JARVIS

AI-DRIVEN TASKS AND ACTION PLANNING

Co-funded by the European Union

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement n° 101135708. The dissemination of results herein reflects only the author's view, and the European Commission is not responsible for any use that may be made of the information it contains.

INTRODUCTION

Agile manufacturing requires planning tools that can react to frequent changes in tasks, resources, and operator conditions. Traditional planning workflows are rigid and difficult to update, especially when unexpected disturbances occur. Within the JARVIS framework, the Task Planner Module (TPM) addresses these limitations by combining mixed-initiative planning with reinforcement learning. The module supports human operators while enabling fast and adaptive scheduling in dynamic industrial environments.

TPM also enables automatic generation of task models from natural language, videos, or images. Large Multimodal Models interpret these raw inputs and extract structured information such as task sequences, resource constraints, and performance parameters. As a result, operators no longer need to define formal models manually. They guide the system through intuitive inputs, and the AI evaluates many scheduling possibilities to propose efficient workplans that balance throughput, cost, safety, and ergonomics.

MODULE OVERVIEW

TPM consists of two tightly connected layers (Figure 1). The upper layer is an LMM-supported interface that accepts raw operator or system inputs describing the process and converts them into a structured configuration file (task model). This file, maintained alongside the training baseline, serves as the single source of truth for all runtime adjustments. Updates to resources, agent applicability, task metrics, global weights, and completion status are expressed exclusively through this override mechanism.

Once updated, the lower layer, based on Reinforcement Learning (RL), ingests the consolidated configuration, executes the current policy in planning mode, and generates a revised schedule. Training is performed offline and is only required when task or resource structures change significantly.

The Reinforcement Learning component typically uses a policy-gradient method with action masking to enforce constraints and feasibility. Observations encode task and resource states. During replanning, the system intercepts policy outputs: the policy proposes task order, while a lightweight post-selection step assigns agents according to the latest override parameters. This enables immediate adaptation to changes in duration, cost, or safety without retraining. Final timings are produced by a deterministic scheduler that ensures exclusivity, dependencies, and operational validity.

The system can be deployed through command-line tools, REST services, or middleware (ROS2). These interfaces allow users or autonomous agents to load configurations, trigger training, and request replanning based on runtime events. Outputs can be converted into downstream scheduling or execution formats for integration with manufacturing, robotic, or enterprise systems.

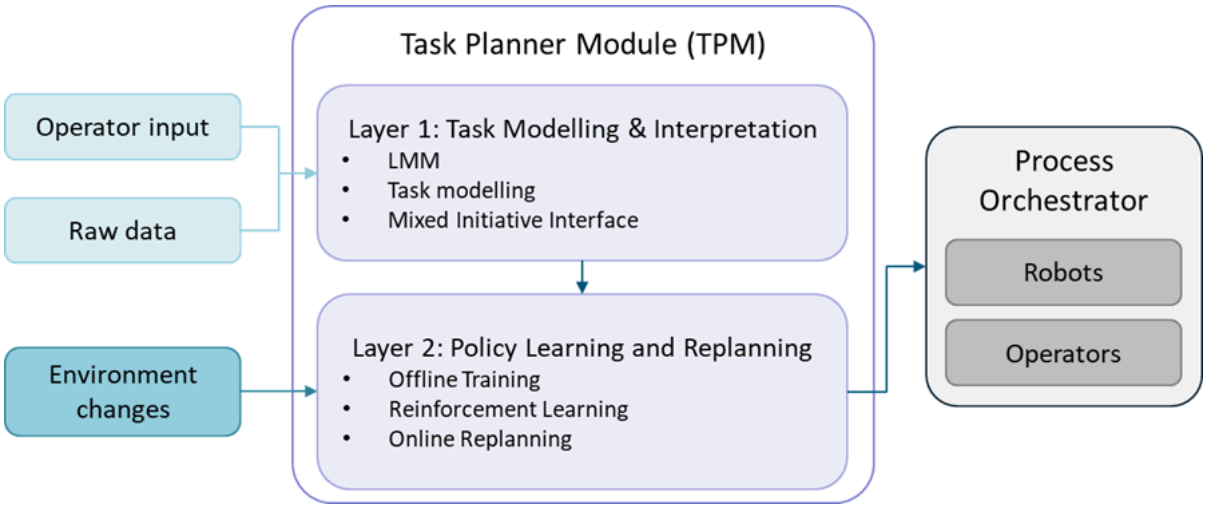

Figure 1: TPM Architecture

PRACTICAL APPLICATIONS AND EXAMPLES

A typical commissioning workflow begins with defining a baseline task model that specifies tasks, precedence relations, admissible resources, and key performance parameters such as duration, cost, safety, and ergonomics. An LMM supports this step by helping engineers and operators interpret raw data and generate a consistent task model. After collecting moderate training data, the reinforcement-learning policy converges to behavior that balances throughput, cost efficiency, and ergonomics according to predefined weights. Once a stable policy is stored, routine operation depends mainly on replanning rather than retraining.

Replanning is driven by updates to the override configuration, allowing the system to adapt quickly to changing conditions. Resource unavailability is handled by marking agents offline; the planner then removes related actions and reallocates tasks. Performance changes, such as new duration or cost values, are applied through parameter updates, and the post-selection mechanism adjusts assignments accordingly. Completed tasks can be flagged so the planner focuses only on remaining nodes. Similarly, shifts in objectives (e.g., prioritizing cost over throughput) are applied through updated global weights,

enabling the system to reflect new priorities while maintaining feasibility and safety.

EXPECTED OPERATIONAL BENEFITS

Overall, the TPM module is expected to reduce configuration time by generating task models directly from natural inputs and to simplify planning through automated optimization. Runtime changes, such as resource availability or performance variations, can be addressed quickly through fast replanning, enabling smoother and more direct adaptation to disturbances. This will improve responsiveness, stabilize workflow execution, and reduce the manual burden on engineers. In practice, teams will benefit from faster deployment, greater operational flexibility, and more seamless integration of human and robotic resources.