Task Planning and Execution for Industrial Automation: A Comprehensive Analysis of Traditional and Emerging Methods

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 277 (2026) 3257–3266

Procedia Computer Science

www.elsevier.com/locate/procedia

7th International Conference on Industry of the Future and Smart Manufacturing
(former International Conference on Industry 4.0 and Smart Manufacturing)

Task Planning and Execution for Industrial Automation: A Comprehensive Analysis of Traditional and Emerging Methods

Dario Antonelli[a]*, Primož Podržaj[b], Antonio Maffei[c]

[a]*Department of Management and Production Engineering, Polytechnic University of Turin, Corso Duca degli Abruzzi 24, 10138 Torino, Italy*
[b]*Faculty of Mechanical Engineering, University of Ljubljana, Aškerčeva 6, 1000 Ljubljana, Slovenia*
[c]*Department of Production Engineering, KTH Royal Institute of Technology, Brinellv¨agen 68, 114 28 Stockholm, Sweden*

Abstract

This study provides a comprehensive analysis of task planning paradigms in industrial automation, evaluating different active and reactive methods like: Finite State Machines (FSMs), Behavior Trees, Hierarchical Task Networks, Symbolic Planning (PDDL), and Reinforcement Learning. It assesses the trade-offs of each method concerning scalability, reactivity, and verifiability. The core focus is the transformative role of generative AI, not as a standalone solution, but as an enabling technology that augments existing frameworks. Key generative AI applications include generating plans from natural language and creating synthetic training environments. The conclusive statement of the study is that the future of automation lies in hybrid deliberative-reactive architectures, such as combining PDDL and BTs, where AI enhances human expertise under rigorous verification.

Keywords: Task Planning; Generative AI; Industrial Automation; Hybrid Deliberative-Reactive Architecture

1. Introduction

In the dominion of industrial automation, the effectiveness of an automatic system is fundamentally tied to its ability to perform complex operations with efficiency and autonomy. Central to this capability is the discipline of

* Corresponding author. Tel.: +39-011-090-7288; fax: +39-011-090-7299.
E-mail address: dario.antonelli@polito.it

task planning, dedicated to creating the precise sequence of actions the system must follow to accomplish a given objective. This process involves the strategic decomposition of a high-level goal, such as "assemble the product," into a series of simpler, executable sub-tasks like "retrieve part A" and "install component B". The successful planning and execution of these sub-tasks have a profound impact on industrial outcomes, directly influencing productivity, manufacturing flexibility, and the overall level of autonomy in every industrial sector. For example, in manufacturing, well-optimized task plans can significantly reduce cycle times and enhance product quality, while in logistics, they enable the efficient management of warehouse inventories and order fulfillment processes [1].

The evolution of task planning has been driven by a core, persistent challenge: the progressive abstraction of human instruction. The ultimate ambition of the field is to transition from a paradigm requiring human operators to meticulously specify every low-level operation to one where they can simply communicate a high-level objective in natural language [2]. This pursuit has led to the development of a wide spectrum of planning methodologies, each offering a different level of abstraction and a unique balance of predictability, flexibility, and computational complexity. This report provides an analysis of these methods, covering both the foundational techniques that continue to be mainstays in industrial control and the emerging paradigms that are shaping the future of intelligent automation.

This paper is structured as follows. Section 2 describes the methodology used for this literature review. Sections 3 and 4 analyze reactive and deliberative task planning strategies, respectively, evaluating the core strengths and weaknesses of each paradigm. Within each section, we discuss the transformative role of generative AI and provide concrete industrial use cases. Section 5 presents a comparative synthesis of all paradigms, discusses the promise of hybrid architectures, and reflects on the key challenges and future directions for the field.

2. Review Methodology

This study employs a structured narrative literature review methodology to synthesize and analyze the state-of-the-art in task planning for industrial automation. The process was conducted in three stages:

1. **Foundational Scoping:** The initial phase involved identifying the seminal and most-cited works for each of the five core paradigms (FSMs, BTs, HTNs, PDDL, RL). This was done to establish a baseline understanding of each method's theoretical underpinnings and historical application in industrial contexts.
2. **Targeted Literature Search:** A systematic search was performed using academic databases including Scopus, IEEE Xplore, Google Scholar, and arXiv. The search was focused on publications from the last five years to capture the latest advancements, particularly the integration of generative AI. Keywords used included combinations of terms such as: ("*task planning*" OR "mission planning") AND ("*industrial automation*" OR "*robotics*" OR "*manufacturing*") AND ("*Generative AI*" OR "*LLM*" OR "*Large Language Model*") AND ("*Finite State Machine*" OR "*Behavior Tree*" OR "*HTN*" OR "*PDDL*" OR "*Reinforcement Learning*").
3. **Inclusion and Exclusion Criteria:** We included papers that offered comparative analyses, presented novel applications of generative AI to a planning framework, or discussed hybrid architectures. We excluded papers focused purely on low-level motion planning or control theory without a task-level component, as well as purely theoretical AI papers lacking a clear application pathway to industrial automation.

This multi-step approach ensures that the review is grounded in the foundational principles of the field while focusing on the most current and impactful research trends.

3. The Hierarchy of Planning

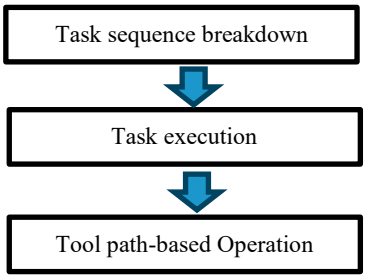

Fig. 1. Hierarchical structure of an industrial job, adapted from [3].

To fully grasp these different methodologies, it is essential to understand the hierarchical nature of planning, as illustrated in Fig.1 [3]. As depicted in the figure, a high-level industrial job is first deconstructed into a *Task sequence breakdown*. Each logical task within this sequence is then handled by the *Task execution layer*, which finally commands the low-level controllers responsible for *Tool path-based Operation*.

A complete automation solution is typically structured in several layers of decision-making, each operating at different levels of abstraction that are corresponding to the job hierarchy. The highest level is mission planning, which often involves coordinating entire fleets of robots, CNC machines or AGVs. In this layer, a central logistics system or fleet manager assigns broad goals or "missions" to the fleet, breaking down a large-scale objective into individual assignments for each individual agent.

Beneath mission planning lies task planning, the principal focus of this analysis. Once an automatic device receives its mission, the task planner takes over. Its responsibility is to convert this high-level goal, in conjunction with its internal world model and predefined skills, into a specific, ordered sequence of actions. This is the logical layer that determines what the automatic system should do and in what order.

Finally, at the lowest level of the hierarchy is tool path planning. This layer takes a single action from the task plan and computes a precise trajectory to execute that action in the real world. The assumption of this study is that modern AI evolved to a level where it can effectively produce and manage the task planning level, forming the critical bridge between strategic mission objectives and their low-level physical execution.

The historical development of task planning methodologies reveals a necessary and clear progression from rigid, deterministic systems to flexible, adaptive frameworks. Early industrial automation was concentrated in highly structured and predictable environments, like assembly lines, where tasks were repetitive and the world was static. In such contexts, simple, robust, and verifiable control systems like Finite State Machines (FSMs) were not only adequate but ideal due to their inherent predictability [4].

However, the demands of modern industry have shifted dramatically, introducing unprecedented levels of dynamism and uncertainty [5]. Flexible manufacturing, mass customization, and human-robot collaboration require automatic machines to work alongside people, handle diverse products, and operate in semi-structured environments. This new reality necessitates planning systems that are not merely pre-programmed but are reactive, modular, and capable of adapting to unforeseen events and changing conditions. This need has fueled the development of more sophisticated paradigms like Behavior Trees (BTs), Hierarchical Task Networks (HTNs), and Reinforcement Learning (RL), which prioritize modularity, reactivity, and learning capabilities over rigid optimization of the process. This evolution also signifies a fundamental shift in where the "intelligence" of the system resides. Initially, the primary focus was on perfecting low-level motion control. FSMs encoded logic at the level of machine states. Behavior Trees abstracted this further into composable behaviors. HTNs and Symbolic Planning with the Planning Domain Definition Language (PDDL) elevated the reasoning process to a symbolic, goal-oriented level, where the system reasons about what to achieve rather than what state to be in [6]. This upward migration of intelligence facilitates more natural human-machine interaction and grants the system greater autonomy. The recent emergence of generative AI is now poised to accelerate this trend, providing a powerful new toolset to automate, augment, and democratize the creation and execution of these advanced task plans. The following sections of this report will

analyze each of these major paradigms, with a particular focus on the transformative, and sometimes perilous, role of generative AI.

4. Reactive Strategies

4.1. The Enduring Role of Finite State Machines

A Finite State Machine (FSM) is a foundational model of computation that has long served as a bedrock for industrial automation. An FSM is defined by a finite set of states, an initial state, and a set of inputs that trigger transitions between these states. A core principle is that the system can only be in one state at any given moment. This model excels at representing sequential logic and is a time-tested tool for programming Programmable Logic Controllers (PLCs), the rugged computers controlling machinery on the factory floor [7]. Standardization efforts like PackML, part of the ISA88 standard, have codified the FSM approach by defining a standard set of machine states (e.g., Idle, Execute, Aborted) and their valid transitions.

The primary strength of FSMs, and the reason for their prevalence in safety-critical applications, is their predictability and reliability. With a finite, well-defined set of states and transitions, an FSM's behavior is deterministic and easily understood, which is crucial for verifiably correct control flow. This explicit nature provides a strong basis for safety, as interlocks can be programmed directly into the transition logic, preventing hazardous operations unless all safety conditions are met. For problems with a manageable number of states, FSMs are simple to implement, debug, and maintain.

Industrial Use Case: An essential example is a packaging line governed by the PackML (ISA-88) standard. The entire line operates as a large FSM, transitioning through standardized states like Idle, Starting, Execute, Held, and Aborted. This ensures that all machines on the line (e.g., fillers, cappers, labelers) are synchronized and that safety interlocks prevent a machine from starting its operation until the upstream and downstream machines are in the correct state.

However, FSMs have inherent limitations that make them unsuitable for highly complex or dynamic systems. Their most significant drawback is "state explosion," where the number of states and transitions grows exponentially as system complexity increases, leading to unmanageable "spaghetti code". FSMs are also monolithic and rigid; adding a new state can require rewiring numerous existing transitions, which hinders modularity and reusability [8].

Generative AI, particularly Large Language Models (LLMs), offers new ways to address these limitations. AI can automate the generation of FSM code from a high-level natural language description, drastically reducing manual coding effort and lowering the complexity barrier for moderately complex systems. However, this introduces significant risks. The greatest danger is "logical hallucination," where a model produces syntactically correct but logically flawed or unsafe FSMs. Deploying AI-generated logic in an industrial setting is unsustainable without a new layer of rigorous formal verification to prove its correctness [9].

Ultimately, a fundamental tension exists: the core value of an FSM is its verifiable reliability, while generative AI is inherently probabilistic. Therefore, the most responsible role for AI is not as an autonomous generator but as a "developer's assistant". It can accelerate the initial drafting of FSMs from a human-authored specification, but the final logic must undergo intense scrutiny, testing, and validation by a human expert.

4.2. The Rise of Modular and Reactive Control: Behavior Trees (BTs)

As industrial automation moves towards more complex and dynamic tasks, the limitations of monolithic control systems like FSMs have led to the rise of modular and reactive frameworks, most notably Behavior Trees (BTs). A BT is a directed tree structure that governs a robot's decision-making policy. Its execution is driven by a periodic "tick" that originates at the root and propagates downwards, orchestrating the robot's actions.

The tree consists of two main node types: *Execution Nodes* (the leaves) and *Control Flow Nodes* (the internal nodes). Execution nodes directly interact with the world, performing actions like moving or checking conditions (e.g., "is the gripper empty?"). They return a status of `Success`, `Failure`, or `Running`. The `Running` status is a crucial feature that enables reactivity, allowing a long-running action to be preempted by a higher-priority task, such

as an emergency stop. Control flow nodes like `Sequence` (an AND gate) and `Fallback` (an OR gate) use the status returned by their children to define the tree's logic and orchestrate the overall behavior [10].

BTs offer significant advantages over FSMs in dynamic environments [8]. They are inherently modular, allowing subtrees representing specific behaviors (e.g., "pick up object") to be developed, tested, and reused independently. This modularity enables BTs to scale gracefully to complex tasks without the "state explosion" problem that plagues FSMs. Their graphical, hierarchical structure makes them more readable and intuitive, and the `Fallback` node provides a clean, built-in mechanism for implementing failure recovery logic. The widespread adoption of BTs is also driven by their tight integration into the Robot Operating System (ROS) through powerful libraries like `BehaviorTree.CPP` and `py_trees`.

Industrial Use Case: BTs are widely used in warehouse logistics for autonomous mobile robots (AMRs). A typical task like "fetch item A from shelf 12" would be a BT. The root node might be a `Fallback`. The first child could be a `Sequence` for the ideal plan: `MapsTo(Shelf12), Identify(ItemA), Pick(ItemA), MapsTo(PackingStation)`. The next child in the `Fallback` could be a recovery behavior, like `ReportObstructionAndReroute`, which only runs if the initial navigation fails. This modular structure makes it easy to add new capabilities or error-handling routines.

Generative AI is emerging as a powerful tool to automate BT creation. LLMs can translate high-level natural language commands directly into a BT's logical structure, often in the standard XML format. More advanced frameworks like BETR-XP-LLM can even use an LLM to dynamically diagnose a failure during execution and automatically generate a new subtree to handle the unforeseen error, enabling the system to learn from its mistakes [11]. However, this approach carries risks. An LLM can generate a syntactically valid tree that is semantically or logically flawed, leading to unsafe behavior. It might also "hallucinate" robot capabilities that don't exist, requiring a robust validation layer to check the generated tree against the robot's known skills.

The most robust application of this technology is a hierarchical control architecture: an LLM acts as a strategic planner, translating human intent into a structured BT; the BT serves as a verifiable tactical sequencer; and the robot's pre-programmed skills form the reliable execution layer. This leverages the BT as a safe, reactive, and verifiable intermediate representation, positioning the LLM as a high-level "what-to-do" suggester, while the BT remains the deterministic "how-to-do-it" engine, ensuring predictable execution.

5. Deliberative strategies

5.1. Leveraging Domain Knowledge: Hierarchical Task Networks (HTNs)

While Behavior Trees (BTs) offer modularity, Hierarchical Task Network (HTN) planning introduces a powerful way to embed expert domain knowledge directly into the planning process. An HTN planner solves problems using a top-down, hierarchical approach. Instead of searching through a vast space of primitive actions, it begins with high-level, abstract tasks (e.g., "build a house") and systematically decomposes them into simpler sub-tasks until only primitive, executable actions remain (e.g., "move to location X").

The core of an HTN's domain knowledge is its library of *Methods*. A method is a pre-defined recipe that specifies how to break down an abstract task into a network of sub-tasks. A single task can have multiple methods, each representing a different approach for different situations. Primitive actions, known as *Operators*, are defined by their preconditions and effects, much like in classical planning. This process changes planning from a search for a goal state to a search for a valid task decomposition.

The primary advantage of this approach is a massive reduction in computational complexity. By providing the planner with a "recipe book" of methods, HTNs guide the search and prevent it from exploring nonsensical action sequences, thus avoiding the "combinatorial explosion" that plagues many other planners. This hierarchical decomposition also mirrors how humans solve complex problems, making the plans and domain knowledge more intuitive for experts to create, understand, and debug. This structure is also highly expressive, reusable, and has been proven to scale in industrial applications. Due to their hierarchical nature, HTNs are well-suited for multi-machine systems and human-robot collaboration, where a shared understanding of the task structure is critical for coordination and trust [12].

Industrial Use Case: HTNs are well-suited for complex, multi-step collaborative assembly tasks. Consider the assembly of an aircraft wing section. The high-level task "AssembleWingbox" can be decomposed using different methods depending on available resources. One method might involve a large robot for heavy lifting and positioning of spars, followed by a human operator for fastening and inspection. An alternative method could be used if the main robot is unavailable, decomposing the task into steps for smaller, coordinated robots. The HTN encodes this expert process knowledge, guiding the planner to valid and efficient assembly sequences.

Despite its power, HTN planning suffers from a significant knowledge acquisition bottleneck; the manual, time-consuming process of an expert creating the domain model is a major challenge. Generative AI offers a promising solution to this problem. An LLM could potentially automate the creation of HTN methods by analyzing expert demonstrations, technical manuals, or even videos of tasks being performed. It could also help translate a user's natural language request (e.g., "Assemble the gearbox") into a formal task for the planner [13].

However, this application has significant risks as a generative model may fail to capture the nuanced, context-dependent knowledge related to safety and efficiency that is critical in industrial settings. It could produce flawed or inefficient hierarchies that degrade performance, and verifying the correctness of an AI-generated domain is still a challenging activity [14].

Therefore, the most viable role for generative AI is not as a replacement planner but as a knowledge engineering tool. Some scholars [15] propose a workflow that involves an LLM analyzing unstructured human knowledge to generate a draft of the HTN domain model. This draft must then be presented to a human expert for critical verification, refinement, and validation, leveraging the AI's ability to process vast information and the human's essential expertise and oversight.

5.2. Formal Logic and Provable Correctness: Symbolic Planning with PDDL

Symbolic planning, with the Planning Domain Definition Language (PDDL) as its standard, represents the peak of formal, logic-based task planning. This approach requires meticulously defining a problem using two files: a Domain File that describes the "physics" of the world—including object types, world states (predicates), and available actions with their preconditions and effects—and a Problem File that defines a specific scenario with its initial state and goal conditions. A symbolic planner then searches for this formal model to find a valid sequence of actions to achieve the goal.

This method embodies a fundamental trade-off. Its advantage is its expressiveness; modern PDDL can create high-fidelity models of complex industrial problems, and the resulting plans are inherently verifiable and explainable. The significant disadvantage, however, is computational complexity. The search for a plan is NP-hard, and as the number of objects and actions grows, the state space explodes exponentially. This "combinatorial explosion" can make planning prohibitively slow, a major limitation in environments that require rapid re-planning.

Despite this, symbolic planning is a cornerstone of modern robotics, particularly in Task and Motion Planning (TAMP). TAMP frameworks use a hybrid approach where a PDDL planner handles the high-level logical reasoning, generating a symbolic plan (e.g., `(pick blockA)`). This plan is then passed to a low-level motion planner that computes the precise, collision-free robot movements to execute each symbolic action [16].

Generative AI is not replacing these planners but is instead revolutionizing the difficult "front-end" task of creating the PDDL models. LLMs have shown a strong ability to translate natural language or even visual scenes into formal PDDL problem files, bridging the gap from perception to planning. They can also use commonsense knowledge to help diagnose plan failures and suggest repairs to the domain model [17].

However, using AI for this purpose has risks. LLMs can generate PDDL with syntactic or logical errors, and they struggle with the long-horizon reasoning needed for complex plans. The most promising architecture, therefore, uses generative AI as an intelligent interface that translates unstructured data like images and text into the formal PDDL representation. This PDDL is then fed to a traditional, reliable symbolic planner to perform the verifiable search for a solution, leveraging the strengths of both AI and formal methods.

5.3. Learning from Interaction: Reinforcement Learning (RL) in Task Planning

Reinforcement Learning (RL) presents a fundamentally different approach to task planning, shifting from explicit programming to learning through trial and error. In this paradigm, an agent, such as a robot, interacts with an environment and receives a numerical reward or penalty for each action it takes. The agent's goal is to learn a *policy* — a strategy mapping states to actions — that maximizes its total reward over time. A central challenge is balancing "exploitation" (using known successful actions) with "exploration" (trying new actions to find better rewards) [18].

While successful in simulations, RL's adoption in real-world industrial automation has been limited by significant challenges. It is notoriously sample-inefficient, often requiring millions of interactions to learn an effective policy—a process that is too slow and costly. More critically, the exploratory nature of learning means the machine will inevitably try random and potentially dangerous actions, which is unacceptable in a production environment where it could damage equipment or endanger workers. Furthermore, successfully training an RL agent is plagued by the difficulty of designing a good reward function and overcoming the "sim-to-real" gap, where policies trained in simulation fail when transferred to a physical robot.

Generative AI can address RL's most significant weaknesses. Its most transformative application is the creation of synthetic data and enriched simulation environments [19]. Instead of training in a single, handcrafted simulation, generative models can produce thousands of randomized variations of an environment. Training an RL agent in this diverse set of virtual worlds forces it to learn a more robust policy that is more likely to succeed in the real world. This provides a safe, cheap, and nearly infinite source of training data, directly tackling the sample efficiency and safety problems.

The most promising workflow combines these approaches: first, a limited amount of real-world data is collected to train a generative model. This model then creates a rich, randomized simulation environment where an RL policy is trained for millions of steps. Then, this robust policy is fine-tuned with a small amount of real-world data to close the final sim-to-real gap, making RL a more viable option for adaptive industrial tasks.

6. Synthesis, Comparative Analysis, and Hybrid Architectures

The five major paradigms for task planning —Finite State Machines, Behavior Trees, Hierarchical Task Networks, Symbolic Planning with PDDL, and Reinforcement Learning— each offer a unique set of capabilities and trade-offs. No single method is universally superior; the optimal choice is highly dependent on the specific requirements of the industrial application, such as the complexity of the task, the dynamism of the environment, and the stringency of safety and verification requirements. The following table provides a comparative analysis across key attributes relevant to industrial automation.

Table 1. Comparative Analysis of Task Planning Paradigms

Attribute	Finite State Machine (FSM)	Behavior Tree (BT)	Hierarchical Task Network (HTN)	Symbolic Planning (PDDL)	Reinforcement Learning (RL)
Scalability	Poor (State Explosion)	High (Modular Composition)	High (Hierarchical Decomposition)	Poor (Combinatorial Explosion)	High (Function Approximation)
Reactivity	Poor (Rigid Transitions)	Excellent (Preemption/Fallbacks)	Moderate (Requires Re-planning)	Poor (Requires Re-planning)	Excellent (Learned Policy)
Development Effort	Low for simple, High for complex	Moderate (Graphical Design)	High (Domain Knowledge Engineering)	High (Formal Modeling)	Very High (Reward Design, Training)

Verifiability/Safety	High (Deterministic, Verifiable)	Moderate (Traceable Logic)	Moderate (Depends on Hierarchy)	High (Provably Correct Plans)	Very Low ("Black Box" Policy)
Primary GenAI Role	NL-to-Code Generation	NL-to-Tree Generation / Failure Repair	Automated Knowledge Engineering	NL/Vision-to-Problem Generation	Synthetic Environment Generation

The integration of generative AI with robotic task planning is a rapidly evolving field. Nowadays, important unsolved challenges are hindering its full exploitation:

- Verification and Validation of AI-Generated Artifacts: This remains the single greatest barrier to the widespread adoption of generative AI for safety-critical industrial tasks. How can we formally verify the correctness, completeness, and safety of a task plan, an FSM, or a BT that has been generated by a probabilistic, "black box" AI model? Developing new theories, tools, and standards for the validation of AI-generated control logic is an urgent and paramount research challenge.
- Safe and Continual Lifelong Learning: For true autonomy, robots must be able to learn from their experiences and adapt their behavior over time. A key challenge is enabling this lifelong learning to happen safely and continuously without "catastrophic forgetting" (where learning a new task causes the model to forget how to do an old one) or performance degradation. Frameworks like BETR-XP-LLM, which automatically update a BT based on failure, are a first step, but ensuring the long-term stability and safety of such self-modifying systems is a major open problem.

The following table summarizes the most impactful and demonstrated applications of generative AI for each planning methodology.

Table 2. Applications of Generative AI in Task Planning Methodologies

Planning Method	Generative AI Application	Description	Example/Framework	Supporting Sources
Finite State Machine	FSM Code Generation	LLM translates natural language specifications into FSM code.	llmstatemachine	[9]
Behavior Tree	BT Generation from NL	LLM translates natural language command into an executable BT.	LLM-BT, LLM-OBTEA	[11]
Behavior Tree	Dynamic Failure Resolution	LLM analyzes an execution failure and suggests a new BT subtree to handle the error.	BETR-XP-LLM	[14][15]
Symbolic Planning (PDDL)	Problem File Generation	A Vision-Language Model analyzes an image or text to automatically generate a PDDL problem file.	Image2PDDL	[17]
Symbolic Planning (PDDL)	Domain Model Augmentation	LLM diagnoses plan failures and suggest missing preconditions to repair the PDDL domain.	LASP	[17]
Symbolic Planning (PDDL)	Neuro-Symbolic Planning	LLM is fine-tuned on thousands of PDDL problem-plan pairs to generate new plans	Teriyaki	[17]

Reinforcement Learning	Synthetic Environment Generation	Generative models create diverse, random simulation worlds for RL training.	NVIDIA Drive Sim, Tesla FSD Sim	[19]

7. Conclusions

The landscape of task planning and execution in industrial automation is undergoing a paradigm shift, driven by the dual pressures of increasing environmental complexity and the transformative potential of artificial intelligence. The trajectory from the rigid determinism of Finite State Machines to the adaptive learning of Reinforcement Learning reflects a continuous search for greater autonomy and flexibility. Each methodology occupies a vital niche, offering a distinct set of trade-offs between verifiability, reactivity, and developmental effort.

Generative AI is not emerging as a sixth, competing paradigm. Rather, it is a profound enabling technology that is fundamentally reshaping how we design, implement, debug, and interact with all existing planning frameworks. It promises to shatter the knowledge acquisition bottlenecks that have hampered HTNs and PDDL, democratize automation machine programming through natural language interfaces for BTs, and solve the critical sample efficiency problem for RL through synthetic data generation. However, this promise is tempered by the significant risks of hallucination, the challenge of verification, and the opaqueness of its decision-making processes.

The future of advanced industrial automation, therefore, does not belong to a single "best" method, nor does it belong to a fully autonomous, end-to-end generative AI. It belongs to intelligently architected, verifiable hybrid systems. In these systems, formal methods like PDDL and FSMs will continue to provide a bedrock of safety and reliability, while adaptive frameworks like BTs and RL will provide the necessary flexibility and reactivity. Generative AI will serve as the critical, symbiotic layer that bridges the gap between high-level human intent and robust, verifiable machine execution, finally enabling a truly intelligent and collaborative industrial automation.

References

[1] Zhao, Z., Cheng, S., Ding, Y., Zhou, Z., Zhang, S., Xu, D., & Zhao, Y. (2024). A survey of optimization-based task and motion planning: From classical to learning approaches. *IEEE/ASME Transactions on Mechatronics*.

[2] Heuss, L., Gebauer, D., & Reinhart, G. (2024). Concept for the automated adaption of abstract planning domains for specific application cases in skills-based industrial robotics. *Journal of Intelligent Manufacturing*, *35*(8), 4233-4258.

[3] Mateus, J. E. C., Aghezzaf, E. H., Claeys, D., Limère, V., & Cottyn, J. (2018). Method for transition from manual assembly to human-robot collaborative assembly. IFAC-PapersOnLine, 51(11), 405-410.

[3] Ju, C., & Son, H. I. (2021). A hybrid systems-based hierarchical control architecture for heterogeneous field robot teams. IEEE Transactions on Cybernetics, 53(3), 1802-1815.

[4] The Benefits of Finite State Machines in Industrial Automation, accessed July 3, 2025, https://www.crossmuller.com.au/news/the-benefits-of-finite-state-machines-in-industrial-automation/

[5] Kokotinis, G., Michalos, G., Arkouli, Z., & Makris, S. (2024). A behavior trees-based architecture towards operation planning in hybrid manufacturing. *International Journal of Computer Integrated Manufacturing*, *37*(3), 324-349.

[6] Attolino, N., Capitanelli, A., & Mastrogiovanni, F. (2025). Achieving Scalable Robot Autonomy via neurosymbolic planning using lightweight local LLM. *arXiv preprint arXiv:2505.08492*.

[7] Cohenour, C. (2017, June). Teaching finite state machines (FSMs) as part of a programmable logic control (plc) course. In *2017 ASEE Annual Conference & Exposition*.

[8] Iovino, M., Förster, J., Falco, P., Chung, J. J., Siegwart, R., & Smith, C. (2023, May). On the programming effort required to generate behavior trees and finite state machines for robotic applications. In *2023 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 5807-5813). IEEE.

[9] Gan, X. R., Song, Y. R., Walker, N., & Cakmak, M. (2024). Can Large Language Models Help Developers with Robotic Finite State Machine Modification?. *arXiv preprint arXiv:2412.05625*.

[10] Dortmans, E., & Punter, T. (2022). Behavior trees for smart robots practical guidelines for robot software development. Journal of Robotics, 2022(1), 3314084.

[11] Styrud, J., Iovino, M., Norrlöf, M., Björkman, M., & Smith, C. (2024). Automatic Behavior Tree Expansion with LLMs for Robotic Manipulation. *arXiv preprint arXiv:2409.13356*.

[12] Georgievski, I., & Aiello, M. (2015). HTN planning: Overview, comparison, and beyond. *Artificial Intelligence*, *222*, 124-156.

[13] Haridasan, P. K., & Jawale, H. (2024). Generative AI in Manufacturing: A Review of Innovations, Challenges and Future Prospects. *J Artif Intell Mach Learn & Data Sci 2024*, *2*(2), 1418-1424.

[14] Wang, J., Shi, E., Hu, H., Ma, C., Liu, Y., Wang, X., ... & Zhang, S. (2024). Large language models for robotics: Opportunities, challenges, and perspectives. Journal of Automation and Intelligence.

[15] Qureshi, F., Terzopoulos, D., & Gillett, R. (2004, May). The cognitive controller: a hybrid, deliberative/reactive control architecture for autonomous robots. In International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (pp. 1102-1111). Berlin, Heidelberg: Springer Berlin Heidelberg.

[16] Chen, G., Yang, L., Jia, R., Hu, Z., Chen, Y., Zhang, W., ... & Pan, J. (2024). Language-augmented symbolic planner for open-world task planning. *arXiv preprint arXiv:2407.09792*.

[17] Nabizada, H., Jeleniewski, T., Beers, L., Weigand, M., Gehlhoff, F., & Fay, A. (2025). Integrating AI Planning Semantics into SysML System Models for Automated PDDL File Generation. *arXiv preprint arXiv:2506.06714*.

[18] Antonelli, D., Zeng, Q., Aliev, K., & Liu, X. (2021). Robust Assembly Sequence Generation in a Human - Robot Collaborative Workcell by Reinforcement Learning. FME Transactions, 49(4).

[19] Zhang, K., Yun, P., Cen, J., Cai, J., Zhu, D., Yuan, H., ... & Chen, H. (2025). Generative artificial intelligence in robotic manipulation: A survey. *arXiv preprint arXiv:2503.03464*.