AgenTRIZ: a Multi-Agent LLM System for Automating the TRIZ Methodology

Jonas Wenk[1], Jürgen Moosburger[2], Timo Baumann[1]

[1] OTH Regensburg, Seybothstraße 2, 93053 Regensburg, Germany
[2] ams-OSRAM International GmbH, Leibnizstraße 4, 93055 Regensburg, Germany

Abstract. TRIZ is a systematic innovation methodology that helps engineering teams resolve design conflicts: situations in which improving one property of a system unavoidably degrades another. TRIZ provides a structural sequence of analytical tools – system decomposition, functional analysis, root cause analysis, contradiction formalization, and matrix-guided solution retrieval – that have been validated across millions of patents. Reliable execution of TRIZ, however, requires rare specialist training that most engineering teams do not possess. We present the design, implementation, and evaluation of an AI system that guides engineers through the complete TRIZ process. Five specialized LLM agents, each responsible for one analytical stage, are coordinated by a central orchestrator that enforces the correct step ordering and prevents skipping ahead. The Contradiction Matrix lookup is performed by a deterministic programmatic tool rather than by the language model's memory, eliminating a known source of result hallucination. A document retrieval pipeline grounds solution synthesis in the engineer's own project files, and a persistent database with explicit confirmation gates ensures that no analytical result is stored without human approval. We evaluate the system in comparison to the work of a certified Level 3 TRIZ practitioner, in interaction with engineers in two case studies, in comparison with a baseline system. Our results both on the system level as well as component-wise indicate that a structured multi-agent system can serve as an effective engineering problem-solving assistant, producing methodologically sound analysis while keeping final judgement and decision-making with the human engineer under a mixed-initiative paradigm.

Keywords: TRIZ, Agentic AI, Large Language Models (LLMs), Case Study

1 Introduction

Engineering progresses by resolving contradictions. A semiconductor package must dissipate heat efficiently while remaining as thin as possible. A mixing device must generate sufficient shear force without overheating the mixture.

Altshuller studied the resolution of contradictions across approx. 200,000 patents and found that roughly 95% of genuine engineering contradictions had already been resolved elsewhere – often in an unrelated technical domain – and that their solutions consistently followed a small number of recurring structural patterns [1]. Mann

subsequently extended the corpus to over two million patents, producing the Matrix 2003 used in this work [2]. TRIZ (Teoriya Resheniya Izobretatelskikh Zadach) collects these patterns into a structured methodology. A domain-specific problem is abstracted into a generic contradiction formulation, resolution strategies are retrieved from the patent corpus, and the resulting abstract solution is translated back into the specific context. This cycle is known as the Abstraction Prism [3] as illustrated in Figure 1.

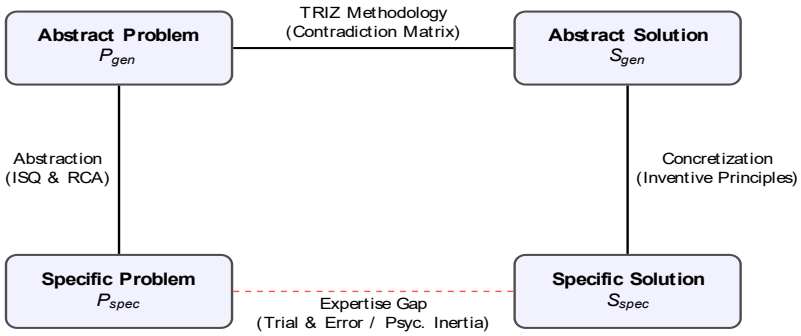

Fig. 1: The TRIZ Abstraction Prism ($P_{spec} \rightarrow P_{gen} \rightarrow S_{gen} \rightarrow S_{spec}$).

Both translation boundaries of the Prism require specialist training that most engineering teams cannot justify [3, 5]: the upward step demands familiarity with the TRIZ parameter system, the downward step demands cross-disciplinary design knowledge. These barriers constitute the Expertise Gap. Large language models (LLMs) are structurally well suited to close it: cross-domain semantic mapping and broad associative reasoning are core LLM strengths, whereas the engineer retains the complementary ability to verify physical feasibility [3, 4].

Prior computational TRIZ support has evolved through three generations. Classical CAI systems digitised the Matrix but left contradiction formulation to the engineer [6]. Semantic TRIZ (S-TRIZ) automated SAO extraction via NLP but could not generate new solutions [5]. Recent LLM-based systems (AutoTRIZ [7], TRIZ-GPT [8]) partially close the upward translation gap but remain stateless single-pass pipelines that hallucinate Matrix lookups [7, 9] and lack document grounding. TRIZAgents [10] pursues multi-agent decomposition but without prerequisite enforcement or persistent state. To the best of our knowledge, no prior system addresses all four capabilities simultaneously. Table 1 summarises coverage.

Table 1. Capability coverage of each computational TRIZ generation. ✓ = addressed; ≈ = partially addressed; × = not addressed.

Approach	ISQ / Formulation	Matrix Lookup	Solution Synthesis	Persistent State
Classical CAI	×	✓	×	×
Semantic TRIZ	≈	✓	×	×
Linear LLM	≈	≈	≈	×
Proposed system	✓	✓	✓	✓

We present AgenTRIZ, a multi-agent LLM system that automates the TRIZ Abstraction Prism end-to-end. The system makes four contributions:

1. An agent topology that enforces the hard dependency ordering of the TRIZ analytical sequence through structural graph constraints rather than prompt-level instructions that could be ignored,
2. A deterministic, hallucination-free Contradiction Matrix lookup that separates (tool-based) parameter identification from (generative) solution synthesis,
3. Solution grounding in the engineer's own project documentation through hybrid retrieval that combines semantic similarities and keyword search,
4. A persistent, human-validated analytical record with explicit confirmation gates at stage boundaries to ensure that only reviewed artefacts propagate to downstream analysis.

We evaluate AgenTRIZ in multiple ways: a stage-by-stage comparison against a certified Level 3 TRIZ practitioner, a controlled comparison against a monolithic single-agent baseline, two industrial case studies with practitioner feedback, and component-level benchmarks on synthetic test scenarios.

2 TRIZ Background

AgenTRIZ implements the standard TRIZ analytical sequence (see [1, 3, 4] for comprehensive treatments). Figure 2 shows the stages: the Innovation Situation Questionnaire (ISQ) captures the system's function, components, and constraints in domain-agnostic terms [4]. Functional Analysis represents every component interaction as a Subject-Action-Object (SAO) triple classified by type and fulfilment [1, 5]. Harmful interactions feed into Root Cause Analysis (RCA), here implemented as a linear 5-Whys trace; in established TRIZ practice, this step is typically performed as a Cause-Effect Chain Analysis (CECA), which supports parallel causal branches and AND/OR nodes [3, 4] – extending the implementation to full CECA is left to future work. The resulting root causes are formalised as contradictions using the IF-THEN-BUT structure [4], mapped onto standardised Engineering Parameters, and resolved via the Contradiction Matrix [1, 2]. Both the upward translation (domain-specific failure to abstract parameters) and the downward translation (abstract Inventive Principles to concrete design) require specialist training that most engineering teams do not possess [3, 5] – this Expertise Gap is the specific problem AgenTRIZ is designed to close.

Design and Implementation of AgenTRIZ

Our architecture follows one principle: wherever an unconstrained LLM could introduce error – by hallucinating matrix entries, skipping analytical stages, or committing results without review – a structural constraint replaces the generative path.

2.1 System Overview and Agent Topology

AgenTRIZ consists of five specialist LLM agents and one orchestrator meta-agent, compiled into a Directed Cyclic Graph (DCG) using the LangGraph framework [11]. Each specialist agent is bound to exactly one TRIZ analytical stage and receives its fixed toolset at graph compilation time. Table 2 shows the mapping.

Table 2. Mapping of specialist agent nodes to TRIZ methodology stages.

Agent node	Prism stage	TRIZ stage	LLM tier
functional_node	Pspec→Pgen	Functional Analysis	Fast (gpt-5-mini)
rca_node	Pspec→Pgen	Root Cause Analysis	Reasoning (o3)
contradiction_node	Pgen	TC/PC formulation	Reasoning (o3)
solver_node	Pgen→Sgen	Matrix / Solution synthesis	Reasoning (o3)
gather_node	Pspec	ISQ Formulation	Fast (gpt-5-mini)

The cyclic graph topology matches the iterative nature of TRIZ analysis: a finding during RCA may require revising the Functional Model, and a poorly formulated contradiction must be corrected before solution synthesis can proceed. Three conditional edges decide at runtime whether execution advances to the next stage or loops back to an earlier one. The decision to split work across single-purpose agents rather than running the full workflow in one prompt follows two observations from the literature: narrow role boundaries prevent agents from attempting tasks they are not equipped for, and the separation between analytical stages creates hard prerequisite gates that a single agent has no structural mechanism to enforce [12].

Each agent's analytical behaviour is governed by a Standardised Operating Procedure (SOP) encoded directly in its system prompt [13]. Rather than instructing the model to "analyse root causes," the RCA agent's SOP specifies an algorithmic loop: select one harmful function from the Functional Model, trace it through a 5-Whys chain, and terminate only when reaching a fundamental physical constraint or an immutable boundary condition. Figure 2 illustrates the execution cycle for a single user turn.

2.2 Orchestrator and Prerequisite Enforcement

The orchestrator's LLM is called through a schema-constrained interface that forces it to produce valid routing decisions and enforces gates at the graph level. Therefore, the model cannot name a node that does not exist or return prose instead of a routing decision based on malicious prompting. Routing follows a four-tier priority hierarchy applied on every user turn: (P1) an explicit user command overrides all passive state evaluation; (P2) an uncommitted draft in the volatile buffer is resolved before new work begins; (P3) the most recently active analytical strand continues; (P4) ambiguous intent routes to a general-purpose agent for clarification. Separately, hard prerequisite gates enforce the TRIZ dependency chain regardless of user request: the solver is inaccessible until at least one committed contradiction exists, and contradiction formulation is blocked until an RCA is on record.

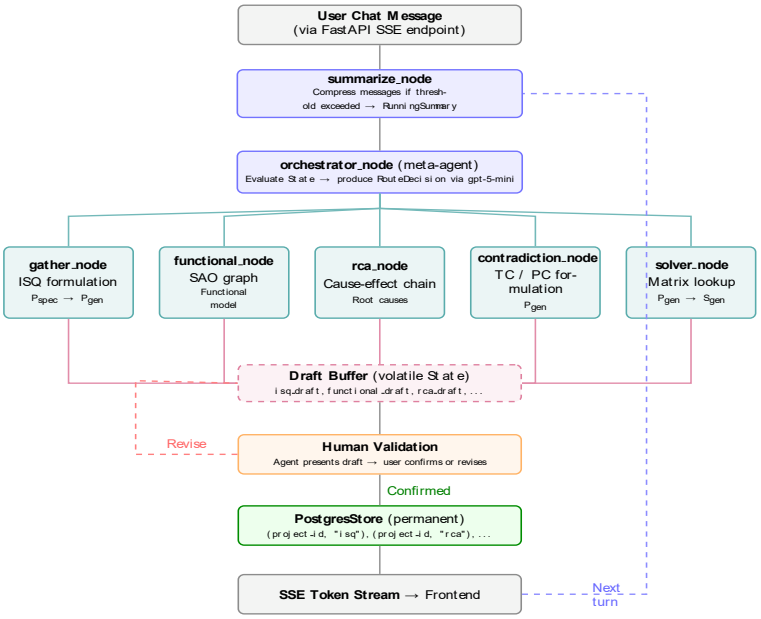

Figure 2: The AgenTRIZ execution cycle, illustrating the stateful transition from user input to human-validated committed artefacts as well as generated output.

2.3 Deterministic Matrix Lookup

The Contradiction Matrix is deterministic: a cell lookup, not a reasoning task. When the lookup is embedded in the generative process, LLMs can produce plausible-sounding but incorrect parameter identifiers and principle assignments [7, 9].

AgenTRIZ keeps the matrix from the generative path through a mandatory two-step sequence. The system exclusively uses the Matrix 2003 with its 48 engineering parameters [2], not Altshuller's original 39-parameter matrix. First, the solver agent calls a reference tool that returns the complete 48-parameter table from a CSV file. The agent matches the contradiction's parameter strings against this table by inspection – the SOP explicitly prohibits recalling parameter identifiers from training data. Second, once both integer identifiers are confirmed, the agent calls a lookup tool that performs a CSV row match against the Matrix 2003 dataset [2] and cross-references the returned principle numbers against a principle description file. No neural inference is involved in either step. If no matching cell exists, the tool returns an error rather than a fallback guess.

For Physical Contradictions, no lookup table exists. The solver assesses all four Separation Principles (Space, Time, Condition, System Level) against the conflicting requirements using the reasoning-tier model – extended deliberation is appropriate here because the task requires physical reasoning, not data retrieval.

2.4 State Management and Human-in-the-Loop Validation

TRIZ analysis can span multiple sessions, and each analytical stage depends on the verified outputs of preceding stages. AgenTRIZ addresses this through dual-tier persistence and a structural validation gate.

Thread checkpoints (written to PostgreSQL at every node transition) allow any in-progress session to resume after interruption. A separate semantic artefact store holds the committed, validated TRIZ outputs – ISQ fields, Functional Models, RCA chains, contradictions, and solutions – keyed by project and analytical stage [12, 14]. Agents at later stages load their predecessors' outputs from this store; they never reason from raw conversation history for structured analytical data.

The Draft-Save pattern enforces human control over all permanent state changes. When an agent completes an analytical step, its tools write exclusively to a volatile draft buffer in session state. A commit requires explicit human confirmation: only after the engineer reviews and approves the draft, does the path to write to persistent storage exist in the agent's toolset.

In addition to the structured artefact store, an asynchronous memory extraction process evaluates recent messages after each conversational turn and extracts implicit project facts – rejected design approaches, informal constraints, domain-specific vocabulary – into a long-term memory namespace. These entries are retrievable by all agents in subsequent sessions through a semantic search tool, providing cross-session continuity for information that does not belong in any formal ISQ field.

2.5 Document Grounding and Model Routing

Solution synthesis must be constrained by verified, project-specific engineering data. AgenTRIZ implements retrieval as an on-demand agent tool rather than a static context prefix: agents invoke it only when they identify a specific knowledge gap, keeping the context window compact and avoiding attention degradation on mid-context content [15, 16]. The retrieval pipeline runs semantic vector search and BM25 keyword matching in parallel, with results fused via Reciprocal Rank Fusion. Both channels are necessary: semantic search captures conceptual relationships during ISQ formulation (e.g., matching "rotary drive mechanism" to "motor and gearbox assembly"), while keyword search preserves exact technical values during contradiction formulation (e.g., "yield stress > 150 Pa").

Not every agent node requires the same model. Routing decisions, ISQ field extraction, and SAO classification are structured information tasks handled by a fast-tier model. RCA, contradiction formulation, and solution synthesis require extended chains of deductive reasoning and are handled by a reasoning-tier model with an internal chain-of-thought mechanism.

3 Worked Example: Resin Mixer

To illustrate the full pipeline, we traced a session on a laboratory resin mixing device, which was also analysed by a certified TRIZ practitioner (see also Sec. 5.1).

The gather agent captured the primary function ("mixes dental 3D-printing resin"), 10 components, and binding constraints (≤50W, ≤40°C, 10-min cycle, ≤20$ BOM). Functional Analysis identified 15 SAO interactions, 3 problematic. RCA traced three root causes: insufficient shaft power, friction-induced heating, and dead zones from bottle geometry. The contradiction agent formulated: IF increased mixing energy, THEN homogeneity within specification, BUT rising abrasive wear yielding micro-plastic debris. The improving parameter was mapped to Loss of Time (#26), the worsening to Harmful Emissions (#30). We note that this mapping is debatable: the improving property describes homogeneity rather than time, illustrating that LLM-driven parameter identification remains an approximation that benefits from human review at the confirmation gate. The matrix lookup returned Inventive Principle #1 (Segmentation), which was concretised into three solution concepts: sacrificial silicone O-ring sleeves, a dual-row roller design separating torque from radial load, and tile-surfaced rollers with replaceable grip elements.

All five analytical stages produced committed, typed artefacts. No TRIZ expertise was required of the user beyond understanding the engineering context of the device.

4 Evaluation

We compare the performance of AgenTRIZ with one expert, and test its utility and acceptability with engineers in 2 more case studies. For the 3 cases together, we evaluate the agentic approach against a monolithic baseline and on the component level. We note that the architecture of AgenTRIZ establishes structural correctness – the dependency chain is enforced, the matrix lookup is deterministic, artefacts cannot be committed without human confirmation. Here we test the analytical soundness.

4.1 Gold-Standard Expert Benchmark

A certified Level 3 TRIZ practitioner independently analysed the resin mixer problem described in Section 4 and documented the full Abstraction Prism in a written report [17]. The same problem description was then submitted to AgenTRIZ with no access to the expert's document. Outputs were compared stage by stage. We investigated the overlap of correct elements and the cases in which either system or expert only provided a valid analytical choice. Table 3 shows the aggregate results.

Table 3. Multi-agent system vs. Level 3 TRIZ practitioner on the resin mixer case study.

TRIZ Stage	Match	Novel-Valid	Expert-Only
Component Analysis	9/11	2	0
Functional Model	12/15	3	0
Root Cause Analysis	4/5	3	1
Engineering Contradictions	3/6	2	1
Physical Contradictions	2/5	2	2
Solution Concepts	4	2	3

Zero methodological errors of the system were found. Every contradiction produced by the system conforms to the required format. All matrix lookups are verifiable as they come from deterministic lookup rather than memory. The 3/6 match rate for Engineering Contradictions reflects the divergent nature of TRIZ: different analysts legitimately formulate different contradictions from the same system. The system's novel-valid contradictions address aspects the expert did not explore, while expert-only items, such as a gearbox contamination root cause and a haptic-feedback contradiction that fell outside the system's ISQ-defined scope, are coverage gaps rather than errors. Broadening the ISQ scope and supporting iterative re-formulation could improve coverage in future versions.

Four convergent solutions were found: sacrificial contact surfaces, multi-roller torque transmission, vibration isolation, and internal fluid mixing. Two additional solution themes from the system, magnetic torque coupling and a split-mass counterbalance, were also generated by the monolithic baseline described below, providing second-source validation without a second human expert.

4.2 Industrial Case Studies and Practitioner Feedback

Two engineering projects at an industrial partner (name withheld for confidentiality; more complex than the worked example) were processed through the full pipeline by domain engineers not involved in the system's development [20]. Three engineers completed a nine-dimension Likert evaluation (1–5 scale) in which they confirm perfect state retention, a usability of 4.7, and TRIZ quality of 4.3. The trust/reliance was scored at 3.3, which underlines the nature of assistance that must be questioned and validated.

4.3 Multi-Agent vs. Monolithic Baseline

We implemented a single-agent monolithic baseline system with just a single large prompt as a replacement for tool usage (memory and document retrieval). Both systems' outputs across the 3 cases were scored by an LLM judge (not involved in either system's generation) on five dimensions (completeness, TRIZ adherence, technical accuracy, context grounding, solution quality; each 1–5) and stage-averaged; see Tab. 4.

Table 4. LLM-judge scores averaged by TRIZ stage across 3 cases for agentic vs. baseline.

Stage	Multi-Agent	Baseline
ISQ	3.67	3.47
Functional Model	4.07	3.27
RCA	4.27	3.13
Contradictions	4.40	3.13
Solutions	3.93	3.47

A one-sided Wilcoxon signed-rank test on the 15 paired stage scores (3 projects × 5 stages) yields $W = 105$, $p = 0.0005$, with a large effect ($r = 0.88$), confirming a consistent advantage. The largest advantages appear at the RCA and contradiction stages – precisely the stages where the architecture enforces structural guarantees.

4.4 Component Benchmarks

Orchestrator routing accuracy was 87.5% across 200 test cases, with four of six routes achieving 96.9% or higher. Solver routing, however, reached only 35.7% in isolated testing. Analysis of the misrouted turns showed a consistent pattern: the orchestrator selected the contradiction agent when the solver agent was intended. Because both agents operate on contradiction data – one formulates contradictions, the other resolves them – the orchestrator's system prompt did not delineate their responsibilities with sufficient precision, causing systematic confusion at the routing boundary. The prerequisite gate mitigates this in production: premature solver invocation is structurally prevented regardless of the routing decision. Two remediation paths are under investigation: refining the orchestrator prompt to sharpen the decision boundary between formulation and resolution, or merging the two stages so that the solver becomes a tool callable by the contradiction agent, eliminating the ambiguous routing decision entirely. RAG retrieval fidelity was 100% across 60 test scenarios. Schema enforcement prevented invalid artefacts from reaching the permanent store across 50 validation scenarios. Short-term summarisation retained 70% of constraints.

5 Conclusion and Future Work

We have presented AgenTRIZ, an agentic AI system for TRIZ inventive problem solving. The system is rated favourably in case studies, and significantly outperfoms a monolithic LLM-based system. Several limitations must be stated. The evaluation scale – 3 case studies, one expert benchmark, 3 evaluating engineers – does not yet establish general performance bounds. LLM-as-a-judge only yields supportive results.

6 References

1. G. Altshuller, The Innovation Algorithm: TRIZ, Systematic Innovation and Technical Creativity. Technical Innovation Center, Worcester, MA, 1999.
2. D. Mann, Hands-On Systematic Innovation. CREAX Press, Ieper, Belgium, 2003.
3. I.M. Ilevbare, D. Probert, R. Phaal, "A review of TRIZ, and its benefits and challenges in practice," Technovation, vol. 33, no. 2, pp. 30–37, 2013.
4. I. Ekmekci, E.E. Nebati, "TRIZ Methodology and Applications," Procedia Computer Science, vol. 158, pp. 303–315, 2019.
5. M. Ghane, M.C. Ang, D. Cavallucci, R.A. Kadir, K.W. Ng, S. Sorooshian, "Semantic TRIZ feasibility in technology development, innovation, and production: A systematic review," Heliyon, vol. 10, no. 1, e23775, 2024.
6. M.G. Moehrle, "How combinations of TRIZ tools are used in companies – results of a cluster analysis," R&D Management, vol. 35, no. 3, pp. 285–296, 2005.
7. S. Jiang, W. Li, Y. Qian, Y. Zhang, J. Luo, "AutoTRIZ: Automating engineering innovation with TRIZ and large language models," Advanced Engineering Informatics, vol. 65, 103312, 2025.
8. L. Chen, Y. Song, S. Ding, L. Sun, P. Childs, H. Zuo, "TRIZ-GPT: An LLM-Augmented Method For Problem-Solving," in ASME 2024 IDETC/CIE, 2024.

9. S. Brad, E. Brad, A. Cirlejan, “Enhancing TRIZ Contradiction Resolution with AI-Driven Contradiction Navigator (AICON),” in World Conference of AI-Powered Innovation and Inventive Design, LNCS vol. 735, pp. 88–105, Springer, 2025.
10. K. Szczepanik, J. Chudziak, “TRIZ Agents: A Multi-Agent LLM Approach for TRIZ-Based Innovation,” in Proc. 17th Int. Conf. on Agents and Artificial Intelligence, pp. 196–207, 2025.
11. Z. Duan, J. Wang, “Exploration of LLM Multi-Agent Application Implementation Based on LangGraph+CrewAI,” arXiv:2411.18241, 2024.
12. T. Masterman, S. Besen, M. Sawtell, A. Chao, “The Landscape of Emerging AI Agent Architectures for Reasoning, Planning, and Tool Calling: A Survey,” arXiv:2404.11584, 2024.
13. S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, C. Zhang, J. Wang, Z. Wang, S.K.S. Yau, Z. Lin, L. Zhou, C. Ran, L. Xiao, C. Wu, J. Schmidhuber, “MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework,” arXiv:2308.00352, 2024.
14. T. Guo, X. Chen, Y. Wang, R. Chang, S. Pei, N.V. Chawla, O. Wiest, X. Zhang, “Large Language Model based Multi-Agents: A Survey of Progress and Challenges,” arXiv:2402.01680, 2024.
15. N.F. Liu, K. Lin, D. Chen, N. Kandpal, R. Tamkin, O. Levy, “Lost in the middle: How language models use long contexts,” Trans. Assoc. Comput. Linguistics, vol. 12, pp. 157–173, 2024.
16. Z. Xi et al., “The rise and potential of large language model based agents: A survey,” Science China Information Sciences, vol. 68, no. 2, 121101, 2025.
17. J. Träger, “Mischgeräte für 3D-Druck-Harze: TRIZ Projektarbeit,” TRIZ Level 3 course project, internal document, 2022.
18. N. Phadnis, M. Torkkeli, “Evaluating the Effectiveness of Generative AI in TRIZ: A Comparative Case Study,” in World Conference of AI-Powered Innovation and Inventive Design, LNCS vol. 735, pp. 175–192, Springer, 2025.
19. V. Čok, L. Samsa, M. Brojan, J. Tavčar, N. Vukašinović, “Case study: Is there a space for TRIZ in the era of ChatGPT?” Proceedings of the Design Society, vol. 5, pp. 871–880, 2025.
20. K.J. Kanarik, W.T. Osowiecki, Y. Lu, D. Talukder, N. Roschewsky, S.N. Park, M. Kamon, D.M. Fried, R.A. Gottscho, “Human-machine collaboration for improving semiconductor process development,” Nature, vol. 616, pp. 707–711, 2023.