	Meaning	Abbrev.	Meaning
General terms			
AI	Artificial Intelligence	CAD	Computer-Aided Design
NeSy	Neurosymbolic	KG	Knowledge Graph
CSP	Constraint Satisfaction Problem	PDE	Partial Differential Equation
PLM	Product Lifecycle Management	MDP	Markov Decision Process
Machine learning and neural networks			
RL	Reinforcement Learning	MORL	Multi-Objective Reinforcement Learning
NeSy-RL	Neurosymbolic Reinforcement Learning	RAG	Retrieval Augmented Generation
CNN	Convolutional Neural Network	GNN	Graph Neural Network
VAE	Variational Autoencoder	GAN	Generative Adversarial Network
LLM	Large Language Model	VLM	Vision-Language Model
PINN	Physics-Informed Neural Network	TINN	Taxonomy-Informed Neural Network
PEFT	Parameter-Efficient Fine-Tuning	LoRA	Low-Rank Adaptation
Symbolic AI and formal methods			
SAT	Boolean Satisfiability	SMT	Satisfiability Modulo Theories
QP	Quadratic Programming	DSL	Domain-Specific Language
DL	Description Logic	OWL	Web Ontology Language
SWRL	Semantic Web Rule Language		
Evaluation and methodology			
SLR	Systematic Literature Review	RMSE	Root Mean Square Error
ORKG	Open Research Knowledge Graph	MAE	Mean Absolute Error
HV	Hypervolume indicator	CSR	Constraint-Satisfaction Rate
PRISMA	Preferred Reporting Items for Systematic Reviews and Meta-Analyses		
Table 3 notation — Domains (Dom)			
CAD	Computer-Aided Design	Mfg	Manufacturing / production
Asm	Assembly planning	Mat	Materials
Topo	Topology optimization	Dsgn	Design (general)
G	General benchmark (none specific product design)	QA	Quality Assurance
Table 3 notation — Neural functions (NF)			
P	Perception / feature extraction	Gen	Generation
Opt	Decision / optimization / control	S	Surrogate / prediction
Table 3 notation — Symbolic substrates (SS)			
O	Ontology	K	Knowledge graph
L	Logic / formal specification	C	Constraints / physics
R	Rules		
Table 3 notation — Other			
(pd)	Paper-defined dataset/benchmark	NR	Not reported as directly comparable scalar

Information sources and search strategy. Records were identified in two stages. First, a systematic automated harvest was performed over the DBLP and Semantic Scholar bibliographic indexes, which aggregate metadata from the major AI and engineering publishers (IEEE, ACM, Springer, Elsevier); this harvest yielded the $n = 1{,}095$ records reported in Figure 1. Searches combined keywords covering (a) the NeSy dimension and (b) the product design dimension. Example queries included: *(neuro-symbolic OR neurosymbolic OR neural-symbolic OR "knowledge graph" OR ontology*

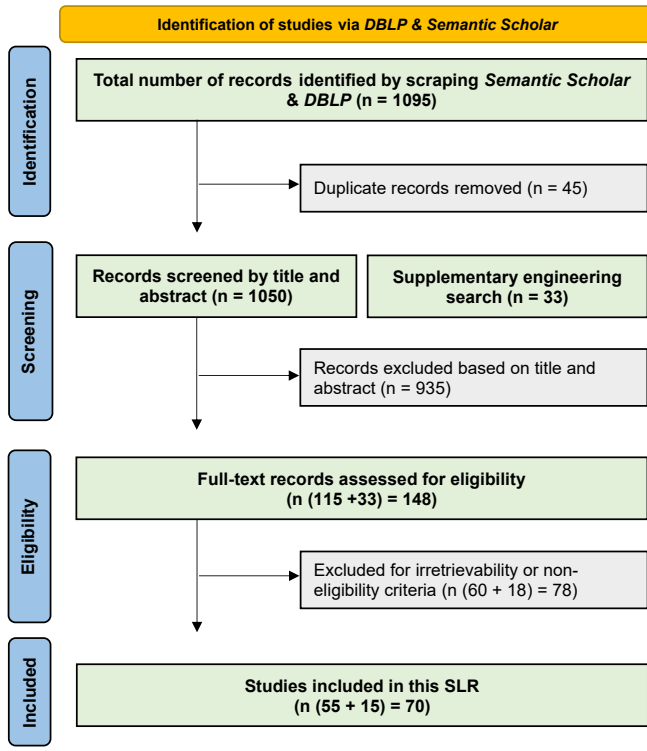

Figure 1. PRISMA-style flow diagram summarizing identification, screening, eligibility assessment, and inclusion for the SLR. The left stream under screening reports identification via bibliographic databases; the right stream reports the supplementary engineering-database search conducted during revision (final SLR corpus: 70 papers).

OR logic OR constraint) AND *(product design OR CAD OR "computer-aided design" OR manufacturing OR "process planning" OR assembly OR disassembly OR topology optimization)*. Second, targeted supplementary searches were run in Google Scholar and arXiv, and backward and forward snowballing was performed on highly relevant survey and cornerstone papers. Publisher portals (IEEE Xplore, ACM Digital Library, SpringerLink, Elsevier/ScienceDirect) were additionally used for full-text retrieval and for reconciling preprint and published versions of the same work.

Eligibility criteria. We included peer-reviewed conference/journal papers and well-established preprints that: (1) address a product-design-related task (e.g., CAD synthesis, manufacturing planning/optimization, assembly planning, topology optimization, materials selection), and (2) contain an explicit integration of symbolic structure with neural learning/inference (or a clearly defined hybrid architecture). We excluded papers that: (i) are purely symbolic expert systems without learning, (ii) are purely neural models without a symbolic component, (iii) focus on generic NeSy

reasoning without a meaningful connection to product design, or (iv) provide insufficient technical detail to assess the integration mechanism. Only English-language publications were considered.

Screening and selection process. Records were first de-duplicated, then screened by title and abstract for topical fit. Remaining papers underwent full-text assessment against the inclusion criteria. When multiple versions existed (e.g., arXiv and later published), we retained the most complete peer-reviewed version. The resulting selection flow is reported in Figure 1.

Supplementary engineering-database search. Because identification relied primarily on computer-science-oriented aggregators, coverage of engineering-specific indexes was potentially incomplete. We therefore conducted a supplementary search of Engineering Village/Compendex, the ASME Digital Collection and Scopus, applying the same eligibility criteria and restricted to journal and conference articles. Knovel was not searched: it indexes handbooks, materials-property data and reference works rather than primary research literature, and is therefore not an appropriate source for methodological studies. The search retrieved 33 studies addressing product-design tasks. Fifteen satisfied the eligibility criteria and were incorporated into the review corpus and the unified classification in Table 3. The remaining 18 were excluded under criterion (ii): they comprise purely neural design-automation methods. For example, design-automation methods such as machining-feature recognition, B-rep representation learning, programmatic CAD reconstruction, and conditional generative design, together with ontology-, rule- and case-based systems that carry no learned component, and machine-learning approaches to life cycle assessment in which the inventory model remains external to the learner. These adjacent studies are not counted in the corpus, but selected examples are cited in the discussion where they establish context, and all 33 records are retained in the ORKG comparison, labelled as core or adjacent, so that the boundary applied here is inspectable. Both streams appear in Figure 1.

Data extraction and coding. For each included paper we extracted: design domain, symbolic substrate(s), neural function(s), integration pattern (following Kautz's taxonomy), objectives/constraints, required inputs (data modalities, CAD representations, simulators), evaluation setting (benchmarks, case studies), and reported limitations. Extraction was performed using a standardized template and coding schema. The corresponding extraction template, coding fields, and structured records (including ORKG-aligned metadata) are provided in the companion repository.

Synthesis and consistency checks. We synthesized findings along three axes: (1) *integration pattern* (Types I–VI), (2) *symbolic substrates* and *neural functions*, and (3) *product design subdomains*. Ambiguous cases were resolved by re-reading the method and evaluation sections and cross-checking terminology across related work.

3 Preliminaries

This section establishes foundational paradigms underpinning NeSy systems in product design. We examine symbolic reasoning approaches that provide constraint

verification but lack learning capabilities; neural learning methods that enable data-driven optimization but cannot guarantee formal correctness; and then NeSy integration paradigms that combine both to address their complementary limitations.

3.1 Symbolic Reasoning for Product Design

Symbolic AI, dominant in engineering automation from the 1970s through the 1990s, provides explicit knowledge representation, logical inference, and formal verification, all of which are essential for constraint-driven design. These methods enable transparent reasoning and correctness guarantees but require extensive manual knowledge engineering and cannot adapt from experience Nardi and Brachman (2003).

Ontologies and Knowledge Representation. Description logics provide formal foundations for ontology-based knowledge bases and taxonomies in product design, enabling precise specification of material properties, geometric relations, manufacturing capabilities, and compatibility constraints Nardi and Brachman (2003). Ontology-based systems encode hierarchical classifications in which metallic alloys, for example, inherit properties from parent metal classes while defining specific characteristics, thereby facilitating automated material selection and process compatibility checking Golpayegani et al. (2024); Kim et al. (2006). However, constructing comprehensive ontologies requires substantial domain expertise, and maintaining them as materials and standards evolve demands continuous manual curation.

Knowledge Graphs (KG) extend ontological frameworks by capturing historical context and design rationale Hao et al. (2021), linking design parameters to performance outcomes, component relationships, and engineering justifications. This enables traceability for regulatory compliance and design reuse through the identification of similar solutions. However, their rigid structure challenges the incorporation of uncertain or probabilistic relationships.

Constraint Satisfaction and Formal Verification. Constraint satisfaction problems (CSPs) ensure designs meet hard requirements before manufacturing. CSP solvers verify that geometric tolerances, material stress limits, and assembly sequences are satisfied simultaneously Rosell (2004). These formal methods provide mathematical correctness guarantees crucial for safety-critical applications, but suffer from scalability limitations as design spaces grow to thousands of parameters and cannot handle perceptual tasks such as CAD model parsing or defect detection.

Limitations. Purely symbolic approaches (including rule-based expert systems and planning-based configurators) rely on sufficiently complete and up-to-date domain knowledge specified upfront. This assumption is often unrealistic in product design, where requirements evolve, suppliers change, and new materials/processes emerge. Moreover, symbolic systems do not naturally leverage large-scale historical data or simulation logs for generalization; instead, new scenarios typically require manual knowledge engineering and rule maintenance. Finally, under partial observability or noisy sensing, purely symbolic pipelines can be brittle, often failing to produce feasible

Table 2. Comparative capabilities of symbolic, neural, and neurosymbolic paradigms for product design. Ratings are qualitative: *Strong* indicates the capability is typically native or enforceable by design; *Medium* indicates partial support or strong dependence on task setup; *Weak* indicates the capability is not reliably achieved without substantial additional machinery; *None* indicates the paradigm does not provide the capability in its standard form. OOD stands for out-of-distribution

Capability	Symbolic	Neural	Neurosymbolic
Hard constraint satisfaction	Strong	Weak	Strong
Learning from data	None	Strong	Strong
Perception from unstructured inputs	Weak	Strong	Strong
Interpretability / rationale	Strong	Weak	Strong
Robustness to noise / uncertainty	Weak	Medium	Strong
OOD generalization / novelty	Medium	Weak/Medium	Medium/Strong
Verification / certification readiness	Strong	Weak	Medium/Strong
Data efficiency	Medium	Weak/Medium	Medium/Strong
Computational scalability	Medium	Strong	Medium
Knowledge maintenance cost	High	Low	Medium

solutions rather than degrading gracefully. To position these limitations in the broader landscape, Table 2 contrasts symbolic, purely neural, and neurosymbolic paradigms across recurring design capabilities and trade-offs.

3.2 Neural Reasoning for Product Design

Neural networks have transformed product design by learning complex patterns directly from data. They automatically extract features from design repositories, simulation databases, and sensor streams, enabling advances in generative design, performance prediction, and multi-objective optimization Kusiak (2020). Yet, as black-box function approximators, they provide neither formal guarantees nor the traceable rationale that engineering decisions typically require.

Perception and Feature Learning. Convolutional neural networks (CNNs) learn aesthetic quality and functional form relationships from design databases without hand-crafted feature engineering Heidari and Iosifidis (2025). Modern CNNs process 3D CAD models, point clouds, and mesh representations for design similarity retrieval, manufacturability assessment, and defect detection. Transformer-based architectures such as vision-language models extend these capabilities by mapping rendered CAD views to structured design programs. OpenECAD, for example, generates executable sketch and 3D CAD operation sequences from images, producing fully editable CAD models Yuan et al. (2024).

Generative Design. Generative models (LLMs, VLMs, VAEs, GANs, diffusion models) enable automatic synthesis of novel geometries satisfying learned patterns Alam and Ahmed (2025); Li et al. (2025c). These models learn compressed latent representations, allowing interpolation between known designs and exploration of unconsidered solutions. Graph neural networks extend generation to topological structures such as assemblies or lattices. However, these models typically optimize reconstruction or perceptual similarity losses and do not enforce manufacturing

constraints or certification rules; generated CAD is therefore *plausible* rather than guaranteed feasible, and must be post-filtered by symbolic checkers or human experts.

Physics-Informed Neural Networks (PINNs) integrate conservation laws and boundary conditions into their loss functions, providing fast surrogates for stress analysis, thermal simulation, and other PDE-governed use-cases Cuomo et al. (2022). However, neural surrogates are prone to overfitting and provide approximations without formal error bounds. Thus, they may fail on out-of-distribution designs.

Multi-Objective Reinforcement Learning (MORL): learns policies that trade off cost, performance, weight, and manufacturability by approximating Pareto fronts Hayes et al. (2022). In product design, this supports automated exploration of design variants under competing objectives. However, purely neural MORL policies treat constraints as soft penalties or hidden rewards and cannot guarantee that all safety, tolerance, or regulatory constraints are satisfied without additional mechanisms. Moreover, they lack interpretability.

Limitations. Black-box neural models share three critical weaknesses: 1) They lack explicit design-rule representations and cannot provide structured explanations required for auditability and certification. 2) Generative models may produce geometries that violate manufacturing constraints since these constraints are not embedded in the training objectives. 3) Most critically, neural systems optimize expected performance over training distributions but offer no formal correctness guarantees, especially under distribution shift.

3.3 Neurosymbolic Integration Paradigms: Product Design Perspective

Neurosymbolic AI integrates neural learning with symbolic reasoning to overcome complementary limitations. Where symbolic systems provide verifiable constraint enforcement but lack learning capabilities, and neural networks enable data-driven optimization but cannot guarantee correctness, NeSy approaches yield systems that learn from experience while maintaining formal guarantees.

Integration Taxonomies. Kautz's taxonomy Kautz (2022) categorizes NeSy systems into six types (Figure 2): *Type I* embeds symbolic knowledge as neural features; *Type II* uses neural oracles within symbolic control; *Type III* maintains parallel modules with structured exchange; *Type IV* compiles symbolic rules into networks; *Type V* incorporates constraints as differentiable losses; *Type VI* interleaves symbolic layers in neural forward passes. Yu et al. propose an orthogonal classification based on information flow Yu et al. (2023): learning for reasoning, reasoning for learning, and fully integrated learning-reasoning systems.

Capabilities and Trade-offs. These integration paradigms enable design systems that learn from data while maintaining verifiable constraint satisfaction and interpretable rationale. Building on the cross-paradigm comparison in Table 2, we next relate each integration type to common design settings. Type II architectures are suitable for scenarios where symbolic planning predominates; Type V methods apply when

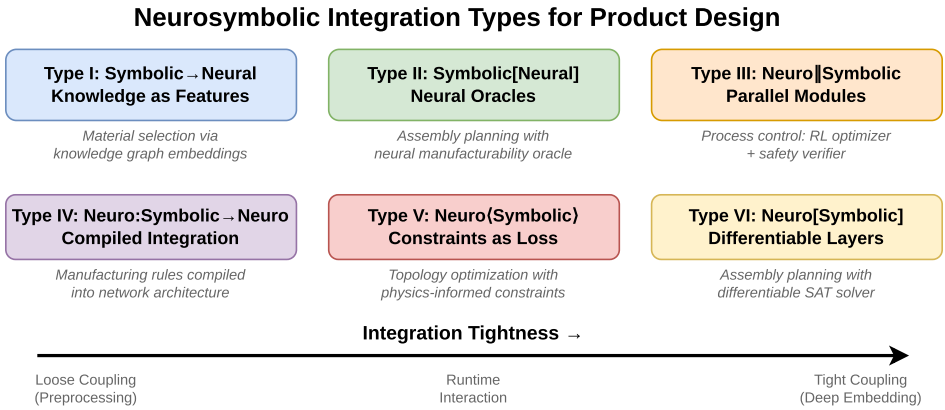

Figure 2. Neurosymbolic integration types (after Kautz), from loose to tight coupling: I neural features, II neural oracles, III parallel neural and symbolic modules, IV compiled rules, V constraints-as-loss, VI differentiable symbolic layers.

constraints can be approximated differentiably; Type VI designs enable real-time constraint checking within neural control loops.

Formal Characterization. Neurosymbolic product design can be formally characterized as constrained optimization over a hybrid space:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{X}} \; f_\theta(\mathbf{x}) \quad \text{subject to} \quad \Phi(\mathbf{x}) \models \top \tag{1}$$

where $f_\theta : \mathcal{X} \to \mathbb{R}$ denotes a neural objective (e.g., a learned performance surrogate or policy value), $\mathcal{X}$ is the design space (continuous parameters, discrete configurations, or both), and $\Phi(\mathbf{x}) \models \top$ asserts that the symbolic specification Φ—a conjunction of constraints $\phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_k$ encoding geometric validity, manufacturing rules, safety specifications, or ontological consistency—is satisfied by design $\mathbf{x}$. The six integration types in Kautz's taxonomy differ in *how* and *when* the symbolic specification Φ interacts with the neural objective f_θ:

- **Types I–II:** Φ shapes inputs or outputs (preprocessing/postprocessing)
- **Type III:** Φ verified in parallel, filtering or shielding neural proposals
- **Type IV:** Φ compiled into network structure
- **Type V:** Φ encoded as differentiable loss: $\mathcal{L} = f_\theta(\mathbf{x}) + \lambda \cdot \mathcal{L}_\Phi(\mathbf{x})$ where $\mathcal{L}_\Phi \to 0$ iff $\Phi(\mathbf{x}) \models \top$
- **Type VI:** Φ solved within forward pass via differentiable SAT/SMT/QP layers

For multi-objective settings common in product design, f_θ generalizes to a vector-valued function $\mathbf{f}_\theta : \mathcal{X} \to \mathbb{R}^m$, and the goal becomes approximating the constrained Pareto front:

$$\mathcal{P}^* = \{\mathbf{x} \in \mathcal{X} \mid \Phi(\mathbf{x}) \models \top \;\wedge\; \nexists\, \mathbf{x}' : \mathbf{x}' \succ_{\text{Pareto}} \mathbf{x}\} \tag{2}$$

Table 3. Neurosymbolic methods (Types I–VI) relevant to product design. Coding for Dom, NF, SS, (pd) and NR follows Table 1. A dynamically updated version of this comparison is maintained in the ORKG (Section 2).

Method	Type	Dom	NF	SS	Constraint / Symbolic Prior	Dataset / Benchmark	Key metrics
Periodic GT Yan et al. (2022)	I	Mat	S	C	crystal periodicity / structure priors	Materials property benchmarks	MAE (property prediction), Infer. Time
Onto-Speech Macioł et al. (2025)	I	QA	P	O	ontology-guided IE schema	production disturbance speech logs	Information Extraction quality (NR)
Onto-KR Yang et al. (2023)	I	QA	P	O	workflow ontology representation	InPro ontology in industrial knowledge base (5M data)	Consistency, Clarity, Coverage, Adaptability
ESNet Huang et al. (2024)	I	Mat	S	K	element-attribute KG	Material KG / multimodal atom data	Test-MAE, Ablation (#layers)
Onto-SAGCN Zhang et al. (2024)	I	QA	S	O	assembly ontology projected to a directed graph	aircraft assembly quality records (pd)	prediction error; ablation
KGE-Defect Wang and Cheung (2023)	I	QA	S	K/O	AM defect ontology; KG embeddings	additive manufacturing defect records (pd)	link-prediction accuracy; MRR; Hits@1/3/10
AKGNN-PC Xu et al. (2024)	I	Asm	S	K	assembly KG encoding error propagation	refrigeration compressor production data	error: MSE, MAE, RMSE, calibration: MAPE
KG-Design Liu et al. (2022)	I	Dsgn	S	K	design-specific KG schema and embeddings	smart-product design KG (pd)	prediction accuracy
TINN Terziyan and Vitko (2023)	II	Mfg	P	O	instance-class rel, Industry 4.0 Ontology	multi-source signal fusion (pd)	Lehmer mean loss, robustness
NeSyGeo Wu et al. (2025)	II	G	Gen	L/O	Geo-DSL; entity–attribute–relation geometry representation	NeSyGeo-Test (2,668 Q&A pairs)	benchmark accuracy; Pass Rate; CoT correctness
LLM4CAD Li et al. (2025b)	II	CAD	Gen	L/C	CAD command (parametric scripts)	Synt: CAD model, sketch, render-img	Parse rate, Intersection/union accuracy
OpenECAD Yuan et al. (2024)	II	CAD	Gen	C/L	CAD command syntax + sketch	SketchGraphs/Fusion360/DeepCAD; test-set: 57 designs	100-pt score; #executable/ #correct; inference time
APKG-CP Li et al. (2025a)	II	Asm	P	K/O	assembly-process ontology schema	enterprise assembly process documents	entity precision/recall/F1; KG retrieval time
APKG-DL Shi et al. (2024)	II	Asm	P	K	top-down assembly-process KG schema	assembly process text corpus (pd)	entity recognition accuracy
MachiningKB Guo et al. (2022)	II	Mfg	P	K/R	process KG compiled to a knowledge base	machining process corpus (pd)	extraction F-value
KG-LLM-QA Shao et al. (2026)	II	Asm	Gen	K	multi-hop reasoning paths over assembly KG	assembly process QA set (pd)	answer accuracy
ARKNESS Hoang et al. (2026)	II	Mfg	Gen	K	CNC machining KG grounding LLM generation	CNC process planning queries (pd)	accuracy; evidence traceability
Onto-RL Golpayegani et al. (2024)	III	Mfg	Opt	O	ontology-based adaptive reward machine	sustainable job-shop scheduling (pd)	resource/throughput/waiting-time
ASAP Tian et al. (2024)	III	Asm	Opt	C	physics feasibility (collision+stability)	Fusion360 (train-test: 1,906-240)	success rate (%); runtime vs #parts
Shield (MDP) Carr et al. (2023)	III	G	Opt	L	formal shield / safety specification	sparse-reward POMDP benchmarks	safety+return/convergence
Shield Odriozola-Olalde et al. (2025)	III	G	Opt	L	adaptive safety const. for robust shields	Control environments: e.g., navigation	safety+performance: efficiency, reward
INSIGHT Luo et al. (2024)	III	G	Opt	L	symbolic policy repr. (equation learner)	Nine Atari games: Arcade Learning Env	game score + textual explanation (NR)
3 Pathway Graf and Emami (2024)	III	G	Opt	L	interpretable/differentiable logic policies	Simul. building energy/ HVAC cntrl env	cost: energy + comfort-penalty, temp
NeSy-Vision Jang et al. (2026)	III	Asm	P	K/R	KG with spatial and precedence rules	annotated site imagery and activity clips (pd)	precision/recall/F1; throughput
cGAN+TO Wang et al. (2023)	III	Topo	Gen	C	topology optimization enforcing volume constraints	2D structural design problems	constraint satisfaction; compliance
HCP-KG Wang et al. (2024)	III	QA	P/Opt	O/K/R	HCP onto. (OWL); SWRL rules; CBR+RBR reasoning; GCN	automotive production line and gear manufacturing (deployed)	nugget prediction accuracy, Hits@1/ 3/ 10
SPRING Jacobson and Xue (2025)	IV	Dsgn	Gen	R	symbolic spatial CSP (sampling/search)	Interior design generation benchmarks	constraint satisfaction; image quality; human study
McGAN Wang et al. (2025)	IV	Mfg	Gen	R	Manufacturing rules embedded in cGAN	2D part designs for injection molding	rule-satisfaction; quality; robustness
DL2 Fischer et al. (2019)	V	G	Opt/S	L	first-order logic loss constraint	Image (MNIST, FASHION, CIFAR-10)	Constraint satisfaction rate, accuracy
Neurcadrecon Dong et al. (2024)	V	CAD	S	C	zero Gaussian curvature prior	CAD surface reconstruction (pd)	Chamfer dist, F1-score, Consistency
GINN Berzins et al. (2025)	V	Topo	S	C	Geometric & topological design const.	PDE/geometry tasks, 3D jet-engine (pd)	Constraint-satisfaction errors; diversity
PINN-Topo Jeong et al. (2023b)	V	Topo	Opt/S	C	Structural equilibrium & boundary cond.	2D-3D structural topology opt. problems	compliance; volume fraction
TOuNN Chandrasekhar and Suresh (2021)	V	Topo	Opt/S	C	NN-parameterised density field; design constraints	2D–3D compliance minimisation problems	compliance; volume fraction
SB-GCN-Screw Long et al. (2026)	V	Asm	S	C	screw-theory DOF constraints as consistency loss	B-Rep assembly mates (pd)	NLL; constrained-DOF variance
OptNet Amos and Kolter (2017)	VI	G	Opt	C	differentiable quadratic program layer	synthetic QP tasks; 4×4 Sudoku	loss convergence, runtime, MSE, accuracy
SATNet Wang et al. (2019)	VI	G	Opt	L	differentiable MAXSAT layer	Sudoku, parity sequences, visual Sudoku	clause satisfaction (solve rate), accuracy
CVXPY Agrawal et al. (2019)	VI	G	Opt	C	differentiable convex program	logistic regression; stochastic control	runtime/canonicalization; control cost/test loss
NURBS-Diff Prasad et al. (2022)	VI	CAD	Opt/S	C	Geometric: NURBS curves and surfaces	point-cloud reconst., CAD geometry	fitting/reconstruction error; runtime/GPU efficiency

The following section applies this taxonomy to classify existing methods across product design domains, revealing systematic patterns in how different integration types align with specific engineering tasks. In Section 7, Φ and $\mathcal{P}^*$ are revisited as the quantities that an evaluation protocol has to measure.

4 Classification of Neurosymbolic Methods for Product Design

An overview of the classified methods is provided in Table 3, which brings the representative systems identified through the main and supplementary engineering-database searches under a common coding scheme. The table classifies 38 systems by integration type (I–VI), symbolic substrate, neural function, and evaluation domain. The broader SLR corpus comprises 70 papers; structured records for the classified systems and contextual studies are maintained in the companion ORKG comparison. This section examines the patterns that emerge from this classification.

Distribution of integration types. Types I–III constitute the majority of product-design-evaluated systems in our corpus, whereas Types IV–VI appear more often in general benchmarks (denoted "G") than in end-to-end engineering workflows. This asymmetry reflects practical constraints: loose couplings preserve modularity, enable independent validation of symbolic components, and integrate naturally with existing CAD, simulation, and PLM toolchains. By contrast, tighter integrations (Types V–VI) often require domain-specific differentiable formulations; available for geometric primitives (e.g., NURBS-Diff) and physics losses (e.g., PINNs) but less mature for discrete manufacturing rules or assembly precedence constraints.

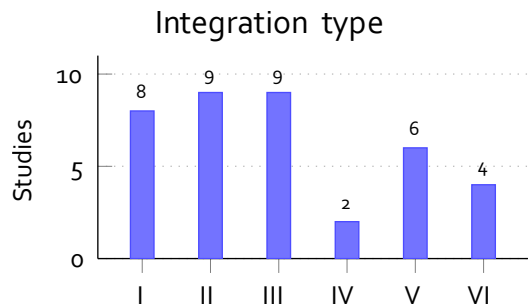

Figure 3. Distribution of integration types across the classified systems.

Among systems evaluated in engineering domains, Types I–III dominate, particularly combinations of learned components with ontologies and knowledge graphs. Type V appears mainly in physics- and constraint-informed topology optimization, whereas Types IV and VI remain rare in end-to-end engineering workflows. Quantitatively, Types I–III account for 26 of the 38 classified systems (Figure 3), indicating that tighter neural–symbolic coupling remains comparatively uncommon in product-design applications.

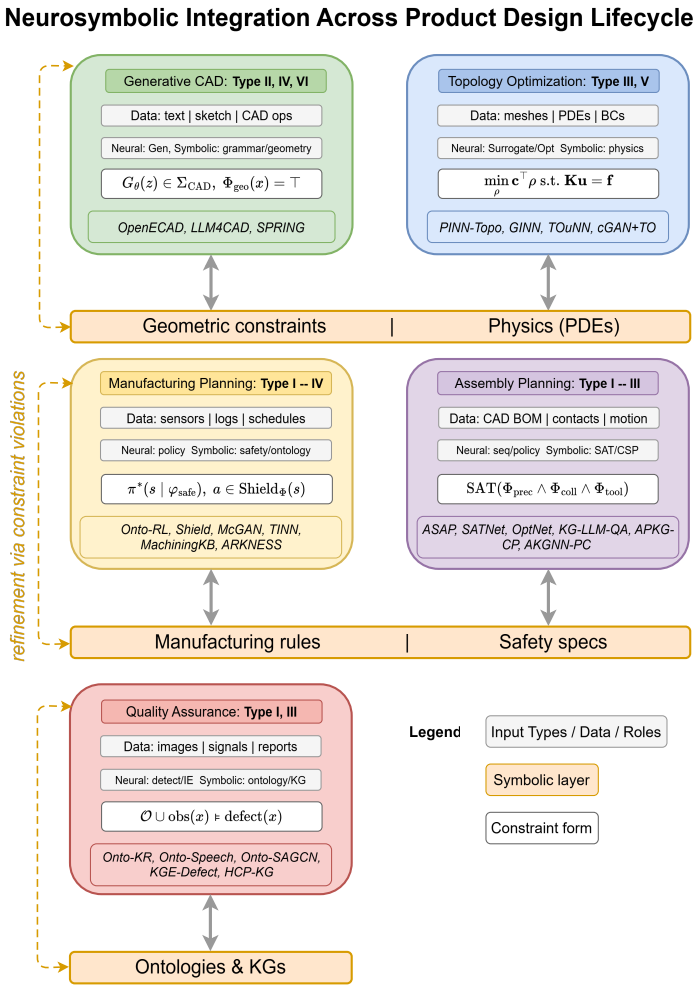

Figure 4. Neurosymbolic integration across the product-design lifecycle (survey lens). Representative stages, integration types, methods, and symbolic substrates are shown. Bidirectional arrows denote neural–symbolic interaction, the dashed arrow denotes iterative refinement, and boxed expressions represent constraint formulations. Examples are illustrative rather than exhaustive.

Lifecycle view. To contextualize the taxonomy within an engineering workflow, Figure 4 maps the dominant integration patterns onto canonical product design stages. Stages here correspond to the evaluation domains coded in Table 3, but a stage may draw on systems from several domains. The figure is not a normative pipeline but a *survey lens*: it illustrates where neural and symbolic components interact through constraint checking, shielding, verification, or structured knowledge exchange, and why feedback

loops arise when candidate designs fail validation. Across stages, these interactions instantiate the constrained optimization formulation of Equation 1; in multi-objective settings, the learned objective becomes a vector of competing criteria, e.g.,

$$\mathbf{J}(\mathbf{x}) = \left[J_{\text{cost}}(\mathbf{x}),\ J_{\text{time}}(\mathbf{x}),\ J_{\text{CO}_2}(\mathbf{x}),\ -J_{\text{quality}}(\mathbf{x})\right], \tag{3}$$

and the goal is to approximate the constraint-feasible Pareto set $\mathcal{P}^*$ (Equation 2).

Symbolic substrates and neural functions. The unified classification in Table 3 reveals recurring pairings between engineering tasks and symbolic–neural components. Constraint- and physics-based substrates (C) are prominent in topology optimization and CAD reconstruction, where governing equations and geometric requirements can be incorporated into learning or optimization. Ontologies and knowledge graphs (O,K) occur frequently in manufacturing, assembly, and quality-related applications, where heterogeneous engineering knowledge must be represented explicitly and connected to learned models. Logic and rule substrates (L,R) are more common in safety-critical shielding and rule-guided generation, where explicit specifications can restrict or verify model outputs.

Constraints are the most frequent symbolic substrate, followed by knowledge graphs, logic, ontologies, and rules (Figure 5).

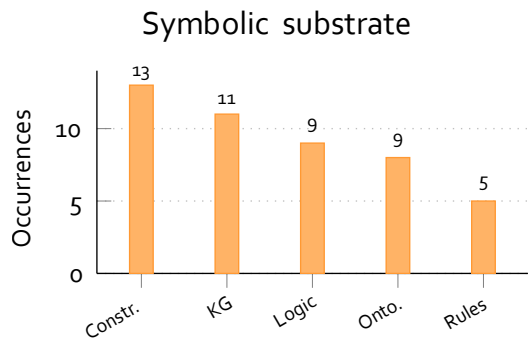

Figure 5. Symbolic substrates used across the classified systems.

Neural configurations. Optimization is the most frequent neural function, followed by surrogate modelling, generation, and perception. Their pairing with symbolic substrates is task-dependent, and graph networks are the most common architecture family (Figure 6).

Ontologies and knowledge graphs in engineering practice. Explicit knowledge representations are particularly prominent in manufacturing, assembly, and quality-related systems. Examples include assembly-process knowledge graphs Li et al. (2025a); Shi et al. (2024), machining knowledge bases Guo et al. (2022), and ontology- or KG-based approaches to defect diagnosis and quality control Wang and Cheung (2023); Wang et al. (2024); Zhang et al. (2024). More recent systems combine such

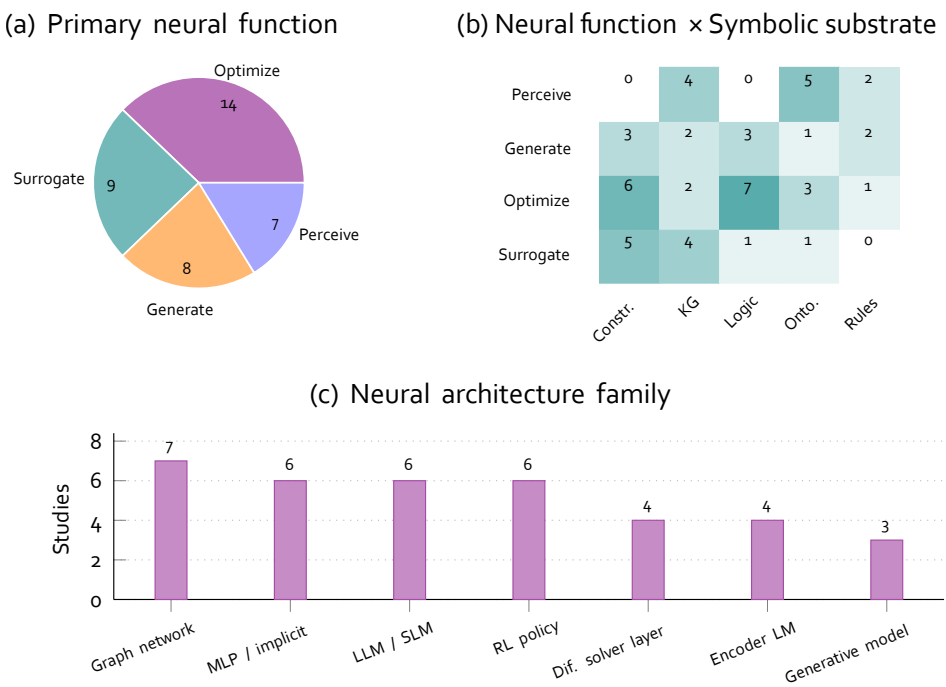

Figure 6. Neural components in the reviewed systems: (a) primary function, (b) function–substrate pairings, and (c) architecture families. Two studies were excluded from (c) due to combined neural architecture; systems with two neural functions contribute to both rows in (b).

representations with language models for process planning and assembly question answering Hoang et al. (2026); Shao et al. (2026); Section 5.6 examines how these systems adapt the language model and couple it to the symbolic layer. In contrast, tightly coupled differentiable symbolic solvers and logic layers remain scarcely represented in engineering applications.

Engineering methods nevertheless incorporate substantial structural knowledge in other forms. For example, B-rep topology guides message passing Lee et al. (2023), governing equations enter differentiable objectives Jeong et al. (2023a), CAD structure is embedded in the decoder Zhang et al. (2025), and physical fields condition generative models Nie et al. (2021). These approaches provide useful domain priors but do not always expose engineering knowledge as an independently represented symbolic component that can be inspected, modified, or reused. Making such knowledge explicit, while retaining the flexibility of learned models, therefore represents an important opportunity for neurosymbolic product-design systems.

Domain-specific patterns. Across application domains, generative CAD emphasizes programmatic representations and editability, topology optimization consistently incorporates physical priors, and manufacturing and assembly planning prioritize

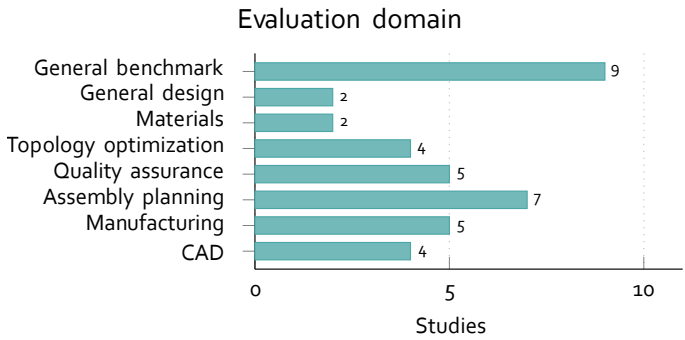

Figure 7. Evaluation domains represented across the classified systems.

symbolic feasibility and safety constraints. Together, these patterns highlight both the diversity of neurosymbolic design strategies and the lack of standardized, cross-domain evaluation protocols. Yet, coverage across these domains remains uneven. Quality assurance and assembly planning are among the better represented engineering domains Zhang et al. (2024); Wang and Cheung (2023); Wang et al. (2024); Xu et al. (2024); Li et al. (2025a); Shi et al. (2024); Shao et al. (2026), whereas materials remain comparatively sparse (Figure 7), with relatively few studies explicitly coupling learned components with symbolic knowledge or constraints. This imbalance indicates that the adoption of neurosymbolic methods is progressing at different rates across product-design domains and points to materials and sustainability as areas with substantial room for further development.

Evaluation gaps. A key limitation across the classified systems in Table 3 is the heterogeneity of evaluation protocols. Reported metrics range from geometric fidelity and prediction error to task success, constraint satisfaction, reasoning quality, and domain-specific process measures, making direct cross-method comparison difficult. Evaluation settings are similarly diverse, spanning synthetic benchmarks, simulated environments, public engineering datasets, enterprise records, and deployed industrial systems (Section 5.4). This diversity points to a recurring trade-off between realistic industrial evaluation and reproducibility, since stronger industrial evidence typically relies on proprietary data and systems. In Section 7, we therefore synthesize the reviewed studies along common evaluation dimensions: task performance, constraint compliance, reasoning quality, robustness, efficiency, reproducibility, and industrial relevance.

5 Applications of Neurosymbolic AI in Product Design

Building on the unified classification in Table 3, this section examines how the identified integration patterns are instantiated across major product-design applications. Across the design-to-manufacturing pipeline, these methods support geometry generation, structural optimization, manufacturing and assembly planning, and quality assurance by combining learned models with explicit engineering knowledge and constraints.

5.1 Generative CAD and Structural Optimization

In generative CAD, neural models such as VAEs, GANs, diffusion models, and LLM/VLM-based generators enable rapid exploration of geometric design spaces but frequently produce outputs that violate geometric, manufacturing, or assembly constraints. A common response is to embed symbolic structure directly into the generation process, for example, through programmatic CAD representations, constraint-aware decoding, or rule-guided generation pipelines. Recent LLM-symbolic hybrid systems translate images or natural language into executable CAD programs rather than static meshes, enabling editability and downstream verification (e.g., OpenECAD, LLM4CAD) Yuan et al. (2024); Li et al. (2025b).

A similar integration is critical in topology and structural optimization, where neural surrogates accelerate simulation-heavy design cycles but struggle to generalize under novel loads or boundary conditions. NeSy approaches augments these surrogates with physics-informed formulations that incorporate governing PDEs, and with differentiable logical or SAT/QP layers that enforce safety, connectivity, and manufacturability constraints. Examples include geometry- and physics-informed neural networks and differentiable CAD layers that embed curvature, equilibrium, or geometric validity constraints directly into optimization loops Jeong et al. (2023b); Dong et al. (2024); Berzins et al. (2025).

Related formulations dispense with the finite-element solve altogether, parameterising the density field with a network trained under explicit design and manufacturing constraints Chandrasekhar and Suresh (2021), or retain a conventional optimizer as a second stage that enforces functional requirements on generated candidates Wang et al. (2023). The two strategies trade differently: embedding the constraint in the loss keeps the pipeline end-to-end differentiable, whereas a separate optimization stage preserves an established constraint-handling procedure whose behaviour is more directly interpretable to practitioners.

5.2 Manufacturing Optimization and Material Compatibility

In manufacturing, hybrid approaches address settings where safety and feasibility are central. In process optimization tasks such as additive manufacturing, machining, and thermal control, pure RL agents may explore unsafe operating regions. NeSy-RL addresses this by combining neural policies with symbolic monitors, temporal logic specifications, or ontology-guided abstractions that restrict exploration to feasible regions and provide interpretable failure explanations Carr et al. (2023); Odriozola-Olalde et al. (2025); Golpayegani et al. (2024).

Knowledge graphs play a complementary role on the process-planning side, where the relevant knowledge is textual rather than dynamic. Machining expertise dispersed across technical documents can be extracted by neural sequence labelling and compiled into a queryable process knowledge base Guo et al. (2022), and grounding a language model in such a graph yields process recommendations that remain traceable to the evidence supporting them Hoang et al. (2026). The symbolic layer here serves interoperability

and auditability rather than constraint enforcement; a distinction that recurs throughout manufacturing applications.

Material selection poses similarly complex challenges due to coupled mechanical, thermal, chemical, and cost considerations Kusiak (2020). For material selection, knowledge-graph and logic–neural approaches combine learned compatibility scores with explicit engineering knowledge over assembly and environmental constraints Huang et al. (2024). For example, element-attribute material knowledge graphs combined with graph or transformer architectures improve property prediction and generalize to sparse or emerging materials, while periodic graph transformers encode crystal symmetries to enhance prediction accuracy on standard materials benchmarks Yan et al. (2022).

5.3 Assembly, Design Exploration, and Quality Assurance

Assembly planning and downstream design exploration require both scalability and formal correctness. These methods combine learned sequence or policy models with symbolic constraint checking, such as differentiable SAT or physics-based feasibility checks, to support assembly sequences that are collision-free and respect precedence and tool access. Robotic assembly systems exemplify this paradigm by combining learned action proposals with symbolic verification of physical feasibility Tian et al. (2024); Wang et al. (2019). Sequence generation is only part of the problem, since planning depends on assembly knowledge that is itself dispersed across drawings, process sheets and enterprise records. Knowledge graphs are increasingly used to consolidate it: process graphs populated by neural extraction from enterprise documents Li et al. (2025a); Shi et al. (2024), graphs encoding error-propagation relations that support performance prediction on production data Xu et al. (2024), and multi-hop reasoning over such graphs to answer process questions posed in natural language Shao et al. (2026).

In design-space exploration, symbolic modules can govern discrete architectural choices while neural models optimize parametric configurations or approximate Pareto fronts. Multi-objective NeSy-RL supports transparent trade-off analysis across cost, performance, manufacturability, and sustainability. In these frameworks, symbolic representations provide interpretable explanations for constraint-driven decisions Hayes et al. (2022); Reymond et al. (2022); Luo et al. (2024).

Quality assurance draws on both capabilities at once. Neural vision and signal models detect surface or structural anomalies, while ontologies and knowledge graphs give explicit form to process, product and defect knowledge, supporting traceable root-cause analysis and compliance checking Yang et al. (2023). Several systems couple the two directly. An assembly ontology projected into a directed graph supplies the structure over which attention-based prediction of aircraft assembly quality operates Zhang et al. (2024); embeddings learned over a defect knowledge graph support diagnosis in additive manufacturing where labelled failures are scarce Wang and Cheung (2023); and a human–cyber–physical ontology with associated inference rules drives case retrieval and diagnosis alongside a graph convolutional reasoner, reported in operation on an automotive production line and a gear manufacturing process Wang et al. (2024). Ontology-based workflow models and NeSy design frameworks additionally

structure inspection criteria and non-conformance handling, enabling more explainable and auditable diagnoses in regulated manufacturing workflows Arachchige et al. (2025).

5.4 Industrial Case Study: HCP-KG Quality Control

The HCP-KG system of Wang et al. (2024) is the clearest deployed case in our corpus and is worth examining against the taxonomy of Section 4 and the evaluation dimensions of Section 7. It organises heterogeneous human, cyber and physical production knowledge through a modular ontology and knowledge graph. The symbolic layer holds the OWL ontology, SWRL rules and a case base used for retrieval and reasoning; the learned layer performs information extraction from maintenance reports and troubleshooting manuals, and graph-convolutional prediction over the resulting graph. The two remain separately inspectable and exchange structured representations, which places the system in Type III. Figure 8 shows the operational interface: the three knowledge partitions are colour-coded in the central graph, the panels below give the entities extracted from each, and the left-hand list gives the quality-control tasks served from the shared representation.

The evidence is stronger than in most of the corpus because the system was run in automotive body-shop and gear-manufacturing environments rather than only on offline benchmarks. Reported results include a welding nugget-diameter prediction accuracy of 95.3% and knowledge-graph retrieval at Hits@1/3/10 of 51.8/97.8/99.7, together with deployment across multiple quality-control applications, including weld inspection, nugget-diameter prediction, stamping-defect identification, and equipment troubleshooting. Applied to Table 4, the case provides particularly strong evidence for task performance and deployment validity, while reproducibility and the causal contribution of the symbolic layer remain harder to assess. Neither the ontology, the rule set, nor the production data is released. More importantly, the symbolic contribution is never isolated: no ablation removes the rules or the case base, and no experiment revises the specification to test whether the system responds consistently. The reported results therefore do not establish how much of the observed performance derives specifically from the symbolic knowledge rather than from the learned components or their interaction. The case shows what deployed neurosymbolic quality control looks like, and equally why task performance, reasoning fidelity and deployment validity have to be reported separately (Section 7.3).

5.5 Sustainable and Human–AI Design

Neurosymbolic methods also support sustainability-oriented design and human–AI collaboration by integrating reasoning, learning, and explainability. Ontologies and causal KGs structure lifecycle and sustainability assessment, enabling early-stage what-if analysis and transparent evaluation of environmental trade-offs. Multi-objective NeSy frameworks balance environmental impact, cost, and performance while generating explanations suitable for certification and stakeholder communication Golpayegani et al. (2024); Bougzime et al. (2025). Quantitative support for such assessments increasingly comes from learned surrogates over life cycle inventories. Neural models have been

Figure 8. Deployed HCP-KG interface for manufacturing quality control. Human, physical, and cyber production knowledge are unified in a single graph serving multiple quality-control applications on the production line. Reproduced from Wang et al. (2024).

used to fill inventory data gaps where measurements are unavailable Khadem et al. (2022), to couple property prediction with LCA-driven multi-objective optimization in materials selection Dabbaghi et al. (2021), and to automate life cycle assessment for additive manufacturing through a co-designed product–process framework validated on a fused-filament case study Naser et al. (2023). These approaches are data-driven rather than neurosymbolic, since the inventory model remains external to the learner. They nevertheless establish the quantitative substrate on which ontology- and causal-KG-based sustainability reasoning must ultimately operate. Ontology-based process knowledge has meanwhile been applied directly to circular-economy planning, for instance in remanufacturing process selection He et al. (2020), indicating where a declarative layer adds the traceability that a surrogate alone cannot provide.

Human–AI collaboration is enhanced through LLM–symbolic interfaces that translate natural-language requirements into verifiable design constraints. Parallel NeSy architectures provide real-time feedback during design exploration, while meta-cognitive components detect high-uncertainty scenarios, solicit human guidance, and maintain audit trails linking decisions, constraints, and rationale. This integration supports traceable and accountable design, bridging the gap between automated exploration and human judgment Luo et al. (2024); Kautz (2022).

5.6 Practical Integration of LLM Components

LLM-based systems in the reviewed corpus already occupy different roles. LLM4CAD[‡] and OpenECAD generate executable CAD representations, ARKNESS grounds

[‡]Model name acronyms are specified in Table 3

manufacturing responses in a machining knowledge graph, and KG-LLM-QA combines language models with multi-hop reasoning over an assembly-process graph Li et al. (2025b); Yuan et al. (2024); Hoang et al. (2026); Shao et al. (2026). Their adaptation strategies are equally diverse. LLM4CAD evaluates GPT-4/GPT-4V largely in a zero-shot setting and adds an iterative debugger; OpenECAD fine-tunes pretrained visual-language models on CAD operation data; KG-LLM-QA includes an LLM-tuning stage for question-answering-chain generation; and ARKNESS leaves the language model unchanged while supplying external, provenance-linked knowledge at inference time.

These adaptation choices concern the neural component of the architecture. Parameter-efficient fine-tuning (PEFT) is particularly relevant when engineering data are limited or models are too large to retrain in full, because only a small subset of additional parameters is optimized while the base model remains largely fixed. In a neurosymbolic system, however, PEFT determines only how the neural component is adapted; the neurosymbolic character comes from its interaction with explicit knowledge, constraints, rules, or verification mechanisms.

Choosing how to adapt the model. The appropriate strategy depends on whether the engineering problem concerns *knowledge* or *behaviour*. When domain knowledge changes frequently, for example through new machine capabilities, revised standards, or updated product families, retrieval-augmented generation (RAG) or knowledge-graph grounding can keep that information outside the model weights. This makes updates possible without retraining and, when provenance is preserved, allows a recommendation to be traced back to the engineering records that support it. This is the role played by the machining knowledge graph in ARKNESS, where retrieved evidence is supplied to the language model before process recommendations are generated Hoang et al. (2026).

Fine-tuning becomes more relevant when the required change is behavioural rather than factual, such as consistently producing a domain-specific vocabulary, command syntax, or structured output format. Full fine-tuning is one option, as demonstrated by OpenECAD, while PEFT methods reduce the amount of trainable model state. Low-rank adaptation (LoRA), for example, keeps the pretrained model weights fixed and learns low-rank updates to selected layers Hu et al. (2022). Quantized LoRA (QLoRA) combines such adapters with a quantized frozen base model, reducing the memory required for adaptation while retaining the same basic separation between the base model and the learned adapters Dettmers et al. (2023). Retrieval and fine-tuning are therefore complementary rather than interchangeable: retrieval changes the knowledge available at inference time, whereas fine-tuning changes how the model responds to that knowledge and to the task.

Coupling the LLM to the symbolic layer. The reviewed systems suggest several ways of placing symbolic computation around a language model. One pattern treats the LLM as a generator of a structured engineering artefact and uses an external component to determine whether that artefact can be executed or accepted. This is visible in CAD generation: OpenECAD produces editable CAD-operation sequences, while LLM4CAD executes the generated program and can return execution errors to an iterative debugger Yuan et al. (2024); Li et al. (2025b).

A second pattern places explicit engineering knowledge *before* generation. Here the symbolic representation restricts or enriches the context from which the LLM forms its answer. ARKNESS follows this arrangement by retrieving provenance-linked machining knowledge before generating process-planning responses Hoang et al. (2026). A third pattern uses the symbolic representation as an intermediate reasoning structure rather than only as input context. KG-LLM-QA first represents the question as a reasoning chain, resolves that chain through multi-hop reasoning over the assembly-process graph, and then uses the LLM to express the resulting answer Shao et al. (2026).

Taken together, these patterns place the symbolic component at different points in the inference process: before generation to ground the context, during inference to structure reasoning, or after generation to execute and verify a produced artefact.

From checking to actionable feedback. What remains comparatively sparse is feedback about the *engineering meaning* of a failed candidate. LLM4CAD demonstrates that an execution error can be returned to the generator and used for repair, but the feedback primarily concerns whether the CAD program executes rather than whether the resulting design satisfies a manufacturing, physical, or assembly specification. A tighter neurosymbolic loop would return structured diagnostics; for example, the violated precedence relation, colliding components, failed tolerance, or exceeded physical limit, and use that information to revise the candidate. The non-LLM literature already supplies many of the required checking mechanisms, including physical-feasibility tests for assembly planning and explicit constraints in topology optimization Tian et al. (2024); Wang et al. (2023). Connecting such diagnostics back to an LLM generator is therefore a concrete direction for moving from generation followed by checking toward iterative, constraint-guided engineering design.

6 Cross-Domain Transfer Potential

The synthesis in Section 4 shows that product-design applications remain concentrated in Types I–III, while tightly coupled Types IV–VI are comparatively scarce in engineering workflows despite established examples in general AI benchmarks. This motivates looking beyond product design for integration mechanisms that provide tighter coupling between learning, structured reasoning, and explicit constraint handling. Relevant architectures have been explored in robotics, game playing, energy systems, and combinatorial optimization, where symbolic specifications and controlled evaluation environments make such integration easier to study.

These systems appear in Table 3 primarily under the domain label “G” (general benchmark). We treat them as architectural transfer candidates rather than demonstrated product-design solutions. Their differentiable solvers, shielding mechanisms, and structured-prior injections address recurring problem structures, such as satisfiability, safe exploration, and structured knowledge, that also arise in geometric validity, assembly precedence, and manufacturing feasibility. Their effectiveness in end-to-end engineering workflows, however, remains largely unvalidated. This section therefore highlights four transfer patterns and the conditions that would need to be met for their adoption in product design.

Differentiable Symbolic Layers (Type VI) place constraint verification inside the network's forward pass. OptNet Amos and Kolter (2017) and CVXpyLayers Agrawal et al. (2019) demonstrate differentiable quadratic and convex program layers that could optimize CAD parameters under geometric constraints. SATNet Wang et al. (2019) shows that differentiable satisfiability solving can enforce complex logical constraints, suggesting applications to assembly precedence, collision rules, or configuration compatibility.

Explainable NeSy-RL (Type III) addresses interpretability requirements in regulated industries. INSIGHT Luo et al. (2024) learns symbolic policy representations with textual explanations, while logic tensor networks Graf and Emami (2024) provide interpretable control policies specified in logical form. Adapting these patterns to design and manufacturing would support audit trails for safety-critical decisions, e.g., explaining why a particular process plan or design variant satisfies given constraints.

Safety-Critical Shielding (Type III) ensures formal correctness during learning and execution. Shield-based methods Carr et al. (2023); Odriozola-Olalde et al. (2025) restrict RL exploration to safe regions via temporal-logic specifications and adaptive constraint sets. The same construction suggests a route toward machining, thermal processing, and robotic assembly, where symbolic envelopes could define safe operating regimes and filter unsafe actions before execution. An open question is whether engineering safety requirements can be represented with sufficiently compact and tractable specifications for such shielding mechanisms.

Structured Priors and Rule-Guided Generation (Types II and IV) show how symbolic structure can be injected into perception and generation without full RL. Taxonomy-informed neural networks (TINN) embed an Industry 4.0 ontology into the network architecture to improve robustness and data efficiency in smart-manufacturing settings Terziyan and Vitko (2023). Similarly, SPRING integrates a symbolic spatial constraint solver into an interior-design generator, demonstrating how rule-guided sampling can enforce layout constraints by construction Jacobson and Xue (2025). These patterns map naturally onto parametric CAD design, assembly layout, and facility planning, where taxonomies and spatial rules are often already available. They may therefore provide a comparatively direct path for introducing symbolic structure into existing learning pipelines.

Comparability Caveat: Direct performance comparison across these cross-domain methods remains challenging. Game scores (e.g., INSIGHT on Atari) Luo et al. (2024), control costs (e.g., logic-based HVAC control in Three Pathways) Graf and Emami (2024), and simulated manufacturing throughput (e.g., Onto-RL for sustainable job-shop scheduling) Golpayegani et al. (2024) rely on incommensurable, domain-specific metrics. Section 7 therefore treats these results as evidence that a given integration mechanism is viable, rather than as measurements directly comparable with those reported for product-design systems, and identifies the evaluation dimensions along which such comparisons could eventually be made.

7 Evaluation Frameworks

Evaluation of neurosymbolic engineering systems must account for more than the quality of their final predictions or designs. A system may achieve strong task performance while violating explicit engineering requirements, or may appear to use symbolic knowledge while relying primarily on statistical correlations Renkhoff et al. (2024); Bortolotti et al. (2024). The evaluation problem therefore has three related levels: the quality of the engineering result, the correctness and contribution of the symbolic component, and the robustness and practical evidence of the complete system. We first organize these levels into an operational protocol and then examine how they can be instantiated for the engineering tasks represented in the reviewed literature.

7.1 Operational Evaluation Protocol

In Figure 9 we present a straightforward framework to summarize the evaluation procedure we propose in this review. Let $\mathcal{D}$ denote the evaluation set, s the neurosymbolic system under evaluation, Φ the applicable symbolic specification, and $Q(s; \mathcal{D})$ a task-specific performance score oriented such that larger values indicate better performance. Evaluation begins by defining the engineering task, its objectives, and the rules, constraints, or other symbolic requirements that determine whether a result is acceptable. The complete system is then evaluated under its intended operating conditions to establish an unperturbed baseline for task performance, constraint compliance, and, where several objectives compete, multi-objective quality.

The baseline is subsequently subjected to targeted diagnostic interventions. Input perturbations and unseen cases test whether performance and feasibility persist outside the nominal evaluation distribution. Ablating or perturbing symbolic information tests whether that information actually contributes to the result, while modifying rules or constraints tests whether the system responds consistently to a revised specification. The resulting changes are compared with the baseline and reported together with computational, reproducibility, and deployment evidence. This separates three questions that are otherwise easily conflated: whether the engineering output is good, whether the symbolic requirements are satisfied and used, and whether the observed behaviour remains credible beyond the nominal benchmark.

7.2 Task-Specific Engineering Performance

The task score Q in Figure 9 cannot be instantiated by a single metric across CAD generation, topology optimization, assembly planning, manufacturing, and quality control. Even within one task, a single score can capture only part of engineering quality. The measures in Table 4 should therefore be interpreted as complementary rather than interchangeable.

CAD generation and reconstruction. In CAD generation downstream tasks, geometric fidelity is commonly measured after sampling point sets P and G from the generated and reference geometries. Representative geometric measures include the

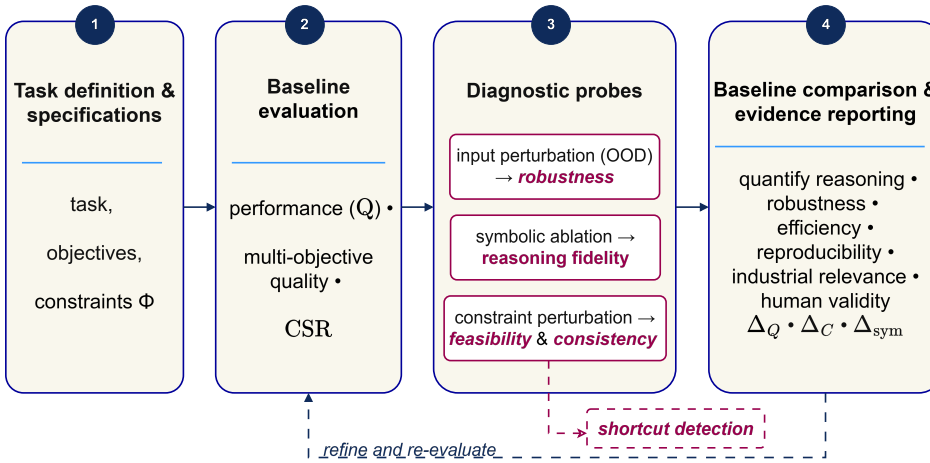

Figure 9. Operational evaluation protocol for neurosymbolic product-design systems. An unperturbed baseline is established before targeted input and symbolic interventions are applied. Their effects are compared with the baseline and reported together with system-level evidence.

Chamfer and Hausdorff distances Regenwetter et al. (2023). Chamfer distance can be formulated using the following:

$$d_{\text{CD}}(P,G) = \frac{1}{|P|}\sum_{p\in P}\min_{g\in G}\|p-g\|_2^2 + \frac{1}{|G|}\sum_{g\in G}\min_{p\in P}\|g-p\|_2^2, \tag{4}$$

and the Hausdorff distance:

$$d_{\text{H}}(P,G) = \max\left\{\max_{p\in P}\min_{g\in G} d(p,g),\ \max_{g\in G}\min_{p\in P} d(g,p)\right\}. \tag{5}$$

These measures quantify geometric agreement, but not whether a generated model is usable as an engineering artefact. A geometrically similar result may still contain an invalid boundary representation (B-Rep), broken topology, self-intersections, or an operation history that cannot be regenerated or edited. CAD evaluation should therefore combine geometric similarity with validity and downstream editability or operation-success tests Willis et al. (2021). This distinction is important because geometric proximity alone is not a reliable proxy for manufacturability or functional performance Regenwetter et al. (2023).

Structural and topology optimization. For structural and topology-design tasks, the engineering objective is often physical rather than geometric. Structural compliance (C), for example, is calculated as follows:

$$C(x) = \mathbf{f}^\top \mathbf{u}(x), \tag{6}$$

where $\mathbf{f}$ is the applied load and $\mathbf{u}(x)$ the displacement resulting from design x. Material use is commonly reported through the volume fraction $v_f(x) = \sum_e \rho_e v_e / \sum_e v_e$, where ρ_e denotes the material-density variable of element e. Reporting compliance without volume, stress, connectivity, or manufacturing feasibility can favour structurally effective but unusable solutions. Conversely, similarity to a reference topology is not necessarily meaningful when several mechanically equivalent designs exist.

Manufacturing and assembly planning. Performance is naturally expressed in terms of whether a feasible process or sequence is produced and how costly it is to execute in assembly-related downstream tasks. For a plan π with operation completion times C_j, its makespan is $T_{\max}(\pi) = \max_j C_j$.

A planning system should therefore not be assessed through makespan alone: precedence relations, collision avoidance, tool accessibility, resource availability, and process rules determine whether the nominally short sequence is executable. Depending on the task, complementary measures include valid-plan rate, number or severity of violated prerequisites, assembly/disassembly cost, and the number of replanning or solver steps required.

Prediction and quality control. Prediction-oriented applications require a different interpretation of Q. For surrogate and material-property prediction, numerical accuracy is commonly reported through 'mean absolute error' and 'root mean square error': $\mathrm{MAE} = N^{-1} \sum_i |y_i - \hat{y}_i|$ and $\mathrm{RMSE} = \sqrt{N^{-1} \sum_i (y_i - \hat{y}_i)^2}$.

These metrics measure numerical accuracy but should be accompanied by checks of physical consistency when the predicted quantity is constrained by known engineering relations. For *quality inspection and defect detection*, precision (P), recall (R), and their harmonic mean, which is $F_1 = 2PR/(P + R)$, should be used. These measures capture recognition quality, whereas decision latency and the quality of the downstream diagnosis or corrective action become important in deployed manufacturing settings.

When knowledge-graph completion or retrieval is itself part of the task, ranking quality can be reported through $\mathrm{Hits@}k = N^{-1} \sum_i \mathbf{1}[\mathrm{rank}_i \leq k]$. The measure $\mathrm{Hits@}k$ quantifies retrieval accuracy, but does not establish that the retrieved knowledge was subsequently used correctly in the engineering decision.

7.3 Symbolic Correctness and Reasoning Fidelity

Task performance alone does not establish the defining property of a neurosymbolic system: that explicit knowledge constrains or contributes to the result. Two complementary questions should therefore be evaluated separately: whether the produced output satisfies the symbolic specification, and whether the symbolic component was actually consequential in producing that output.

Constraint compliance. The first question is whether the produced output satisfies the applicable symbolic specification. A standard approach to quantifying this compliance is via the **Constraint Satisfaction Rate (CSR)**. Following recent engineering-design benchmarking practice Regenwetter et al. (2025), we define CSR as the proportion of evaluated outputs that simultaneously satisfy the applicable specification:

Table 4. Operational evaluation rubric for neurosymbolic engineering systems. Measures are representative examples; the third column states the required evidence needed to accompany the reported score.

Evaluation target	Representative measures	Report alongside
Engineering result (Section 7.2; Section 7.4)		
CAD geometry	Chamfer, Hausdorff	B-Rep validity, topology, regeneration and editability
Structural / design quality	Compliance, volume fraction, stress	Physical feasibility; manufacturing constraints
Manufacturing / assembly planning	Success rate, makespan, cost	Precedence, collision, accessibility, rule satisfaction
Prediction / quality control	MAE, RMSE, precision, recall, F_1	Physical consistency; downstream decision quality
Multi-objective quality	Hypervolume, Pareto coverage	Objectives, reference point, constraints, stakeholder preference
Symbolic contribution (Section 7.3)		
Constraint compliance	CSR, violation rate and severity	Whether enforcement is hard, soft, or post-hoc
Reasoning fidelity	Δ_{sym}, symbolic perturbation	Which artefact was ablated; consistency of the response with the revised Φ
Explanation traceability	Trace agreement or coverage; expert judgement	Reference trace where available; evaluator, task, assessed property
System credibility (Section 7.5; Section 7.6)		
Robustness	Δ_Q, Δ_{CSR}	Perturbation type, severity, OOD construction
Computational efficiency	Latency, memory, throughput, solver and tool calls	Hardware, model configuration, solver tolerance, workload and ontology size
Deployment validity	Task-completion time, decision latency, override rate	Deployment setting, data provenance, expert role, online vs. offline
Reproducibility	Repeated-run variance; artefact completeness	Code, data, model version, symbolic artefacts, seeds, hardware, solver settings

$$\mathrm{CSR}(s) = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \mathbf{1}[\Phi(s(x)) = \mathrm{true}]\,. \tag{7}$$

where Φ acts as a symbolic verification oracle mapping the substrate evaluations to a Boolean valuation based on engineering rules.

Crucially, the structural interpretation of the CSR depends heavily on how Φ enters the system. Constraints enforced structurally by construction behave differently from soft behavioral penalties applied during optimization, and both differ from validation criteria applied solely during post-generation filtering. The specific enforcement mechanism should therefore be explicitly reported alongside the absolute satisfaction rate. Furthermore, when physical or manufacturing boundaries vary across design examples, reporting the per-constraint violation frequency and violation severity offers deeper diagnostic utility. For formally specified sequential requirements or dynamic

processes, temporal-logic satisfaction and per-episode violation measures provide the corresponding system-level view Carr et al. (2023); Renkhoff et al. (2024).

Symbolic contribution and reasoning fidelity. Constraint satisfaction, however, does not prove that symbolic knowledge is *load-bearing*. A model may produce valid outputs because the same regularity is recoverable statistically from its training data. Where the architecture allows controlled intervention, the symbolic contribution can be estimated as:

$$\Delta_{\text{sym}} = Q(s_{\Phi}; \mathcal{D}) - Q(s_{\neg\Phi}; \mathcal{D}), \tag{8}$$

where $s_{\neg\Phi}$ denotes the corresponding system after removing, masking, or perturbing the intended symbolic contribution. The intervention itself must be reported: removing a knowledge-graph edge, disabling a rule module, masking a constraint, or replacing a verified solver with an unconstrained component are not equivalent ablations. Across the evaluation fields coded in Table 3, we did not identify an evaluation that isolates the symbolic contribution through a controlled symbolic-side ablation, nor one that tests sensitivity to a deliberately revised specification. Reported ablations instead concern model or architectural choices rather than the symbolic substrate itself.

A stronger diagnostic changes the symbolic specification rather than simply removing it. If an assembly precedence relation, geometric constraint, material rule, or process requirement is modified, the predicted design or decision should change in a direction consistent with the revised Φ. Failure to respond suggests that the symbolic representation may be present in the architecture without materially controlling its behaviour. Concept accuracy and completeness provide related representation-level checks and have been used to expose reasoning shortcuts in NeSy benchmarks Bortolotti et al. (2024). Thus, constraint satisfaction asks whether the result is valid, whereas reasoning fidelity asks whether the symbolic knowledge was actually relevant to obtaining it.

Explanation traceability. Interpretability should not be inferred simply from the presence of an ontology, knowledge graph, rule set, or textual rationale. An explanation may be plausible without faithfully reflecting the information used to reach the decision. Where systems expose rules, KG paths, constraints, or intermediate symbolic states, evaluation should therefore assess whether the explanation can be traced to the symbolic elements actually involved in inference and whether it covers the decisive requirements. Where a reference reasoning trace is available, trace agreement or coverage can be reported; otherwise, expert assessment should state who evaluated the explanation, against which task or reference, and for which property, such as correctness, completeness, or usefulness. In engineering applications, this amounts to tracing a recommended design or process decision back to the constraints, rules, or evidence that support it Luo et al. (2024); Hoang et al. (2026).

7.4 Conflicting Objectives and Engineering Trade-offs

Multi-objective formulation. Product-design decisions rarely optimize one quantity in isolation. A design variable vector x may simultaneously determine mass, stiffness,

manufacturing cost, assembly effort, energy use, or lifecycle impact. A constrained multi-objective problem can be written as

$$\min_{x\in\Omega} \mathbf{F}(x) = [f_1(x), \ldots, f_m(x)] \qquad \text{s.t.} \quad \Phi(x) = \text{true}, \tag{9}$$

where improvement in one f_i may degrade another. Reducing the mass of a component may increase compliance; shortening an assembly sequence may require more expensive components or tooling; and reducing lifecycle impact may conflict with material or manufacturing requirements. Evaluation should therefore distinguish *how trade-offs are generated* from *how the resulting trade-off set is judged*.

A common strategy is scalar aggregation. After suitable normalization of objectives, a weighted sum takes the form

$$g_{\text{WS}}(x|\mathbf{w}) = \sum_{i=1}^{m} w_i \tilde{f}_i(x), \qquad w_i \geq 0, \quad \sum_i w_i = 1. \tag{10}$$

This is easy to interpret but fixes one preference vector and can miss parts of a non-convex Pareto front. Engineering requirements that are better expressed as limits can instead be handled through an ϵ-constraint formulation,

$$\min_x f_1(x) \qquad \text{s.t.} \quad f_j(x) \leq \epsilon_j,\ j = 2, \ldots, m, \qquad \Phi(x) = \text{true}. \tag{11}$$

For instance, mass may be minimized while manufacturing cost, maximum stress, and assembly time are kept below prescribed thresholds. Decomposition-based methods such as MOEA/D Zhang and Li (2007); Qi et al. (2014) generate multiple trade-offs by associating subproblems with different preference directions. A representative Tchebycheff scalarization is

$$g_{\text{TCH}}(x|\mathbf{w}, \mathbf{z}^*) = \max_i \left\{ w_i \left| \tilde{f}_i(x) - z_i^* \right| \right\}, \tag{12}$$

where $\mathbf{z}^*$ is an ideal objective vector. Adaptive decomposition or preference-conditioned approaches can further move or reweight these directions when the Pareto geometry is irregular or stakeholder priorities change. Such adaptation is particularly relevant to interactive design, where a designer may initially explore the solution space and later prioritize, for example, cost over weight or manufacturability over geometric novelty Reymond et al. (2022).

Generating and evaluating engineering trade-offs. Once the objectives and constraints are defined and a set A of non-dominated alternatives has been generated, their quality must be evaluated independently of the procedure that produced it. Hypervolume Zitzler et al. (2007) measures the objective-space region dominated by A with respect to a reference point $\mathbf{r}$,

$$\text{HV}(A; \mathbf{r}) = \lambda_m \left(\bigcup_{\mathbf{a}\in A} [a_1, r_1] \times \cdots \times [a_m, r_m] \right), \tag{13}$$

where λ_m denotes m-dimensional volume. Hypervolume rewards both convergence and spread, while Pareto coverage can compare whether one returned set dominates alternatives from another method. These measures should be reported together with feasibility: a visually diverse Pareto front is not useful if large portions violate manufacturing, physical, or symbolic requirements. Hypervolume is among the most established indicators for evaluating Pareto-front approximations Zitzler et al. (2007); Audet et al. (2021); it has also been adopted for evaluating multi-objective generative-design quality Regenwetter et al. (2023).

7.5 Robustness, Generalization, and Computational Cost

Robustness and generalization. Engineering robustness should consider both task quality and feasibility. A design generator may retain geometric accuracy on an unseen product family while producing more constraint violations; conversely, conservative symbolic checking may preserve feasibility while task performance deteriorates. For an in-distribution (ID) evaluation and a corresponding out-of-distribution (OOD) or perturbed evaluation, the two changes can be reported as

$$\Delta_Q = Q_{\mathrm{ID}} - Q_{\mathrm{OOD}}, \qquad \Delta_{CSR} = \mathrm{CSR}_{\mathrm{ID}} - \mathrm{CSR}_{\mathrm{OOD}}. \tag{14}$$

The construction of the OOD condition is therefore part of the evaluation rather than an implementation detail. Depending on the application, useful shifts include unseen part geometries, changed loads or boundary conditions, new materials, altered manufacturing parameters, missing or noisy sensor measurements, and combinations not represented during training. Where possible, several perturbation severities should be evaluated rather than reporting a single OOD point. Changes to rules or constraints can also be applied, although their primary purpose is different: in Section 7.3 they test reasoning fidelity, whereas here the interest is whether overall task quality and feasibility remain stable under changing operating conditions.

Computational cost. Computational efficiency should similarly be measured at the level of the complete hybrid pipeline. Symbolic scalability must also be characterized when ontologies or formal reasoners participate in that pipeline. Singh et al. (2025) argue that neurosymbolic reasoner benchmarks should vary ontology size and complexity, knowledge-base structure, ontology profile, and reasoning task. For engineering systems, this suggests reporting reasoning time and memory as the ontology or knowledge graph grows. Wall-clock time, peak memory, throughput, and the number of simulation, optimization, solver, retrieval, or verification calls are often more informative than neural inference time alone Colelough and Regli (2024). For CAD and topology optimization, mesh resolution, solver tolerance, and the number of finite-element or physical simulations can dominate runtime. For LLM-based systems, evaluation should additionally report the model and version, context size, decoding configuration, number of generated candidates or agent/tool calls, token use, and end-to-end latency. Without these settings, comparisons between a local model, an LLM–solver pipeline, and an agentic system can attribute cost to the wrong component.

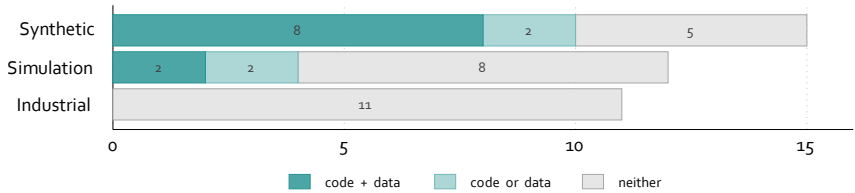

Figure 10. Code and data availability by evaluation setting. Categories are mutually exclusive; industrial evaluations in this corpus release neither.

7.6 Reproducibility and Deployment Validity

Reproducibility. Reproducing a neurosymbolic system requires more than releasing the learned model. Where possible, studies should provide the code, evaluation data, model configuration and checkpoints, together with the symbolic artefacts on which the claimed behaviour depends: ontologies, knowledge graphs, rule sets, constraints, formal specifications, and solver configurations. Random seeds, hardware, preprocessing, library versions, and optimization tolerances should also be reported when they materially affect the result. For proprietary industrial data, releasing the underlying records may be impossible, but the schema, task construction, evaluation protocol, and non-sensitive symbolic specification can often still be documented.

Artifact availability in the reviewed literature varies with the evaluation setting. Synthetic and benchmark-oriented studies more often release code and data, whereas industrial evaluations frequently rely on partially or fully closed artefacts (Figure 10). This distinction is important when interpreting apparently stronger real-world evidence: deployment may increase ecological validity while simultaneously reducing independent reproducibility.

Deployment validity. Deployment evidence should be described explicitly rather than reduced to the label "industrial". At minimum, studies should distinguish evaluation on synthetic or simulated data, retrospective evaluation on real industrial data, expert-in-the-loop evaluation, and operation in a deployed production workflow. Relevant measures can include task-completion time, decision latency, expert agreement, intervention or override rate, and the number of cases requiring manual correction. The role of the engineer must also be stated: expert verification of an offline output provides different evidence from continuous use of the system in an operational decision loop. The HCP-KG deployment discussed in Section 5.4 illustrates the latter setting by integrating heterogeneous production knowledge and learned components within deployed quality-control workflows Wang et al. (2024). Reporting these conditions makes it possible to separate algorithmic performance from evidence that the system is usable, maintainable, and trusted in the engineering environment for which it was designed.

8 Risks and Challenges

Moving neurosymbolic systems from controlled benchmarks into product-design and manufacturing workflows introduces failure modes that predictive performance alone does not capture. In a hybrid system, problems may originate inside the neural–symbolic architecture, at its interface with engineering infrastructure, or in the organizational context that must accept and act on its outputs. We use these three points of failure to organize the challenges in Figure 11 into technical, industrial/operational, and governance/ethical dimensions. This complements Section 7: the evaluation framework asks whether a system behaves as intended, whereas the discussion below considers how that behaviour can be preserved once the system is updated, integrated, and deployed.

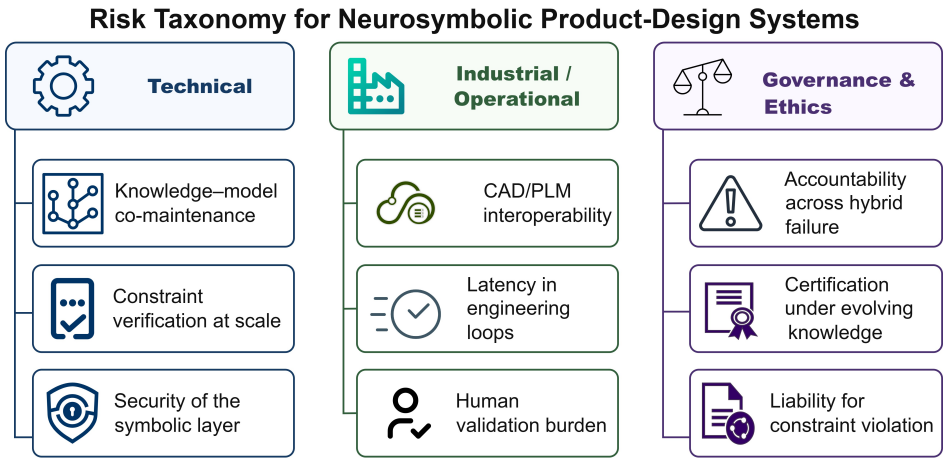

Figure 11. Risk taxonomy Challenges span technical, industrial/operational, and governance/ethical dimensions.

8.1 Technical Challenges

Knowledge–model co-maintenance. Neural models and symbolic artefacts can change independently even though their behaviour is coupled. Updating a tolerance, ontology relation, or process rule may change the conditions under which a trained model is considered correct; retraining the model can in turn alter which cases reach the symbolic layer. A practical way to manage this dependency is to version the model checkpoint, ontology or knowledge graph, rule set, and solver configuration as one system release. A small regression set linking representative inputs to their expected symbolic verdicts can then be re-run whenever either side changes. Knowledge maintenance then becomes a testable engineering activity. Where the underlying records identify individual operators, data-protection obligations extend to the knowledge representation as well as to the training data.

Constraint verification at scale. The value of explicit engineering constraints depends on whether they can still be checked when the design space grows. Verification that is tractable for a small CAD part or short assembly sequence can become costly for assemblies with many interacting components, large knowledge bases, or iterative optimization loops. The problem is particularly visible when symbolic reasoning itself becomes a computational bottleneck: description-logic benchmarks, for example, show that reasoning cost depends strongly on ontology size, expressivity, and reasoning task Singh et al. (2025). Scalability should therefore be evaluated not only for neural inference but also for constraint checking, solver calls, and symbolic reasoning, as discussed in Section 7.5.

Security of the symbolic layer. A hybrid architecture introduces attack and integrity risks that do not appear when only the learned model is considered. For example, removing a manufacturing rule, changing a tolerance, or altering an ontology relation can modify downstream decisions while leaving the neural component untouched Jalaian and Bastian (2023). Symbolic artefacts should therefore be managed with the same discipline as executable code: changes can be versioned and reviewed, their provenance recorded, and deployed knowledge bases checked against the validated release.

8.2 Industrial and Operational Challenges

CAD/PLM interoperability. A NeSy system that performs well in isolation still has to exchange engineering artefacts with the tools used for design, simulation, production planning, shop-floor operation, and product lifecycle management (PLM). Geometry, constraints, process knowledge, and model outputs must move between these systems without losing identifiers, units, tolerances, revision information, or the semantics on which reasoning depends. The useful abstraction here is an interface contract rather than another learned component. To ensure this, symbolic entities should be tied to stable engineering identifiers, exchanged representations should preserve the semantics on which reasoning depends, and transformations between systems should remain traceable. On the knowledge side, provenance-linked systems such as ARKNESS illustrate the same principle by retaining links between generated process recommendations and the records supporting them Hoang et al. (2026).

Latency in engineering loops. Neurosymbolic workflows can combine neural inference, retrieval, symbolic reasoning, optimization, simulation, and verification in a single decision loop. A runtime that is acceptable for offline design exploration may therefore be unsuitable for interactive CAD assistance, process control, or quality monitoring. This distinction becomes important in deployed settings: HCP-KG, for example, reports production monitoring at up to 200 Hz Wang et al. (2024). End-to-end latency should consequently be reported together with the cost of its constituent neural and symbolic operations rather than neural inference time alone, following the evaluation recommendations in Section 7.5.

Human validation burden. Explicit knowledge and reasoning traces can make a system easier to inspect, but they do not remove the need for engineering judgement.

Experts may still have to validate extracted knowledge, resolve inconsistent rules, review constraint violations, and decide whether an automatically generated design or process recommendation is acceptable. As the number of generated alternatives or symbolic checks increases, this can shift rather than eliminate engineering effort. Therefore, human involvement should also be reported as part of deployment validity, including when and why expert intervention is required, rather than being summarized only by the phrase "human in the loop" (Section 7.6).

8.3 Governance and Ethical Challenges

Accountability across hybrid failure. When a hybrid recommendation is wrong, the failure may originate in the training data, the learned model, an outdated rule, incomplete knowledge, a solver configuration, or the interaction between them. Responsibility is therefore difficult to reconstruct after the event unless the relevant state of the system was recorded Novelli et al. (2024). For consequential outputs, a compact decision record can provide that trace: the model and knowledge-base versions, relevant input identifier or hash, symbolic rules or knowledge path used, solver configuration, and final verification outcome. Such records also support the explanation-traceability tests introduced in Section 7.3. The aim is not to log every internal activation, but to preserve enough evidence to determine which component supported or failed to block the engineering decision.

Certification under evolving knowledge. Certification becomes particularly difficult when a system changes after the evidence supporting its validation has been produced. Updating a learned model, ontology, constraint set, or engineering standard can invalidate earlier test results even when the rest of the system remains unchanged. In industrial settings especially, minor changes can be checked first against the established regression suite, whereas changes that alter feasible behaviour or the interaction between neural and symbolic components require broader re-verification. The important point is to define the change policy before deployment, so that updating engineering knowledge does not turn certification into an ad hoc decision each time the system evolves.

Liability for constraint violation. The presence of a symbolic constraint does not by itself guarantee that an engineering constraint will be respected. As discussed in Section 7.3, constraints may be enforced by construction, represented as soft losses, or checked only after generation; these mechanisms provide very different levels of assurance. A plausible CAD model can therefore remain non-manufacturable, an assembly plan can violate a precedence relation, or a process recommendation can exceed an operating limit even when the system contains an explicit symbolic layer. Liability therefore depends not only on what constraints were represented, but on how they were enforced, verified, and communicated to the responsible engineer. Auditable decision histories and explicit documentation of these assumptions are therefore central to accountable deployment Novelli et al. (2024).

In industrial settings, the symbolic layer may itself contain proprietary engineering knowledge, including geometry, tolerances, process capabilities, supplier knowledge, and accumulated defect histories. Access control therefore protects not only system integrity

but also intellectual property. This constraint should also be considered when interpreting the lower artifact availability observed for industrial evaluations in Figure 10.

9 Conclusions

Neurosymbolic artificial intelligence offers a promising route toward product-design systems that combine data-driven adaptation with explicit engineering knowledge and constraint handling. Across a corpus of 70 studies and 38 classified systems, this survey finds that current practice is dominated by loosely coupled integrations, particularly Types I–III, where symbolic knowledge acts as a structured prior, interface, or parallel reasoning and verification component. Tighter couplings (Types IV–VI) remain comparatively sparse and concentrated in narrowly structured subproblems, indicating that their transfer to end-to-end engineering workflows is still limited.

Evaluation emerges as the more consequential gap. Reported protocols are sufficiently heterogeneous to make cross-method comparison difficult, and we did not identify a system that explicitly isolates the contribution of its symbolic component through a controlled symbolic-side ablation or deliberately revised specification. We therefore proposed an operational protocol and reporting rubric that separates engineering task performance, symbolic contribution, and system credibility, and illustrated its relevance through a deployed industrial case study. Scalability, real-time constraint verification, and interoperability with existing CAD and product-lifecycle-management workflows remain open engineering problems. Progress will require evaluation practices that remain valid as both learned components and the engineering knowledge they use are updated over the system lifecycle.

10 Availability of Resources

Living resource vs. static snapshot: Table 3 provides a curated snapshot to support the narrative structure of this review and to remain stable across versions of the manuscript. To enable continuous updates, structured querying, and community extension (e.g., filtering by integration type, domain, substrate, or evaluation evidence), we maintain a living comparison in the Open Research Knowledge Graph (ORKG). The version accompanying this manuscript is archived at doi.org/10.48366/R1937457; The comparison itself continues to be updated. Moreover, an accompanying GitHub repository serves as the curation and dissemination hub (taxonomy figures, “awesome-list” browsing, and contribution workflow), while ORKG remains the canonical, machine-actionable version of the comparison. Thus, the ORKG comparison remains open to community contribution and extension as the literature evolves.

Acknowledgements

This work was supported by the DFG Priority Programme SPP 2443 *Human Decision* (Grant No. 543081196) and the German Ministry of Education and Research (BmBF) for the project KISSKI AI Service Center (01IS22093C).

Declaration of conflicting interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

Acharya K, Raza W, Júnior CMJMD, Velasquez A and Song HH (2024) Neurosymbolic reinforcement learning and planning: A survey. *IEEE Trans. Artif. Intell.* 5(5): 1939–1953.

Agrawal A, Amos B, Barratt ST, Boyd SP, Diamond S and Kolter JZ (2019) Differentiable convex optimization layers. In: *NeurIPS*. pp. 9558–9570.

Alam MF and Ahmed F (2025) Gencad: Image-conditioned computer-aided design generation with transformer-based contrastive representation and diffusion priors. *Trans. M.L. Res.* .

Amos B and Kolter JZ (2017) Optnet: Differentiable optimization as a layer in neural networks. In: *ICML*. PMLR, pp. 136–145.

Arachchige PJ, Iancu B and Lilius J (2025) A roadmap toward neurosymbolic approaches in AI design. *IEEE Access* 13: 174368–174392.

Audet C, Bigeon J, Cartier D, Digabel SL and Salomon L (2021) Performance indicators in multiobjective optimization. *Eur. J. Oper. Res.* 292(2): 397–422.

Berzins A, Radler A, Volkmann E, Sanokowski S, Hochreiter S and Brandstetter J (2025) Geometry-informed neural networks. In: *ICML*, *Proceedings of Machine Learning Research*, volume 267. PMLR / OpenReview.net.

Besold TR, d'Avila Garcez AS, Bader S, Bowman H, Domingos PM, Hitzler P, Kühnberger K, Lamb LC, Lima PMV, de Penning L, Pinkas G, Poon H and Zaverucha G (2021) Neural-symbolic learning and reasoning: A survey and interpretation. In: *Neuro-Symbolic Artificial Intelligence*, *Frontiers in Artificial Intelligence and Applications*, volume 342. IOS Press, pp. 1–51.

Bortolotti S, Marconato E, Carraro T, Morettin P, van Krieken E, Vergari A, Teso S and Passerini A (2024) A neuro-symbolic benchmark suite for concept quality and reasoning shortcuts. In: *NeurIPS*.

Bougzime O, JABBAR S, Cruz C and Demoly F (2025) Evaluating neuro-symbolic AI architectures: Design principles, qualitative benchmark, comparative analysis and results. In: *19th Intern. Conf. on NeuSy Learn. and Reas.*

Carr S, Jansen N, Junges S and Topcu U (2023) Safe reinforcement learning via shielding under partial observability. In: *AAAI*. AAAI Press, pp. 14748–14756.

Chandrasekhar A and Suresh K (2021) Tounn: topology optimization using neural networks. *Structural and Multidisciplinary Optimization* 63: 1135–1149.

Colelough BC and Regli W (2024) Neuro-symbolic AI in 2024: A systematic review. In: *LNSAI@IJCAI*, *CEUR Workshop Proceedings*, volume 3819. CEUR-WS.org.

Cuomo S, Cola VSD, Giampaolo F, Rozza G, Raissi M and Piccialli F (2022) Scientific machine learning through physics-informed neural networks: Where we are and what's next. *J. Sci. Comput.* 92(3): 88.

Dabbaghi F, Tanhadoust A, Nehdi M, Nasrollahpour S, Dehestani M and Yousefpour H (2021) Life cycle assessment multi-objective optimization and deep belief network model for sustainable lightweight aggregate concrete. *Journal of Cleaner Production* 318: 128554.

d'Avila Garcez A and Lamb LC (2023) Neurosymbolic AI: the 3rd wave. *Artif. Intell. Rev.* 56(11): 12387–12406.

Dettmers T, Pagnoni A, Holtzman A and Zettlemoyer L (2023) Qlora: Efficient finetuning of quantized llms. In: *NeurIPS*.

Dong Q, Xu R, Wang P, Chen S, Xin S, Jia X, Wang W and Tu C (2024) Neurcadrecon: Neural representation for reconstructing CAD surfaces by enforcing zero gaussian curvature. *ACM Trans. Graph.* 43(4): 51:1–51:17.

Fischer M, Balunovic M, Drachsler-Cohen D, Gehr T, Zhang C and Vechev MT (2019) DL2: training and querying neural networks with logic. In: *ICML*, *Proceedings of Machine Learning Research*, volume 97. PMLR, pp. 1931–1941.

Golpayegani F, Ghanadbashi S and Zarchini A (2024) Advancing sustainable manufacturing: Reinforcement learning with adaptive reward machine using an ontology-based approach. *Sustainability* 16(14): 5873.

Graf P and Emami P (2024) Three pathways to neurosymbolic reinforcement learning with interpretable model and policy networks. *CoRR* abs/2402.05307.

Guo L, Yan F, Li T, Yang T and Lu Y (2022) An automatic method for constructing machining process knowledge base from knowledge graph. *Robotics Comput. Integr. Manuf.* 73: 102222.

Hao J, Zhao L, Milisavljevic-Syed J and Ming Z (2021) Integrating and navigating engineering design decision-related knowledge using decision knowledge graph. *Adv. Eng. Informatics* 50: 101366.

Hayes CF, Radulescu R, Bargiacchi E, Källström J, Macfarlane M, Reymond M, Verstraeten T, Zintgraf LM, Dazeley R, Heintz F, Howley E, Irissappane AA, Mannion P, Nowé A, de Oliveira Ramos G, Restelli M, Vamplew P and Roijers DM (2022) A practical guide to multi-objective reinforcement learning and planning. *Auton. Agents Multi Agent Syst.* 36(1): 26.

He Y, Hao C, Wang Y, Li Y, Wang Y, Huang L and Tian X (2020) An ontology-based method of knowledge modelling for remanufacturing process planning. *Journal of Cleaner Production* 258: 120952.

Heidari N and Iosifidis A (2025) Geometric deep learning for computer-aided design: A survey. *IEEE Access* 13: 119305–119334.

Hoang D, Gorsich D, Castanier MP and Imani F (2026) Knowledge graph fusion with large language models for accurate, explainable manufacturing process planning. *International Journal of Production Economics* 300: 110091.

Hu EJ, Shen Y, Wallis P, Allen-Zhu Z, Li Y, Wang S, Wang L and Chen W (2022) Lora: Low-rank adaptation of large language models. In: *ICLR*. OpenReview.net.

Huang C, Chen C, Shi L and Chen C (2024) Material property prediction with element attribute knowledge graphs and multimodal representation learning. *CoRR* abs/2411.08414.

Jacobson MJ and Xue Y (2025) Integrating symbolic reasoning into neural generative models for design generation. In: *AAAI*. AAAI Press, p. 28741.

Jalaian B and Bastian ND (2023) Neurosymbolic AI in cybersecurity: Bridging pattern recognition and symbolic reasoning. In: *MILCOM*. IEEE, pp. 268–273.

Jang J, Jeong E, Cho J and Kim TW (2026) Neuro-symbolic vision framework for construction: Knowledge graph-augmented monitoring and productivity analysis of precast concrete assembly. *Journal of Computational Design and Engineering* 13: 90–108.

Jeong H, Bai J, Batuwatta-Gamage CP, Rathnayaka C, Zhou Y and Gu Y (2023a) A physics-informed neural network-based topology optimization (pinnto) framework for structural optimization. *Engineering Structures* 278: 115484.

Jeong H, Batuwatta-Gamage C, Bai J, Xie YM, Rathnayaka C, Zhou Y and Gu Y (2023b) A complete physics-informed neural network-based framework for structural topology optimization. *Computer Methods in Applied Mechanics and Engineering* 417: 116401. DOI: https://doi.org/10.1016/j.cma.2023.116401.

Kautz HA (2022) The third AI summer: AAAI robert s. engelmore memorial lecture. *AI Mag.* 43(1): 93–104.

Khadem SA, Bensebaa F and Pelletier N (2022) Optimized feed-forward neural networks to address co2-equivalent emissions data gaps–application to emissions prediction for unit processes of fuel life cycle inventories for canadian provinces. *Journal of Cleaner Production* 332: 130053.

Kim K, Manley DG and Yang HJ (2006) Ontology-based assembly design and information sharing for collaborative product development. *Comput. Aided Des.* 38(12): 1233–1250.

Kusiak A (2020) Convolutional and generative adversarial neural networks in manufacturing. *Int. J. Prod. Res.* 58(5): 1594–1604.

Lee J, Yeo C, Cheon S, Park JH and Mun D (2023) Brepgat: Graph neural network to segment machining feature faces in a b-rep model. *J. Comput. Des. Eng.* 10(6): 2384–2400.

Li K, Liu J, Zhai S, Zhuang C and Pei F (2025a) Automatic generation method of knowledge graph for complex product assembly processes based on text mining. *Chinese Journal of Mechanical Engineering* 38: 133.

Li X, Sun Y and Sha Z (2025b) LLM4CAD: multimodal large language models for three-dimensional computer-aided design generation. *J. Comput. Inf. Sci. Eng.* 25(1).

Li Y, Lin C, Liu Y, Long X, Zhang C, Wang N, Li X, Wang W and Guo X (2025c) Caddreamer: CAD object generation from single-view images. In: *CVPR*. Computer Vision Foundation / IEEE, pp. 21448–21457.

Liu A, Zhang D, Wang Y and Xu X (2022) Knowledge graph with machine learning for product design. *CIRP Annals* 71: 117–120.

Long T, Guo B, Jin J, Kong C, Zhang C, Xu Y, Wang W, Cao Y, Li T and Xu Z (2026) Neuro-symbolic graph learning for probabilistic tolerance modeling in mechanical assemblies. *Procedia CIRP* 145: 200–205.

Luo L, Zhang G, Xu H, Yang Y, Fang C and Li Q (2024) End-to-end neuro-symbolic reinforcement learning with textual explanations. In: *ICML*, *Proceedings of Machine Learning Research*, volume 235. PMLR / OpenReview.net, pp. 33533–33557.

Macioł A, Macioł P, Gumienny G and Wrzała K (2025) A new ontology-based approach to automatic information extraction from speech for production disturbance management. *The*

Inter. J. Adv. Manufact. Tech. 136(7): 3735–3752.

Nardi D and Brachman RJ (2003) An introduction to description logics. In: *Description Logic Handbook*. Cambridge University Press, pp. 1–40.

Naser AZ, Defersha F, Xu X and Yang S (2023) Automating life cycle assessment for additive manufacturing with machine learning: Framework design, dataset buildup, and a case study. *Journal of Manufacturing Systems* 71: 504–526.

Nie Z, Lin T, Jiang H and Kara LB (2021) Topologygan: Topology optimization using generative adversarial networks based on physical fields over the initial domain. *Journal of Mechanical Design* 143: 031715.

Novelli C, Taddeo M and Floridi L (2024) Accountability in artificial intelligence: what it is and how it works. *AI Soc.* 39(4): 1871–1882.

Odriozola-Olalde H, Zamalloa M, Arana-Arexolaleiba N and Pérez-Cerrolaza J (2025) Towards robust shielded reinforcement learning through adaptive constraints and exploration: The fear field framework. *Eng. Appl. Artif. Intell.* 144: 110055.

Prasad AD, Balu A, Shah H, Sarkar S, Hegde C and Krishnamurthy A (2022) Nurbs-diff: A differentiable programming module for NURBS. *Comput. Aided Des.* 146: 103199.

Qi Y, Ma X, Liu F, Jiao L, Sun J and Wu J (2014) MOEA/D with adaptive weight adjustment. *Evol. Comput.* 22(2): 231–264.

Regenwetter L, Obaideh YA, Chiotti F, Lykourentzou I and Ahmed F (2025) Bikebench: A bicycle design benchmark for generative models with objectives and constraints. In: *NeurIPS*.

Regenwetter L, Srivastava A, Gutfreund D and Ahmed F (2023) Beyond statistical similarity: Rethinking metrics for deep generative models in engineering design. *Comput. Aided Des.* 165: 103609.

Renkhoff J, Feng K, Meier-Doernberg M, Velasquez A and Song HH (2024) A survey on verification and validation, testing and evaluations of neurosymbolic artificial intelligence. *IEEE Trans. Artif. Intell.* 5(8): 3765–3779.

Reymond M, Bargiacchi E and Nowé A (2022) Pareto conditioned networks. In: *AAMAS*. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), pp. 1110–1118.

Roijers DM, Vamplew P, Whiteson S and Dazeley R (2013) A survey of multi-objective sequential decision-making. *J. Artif. Intell. Res.* 48: 67–113.

Rosell J (2004) Assembly and task planning using petri nets: a survey. *Proceedings of the institution of mechanical engineers, part B: journal of engineering manufacture* 218(8): 987–994.

Shao P, Huang Z, Qiao L, Xu X, Wan Y, Chen C, Li Z, Anwer N and Qie Y (2026) A large language model-enhanced knowledge graph multi-hop reasoning method for assembly process question-answering. *J. Comput. Inf. Sci. Eng.* 26(8).

Shi X, Tian X, Ma L, Wu X and Gu J (2024) A knowledge graph–based structured representation of assembly process planning combined with deep learning. *The International Journal of Advanced Manufacturing Technology* 133: 1807–1821.

Singh G, Tommasini R, Bhatia S and Mutharaju R (2025) Benchmarking neurosymbolic description logic reasoners: Existing challenges and a way forward. *Neurosymbolic Artificial*

Intelligence 1: 29498732251339943. DOI:10.1177/29498732251339943.

Terziyan VY and Vitko O (2023) Taxonomy-informed neural networks for smart manufacturing. In: *ISM*, *Procedia Computer Science*, volume 232. Elsevier, pp. 1388–1399.

Tian Y, Willis KDD, Omari BA, Luo J, Ma P, Li Y, Javid F, Gu E, Jacob J, Sueda S, Li H, Chitta S and Matusik W (2024) ASAP: automated sequence planning for complex robotic assembly with physical feasibility. In: *ICRA*. IEEE, pp. 4380–4386.

Wang P, Donti PL, Wilder B and Kolter JZ (2019) Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In: *ICML*, *Proceedings of Machine Learning Research*, volume 97. PMLR, pp. 6545–6554.

Wang R and Cheung CF (2023) Knowledge graph embedding learning system for defect diagnosis in additive manufacturing. *Comput. Ind.* 149: 103912.

Wang S, Yang J, Yang B, Li D and Kang L (2024) An intelligent quality control method for manufacturing processes based on a human–cyber–physical knowledge graph. *Engineering* 41: 242–260.

Wang Z, Melkote S and Rosen DW (2023) Generative design by embedding topology optimization into conditional generative adversarial network. *Journal of Mechanical Design* 145: 111702.

Wang Z, Yan X, Melkote SN and Rosen D (2025) Mcgan: Generating manufacturable designs by embedding manufacturing rules into conditional generative adversarial network. *Adv. Eng. Informatics* 64: 103074.

Willis KDD, Pu Y, Luo J, Chu H, Du T, Lambourne JG, Solar-Lezama A and Matusik W (2021) Fusion 360 gallery: a dataset and environment for programmatic CAD construction from human design sequences. *ACM Trans. Graph.* 40(4): 54:1–54:24.

Wu W, Wang Z, Ye J, Zhou Z, Li Y and Guo L (2025) Nesygeo: A neuro-symbolic framework for multimodal geometric reasoning data generation. *CoRR* abs/2505.17121.

Xu Q, Gao P, Wang J, Zhang J, Ip AWH and Zhang CWJ (2024) AKGNN-PC: an assembly knowledge graph neural network model with predictive value calibration module for refrigeration compressor performance prediction with assembly error propagation and data imbalance scenarios. *Adv. Eng. Informatics* 60: 102403.

Yan K, Liu Y, Lin Y and Ji S (2022) Periodic graph transformers for crystal material property prediction. In: *NeurIPS*.

Yang C, Zheng Y, Tu X, Ala-Laurinaho R, Autiosalo J, Seppänen O and Tammi K (2023) Ontology-based knowledge representation of industrial production workflow. *Adv. Eng. Informatics* 58: 102185.

Yu D, Yang B, Liu D, Wang H and Pan S (2023) A survey on neural-symbolic learning systems. *Neural Networks* 166: 105–126.

Yuan Z, Shi J and Huang Y (2024) Openecad: An efficient visual language model for editable 3d-cad design. *Comput. Graph.* 124: 104048.

Zhang C, Polette A, Pinquié R, Carasi G, Charnace HD and Pernot J (2025) ecad-net: Editable parametric CAD models reconstruction from dumb b-rep models using deep neural networks. *Comput. Aided Des.* 178: 103806.

Zhang Q and Li H (2007) MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* 11(6): 712–731.

Zhang Q, Luo Q, Zhao A, Yu C, Wang Q and Ke Y (2024) Onto-sagcn: Ontology modeling and spatial attention-based graph convolution networks for aircraft assembly quality prediction. *Adv. Eng. Informatics* 60: 102531.

Zitzler E, Brockhoff D and Thiele L (2007) The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In: *EMO*, *Lecture Notes in Computer Science*, volume 4403. Springer, pp. 862–876.