Journal of Artificial Intelligence Practice (2026)
Clausius Scientific Press, Canada

DOI: 10.23977/jaip.2026.090117
ISSN 2371-8412 Vol. 9 Num. 1

Research on a Multi-SOP Interruption–Resumption Agentic Algorithm for Complex Business Workflows

Zhenggang Wei[1,a], Jinbin Xu[1,b], Zelin Zheng[1,c], Xialin Zhang[1,d,*], Haichao Gao[1,e]

[1]Ping An Jin Guan Jia Development Team, Ping An Life Insurance Company of China, Ltd., Shenzhen, China
[a]weizhenggang617@pingan.com.cn, [b]xujinbin623@pingan.com.cn, [c]zhengzelin065@pingan.com.cn, [d]xlinzhang1007@163.com, [e]1011842387@qq.com
**Corresponding author*

Keywords: Multi-SOP Collaboration, Global State Awareness, Diff Reasoning, Dual-State Decoupling, Checkpoint Continuity

Abstract: In complex business scenarios, a customer will often switch quickly among multiple SOPs. The challenge for traditional intelligent customer service systems is that they do not provide the ability to remove siloed information, retain state, and allow for intelligent recovery after interruptions when processing multiple SOPs, requiring users to restart their workflows again from the beginning. In this paper, we will describe an agentic algorithm based on a multi-SOP agentic interruption/resumption perspective. The framework consists of three core capabilities: First, dual-state decoupling separates nondeterministic cognitive decision-making (Coordinator Agent) from deterministic business execution via stateless SOP tools, reducing orchestration complexity and computational overhead. Second, a six-tuple Global State Container serves as the single source of truth for cross-SOP information transfer, enabling automatic reuse of critical business entities and silent propagation of prerequisite parameters across heterogeneous workflows. Third, a Diff Reasoning mechanism measures state discrepancies between frozen snapshots and current global memory, dynamically selecting lossless recovery or safe rollback paths to prevent semantic rollback attacks. The multi-SOP framework reduced the need for users to input the same information more than once, resulting in an 85% reduction to only 12.3, a cross-SOP recovery success rate of 96.3%, and a 73.5% successful chained task completion rate. Average end -to-end time to completion for the multi-SOP framework was also improved by 52.4% (24.6 minutes to 11.7 minutes). Finally, the overall user satisfaction score increased from 2.8 to 4.5.

1. Introduction

As a result of improvements in large language model technologies, there has been an increasing use of AI-based Platforms for customer services across various sectors such as Retail, Travel, Healthcare, etc. within the Insurance Industry; however, industry standards dictate that this process consists of many individual SOPs (Standard Operating Procedures) and one customer's interaction with an SOP requires interaction with numerous processes all of which have many independent

SOPs which are interconnected with each other. Thus, there is often a time lag when a customer interacts with a customer service representative and the customer can change their initial intent (e.g., if they ask a question to get more information) and/or because of the long time frame of the interaction (often over 30 minutes), the customer may be provided with more information and/or distracted (e.g., by an e-mail or phone call) and suspended their current workflow entirely.

Standard architectures that are intelligent do not currently have the ability to process transactions related to collaborative workflows and continue using the same transaction model for these kinds of situations based on the fact that the execution logic used in each of these types of workflows, by design, is independent. In addition, customers will have to provide their same information in order to finalize their transaction again which causes an enormous effect on the customer's experience known as information silo.Because each transaction does not have a mechanism for enabling the preservation of the necessary information needed to restart the workflow after a failure in the current workflow situation and before the next interaction with a different representative is initiated, the customer must start again from the beginning, thus creating a situation of customer interactions being over 85% repeatable. Another significant issue is that the current scheduling models being used by these companies utilize static Directed Acyclic Graph (DAG) based orchestration which prevents them from achieving dynamic routing or providing customers with intelligent task continuity associated with dynamically changing global states.

2. Related Work

2.1. Analysis of Interruption – Resumption Agentic Algorithms in Complex Business Domains

In complex business environments have driven the development of a number of different technical solutions to Multi-SOP interruptions from both academia and industry [1,2,3]. Two representative methods for handling multiple SOPs are discussed here, including Checkpoint Persistence Mechanisms and Finite State Machine (FSM) Frameworks (Table 1).

2.1.1. Checkpoint Persistence Mechanisms

Within the intelligent agent orchestration domain, checkpointing is one of the most popular forms of state persistence utilized today. LangGraph is one of the key example frameworks that utilize checkpoint state persistence, allowing for workflow-based rollbacks by storing each channel's states at the end of each computational superstep during the workflows. Checkpoint state persistence has several advantages over other persistence approaches both in terms of ease of implementation and the ability to persistently store states. The main drawback to the use of checkpoint state persistence is that it does not allow for state transfer between threads. When user intent has fundamentally changed, such as when transitioning between Workflows A & B, frameworks with checkpoint state persistence do not permit the extraction of state from Workflows A or B; hence there is no mechanism for moving, merging, or migrating state between workflows [4]. Currently, the recovery process after a failure does not take into account the semantic impact of changes in the global state; therefore, this will greatly increase the risk of "Semantic Rollback Attacks" that can result in many dire consequences such as duplicate financial deductions in high-risk areas [5,6,7].

2.1.2. Finite State Machines and Rule Engines

Finite State Machine (FSM) is a traditional technique that can be used in the management of dialogues in intelligent customer service systems. FSMs allow for strict controls on the flow of dialogue by representing business processes as a collection of nodes (states) connected by

transitions (links or edges). FSMs are most effective in terms of controllability, as the ability to manage a single workflow through an FSM is strong; yet, FSMs have been built on the assumption of a single closed state space for many reasons [8]. When multiple SOPs must be executed in parallel or interleaved, a major problem arises as there will be an enormous number of possible state combinations for each of the FSMs involved. In addition, one of the major limitations of FSMs is that they do not possess any ability to understand semantic entities, therefore making it impossible for the FSMs to automatically identify reusable entities across the different SOPs. This limits the ability to share information across processes, which is accomplished solely through manually created mapping rules, resulting in high maintenance costs and limited scalability [9,10].

2.2. Feasibility Analysis and Innovation Discussion

This section focuses on the technical feasibility of the proposed framework and summarizes its usefulness when compared to other similar frameworks across 4 different design areas.

2.2.1. Feasibility Analysis

The multi-SOP interruption and resumption design showed excellent technical feasibility when examined according to the 4 design dimensions outlined in [11, 12].

Architecture: The dual-state decoupling architecture consists of three layers: micro-frontend interaction layer, Coordinator Agent Layer and business SOP execution layer. These three separate layers each have different functions, and provide clear functional boundaries and extensible modular interface. Management of State Transitions: The Global State Container (GSC) is used to propagate cross SOP states and to enable parameters to be reused automatically from one task to another [13]. The six-tuple model allowed for uniquely defined modular responsibilities to the extent of having low-level coupling to sub-modules thereby creating practical implementation of the six-tuple model.

Recovery Engine: The Reasoning Machine for Diff aims to compare a state of a task against a State in the Global Memory. Once a difference has been detected, the Recovery Engine uses the quantifiable threshold of deviation to determine whether a data mutation occurred and therefore the Recovery Engine can either select a lossless recovery process or initiate a safe rollback process based on a user-defined quantifiable threshold of difference [14]. The use of a quantifiable threshold to make a decision has removed the subjectivity from the decision-making process preventing semantic rollback attacks through quantifiable measurement of the State Differences [15,16,17].

Task Scheduling: The ability to carry out dynamic multi-task scheduling combines HTN planning [18] with priority queue management, which allows for high-dimensional user intention to be broken down into hierarchical Task queues made up of multiple SOPs. The framework also provides support to automatic task suspension/switching/continuity. The propagation of parameter prerequisites via the Global State Container allows for "silent transitions" during task switch that creates task workflow continuity across complex financial environments [19,20].

2.2.2. Differences from Existing Technologies

This framework, in contrast to the other algorithms that are currently usable on the market (Table 1), has four dimensions of difference between existing algorithms and this framework [21,22]—see below:

Global State Management: The Global State Container provides six-tuple global state management of all cross-SOP states, semantic memory, and episodic memory, as compared to the localized state confinement by FSMs and single-graph checkpoints. Global State Container's six-

tuple representation is used to facilitate cross-SOP state propagation. FSMs' traditional localized representation is constrained within individual workflows while LangGraph Checkpointer states are constrained within each thread's single-thread local graph.

Differentiated Recovery Mechanism: The proposed algorithm employs the Diff(·) function to compute a semantic-level comparison and inference of user-intent within the current environment and any underlying changes made to the data, allowing it to perform a different recovery strategy based on user intent or data changes. Traditional algorithms can only provide hard rollbacks or require manual intervention.

Dynamic Multi-SOP Scheduling: Based on HTN and priority queue methods, the proposed system supports task suspension/insertion/automatic continuity within multi-SOP scheduling. Conversely, the existing static DAG orchestration first at creation and then at the time of processing lacks the ability to support dynamic interleaving of multi-SOP tasks.

End-to-End Efficiency Optimization: The experimental analysis conducted demonstrates that the proposed algorithm has demonstrated superior performance metrics, when compared with existing mainstream algorithm baseline methods; such as Rate of Repeated Information Input, Cross-SOP Recovery Success Rate, and Task Completion Time. The information sharing capability/utilization capability provided by Global State Container will resolve the information silo issue that has limited the operational effectiveness of existing traditional algorithms.

Table 1: Capability Comparison of Mainstream Interruption-Resumption Algorithms.

Algorithm / Capability	Cross-SOP Transparent State Propagation	Semantic-level Resumption	Divergence Reasoning	Safe Rollback	Intelligent Context Bridging
Traditional FSM (Finite State Machine)	×	×	×	×	×
LangGraph	×	○	×	○	×
Rule Engine	×	×	×	×	×
MemGPT	○	○	×	×	×
The Proposed Algorithm	√	√	√	√	√

2.2.3. Summary of Algorithmic Advantages and Innovations

In summary, three main benefits and innovations are highlighted from the implementation of the proposed Cross SOP Re-Use Algorithm:

Semantic-Aware Recovery Capability: Cross-SOP-Information Re-use can be achieved through an automatic injection of key entities from the Global State Container into each of the SOP tasks performed by your system. This reduces the user interaction burden of having to enter repeated information by a factor from 85.0% down to 12.3%.

Semantic-Aware Recovery Capability: The differentiated reasoning mechanism provides the capability of automatically recovering from complex data mutations and intent switching related to a particular workflow. As such, the ability for workflows to continue securely and without any loss, due to a recovery has been significantly increased to 96.3%. As a result of establishing and maintaining formal measurement and evaluation criteria based on Discrepancy Measurement Functions and Dynamic Threshold Evaluation, this new recovery process has transitioned from a

traditional Guess & Retry model to a Reasoning Driven Decision Making model.

Multi-Task Continuity Assurance: The deep level of integration between HTN Scheduling and Global State Management allows for the automatic suspension/switching/resuming of all SOP Task Types and, therefore, the ability to maintain a high degree of workflow continuity. Completion Time for a given SOP Completion Process has decreased by 52.4% from initial time estimates, while acceptance of Chained Workflows' Continuity has increased from an average of 15.0% to an average of 91.0%.

3. Methodology

3.1. Architectural Overview and Dual-State Decoupling Mechanism

The entire framework, in combination with an interruption-resumption algorithm based on knowledge of the system at a global state level and dynamic checkpoint continuity technique, is designed to eliminate the known limitations of current systems.

The unique aspect of this design is that both deterministic execution of business SOPs (entirely determined by a finite and deterministic transition) and nondeterministic planning of a coordinator agent's intent will function independently of each other, with neither affecting the other's performance. The system is organized into three functional layers with a high degree of decoupling from one another (Figure 1).

Micro-Frontend - Multimodal Interaction Layer - This layer is the actual interface of the system and is responsible for the reception and processing of complex user input, abnormal feedback signals, and interaction between the user and interface in real time.

Coordinator Agent Layer - This layer represents the "cognitive orchestration hub" of the whole framework. The focus of the coordinator agent will be on high-level user-oriented interaction contexts and utilize the abstraction of all complex lower-level tasks associated with data validation and low-level API encapsulation, as well as checking the results of the previous two layers. Examples/elements of the Coordinator Agent Layer include: high-level intent topology (i.e., the overall structure of what the user has requested); dynamic SOP graph distribution; and the state container is used for maintaining coordinated activity among the various Agent layers.

The Business SOP Execution Layer or Slave Agents (Tools) - A number of unique and highly specialized subordinate SOPs will be created for execution in accordance with specific business-related activities and include categories such as Claims Verification and Bank Card Authentication.

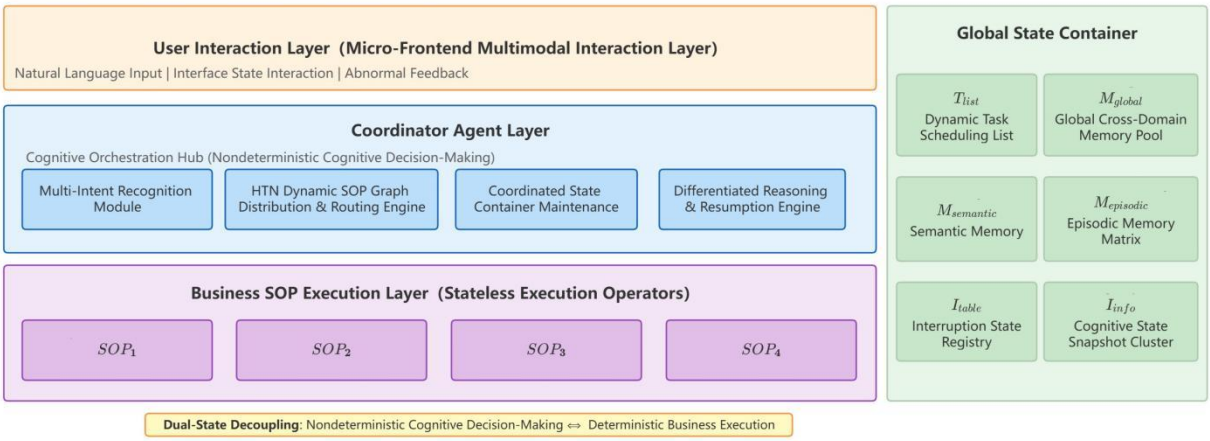

Figure 1: Global State-Aware Multi-SOP Collaborative Architecture.

A System Processor (SP) is the formal expression of a discrete representation of all Business

Domains to be processed.

A System Processor (SP) is the formal expression of a discrete representation of all Business Domains to be processed.

Each business SOP_i is rigorously modeled as a Directed Acyclic Graph (DAG) containing finite states and transition logic:

$$SOP_i = (V_i, E_i) \quad (1)$$

Where:

- SOP_i: the i-th Standard Operating Procedure (e.g., dividend withdrawal, survival benefit disbursement, policy loan)
- V_i: the vertex set of the SOP, where each vertex $v_k \in V_i$ corresponds to an atomic operational step in the business workflow (e.g., facial recognition verification, dynamic verification code request)
- E_i: the directed edge set of the SOP, where each edge $e_{jk} \in E_i$ represents the strict prerequisite dependency between workflow steps (i.e., step j must be completed before step k)

The entire business domain can therefore be formally represented as the union of all SOP graphs:

$$\Omega_{SOP} = \bigcup G_i, \quad i \in [1, n] \quad (2)$$

Where:

- Ω_{SOP}: the complete insurance business domain containing all supported SOP types
- G_i: the DAG corresponding to the i-th SOP (i.e., SOP_i in Formula (1))
- $\bigcup G_i$: the union operation that merges all independent SOP graphs into a unified business-domain set
- n: the total number of SOPs in the system

The dual-state decoupled design described previously allows for deterministic compliance boundaries (i.e. fixed SOP graph transitions) to be fully decoupled from human-machine interaction (i.e. the Coordinator Agent's semantic reasoning). Consequently, the underlying SOP nodes do not maintain memory states or cross-turn memory and therefore, function solely as stateless execution operators. The design of dual-state decoupled design will result in reduced computational overhead for orchestrating workflows and easier debugging of the system.

3.2. Multidimensional Mathematical Construction of the Global State Container

Cross-SOP execution between workflows G_i and G_j requires state propagation across individual workflow boundaries [23,24]. The framework introduces a Global State Container that transcends lifecycle boundaries of any single workflow.

As the Single Source of Truth for all cross-SOP transitions and parameter injections, the container C is formally defined as a multidimensional coupled six-tuple structure:

$$C = \langle T_{list}, M_{global}, M_{semantic}, M_{episodic}, I_{table}, I_{info} \rangle \quad (3)$$

The mathematical semantics and operational mechanisms of each submodule are defined as follows:

1) T_{list}: Dynamic Task Scheduling List

This module maintains the queue of SOPs pending execution within the current session.

Its state space is defined as:

$$T_{list} = \{(id, SOP_i, status, priority) | status \in \{pending, active, interrupted, completed, deferred\}\} \quad (4)$$

Equation (4) defines the task entry schema for T_{list}.

Where:

- id: unique task identifier
- SOP_i: associated Standard Operating Procedure
- status: task execution status, whose value space includes: pending, active, Interrupted, Completed, deferred
- priority: task priority used for scheduling order determination

2) M_{global}: Global Cross-Domain Memory Pool

This module stores critical business entities shared across SOPs using a structured key-value architecture.

Built upon a high-concurrency key-value database, the module persistently stores reusable business entities extracted during SOP execution, including: identity credentials, monetary amounts, authentication identifiers, and other shared contextual parameters.

When SOP_A completes validation and writes core entities into the memory pool, and SOP_B is subsequently activated, the system automatically injects reusable entity parameters into the corresponding nodes of SOP_B according to the dependency attribute tree.

3) $M_{semantic}$: Semantic Memory

This module extracts and persistently preserves desensitized user interaction preferences and behavioral patterns.

4) $M_{episodic}$: Episodic Memory Matrix

This module records the complete event sequence of the current session and provides causal evidence for recovery-time response generation and safe rollback decisions.

5) I_{table} : Interruption State Registry

The interruption state registry records the IDs and status labels of all suspended SOP instances.

It functions as a rigorously structured system scheduling ledger that tracks all SOP instances whose execution privileges have been explicitly revoked or deeply suspended due to I/O waiting conditions.

The state label space is formally defined as follows:

$$I_{table} = \{(sop_{id}, label, timestamp) | label \in \{INTENT\_SWITCH, IO\_BOCK, TIMEOUT, COMPLIANCE\_HALT\} \quad (5)$$

Equation (5) defines the record schema and label types for I_{table}.

Where:

- sop_{id}: unique identifier of the interrupted SOP
- label: interruption type label, whose value space contains four categories:
- INTENT_SWITCH: proactive user intent switching
- IO_BLOCK: I/O blocking (e.g., external API timeout)
- TIMEOUT: session timeout expiration
- COMPLIANCE_HALT: compliance interception triggered by risk-control rules
- timestamp: the timestamp indicating when the interruption occurred

6) I_{info}: Cognitive State Snapshot Cluster

This module is used to preserve fine-grained contextual snapshots at interruption time.

When the system determines that traversal of graph G_i must be immediately suspended, the module captures and stores the fine-grained contextual snapshot surrounding the current node v_k:

$$I_{info}(sop_{id}) = \left\langle S_{G_i}(v_k), \mathcal{L}_{v_k}, \tau_{freeze} \right\rangle \quad (6)$$

Equation (6) formalizes the snapshot structure for I_{info}, precisely preserving business execution

context at the moment of interruption.

Where:

- $I_{info}(sop_{id})$: snapshot entry indexed by the SOP identifier
- $S_{G_i}(v_k)$: the state vector of graph G_i at node v_k, including the current DAG execution position
- $\mathcal{L}_{v_k}$: the accumulated transient local variable set within the dialogue slots at node v_k(e.g., identity information, bank card numbers, etc.)
- τ_{freeze}: the freeze timestamp recording the exact interruption moment

3.3. Dynamic Interruption and In-Situ Resumption Algorithm

Building on the dual-state architecture and Global State Container, the framework implements a dynamic interruption and in-situ resumption pipeline [25,26]. The algorithm proceeds through four phases (Figure 2): intent detection and state freezing, differentiated recovery strategy reasoning, stateless node execution with global injection, and dynamic automatic task continuity.

Phase 1: Intent Detection and Interruption Freezing

When the control engine drives an SOP traversal to a specific node, the Coordinator Agent monitors for emergent abnormal signals.

If deterministic external intent switching or long-duration I/O blocking is detected, the system immediately triggers a forced suspension mechanism.

The current DAG state vector and local variables are encapsulated into immutable snapshot slices and pushed into the I_{info} snapshot cluster.

Simultaneously, the interruption registry I_{table} updates the corresponding status labels, strips the execution pointer from the current workflow, and dynamically routes execution toward the target workflow network.

Phase 2: Differentiated Recovery Strategy Reasoning

When the inserted target task is completed or the user expresses an intention to return to the suspended task, the system precisely retrieves and thaws the suspended state from the snapshot stack. The Diff Reasoning module then compares the snapshot entities against the latest data stored in global memory. If all prerequisite dependencies remain intact, the execution engine performs a lossless recovery at the original node. If substantive data changes are detected, the large language model generates corrective dialogue grounded in episodic memory and safely rolls back the workflow node.

Phase 3: Stateless Node Execution and Global Injection

Each SOP node executes in a stateless manner.

After execution, newly extracted business entities are structured and injected into the global memory pool M_{global} for subsequent workflow reuse.

Phase 4: Dynamic Automatic Task Continuity

After the current SOP is completed, a CHAIN_EVAL event triggers the task scheduling module to scan the remaining pending queue. The system extracts the user’s original high-level intent from episodic memory and proactively generates continuity prompts that combine personalized guidance with information reuse, thereby enabling “silent transitions” across multiple SOPs.

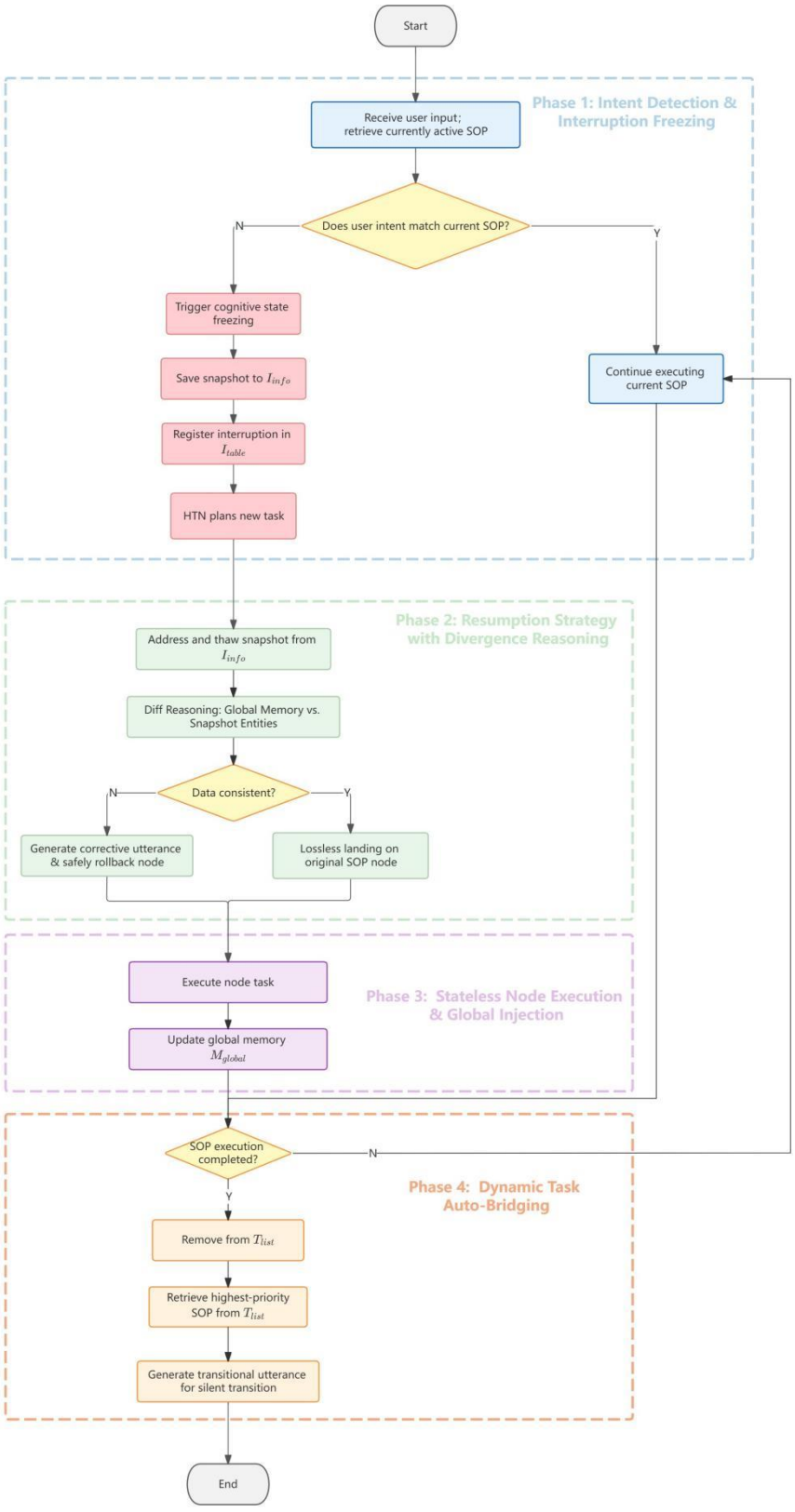

Figure 2: Workflow of the Dynamic Interruption–Resumption Agentic Algorithm.

3.3.1. Intent Switching and Cognitive State Freezing Protocol

When the primary workflow engine drives SOP G_i to traverse node v_k, and the micro-frontend captures emergent abnormal signals, the Coordinator Agent activates a classifier to determine the interruption type I_{table}.

If deterministic external intent switching or long-duration I/O blocking is identified, the system immediately triggers the forced suspension mechanism.

Definition 1 (Interruption Type Classification)

The interruption event set Delta is strictly partitioned into the following mutually exclusive subsets:

- INTENT_SWITCH: proactive intent switching
- IO_BLOCK: I/O blocking
- TIMEOUT: timeout expiration
- COMPLIANCE_HALT: compliance interception

The snapshot generation and encapsulation process can be formally represented through the cognitive state freezing function Φ_{freeze}:

$$\Phi_{freeze}(G_i, v_k) \rightarrow I_{info};\ label(G_i) \leftarrow SUSPENDED\ in\ I_{table} \quad (7)$$

Equation (7) formalizes the cognitive state freezing operation, capturing how the system persists the current execution state into the Global State Container upon interruption detection.

Where:

- $\Phi_{freeze}(G_i, v_k)$: the state-freezing function, representing one of the core operations of the system
- G_i: the currently executing SOP graph
- v_k: the current execution node
- $\rightarrow I_{info}$: stores the current state vector, local variables, and timestamps into the snapshot cluster
- $label(G_i) \leftarrow SUSPENDED\ in\ I_{table}$: simultaneously marks the corresponding SOP as "suspended" in the interruption registry

Pointer stripping and dynamic routing transfer are executed concurrently.

The system updates the state of SOP_i in T_{list} to either interrupted or deferred, updates the corresponding marker of G_i in I_{table} to "suspended," revokes its computational resource allocation, and redirects the execution pointer to the target workflow network G_j.

3.3.2. Differentiated Reasoning Mechanism for Recovery Strategies

When the inserted high-priority task is completed, or when the user expresses an intention to return to the suspended task G_i through natural language interaction, the system activates a context-aware Resumption Reasoning Mechanism.

Definition 2 (Diff Reasoning Function)

Let the thawed snapshot state be denoted as θ_{thawed}, and let the current global memory state be denoted as $M_{global}^{current}$.

The Diff Reasoning function is defined as:

$$Diff\left(\theta_{thawed}, M_{global}^{current}\right) = \sum w_i \cdot d\left(e_i^{\theta}, e_i^{M}\right) \quad (8)$$

Equation (8) measures the discrepancy between the frozen snapshot state and the current global memory state during workflow recovery.

Where:

- θ_{thawed}: the restored state data thawed from snapshot I_{info}
- $M_{\text{global}}^{\text{current}}$: the latest state stored in the global memory pool
- w_i: weight coefficient of entity eie_iei, reflecting its importance within the business process
- $d(e_i^{\theta}, e_i^{M})$: entity discrepancy measurement function used to calculate differences between two versions of an entity
- e_i^{θ}: the value of the iii-th entity in the snapshot
- e_i^{M}: the current value of the corresponding entity in global memory
- $\sum$: weighted discrepancy accumulation across all shared entities

Decision Logic:

- If $\text{Diff}(\cdot) < \epsilon_{\{\text{threshold}}$, the execution engine performs a lossless recovery at node v_k or seamlessly continues to node v_{k+1}.
- If $\text{Diff}(\cdot) \geq \epsilon_{\text{threshold}}$, the large language model generates corrective dialogue responses based on episodic memory and safely rolls back the workflow to an appropriate node.

The recovery policy guards against erroneous replay and semantic rollback by grounding decisions in state discrepancy measurement rather than heuristic retry.

Pseudocode for the Core Scheduling Algorithm

The algorithm describes the complete scheduling logic for dynamic multi-SOP routing and cognitive state freezing–thawing.

The algorithm accepts user query U and global state container C as inputs and outputs the final action response.

The overall process consists of four execution stages: intent detection, recovery reasoning, stateless execution, and automatic continuity.

The algorithm of Dynamic Multi-SOP Routing with Cognitive State Freezing and Thawing is as follows:

```
Input:
  User Query U,
  Global State Container C
Output:
  Final Action Response to User

  function CoordinatorAgent(U, C):
    current_sop = C.T_list.get_active_task()
  // Phase 1: Intent Detection and Interruption Freezing
  intent_vector = DetectIntent(U, current_sop.context,
C.M_semantic)

  if intent_vector != current_sop.expected_intent_boundary:
      // Trigger Cognitive State Freezing
            frozen_snapshot = CaptureSnapshot(current_sop.node,
current_sop.local_vars)
            C.I_info.push(current_sop.id, frozen_snapshot)
            C.I_table.add(current_sop.id,
label="External_Intent_Interrupt")
            // Route to newly identified SOP or handle Chit-chat
      target_sop = PlanNewHTNTask(intent_vector, C.M_global)
            C.T_list.insert_front_priority(target_sop)
            current_sop = target_sop
```

```
        // Phase 2: Trace-grounded Resumption Reasoning
        if C.I_table.contains(current_sop.id):
            // Thaw the state and diff with current global reality
            thawed_snapshot = C.I_info.pop(current_sop.id)
            resume_node = DiffReasoning(thawed_snapshot, C.M_global,
C.M_episodic)
            current_sop.set_execution_pointer(resume_node)
            C.I_table.remove(current_sop.id)

        // Phase 3: Stateless Node Execution and Global Injection
        response, new_entities, status =
ExecuteStatelessNode(current_sop, C.M_global)
        UpdateGlobalMemory(C.M_global, new_entities)


        // Phase 4: Dynamic Task Auto-transition
        if status == "COMPLETED":
            C.T_list.remove(current_sop)
            next_task = C.T_list.get_highest_priority_pending()
            if next_task:
                response += generate_transition_prompt(next_task,
C.M_episodic)

        return response
```

3.3.3. Cross-Task Information Inheritance and Data Field Conflict Resolution

When multiple highly correlated yet rule-divergent SOPs operate simultaneously within the same contextual space, incorrect data inheritance can easily occur.

Recent research has identified this issue as Cross-Task State Pollution (UCC).

Different insurance SOPs impose divergent validation standards on identical data fields. The coordination layer enforces Field-Level Inheritance Policies that prevent cross-task state pollution during intent transitions. The inheritance policies are categorized as follows (Table 2).

Table 2: Field-Level Inheritance Policies.

Field Inheritance Policy	Mechanism Semantics	Financial Domain Examples	Global Container Handling Rule
Inheritable (Unconditional)	Underlying attributes exhibiting cross-lifecycle consistency and globally unique static properties.	User identity numbers, global contract identifiers, identity authentication hash markers.	Silently propagated across task workflows without generating additional customer interactions.
Overrideable (Conditional)	Data entities with extended temporal validity but subject to real-time user modification.	Settlement bank accounts, default contact information, backup mailing addresses.	Auto-prefilled during workflow transitions; the LLM generates mandatory confirmation dialogue for secondary user verification.

Isolated (Non-Inheritable)	Sensitive contextual parameters tightly coupled with individual SOP executions.	Specific loan application amounts, exact dividend transfer ratios.	Snapshot registers forcibly cleared before new SOP extraction; reuse prohibited; dedicated inquiry procedures reinitiated.

3.3.4. Dynamic Multi-Task Planning and Hierarchical Collaboration

In composite scenarios such as “dividend withdrawal + survival benefit combined processing,” a single dialogue session may require planning and coordinating multiple objectives simultaneously.

At the underlying implementation level, the task management module adopts planning concepts inspired by Hierarchical Task Networks (HTN) [24].

Macro-Level Task Generation

The Coordinator Agent receives high-dimensional user requests and generates an S-DAG (subtask graph) sequence with prioritized ordering based on HTN logic, which is subsequently pushed into T_{list}.

Micro-Level State-Aware continuity

When an SOP either completes normally or encounters deep interruption, the system does not passively wait for further user input.

Instead, it proactively extracts the next high-priority task from T_{list}.

By leveraging the state-sharing mechanism of M_{global}, the framework achieves "Silent Transition" across tasks, improving continuity and resource scheduling efficiency under concurrent multi-SOP execution scenarios.

4. Experimental Design

To comprehensively evaluate and validate both the theoretical completeness and practical commercial value of the proposed “multi-SOP intelligent scheduling and dynamic interruption–resumption global state framework” [27,28], the research team constructed a closed conversational sandbox environment based on extracted real-world business corpora.

4.1. Experimental Sandbox Design and Baseline Comparisons

The test dataset was derived from deeply desensitized real digital insurance business conversation logs, resulting in the construction of 500 high-frequency and highly complex adversarial test cases involving composite intents. Each simulated interaction session concurrently interleaved two to four standard business workflows, including dividend withdrawal and policy loan processing.

To evaluate architectural robustness under extreme conditions, the experiments introduced varying frequencies of proactive intent switching, passive API-level service interruptions, and off-topic semantic interference.

Two baseline architectures were selected for comparison:

Baseline System 1 — Traditional Single-Thread Rule-Based Agent Architecture: This architecture represents the mainstream production deployment model currently adopted by many financial institutions. It lacks cross-task memory bridges and persistent checkpoint mechanisms, causing interruptions to trigger process collapse or execution deadlock.

Baseline System 2 — Single-Graph Persistent Framework Based on LangGraph Checkpoints: This framework relies on the mainstream LangGraph Checkpointer mechanism. Although it

supports time-travel debugging and restoration within a single-graph, all states remain locked within a specific thread graph and therefore lack support for a cross-boundary global state container.

4.2. Comparison and Quantitative Analysis of Core Evaluation Results

To precisely characterize the framework's ability to reduce information entropy, improve scheduling stability, and enhance service continuity, the following multidimensional metrics were defined:

Interaction Denoising Rate: The proportion of redundant requests for factual information already provided by users due to state isolation.

Cross-SOP Resumption Rate: The success rate at which the system can recover suspended workflows without reset or deadlock after interruptions and business-domain switching.

Chain Acceptance Rate: Measures whether the intelligent Coordinator Agent can proactively recommend the next queued business process based on the global HTN topology and successfully obtain user continuity.

End-to-End JCT (minutes): The total elapsed time from the user's first interaction to the completion of all requested business combinations.

The comprehensive adversarial testing results were aggregated and normalized into a comparative benchmarking matrix containing the following dimensions (Table 3).

Table 3: Adversarial Test Results Summary.

Core evaluation metrics	Baseline System 1 (traditional stateless single-thread execution)	Baseline System 2 (LangGraph single-graph snapshot-based architecture)	Proposed method (Global State multi-SOP container architecture)
Repeated interaction information extraction rate	85.0%	62.5%	12.3%
Cross-SOP rapid recovery success rate	0%	0%	96.3%
High-priority task chain continuity success rate	N/A	N/A	73.5%
Average end-to-end completion time for composite objectives	24.6 min	19.8 min	11.7 min
Overall user experience score (5-point scale)	2.8	3.4	4.5

4.3. In-Depth Analysis of Experimental Results

Experimental results are as follows:

4.3.1. Eliminating Data Silos and Achieving Interaction Denoising

The first metric demonstrates that Baseline System 1 exhibits an 85% repetition rate in multi-SOP scenarios due to its closed execution logic encapsulation. The checkpointing mechanism adopted by Baseline System 2 reduces the repetition rate to 62.5%. By contrast, the framework leverages the global cross-domain memory pool M_{global} as a hub, enabling unobstructed propagation of core business entities and reducing the repeated information extraction rate to 12.3%.

4.3.2. Cross-Domain Transition Capability and Unbreakable Workflow Resilience

Under extreme stress-testing conditions, Baseline System 1 and Baseline System 2 completely collapsed when encountering intent migration due to the limitations of their one-way state rollback designs, resulting in recovery rates of 0%. In contrast, the proposed Coordinator Agent architecture, equipped with a structured interruption registry and frozen snapshot stack (I_{table} and I_{info}), achieved graceful suspension and lossless callback recovery, establishing a workflow resilience benchmark with a 96.3% recovery success rate.

4.3.3. Computational Efficiency Reconstruction and Concurrent Workflow Coordination

The overall completion time of the composite testing environment was reduced from nearly 25 minutes to 11.7 minutes (a performance improvement of 52.4%). This result not only demonstrates the system's capability to reduce workflow restart overhead, but also validates the effectiveness of the proposed implicit multi-task continuity queues and Silent Transition parameter propagation mechanism. These capabilities fundamentally surpass the rigid DAG-style workflow transitions used in traditional orchestration systems and demonstrate a globally coordinated workflow orchestration capability.

5. Discussion

We investigate the core challenges of collaborative multi-SOP execution in complex business service scenarios [29,30] and present an interruption-resumption architecture based on global state awareness and dynamic checkpoint continuity.

The architecture introduces dual-state decoupling that separates nondeterministic cognitive decision-making from deterministic business execution. This design yields a three-layer system comprising a micro-frontend multimodal interaction layer, a Coordinator Agent layer, and a business SOP execution layer.

The six-tuple Global State Container operates as the single source of truth for cross-SOP information transfer. It overcomes localized state isolation inherent in FSMs and single-graph checkpoints, enabling automatic cross-workflow reuse of key business entities and silent propagation of prerequisite parameters.

The recovery pipeline proceeds through four phases: intent detection and state freezing, differentiated recovery strategy reasoning, stateless node execution with global injection, and dynamic automatic task continuity. Diff Reasoning compares weighted state discrepancies between frozen snapshots and current global memory to dynamically determine lossless recovery or safe rollback paths.

Experimental validation based on 500 high-frequency adversarial composite-intent test cases demonstrates that the proposed algorithm reduces repeated information input rates from 85% to 12.3%, achieves a cross-SOP recovery success rate of 96.3%, shortens end-to-end completion time from 24.6 minutes to 11.7 minutes, and improves overall user experience scores from 2.8 to 4.5.

From the perspective of customer experience optimization, the proposed framework introduces a

cross-process seamless continuity mechanism. Supported by the Global State Container, information only needs to be entered once and can be automatically reused across workflows, thereby eliminating repeated identity verification and document submission procedures. When service processes are interrupted unexpectedly, the system accurately restores execution states through dynamic checkpoint continuity. When users switch business objectives, workflows can be suspended and resumed at any time through global state synchronization.

Future research directions may include the following aspects:

Strengthening memory security in high-concurrency environments by exploring deeper encryption cleansing and access isolation mechanisms for global shared memory systems.

Developing intelligent predictive suspension and autonomous combinational reasoning by introducing graph neural networks to model nonlinear and topology-free ultra-large SOP networks.

Integrating multimodal perception and affective computing by incorporating speech and image modalities into the Global State Container and dynamically adjusting recovery strategies according to user emotional states.

Constructing personalized customer experience evaluation systems by introducing subjective user perception metrics such as fluency, trust, and service willingness, thereby enabling a paradigm shift from “system optimization” toward “experience optimization.”

As cognitive infrastructures continue to evolve, globally state-aware agent architectures driven by large language models will play an increasingly critical role in reshaping seamless, self-regulated, and seamless collaborative intelligent services.

6. Conclusion

This paper presents a globally state-aware interruption–resumption agentic algorithm for complex multi-SOP business workflows. By introducing dual-state decoupling, a six-tuple Global State Container, and a Diff Reasoning mechanism, the framework enables seamless cross-SOP recovery and eliminates information silos inherent in traditional systems. Experimental validation on 500 real-world insurance adversarial test cases demonstrates that the proposed approach reduces repeated information input from 85% to 12.3%, achieves a 96.3% cross-SOP recovery success rate, shortens end-to-end completion time by 52.4%, and improves user experience scores from 2.8 to 4.5. These results confirm that global state awareness combined with dynamic checkpoint continuity provides a robust, scalable foundation for next-generation intelligent customer service systems in complex business domains.

References

[1] Wang L, Ma C, Feng X, et al. A survey on large language model based autonomous agents[J]. Frontiers of Computer Science, 2024, 18(6): 186345.

[2] Li G, Hammoud H, Itani H, et al. Camel: Communicative agents for" mind" exploration of large language model society[J]. Advances in neural information processing systems, 2023, 36: 51991-52008.

[3] Romero O J, Zimmerman J, Steinfeld A, et al. Synergistic integration of large language models and cognitive architectures for robust AI: an exploratory analysis[J]. Proceedings of the AAAI Symposium Series, 2024, 2(1): 396-405.

[4] Hatalis K, Christou D, Myers J, et al. Memory matters: The need to improve long-term memory in LLM-agents[C]//Proceedings of the AAAI Symposium Series. 2024, 2(1): 277-280.

[5] Hu M, Chen T, Chen Q, et al. Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model[C]//Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2025: 32779-32798.

[6] Gao P, Zhao J, Chen X, et al. An efficient context-dependent memory framework for llm-centric agents[C]//Proceedings of the 2025 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: Industry Track). 2025: 1055-1069.

[7] Qin Y, Liang S, Ye Y, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis[C]//International Conference on Learning Representations. 2024, 2024: 9695-9717.
[8] Ma C, Zhang J, Zhu Z, et al. Agentboard: An analytical evaluation board of multi-turn llm agents[J]. Advances in neural information processing systems, 2024, 37: 74325-74362.
[9] Reimann M M, Kunneman F A, Oertel C, et al. A survey on dialogue management in human-robot interaction[J]. ACM Transactions on Human-Robot Interaction, 2024, 13(2): 1-22.
[10] Zhang X, Dong X, Wang Y, et al. A survey of multi-ai agent collaboration: Theories, technologies and applications[C]//Proceedings of the 2nd Guangdong-Hong Kong-Macao Greater Bay Area International Conference on Digital Economy and Artificial Intelligence. 2025: 1875-1881.
[11] Sumers T R, Yao S, Narasimhan K, et al. Cognitive architectures for language agents[J]. Transactions on Machine Learning Research, 2024.
[12] Zhang Z, Dai Q, Bo X, et al. A survey on the memory mechanism of large language model based agents[J]. ACM Transactions on Information Systems, 2025, 43(6): 1-47.
[13] Liu X, Yu H, Zhang H, et al. Agentbench: Evaluating llms as agents[C]//International Conference on Learning Representations. 2024, 2024: 52989-53046.
[14] Xie T, Zhang D, Chen J, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments[J]. Advances in Neural Information Processing Systems, 2024, 37: 52040-52094.
[15] Shinn N, Cassano F, Gopinath A, et al. Reflexion: Language agents with verbal reinforcement learning[J]. Advances in neural information processing systems, 2023, 36: 8634-8652.
[16] Chen Z, Xiang Z, Xiao C, et al. Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases[J]. Advances in Neural Information Processing Systems, 2024, 37: 130185-130213.
[17] Zhao A, Huang D, Xu Q, et al. Expel: Llm agents are experiential learners[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2024, 38(17): 19632-19642.
[18] Xie J, Zhang K, Chen J, et al. Travelplanner: A benchmark for real-world planning with language agents[J]. arXiv preprint arXiv:2402.01622, 2024.
[19] Nau D S, Au T C, Ilghami O, et al. SHOP2: An HTN planning system[J]. Journal of Artificial Intelligence Research, 2003, 20: 379-404.
[20] Sun C, Huang S, Pompili D. LLM-based multi-agent decision-making: challenges and future directions[J]. IEEE Robotics and Automation Letters, 2025, 10(6): 5681-5688.
[21] LangChain. LangGraph: Stateful, graph-based orchestration for LLM agents[EB/OL]. https://github.com/langchain-ai/langgraph, 2024.
[22] Xi Z, Chen W, Guo X, et al. The rise and potential of large language model based agents: A survey[J]. Science China Information Sciences, 2025, 68(2): 121101.
[23] Zheng J, Qiu S, Shi C, et al. Towards lifelong learning of large language models: A survey[J]. ACM Computing Surveys, 2025, 57(8): 1-35.
[24] Croissant M, Frister M, Schofield G, et al. An appraisal-based chain-of-emotion architecture for affective language model game agents[J] . PLOS ONE, 2024, 19(5): e0301033.
[25] Wu D, Wang H, Yu W, et al. Longmemeval: Benchmarking chat assistants on long-term interactive memory[J]. arXiv preprint arXiv:2410.10813, 2024.
[26] Xu X, Gou Z, Wu W, et al. Long time no see! open-domain conversation with long-term persona memory[C]//Findings of the Association for Computational Linguistics: ACL 2022. 2022: 2639-2650.
[27] Portugal I D S, Alencar P, Cowan D. An agentic AI-based multi-agent framework for recommender systems[C]//2024 IEEE International Conference on Big Data (BigData). IEEE, 2024: 5375-5382.
[28] Lim J, Vogel-Heuser B, Kovalenko I. Large language model-enabled multi-agent manufacturing systems[C]//2024 IEEE 20th International Conference on Automation Science and Engineering (CASE). IEEE, 2024: 3940-3946.
[29] Hong S, Zhuge M, Chen J, et al. MetaGPT: Meta programming for a multi-agent collaborative framework[C]//International Conference on Learning Representations. 2024, 2024: 23247-23275.
[30] Park J S, O'Brien J, Cai C J, et al. Generative agents: Interactive simulacra of human behavior[C]//Proceedings of the 36th annual acm symposium on user interface software and technology. 2023: 1-22.