[251114] MAS(멀티 에이전트 시스템) 설계 서베이

요약

- **단일 에이전트 (Luo et al.)**: LLM 에이전트를 **Profile–Memory–Planning–Action** 4 모듈로 정의하며, 단일 Agent의 사고·행동 구조를 설명.
- **멀티 에이전트 시스템(MAS, Li et al./Chen et al.)**: MAS를 **Profile–Perception–Self-action–Mutual Interaction–Evolution** 5단계로 정립하고, 역할·상호작용·통신·도메인 응용 관점을 제시.
- **Self-Evolving Agent (Guo et al.)**: 에이전트가 **System Inputs–Agent System–Environment–Optimisers** 피드백 루프를 통해 프롬프트, 메모리, 툴, 워크플로, 구조까지 스스로 개선하는 프레임 제안.
- **설계 핵심 요소**: 프로파일 설계, 하이브리드 메모리, 계획(단일 경로/Multi-path/PDDL), Action·Tool 스키마, MAS 상호작용 구조(협업·토론·검증

| 0. 목적 및 개요

이 보고서는 다음 네 편의 서베이 논문을 바탕으로:

1. A Survey on Large Language Model based Autonomous Agents (*Luo et al., 2025,* 단일 LLM 에이전트)
2. A survey on LLM-based multi-agent systems: workflow, infrastructure, and challenges (Li et al., 2025, 워크플로·인프라 중심 MAS)
3. A Survey on LLM-based Multi-Agent System: Recent Advances and New Frontiers in Application (Chen et al., 2025, 응용·도메인 중심 MAS)
4. A Comprehensive Survey of Self-Evolving AI Agents (Guo et al., 2025, 자기진화 에이전트)

"AI 에이전트 설계자 관점"에서 공통된 설계 프레임을 추출하고, **단일 에이전트 → 멀티 에이전트 시스템 → 자기진화형 에이전트**까지 연속선 위에서 정리·재구성한 설계 지침을 제시합니다.

| 1. 각각의 서베이 논문이 제시하는 큰 그림

| 1-1. 단일 LLM 에이전트: 4모듈 프레임워크

*Luo et al.*의 Autonomous Agent 서베이는 LLM 기반 에이전트를 **Profile – Memory – Planning – Action** 4개 모듈로 통합 프레임을 제안합니다.

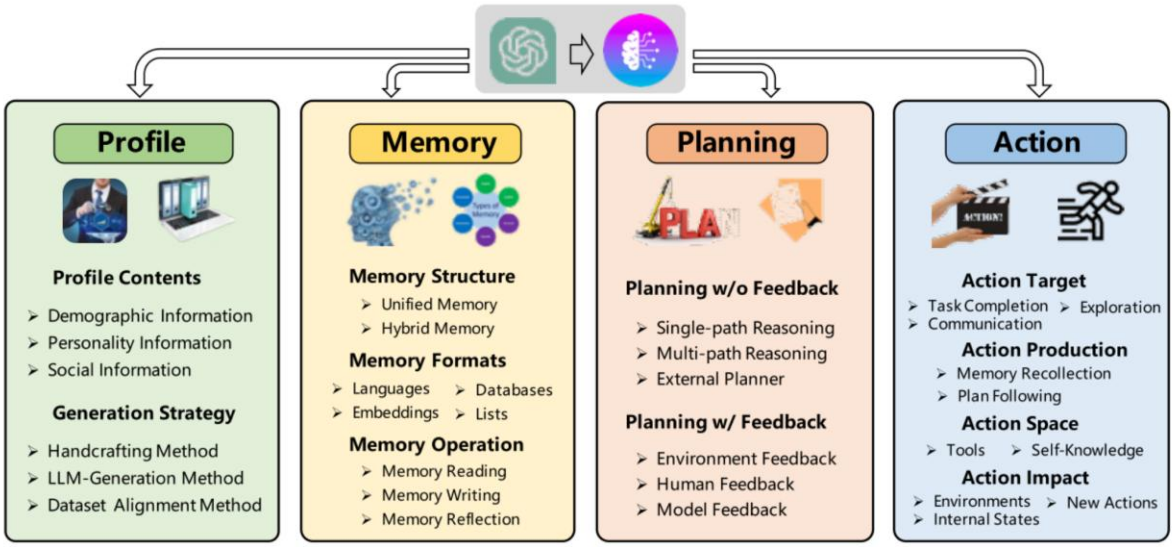


Fig. 2 A unified framework for the architecture design of LLM-based autonomous agent.

- **Profile**: 역할, 인격, 배경 정보(나이·직업·성격·관계 등)를 프롬프트에 encode하여 LLM 행동을 규정
- **Memory**: 읽기/쓰기/반성(reflection)을 포함하는 통합/하이브리드 메모리 구조
- **Planning**: 피드백 없이 단일 경로 추론 vs 피드백/외부 플래너를 활용한 계획
- **Action**: 도구(툴 호출/DB/API 등)와 환경 상호작용을 통해 계획을 실행

→ 이 프레임은 **"하나의 에이전트 인스턴스" 설계에 최적화**되어 있습니다.

| 1-2. LLM 기반 멀티에이전트 시스템: 5단계 워크플로

Li et al.의 MAS 서베이는 LLM-MAS를 **Profile – Perception – Self-action – Mutual Interaction – Evolution** 5개의 워크플로 모듈로 정리합니다.

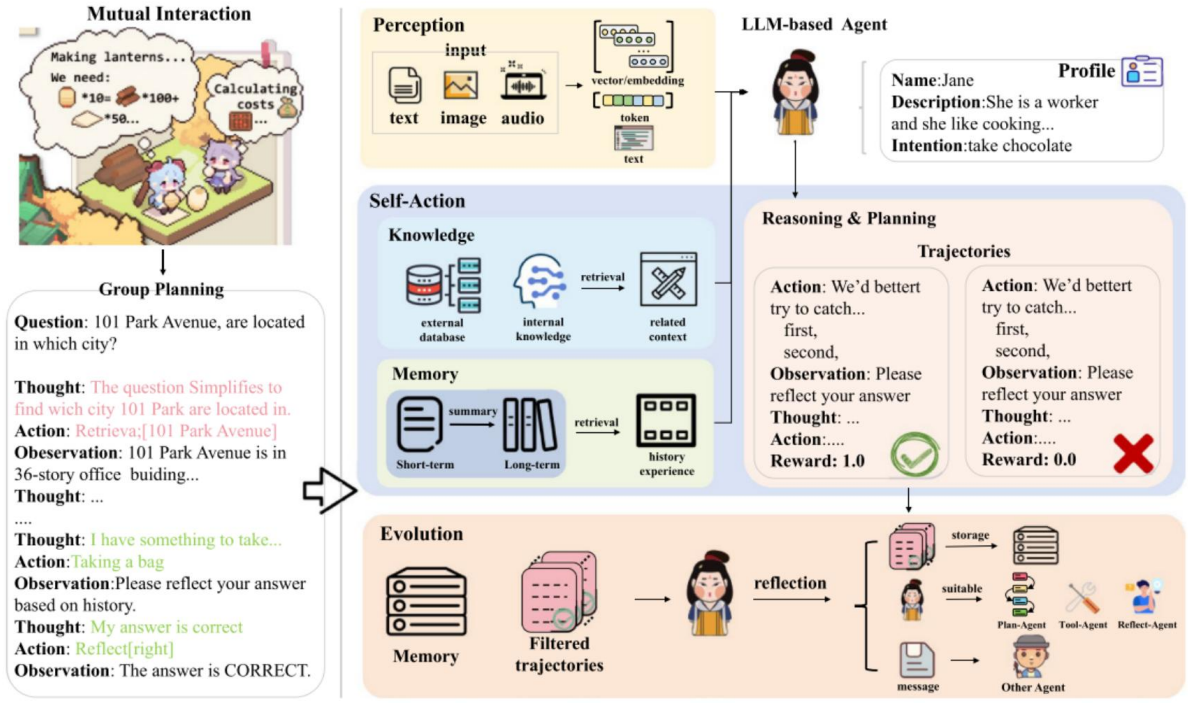


Fig. 1 Overview of the general multi-agent system. Typically, in a multi-agent system, the initial step involves the creation of profiles that endow each agent with personalized characteristics and subtask allocations. Based on the task planning, the agent formulates specific plans to perceive multi-modal information from the interactive environment, accesses external knowledge, and retrieves their historical experiences and knowledge from memory. Utilizing the profound abilities of LLMs, agents are able to devise concrete action plans. Simultaneously, agents engage in evolution, which involves the ongoing reflection on their decisions and actions. Throughout this process, the execution of tasks relies on the interactions among agents, which collectively contribute to the planning and implementation of the overall mission

- **Profile**: 역할·목표·능력·권한을 정의해 에이전트의 행동 스타일을 결정
    - 책임 분담을 위해 Planner / Worker / Critic / Aggregator 에이전트 등으로 구체화된다.
    - 입력: Domain 지식, 요구 기능 / 출력: 에이전트 설정(Profile JSON)
- **Perception**: 환경/유저/다른 에이전트의 멀티모달 정보를 구조화하여 인식
    - 환경 이벤트(log/state), 다른 에이전트 메시지, 자기 행동에 대한 Observation까지 모두 포함한다.
    - 입력: 유저 질의, 환경 상태, 타 에이전트 메시지 / 출력: 정제된 관찰 정보(observation)
- **Self-action**: 메모리·지식·툴을 활용해 추론→계획→행동을 선택
    - Retrieval, Tool 선택, 계획 생성, 응답 산출까지 포함한 "개별 에이전트의 두뇌" 역할.
    - 입력: observation + memory / 출력: action(툴 호출, 메시지, 계산 결과 등)
- **Mutual Interaction**: 여러 에이전트가 메시지 교환·토론·검증·합의를 수행

- 구조는 중앙집중/분산/계층/그래프형 중 선택하여 설계한다.
- 입력: 개별 에이전트의 action / 출력: 그룹 의사결정 결과(계획안, 검증통과, 합의안 등)

- **Evolution**: 보상·오류·피드백에 기반해 전략/역할/프롬프트/메모리를 지속 개선
  - 자기반성(Self-reflection), 타인 평가, 사용자 피드백까지 포함하여 장기적 성능을 향상시킨다.
  - 입력: reward/feedback/log / 출력: 개선된 프로파일·프롬프트·전략·메모리

→ 이 프레임은 **"에이전트 집단과 상호작용 구조" 설계에 초점**이 있습니다.

Chen et al.의 LLM-MAS 서베이는 멀티에이전트 시스템을 **적용 도메인 중심**으로 분류하며, 복잡 문제 해결·시뮬레이션·사회 시스템·정책/경제 모델링·에이전트 평가·훈련 등 다양한 활용 영역을 체계적으로 제시.

.

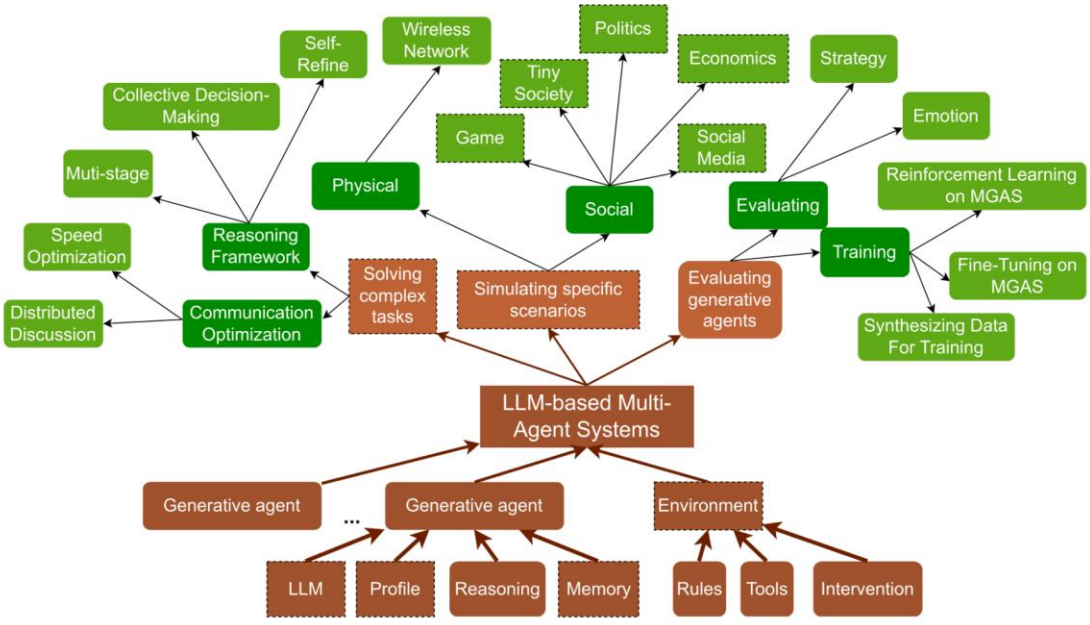


Figure 1: Overview of the application framework and relationship of LLM-MAS, generative agent, and LLM. Dashed-bordered right-angled rectangles represent content aligned with previous surveys, while rounded rectangles indicate original contributions introduced in this study.

- MAS가 적용되는 분야를 **추론·의사결정·커뮤니케이션 최적화·집단 협력**까지 확장된 능력 기반 영역으로 구조화한다.

- 게임·사회·정치·경제 등 **시뮬레이션 중심 도메인**에서 다중 에이전트 간 상호작용을 통해 복잡한 사회적 현상을 모델링한다.
- 생성형 에이전트 평가, 데이터 생성, RL 기반 훈련 등 **평가·학습 도메인**도 별도 축으로 분리해 MAS의 발전 로드맵을 제시한다.
- 환경(Environment), 규칙(Rules), 툴(Tools), 프롬프트, 메모리 등 MAS 구성 요소가 **각 도메인 요구에 맞게 재조합**되는 구조를 설명한다.
- 전체 그림은 LLM과 개별 에이전트, 그리고 환경이 **도메인별 Agentic System으로 통합되는 방식**을 보여주는 도메인 지도 역할을 한다.

| 1-3. Self-evolving 에이전트: 피드백 루프와 4요소

Self-Evolving 서베이는 "정적 프롬프트+워크플로" 한계를 지적하면서, **자기진화(Self-evolving) 에이전트**를 위해 다음 네 컴포넌트로 묶인 피드백 루프를 제안합니다.

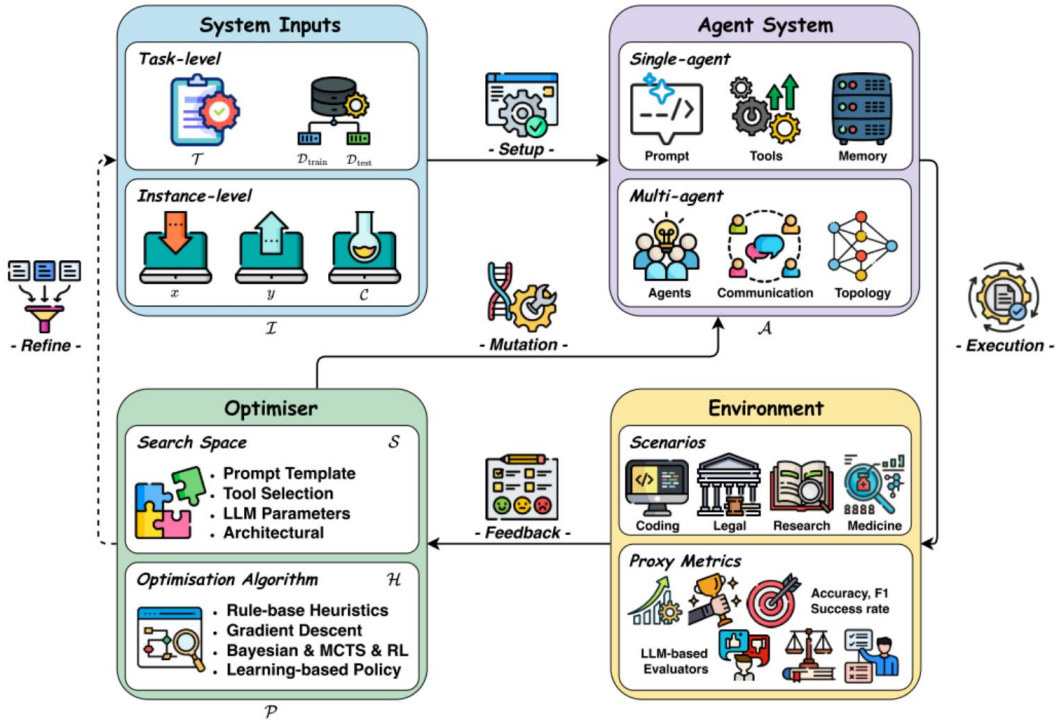


Figure 3 Conceptual framework of the self-evolving process in agent systems. The process forms an iterative optimisation loop comprising four components: *System Inputs*, *Agent System*, *Environment*, and *Optimiser*. System inputs define the task setting (e.g., task-level or instance-level). The agent system (in single- or multi-agent form) executes the specified task. The environment (depending on different scenarios) provides feedback via proxy metrics. The optimiser updates the agent system through a defined search space and optimisation algorithm until performance goals are met.

- **System Inputs**: 사용자 목표, 도메인 지식, 데이터
- **Agent System**: LLM, 메모리, 툴, 워크플로, 통신 메커니즘
- **Environment**: 시뮬레이션/실세계 시스템에서 오는 관측·보상·로그

- **Optimisers**: 프롬프트, 워크플로, 메모리 구조, 툴 셋, 에이전트 토폴로지를 자동 개선하는 모듈

또한 LLM 중심 학습 패러다임을 **MOP → MOA → MAO → MASE**로 정리합니다.

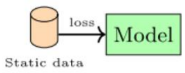
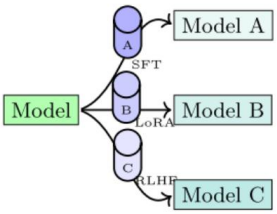
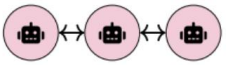
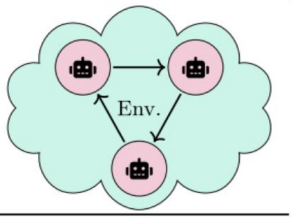
Paradigm	Interaction & Feedback	Key Techniques	Diagram
Model Offline Pretraining (MOP)	Model ⇔ Static data (loss/backprop)	• Transformer Pretraining (Causal LM, Masked LM, NSP) • BPE / SentencePiece • MoE & Pipeline Parallelism	
Model Online Adaptation (MOA)	Model ⇔ Supervision (labels/scores/rewards)	• Task Fine-tuning • Instruction Tuning • LoRA / Adapters / Prefix-Tuning • RLHF (RLAIF, DPO, PPO) • Multi-Modal Alignment • Human Alignment	
Multi-Agent Orchestration (MAO)	$Agent_1$ ⇔ $Agent_2$ (message exchange)	• Multi-Agent Systems • Self-Reflection • Multi-Agent Debate • Chain-of-Thought Ensemble • Function / Tool Calling / MCP	
Multi-Agent Self-Evolving (MASE)	Agents ⇔ Environment (signals from env.)	• Behaviour Optimisation • Prompt Optimisation • Memory Optimisation • Tool Optimisation • Agentic Workflow Optimisation	

Table 1 Comparison of four LLM-centric learning paradigms – Model Offline Pretraining (MOP), Model Online Adaptation (MOA), Multi-Agent Orchestration (MAO), and Multi-Agent Self-Evolving (MASE), highlighting each paradigm's interaction & feedback mechanisms, core techniques, and illustrative diagrams to trace the progression from static model training to dynamic, autonomous agent evolution.

- MOP(Model Offline Pretraining): 우리가 아는 일반 pretraining 모델을 가진 Agent
- MOA(Model Online Adaptation): SFT, RLHF 등 온라인 적응
- MAO(Multi-Agent Orchestration): 여러 LLM/에이전트의 협업·토론·툴 호출 (A2A, MCP)
- MASE(Multi-Agent Self-Evolving): 에이전트가 환경 신호를 이용해 자기 구조/정책까지 최적화

→ 이 논문은 "**에이전트를 누가 어떻게 계속 개선할 것인가**"에 포커스가 있습니다.

| 2. 단일 LLM 에이전트 설계를 위한 정리

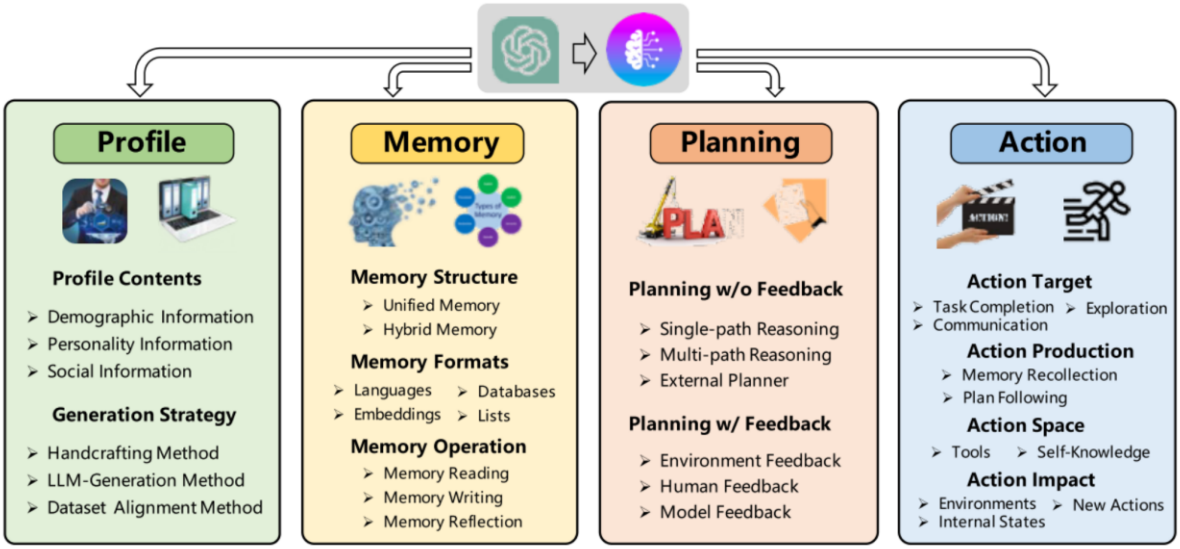


Fig. 2 A unified framework for the architecture design of LLM-based autonomous agent.

| 2-1. Profile 설계

Autonomous Agent 서베이와 MAS 서베이 모두 **프롬프트 기반 프로파일링**을 핵심 출발점으로 봅니다.

구성 요소

- 기초 정보: 이름, 역할/직무, 도메인, 전문성 레벨
- 심리 정보: 성향(보수적 vs 공격적), 말투, 협업 스타일
- 사회 정보: 상하 관계, 팀 구조, 책임 범위
- 제약: 하지 말아야 할 행동, 권한 범위

프로파일 생성 전략

1. **Handcrafting**: 사람이 직접 모든 역할 프롬프트를 설계
2. **LLM-generation**: "역할 생성 규칙"을 정의하고 LLM이 다수의 에이전트 프로파일을 생성
3. **Dataset alignment**: 실제 로그/대화/행동 데이터를 요약해 프로파일로 추출

| 2-2. Memory 모듈 설계

메모리는 에이전트의 "내적 상태"를 유지하고 진화를 가능케 하는 핵심입니다.

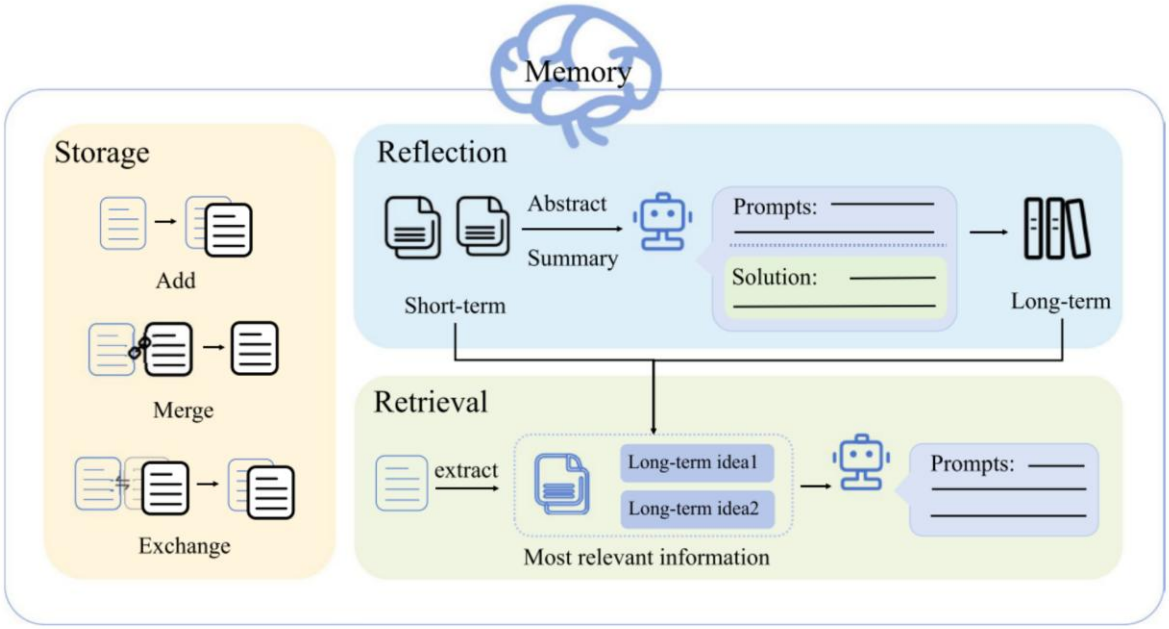


Fig. 2 The operational mechanism of the memory module

(1) 메모리 오퍼레이션

- Retrieval: Recency / Relevance / Importance 기반 자동 검색
- Storage: 성공/실패 경험, 요약 정보, structured triplet 등 저장
- Reflection: 여러 기록을 상위 개념으로 요약해 "교훈"이나 "규칙"으로 재구성

(2) 메모리 구조

- Unified memory: 하나의 저장소에 대화·이벤트·지식이 뒤섞인 형태
- Hybrid memory: 단기/장기, 에피소드/지식, 벡터DB/키밸류 DB 등으로 계층화

| 2-3. Planning 모듈 설계

서베이는 계획을 크게 **피드백 없는 계획** vs **피드백 기반 계획**(환경·인간·모델 피드백)으로 나눕니다.

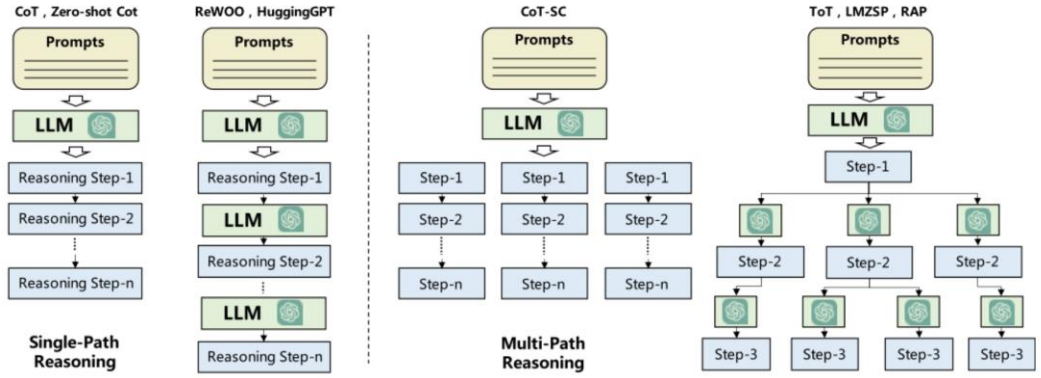


Fig. 3 Comparison between the strategies of single-path and multi-path reasoning. LMZSP is the model proposed in [44].

- **Single-path reasoning**: CoT 한 줄로 쭉 진행
- **Multi-path reasoning**: 다수의 계획을 생성–비교–투표
- **External planner**: PDDL/외부 플래너에 문제를 변환해 최적 경로를 찾고, 다시 자연어로 풀어씀
    - **PDDL (Planning Domain Definition Language)**: AI가 계획 문제를 이해하도록 만드는 표준 언어, 현재 상태(initial), 목표(goal), 가능한 행동(actions: 조건·효과) 정의

| 2-4. Action 및 Tool 사용 설계

Action 모듈은 "계획을 실제로 시스템에 반영"하는 부분입니다.

- Action target:
    - 사용자 답변, API 호출, DB 질의, n8n/EAIP 워크플로 호출 등
- Tool space:
    - 툴 목록, 파라미터 스키마, 호출 제약
- Execution policy:
    - 계획을 그대로 따를지(Plan following) vs 중간에 새로운 action을 synthesise 할지

| 3. 멀티 에이전트 시스템(MAS) 설계 포인트

| 3-1. 에이전트 프로파일과 역할 분할

Li et al.와 Chen et al.은 LLM-MAS의 핵심을 **"다양한 역할을 가진 에이전트들이 상호작용하며 집단 지능을 만든다"**라고 봅니다.

| 3-2. Perception: 메시지 소스와 모달리티

MAS에서 Perception은 단일 에이전트보다 더 복잡합니다.

메시지 소스

1. Environment message: 시간, 장소, 시스템 상태, 로그, 이벤트
2. Interaction message: 에이전트 간 대화, 명령, 협상 내용
3. Self-reflection message: 자기 행동·실패/성공에 대한 내적 서술

모달리티

- Text가 기본, Vision·Audio·센서 정보까지 확장하여 MLLM 기반 Perception 구성

| 3-3. Mutual Interaction: 통신 구조

Li et al.은 MAS 커뮤니케이션을 **패러다임, 구조, 콘텐츠**로 나눕니다.

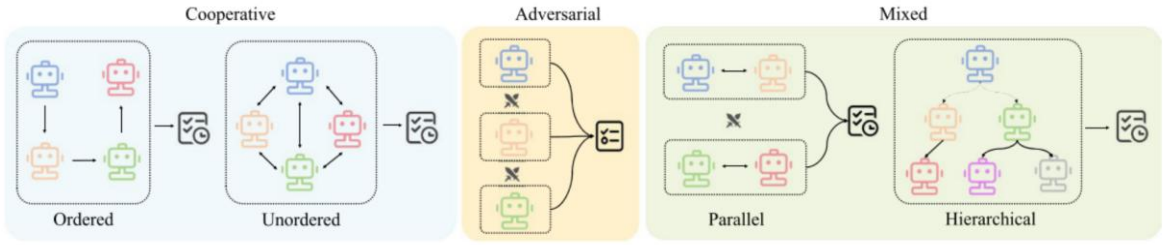


Fig. 5 The agent interaction scene

- Paradigm:
    - 협력/cooperative, 경쟁/adversarial, 혼합/mixed
- Structure:
    - 중앙집중형 (Coordinator/Manager 에이전트)
    - 분산형 (모두가 서로 메시지를 교환)
    - 계층형 (L1 Planner, L2 Worker 등)
    - 중첩형/그래프형 (부분집단/서브팀 존재)

- Content:
    - 자연어만 vs structured protocol (JSON, key-value)

| 3-4. Evolution: MAS 관점의 진화

MAS에서 Evolution은 **개별 에이전트 수준 + 시스템 수준(토폴로지/프로토콜)** 두 레벨입니다.

- Agent level:
    - 메모리 reflection으로 행동 규칙 업데이트
    - 역할 전환 또는 스페셜리제이션 심화
- System level:
    - 어떤 에이전트 조합/구조가 성능이 좋은지 탐색
    - 통신 프로토콜(메시지 길이, 형식, 빈도) 최적화

| 4. Self-Evolving AI Agent 설계 관점

Self-Evolving 서베이는 "현재 Agent 시스템의 끝판왕 요구사항"을 정의합니다.

| 4-1. 네 컴포넌트 피드백 루프

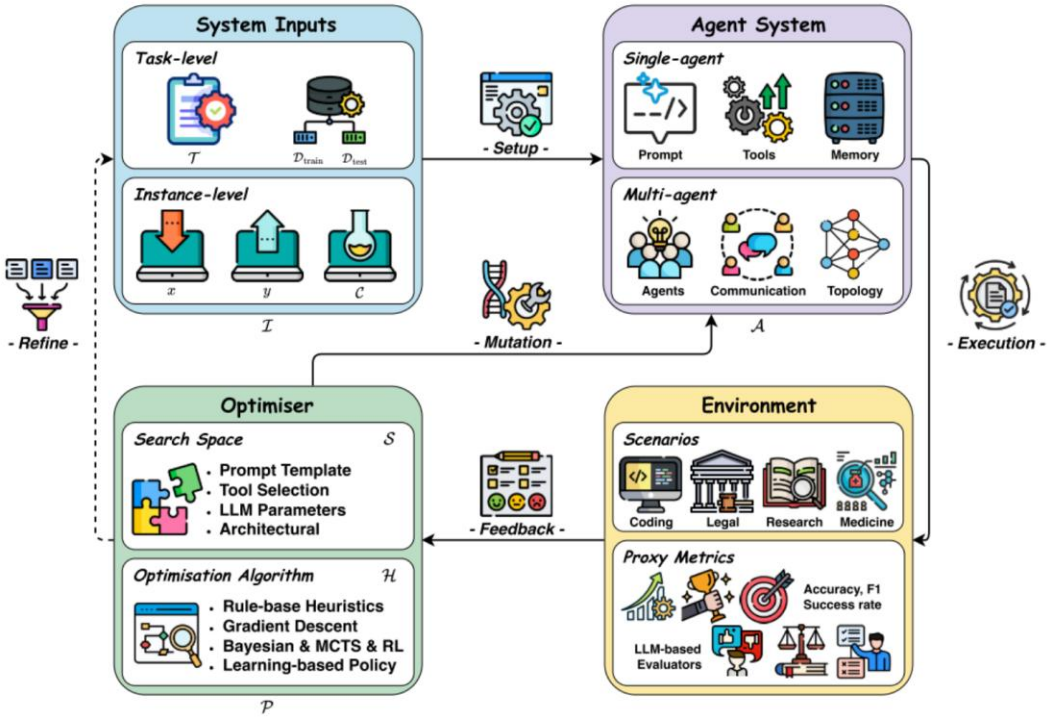


Figure 3 Conceptual framework of the self-evolving process in agent systems. The process forms an iterative optimisation loop comprising four components: *System Inputs*, *Agent System*, *Environment*, and *Optimiser*. System inputs define the task setting (e.g., task-level or instance-level). The agent system (in single- or multi-agent form) executes the specified task. The environment (depending on different scenarios) provides feedback via proxy metrics. The optimiser updates the agent system through a defined search space and optimisation algorithm until performance goals are met.

1. **System Inputs**:
    - 목표, 평가 메트릭, 도메인 제약(법률/리스크 등)
2. **Agent System**:
    - 지금까지 설명한 Profile·Memory·Planning·Action + MAS 토폴로지 전체
3. **Environment**:
    - 실제 사용자, 비즈니스 시스템, 시뮬레이션 환경에서 로그·보상·실패 사례를 수집
4. **Optimisers**:
    - 프롬프트, 툴 셋, 메모리 구조, 에이전트 구성, 워크플로 그래프를 자동/반자동으로 개선하는 모듈

| 4-2. 무엇을 진화시킬 것인가? (What to evolve)

진화의 대상들.

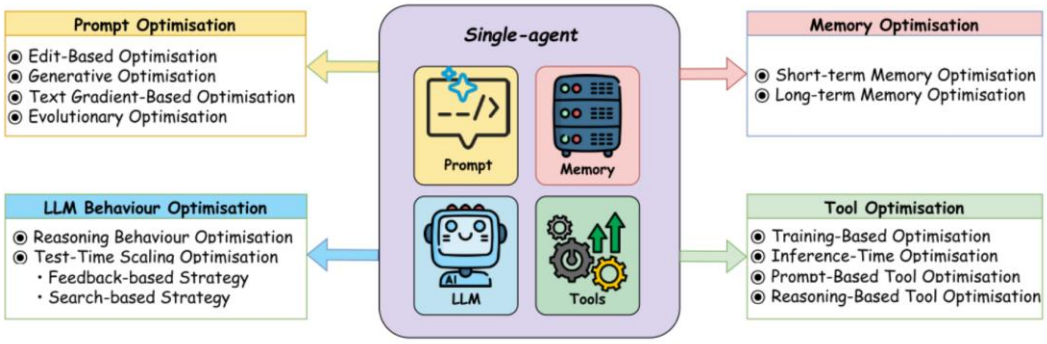


Figure 4 An overview of single-agent optimisation approaches, categorised by the targeted component within the agent system: prompt, memory, and tool.

- Foundation model (라벨·로그 기반 추가 파인튜닝)
- Agent prompts (역할/지침 최적화)
- Memory (저장 정책, 구조, 요약 방식)
- Tools (어떤 툴을 쓸지, 파라미터 템플릿)
- Workflows (step 순서, 분기 조건)
- Communication mechanisms (토론/투표/질문 전략)

| 4-3. 언제·어떻게 진화시킬 것인가? (When / How to evolve)

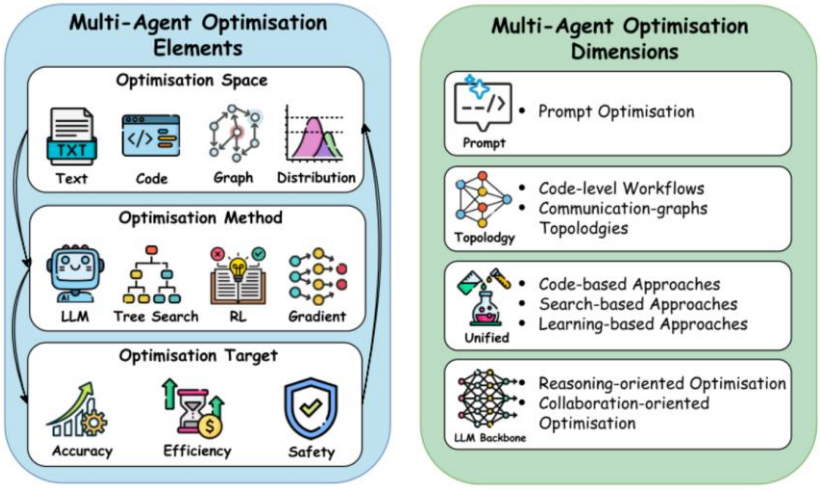


Figure 6 An overview of multi-agent systems optimisation approaches, with core optimisation elements (space, methods, and targets) on the left and optimisation dimensions (prompt, topology, unified, and LLM backbone) on the right.

When

- 주기적(배치): 하루/주 단위 로그로 retraining 또는 offline optimisation
- 이벤트 기반: 실패율 급증, 특정 에러 패턴, 사용자의 강한 피드백 발생 시
- 온라인 소규모 업데이트: 메모리/프롬프트 수준의 빠른 patch

How

- Search / RL: 프롬프트·워크플로를 검색 공간으로 보고 성능 기반 탐색
    - 프롬프트나 워크플로의 여러 버전을 만들어보고, 그중 가장 잘 작동하는 조합을 자동으로 찾아낸다
- Gradient-based editing & model editing: 특정 지식/규칙을 파라미터 레벨에서 수정
    - LLM 내부의 파라미터를 직접 조금만 고쳐서, 특정 지식이나 규칙을 '주입·수정
- Multi-agent self-play & debate: 에이전트끼리 토론/자기비판을 통해 더 나은 정책 도출