

Technical paper

Physics-grounded multi-agent architecture for traceable, risk-aware human–AI decision support in manufacturing[☆]

Danny Hoang^{a,} , Ryan Matthiessen^b, Christopher Miller^b, Nasir Mannan^b, Ruby ElKharboutly^c, David Gorsich^d, Matthew P. Castanier^{d,} , Farhad Imani^{a, *}

^a School of Mechanical, Aerospace, and Manufacturing Engineering, University of Connecticut, Storrs, CT, USA

^b Connecticut Center for Advanced Technology, East Hartford, CT, USA

^c Quinnipiac University, Hamden, CT, USA

^d US Army DEVCOM Ground Vehicle Systems Center, Warren, MI, USA

ARTICLE INFO

Keywords:

Multi-agent systems

Industrial large language models

Physics-grounded AI

Traceable decision support

ABSTRACT

High-precision CNC machining of free-form aerospace components requires bounded compensations informed by inspection, simulation, and process knowledge. Off-the-shelf large language model (LLM) assistants can generate text, but they do not reliably execute risk-constrained multi-step numerical workflows or provide auditable provenance for high-stakes decisions. We present multi-agent knowledge analysis (MAKA), a decision-support architecture that separates intent routing, tools-only quantitative analysis, knowledge graph retrieval, and critic-based verification that enforces physical plausibility, safety bounds, and provenance completeness before recommendations are surfaced for human approval. MAKA is instantiated on a Ti-6Al-4V rotor blade machining testbed by fusing virtual-machining path-tracking error fields, cutting-force and deflection simulations, and scan-based 3D inspection deviation maps from 16 blades. The analysis decomposes deviation into an evidence-linked pathing component, a drift-based wear proxy capturing systematic evolution across parts, and a residual systematic compliance term. In a three-level tool-orchestration benchmark spanning single-step to three-or-more-step stateful sequences, MAKA increased the pass rate for correct tool selection, argument assignment, and dependency-consistent execution by a mean, median, and maximum of 22.9, 8.0, and 64.0 percentage points, respectively, across 15 model-level comparisons over the stronger of the ReAct-style and structured function-calling baselines with identical model and tool access. Digital twin what-if studies show MAKA can coordinate traceable compensation candidates that reduce predicted surface deviation from order 10^{-2} in to approximately $\pm 10^{-3}$ in over most of the blade within the simulation environment, providing a pre-deployment verification signal for risk-aware human decision-making.

1. Introduction

Advanced computer numerical control (CNC) manufacturing systems are increasingly embedded in complex production environments in which cutting mechanics, structural compliance, servo control, fixturing, sensing, and part geometry interact in strongly nonlinear and state dependent ways [1–3]. Achieving consistent part quality in this setting requires decisions that reconcile heterogeneous evidence streams under stringent tolerance, throughput, and safety constraints. Micrometer level requirements are often specified for critical free form surfaces, yet the limiting factor in many industrial workflows is not the availability of data. The limiting factor is the lack of integrated and accountable interpretation that converts multi-source evidence into bounded actions

with explicit justification. Conventional process planning pipelines and basic statistical monitoring typically treat operations or features in isolation and therefore provide no unified view of simulation outputs, inspection metrology, and prior machining knowledge [4]. This fragmentation is increasingly misaligned with modern manufacturing practice where engineering decisions must be justified, auditable, and robust to uncertainty.

In standard industrial practice, tool paths are generated and verified offline in computer aided manufacturing (CAM) and virtual machining environments [5], while process parameters are tuned using a combination of design of experiments studies, empirical rules, and shop floor experience [6,7]. Quality control is then performed downstream via

[☆] DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited. OPSEC11018.

* Corresponding author.

E-mail address: farhad.imani@uconn.edu (F. Imani).

<https://doi.org/10.1016/j.jmansys.2026.09.010>

Received 1 April 2026; Received in revised form 7 August 2026; Accepted 16 September 2026

0278-6125/© 2026 The Society of Manufacturing Engineers. Published by Elsevier Ltd. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

sparse inspection and statistical process control charts, which summarize deviations at a limited set of features but provide only coarse diagnostics of underlying error mechanisms [8]. These stages are typically connected through manual interpretation rather than through a formal evidence fusion and verification pipeline. As a result, rich multimodal information available in modern CNC manufacturing systems [9,10], including deflection simulations, virtual machine path tracking predictions, dense scan derived deviation fields, machine telemetry, and documented best practices remains distributed across software tools and data silos. The absence of a consistent mechanism for coordinate consistent fusion, provenance tracking, and risk bounded recommendation makes it difficult to deploy automated decision support that humans can rely on in high consequence settings.

These limitations become more pronounced for non-prismatic, free-form components where local geometry [11], tool engagement [12], and fixture stiffness vary continuously along the tool path [13]. In such settings, offline one-shot planning struggles to keep up with variability introduced by changes in machines, tooling, materials, and fixturing configurations across production systems. Even small modifications to any of these elements can invalidate previously calibrated parameter sets and degrade transferability of best settings between nominally similar setups. Engineers are therefore often forced into iterative trial and error adjustment of offsets, tool paths, and parameters [14,15]. At the same time, attributing observed geometric deviation to specific contributors such as path tracking error, tool wear, or workpiece compliance requires combining imperfect models and noisy measurements [16]. Incorrect attribution directly leads to incorrect compensation and costs that include scrap, rework, tool damage, and schedule disruption.

Large language models (LLMs) offer capabilities that are attractive for industrial decision-support, including natural language interaction for operators [17,18], automated organization and summarization of heterogeneous artifacts [19,20], and coordination across multiple software tools through tool calling patterns [21,22]. In principle, an LLM enabled assistant could read simulation reports [23], interpret metrology outputs [24], consult machining studies, and propose targeted updates to tool paths and process parameters [25]. In practice, off-the-shelf LLMs are not designed for high stakes industrial workflows. They lack explicit representations of machine kinematics, cutting mechanics, and CAM and CNC constraints, are vulnerable to hallucination and inconsistent numerical reasoning, and provide weak guarantees of provenance, repeatability, and bounded action, which are central requirements for trustworthy human AI collaboration in production environments [26]. These limitations are amplified by resource and security constraints at or near the edge, where extensive retraining and unconstrained multimodal reasoning are impractical. Effective human AI symbiosis in CNC manufacturing therefore requires architectures that constrain machining critical computation to deterministic tools, retrieve domain knowledge with traceable evidence, and verify candidate actions against physical plausibility and safety bounds before they are presented for human decision-making.

We introduce the multi-agent knowledge analysis (MAKA) framework as an LLM-driven agentic decision-support layer for CNC manufacturing systems. Rather than treating the LLM as a monolithic generator of machining strategies, MAKA decomposes the evidence to recommendation pipeline into specialized auditable roles aligned with risk aware industrial symbiosis: a Central agent performs intent interpretation and task routing; an Analysis agent is constrained to tools only quantitative computation; a Knowledge agent performs evidence linked retrieval over domain knowledge constructs; and a Critic agent evaluates whether candidate outputs are coherent, physically plausible, within safety bounds, and supported by explicit provenance before they are surfaced for human review. This separation of concerns supports accountable operation because machining critical quantities are produced through deterministic tool calls and traceable retrieval rather than through unconstrained text generation. By offloading bound checking, consistency checks, and provenance completeness to the Critic agent,

MAKA reduces the cognitive burden of multimodal evidence synthesis and enables the human expert to focus on higher level tradeoffs such as quality, tool life, and cycle time.

We instantiate MAKA on an aerospace application, the production of Ti-6Al-4V titanium rotor blades on industrial CNC equipment. MAKA fuses evidence across the manufacturing digital thread, including virtual machining outputs for path tracking deviation, cutting force and workpiece deflection simulations, and dense scan based inspection deviation maps acquired across multiple manufactured blades, together with a machining knowledge graph derived from a curated corpus of technical literature capturing machining mechanisms and constraints across multiple CNC manufacturing systems. The agents structure this evidence into interpretable deviation attributions including pathing, a tool wear proxy, and a residual compliance term, then synthesize candidate compensation strategies expressed as tool path modifications, tool length and radius offsets, and process parameter adjustments with provenance linked explanations and verification checks. The contributions of this work are threefold. First, we present a physics-grounded adaptation of LLM-based decision-support for CNC manufacturing via explicit tool interfaces and knowledge graph integration that operationalizes accountable human oversight. Second, we propose an agent-based architecture that improves reliability of multi-step industrial tool orchestration under resource constraints through separation of intent routing, computation, retrieval, and verification. Third, we demonstrate within an industrial digital twin toolchain, using virtual machining and physics-based simulation environments, that multimodal data fusion combining simulation outputs, scan derived deviation fields, and symbolic machining knowledge can be executed within a single traceable decision support workflow that provides a verified pre-deployment baseline for risk aware human decision-making.

The remainder of the paper is organized as follows. Section 2 reviews related work in manufacturing knowledge systems, LLM-based decision-support, CNC automation, and geometric error modeling, along with their limitations. Section 3 presents the MAKA architecture as an agentic layer for CNC manufacturing systems, detailing the roles of the Central, Analysis, Knowledge, and Critic agents and the associated agentic tools. Section 4 describes the experimental setup used to evaluate MAKA on rotor blade machining. Section 5 reports the results on physics-based deviation decomposition, automated parameter selection, and toolpath compensation. Finally, Section 6 discusses limitations and opportunities for deploying such agentic LLM frameworks in broader CNC manufacturing environments.

2. Research background

In industrial systems, human–AI decision support is not achieved by adding a conversational interface alone. It requires that the AI system (i) integrate heterogeneous evidence across the manufacturing digital thread, (ii) remain grounded in validated physics and deterministic computation where required, (iii) expose provenance and uncertainty in forms that support human oversight, and (iv) enforce bounded autonomy through verification and safety checks before recommendations are acted upon. Accordingly, this background progresses from foundational knowledge representations to increasingly autonomous LLM-based manufacturing systems. It first reviews knowledge graphs, ontologies, and skill-based infrastructures for representing manufacturing knowledge and capabilities. It then examines LLM-based assistants and multi-agent systems across manufacturing processes, including recent applications in material extrusion, before discussing alternative mechanisms for incorporating domain knowledge through fine-tuning (FT), retrieval-augmented generation, and knowledge graphs. Finally, the review narrows to CNC programming, monitoring, and data-driven modeling of geometric error in aero-engine blades and related thin-walled, free-form components.

A long-standing response to fragmentation in manufacturing knowledge is to formalize process knowledge and system capabilities into explicit, machine-interpretable structures. Knowledge-graph-based process-planning frameworks encode parts, operations, resources, and precedence constraints as symbolic graphs and use graph reasoning to support computer-aided process planning, knowledge reuse, and consistency checking [27]. These representations capture best practices and make process dependencies explicit, which is valuable for interoperability and lifecycle management. However, they typically operate at the level of discrete operations and resources rather than spatially resolved process physics, and they rarely represent dense simulation fields, scan-derived deviation maps, or uncertainty in a form that directly supports high-stakes compensation decisions on free-form geometries. In parallel, skill-based architectures expose machine capabilities as standardized skills that can be orchestrated at runtime across heterogeneous equipment [28]. Although these approaches improve modularity and reconfigurability, they often treat the internal manufacturing process as a black box and provide limited support for combining simulation, metrology, and expert heuristics into an auditable explanation of why a part deviates and which bounded intervention is appropriate. This creates a gap between what a production system can do, as represented by its skills and operations, and why a particular action should be selected, as supported by evidence, provenance, and physical plausibility.

More recently, LLMs have been explored as a means of accessing unstructured manufacturing knowledge and providing natural-language interfaces to complex production systems. Survey work on LLMs for manufacturing identifies opportunities across the product lifecycle, including design assistance, documentation mining, process planning, and decision-support, and emphasizes retrieval-augmented generation (RAG) and tool use as important mechanisms for connecting language models with domain knowledge and engineering software [29]. Empirical studies of LLM-powered knowledge-sharing systems further demonstrate that shop-floor personnel can query manuals, procedures, and historical reports more effectively than through conventional keyword search alone [30]. These contributions are important for human-centered industrial interaction, but most remain predominantly text-centric. They reason over documents and structured tables rather than the multimodal artifacts that dominate machining decisions, such as CAM outputs, virtual-machining fields, scan-derived deviation maps, simulation results, and CNC telemetry. They also generally provide limited mechanisms for provenance completeness, in which each numerical or procedural recommendation is linked to verifiable evidence, and for risk-aware verification, in which physically implausible or unsafe recommendations are rejected or escalated.

A complementary branch of research embeds LLMs within multi-agent or agent-based manufacturing architectures. In these systems, language models interpret human instructions, decompose complex requests into subtasks, and coordinate specialized agents that may retrieve information, execute operations, or generate machine-control programs [31]. Other frameworks translate informal textual specifications into structured process plans using LLM-based task decomposition and workflow synthesis, sometimes combined with verification layers, such as state-machine checking, to establish basic logical correctness before deployment on industrial platforms [32]. These approaches are promising for interoperability and adaptive autonomy, but their support for risk-aware decision-making is often limited by two factors. First, numerical and physical calculations may still be performed within the language-model reasoning loop without deterministic guarantees. Second, verification commonly focuses on syntactic or logical consistency rather than physical plausibility, bounded action constraints, and provenance completeness over heterogeneous manufacturing evidence. These limitations are not restricted to subtractive manufacturing; similar opportunities and reliability concerns are emerging in additive-manufacturing applications in which LLMs interact with process data, machine parameters, images, and G-code.

These broader developments in agentic manufacturing are also evident in material-extrusion additive manufacturing, where LLMs have been used to monitor processes, evaluate machine instructions, and recommend parameter modifications. For example, LLM-3D Print analyzes layerwise images together with printer parameters to identify common fused-deposition-modeling defects and generate corrective actions during printing [33]. At the pre-processing stage, LLM-ADAM decomposes G-code screening into specialized extraction, reference, and judgment roles that compare structured process parameters with equipment-specific operating ranges and deterministic deviation evidence [34]. LLM-generated recommendations have also been incorporated into Bayesian-optimization loops for FDM parameter selection, allowing subsequent recommendations to be conditioned on measured outcomes from earlier print configurations [35]. Although these studies address a different manufacturing process, they demonstrate a transferable architectural principle: LLM reliability improves when language-based reasoning is constrained by sensing, structured process states, optimization algorithms, equipment specifications, and deterministic checks. However, existing material-extrusion systems primarily focus on defect detection, G-code screening, or parameter selection rather than the multi-source attribution of geometric error and bounded compensation using physics simulations, inspection measurements, and traceable process knowledge.

Across both additive and subtractive manufacturing, a central challenge is therefore how to ground LLM outputs in domain knowledge and process evidence. To improve numerical reliability and transparency, several systems couple LLMs with structured manufacturing knowledge. Knowledge-graph-augmented frameworks construct machining knowledge graphs from technical documents, vendor datasheets, experimental studies, and prior process plans, and use RAG over these graphs to answer CNC process-planning queries with traceable numerical recommendations [18]. Multi-agent retrieval frameworks further decompose retrieval, ranking, evidence integration, and answer generation into specialized roles to combine information from heterogeneous manufacturing sources [36]. In mechanical engineering more broadly, multimodal systems such as MechRAG integrate CAD and computer-aided engineering assets into LLM responses, enabling conversational access to simulation results and model artifacts [23]. CNC-focused assistants such as ChatCNC connect LLMs to real-time machine-data streams to support conversational monitoring and diagnostics [25]. Collectively, these studies show that LLMs can be grounded in structured knowledge, multimodal engineering artifacts, and live telemetry. Nonetheless, they often stop short of (i) decomposing geometric error into physically interpretable contributors under explicit assumptions, (ii) orchestrating deterministic analysis tools through an auditable tool-call interface, and (iii) enforcing a verification gate that bounds recommendations and escalates ambiguous cases to human experts.

This grounding challenge raises an important distinction between internalizing manufacturing knowledge within model parameters and retrieving it from external sources at inference time. Parametric approaches, including continued domain pretraining, supervised instruction tuning, and parameter-efficient fine-tuning, modify model weights to improve domain terminology, task behavior, or output structure. However, fine-tuning does not inherently guarantee factual correctness, and controlled experiments indicate that introducing new factual knowledge through supervised fine-tuning can increase hallucination when the model learns incomplete or weakly supported associations [37]. Non-parametric approaches, including retrieval-augmented generation and knowledge-graph-based retrieval, instead maintain technical knowledge outside the model and provide query-relevant evidence during inference. In polymer-composite additive manufacturing, RAG improved the accuracy and relevance of domain responses, whereas naive fine-tuning on raw technical documents reduced performance relative to the pretrained baseline [38]. These findings do not imply that fine-tuning is unsuitable for manufacturing. Rather, they suggest that fine-tuning is better suited to learning behaviors such as

task routing, tool selection, structured response generation, evidence adherence, and abstention than to serving as the sole repository of changing technical facts. Retrieval-aware fine-tuning can therefore complement external grounding by teaching the model to prioritize supported evidence and resist irrelevant or misleading retrieved information [39]. Fine-tuning, retrieval, and knowledge graphs should consequently be viewed as complementary mechanisms: fine-tuning adapts model behavior, whereas retrieval and knowledge graphs provide updateable, traceable, and auditable domain knowledge.

The consequences of grounding and adaptation are especially important in CNC applications, where an incorrect model output may be translated into machine-executable instructions. Several studies have evaluated or proposed LLM-based generation of CNC programs directly from natural-language descriptions or simple geometric sketches, highlighting both the promise and the risks of treating toolpath synthesis as a text-generation task [25,40]. Comparative evaluations show that general-purpose LLMs can produce syntactically valid G-code, but that the resulting programs often require substantial human correction and verification, particularly for complex geometries or nonstandard machine configurations [41]. More specialized frameworks introduce domain-specific fine-tuning, retrieval, self-correction loops, and syntactic or geometric validation [42]. Nevertheless, these systems generally operate outside physics-based digital-twin environments and often rely on a single language model to make intertwined geometric, physical, and process decisions. For safety-critical CNC applications, the LLM should therefore function primarily as an orchestrator of validated planning, analysis, and physics tools rather than as an unconstrained generator of machining strategies.

While the preceding studies focus primarily on knowledge access, process planning, and machine interaction, reliable compensation also requires quantitative models of how geometric errors arise and vary across a manufactured component. A substantial body of work has therefore examined data-driven error distributions in thin-walled aero-engine blades and related components. For example, Gaussian-mixture-model-based frameworks have been proposed to predict full-field milling-error distributions for blades and impellers across different process conditions, using experimental data to characterize complex spatial patterns of deviation [43]. Such models capture important aspects of process behavior and can provide informative priors for compensation. However, they are typically developed as standalone prediction tools, with limited integration into decision-making pipelines that also incorporate virtual machining, physics-based deflection simulation, inspection measurements, and symbolic knowledge of machining constraints. They also rarely provide an evidence-linked explanation of how contributors such as path-tracking error, process drift, compliance effects, and residual variability combine to produce the observed deviation pattern on a specific production system.

Table 1 summarizes the progression from symbolic manufacturing knowledge systems to LLM-based assistants, agentic manufacturing frameworks, CNC-oriented systems, additive-manufacturing applications, and physics-based geometric-error models. The comparison emphasizes whether each approach integrates structured knowledge, language-model reasoning, multi-agent coordination, retrieval or fine-tuning, physics-based decomposition, digital-twin verification, and deterministic tool orchestration.

Taken together, the reviewed approaches address important but largely separate parts of the manufacturing decision-support problem. Knowledge graphs and skill-based systems provide structured representations but generally lack spatially resolved process analysis. LLM assistants and agentic systems improve knowledge access, human interaction, and task coordination but often provide limited guarantees of numerical correctness, physical plausibility, and provenance completeness. Material-extrusion applications demonstrate the value of combining LLMs with sensing, optimization, and structured process constraints, but primarily address defect monitoring, G-code

screening, and parameter selection. CNC-generation systems demonstrate machine-level interaction but continue to require substantial validation, while geometric-error models provide quantitative predictions without integrating them into a traceable, knowledge-grounded decision workflow. MAKa addresses this gap by combining knowledge graph retrieval, specialized agent roles, deterministic quantitative tools, physics-based error decomposition, critic-mediated verification, and digital-twin evaluation within a single auditable framework. Rather than relying on fine-tuning to internalize changing process knowledge, MAKa externalizes factual knowledge and manufacturing constraints through traceable retrieval while using the LLM primarily for intent interpretation, task coordination, and evidence-based synthesis.

3. Methodology

The multi-agent knowledge analysis framework comprises four agents (Central, Knowledge, Analysis, and Critic) that collaborate to answer and execute decision-support queries arising within a CNC manufacturing workflow. Fig. 1 summarizes the procedure as an eight-step loop.

- ① A natural-language query is provided by a user or a higher-level planning system.
- ② The Central Agent (CA) interprets the query in the context of the current manufacturing state, including the active machine, toolpath, available datasets, and previously computed summaries.
- ③ The CA decomposes the request, constructs a task plan, and determines whether quantitative analysis, knowledge retrieval, or both are required. It then routes task-specific instructions to the Analysis Agent (AA) for tool-grounded computation and/or the Knowledge Agent (KA) for evidence retrieval from the machining knowledge graph.
- ④ The downstream agents execute their assigned tasks using deterministic tools and retrieval operators to produce intermediate outputs, such as deviation attributions, compensation candidates, and evidence-linked constraints. When both retrieval and quantitative analysis are required, the KA and AA may operate in parallel.
- ⑤ Their candidate outputs are passed to the Critic Agent (CRA).
- ⑥ The CRA synthesizes the retrieved evidence and analytical results and evaluates their logical coherence, physical plausibility, compliance with safety bounds, and provenance completeness.
- ⑦ If any requirement is unmet, the CRA issues a revised instruction to the appropriate agent for additional retrieval or analysis.
- ⑧ Otherwise, the verified recommendation and its audit trail are returned to the CA, which assembles the final response and presents it to the user.

Within each experiment, all four agents use the same underlying base model; they differ only in their lightweight role-specific instructions and access to specialized tools or retrieval operations. No additional model fine-tuning is performed, and the framework does not depend on extensive prompt engineering or separately optimized prompts for each task. Instead, agent behavior is defined through concise instructions corresponding to routing, retrieval, deterministic analysis, and verification. Consequently, the framework obtains task specialization primarily through the division of responsibilities, controlled tool access, structured intermediate outputs, and critic-based verification rather than through model retraining or complex prompt design.

MAKa follows a human-supervised decision-support paradigm rather than an explicit human-in-the-loop optimization strategy. The internal execution loop consists of autonomous coordination among the four agents, while human oversight occurs only after the verified recommendation has been produced. At this stage, a domain expert may inspect the rationale and supporting evidence, request additional analysis by refining or resubmitting the query, accept the recommendation, or reject it prior to downstream CAM implementation. The recommendation therefore functions as an auditable decision-support output rather than an autonomous control command, and converts natural language intent into tool-grounded analysis, evidence-linked reasoning, and traceable recommendations. The following subsections define each agent and the constraints used to enforce deterministic computation and evidence grounding.

Table 1
Capabilities of representative related work compared with the MAKA framework.

Model	KG/Skills	LLM	Multi-Agent	RAG/FT	Physics Decomposition	Digital-twin Verification	Tool Orchestration
Hoang et al. [18]	✓	✓	×	✓	×	×	×
Li & Corney [23]	×	✓	×	✓	×	×	×
Jeon et al. [25]	×	✓	×	✓	×	×	×
Xiao et al. [27]	✓	×	×	×	×	×	×
Hossfeld & Wortmann [28]	✓	×	×	×	×	×	×
Li et al. [29]	×	✓	×	×	×	×	×
Freire et al. [30]	×	✓	×	×	×	×	×
Lim et al. [31]	×	✓	✓	×	×	×	✓
Ni et al. [32]	×	✓	×	×	×	×	✓
Jadhav et al. [33]	×	✓	×	×	×	×	×
Eslaminia et al. [34]	×	✓	✓	×	×	×	×
Samani et al. [35]	×	✓	×	×	×	×	×
Mao et al. [36]	×	✓	✓	✓	×	×	✓
Sagor et al. [38]	×	✓	×	✓	×	×	×
Šket et al.; Abdelaal et al. [41,42]	×	✓	×	✓	×	×	×
Yang et al. [43]	×	×	×	×	✓	×	×
MAKA (Ours)	✓	✓	✓	✓	✓	✓	✓

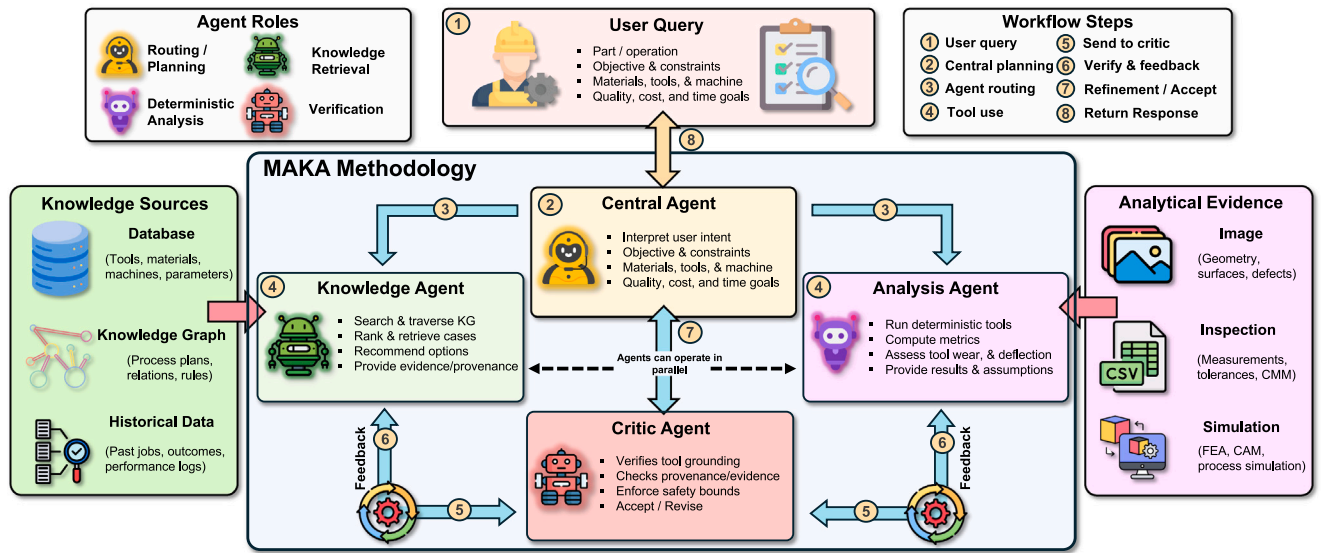


Fig. 1. Multi-agent knowledge framework for CNC manufacturing. The Central, Knowledge, Analysis, and Critic agents collaborate in an eight-step loop: (1) user query, (2) to (3) central planning and routing, (4) downstream analysis and knowledge retrieval, (5) to (7) critic evaluation and refinement, and (8) recommendation returned for human review and downstream CAM implementation.

3.1. Central agent

The Central agent serves as the interface between the user and the downstream agents. Let $\mathcal{Q}$ denote the space of user queries, $\mathcal{X}$ denote the system-state space, $\mathcal{A}$ denote the set of available downstream agents, and $\tilde{\mathcal{Q}}$ denote the space of structured agent instructions.

At interaction step t , the CA applies the routing policy

$$(a_t, \tilde{q}_t, x_t^+) = \pi_C(q_t, x_t), \quad (1)$$

where $a_t \in \mathcal{A}$ identifies the selected downstream agent, $\tilde{q}_t \in \tilde{\mathcal{Q}}$ is the structured instruction provided to that agent, and $x_t^+ \in \mathcal{X}$ is the updated system state.

Before applying the routing policy in Eq. (1), the CA extracts structured information from the user query. This preprocessing operation is expressed as

$$x_t^p = u_C(q_t, x_t), \quad (2)$$

where $u_C(\cdot)$ identifies information such as file references, part-number ranges, inspection locations, requested tool categories, and state-reset instructions. The routing policy is therefore evaluated using the preprocessed state x_t^p .

In implementation, the CA combines a language-model router with schema validation and a deterministic fallback mechanism. The language-model router produces a structured JSON object containing the selected agent, a normalized instruction, the required resource references, and the requested tool categories.

A JSON-schema validator determines whether the router output contains the required fields and valid values. If validation fails, the CA applies a deterministic routing policy π_D . This policy selects the appropriate agent based on task-specific keywords and the resources available in the current system state.

3.2. Analysis agent

The Analysis agent performs numerical and data-centric operations. It receives the structured instruction $\tilde{q}_t$, the current system state x_t , and a collection of external resources r_t . These resources may include inspection datasets, simulation results, data handles, and computational APIs.

Let

$$\mathcal{T} = T_1, T_2, \dots, T_M \quad (3)$$

denote the library of M deterministic tools available to the AA.

For a given instruction, the AA constructs a finite sequence of K tool calls:

$$I_t = [(j_1, \xi_1), (j_2, \xi_2), \dots, (j_K, \xi_K)], \quad (4)$$

where j_k identifies the selected tool and ξ_k contains the corresponding input arguments.

Each tool output is obtained as

$$z_{t,k} = T_{j_k}(\xi_k), \quad k = 1, \dots, K \quad (5)$$

The AA uses the resulting tool outputs to generate a structured candidate result y_t^A and an updated system state x_{t+1} . Reusable intermediate results are stored in the system state to avoid unnecessary recomputation.

To enforce tool-grounded reasoning, the AA output contains three components: the executed tool calls and their input arguments, the corresponding tool outputs and a provenance map linking each reported numerical value to a specific tool-output identifier.

The language model is therefore not permitted to independently generate non-trivial numerical quantities. During verification, the CRA checks that every numerical value included in the final recommendation can be traced to the AA provenance map.

3.3. Knowledge agent

The Knowledge agent answers conceptual, relational, and constraint-related queries using a machining knowledge graph. At interaction step t , the knowledge graph is represented as

$$\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathcal{R}_t), \quad (6)$$

where $\mathcal{V}_t$ is the set of entities, $\mathcal{E}_t$ is the set of graph edges, and $\mathcal{R}_t$ is the set of relation types.

Given the routed instruction $\tilde{q}_t$, the KA retrieves a supporting context c_t according to

$$c_t = \mathcal{R}_K(\tilde{q}_t, \mathcal{G}_t), \quad (7)$$

where $\mathcal{R}_K(\cdot)$ denotes the knowledge-retrieval procedure. The KA uses the retrieved context and any relevant observations stored in x_t to construct the candidate knowledge response y_t^K .

3.3.1. Semantic scoring

Each indexed knowledge item is represented as

$$e_i = (h_i, r_i, o_i, c_i), \quad (8)$$

where h_i is the head entity, r_i is the relation, o_i is the object entity, and c_i is the associated textual context.

Let $f(\cdot) \in \mathbb{R}^d$ denote the shared embedding model. The query embedding is calculated as

$$\mathbf{e}_t^q = f(\tilde{q}_t). \quad (9)$$

For each knowledge item, $\mathbf{v}_i$ denotes the embedding of the complete graph item (h_i, r_i, o_i, c_i) , while $\mathbf{u}_i$ denotes the embedding of its supporting context c_i .

The combined retrieval score is calculated as

$$\gamma_i = \cos(\mathbf{e}_t^q, \mathbf{v}_i) + \lambda \cos(\mathbf{e}_t^q, \mathbf{u}_i), \quad (10)$$

where $\lambda \in [0, 1]$ controls the contribution of contextual similarity.

3.3.2. Candidate pruning

To limit computational cost, the KA first constructs a pre-pool containing the most relevant knowledge items according to their graph-item similarity.

Let N_G denote the total number of indexed knowledge items. The number of items retained in the pre-pool is

$$p = \min[N_G, \max(m, \lfloor \alpha N_G \rfloor)], \quad (11)$$

where $\alpha \in (0, 1]$ is the pruning fraction and m is the minimum number of retained items.

The resulting pre-pool $\mathcal{P}_t$ contains the p knowledge items with the highest base similarity values

$$\beta_i = \cos(\mathbf{e}_t^q, \mathbf{v}_i). \quad (12)$$

The KA then calculates an adaptive inclusion threshold from the combined scores in $\mathcal{P}_t$:

$$\eta_t = \mu_{\mathcal{P}_t} + z_\eta \sigma_{\mathcal{P}_t}, \quad (13)$$

where $\mu_{\mathcal{P}_t}$ and $\sigma_{\mathcal{P}_t}$ are the mean and standard deviation of the scores γ_i in the pre-pool. The parameter z_η controls the strictness of the threshold.

The initial candidate set is defined as

$$\mathcal{C}_{t,0} = e_i \in \mathcal{P}_t : \gamma_i \geq \eta_t. \quad (14)$$

If Eq. (14) produces an empty set, the KA retains a fixed number k_0 of the highest-scoring items. The final number of retained candidates is bounded between the lower and upper limits k_ℓ and k_u , respectively.

3.3.3. Neighborhood expansion and evidence generation

To retrieve multi-hop evidence, the KA expands the initial candidate set through the knowledge graph. Expansion is limited by a maximum graph depth d_M and a maximum beam width b_M . A visited-item set prevents the same graph item from being processed more than once.

The final retrieved context c_t contains the selected graph triples and their associated supporting passages. The KA returns both a structured answer and an evidence list linking each claim to one or more retrieved triple identifiers.

Claims that describe machining constraints, recommended practices, or known failure modes must include supporting evidence identifiers. Claims without sufficient retrieved evidence are returned for revision by the CRA.

3.4. Critic agent

The Critic agent evaluates candidate outputs before they are presented to the user. After either the AA or KA produces a candidate answer y_t , the CRA receives the original query, the candidate answer, and the current system state.

The CRA decision policy is expressed as

$$(d_t, a_{t+1}, \tilde{q}_{t+1}, x_{t+1}^+) = \pi_R(q_t, y_t, x_t), \quad (15)$$

where the decision variable is defined as

$$d_t \in \{\text{accept}, \text{revise}\}. \quad (16)$$

An accept decision returns the candidate answer together with its provenance information and audit trail. A revise decision selects a downstream agent a_{t+1} and generates a refinement instruction $\tilde{q}_{t+1}$. The selected agent then performs the requested calculation, retrieval, or correction before returning an updated candidate answer to the CRA.

The CRA evaluates four classes of criteria:

1. Intent alignment: the candidate answer must address the original query q_t .
2. Tool grounding: every reported numerical quantity must be linked to an AA tool output.
3. Evidence grounding: every constraint or best-practice claim must be linked to evidence retrieved from the knowledge graph.
4. Safety and physical plausibility: all recommended actions must satisfy the defined coordinate conventions, allowable compensation directions, and offset bounds.

An internal quality score may be calculated as

$$J(q_i, y_i, x_i) \in [0, 1], \quad (17)$$

but the quality score alone does not determine whether a candidate answer is accepted. Acceptance requires all mandatory grounding, consistency, and safety checks to pass.

To prevent unlimited refinement, the number of CRA evaluations for a single query is restricted by a critic budget $L \in \mathbb{N}$. Let n_i^R denote the number of CRA evaluations performed for the current query. The refinement limit is reached when

$$n_i^R \geq L. \quad (18)$$

If the candidate answer does not satisfy the acceptance criteria after the refinement limit is reached, the system terminates the refinement loop. The system then returns the candidate result together with the failed verification checks and explicitly identifies the result as unverified. This prevents an unsupported or unsafe recommendation from being presented as an accepted result.

3.5. Agentic tools for geometric-deviation compensation

The rotor blade case study applies the MAKa framework to geometric-deviation analysis and compensation. The tool library integrates three forms of information: inspection-derived deviation fields, virtual-machining path-tracking errors, and process constraints retrieved from the machining knowledge graph (Table 2).

The computational procedure consists of five stages:

1. Align inspection and simulation data in a common CAD coordinate frame
2. Separate pathing, baseline systematic, drift, and variability components
3. Calculate component-attribution and variability indicators
4. Convert selected corrections into tool length and tool radius offsets
5. Verify the resulting offsets using process constraints and shop-floor bounds.

Although the case-study tools use rotor blade-specific identifiers, the agents interact with the tools through standardized geometric inputs. These inputs include point clouds, deviation vectors, part indices, inspection pair identifiers, and metadata.

Applying MAKa to another component therefore requires replacement of the component-specific computational tools rather than retraining the agent models or changing the orchestration architecture.

3.5.1. Inspection data and simulation alignment

Let $k \in 1, \dots, K_p$ identify an inspection pair and $n \in 1, \dots, N$ identify a manufactured part.

The signed normal deviations measured on the pressure and suction surfaces are denoted by $\delta_{k,n}^P$ and $\delta_{k,n}^S$, respectively. Both measurements are obtained after registering the scanned component to the nominal CAD model.

The combined thickness-direction deviation is defined as

$$v_{k,n} = \delta_{k,n}^P + \delta_{k,n}^S \quad (19)$$

Under the paired-surface sign convention, the corresponding per-surface deviation is

$$s_{k,n} = \frac{v_{k,n}}{2} \quad (20)$$

The tool `compute_inspection_pairs` constructs the matched pressure–suction pairs and returns the measured and derived quantities in Eqs. (19) and (20).

The tool `fetch_inspection_slices` retrieves selected parts or inspection regions and caches their normalized representations.

Run MyVirtual Machine (RMVM) provides the simulated path-tracking error field. A deterministic mapping operator projects this field onto the inspection pair locations in the common CAD coordinate frame.

Let r_k denote the simulated combined pathing deviation at inspection pair k . The corresponding per-surface pathing deviation is

$$p_k = \frac{r_k}{2}. \quad (21)$$

The tool `rb_compute_pathing_dev` performs the simulation-to-inspection mapping and returns the per-surface pathing deviation p_k .

3.5.2. Multi-part deviation decomposition

After removing the simulated pathing contribution, the non-pathing deviation for inspection pair k and part n is

$$u_{k,n} = s_{k,n} - p_k. \quad (22)$$

The sequence of non-pathing deviations across the manufactured parts is modeled using the affine relationship

$$u_{k,n} = c_k + b_k(n - 1) + \varepsilon_{k,n}, \quad (23)$$

where c_k is the baseline systematic term, b_k is the estimated change in deviation per manufactured part, and $\varepsilon_{k,n}$ is the remaining residual.

The mean part index is

$$\bar{n} = \frac{N + 1}{2}. \quad (24)$$

The mean non-pathing deviation at inspection pair k is

$$\bar{u}_k = \frac{1}{N} \sum_{n=1}^N u_{k,n}. \quad (25)$$

The least-squares estimate of the drift rate is

$$b_k = \frac{\sum_{n=1}^N (n - \bar{n})(u_{k,n} - \bar{u}_k)}{\sum_{n=1}^N (n - \bar{n})^2}. \quad (26)$$

The corresponding baseline systematic term is

$$c_k = \bar{u}_k - b_k(\bar{n} - 1). \quad (27)$$

The accumulated drift at the final observed part is

$$w_k^d = b_k(N - 1). \quad (28)$$

The residual associated with part n is

$$\varepsilon_{k,n} = u_{k,n} - c_k - b_k(n - 1). \quad (29)$$

For $N \geq 3$, the residual dispersion is quantified as

$$w_k^v = \sqrt{\frac{1}{N - 2} \sum_{n=1}^N \varepsilon_{k,n}^2}. \quad (30)$$

The quantity w_k^d is treated as a drift proxy describing systematic evolution across the part sequence. The quantity w_k^v is treated as a variability proxy describing process instability and unmodeled fluctuations.

The variability proxy w_k^v is not interpreted as a direct physical measurement of tool wear. Instead, it is used as an indicator of uncertainty in the predicted systematic deviation.

The tools `rb_compute_wear_drift`, `rb_compute_process_variability`, and `rb_compute_residual_systematic` implement Eqs. (26)–(30).

For reporting, the mean measured surface deviation at inspection pair k is

$$\bar{s}_k = \frac{1}{N} \sum_{n=1}^N s_{k,n}. \quad (31)$$

For a target part index n^* , the predicted surface deviation is

$$\hat{s}_k(n^*) = p_k + c_k + b_k(n^* - 1). \quad (32)$$

The baseline term c_k in Eq. (32) represents the systematic deviation not explained by the rigid-body path-tracking simulation or the estimated part-to-part drift.

The term may include compliance, stiffness-related effects, registration bias, and other repeatable sources of geometric deviation. Therefore, c_k is not interpreted as structural deflection alone.

3.5.3. Attribution and variability indicators

Attribution ratios are calculated using component magnitudes to prevent cancellation between positive and negative deviations.

For target part n^* , the pathing, baseline, and drift magnitudes are defined as

$$p_k^a = |p_k|, \quad c_k^a = |c_k|, \quad d_k^a = |b_k(n^* - 1)|. \quad (33)$$

The stabilized total magnitude is

$$A_k = p_k^a + c_k^a + d_k^a + \epsilon, \quad \epsilon > 0, \quad (34)$$

where ϵ is a small positive constant that prevents division by zero.

The corresponding attribution fractions are

$$\phi_k^p = \frac{p_k^a}{A_k}, \quad \phi_k^c = \frac{c_k^a}{A_k}, \quad \phi_k^d = \frac{d_k^a}{A_k}. \quad (35)$$

The variability indicator is calculated separately as

$$\psi_k^v = \frac{w_k^v}{|\hat{s}_k(n^*)| + \epsilon}. \quad (36)$$

The CRA uses ψ_k^v to identify inspection locations where the predicted systematic deviation is small relative to the observed variability. At these locations, compensation may be reduced or rejected because the estimated correction is not sufficiently stable.

The tool `rb_compute_attribution_fractions` implements Eqs. (33)–(36).

3.5.4. Compensation under a fixed tool-tilt angle

The cutting tool is mounted at a fixed tilt angle $\theta = 25^\circ$ relative to the local compensation reference axis.

Let Δ_k denote the signed equivalent correction selected for inspection pair k . Depending on the requested analysis, Δ_k may represent the predicted drift contribution, a bounded portion of the baseline systematic term, or another correction approved by the CRA.

The equivalent correction is decomposed into a tool length offset as

$$t_{l,k} = \Delta_k \cos \theta \quad (37)$$

and a tool radius offset as

$$t_{r,k} = \Delta_k \sin \theta \quad (38)$$

The resulting compensation vector is

$$\mathbf{t}_k = \begin{bmatrix} t_{r,k} \\ t_{l,k} \end{bmatrix} \quad (39)$$

The tools `rb_compute_tool_length`, `rb_compute_radius_offset`, and `rb_compute_pair_tool_comp` implement Eqs. (37)–(39).

Before a compensation vector is accepted, the CRA verifies the signs, coordinate conventions, and allowable values of $|t_{r,k}|$ and $|t_{l,k}|$.

3.5.5. Knowledge-based constraint verification

The tools `kg_initial` and `kg_retrieve` construct and query the machining knowledge graph. These tools return empirical rules, process constraints, and known failure modes relevant to titanium machining and tool offset application. Each retrieved constraint is accompanied by a graph-triple identifier and its supporting text. When the CRA evaluates the compensation vector in Eq. (39), it verifies that every applied constraint is supported by a retrieved evidence identifier. A recommendation is returned for revision when its supporting evidence is missing or when the proposed offset violates a safety constraint or shop-floor limit.

4. Experimental design

We evaluate the MAKa framework along five axes that map directly to the results section. First, we measure *tool-orchestration reliability* on rotor blade analytics tasks, where the system must select and sequence deterministic tools to compute deviation components and compensation values. Second, we isolate the contribution of the *Critic Agent* using paired critic-enabled and no-critic runs under undegraded and progressively degraded routing conditions. Third, we quantify the effect of *knowledge graph grounding* on quantitative machining question answering using paired KG and no-KG trials. Fourth, we present an *end-to-end rotor blade case study* in which inspection and digital-twin evidence are converted into staged compensation candidates and evaluated within the simulation environment. Fifth, we conduct a *formal expert evaluation* of MAKa outputs to assess practical Acceptance, Correctness, Decomposition Clarity, Provenance Trust, Critic Helpfulness, and Overall Interpretability.

4.1. Manufacturing case study and inspection data

The MAKa framework was evaluated using an experiment on the manufacturing of a rotor blade conducted at the Connecticut Center for Advanced Technology (CCAT). Fig. 2 summarizes the overall fabrication, simulation, and inspection workflow. Ti-6Al-4V titanium rods with a diameter of 1 1/4 in (31.75 mm) were used to manufacture a rotor blade with a length of 3 in (76.2 mm) designed at CCAT. Siemens NX CAM was used to generate baseline tool paths required to manufacture the rotor blade on a 4-axis Trak VMC10si.

As shown in Fig. 3, ten operations are required to manufacture the blade: (1) tip engage, (2) top half rough, (3) top half semi-finish, (4) bottom half rough, (5) bottom half semi-finish, (6) platform outer rough, (7) platform outer finish, (8) platform top finish, (9) blade finish, and (10) tip finish. From these operations, four test cuts were selected to modify tool usage and update tool paths based on expert observations to minimize tool wear and improve surface finish. A four-flute 1/2 in TiAlN-coated carbide ball end mill was selected for all roughing and semi-finishing operations, and a six-flute 1/4 in solid carbide undercutting end mill was used for finishing operations. The tool was offset at an angle of 25 degrees relative to the part during machining (Fig. 3a).

A total of 16 rotor blades were created and measured for geometric deviation relative to the nominal CAD model using a GOM ATOS ScanBox. For each blade, surface scans were registered to the nominal model and deviation measurements were recorded at labeled inspection points. We defined 30 inspection points distributed across the blade surface and arranged them into matched pairs across the pressure and suction sides using a consistent *pair-key* convention of the form $i+(i+15)$ (e.g., 2+17, 3+18, ..., 16+31) so corresponding locations on the two sides could be directly compared. After measuring deviations, inspection data for all blades were exported to CSV and aggregated for subsequent analysis. These inspection deviations serve as the ground-truth geometric error signal that MAKa analyzes and attempts to reduce via compensation.

Table 2
Main tool categories used by MAKa for rotor blade geometric-deviation analysis and compensation.

Category	Tool Identifiers	Function
Inspection data preparation	compute_inspection_pairs, fetch_inspection_slices	Load inspection measurements, construct matched pressure–suction pairs, calculate per-surface deviations, and return normalized subsets for selected parts and inspection locations.
Statistics and indexing	rb_compute_values, rb_compute_average, rb_compute_std_dev, rb_compute_level, rb_compute_position_in_level	Return inspection pair measurements and summary statistics and assign spanwise and within-level position indices for aggregation.
Pathing projection	rb_compute_pathing_dev	Project RMVM path-tracking errors onto inspection locations in the common CAD coordinate frame and calculate the per-surface pathing deviation p_k .
Deviation decomposition	rb_compute_wear_drift, rb_compute_process_variability, rb_compute_residual_systematic	Estimate the part-to-part drift b_k , accumulated drift u_k^d , residual variability u_k^v , and baseline systematic deviation c_k .
Attribution metrics	rb_compute_attribution_fractions	Calculate stabilized component-attribution fractions and the variability indicator used for risk-aware verification.
Compensation geometry	rb_compute_tool_length, rb_compute_radius_offset, rb_compute_pair_tool_comp	Convert a selected signed correction Δ_k into tool radius and tool length offsets under the fixed-tilt model.
Knowledge retrieval	kg_initial, kg_retrieve	Construct and query the machining knowledge graph and return evidence-linked process constraints, best practices, and known failure modes.

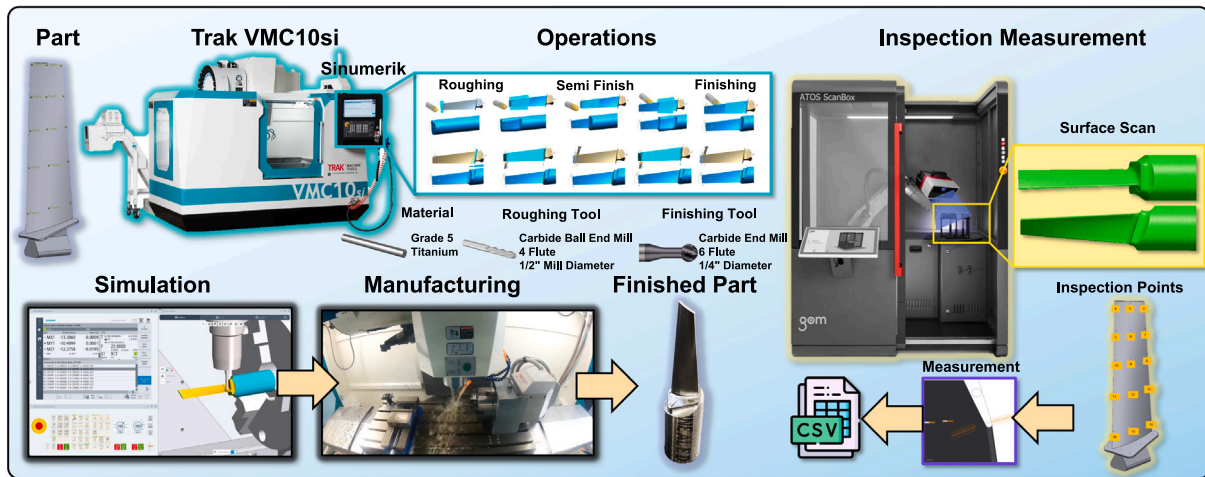


Fig. 2. Illustration of the experimental workflow from part design, to manufacturing simulation, manufacturing using a 4-axis Trak VMC10si, and inspection measurements using a GOM ATOS ScanBox.

4.2. Digital twin signals and deviation decomposition

MAKA forms an aggregate estimate of total geometric deviation by combining three complementary error signals with a residual term. The first signal captures machine path-tracking deviation, the second captures systematic drift across parts as a proxy for wear-related evolution, and the third captures the remaining baseline systematic deviation. Together, these signals provide a structured decomposition that separates kinematic execution errors, time-varying changes that accumulate over a production run, repeatable baseline effects, and residual variability. Cutting-induced workpiece deflection is evaluated separately as complementary physics-based evidence for parameter screening and compensation selection.

Workpiece deflection is estimated using ThirdWave simulations that model how the blade deforms as the tool engages the pressure and

suction sides under candidate process parameters such as spindle speed, feed rate, and surface feet per minute. The resulting deflection fields provide physics-based screening evidence of deformation sensitivity along the tool path, which allows MAKa to screen parameter sets toward reduced deflection sensitivity and prioritize corrective actions in regions predicted to be most compliant. These simulations are used as complementary evidence rather than as a directly isolated algebraic component of the measured deviation.

Path-tracking deviation is estimated using Siemens Run MyVirtual Machine by comparing the predicted executed tool motion with the nominal G-code trajectory. These virtual-machining results capture systematic tracking errors arising from machine kinematics and controller behavior, and they provide a trajectory-following deviation field that MAKa can use to propose toolpath compensations aimed at counteracting repeatable execution bias.

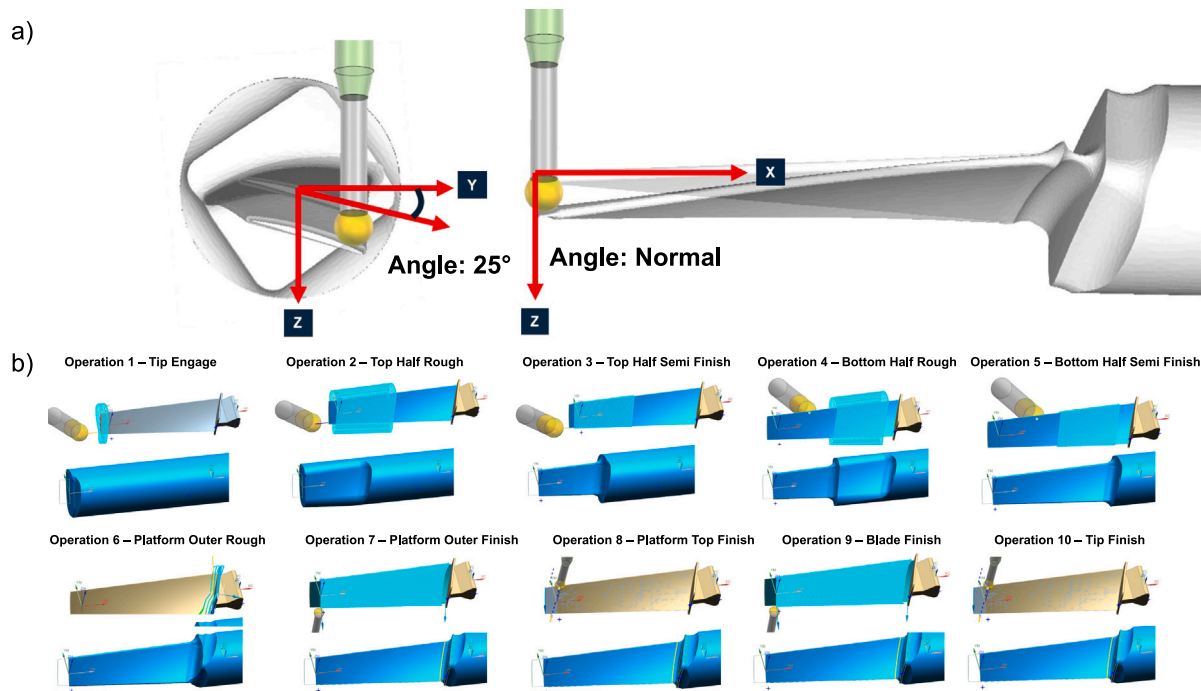


Fig. 3. Experimental setup where (a) the spindle is set at an angle of 25° relative to the blade and (b) all ten operations required to manufacture the rotor blade.

Wear-related evolution is inferred directly from inspection measurements by tracking systematic drift in deviation patterns across the sequence of manufactured blades at fixed inspection pairs. Persistent, progressive changes across parts are treated as a proxy for wear-driven changes in effective tool geometry and engagement conditions. This drift-based signal is useful because it summarizes time-dependent behavior that is not captured by a single static simulation or a single part, and it provides a basis for wear-aware updates and conservative bounding when the process becomes less stable.

After accounting for predicted path-tracking effects and estimating drift-based evolution, any remaining repeatable structure is treated as a baseline systematic contribution. This term may include structural compliance, registration bias, fixturing effects, stiffness-related behavior, and other repeatable but unmodeled sources. It is therefore not uniquely attributed to structural deflection. Residual variability is computed separately to characterize instability and prevent the decomposition from forcing all unexplained variation into one of the systematic contributors, which could otherwise lead to overconfident or misdirected compensations.

Within MAKa, these decomposition components are computed through the rotor blade analysis tool suite, including `rb_compute_pathing_dev`, `rb_compute_surface_dev`, `rb_compute_wear_drift`, `rb_compute_process_variability`, and `rb_compute_residual_systematic`. The resulting components are combined into the decomposed representation used to generate compensation actions through modified tool paths and adjusted process parameters. ThirdWave deflection simulations are used separately as physics-based screening evidence for selecting and evaluating candidate parameter adjustments. Although direct point-to-point numerical agreement between simulation outputs and physical scans is not the primary objective, strong topological alignment was observed between the simulated and measured error distributions. ThirdWave consistently identifies the blade tip and leading edge as regions of maximum deflection, which correspond to the high-deviation regions observed during inspection of the 16 blades. This qualitative spatial agreement supports use of the digital twin as a directional guide for decision-making, while the baseline systematic and residual variability terms provide flexibility for representing differences in absolute magnitude and remaining systematic effects.

4.3. Knowledge graph construction

The knowledge graph used in this experiment was constructed from a curated set of technical documents on machining Ti-6Al-4V (Table 3), spanning cutting temperature and surface roughness optimization, tool wear mechanisms, sustainable cutting fluids, and machining deformation control in near-net-shaped jet engine blades. Together, these sources cover process parameters (cutting speed, feed, depth of cut, rake angle), performance measures (surface roughness, tool wear, tool life, cutting temperature), coolant strategies (including sustainable cutting fluids), and fixturing and deformation behavior in adaptive machining. The integrated KG contains 2701 triples over 3401 unique entities and 1151 unique relation types.

The KG was constructed automatically using the open-sourced `gpt-oss-120b` model served locally via an OpenAI-compatible HTTP API (LM Studio). Each paper was converted to Markdown and processed by a Python pipeline that splits text into overlapping windows (up to 1000 characters with a 500-character overlap). For each window, a prompt marks a central extraction region (CHUNK_MAIN) and surrounding context windows, and specifies a strict five-field triple format: ENTITY_1 <TAB> RELATIONSHIP_TYPE <TAB> ENTITY_2 <TAB> "RELATIONSHIP_DESCRIPTION" <TAB> FIGURE_REFERENCE. Triples are extracted from CHUNK_MAIN only; surrounding context is used solely to resolve references to tables/figures. A post-processing stage enforces TSV consistency, repairs minor formatting errors, and propagates table references to related cell triples using schema triples. Triples are written per-document and then aggregated into the global KG used in subsequent experiments.

4.4. Evaluation protocols

4.4.1. Tool-use depth protocol (L1/L2/L3)

To quantify tool-orchestration reliability on rotor blade analytics, we constructed 75 natural-language questions, comprising 25 questions at each of three tool-use depths where **L1** requires a single correct tool invocation; **L2** requires exactly two dependent tool calls with correct intermediate-result propagation; and **L3** requires three or more calls, stressing longer-horizon planning, state tracking, and dependency

Table 3
Overview of source documents and their corresponding knowledge graph sizes.

Document	Description	Knowledge Graph Size
Optimization of cutting temperature and surface roughness in CNC turning of Ti-6Al-4V alloy using response surface methodology [44]	Introduced Response Surface Methodology to optimize cutting speed, feed rate, and depth of cut in dry CNC turning of Ti-6Al-4V alloy, aiming to minimize tool-chip interface temperature and surface roughness.	# of triples: 827 # of unique entities: 978 # of unique relations: 324
Evaluation of tool wears mechanism considering machining parameters and performance parameters for titanium alloy in turning operation on CNC [45]	Experimentally studies CNC turning of Ti-6Al-4V to evaluate how machining parameters (cutting speed, feed, depth of cut, and rake angle) affect tool wear, tool life, and surface roughness.	# of triples: 673 # of unique entities: 802 # of unique relations: 249
Sustainable green cutting fluid for interpreting optimization of process variables while machining on various CNC manufacturing systems—an experimental approach for exploring [46]	Compares several eco-friendly cutting fluids and finds that aerosol-mist neem oil is the most sustainable and effective choice for machining Ti-6Al-4V on CNC lathe and turn-mill systems, improving surface finish and reducing temperatures, especially in 4-axis turn-milling.	# of triples: 541 # of unique entities: 707 # of unique relations: 239
Analysis of machining deformation for adaptive CNC machining technology of near-net-shaped jet engine blade [47]	Develops and validates a rigid–flexible coupling fixture using PEEK-GF30 multi-point supports and an empirical cutting force model to analyze and control machining deformation of near-net-shaped TC4 jet engine blades in adaptive CNC machining.	# of triples: 660 # of unique entities: 914 # of unique relations: 339

management. For each question, we compare the agent's called-tool sequence (tools, order, and arguments) against a reference implementation derived from expert-designed CCAT scripts. It is important to note that the CCAT scripts were used only to define the deterministic tool library and the reference tool sequences used for scoring. The benchmark questions were written as natural-language engineering requests and did not expose script names, function names, or reference implementations to the model. Therefore, this benchmark evaluates controlled in-system tool orchestration under a fixed tool schema. L1 questions primarily evaluate direct tool grounding, whereas L2 and L3 evaluate increasingly compositional tool selection, dependency ordering, and intermediate-state propagation. We report tool-selection pass rate (%) as the fraction of queries where the required tools are invoked with valid arguments and dependency ordering.

4.4.2. Critic agent evaluation

To quantify the contribution of the Critic Agent within MAKa, we conducted a paired controlled evaluation in which each query was executed twice under matched conditions using the same base model, question prompt, initialization, and routing context: one run with critic-mediated verification and one run without critic mediation. In the critic-enabled condition, the execution workflow permits the Critic Agent to inspect the selected tools, identify missing dependencies or invalid tool sequences, and request additional routing or analysis actions before termination. In the no-critic condition, the outputs of the routing and task agents proceed directly to finalization without an explicit verification or repair stage. This paired design isolates the marginal contribution of critic-mediated verification while controlling for differences in model capability, prompt formulation, and query difficulty.

To evaluate robustness under controlled orchestration faults, required tool hints are deterministically removed from the initial routing context at dropout rates of 0%, 10%, 30%, and 50%. The same degraded routing context is used for the critic-enabled and no-critic runs within each pair. A 0% dropout rate represents the undegraded condition, whereas the higher dropout rates progressively increase the likelihood that the initial execution plan omits one or more required tools or dependencies.

Each execution is evaluated against the query-specific reference tool set using tool-selection precision, recall, and F1-score. In addition to the mean F1-score at each dropout rate, three paired recovery metrics are reported. The improved-tool rate is the percentage of paired queries for which the critic-enabled F1-score exceeds the corresponding no-critic F1-score. The reduced-missing rate is the percentage of paired queries for which critic mediation decreases the number of missing required tools. The full recovery rate is calculated over degraded cases in which

the no-critic execution omits at least one required tool and measures the percentage for which the critic-enabled execution recovers all missing required tools. Together, these metrics distinguish general improvements in tool-selection quality from the Critic Agent's specific ability to detect and repair incomplete tool-execution chains.

4.4.3. Knowledge agent evaluation protocol

Knowledge agent performance was evaluated in two linked phases using paired KG versus no-KG runs with the same base model family and identical question prompts: an open-ended quantitative phase and a multiple-choice phase derived from the same underlying items. The open-ended bank of 75 questions was programmatically constructed from the titanium machining sources and contains quantitative decision-style questions that require selecting operating points under thermal, roughness, wear, and throughput constraints, together with boundary and extrapolation checks, fixture-force reasoning, and speed-window compliance. Each item includes explicit ground-truth numeric targets, required terms, and scoring tolerances. The multiple-choice bank was generated from the same items by converting each ground-truth answer into one correct option and creating deterministic distractors via seeded numeric perturbations that remain outside acceptance tolerances. Performance reporting focuses on average numerical score versus average time per question for open-ended responses, and accuracy versus average time per question for multiple-choice responses, where time per question quantifies the inference-time overhead of KG retrieval relative to its corresponding gain in answer quality. This evaluation should be interpreted as a closed-corpus grounding study. The questions are source-conditioned on the same curated titanium machining corpus used to construct the KG, and therefore the experiment does not test open-domain RAG generalization. Instead, it tests whether structured, provenance-linked KG retrieval improves quantitative grounding and reduces unsupported numeric claims within a fixed technical corpus when compared against the same model family without KG retrieval.

4.4.4. Expert evaluation protocol

To evaluate the practical utility and interpretability of the generated recommendations, CNC machining experts at CCAT reviewed 250 MAKa outputs. Each of the selected LLM models for the framework were applied to the same set of 50 randomly sampled benchmark questions spanning the different tool-use depths, resulting in 250 total outputs. For each output, the reviewers examined the original question, final response, invoked tools, provenance trace, and recorded Critic actions. Reviewers determined whether the response could be accepted as-is, required minor editing, required major editing, or should be rejected. They also rated Correctness, Decomposition Clarity, Provenance

Table 4

Model specifications and requirements.

Model	# of Parameters (billions)	VRAM Required (GB)	Quantization
GPT-OSS-20b	20	12.1	MXFP4
Granite-4H-Tiny	7	4.2	Q4
Qwen3-VL-30b	30	19.6	Q4
Qwen3-VL-8b	8	6.2	Q4
Qwen3-VL-4b	4	3.3	Q4

Trust, Critic Helpfulness, and Overall Interpretability on five-point scales, with higher scores indicating stronger performance. Correctness measured whether the reported values matched the inspection data and whether the conclusions followed from those values. Decomposition Clarity assessed how clearly the response distinguished related factors such as tool wear, deflection, compensation, pathing deviation, surface deviation, and variability. Provenance Trust measured confidence in the evidence trail connecting source files, tool calls, calculations, reported values, and final conclusions. Critic Helpfulness assessed how clearly the Critic’s validation, retries, corrections, or routing decisions contributed to confidence in the final response. Overall Interpretability captured whether the response was understandable, justified, transparent, and easy to audit. Higher scores indicated stronger performance for all rating dimensions.

5. Experimental results

MAKA was implemented in Python with a PostgreSQL backend used to store and query the machining knowledge graph. Results are reported in the same order as the experimental design: (i) tool-orchestration reliability, (ii) Critic Agent ablation, (iii) KG-grounding ablation, (iv) the end-to-end rotor blade case study and digital-twin compensation analysis, and (v) formal expert evaluation of the resulting recommendations.

5.1. Tool-orchestration reliability

We first evaluate whether the agents can correctly select and sequence rotor blade analysis tools needed to compute deviation components and compensation values from inspection data. Queries require composing tools such as `rb_compute_surface_dev`, `rb_compute_pathing_dev`, `rb_compute_wear_drift`, `rb_compute_process_variability`, and `rb_compute_residual_systematic` under increasing tool-chain depth (L1–L3). Tool-use pass rate (%) is computed against expert reference scripts, counting a query as correct only when required tools are called with valid arguments and in dependency-consistent order.

Fig. 4 compares MAKa with two strengthened single-agent baselines across three levels of tool-use complexity, with 25 questions evaluated at each level. The evaluation includes GPT-OSS-20b, Granite-4H-Tiny, Qwen3-VL-30b, Qwen3-VL-8b, and Qwen3-VL-4b. The ReAct-style baseline iteratively reasons over the query, selects a tool, observes the returned output, and continues until producing a final response. The function-calling baseline, denoted as “Single” in Fig. 4, invokes tools through the same structured function schemas used by MAKa. All configurations use the same natural-language questions, deterministic tool library, tool descriptions, argument schemas, scoring criteria, and underlying base model. The comparison therefore isolates the effects of the orchestration architecture rather than differences in model capability, prompting resources, or tool access. Table 4 summarizes the parameter count, required VRAM, and quantization of the evaluated models.

For L1 questions requiring only one tool invocation, all three configurations achieve similarly high pass rates. The ReAct-style baseline achieves a pass rate of 100% for all five models, while the function-calling baseline ranges from 96% to 100% and MAKa also ranges from

96% to 100%. MAKa matches the strongest single-agent result for four models and is four percentage points lower for Granite-4H-Tiny. The mean L1 pass rates are 100% for ReAct, 98.4% for function calling, and 99.2% for MAKa. The small uplifts or absence of improvement at this level indicate that L1 questions primarily evaluate direct tool recognition rather than multi-step orchestration. Because these tasks require only one valid tool call and involve little or no dependency ordering, intermediate-result propagation, or state management, a well-prompted single agent can complete most requests reliably. Consequently, the architectural advantages of MAKa are less visible at L1, and the near-equivalent performance provides evidence that its later gains are not simply caused by stronger prompting or tool access.

The performance differences become substantially larger for L2 questions, which require two dependent tool calls and correct propagation of intermediate results. MAKa achieves a 100% pass rate for all five models. In comparison, the ReAct-style baseline ranges from 48% to 92%, while the function-calling baseline ranges from 48% to 96%. Relative to the better-performing single-agent baseline for each model, MAKa improves the pass rate by 48, 44, 8, 4, and 4 percentage points for GPT-OSS-20b, Granite-4H-Tiny, Qwen3-VL-30b, Qwen3-VL-8b, and Qwen3-VL-4b, respectively. The average pass rates across models are 68.8% for ReAct, 72.0% for function calling, and 100% for MAKa. Unlike L1, these questions require the system to preserve the output of an initial tool call, pass it correctly to a dependent operation, and maintain the proper execution order. The larger performance gap therefore reflects MAKa’s ability to coordinate dependent calculations and verify intermediate steps rather than merely identify an appropriate tool.

The largest separation occurs for L3 questions, which require three or more dependent tool calls, correct dependency ordering, and longer-horizon preservation of intermediate state. MAKa achieves a 100% pass rate for four models and 92% for Granite-4H-Tiny, corresponding to a cross-model average of 98.4%. By contrast, the ReAct-style baseline ranges from 20% to 60%, with an average of 44.0%, and the function-calling baseline ranges from 24% to 68%, with an average of 43.2%. Compared with the stronger single-agent result for each model, MAKa provides improvements of 60, 64, 40, 44, and 32 percentage points for GPT-OSS-20b, Granite-4H-Tiny, Qwen3-VL-30b, Qwen3-VL-8b, and Qwen3-VL-4b, respectively. The average improvement over the better-performing single-agent baseline is therefore 48.0 percentage points at L3. These results show that the value of the multi-agent architecture increases with orchestration depth, since longer workflows create more opportunities for missing dependencies, incorrect sequencing, invalid argument propagation, and loss of intermediate state.

The two single-agent baselines do not exhibit a consistent advantage over one another. ReAct performs better for some models and complexity combinations, whereas structured function calling performs better for others. Both approaches nevertheless degrade sharply as tool-use depth increases. Their common failure modes include omitted dependencies, incorrect tool ordering, invalid argument propagation, and loss of intermediate results across calls. In contrast, MAKa maintains consistently high performance through explicit task decomposition, controlled agent responsibilities, structured intermediate outputs, and critic-mediated verification. The near-equivalent L1 results, followed by increasingly large L2 and L3 improvements, indicate that MAKa’s gains are not attributable simply to stronger prompting or structured function calling. Instead, the primary advantage arises from reliable coordination and verification of compositional tool-execution workflows.

Table 5 shows an interaction in which the user requests compensation values over a selected part window. MAKa loads the aggregated inspection file, computes required pairwise statistics, and returns compensation values at each inspection pair key. This example is representative of the tool-use setting evaluated in Fig. 4: the agent must map a high-level request onto the correct rotor blade tools, respect dependencies among intermediate computations, and return numerically valid outputs in the expected format.

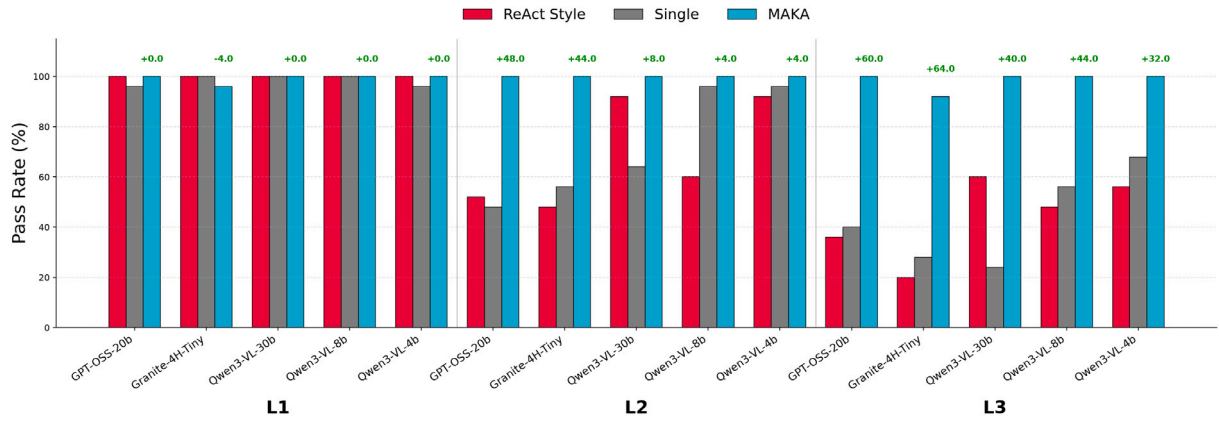


Fig. 4. Comparison between MAKa, ReAct Style, and Single function calling language model in terms of pass rate (%) for correct tool usage.

Table 5

Example of querying tool compensation using the MAKa framework.

Execution (CLI):

Central Agent --- What would you like to do?

> load './Inspection_Aggregated.csv' and give me compensation for parts 4 to 16

Model analysis:

The file Inspection_Aggregated.csv is loaded, pairwise statistics are computed, and tool compensation values are returned for parts 4–16. Here, Trc denotes tool radius compensation and Tlc denotes tool length compensation:

Pair Key	Trc	Tlc
2+17	0.001164	0.002497
3+18	0.001030	0.002209
4+19	0.001063	0.002280
5+20	0.001104	0.002367
6+21	0.001138	0.002440
7+22	0.001253	0.002686
8+23	0.001466	0.003144
9+24	0.001290	0.002766
10+25	0.001356	0.002909
11+26	0.001513	0.003245
12+27	0.001475	0.003163
13+28	0.001556	0.003337
14+29	0.001670	0.003581
15+30	0.001528	0.003278
16+31	0.001620	0.003474

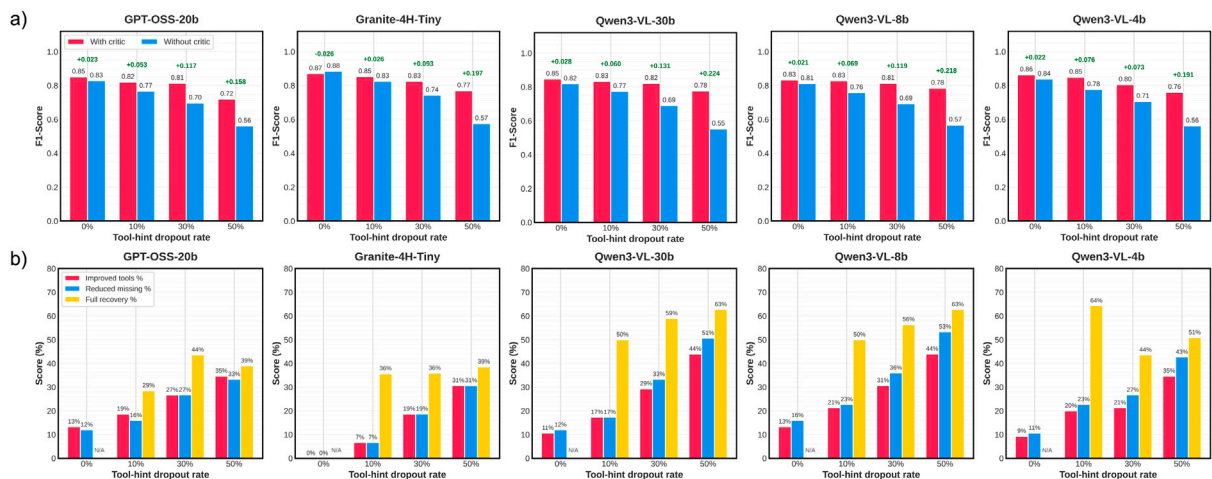


Fig. 5. Critic agent ablation across five base models comparing critic-enabled and no-critic execution. (a) Average tool-selection F1-score for each model with critic versus without critic across tool-hint dropout rates. (b) Critic-value metrics from paired runs: improved tools (%), reduced missing required tools (%), and full recovery (%) under various degraded routing conditions.

5.2. Critic agent ablation

Fig. 5 reports the critic-agent evaluation described in Section 4.4.2. Across GPT-OSS-20b, Granite-4H-Tiny, Qwen3-VL-30b, Qwen3-VL-8b, and Qwen3-VL-4b, each dropout condition contains 75 paired queries per model, corresponding to 375 paired comparisons per dropout rate and 1500 paired comparisons across the complete sweep. As shown in Fig. 5(a), critic-enabled and no-critic execution perform similarly at 0% dropout. The paired F1-score differences range from -0.026 for Granite-4H-Tiny to $+0.028$ for Qwen3-VL-30b, while the remaining models show only small positive differences. The average F1-scores are approximately 0.852 with the critic and 0.836 without the critic. These results indicate that the critic provides limited additional benefit when the initial routing context already contains the required tool information. This finding is important because it shows that the critic does not artificially inflate performance under normal conditions and that most undegraded queries can be completed directly by the routing and task agents.

The benefit of critic mediation becomes increasingly visible as required tool hints are removed. At 10% dropout, the critic improves F1-score by approximately 0.026 to 0.076 across the five models, with a mean paired improvement of approximately 0.057. At 30% dropout, the improvement increases to between 0.073 and 0.131, with a mean of approximately 0.107. At 50% dropout, the critic produces improvements of 0.158 for GPT-OSS-20b, 0.197 for Granite-4H-Tiny, 0.224 for Qwen3-VL-30b, 0.218 for Qwen3-VL-8b, and 0.191 for Qwen3-VL-4b. The mean paired improvement at this level is approximately 0.198. Correspondingly, the cross-model mean F1-score at 50% dropout is approximately 0.762 with critic mediation and 0.562 without it. The increasing separation across dropout rates indicates that the critic's primary value lies in robustness and repair rather than in improving already-correct undegraded executions.

Fig. 5(b) further explains the mechanism behind these improvements. At 0% dropout, the improved-tool and reduced-missing rates remain low, ranging from 0% to 13% and from 0% to 16%, respectively. Full recovery is not reported at this level because no tool hints are intentionally removed. As the dropout rate increases, the critic more frequently improves the selected tool set and restores omitted dependencies. At 30% dropout, the improved-tool rate ranges from 19% to 31%, the reduced-missing rate ranges from 19% to 36%, and the full recovery rate ranges from 36% to 59%. At 50% dropout, the improved-tool rate has increased ranges between 31% to 44%, the reduced-missing rate has increased ranges from 31% to 53%, and full recovery achieved increased ranges from 39% to 63% of eligible degraded cases.

The strongest recovery behavior at 50% dropout is observed for Qwen3-VL-30b and Qwen3-VL-8b. Qwen3-VL-30b improves the selected tools in 44% of paired queries, reduces missing tools in 51%, and fully recovers 63% of eligible degraded cases. Qwen3-VL-8b achieves corresponding rates of 44%, 53%, and 63%. Even for GPT-OSS-20b, which shows the smallest F1-score improvement at 50% dropout, critic mediation improves the tool set in 35% of cases, reduces missing tools in 33%, and fully recovers 39% of eligible degraded executions. Taken together, the 0% condition and dropout sweep distinguish the critic's normal-operation effect from its fault-recovery effect. Under undegraded routing, the critic produces only marginal changes because the required tool information is already available. Under controlled degradation, however, it increasingly detects omitted dependencies, requests missing tool calls, and repairs incomplete execution chains. The ablation therefore characterizes the Critic Agent primarily as a reliability and recovery layer that prevents routing faults from propagating into the final response, rather than as a mechanism that uniformly increases performance on otherwise correct executions.

5.3. Knowledge graph grounding

Fig. 6 summarizes paired KG versus no-KG evaluation across the same five base models using 75 questions per model in each task format. In Fig. 6(a), every model moves from the no-KG point to a higher numeric score with KG enabled, increasing the cross-model mean from 0.3462 to 0.4736 ($+0.1274$), while average time per question increases from 0.7305 s to 2.6496 s ($+1.9191$ s). The largest open-ended gains occur for GPT-OSS-20b ($+0.2418$), Qwen3-VL-4b ($+0.2262$), and Qwen3-VL-8b ($+0.1829$). Granite-4H-Tiny improves moderately ($+0.0911$), while Qwen3-VL-30b positive gains ($+0.0048$), consistent with stronger no-KG baselines and diminishing marginal returns when parametric knowledge is already strong. In Fig. 6(b), the same trend appears for multiple-choice accuracy: mean accuracy rises from 0.4733 to 0.5733 ($+0.1000$), with average latency increasing from 0.2413 s to 1.6002 s ($+1.3588$ s). The largest MCQ uplifts are from Qwen3-VL-4b and Qwen3-VL-8b ($+0.1600$ each), and GPT-OSS-20b both improves ($+0.1200$) and achieves the highest with-KG accuracy (0.7067). Overall, KG grounding provides reliable average gains in quantitative correctness and answer selection, with the largest benefits concentrated in models that most need external numeric evidence integration, and with a stable and interpretable latency tradeoff.

5.4. End-to-end rotor blade case study

We next present an end-to-end case study that follows a complete MAKa loop from digital-twin evidence and inspection measurements to a staged compensation plan that is verified in a high-fidelity digital twin prior to any shop-floor deployment. The intent is not to claim that any single simulation output numerically reproduces the scan data point by point. Instead, this case study shows that MAKa can extract directional guidance from physics-based simulation, reconcile that guidance with measured deviation structure and its evolution across parts, and translate the combined evidence into compensation actions that are auditable and bounded by uncertainty.

In this case study, MAKa receives three complementary inputs. ThirdWave provides deflection fields that indicate where the blade is most compliant and how sensitive that compliance is to changes in spindle speed and feed. The inspection scans provide the ground-truth geometric error signal relative to the nominal model at paired locations across the pressure and suction sides, revealing where systematic shape bias concentrates on the blade. Multi-part statistics derived from the inspection history summarize systematic evolution and dispersion at each inspection pair. MAKa uses these as proxies for wear-driven changes and process instability. These signals serve distinct roles. Simulation supplies a physics-based sensitivity map. Inspection supplies the realized error topology. Drift and variability provide a confidence and risk envelope that governs how aggressively compensation should be applied.

Fig. 7 shows ThirdWave deflection simulations for two representative parameter sets selected to contrast an original Siemens NX recommendation with an alternative derived from expert machinist practice. Tool geometry, engagement conditions, and fixturing are held constant so the differences in the predicted fields are primarily attributable to spindle speed and feed. Beyond identifying a single minimum-displacement value, the simulations act as a screening mechanism that localizes deformation hot spots along the blade and establishes which parameter regimes are likely to reduce compliance-driven error. In thin-walled blade regions that behave like a cantilever, reductions in feed per tooth typically reduce cutting force peaks and can suppress elastic deflection, while increases in cutting speed can introduce thermal and wear considerations that must be evaluated in parallel. For this reason, MAKa treats the best performing parameter set in the simulation as a candidate operating point that must be contextualized against stability, tool life, coolant strategy, and machine capability rather than as a universally optimal choice.

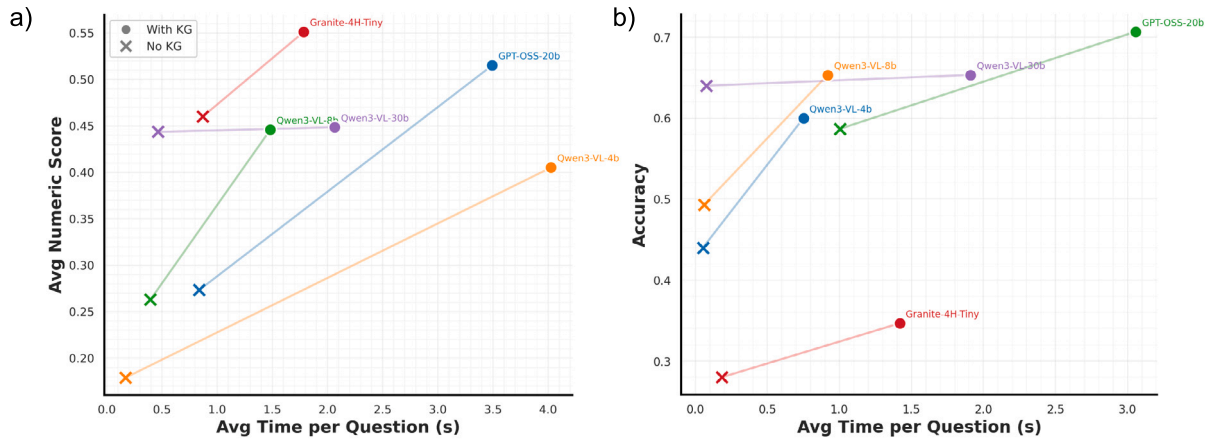


Fig. 6. Knowledge graph evaluation across five base models using paired runs with and without KG retrieval. (a) Open-ended quantitative performance: average numerical score versus average time per question. (b) Multiple-choice performance: accuracy versus average time per question. Connected pairs show the shift from no-KG to with-KG, illustrating quality gains and inference-time overhead.

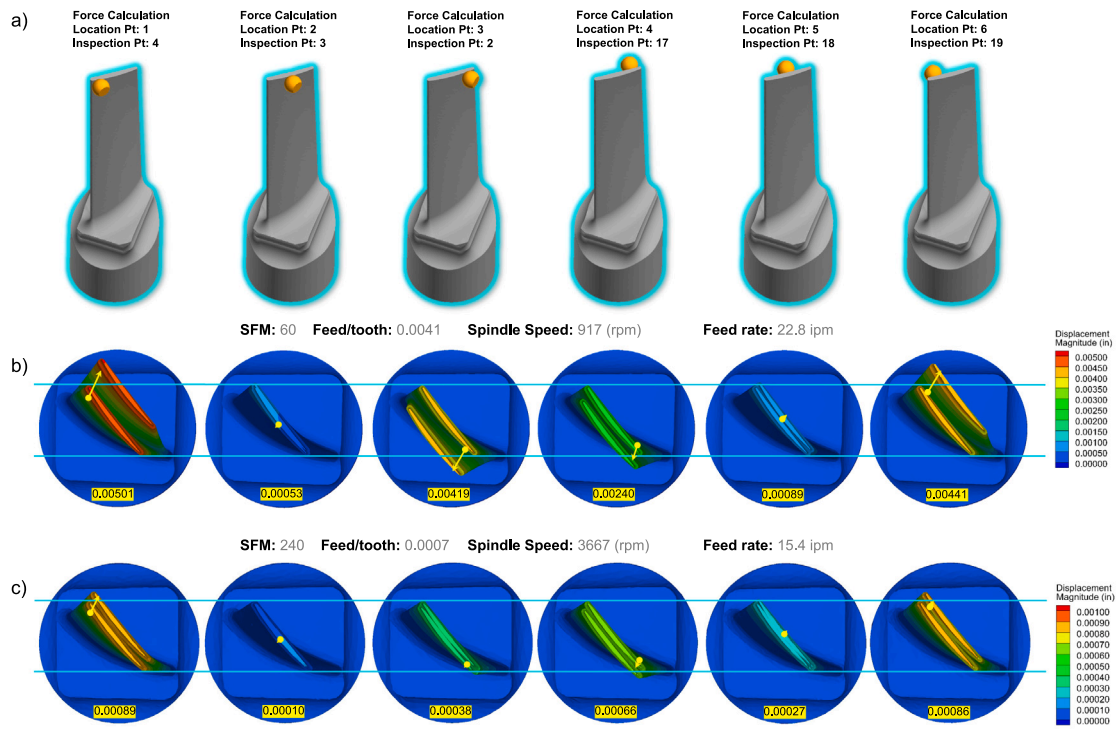


Fig. 7. Deflection results. (a) The location where force calculations are applied. (b) Blade deflection for a spindle speed of 917 rpm and feed rate of 22.8 ipm. (c) Blade deflection for a spindle speed of 3667 rpm and feed rate of 15.4 ipm.

Table 6 illustrates how MAKa converts the deflection visualization into an actionable recommendation in response to a natural language request. The agent identifies the minimum-displacement candidate across the shown parameter grid and then enumerates deployment considerations that are consistent with machining constraints. This step is important for closing the loop between physics and process engineering. The twin narrows the candidate space toward lower deformation sensitivity, and the KG-grounded considerations make explicit the failure modes that can negate those gains in practice, including accelerated wear at higher SFM, chatter margins, surface integrity effects, and coolant demands. MAKa therefore does not stop at ranking parameter sets. It surfaces the trade-offs that an engineer would normally check before committing to an update.

While deflection screening addresses deformation sensitivity, inspection history is required to separate stable, compensable bias from evolving behavior that should be handled conservatively. Table 7 shows how MAKa analyzes a wear-percentage envelope derived from inspection deviations over a selected part window. The key analytical signal is the envelope shape across inspection pairs. A narrow separation between minimum and maximum curves indicates that evolution is relatively consistent and can be treated as bounded drift. Widening separation indicates growing instability or mixed contributors that are not captured by a single drift trajectory. Envelope widening is a warning flag. It reduces confidence in applying aggressive offsets at those locations and instead favors bounded compensation, increased inspection frequency, or earlier tool-change thresholds. This drift and

Table 6
Example of analyzing simulation results using the MAKa framework.

Execution (CLI) Central Agent - What would you like to do > analyze the simulation results and give the process parameters with the lowest deviation and then list any potential considerations when using these parameters './data/Simulation_results.png'	
Model analysis Final answer Based on the simulation results shown in the image, the process parameters with the lowest deviation, meaning the smallest displacement magnitude, are as follows. Top Row (SFM 60) • Feed per tooth 0.0041 • Spindle Speed 917 rpm • Feed Rate 22.8 ipm • Lowest Displacement 0.00089 (seen in the 5th image from the left) Bottom Row (SFM 240) • Feed per tooth 0.0007 • Spindle Speed 3667 rpm • Feed Rate 15.4 ipm • Lowest Displacement 0.00010 (seen in the 2nd image from the left) Conclusion The parameters in the bottom row (SFM 240) , specifically Feed per tooth 0.0007, Spindle Speed 3667 rpm, Feed Rate 15.4 ipm , result in the lowest overall displacement magnitude (0.00010), indicating the most stable simulation outcome. Potential considerations when using these parameters 1. Tool wear and life. Operating at higher SFM and spindle speed increases heat and can accelerate wear. 2. Machine tool capability. Ensure the machine can sustain the spindle speed and feed without vibration. 3. Material removal rate. Higher SFM changes MRR. Manage chip load and cutting forces accordingly. 4. Surface finish. Lower deflection does not guarantee better finish. Verify with roughness checks. 5. Coolant requirements. Higher cutting speeds may require more aggressive cooling. 6. Process stability. Higher-speed regimes may be more chatter-sensitive if the system is not tuned.	

variability analysis therefore provides the gating logic that prevents MAKa from over-correcting regions where the process is no longer stable.

In addition to producing numeric recommendations, MAKa is designed to provide explanations that a process engineer can audit. Table 8 illustrates this interpretability pathway. The agent takes a deflection trend summary and retrieves supporting causal relationships from the machining KG to produce a mechanism-level narrative for why deformation occurs. The retrieved factors include cutting deformation as the primary mechanism, sensitivity to cutting-force dynamics, low stiffness due to thin-walled cantilever geometry, stress concentration near clamping heads, and interactions between speed and wear. These factors justify why parameter tuning is a plausible corrective lever when deformation dominates, and they also highlight conditions under which the same parameter changes could increase risk, including higher wear rates or reduced stability margin. This makes the resulting compensation plan less of a black box because the system attaches physical and empirical rationale that explains why a given corrective action is being proposed.

Taken together, the deflection screening, drift and variability bounding, and KG-grounded explanations provide the evidence needed to justify staged compensation rather than a single monolithic correction. Toolpath updates are most effective when the observed error is spatially consistent and aligns with kinematic or path-tracking signatures.

Parameter updates are most effective when the digital twin indicates high compliance sensitivity and the inspection error topology matches predicted deformation hot spots. Wear-aware bounding becomes critical when drift and variability indicate evolving behavior that would otherwise cause over-correction. This staged structure is reflected in the subsequent compensation verification (Fig. 8), where successive steps reduce predicted deviation magnitude from the 10^{-2} in range toward the 10^{-3} in level by targeting distinct contributors in sequence, while using drift and variability as a safeguard against applying aggressive offsets in unstable regions.

5.5. Expert evaluation

Lastly, we assessed whether the agentic workflow produced technically useful and interpretable inspection analyses described in Section 4.4.4. The sampled questions covered numerical inspection analysis, operator-facing recommendations, multi-factor decomposition, provenance-sensitive calculations, and knowledge-graph-supported interpretation. For each output, CNC machining experts at CCAT reviewed the original question, final response, invoked tools, provenance trace, and recorded Critic actions.

As reported in Table 9, 71.6% of the 250 outputs were accepted as-is, while 94.8% required no more than minor editing. Correctness

Table 7
Example of analyzing tool wear results using the MAKa framework.

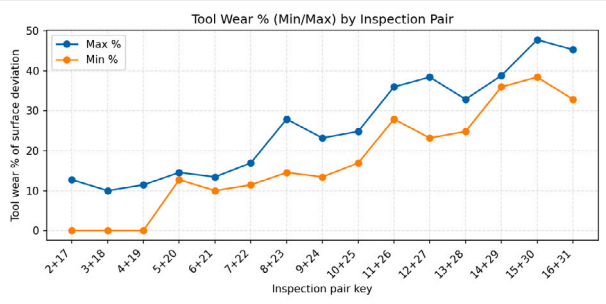
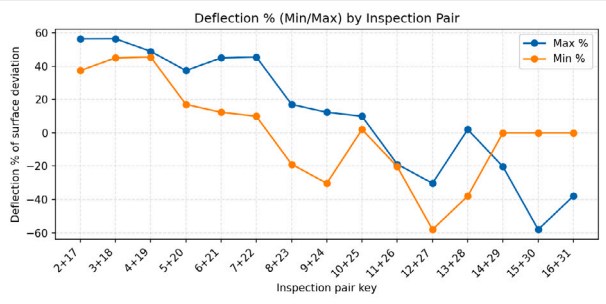
Execution (CLI) Central Agent - What would you like to do > analyze the tool wear percentage ranges and explain what can be shown by the plot (parts 4 to 20)

Model analysis Wear percentage envelope metrics are computed and displayed for the selected part window. All values are calculated and non-NULL.
Final answer Based on the plot titled "Tool Wear % (Min/Max) by Inspection Pair", here is what can be inferred. 1. What is being summarized - The vertical axis represents a normalized wear indicator expressed as a percentage contribution relative to the local deviation baseline used in the analysis. - The horizontal axis lists inspection pair keys (for example "2+17" and "3+18") that correspond to fixed locations on the blade where multi-part deviations were tracked.
2. How the two series should be interpreted under the drift and variability methodology - The two curves can be interpreted as an envelope over the evaluated part window that summarizes systematic evolution across parts. - The separation between the curves provides an empirical indication of instability because a widening envelope implies that deviation evolution is not explained by a single tightly bounded drift trajectory.
3. Key trends visible in the plot Drift envelope growth. The envelope generally increases moving from left to right, indicating larger systematic evolution at later inspection locations. Instability signature. The gap between the envelope bounds becomes larger for later pairs, consistent with increasing process instability or unmodeled variation. Peak location. The largest envelope values occur near inspection pair "15+30", suggesting a wear-sensitive region for tighter monitoring or conservative offset bounds.
Summary. The plot summarizes a drift-based wear envelope across inspection pairs and indicates where envelope widening suggests instability, supporting maintenance thresholds and conservative compensation decisions.

Table 8
Example of using a knowledge graph to explain rotor blade deflection.

Execution (CLI) Central Agent - What would you like to do > causes of this deflection for the rotor blade

Model analysis (KG-enhanced) Based on the retrieved knowledge and the analysis summary, the primary cause of rotor blade deflection is cutting deformation , which directly leads to the final deformation of the blade. This deformation is influenced by several factors. 1. Cutting force frequency. Deflection increases significantly when the cutting force frequency exceeds 1500 Hz. At frequencies between 800 and 1500 Hz, the deformation is relatively small. 2. Blade structure. The blade's complex structure, including its thin-walled body, free surface, and large curvature changes, makes it susceptible to deformation. Its stiffness is low due to its cantilever beam structure. 3. Stress distribution. Stress is higher near the clamping heads, which can lead to localized deformation or stress concentration. 4. Rotational speed and tool wear. While increasing rotational speed slightly reduces cutting forces, it can also increase tool wear, which may indirectly affect blade deformation.
The observed trends in the deflection chart, such as high deflection at the start (for example pair 2+17) and more negative deflection later (for example pair 12+27), likely reflect variations in cutting conditions such as force frequency, spindle speed, or tool wear across inspection pairs.

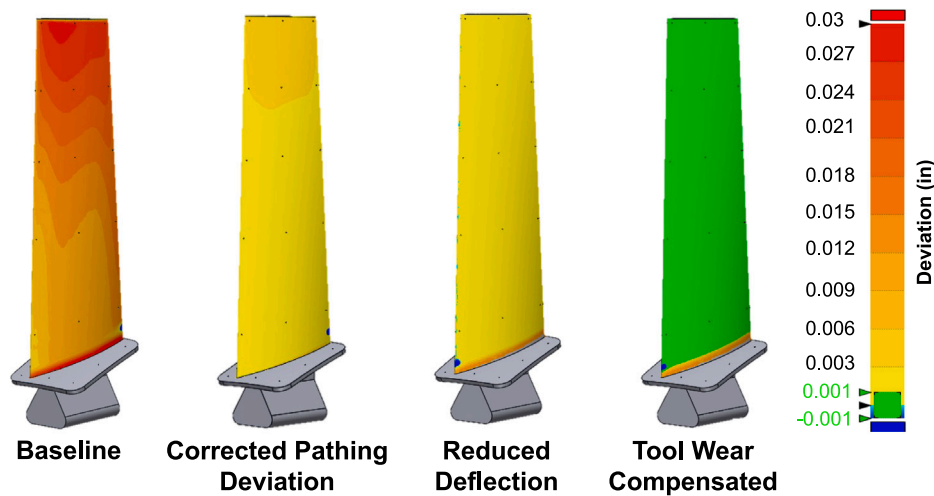


Fig. 8. Successive MAKAs-driven compensations for toolpath adjustment, deflection-focused parameter tuning, and tool-wear-aware updates, verified in the high-fidelity digital twin.

received a cross-model mean score of 4.58 out of 5, with 94.4% of responses rated 4 or 5, and only 13 outputs, or 5.2%, required major revision or rejection, including two cases in which no answer was generated. Model-level acceptance ranged from 66.0% for Qwen3-VL-30B to 74.0% for GPT-OSS-20B and Granite-4H-Tiny. Provenance Trust was the strongest interpretability dimension, with a cross-model mean of 4.56, model-level scores ranging from 4.44 to 4.68, and 90.0% of responses rated 4 or 5. Decomposition Clarity and Overall Interpretability received mean scores of 3.70 and 3.72, respectively, with approximately three-quarters of responses rated 4 or 5. These results indicate that MAKAs generally produced technically grounded and practically usable analyses whose calculations, evidence trails, and decomposition of inspection factors were understandable and trusted by the experts.

Critic Helpfulness received a cross-model mean score of 2.96, with model-level scores ranging from 2.90 to 3.14 and 4.0% of responses rated 4 or 5. This reflects the Critic's primary role as an internal validation and correction mechanism rather than a standalone user-facing explanation component, since its retries, decisions, and corrections were not always explicitly summarized in the final response. The free-text evaluations showed experts most trusted outputs that agreed with known reference values, provided complete pair-level results, included recomputed consistency checks, distinguished dataset-wide relationships from pair-specific exceptions, and separated wear, deflection, compensation, and pathing contributions without introducing unsupported causal interpretations. Confidence was reduced by unsupported pair values, threshold or comparison-direction errors, incorrect interpretation of global and local patterns, overconfident physical explanations, unrelated knowledge graph context, and insufficient access to source values for direct verification. These findings motivate stronger output validation, more selective use of retrieved context, and clearer human-facing summaries of critic interventions. Overall, the expert evaluation demonstrates MAKAs can produce technically grounded, traceable, and practically usable CNC inspection analyses, while also identifying specific improvements needed to make its internal verification process more transparent to engineers and operators.

6. Conclusion

This paper introduces the multi-agent knowledge analysis (MAKA) framework as a physics-grounded, LLM-driven decision-support layer for CNC manufacturing, in which a Central Agent, Analysis Agent, Knowledge Agent, and Critic Agent translate natural-language intent

into verifiable tool use, evidence-linked retrieval, and risk-aware recommendations suitable for human approval prior to downstream CAM implementation. The architectural separation of intent routing, deterministic computation, knowledge retrieval, and verification is designed to support human–AI symbiosis in high-stakes manufacturing by reducing the cognitive burden of multimodal data synthesis while preserving auditability, bounded action, and provenance.

In the rotor blade case study, MAKAs reliably orchestrated domain tools to align scan-based inspection deviations and virtual-machining outputs at common inspection keys and to construct a physically interpretable decomposition consistent with the updated methodology. Specifically, pathing deviation was obtained from Run MyVirtual Machine projections, systematic evolution across parts was captured through a drift-based proxy estimated from multi-part inspection sequences, and the remaining baseline systematic term was treated as a combination of repeatable effects that may include compliance, stiffness-related behavior, registration bias, and other unmodeled sources rather than being uniquely attributed to structural deflection. A variability proxy was also computed to characterize instability and support conservative recommendations when confidence was insufficient. This tool-grounded decomposition reproduces the structure of expert workflows while making its assumptions explicit and inspectable by both the human operator and the Critic Agent verification gate.

Beyond the case study demonstration, the controlled evaluations show that MAKAs's multi-agent decomposition becomes most valuable when tasks require compositional tool use. The strengthened ReAct-style and structured function-calling baselines performed competitively on L1 queries requiring only one tool call, where little dependency ordering, intermediate-result propagation, or state management is required. MAKAs's advantage increased substantially for L2 and L3 queries involving dependent tool chains and longer-horizon preservation of intermediate results. Across the 15 model-by-level comparisons spanning L1, L2, and L3, MAKAs achieved a mean, median, and maximum uplift of 22.9, 8.0, and 64.0 percentage points, respectively, compared with the better performing ReAct-style and function-calling baselines. This depth-dependent pattern shows that the architectural benefit arises primarily from reducing invalid argument bindings, incorrect call ordering, omitted dependencies, and intermediate-state loss rather than from improving already straightforward single-tool tasks.

Critic and knowledge graph ablations further clarify how the specialized agents contribute to robustness and grounding. Under ungraded routing, critic-enabled and no-critic executions achieved similar cross-model mean F1-scores of approximately 0.852 and 0.836, indicating that the Critic provides limited additional benefit when the

Table 9

Expert evaluation of MAKa recommendations across the benchmark questions. Acceptance is the percentage of recommendations accepted as-is by domain experts. Rating columns report mean scores on a 1–5 scale, where higher is better.

Model	Acceptance (%)	Correctness	Decomposition Clarity	Provenance Trust	Critic Helpfulness	Overall Interpretability
GPT-OSS-20B	74.0	4.74	3.86	4.62	3.14	3.88
Granite-4H-Tiny	74.0	4.66	3.72	4.68	2.94	3.74
Qwen3-VL-30B	66.0	4.42	3.62	4.44	2.92	3.66
Qwen3-VL-8B	72.0	4.54	3.64	4.54	2.90	3.66
Qwen3-VL-4B	72.0	4.54	3.64	4.54	2.90	3.64
Overall	71.6	4.58	3.70	4.56	2.96	3.72

initial routing context already contains the required tool information. Its contribution became increasingly visible as tool hints were removed. At 50% dropout, critic-enabled execution achieved a cross-model mean F1-score of approximately 0.762 compared with 0.562 without critic mediation, while full recovery was achieved in 39%–63% of eligible degraded cases across the evaluated models. These results characterize the Critic primarily as a reliability and fault-recovery mechanism that prevents incomplete routing decisions from propagating into the final response. Similarly, knowledge graph grounding improved quantitative machining answers, increasing the mean open-ended score from 0.3462 to 0.4736 and mean multiple-choice accuracy from 0.4733 to 0.5733. The associated retrieval latency remained small relative to machining cycle times. Taken together, the results show that multi-agent orchestration provides the structure required for correct tool chaining, the Critic supplies a recovery layer under missing or degraded dependencies, and the Knowledge Agent provides provenance-linked numerical grounding and constraint-aware context.

The formal expert evaluation provided an additional assessment of the practical utility and interpretability of the generated recommendations. Across 250 outputs produced by applying five models to the same 50 sampled benchmark questions, 71.6% were accepted as-is and 94.8% required no more than minor editing. Correctness and Provenance Trust received mean scores of 4.58 and 4.56 out of 5, respectively, while Decomposition Clarity and Overall Interpretability received mean scores of 3.70 and 3.72. Experts expressed the greatest confidence in outputs that agreed with known reference values, exposed complete pair-level calculations, included recomputed consistency checks, and separated computed contributors without introducing unsupported causal interpretations. Critic Helpfulness received a mean score of 2.96, reflecting that its internal retries, validation decisions, and corrections were not always explicitly summarized in the final response. These findings indicate that MAKa generally produced technically grounded, traceable, and practically usable analyses while identifying clearer human-facing summaries of Critic interventions as an important improvement for future deployment.

By coupling the decomposed contributors with geometric compensation relationships under a fixed tool-tilt angle and with physics-based deflection simulation, MAKa generated actionable toolpath and parameter compensation candidates compatible with NX CAM. The effect of successive compensations was evaluated within a high-fidelity digital-twin toolchain, providing a pre-deployment verification baseline for risk-aware human decision-making rather than a claim of physical closed-loop correction. Across the staged compensation sequence, the predicted deviation field progressed from large localized errors on the order of 10^{-2} in to a predominantly near-nominal state within approximately ± 0.001 in over most of the surface. The complete multi-agent reasoning and retrieval loop executed in seconds, which is small relative to aerospace machining operations that typically require minutes to hours. MAKa can be invoked between machining stages to assimilate inspection data or process telemetry and inform subsequent decisions without substantial machine idle time. Furthermore, the results demonstrate that separating LLM-based coordination from deterministic computation, evidence grounding, and verification provides a practical foundation for traceable and risk-aware CNC decision support.

CRediT authorship contribution statement

Danny Hoang: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Data curation, Conceptualization. **Ryan Matthiessen:** Investigation, Data curation. **Christopher Miller:** Investigation, Data curation. **Nasir Mannan:** Visualization, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Ruby ElKharchouty:** Investigation, Formal analysis. **David Gorsich:** Resources, Funding acquisition. **Matthew P. Castanier:** Writing – review & editing, Resources, Funding acquisition. **Farhad Imani:** Writing – review & editing, Resources, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported under Cooperative Agreement W56HZV-21-2-0001 with the U.S. Army DEVCOM Ground Vehicle Systems Center (GVSC), through the Virtual Prototyping of Autonomy Enabled Ground Systems (VIPR-GS) program; the National Science Foundation (grant no. 2434519 and 2543603); and the Department of Energy (grant no. DE-EE0011029). The authors gratefully acknowledge the valuable contributions from the Connecticut Center for Advanced Technology (CCAT) for this research. Disclaimer: Reference herein to any specific commercial company, product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the Department of the Army (DoA). The opinions of the authors expressed herein do not necessarily state or reflect those of the United States Government or the DoA and shall not be used for advertising or product endorsement purposes.

Data & code availability

The raw/processed data, code, prompts, question banks, and knowledge graph scripts required to reproduce the above findings cannot be shared at this time due to proprietary restrictions.

References

- [1] Zhao X, Zheng L, Shi M, Zhang X, Zhang Y. Unified modelling for continuous-discrete hybrid adaptive machining CPS of large thin-walled parts. *International Journal of Production Research* 2024;62(9):3099–119.
- [2] Chen L, Xu H, Huang Q, Wang P. An integrated method for compensating and correcting nonlinear error in five-axis machining utilizing cutter contacting point data. *Scientific Reports* 2024;14(1):8763.
- [3] Williams B, Awad RA, Mulkey C, Ciocarlie G, Ismail M, Saleeby K. Securing smart manufacturing: detection of cyber-physical attacks in cnc-based systems. In: 2025 IEEE international symposium on hardware oriented security and trust. IEEE; 2025, p. 428–38.

- [4] Markatos NG, Mousavi A. Manufacturing quality assessment in the industry 4.0 era: a review. *Total Quality Management & Business Excellence* 2023;34(13–14):1655–81.
- [5] Liao J, Huang Z. Data model-based toolpath generation techniques for cnc milling machines. *Frontiers in Mechanical Engineering* 2024;10:1358061.
- [6] Cao Y. Multisensor data fusion-driven digital twins in computer numerical control machining: a review. *Machines* 2025;13(10).
- [7] Mohanta DK, Sahoo B, Mohanty AM. Experimental analysis for optimization of process parameters in machining using coated tools. *Journal of Engineering and Applied Science* 2024;71(1):38.
- [8] Escobar CA, McGovern ME, Morales-Menendez R. Quality 4.0: a review of big data challenges in manufacturing. *Journal of Intelligent Manufacturing* 2021;32(8):2319–34.
- [9] Hoang D, Chen R, Bollas G, Imani F. Hyperdimensional computing for explainable information fusion and multi-task adaptation in advanced manufacturing. *Information Fusion* 2025;103898.
- [10] Chen Z, Hoang D, Piran FJ, Chen R, Imani F. Federated hyperdimensional computing for hierarchical and distributed quality monitoring in smart manufacturing. *Internet of Things* 2025;31:101568.
- [11] Wang X, Bai Q, Gao S, Zhao L, Cheng K. A toolpath planning method for optical freeform surface ultra-precision turning based on nurbs surface curvature. *Machines* 2023;11(11):1017.
- [12] Singh RP, Chen Y. Curvature-adoptive cnc machining of freeform optics via dynamic tangential toolpath optimization. *Materials* 2025;18(22):5153.
- [13] Li E, Zhou J, Yang C, Zhao J, Li Z, Zhang S, Wang M. Part machining deformation prediction based on spatial-temporal correlation learning of geometry and cutting loads. *Journal of Manufacturing Processes* 2023;92:397–411.
- [14] Kukreja A, Pande SS. Optimal toolpath planning strategy prediction using machine learning technique. *Engineering Applications of Artificial Intelligence* 2023;123:106464.
- [15] Garba UH, Wang T, Dong J, Tian Y, Kang J, Tian C. Enhancing propeller design with freeform contours through nurbs interpolation for 2D fabrication, CAD/CAM for 3D production, optimized with taguchi method and artificial neural network. *Results in Engineering* 2025;107069.
- [16] Chen Y, Wang J, Tang Q, Li J. A study on the coarse-to-fine error decomposition and compensation method of free-form surface machining. *Applied Sciences* 2024;14(19):9044.
- [17] Yao J, Zhang L, Huang J. Evaluation of large language model-driven AutoML in data and model management from human-centered perspective. *Frontiers in Artificial Intelligence* 2025;8:1590105.
- [18] Hoang D, Gorsich D, Castanier MP, Imani F. Knowledge graph fusion with large language models for accurate, explainable manufacturing process planning. *International Journal of Production Economics* 2026;110091.
- [19] Chen Z, Imani F. A multi-expert framework for enhancing multimodal large language models in industrial anomaly detection. *Pattern Recognition* 2025;112752.
- [20] Chen Z, Chen H, Imani M, Imani F. Can multimodal large language models be guided to improve industrial anomaly detection? In: *International design engineering technical conferences and computers and information in engineering conference*, vol. 89213, American Society of Mechanical Engineers; 2025, V02BT02A051.
- [21] Qu C, Dai S, Wei X, Cai H, Wang S, Yin D, Xu J, Wen J-R. Tool learning with large language models: A survey. *Frontiers of Computer Science* 2025;19(8):198343.
- [22] Xu W, Huang C, Gao S, Shang S. LLM-based agents for tool learning: A survey: W. Xu et al.. *Data Science and Engineering* 2025;1–31.
- [23] Li S, Corney J. MechRAG: a multimodal large language model for mechanical engineering. *Communications Engineering* 2025;4(1):187.
- [24] Fan H, Liu C, Janvisloo NE, Bian S, Fuh JYH, Lu WF, Li B. MaViLa: Unlocking new potentials in smart manufacturing through vision language models. *Journal of Manufacturing Systems* 2025;80:258–71.
- [25] Jeon J, Sim Y, Lee H, Han C, Yun D, Kim E, Nagendra SL, Jun MB, Kim Y, Lee SW, et al. ChatCNC: conversational machine monitoring via large language model and real-time data retrieval augmented generation. *Journal of Manufacturing Systems* 2025;79:504–14.
- [26] Chen X, Lei Y, Li Y, Parkinson S, Li X, Liu J, Lu F, Wang H, Wang Z, Yang B, et al. Large models for machine monitoring and fault diagnostics: opportunities, challenges, and future direction. *Journal of Dynamics, Monitoring and Diagnostics* 2025;4(2):76–90.
- [27] Xiao Y, Zheng S, Shi J, Du X, Hong J. Knowledge graph-based manufacturing process planning: A state-of-the-art review. *Journal of Manufacturing Systems* 2023;70:417–35.
- [28] Hossfeld M, Wortmann A. A universal framework for skill-based cyber-physical production systems. *Journal of Manufacturing and Materials Processing* 2024;8(5):221.
- [29] Li Y, Zhao H, Jiang H, Pan Y, Liu Z, Wu Z, Shu P, Tian J, Yang T, Xu S, et al. Large language models for manufacturing. 2024, arXiv preprint arXiv:2410.21418.
- [30] Kernan Freire S, Wang C, Foosherian M, Wellsandt S, Ruiz-Arenas S, Niforatos E. Knowledge sharing in manufacturing using LLM-powered tools: user study and model benchmarking. *Frontiers in Artificial Intelligence* 2024;7:1293084.
- [31] Lim J, Vogel-Heuser B, Kovalenko I. Large language model-enabled multi-agent manufacturing systems. In: *2024 IEEE 20th international conference on automation science and engineering. IEEE*; 2024, p. 3940–6.
- [32] Ni M, Wang T, Leng J, Chen C, Cheng L. A large language model-based manufacturing process planning approach under industry 5.0. *International Journal of Production Research* 2025;1–20.
- [33] Jadhav Y, Pak P, Farimani AB. Llm-3D print: large language models to monitor and control 3D printing. *Additive Manufacturing* 2025;105027.
- [34] Eslaminia A, Cai C, Smith C, Mei R-S, Li S, Malhotra R, Nahrstedt K, Shao C. Llm-adam: a generalizable llm agent framework for pre-print anomaly detection in additive manufacturing. 2026, arXiv preprint arXiv:2605.03328.
- [35] Samani EU, Atkeson CG. Programming manufacturing robots with imperfect ai: llms as tuning experts for fdm print configuration selection. 2026, arXiv preprint arXiv:2603.22118.
- [36] Mao T, Yang S, Fu B. A multi-agent framework for multi-source manufacturing knowledge integration and question answering. In: *Companion proceedings of the ACM on web conference 2025*. 2025, p. 1687–95.
- [37] Gekhman Z, Yona G, Aharoni R, Eyal M, Feder A, Reichart R, Herzig J. Does fine-tuning llms on new knowledge encourage hallucinations? In: *Proceedings of the 2024 conference on empirical methods in natural language processing*. 2024, p. 7765–84.
- [38] Sagor SI, Haghighi T, Alam MN, Joyee EB. Domain adaptation of large language models for polymer-composite additive manufacturing using retrieval-augmented generation and fine-tuning. 2026, arXiv preprint arXiv:2605.12516.
- [39] Lee ZP, Lin A, Tan C. Finetune-rag: fine-tuning language models to resist hallucination in retrieval-augmented generation. 2025, arXiv preprint arXiv:2505.10792.
- [40] Jignasu A, Marshall K, Ganapathysubramanian B, Balu A, Hegde C, Krishnamurthy A. Towards foundational ai models for additive manufacturing: Language models for g-code debugging, manipulation, and comprehension. 2023, arXiv preprint arXiv:2309.02465.
- [41] Šket K, Potočník D, Brezocnik M, Ficko M, Klančnik S. Large language models for G-code generation in CNC machining: A comparison of ChatGPT-3.5 and ChatGPT-4o. *Advances in Production Engineering & Management* 2025;20(2):224–38.
- [42] Abdelal M, Lokadjaja S, Engert G. Gllm: self-corrective g-code generation using large language models with user feedback. 2025, arXiv preprint arXiv:2501.17584.
- [43] Yang H, Wang H, Huang Q, Wu X, Ji W, Li Z, Han X. Aero-engine blade error distributions predictions using novel machine learning models. *International Journal of Mechanical Sciences* 2025;110262.
- [44] Hossain S, Abedin MZ, Saha RK, Touhiduzzaman M, Hossen MJ. Optimization of cutting temperature and surface roughness in cnc turning of ti-6al-4v alloy using response surface methodology. *Heliyon* 2025;11(1).
- [45] Ingle S, Raut D. Evaluation of tool wears mechanism considering machining parameters and performance parameters for titanium alloy in turning operation on CNC. *Advances in Materials and Processing Technologies* 2024;10(3):1380–400.
- [46] Ramena DVP, Vikram KA, Chebolu R, Barmavatu P, Sikarwar VS, Giri J, Sathish T. Sustainable green cutting fluid for interpreting optimization of process variables while machining on various CNC manufacturing systems—an experimental approach for exploring. *The International Journal of Advanced Manufacturing Technology* 2025;136(1):329–42.
- [47] Wu D, Wang H, Peng J, Zhang K, Yu J, Li Y, Wang M, Zhang X. Analysis of machining deformation for adaptive CNC machining technology of near-net-shaped jet engine blade. *The International Journal of Advanced Manufacturing Technology* 2019;104(9):3383–400.