Interaction, Process, Infrastructure: A Unified Framework for Human–Agent Collaboration

YUN WANG and YAN LU, Microsoft Research Asia, China

While AI tools are increasingly prevalent in knowledge work, they remain fragmented, lacking the architectural foundation for sustained, adaptive collaboration. We argue this limitation stems from their inability to represent and manage the structure of collaborative work. To bridge this gap, we propose a layered conceptual framework for human-agent systems that integrates Interaction, Process, and Infrastructure. Crucially, our framework elevates Process to a first-class concern, an explicit, inspectable structural representation of activities. The central theoretical construct is Structural Adaptation, enabling the process to dynamically reorganize itself in response to evolving goals. We introduce a five-module Process Model as the representational basis for this adaptation. This model offers a unified theoretical grounding, reimagining human-agent collaboration as a coherent system for complex real-world work.

1 Introduction

Today's AI landscape is populated by powerful yet isolated tools. Language models craft polished text, coding assistants generate sophisticated algorithms, and specialized agents automate complex tasks. While each tool demonstrates impressive capabilities individually, these systems exist largely in isolation, disconnected from each other and unaware of their broader context, creating significant practical challenges. Users regularly switch between intelligent systems. A writing assistant may generate a project plan, but has no awareness of the code later written to implement it. A research summarizer may extract key findings, but has no sense of how they are interpreted in a broader analysis pipeline. This result is predictable: users act as *human glue*, repeatedly restating intent, manually tracking dependencies, and stitching together disparate tools.

Recent research begins to explore parts of this puzzle through systems that exhibit agentic behavior. Integration frameworks [14, 17, 20, 23] connect foundation models with external tools and APIs. Autonomous systems [2, 12] enable multi-step planning and execution; Adaptive and generative UIs surface contextual actions [6]; while protocols like Model Context Protocol coordinate interactions across applications. Yet most of the techniques focus on AI-to-AI or AI-to-tool orchestration, sidelining the human. Users are either asked to predefine rigid workflows or left watching autonomous executions unfold, functioning more as *disposable utilities* than *collaborative partners*, requiring humans to be equipped with *prompt-engineering* skills just to gain meaningful access.

In contrast, real-world professional workflows (*e.g.,* software engineering, scientific research, or creative design) rarely follow predefined steps. Goals shift, constraints emerge, and strategies evolve. Professionals explore, reflect, revise, and adapt. They uncover the problem itself through the act of working. Effective human-agent collaboration should embrace this complexity, supporting human judgment and shared adaptation.

This motivates our central argument: what is missing is not AI capability, but the systems' inability to represent and adapt the structure of collaborative work. Current systems treat collaboration as a sequence of tasks, conversations, or execution traces, rather than as an evolving activity with changing interpretations, goals, workflows, and dependencies. Without an explicit and adaptable representation of process, systems cannot maintain continuity alongside the human. We call this missing capability structural adaptation.

Authors' Contact Information: Yun Wang, wangyun@microsoft.com; Yan Lu, yanlu@microsoft.com, Microsoft Research Asia, China.

This paper offers a conceptual and theoretical framework for representing collaborative activity and its evolution. Our goal is to articulate the representational foundation needed for AI-native collaboration. We contribute: (1) **Structural adaptation as a core theoretical construct**. We identify and formalize structural adaptation, the capacity of a collaborative process to reorganize its own structure as goals, constraints, and interpretations change. We characterize how this construct explains the structural limitations of current AI systems. (2) **A process-based representational foundation for collaborative activity.** We introduce process as the explicit representation that supports structural adaptation. We define the representational requirements of collaborative activity and propose a minimal, domain-agnostic five-part process model that captures problems, workflows, operations, environment, and reflection. (3) **A conceptual framework for process-first collaboration.** We outline a three-layer conceptual framework that situates process within system design and illustrates how explicit, persistent process representations can support long-term collaboration. Together, these contributions provide a unified theoretical foundation for process-first, structurally adaptive human-agent collaboration.

2 Related Work

Research on human–agent collaboration intersects three major areas: mixed-initiative interaction, activity-centric theories, and agent or workflow architectures. While each line of work contributes important insights, none a mechanism for evolving that structure as human and agent jointly refine their understanding of the work. We outline most relevant traditions and their limitations with respect to this need.

Mixed-initiative (MI) frameworks examine how humans and automated services negotiate control, delegate actions, and coordinate initiative [1, 8]. Classic MI systems introduced principles for adjustable autonomy, forming the backbone of intelligent user interfaces. However, MI models assume the structure of the task is fixed or implicit, and do not represent goals and workflows as editable objects. Recent research on human–AI collaboration similarly emphasize autonomy, transparency, and role negotiation [13, 22, 26], but mainly focus on interactional or social dynamics. Our work addresses a different layer of collaboration: explicit, semantic representations of activity structure and their adaptation over time. A process-first perspective exposes the structure itself. This capability of structural adaptation is essential for open-ended human-agent collaboration.

Activity theory [7], distributed cognition [9, 27], and sensemaking frameworks [19, 21] describe how humans structure and evolve their own activities through interpretation and mediation, offering strong conceptual grounding. They emphasize that activity, not task, is the meaningful unit of human work. Activity-based computing further operationalizes by treating activities as first-class computational constructs across sessions [3, 4]. However, activity-based theories and systems model human behaviors, analyze how people coordinate with tools or with each other, and treat tools as passive resources. Our work extends these foundations by introducing an explicit representation of collaborative process, with structure can be reshaped as human and agent co-develop the activity. It enables agentic partners capable of modifying goals, workflows, or representations.

Recent LLM-based agents can generate multistep plans, invoke tools, and perform iterative refinement [24, 32, 33]. Workflow and planning systems similarly encode task sequences and control flow through explicit schemas [11, 16, 28], but they are fundamentally execution-centered: plans are constructed internally, and workflows remain difficult for users to inspect or modify. In addition, recent coordination frameworks such as Model Context Protocol (MCP) focus on agent-to-agent interoperability, but they do not provide a manipulable representation of the collaborative activity that can evolve with human intent. Our work focuses on how the structure of the collaborative activity is represented and adapted over time, offering capabilities essential for open-ended human–agent collaboration.

3 Collaborative Activity, Process, and Structural Adaptation

We characterize collaborative activity as the unit of work, introduce process as its structural representation, and develop structural adaptation as the key capability through which collaborative structure evolves.

Activity as the Unit of Collaborative Work. Collaborative work unfolds not as a sequence of isolated tasks but as an evolving activity. An *activity* is a purposeful, meaning-laden unit of work. In real practice, including writing, analysis, design, or research, people navigate cycles of framing, exploration, synthesis, and adjustment. These shifts arise from engagement with the problem as it develops (Figure 1). Activities possess properties essential for collaboration. They are interpretive, requiring ongoing sense-making; structured, involving relationships among goals, constraints, and subproblems; contextual, drawing on heterogeneous information and artifacts; and dynamic, continually evolving as understanding deepens. Systems that respond only to individual commands or task specifications cannot participate in this evolving structure. Effective collaboration therefore requires a representational account of the activity itself.

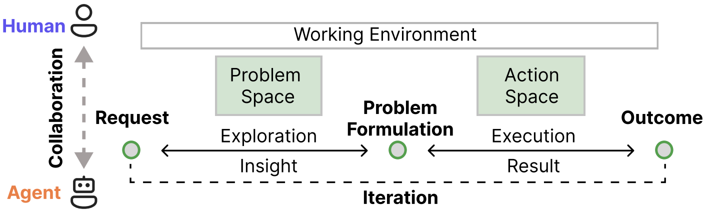

Fig. 1. Collaborative activity unfolds through iterative cycles in which participants move between the problem space and the action space. Exploration generates insight that reshapes the problem formulation, while execution produces results that inform subsequent steps. This evolving structure motivates the need for an explicit process representation.

Process as the Representation of Activity. The representation of activity's structure is what we call a process. A *process* is a structured depiction of an activity's goal, organization, available operations, contextual dependencies, and current state. Unlike plans, histories, or execution logs, a process is persistent, explicit, and editable. It provides a shared substrate through which both human and AI can interpret what is being done, why it is being done, and how the work is unfolding. Processes vary in granularity: a process may represent an entire activity or a focused sub-activity. Activities may contain sub-activities, and each can be represented as a process using the same structural schema. What defines a process is not its size but its role as the structured representation of a coherent unit of activity. Because activities evolve, the process is expected to also evolve, adjusting goals, reorganizing structure, and shifting where work is being advanced. A process enables continuity, shared understanding, and coordinated adaptation over time.

Structural Adaptation as a Core Construct. As collaborative activities unfold, their structure rarely evolves uniformly. The Components of a process shift at different times, at different rates, and for different reasons. Structural adaptation captures these non-uniform dynamics: it is the capacity of a process to modify its own representational structure as collaboration progresses. This construct explains how human–AI collaboration maintains coherence, alignment, and continuity across extended, evolving work. Structural adaptation takes three fundamental forms. These three forms together constitute the full space of structural change necessary for activity-level collaboration:

- **Reframing.** Changes to the meaning structure of the activity, *e.g.*, shifting goals, altering assumptions, rearticulating problem boundaries, or redefining what success entails. Reframing realigns the overarching purpose of the collaborative work.
- **Reorganization.** Changes to the organizational structure of the process, *e.g.*, rearranging steps, restructuring dependencies, introducing new pathways, or consolidating existing ones. Reorganization shapes how work is coordinated and sequenced.
- **Reorientation.** Changes to where the activity is presently being advanced within the process representation. Reorientation redirecting effort toward a different sub-problem, component, or level of detail, without altering the overall topology of the process. It captures the dynamic movement of the activity's structural frontier.

4 The Collaborative Process Model

As discussed in Section 3, without process structure, systems cannot maintain continuity, support reframing, or participate in reorganizing the work. To provide the representational basis for structural adaptation, we introduce a five-module process model that captures the structural elements necessary for representing collaborative work.

The Five Modules. A process must represent multiple facets of an evolving activity: its purpose, organization, available actions, resources, and the meta-level reasoning through which the work is shaped. The five modules described below constitute a minimal yet sufficient set of representational components. Each module captures a distinct structural dimension, and together they form a coherent substrate for supporting structural adaptation (Figure 2).

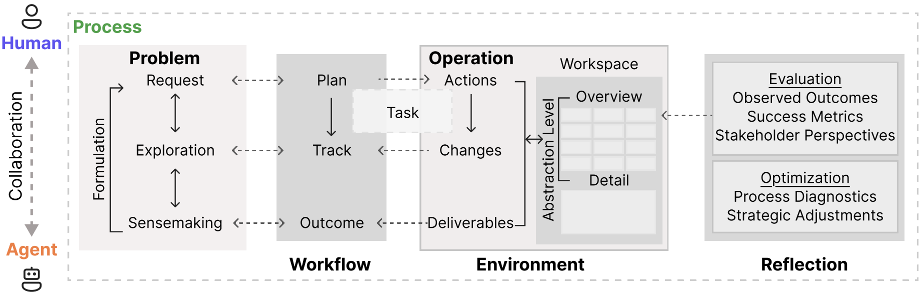

Fig. 2. The process as an explicit representation of collaborative activity. It consists of five structural modules: Problem Space, Workflow, Operations, Environment, and Reflection. They capture the evolving organization, context, and methods of the activity. This representational substrate enables structural adaptation across reframing, reorganization, and reinterpretation.

Problem Space The Problem Space represents the evolving interpretation of the activity, its goals, assumptions, hypotheses, constraints, and framing. It captures what the work is about and how the problem is currently understood. As understanding deepens or shifts, the problem space can be reframed, allowing the collaboration to redefine aims, criteria, or conceptual boundaries.

Workflow The Workflow part models the organization of the work: its stages, partial orders, dependencies, and parallel threads. It provides a structural scaffold rather than a fixed script. As strategies change, the workflow may be reorganized: reordering steps, merging or splitting paths, or introducing new subactivities. Workflow captures how the work is structured at any moment.

Operations The Operations module specifies the set of actions, procedures, tools, or model capabilities involved during collaboration. It describes what and how tasks are completed. Operations may be high-level reasoning modes, domain-specific methods, tool invocations, or user-defined procedures. As collaboration evolves, operations may be reparameterized or new operations introduced to reflect emerging needs or strategies.

Environment The Environment is the shared workspace of the collaboration: the artifacts, intermediate results, external resources, available tools, and workspace state on which the work depends. It may include documents, code, datasets, sketches, diagrams, generated outputs, and any tool or resource accessible during the activity. The environment changes dynamically as new artifacts are created, or intermediate results become input for subsequent steps. It provides the material substrate on which operations act and the persistent grounding for continuity over time.

Reflection The Reflection module encompasses the meta-level processes that examine progress, detect inconsistencies, critique assumptions, and decide when structural change is needed. Reflection coordinates reframing, workflow reorganization, abstraction shifts, and other forms of structural adaptation. It answers why the work should change and how the structure should evolve.

Dynamics and Interactions Among Modules. Structural adaptation arises from coordinated changes across modules. Reframing in the Problem Space may prompt reorganization in the Workflow; new workflows require updated Operations; newly created artifacts reshape the Environment; and changes in the environment can, in turn, reveal new constraints or opportunities that alter the Problem Space. Reflection mediates these interactions, ensuring that structural changes remain coherent and aligned with the evolving activity. Together, these five modules form a coherent representational scaffold through which human and AI can jointly maintain, inspect, and adapt the structure of collaborative work. These interactions create a feedback loop in which structure is continuously assessed, reshaped, and realigned with evolving human intent.

5 Framework for Human–Agent Collaboration

With Process as the representational substrate that links user intent with system behavior, we propose a unified architecture that separates interaction, representation, and execution into three conceptual layers. This framework treats collaboration as a system-level capability supported by explicit process representation (Figure 3): (1) an Interaction Layer, where users express intent and inspect or modify structure; (2) a Process Layer, which maintains the explicit, evolving representation introduced; and (3) an Infrastructure Layer, which grounds operations in model capabilities, external tools, and computational resources. The framework clarifies how structural adaptation is exposed to users, how it is maintained over time, and how it is operationalized through available tools and resources. In doing so, it frames collaboration as a representational capability rather than a sequence of tasks, enabling both humans and AI systems to participate in shaping the work.

Interaction Layer. The Interaction Layer provides the projections through which users encounter and shape the process. This layer exposes structured representations of the activity, allowing users to interpret the work and influence its evolution. Interactions vary along two fundamental dimensions. *Modality* determines how structure is expressed (*e.g.,* text, voice, sketches, diagrams, canvases, etc.). *Initiative and control* determine how collaboration unfolds, whether user-driven, system-suggested, or mixed-initiative. These dimensions shape how users participate in the collaboration: articulating intent, refining assumptions, steering direction, or restructuring parts of the process.

A central property of this layer is decoupling user interface from process logic. The same underlying process may be projected as a conversational exchange, a workflow diagram, a timeline, or a spatial workspace, depending on user preference, task context, or stage of work. These projections provide different views of process representation, enabling structural inspection and manipulation at appropriate levels of abstraction. This decoupling echoes prior work on instrumental interaction [5]. However, in our context, instruments serve beyond mediators of action, but the evolving

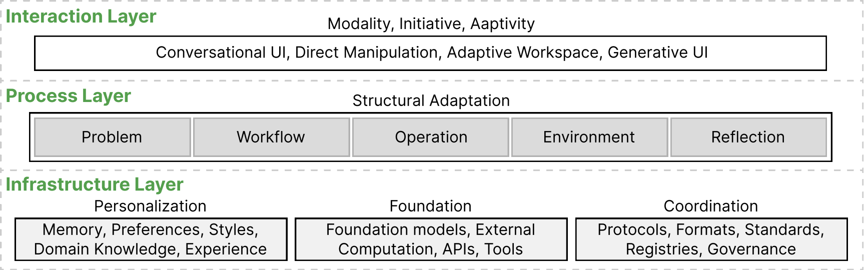
Interaction Layer
Modality, Initiative, Aaptivity
Conversational UI, Direct Manipulation, Adaptive Workspace, Generative UI
Process Layer
Structural Adaptation
Problem
Workflow
Operation
Environment
Reflection
Infrastructure Layer
Personalization
Memory, Preferences, Styles, Domain Knowledge, Experience
Foundation
Foundation models, External Computation, APIs, Tools
Coordination
Protocols, Formats, Standards, Registries, Governance

Fig. 3. Three-layer conceptual framework for process-first collaboration. The Interaction Layer provides user-facing projections of the activity; the Process Layer maintains the explicit, editable structure of the activity; and the Infrastructure Layer supplies the models, tools, memory, and resources that support execution and continuity.

process that reveals different stages of work. The layer supports more than input and output. Users can interrogate decisions, examine dependencies, annotate assumptions, and request reframing or reorganization. In this sense, the interaction layer aligns user interactions with underlying activity.

Process Layer. The Process Layer serves as the representational core of the architecture. It maintains the explicit, evolving structure of the collaborative activity as introduced in Section 4. These modules form the semantic substrate through which collaboration is interpreted, guided, and adapted. The Process Layer preserves continuity across time, tools, and collaborators and records the rationale behind decisions and maintains the representational state that makes different forms of collaboration possible.

Because it sits between interaction and execution, the Process Layer also mediates cross-layer propagation. User actions at the Interaction Layer, such as reframing goals, restructuring workflows, annotating assumptions, or selecting operations, are interpreted as modifications to the process representation. Conversely, updates within the Process Layer guide the Infrastructure Layer by determining which operations are valid, which tools should be invoked, and how results should be integrated or interpreted. This positioning establishes a stable locus of meaning across the system: a shared, inspectable representation that both humans and AI systems rely on to coordinate action, maintain alignment, and evolve the activity over time.

Infrastructure Layer. The Infrastructure Layer provides the computational substrate that grounds the process in executable capabilities. It operationalizes the structures maintained in the Process Layer. While the Process Layer specifies what the activity is and how it is structured, the Infrastructure Layer determines how operations are carried out and how their results return to update the representation. This layer may integrate a wide range of mechanisms, including foundation models, domain-specific APIs and tools, contextual memory systems, and communication protocols. These components supply the computational resources required to enact the operations. Because processes evolve, the infrastructure must support flexible, heterogeneous, and extensible forms of execution, allowing different models, tools, or data sources to be combined or substituted without altering the representational state.

A key property of this layer is its decoupling from how processes are represented or projected. The same process structure can be enacted through different computational pathways depending on available tools, model capabilities, or task constraints. This separation enables scalable, multi-tool collaboration while preserving a stable semantic core in the Process Layer. It ensures that structural adaptations translate coherently into action, while execution details remain modular and replaceable. Through this coordination between representation and execution, the Infrastructure Layer completes the architectural stack, enabling adaptive collaboration to be realized in real tasks and environments.

6 Representational Analysis of AI Systems

Contemporary AI systems commonly appear as conversational assistants, application-embedded copilots, or task-oriented agents (Figure 4). Although these systems operate differently, they share a common characteristic: each encodes implicit assumptions about how collaborative work is structured. We do not aim to provide a taxonomy, nor an exhaustive categorization. Instead, this section applies the representational model as an analytical lens to illustrate how contemporary system forms express, or fail to express the structural elements required for collaborative activity, demonstrating the explanatory power and diagnostic clarity of the proposed framework.

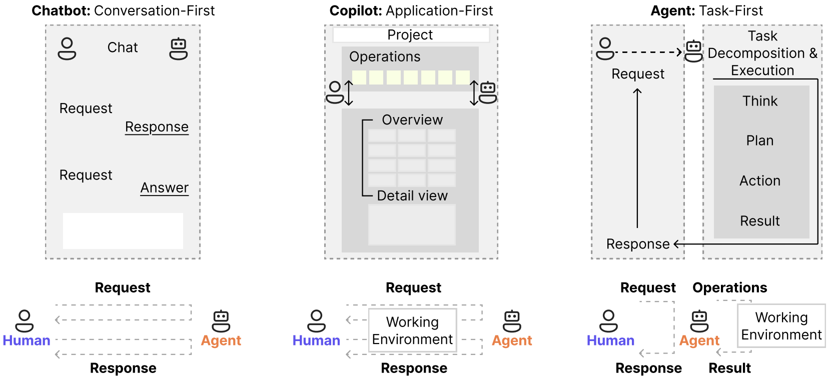

Fig. 4. Representative forms of systems in today's AI landscape: chatbots, copilots, and agents. Each encodes a distinct model of collaboration but remains isolated, lacking connective logic across tasks and interfaces.

Structural Adaptation Lens. Viewed through the lens of structural adaptation, all three system forms exhibit the same limitation: they treat changes in activity implicitly, locally, and reactively, making it difficult for users and systems to jointly evolve the work.

Chat-based assistants Chat interfaces support a fluid exchange of language but provide no representational structure for the activity. All history is compressed into a linear conversation. Users can request new directions, but each redirection is handled as a local conversational turn rather than as a modification of an underlying activity structure.

Application-embedded copilots Copilots offer contextual assistance anchored to a specific application, enabling local adjustments but not structural change. They can refine code within an editor, rewrite text within a document, or summarize a dataset inside a notebook, yet the underlying activity remains outside the system's representational horizon. Any shift in direction requires users to manually realign artifacts across tools.

Task-oriented agents Agents often possess internal plans or tool-use sequences, but these plans are opaque, brittle, and difficult for users to inspect or reshape. Once execution begins, deviations from the anticipated structure cause plans to fail or restart. Adaptation occurs only within the narrow confines of a predefined task.

Process Analysis. The five components of process provide a representational vocabulary for describing the structure of a collaborative activity. Applying these components to existing AI system forms makes visible how each system organizes activity structure and which structural elements remain implicit, inaccessible, or absent (Table 1).

- **Problem Space Representation** Chat-based assistants treat the problem space as implicit in the conversation. Goals must be continually restated and cannot be maintained as a coherent representation. Copilots ground the problem space in a specific application. Agents typically generate an internal problem framing when initiating a plan, yet this framing usually remains opaque and cannot be reshaped or negotiated by the user.
- **Workflow and Structural Rigidity** Chat interfaces provide no workflow representation: all progress collapses into a linear turn-by-turn trace. Copilots operate within the workflow implicitly built into the host application, reflecting the software's operational model, such as editing a file, modifying code cells, and are fixed by design. Because these structures are predefined, they offer no representation of the activity's broader workflow. Agents construct workflows by decomposing a user-specified goal into an internal plan. This plan guides tool calls and action sequences, but it cannot be reshaped, extended, or reorganized once generated.
- **Operation Space Exposure** Chat-based systems do not expose a structured set of operations. Interactions take the form of open-ended language requests. Copilots surface operations that are bound to the application. Agents leverage tool APIs and planners, yet the available operations and their sequencing remain inaccessible to users, limiting opportunities for mixed-initiative organization.
- **Environment as Shared Context** In chat-based systems, the environment is confined to the transient context window. Copilots access the local environment of the application but cannot integrate information across applications. Agents maintain internal memory, but these remain unmanageable to users for collaboration.
- **Reflection and Process-Level Feedback** Across all three system forms, reflective activity remains largely limited. Chat-based assistants do not maintain a representation of what has been learned, reconsidered, or revised. Copilots may offer suggestions or alternative edits, yet these moments arise from local heuristics rather than from a process that the user can inspect or participate in. Agents sometimes perform internal retries or self-corrections. But these reflective adjustments cannot coordinate with user-led reframing or restructuring.

Process Module	**Chat**	**Copilot**	**Agent**
Problem Space	Implicit	Local	Internal
Workflow	None	software-implied	Fixed-plan structure
Operation	Language-driven	software-defined	Planner-driven, hidden
Environment	Context window	Local workspace	Internal state
Reflection	None	Local suggestions	Opaque adjustments

Table 1. Process modules expressed in three contemporary AI system forms.

Architectural Analysis. The three-layer architecture offers a system-level view on how contemporary AI systems organize interaction, reasoning, and execution. Examining through this perspective reveals how distinct architectural commitments shape what users can see, modify, or coordinate during collaborative work.

Interaction Layer Chat-based assistants expose a single interaction surface: a conversational box. Intent, clarification, and feedback all pass through text exchanges, making interaction flexible but limiting. Copilots ground interaction in the host application's interface. Users operate through familiar controls, and the AI's suggestions are tightly coupled to the artifact at hand. Agent systems vary widely. Some rely on text input with a chat-like surface; others expose simple task forms or parameter panels. Automation-oriented platforms, such as RPA tools, present node–link flows that show what the agent is doing [31]. These visualizations enable definition of automation sequences rather than editable representations of an evolving activity.

Process Layer Across these system forms, the process layer is either absent or implicit. Chat systems maintain no explicit representation of goals, workflow, or reflective state. Copilots follow the software's built-in operational logics. Agents generate internal plans or states. None of these systems offer a shared process representation that could support structural adaptation.

Infrastructure Layer The infrastructure layer is where current systems differ most. Chat-based assistants rely primarily on model inference. Copilots combine model inference with application APIs, enabling richer operational grounding but keeping it local to the tool. Agents integrate planners, retrieval modules, and tool APIs, forming complex execution pipelines. They allow the agent to perform operations, but not to act on process representations or expose the intermediate structure of the activity. Viewed through the architectural lens, current AI systems differ in how they distribute capability across interaction and infrastructure layers, yet share a common absence in the process layer. This absence shapes how collaboration unfolds: users interact with surfaces or execution traces rather than with the evolving structure of the activity. As a result, adaptation remains local to the system's interaction modality or execution pipeline, rather than grounded in a shared model of the activity.

Synthesis. The three analytical lenses—structural adaptation, process components, and architectural layering—converge on a consistent pattern: Contemporary AI systems support local task execution but do not maintain an explicit representation of the evolving activity. Across the five components of process, many elements of activity remain implicit, fragmented, or inaccessible. Viewed architecturally, these limitations reflect the absence of a process layer. Taken together, these analyses demonstrate how the representational model introduced in this paper can be used to examine contemporary systems. Here we apply the model to three broad system forms, but the same lenses can be used at finer granularity to analyze individual systems or components.

7 Research Challenges and Opportunities

The layered model presented in this paper recasts collaborative intelligence as a system-level human-agent interaction and coordination. Effective support for open-ended, evolving work demands explicit representations of process across time, tools, and participants.

Toward a Unified, Adaptive Framework. As human-agent collaboration becomes more pervasive, the lack of a unified representation of work remains a fundamental bottleneck. A unified architecture offers structural coherence and design flexibility. Like the von Neumann model in computing [29] or the MVC pattern in UI design [10], our model treats collaboration as modular and composable. It enables reuse of components (*e.g.,* planning, memory) and extension of capabilities without rearchitecting the entire stack. It also supports *epistemic generalization*: while tools vary, domains such as design, law, and research often share high-level process patterns.

As analyzed in Section 6, Current AI tools are largely static. Adaptation, when it occurs, is often local and reactive. Our framework addresses this by combining modular separation with explicit linkage. Each layer is independently evolvable yet structurally aligned, allowing updates in one layer to propagate to others. More importantly, because the process layer is explicitly modeled, the system gains visibility into task structure, delegation patterns, and breakdowns. This enables the capture of structured traces of collaboration, data that can support meta-reasoning across the human-agent workflow. Future agent system can then revise workflows, adjust interface patterns, reallocate tools, or propose model changes. In this way, ongoing experience fuels *architectural self-improvement.*

Interfaces that Adapt to Users, Scenarios, and Evolving Processes. Interaction in AI systems has largely focused on usability, responsiveness, and intuitive input/output modalities. Process-aligned systems require a broader perspective: interfaces are not merely endpoints for command issuance or response display, but *active surfaces* for perceiving, guiding, and reflecting on collaborative workflows. This perspective draws on foundational HCI concepts such as *interface transparency* [15], *mixed-initiative coordination* [8], and *calm computing* [30], while extending them toward dynamic, process-aware interaction. To support such workflows, interfaces face three interrelated challenges: (1) *Dynamic Representation.* Interfaces need to adapt process granularity (overview vs. detail), modality (text, diagrams, timelines), and semantic framing (goal-oriented vs. action-oriented) to fit evolving contexts. Designing these representational grammars is nontrivial. Prior work on adaptive interfaces [25] and progressive disclosure [15] provides useful principles, yet process-aligned systems call for richer models. (2) *Cross-Level Navigation.* Professionals frequently shift across abstraction levels. Interfaces should support bidirectional navigation: from plans to operations, annotations to execution traces, or breakdowns to causal rationale. This recalls semantic zooming [18], but with an emphasis on aligning mental models across users, agents, and shifting process state. (3) *Attention Orchestration.* Managing user attention becomes essential as workflows evolve. Systems need ways to surface process drift, highlight misalignments, or prompt reflection without overwhelming users. This challenge revisits classic notions of system visibility [15] and calm technology [30], now situated in dynamic, multi-stage workflows. Design questions arise: When should the system indicate ambiguity? How might it encourage recalibration? What cues best elicit user feedback on process plans?

Co-Evolution and Longitudinal Human-Agent Collaboration. Beyond short-term performance or flexibility, process-aligned systems open the door to longitudinal co-evolution between humans and agents. Rather than acting as disposable utilities or context-free assistants, future agents could develop shared memory, mutual understanding, and evolving collaboration patterns that persist across sessions, tools, and goals. Evaluating such dynamics requires methodological innovation. More importantly, co-evolution is not one-sided. As systems adapt, so do users. Agents should not only respond to input, but also facilitate meta-cognition, helping both parties reflect on *how* they are collaborating and *how* that collaboration is changing. This reframes the agent's role from reactive assistant to reflective partner, capable of maturing alongside its user, adjusting to evolving contexts, and sustaining the logic of complex, unfolding work.

Limitations and Tradeoffs. Explicit process representations introduce several limitations and tradeoffs. First, maintaining structured activity representations may create representation overhead, and not all tasks justify this additional cognitive and computational cost. Second, explicit structure risks over-structuring early, open-ended, or exploratory work, potentially biasing how problems are framed even though structural adaptation enables later revision. Third, collaborative restructuring introduces coordination costs: inconsistencies, invalid dependencies, or divergent interpretations between human and AI may require additional effort to detect and resolve. Fourth, making process structure visible improves transparency but does not eliminate trust tensions, as users may still question the validity of AI-proposed structural changes. Finally, the framework presumes that operations from heterogeneous models and tools can be meaningfully organized under a unified process representation—an aspiration that may be difficult to realize in practice. These limitations do not undermine the conceptual contribution but instead clarify the boundaries of the framework and point to open challenges for future process-first systems.

8 Conclusion

The rise of increasingly capable AI systems has unlocked new possibilities for supporting human creativity, analysis, and decision-making. Looking ahead, we advocate a shift from task augmentation to process-first system design. Our layered architecture, spanning interaction, process, and infrastructure, offers a conceptual foundation for building intelligent systems that are not only capable but coordinated; not only responsive but reflective. By embedding shared logic at the core of human-agent systems, we open the door to a new generation of tools that can adapt, evolve, and genuinely collaborate with their users.

References

[1] James E Allen, Curry I Guinn, and Eric Horvtz. 1999. Mixed-initiative interaction. *IEEE Intelligent Systems and their Applications* 14, 5 (1999), 14–23.

[2] AutoGPT Team. 2023. Auto-GPT: An Autonomous GPT-4 Experiment. https://agpt.co/. Accessed: 2025-05-01.

[3] Jakob E Bardram. 2005. Activity-based computing: support for mobility and collaboration in ubiquitous computing. *Personal and Ubiquitous Computing* 9, 5 (2005), 312–322.

[4] Jakob E. Bardram, Jonathan Bunde-Pedersen, and Mads Søgaard. 2006. Support for Activity-Based Computing in a Personal Computing Operating System. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Montreal, Canada, 211–220.

[5] Michel Beaudouin-Lafon. 2000. Instrumental interaction: an interaction model for designing post-WIMP user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 446–453.

[6] Michel Beaudouin-Lafon, Susanne Bødker, and Wendy E Mackay. 2021. Generative theories of interaction. *ACM Transactions on Computer-Human Interaction (TOCHI)* 28, 6 (2021), 1–54.

[7] Yrjö Engeström. 2001. Expansive learning at work: Toward an activity theoretical reconceptualization. *Journal of education and work* 14, 1 (2001), 133–156.

[8] Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 159–166.

[9] Edwin Hutchins. 2000. Distributed cognition. *International encyclopedia of the social and behavioral sciences* 138, 1 (2000), 1–10.

[10] Glenn E. Krasner and Stephen T. Pope. 1988. A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80. *Journal of Object-Oriented Programming* 1, 3 (1988), 26–49.

[11] Mary C. Lacity and Leslie P. Willcocks. 2016. Robotic Process Automation at Telefónica O2. *MIS Quarterly Executive* 15, 1 (2016), 21–35.

[12] LangChain Team. 2022. LangChain: Building Applications with LLMs through Composability. https://www.langchain.com/. Accessed: 2025-05-01.

[13] Haotian Li, Yun Wang, and Huamin Qu. 2024. Where are we so far? understanding data storytelling tools from the perspective of human-ai collaboration. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–19.

[14] Yaobo Liang, Chenfei Wu, Ting Song, Wenshan Wu, Yan Xia, Yu Liu, Yang Ou, Shuai Lu, Lei Ji, Shaoguang Mao, et al. 2024. Taskmatrix. ai: Completing tasks by connecting foundation models with millions of apis. *Intelligent Computing* 3 (2024), 0063.

[15] Donald A. Norman. 1988. *The Design of Everyday Things*. Basic Books, New York, NY.

[16] Object Management Group. 2011. Business Process Model and Notation (BPMN), Version 2.0. https://www.omg.org/spec/BPMN/2.0

[17] Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2024. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems* 37 (2024), 126544–126565.

[18] Ken Perlin and David Fox. 1993. Pad: an alternative approach to the computer interface. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. 57–64.

[19] Peter Pirolli and Stuart Card. 2005. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of international conference on intelligence analysis*, Vol. 5. McLean, VA, USA, 2–4.

[20] Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Xuanhe Zhou, Yufei Huang, Chaojun Xiao, et al. 2024. Tool learning with foundation models. *Comput. Surveys* 57, 4 (2024), 1–40.

[21] Daniel M Russell, Mark J Stefik, Peter Pirolli, and Stuart K Card. 1993. The cost structure of sensemaking. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*. 269–276.

[22] Isabella Seeber, Eva Bittner, Robert O Briggs, Triparna De Vreede, Gert-Jan De Vreede, Aaron Elkins, Ronald Maier, Alexander B Merz, Sarah Oeste-Reiß, Nils Randrup, et al. 2020. Machines as teammates: A research agenda on AI in team collaboration. *Information & management* 57, 2 (2020), 103174.

[23] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems* 36 (2023), 38154–38180.

[24] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems* 36 (2023), 8634–8652.

[25] Ben Shneiderman. 1998. *Designing the User Interface: Strategies for Effective Human–Computer Interaction* (3rd ed.). Addison-Wesley, Reading, MA.

[26] Ben Shneiderman. 2022. *Human-centered AI*. Oxford University Press.

[27] Lucy A. Suchman. 1987. *Plans and Situated Actions: The Problem of Human–Machine Communication*. Cambridge University Press, Cambridge, UK.

[28] Wil MP Van der Aalst. 2022. Process mining: a 360 degree overview. In *Process mining handbook*. Springer, 3–34.

[29] John von Neumann. 1945. *First Draft of a Report on the EDVAC*. Technical Report. Moore School of Electrical Engineering, University of Pennsylvania.

[30] Mark Weiser and John Seely Brown. 1996. *The Coming Age of Calm Technology*. Technical Report. Xerox PARC. Technical report.

[31] Leslie Willcocks and Mary Lacity. 2016. Robotic process automation: The next transformation lever for shared services. *Journal of Information Technology Teaching Cases* (2016).

[32] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems* 36 (2023), 11809–11822.

[33] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *Proceedings of the 11th International Conference on Learning Representations (ICLR 2023)*.