LLM 기반 AI 시스템 아키텍처

(Agent, RAG, MCP, A2A, Multimodal의 이해)

Agent, RAG, MCP, A2A, Multimodal 의 비교

비교표

개념	핵심 목적	주요 활용	장점	한계
Agent	AI의 **행동 주체화**	자동화, 워크플로우 실행	반복작업 제거, 생산성 ↑	신뢰성 부족
RAG	최신/외부 데이터 반영	QA, 검색 기반 챗봇	정확도 ↑, 비용 효율	검색 품질 의존
MCP	**표준화된 연동**	IDE ↔ 시스템 ↔ AI 연결	생태계 확장성	초기 표준, 확산 부족
A2A	AI 간 협업	프로젝트 관리, 멀티 에이전트	확장성 ↑	제어/비용 리스크
Multimodal	다양한 입력 처리	이미지 설명, 음성 인터랙션	UX 자연스러움	학습/운영 비용 ↑

AI Agent

Agent 개요

정의

명시된 목표를 달성하기 위해 여러 단계를 계획하고, 필요한 외부 도구를 선택·실행하며, 각 단계의 결과를 평가해 다음 행동을 자율적으로 결정하는 정책·검증·실행 책임을 가진 AI 오케트레이션 시스템이다.

핵심 특징

- **제한된 자율성**: 정책·권한·비용 범위 내에서 목표 달성
- **도구 사용**: API, 파일, 웹, DB 등 외부 세계와 상호 작용
- **반복 실행**: 목표 달성까지 “추론 → 행동 → 관찰” 반복
- **적응성**: 실패 시 대안 경로 탐색
- **상태 관리**: 과거 실행 이력과 중간 결과를 기억하며 진행

Agent 개요

구성요소

1. **LLM (추론 엔진)**

   상황 분석, 다음 행동 제안, 자연어 생성. “제안만” 하고 실행하지 않음

2. **Agent Controller (오케스트레이터)**

   정책 적용, 입출력 검증, 도구 실행, 상태 관리. 모든 실행 책임, 안전 장치

3. **Tools (실행 도구)**

   외부 세계와 상호작용 (API, 파일, DB, 웹). 판단 없이 실행만

4. **Memory (기억 저장소)**

   현재 작업 컨텍스트, 중간 결과. 과거 실행 이력, 학습된 패턴

5. **Planner (계획 모듈)**

   복잡한 목표를 하위 작업으로 분해. LLM 기반 또는 규칙 기반

Agent 개요

동작 방식 (ReAct 루프 – 실무 관점)

1. **추론** (Reasoning)

   LLM이 현재 상황 분석, 다음 단계 계획

2. **행동** (Action)

   Agent가 검증·승인한 도구 실행

3. **관찰** (Observation)

   실행 결과 확인 및 수집

4. **반성** (Reflection)

   목표 달성 여부 평가 및 전략 수정 (선택)

5. **반복** (Iteration)

   목표 달성까지 1~4단계 반복

Agent 개요

장점

- 복잡한 작업 자동화: 멀티 스텝 워크플로우 처리
- 사람 개입 최소화: “~해줘”만 하면 끝까지 진행
- 적응적 실행: 예상 못한 상황에도 대안 찾기
- 확장성: 도구 추가로 능력 확장

과제 및 한계

- 신뢰성: 환각, 무한 루프, 비결정성
- 안전성: 의도하지 않은 위험한 행동
- 책임 소재: 실수 시 누구 책임?
- 비용: 여러 LLM 호출로 비용 증가
- 해석 가능성: 왜 이 결정을 했는지 설명 어려움
- 제어 가능성: 예상 밖 경로로 진행 시 개입 어려움
- 성능: 여러 단계 거치며 지연 시간 증가

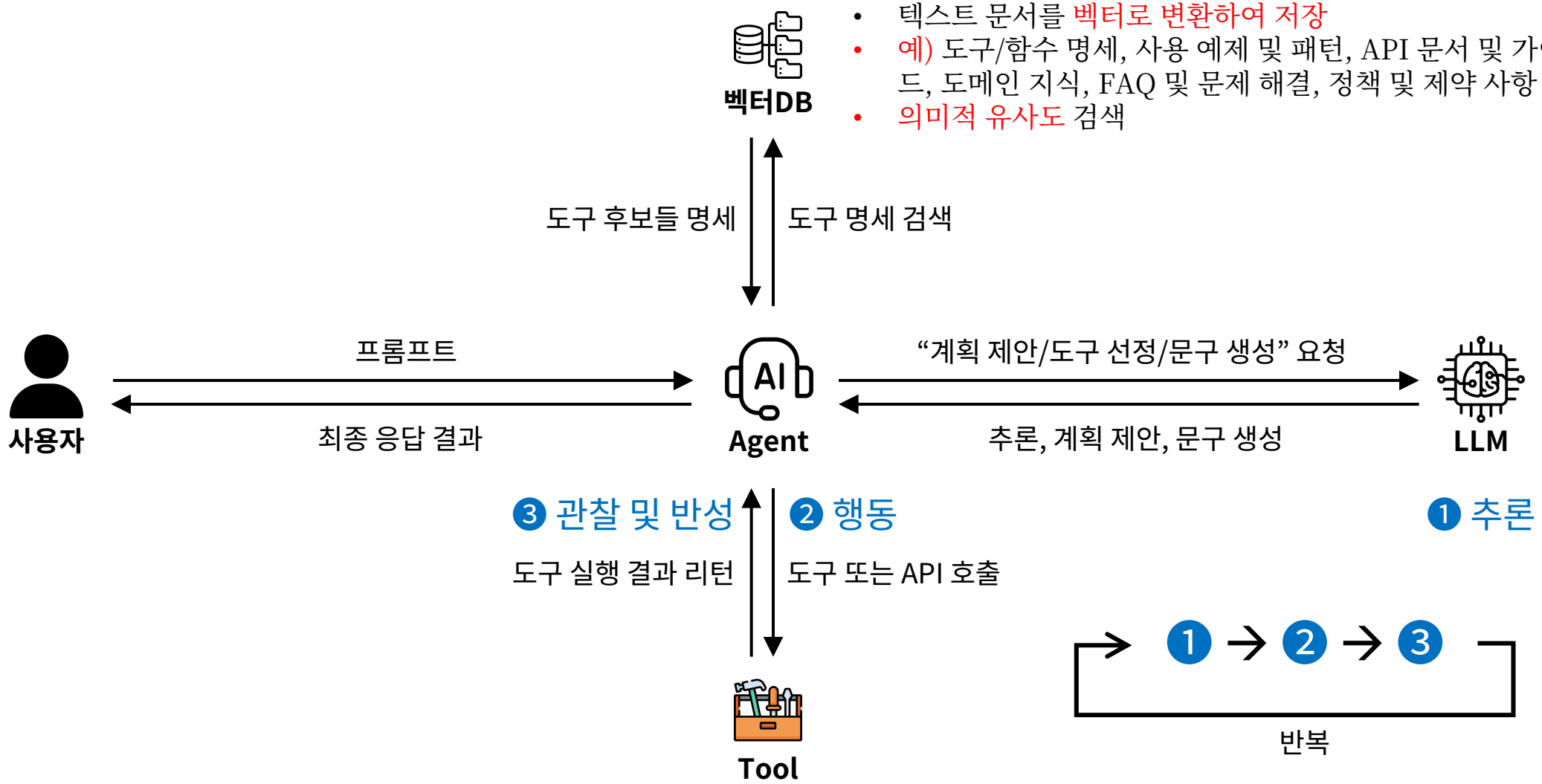
해결 방안

- Validation 및 재시도 제한
- 샌드박싱 및 권한 관리
- Human-in-the-Loop (중요 결정은 사람 승인)
- 감사 로그 및 비용 한도 설정
- 중간 단계 가시화 및 중단 기능

실제 사례

- **LangChain Agent**: Python 기반 Agent 개발 프레임워크
- **LlamaIndex Agent**: 데이터 중심 Agent
- **AutoGPT**: 자율 작업 수행 Agent
- **BabyAGI**: 작업 분해 및 실행
- **GitHub Copilot Workspace**: 코드 작성 자동화
- **Devin**: 자율 소프트웨어 엔지니어 (전체 프로젝트 수행)

Agent 워크플로우


벡터DB
• 텍스트 문서를 벡터로 변환하여 저장
• 예) 도구/함수 명세, 사용 예제 및 패턴, API 문서 및 가이드, 도메인 지식, FAQ 및 문제 해결, 정책 및 제약 사항 등
• 의미적 유사도 검색
도구 후보들 명세
도구 명세 검색
사용자
프롬프트
최종 응답 결과
AI
Agent
"계획 제안/도구 선정/문구 생성" 요청
추론, 계획 제안, 문구 생성
LLM
❸ 관찰 및 반성
도구 실행 결과 리턴
❷ 행동
도구 또는 API 호출
❶ 추론
Tool
1 → 2 → 3
반복

Agent 워크플로우 참여 컴포넌트

사용자

- **역할:** 목표 제공
- **특징:** 어떻게는 몰라도 되고, 무엇을만 말함.

Agent

- **역할:** 전체 시스템 제어 및 구성 요소 간 데이터 중계
- **핵심:** 어떤 단계로 진행할 지 결정, LLM의 제안을 검증하고 실제로 Tool 호출, 실패 처리/재시도/권한 체크 책임

벡터DB

- **역할:** 많은 도구/문서 중 관련 있는 것을 빠르게 검색해 컨텍스트 제공
- **핵심:** 정답을 만들지 않음(생성X), 찾아줌(검색)
- **예:** Chroma, Pinecone, Weaviate, Milvus, Qdrant 등

LLM

- **역할:** 추론/계획 제안/문구 생성
- **핵심:** 실행하지 않음, 호출안을 제안하고 텍스트를 생성

Tool

- **역할:** 실제 외부 세계와 상호작용(API, DB 조회, 캘린더 생성 등)
- **책임:** 판단하지 않음. 실행하고 결과를 반환

Agent 워크플로우 – 1단계 사용자 요청 수신 및 분석

벡터DB

사용자

1. 자연어로 목표 입력

예) “오늘 서울 날씨 보고 캘린더에 등록해줘”

Agent

- 인증·인가
- 입력 검증
- 컨텍스트 해석 및 생성
- Rate Limit·Cost Guard
  (속도, 빈도, 비용 관점에서 요청 처리를 사전에 차단하는 보호 장치)

LLM

Tool

Agent 워크플로우 – 2단계 관련 도구 명세 검색

벡터DB

2. 관련 도구 명세 검색
질의 벡터를 주며 도구 검색을 요청한다.

사용자

Agent

- 검색 질의 생성
  → 민감 정보 제거
  → 키워드 정규화(“서울”, “날씨” 등)
  → 질의 임베딩 생성(벡터화)
- 접근 제어 조건 적용

LLM

Tool

Agent 워크플로우 – 2단계 관련 도구 명세 검색(계속)

벡터DB

- 미리 저장된 수백 개의 도구 명세서 벡터 중에서 사용자 질의 벡터와 가장 유사한 벡터 검색
  → 코사인 유사도 계산
- Top-K 결과 반환

Top-K 후보 도구들 명세 리턴

- 입력/출력 스키마
- 사용 예시
- 제약 조건
- 보안 요구사항

사용자

Agent

LLM

Tool

Agent 워크플로우 – 3단계 실행 계획 수립

벡터DB

사용자

Agent

3. 계획 수립 및 호출안 제안 요청

- LLM 입력 컨텍스트 구성
  → 사용자 요청(마스킹 처리)
  → 해석된 컨텍스트(날짜/위치)
  → 권한 검증 완료된 도구 명세
  → 시스템 정책 주입
- LLM 호출 제약 조건

LLM

Tool

Agent 워크플로우 – 3단계 실행 계획 수립(계속)

벡터DB

사용자

Agent

LLM

계획 및 도구, 아이디어 제안

- 요청을 단계로 분해
  → 날씨 조회
  → 결과 요약
  → 캘린더 등록
- 각 단계 별 적절한 도구 제안
  → 호출 파라미터 초안 제안
- 실패 시나리오에 대한 아이디어 제안

Tool

Agent 워크플로우 – 3단계 실행 계획 수립(계속)

벡터DB

사용자

Agent

LLM 출력 검증

- 파싱 가능 여부
- 존재하지 않는 도구 제안 차단
- 스키마/타입 검증
- 정책 위반 여부 확인

LLM

Tool

Agent 워크플로우 – 4단계 도구 실행 및 결과 수집

벡터DB

사용자

Agent

도구 실행 전 다층 검증(날씨 조회)

- 스키마 검증(필수 파라미터, 타입)
- 권한/정책/보안 검증

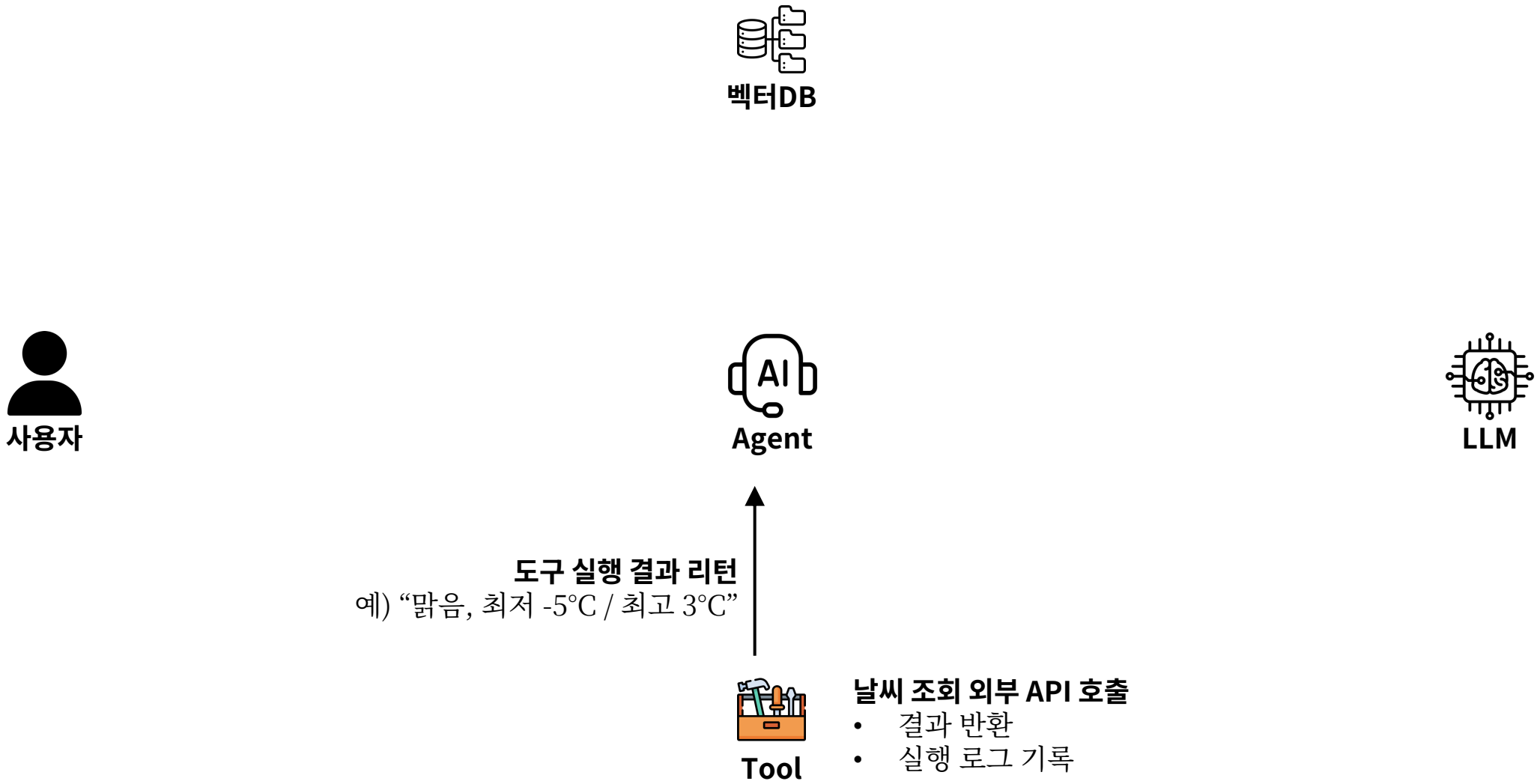
LLM

4. 도구 실행(날씨 조회)

- TLS/SSL 검증
- Secrete Manager에서 API Key 로드
- 타임아웃 설정

Tool

Agent 워크플로우 – 4단계 도구 실행 및 결과 수집(계속)



Agent 워크플로우 – 4단계 도구 실행 및 결과 수집(계속)

벡터DB

사용자

도구 실행 결과 후 처리

- 응답 스키마 검증
- XSS 패턴 제거
- 민감 정보 필터링
- 캐싱(예: TTL)
- 감사 로그 기록

Agent

LLM

Tool

Agent 워크플로우 – 5단계 최종 응답 생성 및 반환

벡터DB

사용자

Agent

5. 결과 해석 및 다음 액션 요청

LLM

- 정제된 날씨 결과 전달
- 요청:
  → 캘린더 제목/메모 생성
  → 캘린더 도구 파라미터 제안
- 제약 조건 재주입
  → 길이 제한
  → 개인정보 금지
  → 시간 정책

Tool

Agent 워크플로우 – 5단계 최종 응답 생성 및 반환 (계속)

벡터DB

사용자

Agent

LLM

날씨 해석 문구 및 캘린더 문구 생성

- 날씨 데이터를 자연어로 요약
- 캘린더 문구 생성
- 캘린더 도구 호출 파라미터 제안

Tool

Agent 워크플로우 – 5단계 최종 응답 생성 및 반환 (계속)

벡터DB

사용자

Agent

LLM 출력 검증

- 생성 문구 정책/보안 검증
- 캘린더 API 스키마 검증
- 시간 값 유효성 검증

LLM

Tool

Agent 워크플로우 – 5단계 최종 응답 생성 및 반환 (계속)

벡터DB

사용자

Agent

도구 실행 전 다층 검증(캘린더 조회)

- 스키마 검증(필수 파라미터, 타입)
- 권한/정책/보안 검증

LLM

6. 도구 실행(캘린더 등록)

- TLS/SSL 검증
- Secrete Manager에서 API Key 로드
- 타임아웃 설정

Tool

Agent 워크플로우 – 5단계 최종 응답 생성 및 반환 (계속)

벡터DB

사용자

도구 실행 결과 후 처리
- 이벤트 ID 로깅
- 할당량 업데이트
- 롤백 정보 저장

Agent

LLM

도구 실행 결과 리턴
예) 등록 유무(true/false)

Tool

캘린더 등록 외부 API 호출
- 결과 반환
- 실행 로그 기록

Agent 워크플로우 – 5단계 최종 응답 생성 및 반환(계속)

벡터DB

사용자

7. 최종 응답 결과

Agent

최종 응답 준비

- 성공 메시지 구성
- 민감 정보 마스킹 재확인
- 응답 길이 제한

LLM

"오늘 서울은 맑음(최저 -5°C / 최고 3°C)입니다.
'따뜻하게 입기' 일정이 캘린더에 등록되었습니다."

Tool

RAG
(Retrieval-Augmented Generation)

RAG 개요

정의

LLM이 답변을 생성하기 전에 외부 지식 저장소(문서, DB, 벡터스토어 등)에서 관련 정보를 **검색(Retrieve)**하여 **컨텍스트로 주입(Augment)**하고, 그 근거 위에서 답변을 **생성(Generate)**하는 AI 아키텍처 패턴이다. 즉, LLM의 기억을 확장하는 구조이며 LLM 자체를 재학습시키지 않고도 최신 정보·사내 지식·도메인 지식을 활용하게 한다.

핵심 특징

- **지식 보강**: LLM 내부 파라미터가 아닌 외부 데이터 기반 응답
- **최신성 확보**: 모델 재학습 없이 즉시 데이터 반영
- **환각 감소**: 추측 대신 실제 문서에 근거한 답변
- **도메인 특화**: 사내 문서, 정책, 법률, 의료, 기술 문서에 강함
- **Agent와 결합 가능**: 언제 검색할지 판단은 Agent가 담당
- **비용 효율**: Fine-tuning 대비 훨씬 저렴하고 빠름

RAG 개요

구성요소

1. **Retriever (검색기)**

   질의 임베딩 생성, 유사도 검색 수행. 벡터 검색, 키워드 검색, 하이브리드 검색. 예) FAISS, ElasticSearch, Pinecone

2. **Vector Store (벡터 저장소)**

   문서/데이터를 임베딩 형태로 저장. 문서 chunk + 메타데이터 + 벡터. 예) Chroma, Weaviate, Qdrant

3. **Embedding Model (임베딩 모델)**

   텍스트를 벡터로 변환. 질의와 문서를 같은 모델로 임베딩 필수. 예) Open text-embedding-ada-002, Sentence-BERT

4. **LLM (생성 모델)**

   검색된 컨텍스트를 해석하여 답변 생성. 시스템 프롬프트 + 검색 문서 + 사용자 질의. 예) GPT-5.2, Claude, Gemini

5. **(선택) Agent / Orchestrator**

   검색 필요성 판단, 검색 쿼리 생성 전략 결정, 재검색, 결과 검증. 다중 검색, 결과 병합, 신뢰도 평가

RAG 개요

동작 방식 (RAG 파이프라인)

1. **질의 전처리**

   사용자 질문을 검색에 적합한 형태로 변환. 예) “서울 날씨 어때?” → “서울 기상 정보”

2. **검색**

   질의를 벡터로 임베딩, 벡터DB/검색시스템에서 유사 문서 조각 검색 (top-k)

3. **컨텍스트 증강**

   검색 결과를 정제·압축·재정렬하여 LLM 입력 컨텍스트로 삽입

4. **생성**

   LLM이 검색된 문서를 근거로 답변 생성. 출처 인용 포함(선택)

5. **(선택) 검증 및 후처리**

   답변과 문서간 정합성 확인. 환각 탐지 및 필터링

RAG 개요

장점

- **환각 감소 →** “아는 척” 대신 실제 문서 기반 답변
- **최신성 →** 학습 이후 데이터도 즉시 활용 가능
- **비용 효율 →** Fine-tuning 대비 훨씬 저렴
- **설명 가능성 →** “이 답변은 이 문서를 근거로 했다” 제시 가능

과제 및 한계

- **검색 품질 의존성**: 잘못된 문서 → 잘못된 답변, 오래된 문서 → 시대에 뒤진 답변, 관련 없는 문서 → 혼란스러운 답변
- **컨텍스트 길이 제한**: 너무 많으면 토큰 초과/비용 증가/처리 느림, 너무 적으면 근거 부족/불완전한 답변
- **문서 관리 부담**: 중복, 버전, 권한, 폐기 정책 필요
- **완전한 환각 제거 불가**: 문서 간 연결 추론에서 여전히 발생 가능
- **지연 시간**: 검색 → 증강 → 생성 단계로 응답 느림
- **임베딩 모델의 한계**: 동음이의어, 은유적 표현 처리 어려움

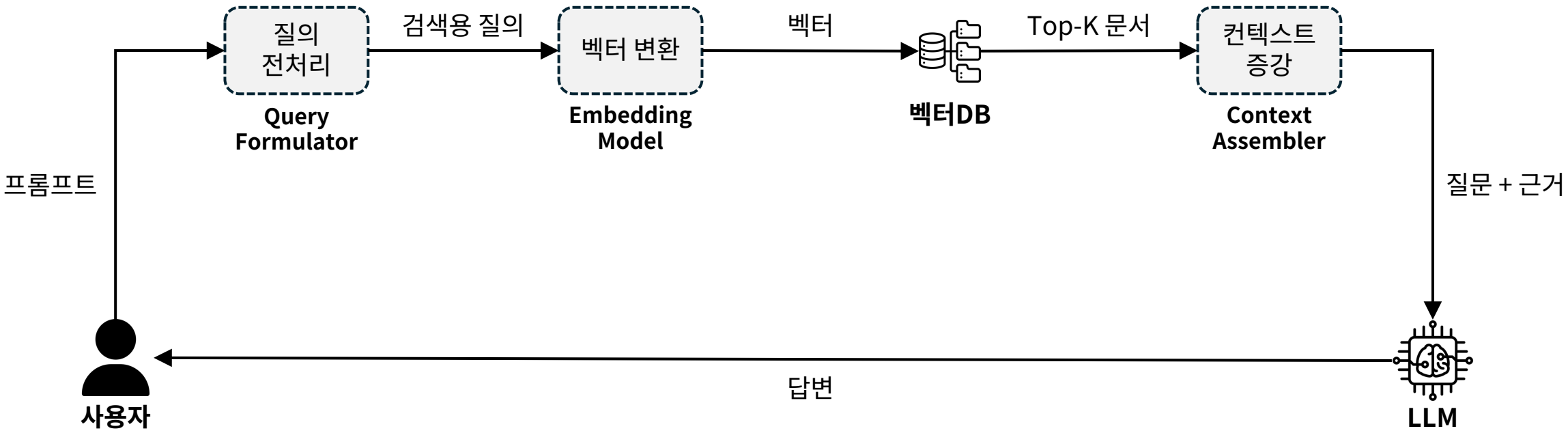
해결 방안

- **검색 품질 향상**: 하이브리드 검색(키워드 + 벡터), Re-ranking 모델 도입, 다중 검색, 질의 확장(동의어, 관련어 추가)
- **컨텍스트 최적화**: Chunking 전략 개선, Overlap 전략, 컨텍스트 압축, 메타데이터 필터링
- **문서 관리**: 버전관리, 자동 업데이트, 중복 제거
- **환각 방지**: 시스템 프롬프트(“문서에 없으면 ‘모른다’고 답변”), 출처 인용 강제, 답변 신뢰도 표시
- **성능 개선**: 캐싱, 인덱싱 최적화
- **Agent 활용**: Agent 기반 재검색·검증 루프 적용

실제 사례

- 사내 문서 Q&A: 규정, 위키, 기술 문서
- 법률: 판례 검색, 법령 조회, 계약서 분석
- 의료: 진료 가이드라인, 약물 정보, 임상 프로토콜
- 금융: 규정 준수, 상품 설명서, 투자 리포트
- 고객 지원 챗봇: FAQ, 제품 매뉴얼, 트러블슈팅 문서 기반 답변

RAG 워크플로우


질의
전처리
Query
Formulator
검색용 질의
벡터 변환
Embedding
Model
벡터
벡터DB
Top-K 문서
컨텍스트
증강
Context
Assembler
질문 + 근거
LLM
답변
사용자
프롬프트

RAG 워크플로우 참여 컴포넌트

사용자	• **역할:** 무엇을 알고 싶은지 질문한다. • **특징:** 어떻게 검색할지는 알 필요 없음. 단순히 질문만 제시. • **예:** “사내 휴가 규정의 뭐야?”
Query Formulator	• **역할:** 사용자 질문을 검색에 적합한 형태로 변환. 검색 전략을 결정. • **특징:** 알 필요 없음.질문을 의미 중심으로 재작성. 동의어, 핵심 키워드 강조. • **예:** “사내 휴가 규정의 뭐야?” → “사내 휴가 정책, 연차 규정”
Embedding Model	• **역할:** 텍스트를 의미 벡터로 변환. • **특징:** 사람이 읽는 문장을 숫자 벡터로 변환. 의미적 유사도 계산 가능. • **예:** “휴가 규정” → [0.021, -0.33, 0.78, …]
Vector DB	• **역할:** 대량의 문서 중 의미적으로 관련 있는 문서 조각을 검색 • **특징:** 사문서가 미리 벡터로 저장되어 있음. Top-K 결과 반환. 정답 생성(X), 후보 압축(O) • **예:** “입력” → 질의 벡터, “출력” → 관련 문서 Top-3
Context Assembler	• **역할:** 검색된 문서를 LLM이 이해할 수 있는 입력 형태로 구성 • **특징:** 불필요한 부분 제거. 길이 압축. 중요 문서 우선 배치. 질문 + 문서 묶기. • **예:** “질문” + [문서 A 요약] + [문서 B 요약]
LLM	• **역할:** 검색된 문서를 근거로 답변 생성 • **특징:** 새로운 지식을 만들어내지 않음. 문서에 기반해 설명. 문서에 없으면 “모른다” 가능 • **예:** “문서에 따르면 연차는 연 15일입니다.”

MCP
(Model Context Protocal)

MCP 개요

정의

LLM 또는 Agent가 외부 시스템(API, DB, 파일, 서비스 등)과 상호작용할 때 사용할 수 있는 기능과 컨텍스트를 표준화된 방식으로 노출하고 연결하는 AI 통합 프로토콜이다. 즉, MCP는 LLM/Agent와 외부 세계를 연결하는 표준 인터페이스 계층이며, 제각각이던 Tool 연동 방식을 일관된 규약으로 통합한다.

핵심 특징

- **표준화된 연결 방식 →** Tool/API/데이터 소스를 공통 규약으로 노출
- **컨텍스트 중심 설계 →** 단순 API 호출이 아니라 LLM이 이해할 수 있는 컨텍스트 제공
- **모델 독립성 →** 특정 LLM에 종속되지 않음
- **Agent 친화적 구조 →** 어떤 도구를 쓸 수 있는지, 어떻게 호출할 수 있는지 명확히 기술
- **보안·권한 내장 →** 접근 제어, 스코프, 인증을 프로토콜 수준에서 관리
- **확장성 →** 새로운 Tool 추가 시 Agent/LLM 수정 최소화

MCP 개요

구성요소

1. **MCP 클라이언트**

   MCP 서버와 통신하는 Agent/LLM 측 구성요소. MCP 서버 연결 및 Tool 탐색. Tool 호출 요청 전송. 결과 수신 및 LLM에 전달.

2. **MCP 서버**

   외부 시스템을 MCP 규약으로 노출. Tool/Resource 명세 제공. 실제 API/DB/파일 시스템 연동. 권한 검증 및 실행.

3. **Tool 명세**

   MCP 핵심 개념. 예) Tool 이름, 기능 설명, 입력 파라미터, 출력 구조, 권한/제약 조건.

4. **(선택) Resource**

   MCP가 제공하는 또 다른 개념. 읽기 가능한 데이터/컨텍스트 제공. 예) 파일 내용, DB 스키마, 현재상태.

4. **(선택) Prompts**

   MCP 서버가 제공하는 프롬프트 템플릿. 특정 작업을 위한 미리 정의된 프롬프트. 예) “코드 리뷰 프롬프트”

5. **LLM**

   MCP 명세를 이해, 호출 가능한 기능을 기반으로 호출안 제안, 실행 결과를 해석해 응답 생성. 어떤 LLM이든 MCP 프로토콜만 이해하면 사용 가능

6. **(선택) Agent / Orchestrator**

   MCP 사용 여부 판단, 어떤 Tool을 호출할지 결정. 호출 결과 검증 및 재시도 수행. MCP 없이도 동작 가능하지만, 함께 사용 시 강력함

MCP 개요

동작 방식 (MCP 파이프라인)

1. MCP 서버 구축

외부 시스템(API, DB, 파일 등)을 MCP 규약으로 감싸기. 제공 기능, 입력/출력 스키마, 권한 정책을 명세로 선언

예) Weather API를 “getWeather” Tool로 MCP 서버에 등록

2. 연결 및 탐색 (Discovery)

MCP 클라이언트(Agent/LLM)가 MCP 서버에 연결. 사용 가능한 Tool/Resource 목록 조회. 각 Tool의 명세(기능 설명, 파라미터, 제약사항) 수신

3. 컨텍스트 제공

MCP 서버가 현재 상태/데이터를 컨텍스트로 제공. LLM이 컨텍스트를 바탕으로 더 정확한 판단 가능.

예) “현재 접속 사용자”, “최근 수정된 파일 목록” 등

4. 호출 제안 및 검증

LLM이 MCP 명세를 바탕으로 “어떤 기능을 쓰면 좋을지” Tool 호출안 제안. Agent가 제안을 검증(권한, 파라미터, 정책)

예) “getWeather(location=‘Seoul’, date=‘2026-01-17’)

5. 실행

MCP 클라이언트가 MCP 서버에 요청. MCP 서버가 실제 시스템(API, DB)과 연동하여 실행. 결과를 표준화된 형식으로 반환

6. 결과 해석

LLM이 반환된 결과를 해석. 사용자에게 전달할 응답 생성.

MCP 개요

장점

- **통합 비용 감소** → Tool마다 다른 인터페이스를 만들 필요가 없음.
- **유지보수 용이** → Tool 변경 시 MCP 서버만 수정
- **안정성 향상** → 허용된 기능만 노출, 임의 호출 방지
- **Agent 확장성** → 새로운 Tool 추가가 곧 새로운 능력 추가
- **멀티모달/멀티모델 대응** → 다양한 LLM과 공통 방식으로 연동 가능

과제 및 한계

- **초기 설계 비용**: MCP 서버 및 명세 정의 필요
- **명세 품질 의존성**: 설명/스키마가 부실하면 LLM이 잘못 사용
- **실행 지연**: MCP 계층 추가로 호출 경로 증가
- **완전 자동화의 한계**: 잘못된 호출 제안은 여전히 발생 가능
- **Agent 없이는 활용 제한**: MCP는 판단 주체가 아님
- **디버깅 어려움**: 3단계 구조로 문제 지점 파악 어려움
- **버전 관리**: Tool 명세 변경 시 하위 호환성 고려 필요

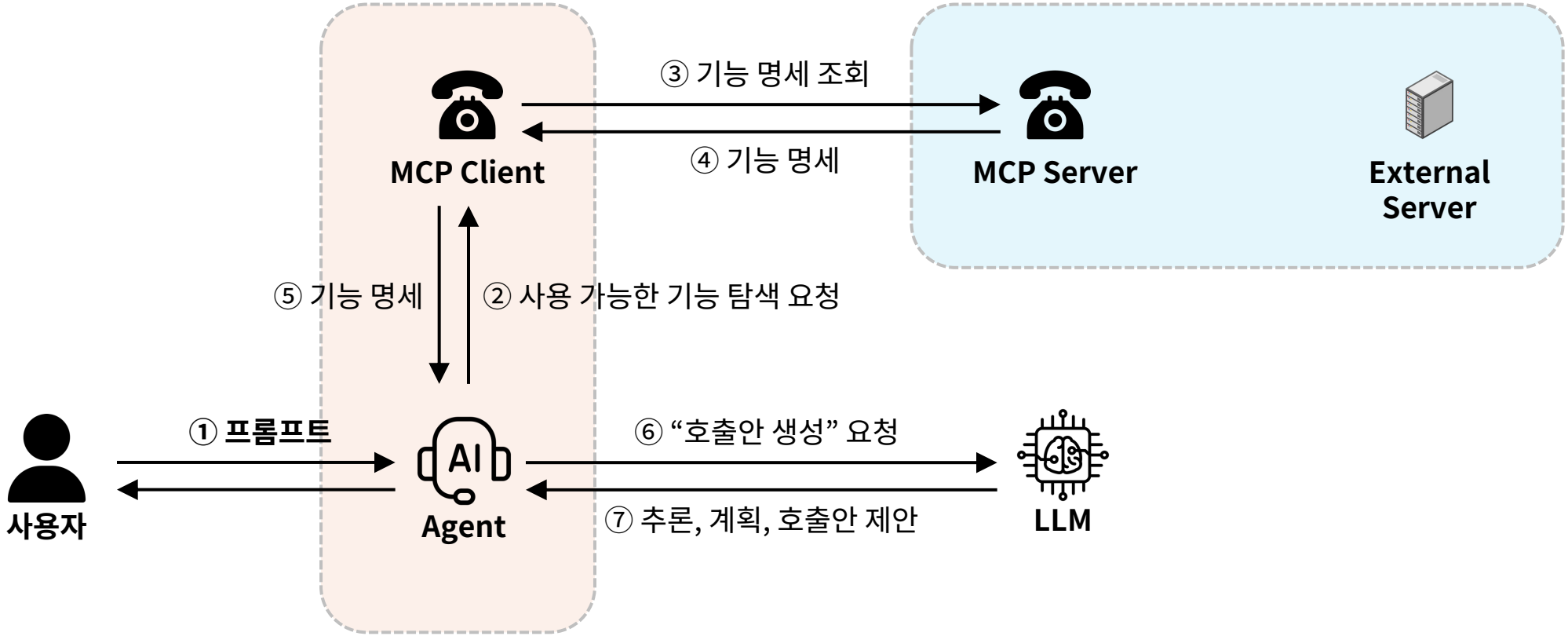
해결 방안

- 명확한 Tool 명세 작성(의도, 예제 포함)
- 권한스코프 최소화 원칙 적용
- Agent 기반 호출 검증 및 재시도 로직
- 캐싱 및 배치 호출로 성능 최적화
- 호출 실패 시 안전한 fallback 제공
- RAG=무엇을 참고할지 제공, MCP=무엇을 할 수 있는지 제공

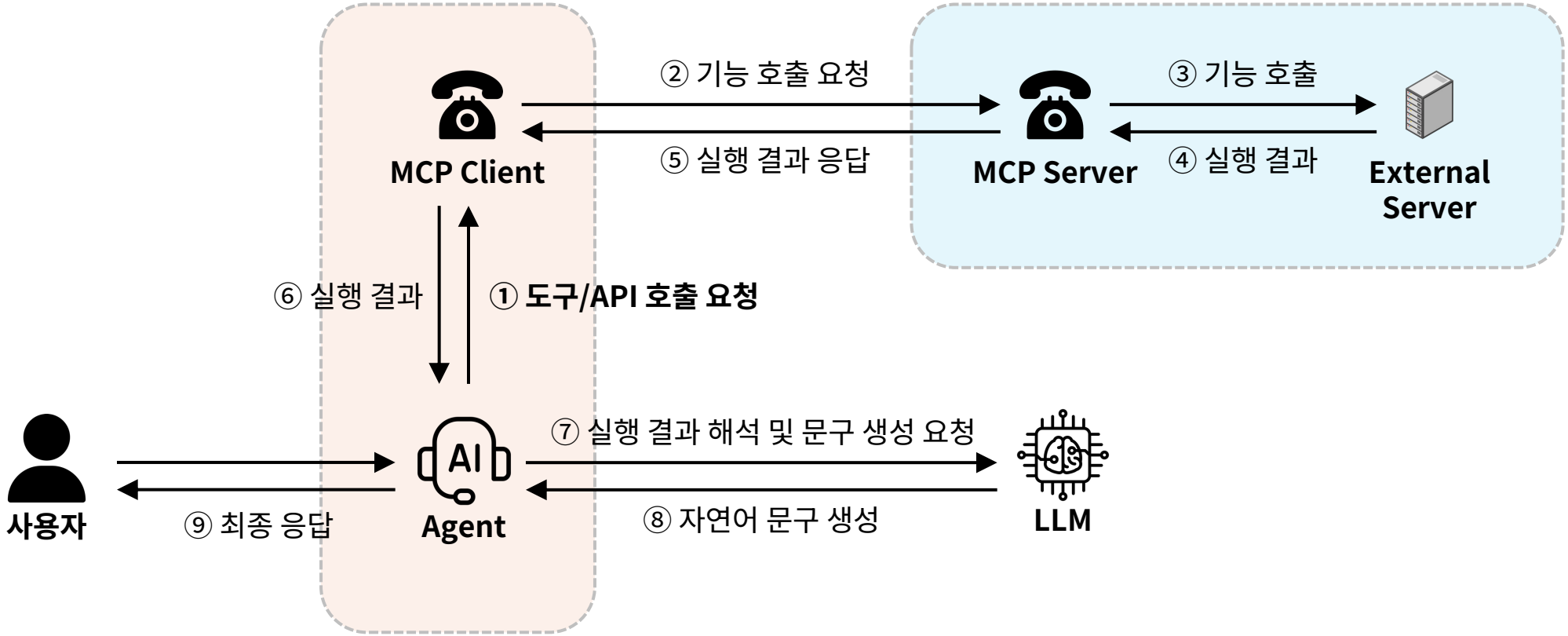
실제 사례

- 사내 시스템 연동 → 인사/급여/휴가 API를 MCP 서버로 노출
- 개발 도구 연동 → GitHub, Jira, CI/CD 시스템 제어
- 업무 자동화 Agent → 메일 발송, 캘린더 등록, DB 업데이트
- 멀티모달 Agent → 파일 시스템, 이미지/음성 처리 도구와 연동
- 기업용 AI 플랫폼 → 내부 Tool을 MCP로 표준화하여 여러 Agent에서 재사용

MCP 워크플로우 – 기능 탐색



MCP 워크플로우 – 기능 실행



A2A
(Agent-to-Agent)

A2A 개요

정의

여러 AI Agent가 서로 메시지를 주고받으며 역할을 분담하고 협업할 수 있도록 하는 아키텍처 패턴/통신 방식이다. 즉, A2A는 단일 Agent의 한계를 넘어, **Agent들을 하나의 조직처럼 구성**하여 복잡한 목표를 달성하게 하는 구조이며, Agent 간의 **책임 분리·검증·병렬 실행을 가능하게** 한다.

핵심 특징

- **역할 분담 →** Planner, Executor, Reviewer, Specialist 등 Agent를 기능별로 분리
- **협업 중심 구조 →** 하나의 목표를 여러 Agent가 나누어 수행
- **Agent 간 통신 →** 메시지, 상태, 결과를 주고받는 구조
- **확장성 →** Agent 추가 = 조직 확장
- **검증·상호 견제 →** 한 Agent의 결과를 다른 Agent가 검토
- **병렬 처리 가능 →** 독립 작업을 동시에 수행하여 성능 향상

A2A 개요

구성요소

1. **Agent**

   A2A의 기본 단위. 각 Agent는 독립적인 판단 능력과 역할을 가짐. 예) Planner Agent, Review Agent 등

2. **Agent 역할 정의(Role Specification)**

   각 Agent가 무엇을 책임지는지 명확히 정의한 계약. 담당 업무, 입력/출력 형식, 제약 조건, 다른 Agent와의 인터페이스 포함.

   예) Planner(작업 분해 및 순서 결정), Executor(Tool/MCP 실행), Reviewer(결과 검증 및 오류 탐지)

3. **Agent 간 메시지 채널**

   Agent들의 의사소통하는 수단. 예) Task 요청, 중간 결과, 검증 피드백, 승인/거절 메시지

4. **(선택) Orchestrator/Coordinator**

   전체 Agent 흐름을 조율하는 상위 Agent 또는 시스템. 어떤 Agent에게 일을 맡길지 결정, 병렬/순차 실행 제어.

5. **(선택) Shared Context/Memory**

   Agent 간 공유되는 상태/지식. 작업상태, 중간결과, 공통규칙.

6. **LLM**

   각 Agent의 추론 엔진. Agent마다 동일하거나 서로 다른 LLM 사용 가능. 역할에 따라 프롬프트/제약 다르게 설정 가능.

A2A 개요

동작 방식 (A2A 파이프라인)

1. **역할 분해**

   하나의 사용자 요청을 여러 역할로 분해. 예) “보고서 작성” → 조사 / 요약 / 검증 / 편집

2. **작업 위임**

   Planner Agent가 각 하위 작업을 적절한 Agent에게 할당

3. **병렬 또는 순차 실행**

   각 Agent가 자신의 역할에 맞는 작업 수행. 독립 작업은 병렬 실행, 의존 작업은 순차 실행.

4. **결과 교환**

   Agent 간 중간 결과를 공유. 예) Executor → Reviewer: “이 결과를 검토해줘”

5. **검증 및 합의**

   Reviewer/Validator Agent가 결과 검증. 문제 발견 시 재작업 요청.

6. **최종 통합**

   Coordinator 또는 최종 Agent가 결과를 종합하여 사용자 응답 생성

A2A 개요

장점

- **복잡한 문제 처리 가능** → 단일 Agent로 감당하기 어려운 작업 분해
- **신뢰성 향상** → 상호 검증으로 오류 감소
- **확장성** → Agent 추가로 기능 확장
- **유연한 구조** → 역할 교체/추가 용이
- **조직 모델과 유사** → 실제 업무 프로세스에 자연스럽게 매핑

과제 및 한계

- **설계 복잡도 증가**: Agent 수 증가에 따른 관리 부담
- **통신 비용**: Agent 간 메시지 왕복으로 지연 발생
- **조정 난이도**: 역할 충돌, 책임 불명확 가능성
- **비용 증가**: 다수 LLM 호출
- **디버깅 어려움**: 어느 Agent에서 문제가 생겼는지 추적 필요
- **일관성 문제**: Agent 간 공유 상태 동기화 어려움
- **신뢰성 체인**: 한 Agent 실패 시 전체 영향
- **무한 루프 위험**: Agent 간 순환 의존

해결 방안

- 명확한 역할 정의 및 책임 경계 설정
- Orchestrator 도입으로 흐름 중앙 통제
- Agent 수 최소화 원칙 적용
- 중요 단계에만 Reviewer Agent 배치
- Agent 간 메시지 구조 표준화
- 캐싱·요약으로 통신 비용 절감

실제 사례

- 대규모 리포트 작성 → 조사 + 요약 + 검증 Agent
- 코드 생성 시스템 → 설계 + 구현 + 리뷰 Agent
- 기업 업무 자동화 → 일정관리 + 메일 + 승인 Agent
- AI 연구보조 → 문헌검색 + 분석 + 정리 Agent
- 고신뢰 AI 시스템 → 실행 + 감사/감시 Agent

A2A 개요

오픈소스 프레임워크

1. **AutoGen (Microsoft)**

   Agent 간 대화 자동화. 코드 생성 + 실행 + 디버깅 협업

2. **CrewAI**

   역할 기반 Agent 오케스트레이션. 순차/병렬 실행 지원

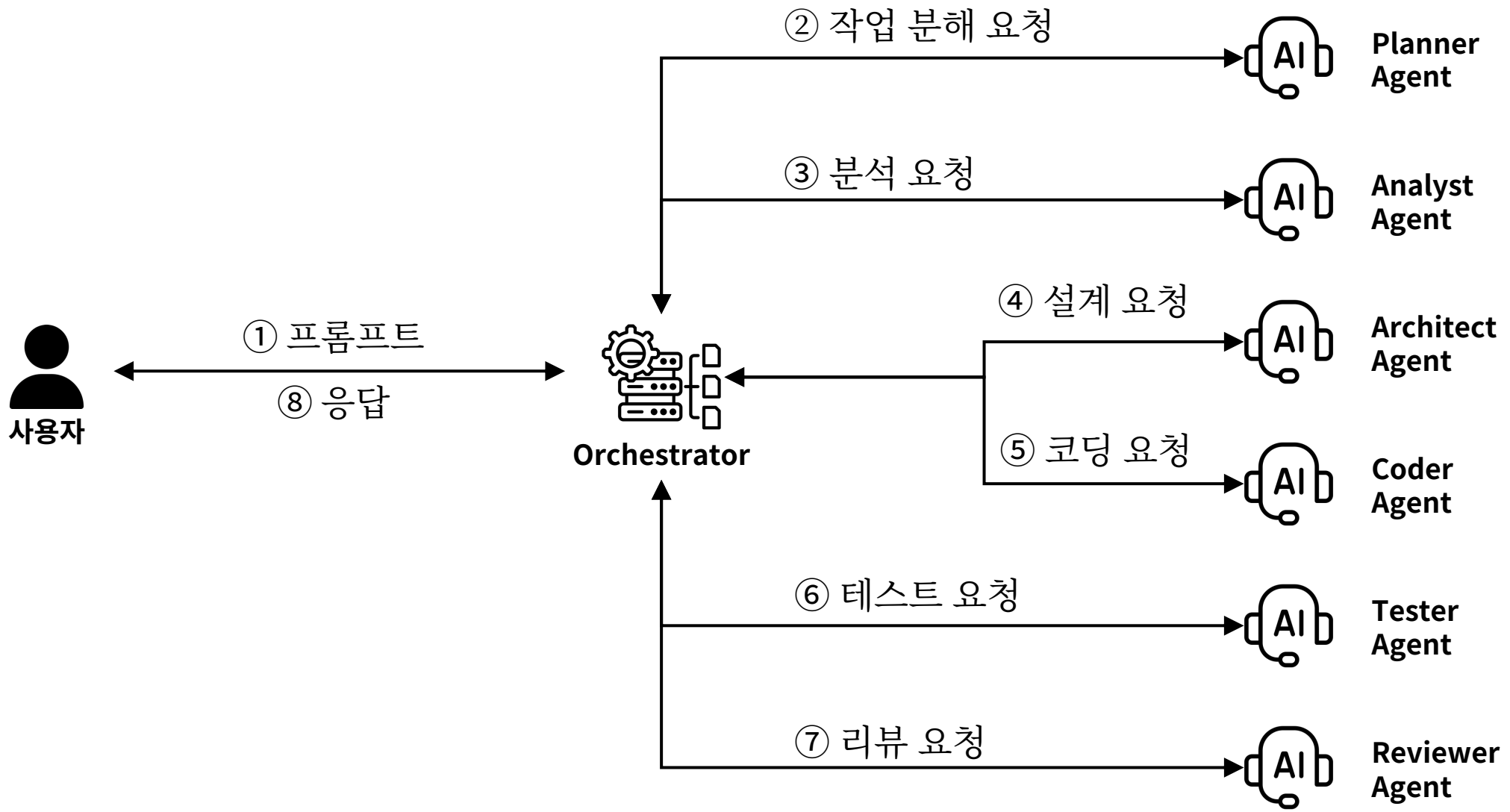
3. **LangGraph**

   그래프 기반 Agent 워크플로우. 상태 관리, 분기, 루프 지원

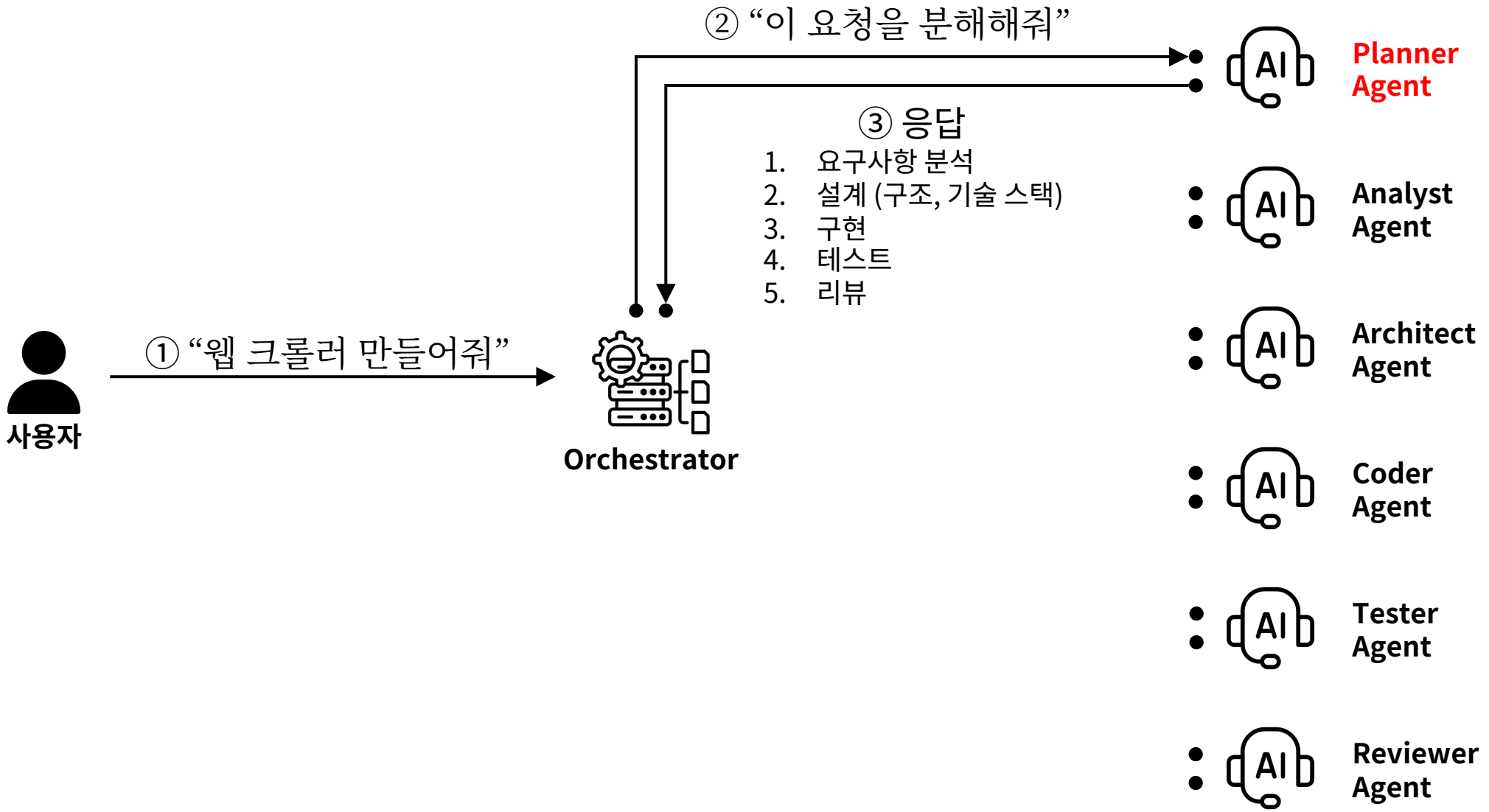
4. **MetaGPT**

   소프트웨어 개발 시뮬레이션. PM → Architect → Engineer → QA 흐름

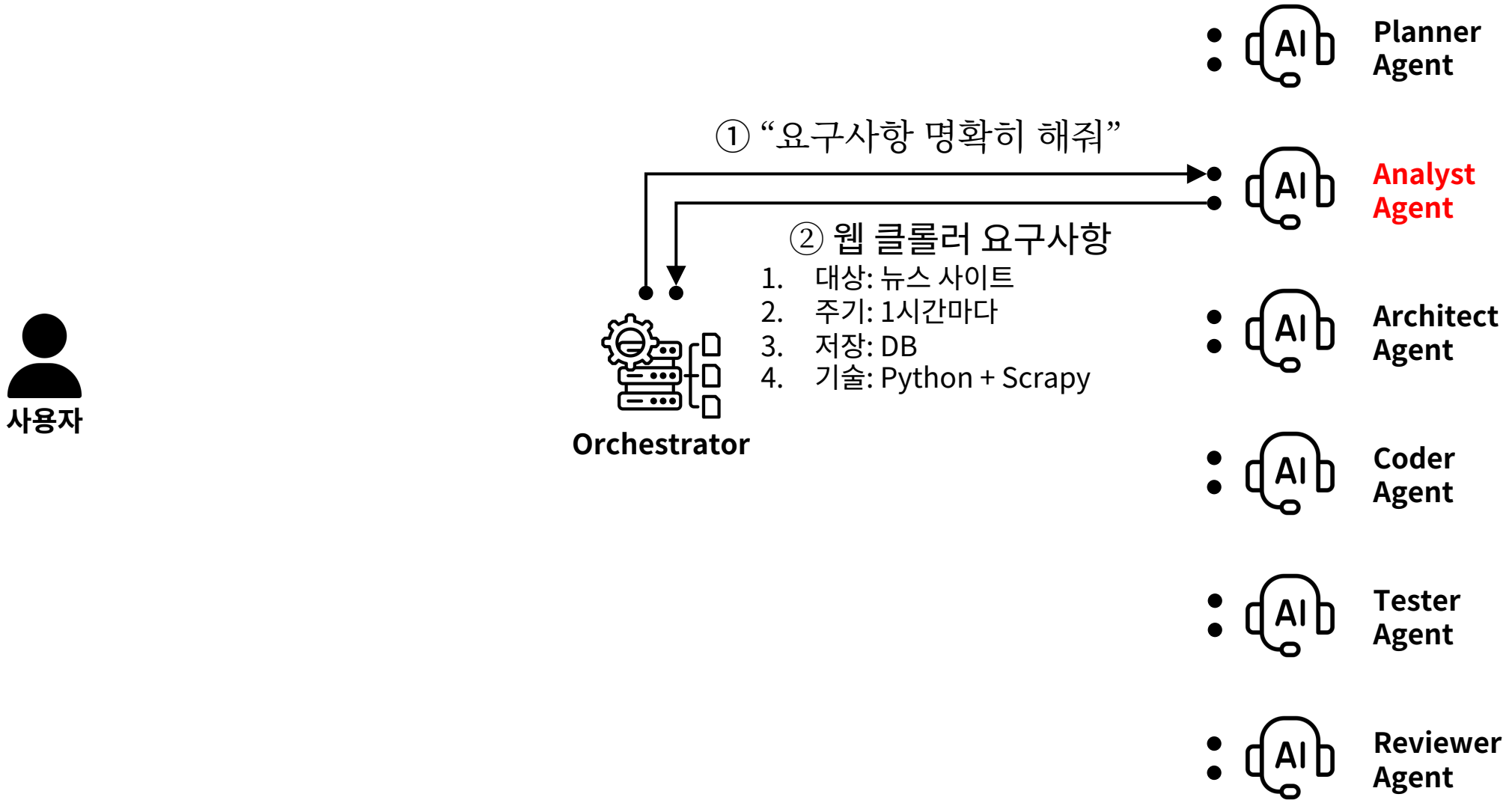
A2A 워크플로우


② 작업 분해 요청
Planner Agent
③ 분석 요청
Analyst Agent
④ 설계 요청
Architect Agent
① 프롬프트
⑧ 응답
사용자
Orchestrator
⑤ 코딩 요청
Coder Agent
⑥ 테스트 요청
Tester Agent
⑦ 리뷰 요청
Reviewer Agent

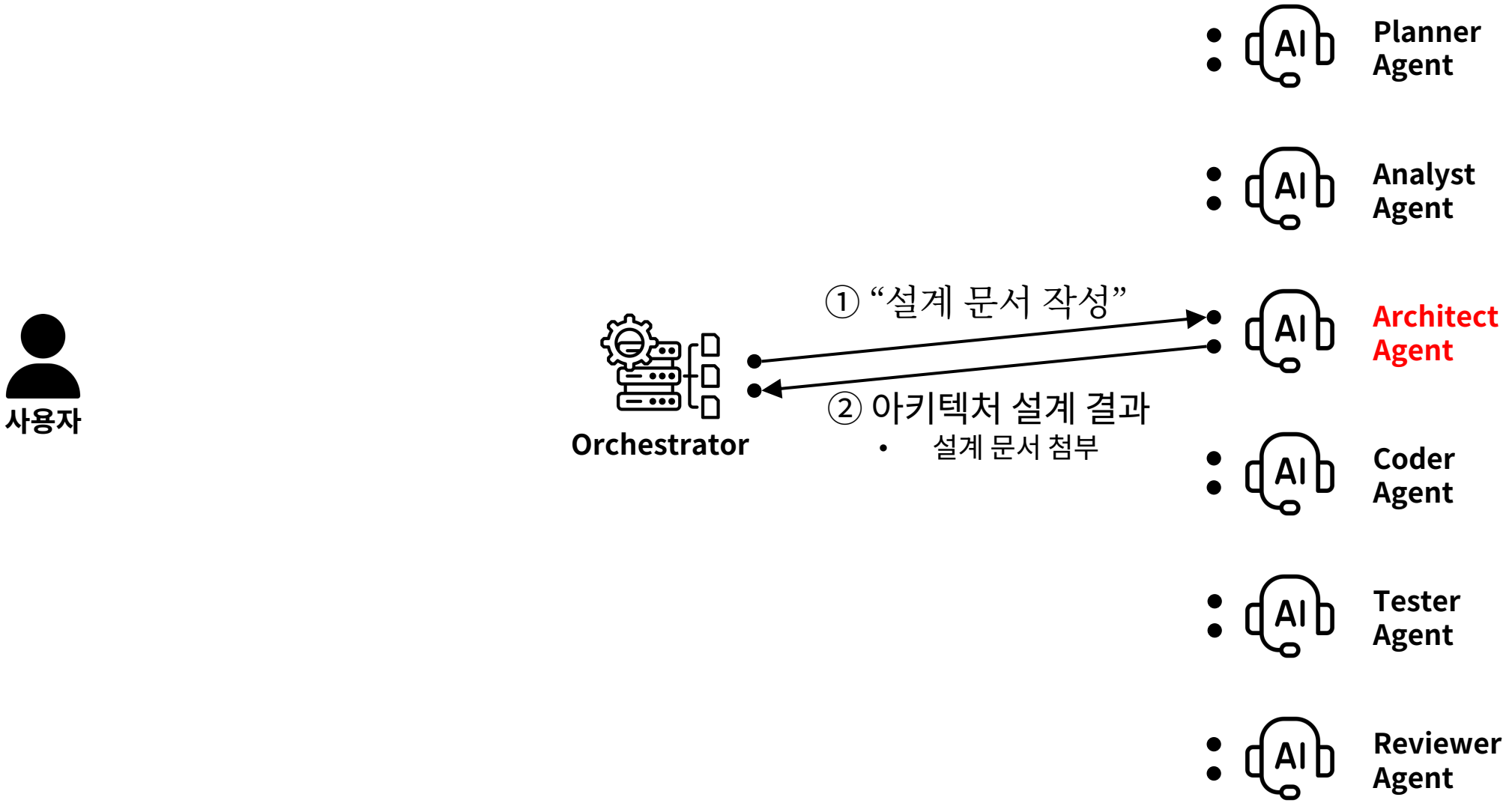
A2A 워크플로우 – 1단계) 작업 분해



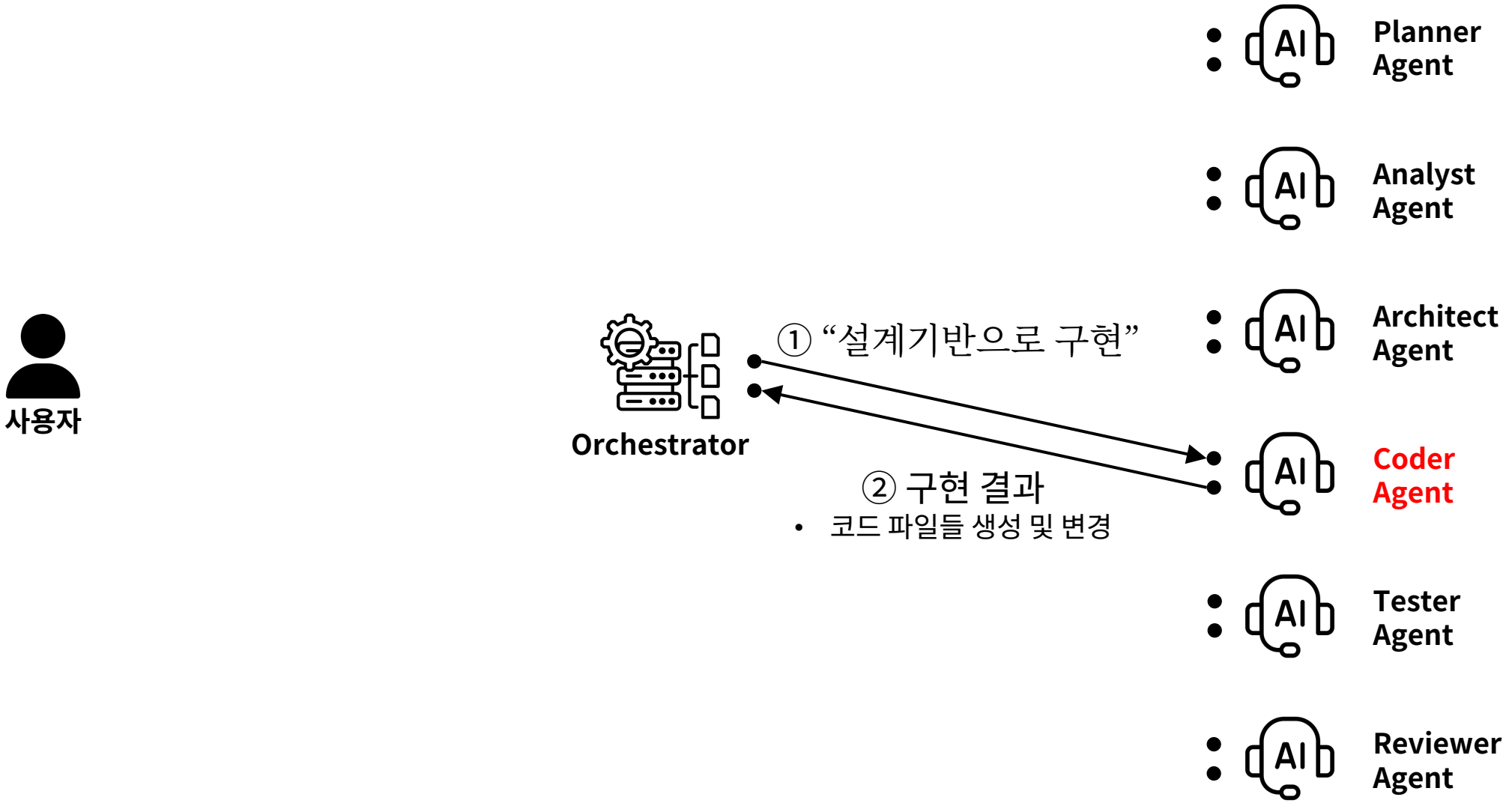
A2A 워크플로우 – 2단계) 요구사항 분석



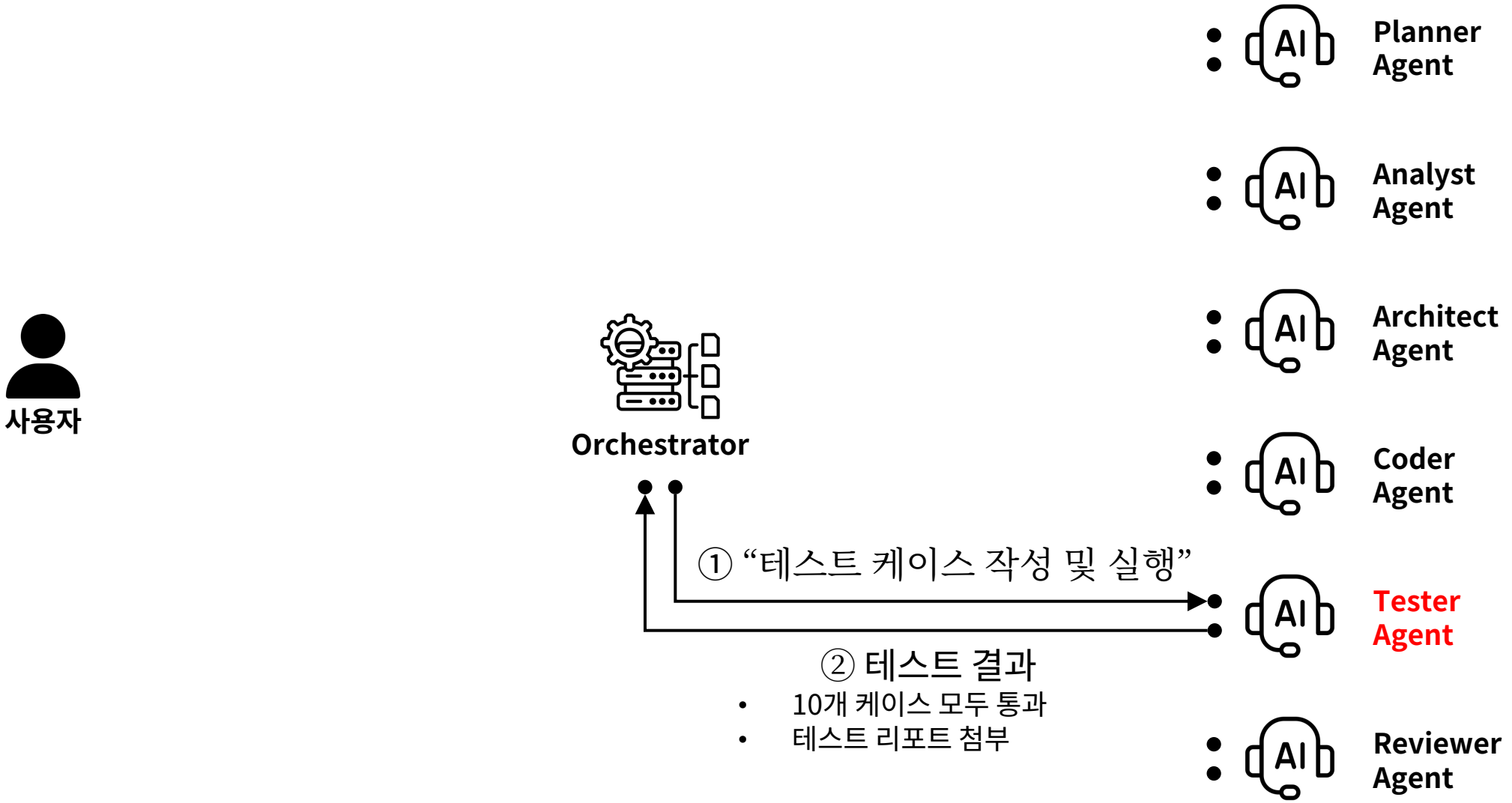
A2A 워크플로우 – 3단계) 설계



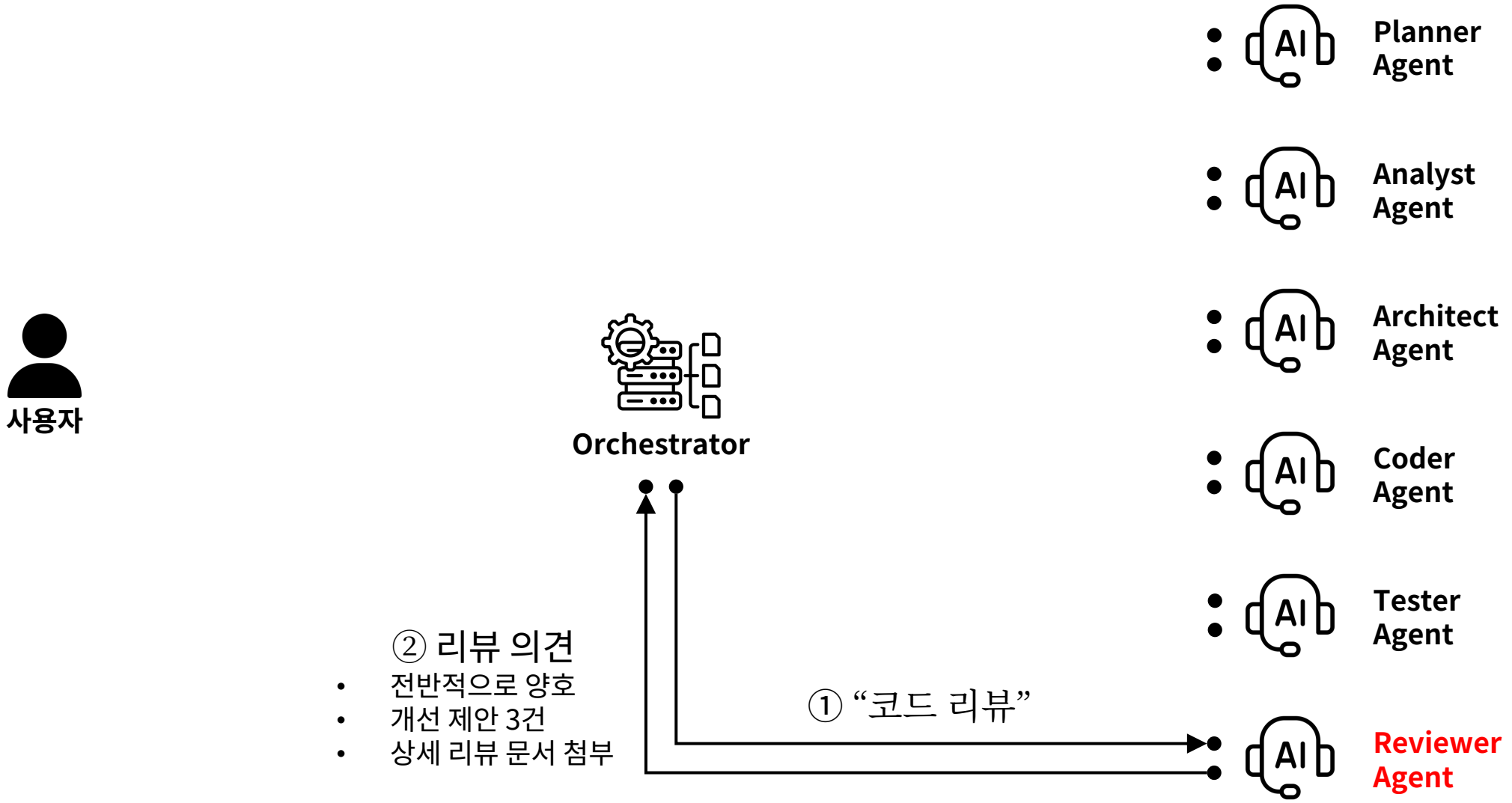
A2A 워크플로우 – 4단계) 구현



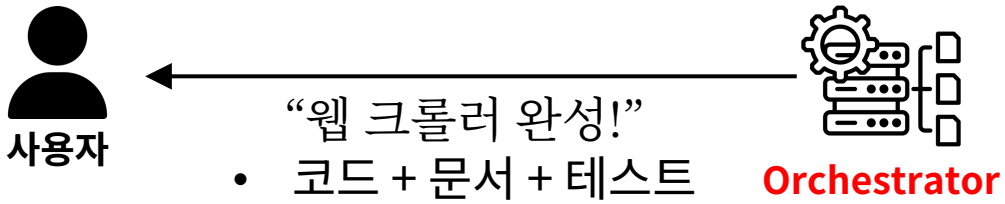
A2A 워크플로우 – 5단계) 테스트



A2A 워크플로우 – 6단계) 리뷰



A2A 워크플로우 – 7단계) 응답



- Planner Agent
- Analyst Agent
- Architect Agent
- Coder Agent
- Tester Agent
- Reviewer Agent

Multimadal

멀티모달 개요

정의

텍스트뿐 아니라 이미지, 음성, 영상, 센서 데이터 등 여러 형태(모달리티)의 입력과 출력을 동시에 이해하고 생성할 수 있는 AI 모델 및 시스템이다. 즉, 멀티모달은 **AI의 인식 범위를 텍스트 세계에서 현실 세계로 확장**하며, "보고·듣고·말하고·해석하는" 능력을 제공한다.

핵심 특징

- **다중 입력 처리 →** 텍스트 + 이미지 + 음성 + 영상 등 동시 이해
- **다중 출력 생성 →** 텍스트 설명, 이미지 생성, 음성 합성 등
- **현실 세계 인식 →** 문서, 사진, 화면, 음성 신호 해석
- **컨텍스트 융합 →** 서로 다른 모달 정보를 하나의 의미 공간에서 통합
- **Agent 친화적 →** "보고 판단 → 행동" 흐름과 자연스럽게 결합
- **확장성 →** 새로운 모달 추가 기능(센서, 로그, UI 상태 등)

멀티모달 개요

구성요소(입력 계층)

멀티모달 입력 (Input Modalities)

- AI가 받아들이는 형태:
  - 텍스트: 자연어, 명령어, 문서
  - 이미지: 사진, 스크린샷, 문서 스캔, 차트
  - 음성: 음성 파일, 실시간 오디오 스트림
  - 영상: 비디오 프레임, 장면, 자막
  - 기타: 센서 데이터, 로그, UI 상태
- 예:
  - 텍스트 + 이미지: “이 사진 설명해줘” + [image.jpg]
  - 음성 + 화면: 음성 명령 + 화면 캡처
  - 영상 + 텍스트: 비디오 + "이 장면에서 문제점은?"

구성요소(처리계층)

모달별 인코더 (Modality Encoders)

- 각 입력을 공통 의미 공간으로 변환
- 구성:
  - Vision Encoder: 이미지/영상 → 벡터
    예) CLIP, ViT (Vision Transformer)
  - Audio Encoder: 음성 → 벡터
    예) Whisper, Wav2Vec
  - Text Encoder: 텍스트 → 벡터
    예) BERT, GPT Tokenizer

멀티모달 통합 모델 (Multimodan Fusion Model)

- 여러 모달 정보를 통합하여 추론
- 기능:
  - Cross-model Attention: 모달 간 관계 파악
  - Context Fusion: 통합 컨텍스트 생성
  - Reasoning: 질문, 응답, 판단, 설명 생성
- 예시 모델
  - GPT-4V, Claude 3, Gemini, LLaVA

멀티모달 개요

구성요소(출력 계층)

멀티모달 출력 (Output Modalities)

- AI가 생성하는 형태:
  - 텍스트 텍스트: 설명, 분석, 요약, 명령어
  - 이미지: DALL-E, Midjourney, Stable Diffusion
  - 음성: TTS (Text-to-Speech)
  - 구조화 데이터: JSON, XML, 표
  - 코드: Python, SQL 등
- 예:
  - 텍스트 보고서 (분석 내용)
  - 차트 이미지 (시각화)
  - JSON 데이터 (구조화된 정보)

구성요소(통합계층)

(선택) Tool / MCP 연계

- 멀티모달 결과를 실제 행동으로 연결
- 예시:
  - 이미지 분석 → 결함 발견 → MCP로 유지보수 요청
  - 음성 명령 → 의도 파악 → MCP로 시스템 제어
  - 영상 분석 → 이상 감지 → Tool로 알림 발송

(선택) Agent / Orchestrator

- 멀티모달 사용 전략 결정
- 기능:
  - 어떤 모달을 사용할지 판단
  - 입력/출력 조합 결정
  - 멀티모달 결과 기반 후속 행동 수행
- 예시:
  - “이미지만으로 충분한가, 아니면 음성도 필요한가?”
  - “출력은 테스트만? 아니면 이미지 생성도?”
  - “결과가 신뢰할 만한가? 재확인 필요한가?”

멀티모달 개요

동작 방식 (멀티모달 파이프라인)

1. 입력 수집

- 사용자 또는 환경으로부터 다양한 모달 입력 수신
- 예시:
  - “이 사진에 문제가 있나?” + 사진첨부
  - 음성으로 질문 + 화면 공유
  - 비디오 + “이 장면 설명해줘”

2. 전처리

- 각 입력의 품질 향상
- 예시:
  - 이미지: 해상도 조정, 노이즈 제거
  - 음성: STT (Speech-to-Text), 노이즈 제거
  - 영상: 키 프레임 추출, 자막 분리
  - 텍스트: 정규화, 토큰화

3. 모달별 인코딩

- 각 입력을 벡터 표현으로 변환
- 예시:
  - 이미지 → 이미지 임베딩
  - 음성 → 텍스트 또는 음성 임베딩
  - 텍스트 → 텍스트 임베딩
- 모든 임베딩을 동일한 차원 공간으로 프로젝션

4. 컨텍스트 융합

- 여러 모달 정보를 하나의 컨텍스트로 겹합
- 방법:
  - Early Fusion: 인코딩 전 결합
  - Late Fusion: 인코딩 후 결합
  - Cross-modal Attention: 모달 간 상호작용 모델링
- 결과:
  - 텍스트 의미 + 이미지 내용 + 음성 뉘앙스가 하나의 의미 공간에서 통합됨

멀티모달 개요

동작 방식 (멀티모달 파이프라인)

5. 추론 및 생성

- 멀티모달 모델이 통합 컨텍스트를 기반으로 판단/응답 생성.
- 예시:
  - 이미지 설명: “사진 속 고양이가 소파에 앉아있습니다”
  - 문제 분석: “배관 연결부에서 누수 흔적이 보입니다”
  - 다음 행동 제안: “수리 기사 호출이 필요합니다”

6. (선택) 출력 생성

- 필요시 멀티모달 출력 생성
- 예시:
  - 텍스트 보고서
  - 이미지 생성 (문제 부위 강조)
  - 음성 안내 (TTS)

7. (선택) 실행 연계

- Agent가 결과를 바탕으로 MCP/Tool 호출.
- 예시:
  - 결함 감지 → 작업 지시 생성
  - 음성 명령 → 시스템 설정 변경
  - 이상 감지 → 알림 발송 + 로그 기록

멀티모달 개요

장점

- **현실 적합성 향상** → 실제 업무/환경 데이터를 직접 다룸
- **사용자 경험 개선** → 말하고, 보여주고, 듣는 자연스러운 인터페이스
- **정보 손실 감소** → 텍스트 변환 없이 원본 그대로 이해
- **Agent 능력 향상** → "보는 AI, 듣는 AI" 구현 가능
- **적용 범위 확대** → 제조, 의료, 보안, 고객지원 등

과제 및 한계

- **모델 비용 증가**: 멀티모달 모델은 연산 비용이 큼
- **정합성 문제**: 모달 간 정보 불일치 가능성
- **해석 오류**: 이미지/음성 인식 오류 전파
- **데이터 품질 의존성**: 흐릿한 이미지, 잡음 많은 음성
- **프라이버시 이슈**: 이미지/음성에 민감 정보 포함 가능
- **실시간 처리 어려움**: 영상/음성은 스트리밍 처리 복잡
- **언어/문화 편향**: 특정 언어/문화의 이미지에 편향

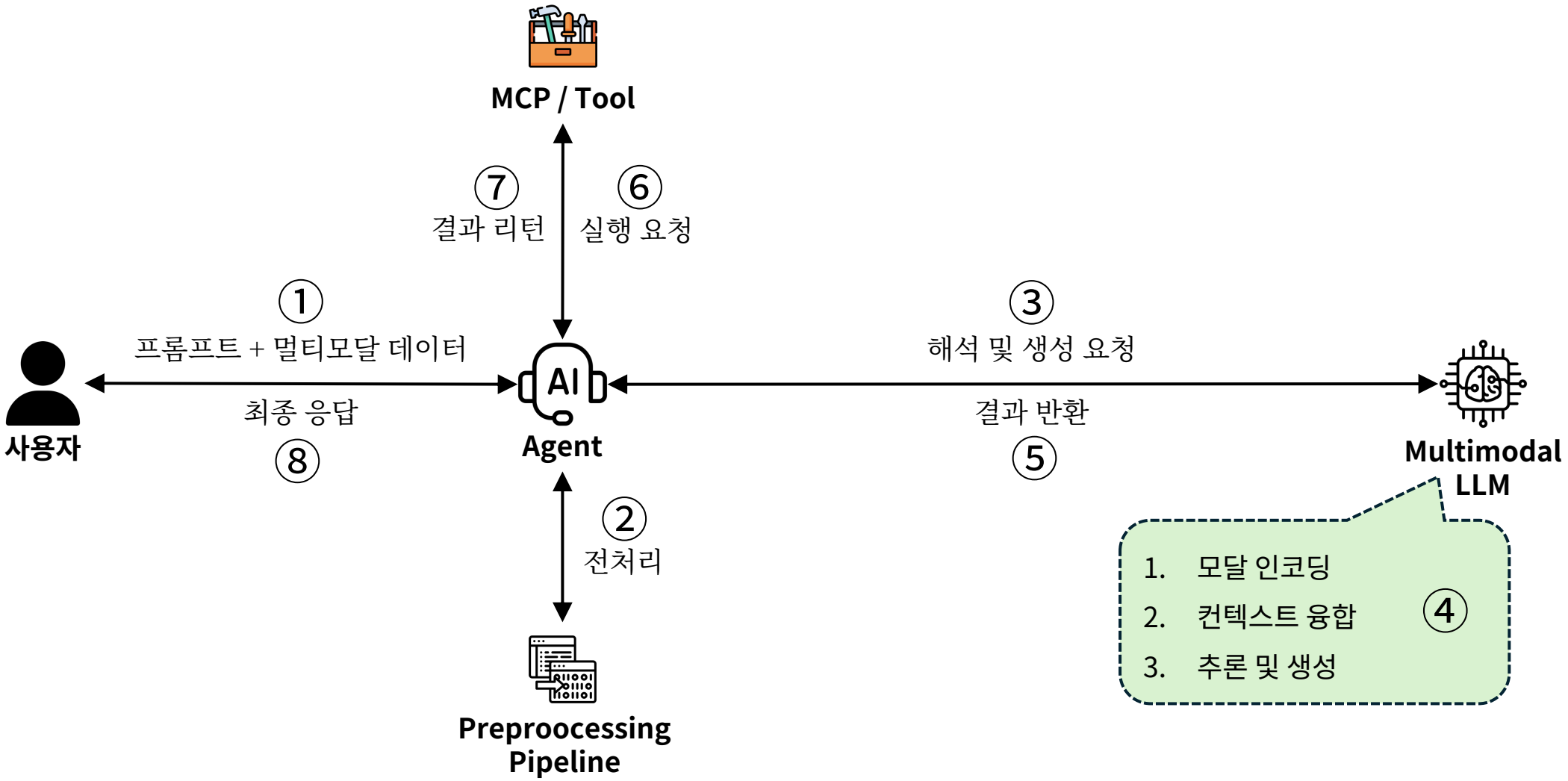
해결 방안

- 입력 전 정제(해상도, 노이즈 제거)
- 중요 판단은 텍스트 기반 이중 검증
- 모달별 신뢰도 점수 활용
- 민감 정보 마스킹/차단
- 멀티모달 사용 범위 최소화 원칙 적용

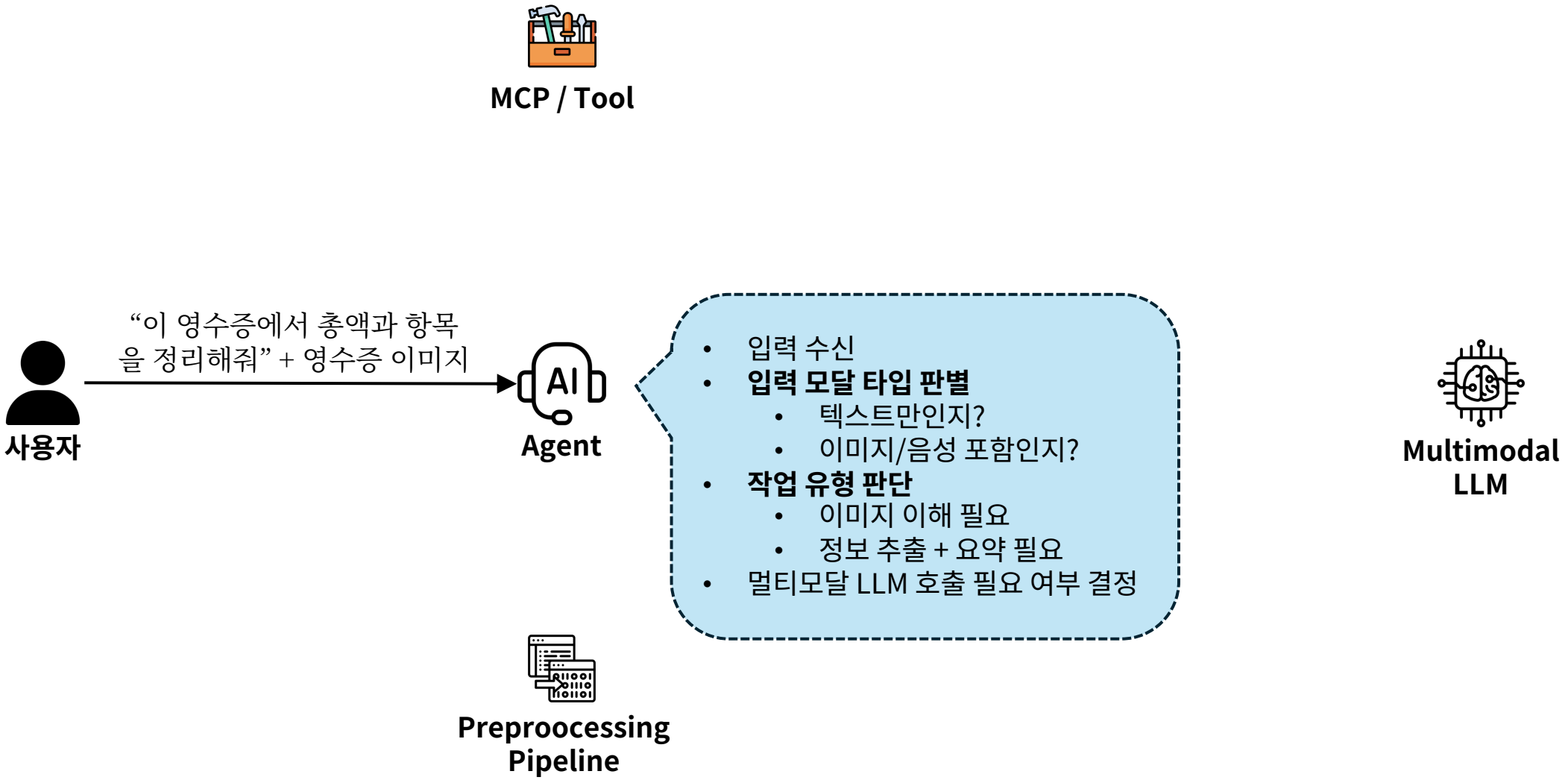
실제 사례

- 문서 이해 → 스캔 PDF 이미지 + 텍스트 질의
- 제조/품질 검사 → 설비 이미지 분석
- 의료 → 의료 영상 + 환자 기록 분석
- 고객 지원 → 스크린샷 기반 오류 진단
- 음성 비서 → 음성 명령으로 시스템 제어

멀티모달 워크플로우


MCP / Tool
⑦
결과 리턴
⑥
실행 요청
①
프롬프트 + 멀티모달 데이터
③
해석 및 생성 요청
사용자
AI
Agent
Multimodal
LLM
최종 응답
⑧
결과 반환
⑤
②
전처리
1. 모달 인코딩
2. 컨텍스트 융합
3. 추론 및 생성
④
Preproocessing
Pipeline

멀티모달 워크플로우 – 1단계) 멀티모달 입력 수신



멀티모달 워크플로우 – 2단계) 전처리 판단 및 준비

MCP / Tool

사용자

Agent

- **품질·안전·정책 관점에서 전처리 필요성 판단**
  - 이미지 해상도/크기가 적절한가?
  - 음성 길이가 제한 내인가?
  - 민감 정보 포함 가능성은?
- **필요시 Preprocessing Pipeline 호출:**
  - “이미지 1024x768 조정해줘”
  - “음성 30초로 분할해줘”

Multimodal LLM

Preproocessing Pipeline

멀티모달 워크플로우 – 2단계) 전처리 판단 및 준비

MCP / Tool

사용자

Agent

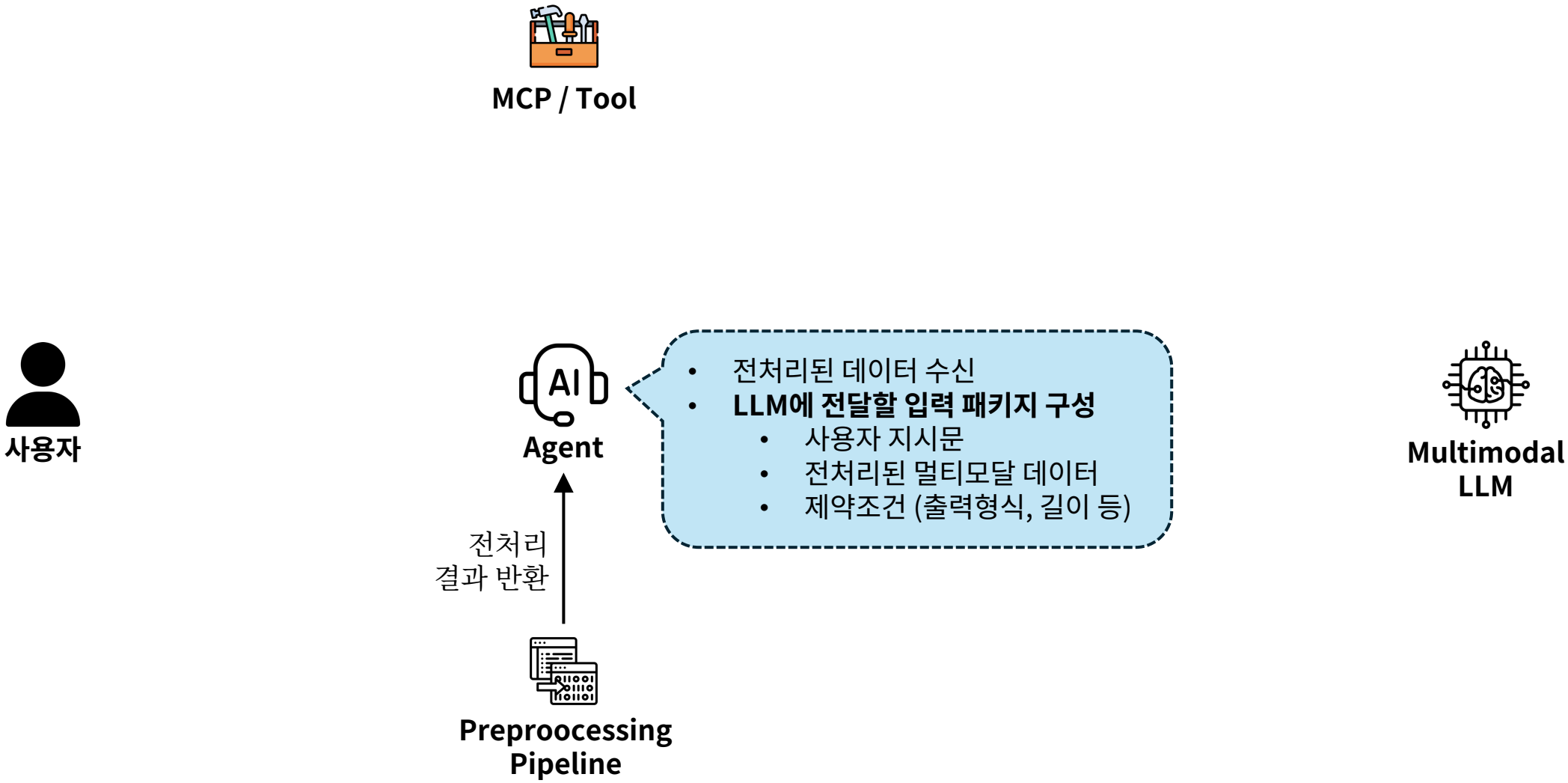
전처리 요청

Preproocessing Pipeline

- Agent 요청에 따라 **실제 전처리 실행**
  - 이미지: 리사이즈, 노이즈 제거, 정규화
  - 영상: 키 프레임 추출, 자막 분리
  - 음성: STT 전처리, 잡음 제거
- **LLM 입력 형식으로 변환**
  - 이미지 → Base64 또는 바이너리
  - 음성 → 텍스트 또는 오디오 임베딩 준비

Multimodal LLM

멀티모달 워크플로우 – 2단계) 전처리 판단 및 준비



멀티모달 워크플로우 – 3단계) 모달별 인코딩

MCP / Tool

사용자

Agent

해석 및 생성 요청 (입력 전달)

Multimodal LLM

Preproocessing Pipeline

- **모달별 인코딩**
  - 이미지 → 이미지 임베딩
  - 텍스트 → 텍스트 임베딩
  - 음성 → 음성/텍스트 임베딩
- **서로 다른 입력을 공통 의미 공간으로 변환**

멀티모달 워크플로우 – 4단계) 컨텍스트 융합

MCP / Tool

사용자

Agent

Multimodal LLM

- **텍스트 지시문 + 이미지/음성 의미를 결합**
- “무엇을 해야 하는지”를 시각 정보와 함께 이해
  예) “영수증에서 ‘총액’과 ‘항목’을 추출하라”

Preproocessing Pipeline

멀티모달 워크플로우 – 5단계) 추론•생성

MCP / Tool

사용자

Agent

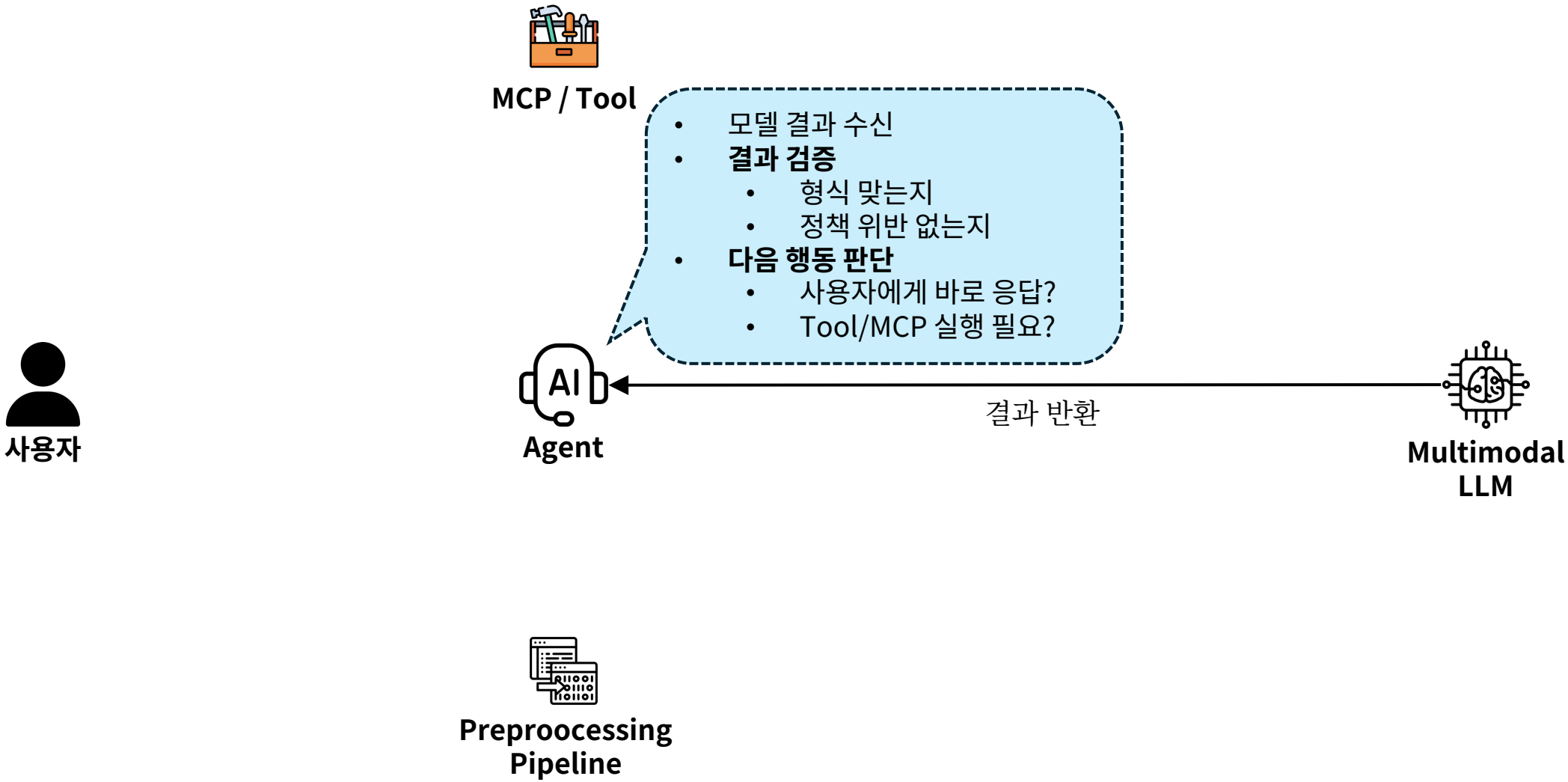
Preproocessing Pipeline

- 필요한 정보 추출
- **구조화(JSON 등) 또는 자연어 생성**
- 불확실성 판단
  예) “일부 항목은 흐려서 확실하지 않음”
- JSON 예:

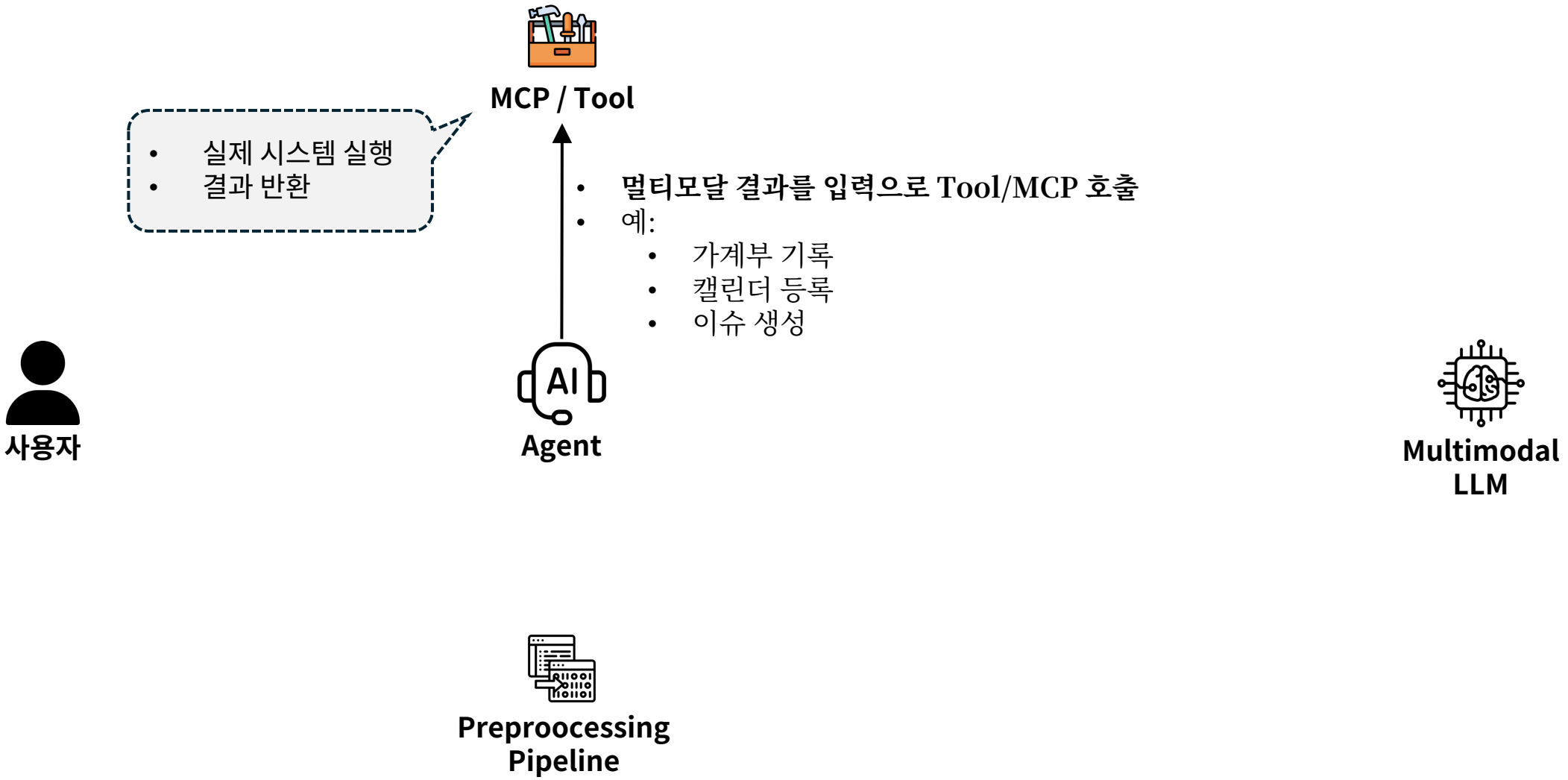
```
{
  "merchant": "ABC Coffee",
  "total": 13500,
  "items": [
    {"name": "Latte", "price": 5500},
    {"name": "Sandwich", "price": 8000}
  ]
}
```

Multimodal LLM

멀티모달 워크플로우 – 6단계) 결과 반환



멀티모달 워크플로우 – 7단계) 실행 연계



멀티모달 워크플로우 – 선택) 실행 결과를 LLM에게 해석 요청

MCP / Tool

사용자

Agent

해석 및 생성 요청

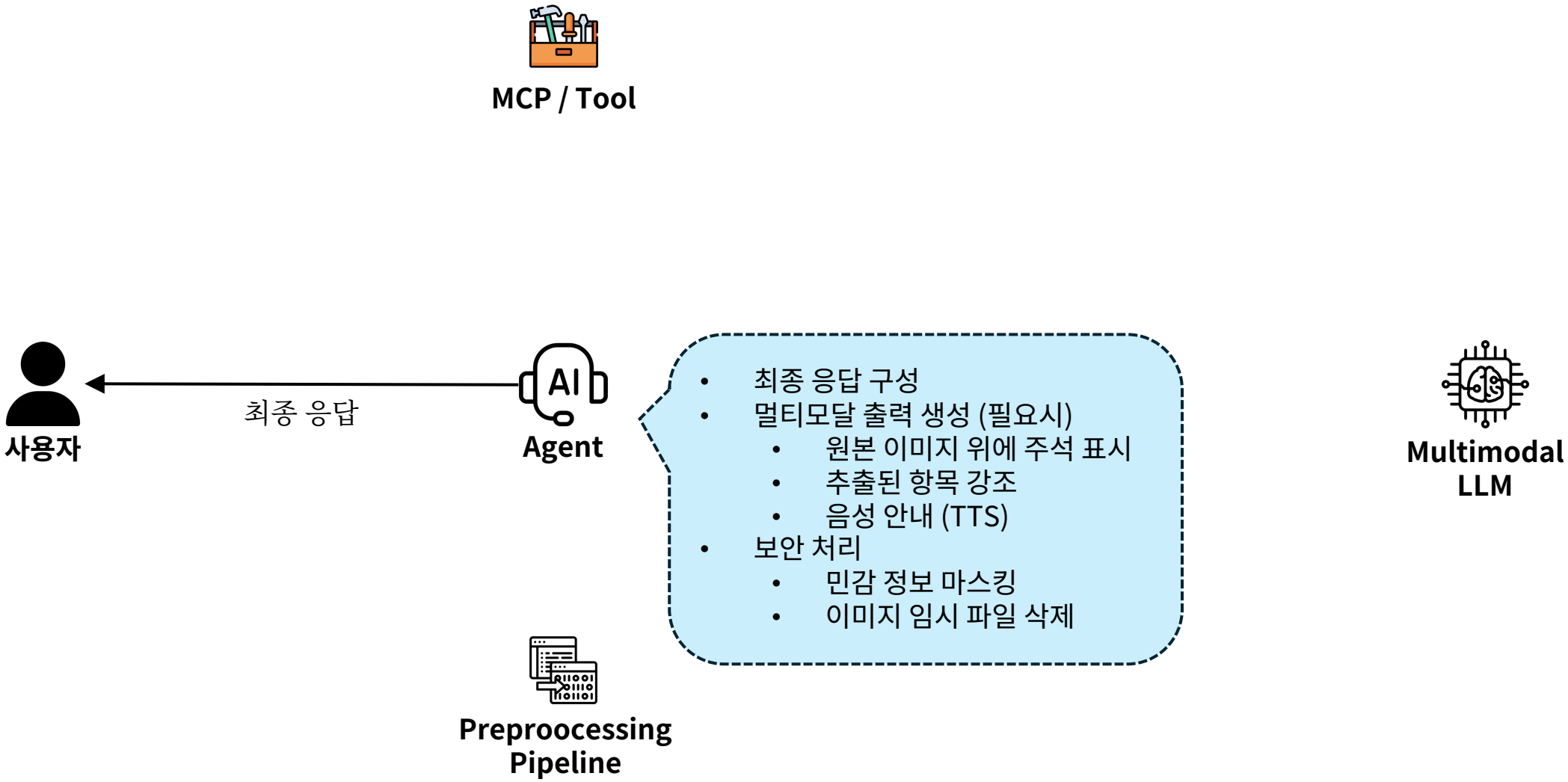
결과 반환

Multimodal LLM

- Tool/MCP 실행 결과를 다시 LLM에게 해석 요청

Preproocessing Pipeline

멀티모달 워크플로우 – 8단계) 최종 응답



멀티모달 워크플로우 예시1

- 제조 품질 검사 시스템 -

1단계) 멀티모달 입력 수신

MCP / Tool

프롬프트 + 사진 파일

사용자

Agent

- **텍스트 프롬프트 입력**
  “이 회로 기판 사진을 검사하고 불량이 있으면 작업 지시서 작성해줘”
- **이미지 첨부**

- **입력 수신 및 임시 저장**
- 입력 타입 식별
  - 입력 1: 텍스트 (UTF-8)
  - 입력 2: 이미지 (JPEG, 5MB)
- 보안 검증
  - 파일 크기 체크 (< 10MB)
  - 악성 파일 스캔
  - 허용된 확장자 확인

Multimodal LLM

Preproocessing Pipeline

2단계) 전처리

MCP / Tool

텍스트 전처리 담당자

사용자

Agent

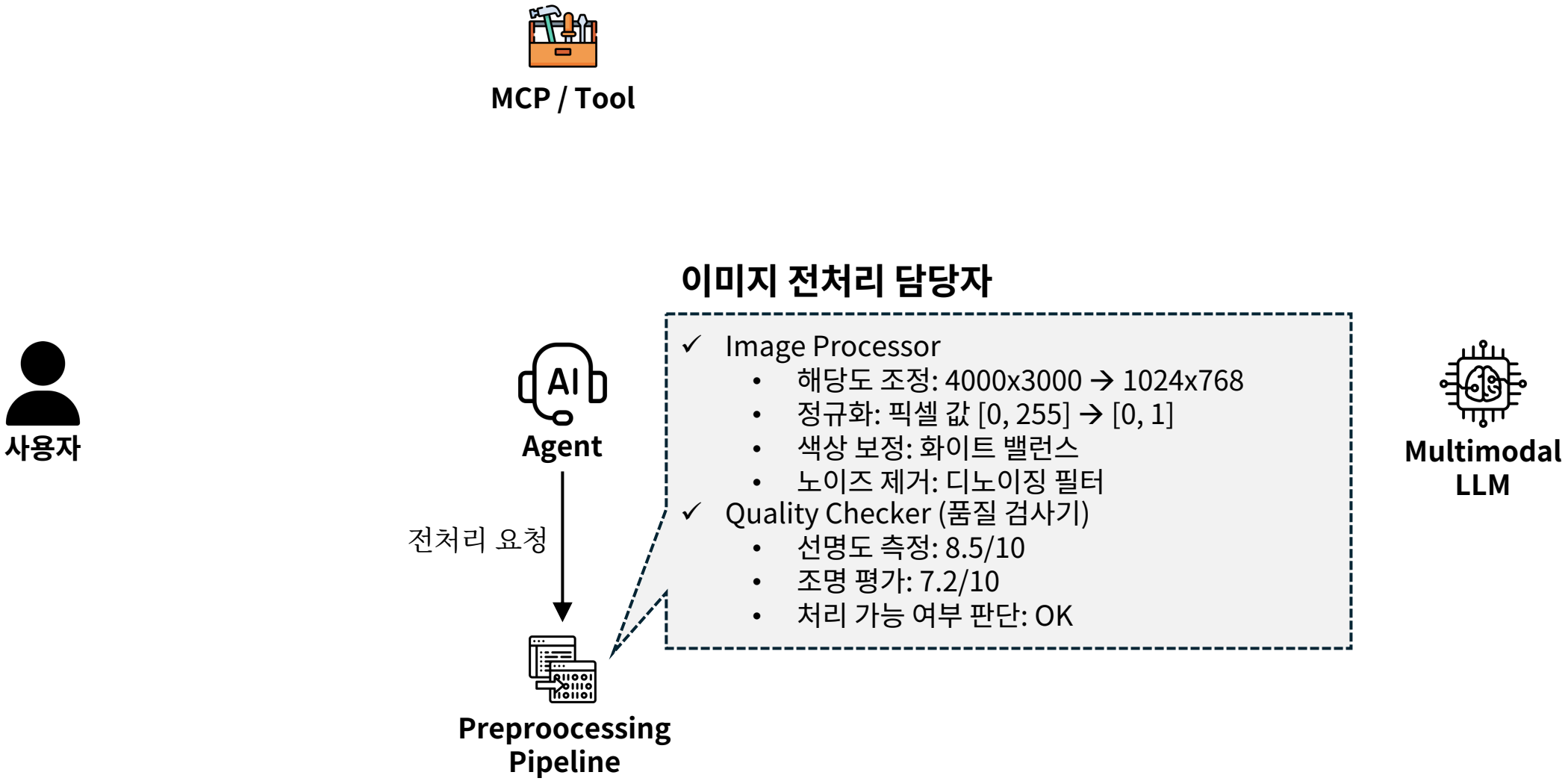
전처리 요청

Preproocessing Pipeline

- ✓ Tokenizer
  - 텍스트 분절: “회로”, “기판”, “사진”, “검사”, …
  - 특수문자 처리
  - 정규화
- ✓ Intent Classifier (의도 분류기)
  - 작업 유형 파악: “품질 검사” + “보고서 생성”
  - 키워드 추출: [“회로 기판”, “검사”, “불량”, “작업 지시서”]

Multimodal LLM

2단계) 전처리


MCP / Tool
사용자
AI
Agent
전처리 요청
Preproocessing
Pipeline
이미지 전처리 담당자
✓ Image Processor
• 해당도 조정: 4000x3000 → 1024x768
• 정규화: 픽셀 값 [0, 255] → [0, 1]
• 색상 보정: 화이트 밸런스
• 노이즈 제거: 디노이징 필터
✓ Quality Checker (품질 검사기)
• 선명도 측정: 8.5/10
• 조명 평가: 7.2/10
• 처리 가능 여부 판단: OK
Multimodal
LLM

3단계) 모달별 인코딩

MCP / Tool

사용자

Agent

해석 및 생성 요청 (입력 전달)

Multimodal LLM

Preproocessing Pipeline

3단계) 모달별 인코딩

MCP / Tool

사용자

Agent

Vision Encoder (이미지 → 벡터)

- ✓ 모델: ViT-Large (Vision Transformer)
- ✓ 처리:
  1. 이미지를 16x16 패치로 분할 (64x48=3072 패치)
  2. 각 패치를 768차원 벡터로 임베딩
  3. Position Embedding 추가
  4. Transformer 24 layers 통과
  5. [CLS] 토큰 추출 → Image Embedding
- ✓ 출력: [1, 768] 차원 벡터

Multimodal LLM

Preproocessing Pipeline

3단계) 모달별 인코딩

MCP / Tool

Text Encoder (텍스트 → 벡터)

- ✓ 모델: BERT-based Encoder
- ✓ 처리:
  1. 토큰을 임베딩으로 변환
  2. BERT 레이어 통과 (12 layers)
  3. [CLS] 토큰 추출 → Text Embedding
- ✓ 출력: [1, 768] 차원 벡터

사용자

Agent

Multimodal LLM

Preproocessing Pipeline

4단계) 컨텍스트 융합

MCP / Tool

Fusion Module

여러 모달 정보를 통합 컨텍스트로 융합

- ✓ Self-Attention (각 모달 내부)
  - Vision Self-Attention
    이미지 패치 간 관계 파악
  - Text Self-Attention
    단어 간 관계 파악
- ✓ Cross-Modal Attention (모달 간 상호작용)
  - Image Features ←→ Text Features
  - “이미지의 어느 부분이 텍스트의 어떤 단어와 관련?”
- ✓ 가중치 조합
  - Unified_Context =
    0.6 x Image_Context +
    0.4 x Text_Context +
    Cross_Attention(Image, Text)
- ✓ 출력: Unified Context Vector [1, 1024]

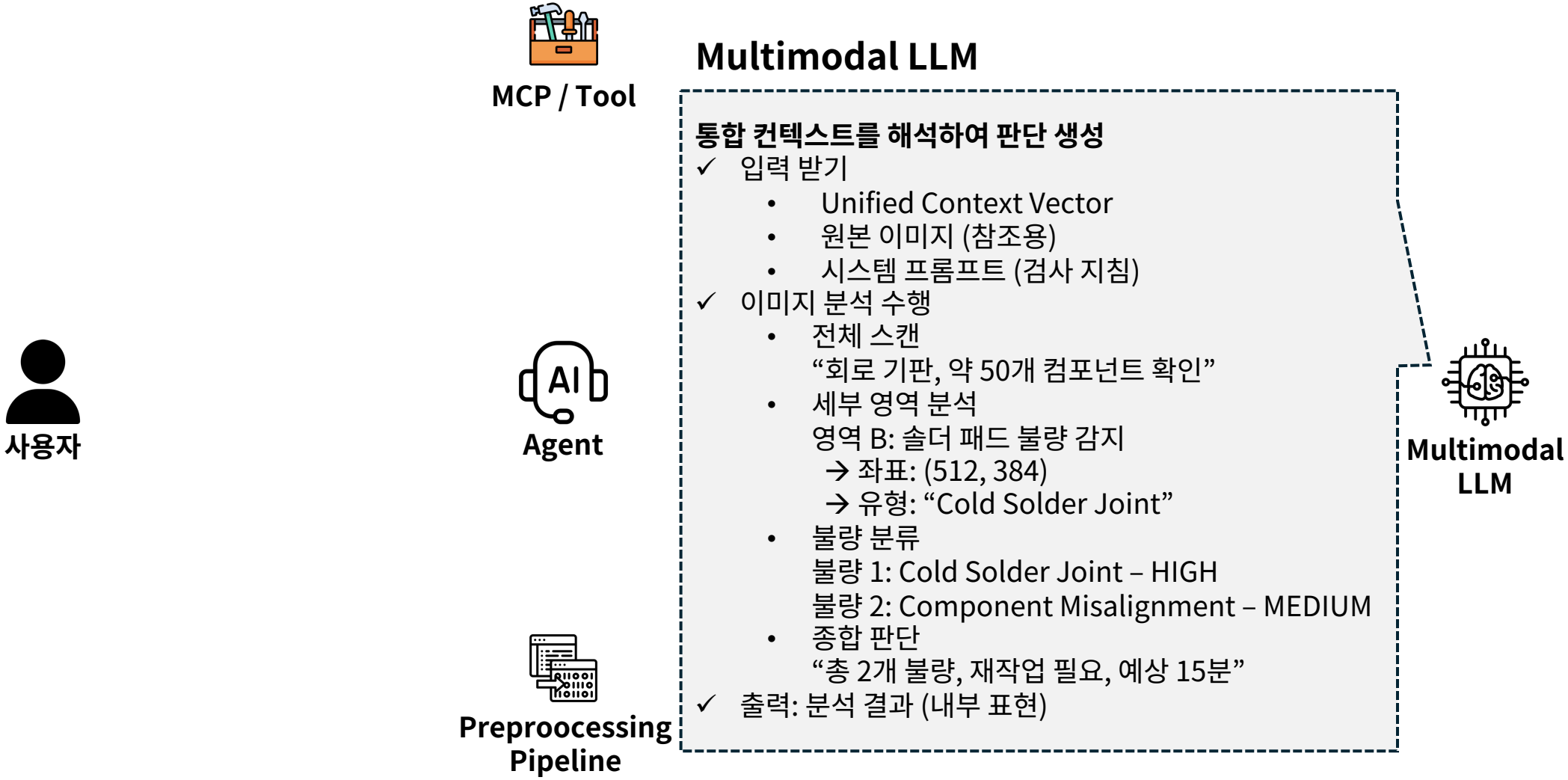
사용자

Agent

Multimodal LLM

Preproocessing Pipeline

5단계) 멀티모달 추론


MCP / Tool
Multimodal LLM
통합 컨텍스트를 해석하여 판단 생성
✓ 입력 받기
• Unified Context Vector
• 원본 이미지 (참조용)
• 시스템 프롬프트 (검사 지침)
✓ 이미지 분석 수행
• 전체 스캔
"회로 기판, 약 50개 컴포넌트 확인"
• 세부 영역 분석
영역 B: 솔더 패드 불량 감지
→ 좌표: (512, 384)
→ 유형: "Cold Solder Joint"
• 불량 분류
불량 1: Cold Solder Joint – HIGH
불량 2: Component Misalignment – MEDIUM
• 종합 판단
"총 2개 불량, 재작업 필요, 예상 15분"
✓ 출력: 분석 결과 (내부 표현)
사용자
AI
Agent
Multimodal LLM
Preproocessing Pipeline

6단계) 출력 생성

MCP / Tool

사용자

Agent

Preproocessing Pipeline

Multimodal LLM

분석 결과를 다양한 형태로 생성

- ✓ 텍스트 보고서 생성
  - 구조화된 한글 보고서
  - 불량 목록, 조치 사항, 예상 시간
- ✓ JSON 데이터 생성
  - inspection_id, defects 배열
  - 구조화된 정보 (시스템 간 연동용)
- ✓ 작업지시서
  - 재작업 지시서
  - 단계별 작업 내용

Multimodal LLM

6단계) 출력 생성

MCP / Tool

사용자

Agent

Image Generator

주석 이미지 생성

✓ 입력: 원본 이미지 + 불량 좌표

✓ 처리:

- 원본 이미지 위에 오버레이
- 빨간 박스 그리기 (512, 384)
- 라벨 추가: “Cold Solder”

✓ 출력: pcb_analysis_annotated.jpg

Multimodal LLM

Preproocessing Pipeline

6단계) 출력 생성

MCP / Tool

사용자

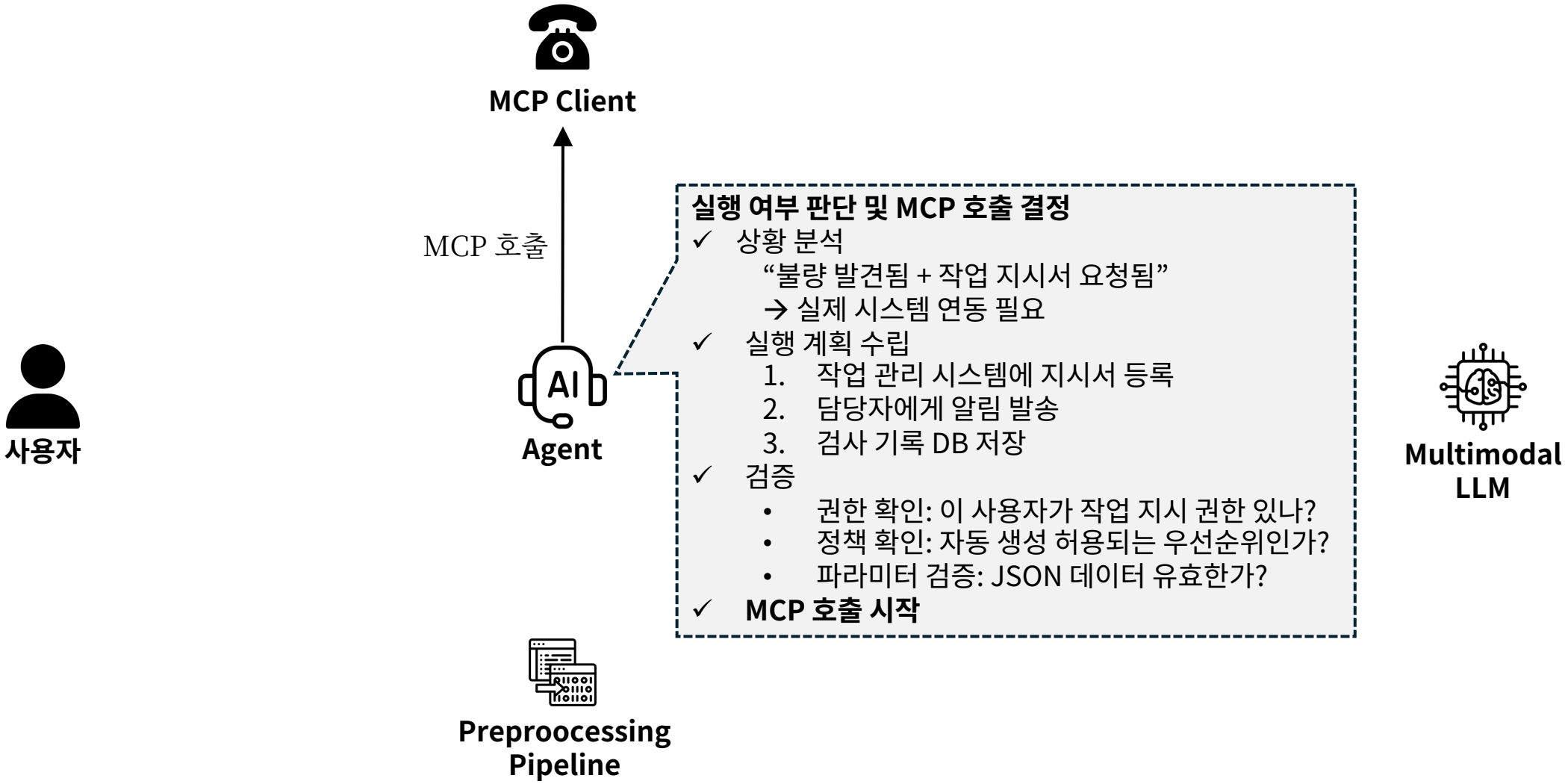
Agent

결과 반환

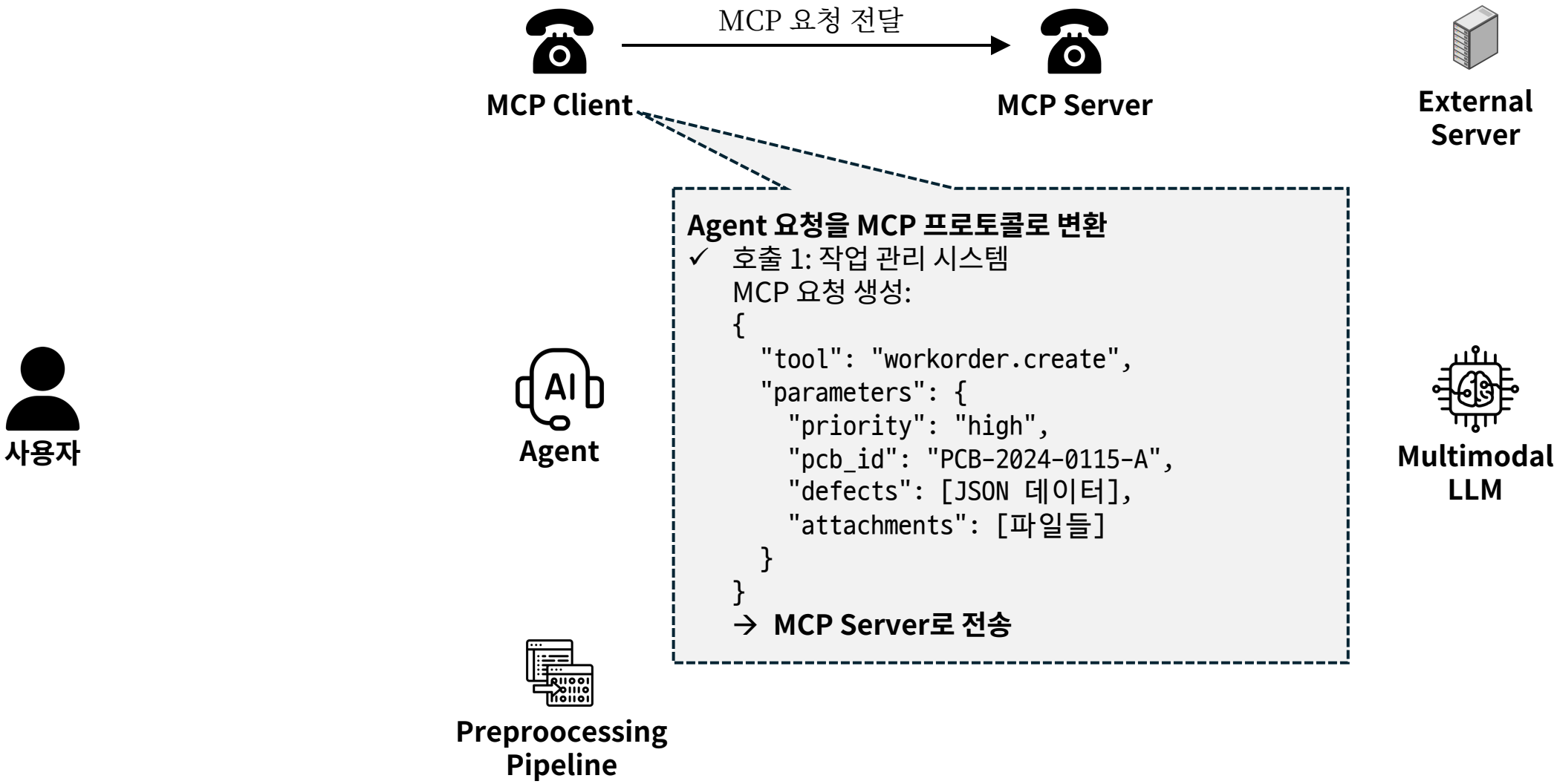
Multimodal LLM

Preproocessing Pipeline

7단계) 실행 연계(선택)


MCP Client
MCP 호출
사용자
AI
Agent
실행 여부 판단 및 MCP 호출 결정
✓ 상황 분석
"불량 발견됨 + 작업 지시서 요청됨"
→ 실제 시스템 연동 필요
✓ 실행 계획 수립
1. 작업 관리 시스템에 지시서 등록
2. 담당자에게 알림 발송
3. 검사 기록 DB 저장
✓ 검증
• 권한 확인: 이 사용자가 작업 지시 권한 있나?
• 정책 확인: 자동 생성 허용되는 우선순위인가?
• 파라미터 검증: JSON 데이터 유효한가?
✓ MCP 호출 시작
Multimodal LLM
Preproocessing Pipeline

7단계) 실행 연계(선택)


MCP 요청 전달
MCP Client
MCP Server
External Server
Agent 요청을 MCP 프로토콜로 변환
✓ 호출 1: 작업 관리 시스템
MCP 요청 생성:
{
"tool": "workorder.create",
"parameters": {
"priority": "high",
"pcb_id": "PCB-2024-0115-A",
"defects": [JSON 데이터],
"attachments": [파일들]
}
}
→ MCP Server로 전송
사용자
AI
Agent
Multimodal LLM
Preproocessing Pipeline

7단계) 실행 연계(선택)

MCP Client

MCP Server

External Server

MCP 요청을 실제 Tool 호출로 중개

✓ 요청 수신 및 검증
- MCP 프로토콜 문법 확인
- Tool 권한 확인
- 파라미터 스키마 검증

✓ Tool 실행 준비
- Work_order_system Tool 선택
- API 키 로드
- 연결 확인

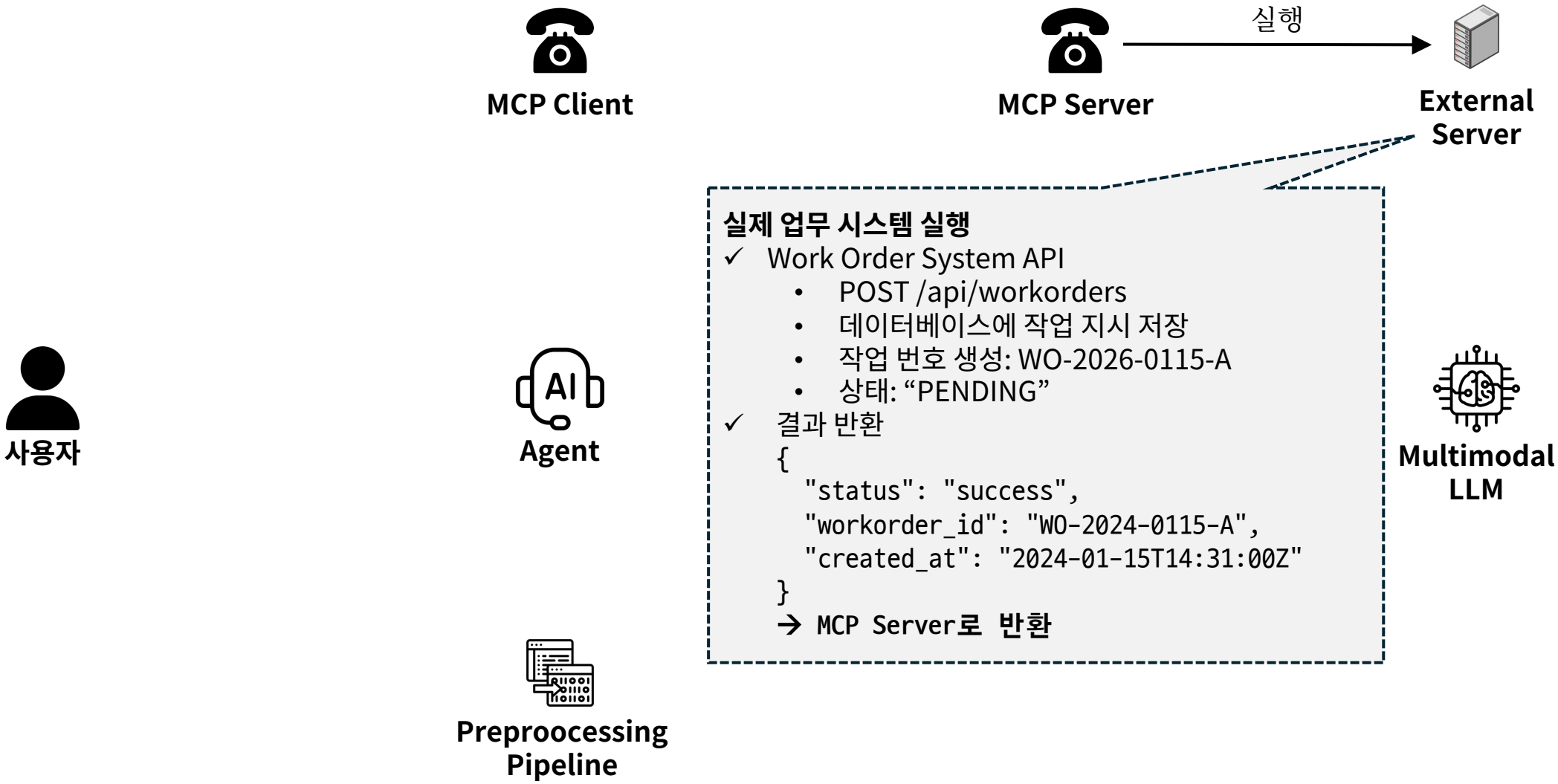
사용자

Agent

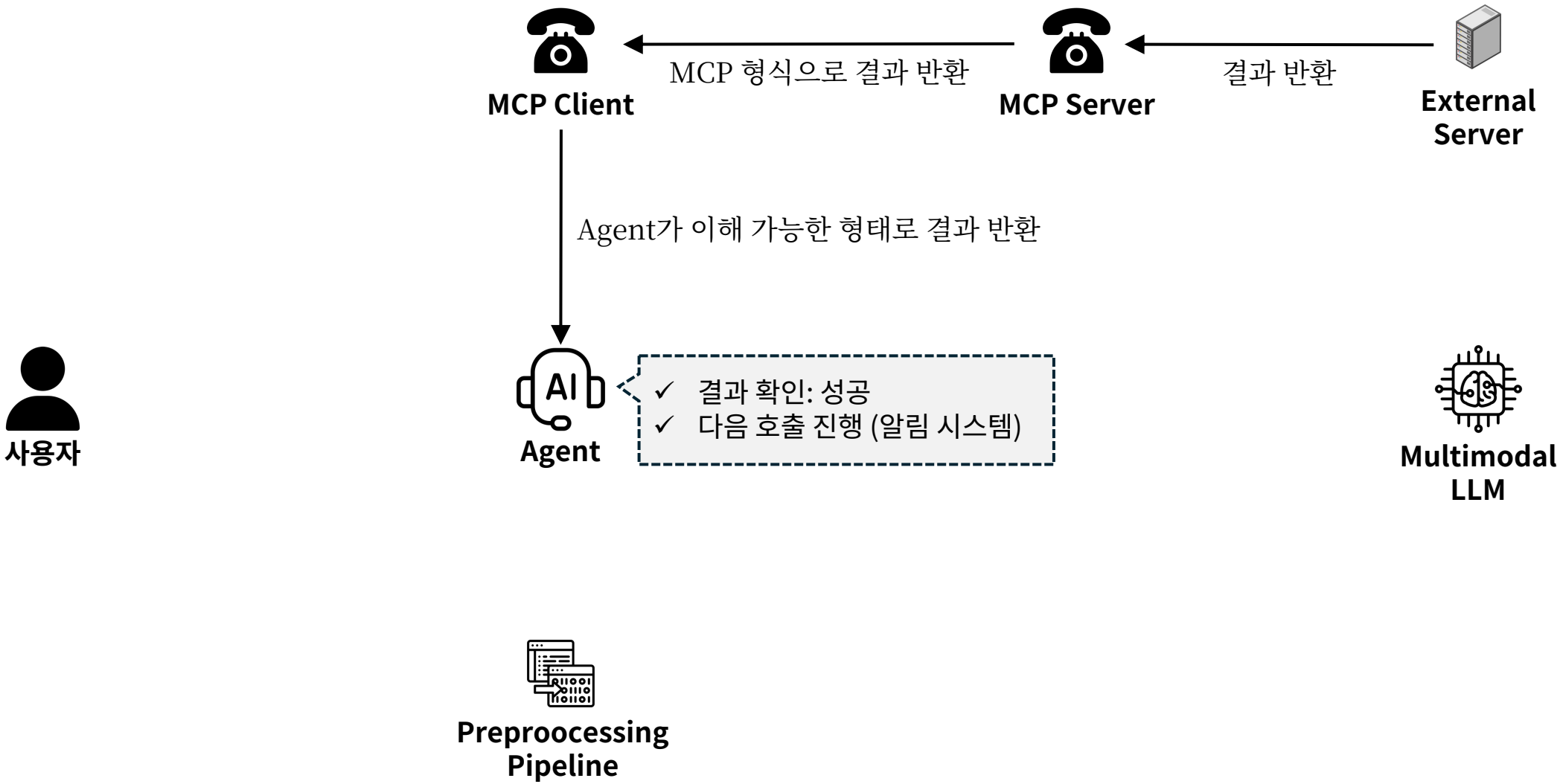
Multimodal LLM

Preproocessing Pipeline

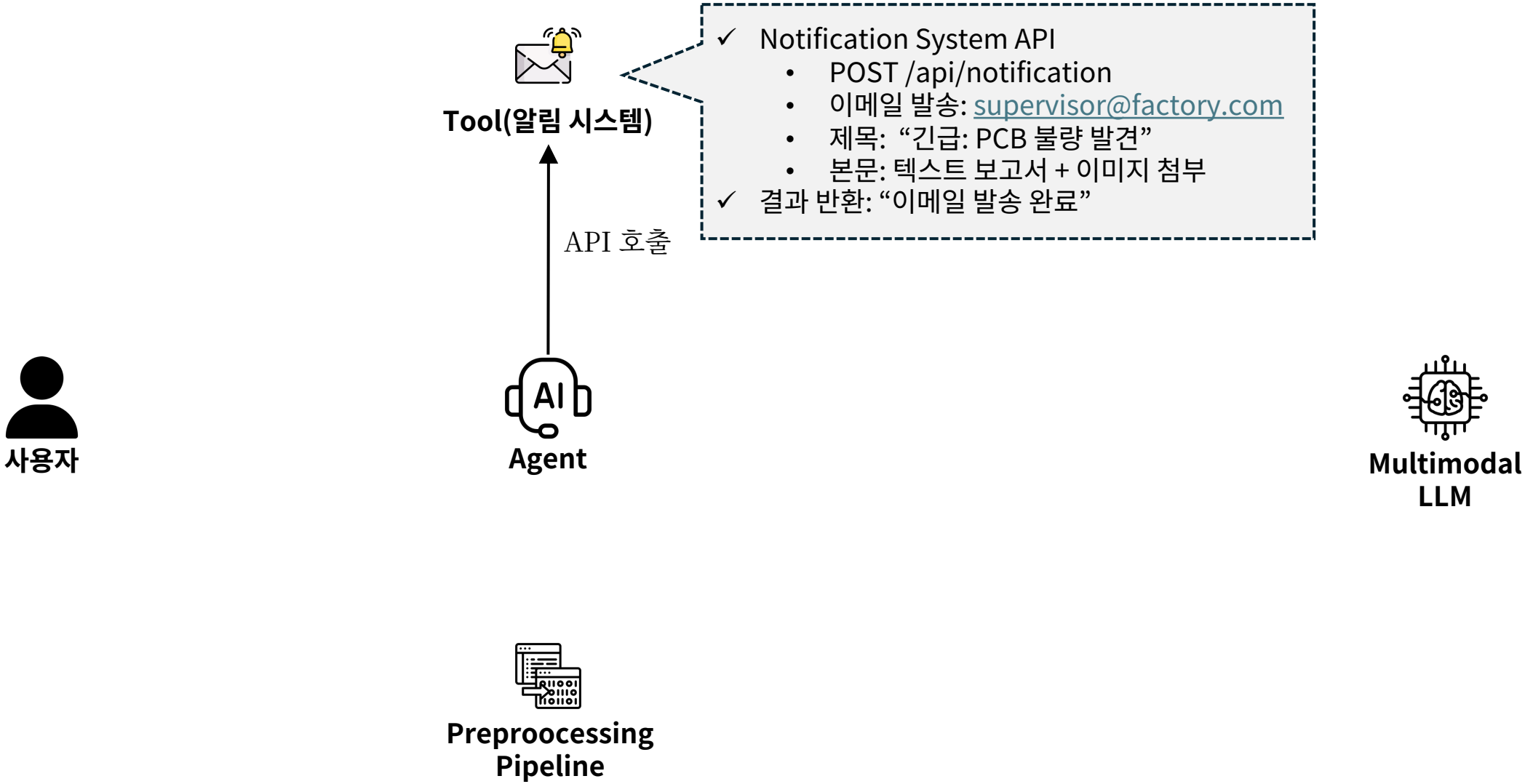
7단계) 실행 연계(선택)


MCP Client
MCP Server
실행
External Server
실제 업무 시스템 실행
✓ Work Order System API
• POST /api/workorders
• 데이터베이스에 작업 지시 저장
• 작업 번호 생성: WO-2026-0115-A
• 상태: “PENDING”
✓ 결과 반환
{
"status": "success",
"workorder_id": "WO-2024-0115-A",
"created_at": "2024-01-15T14:31:00Z"
}
→ MCP Server로 반환
사용자
AI
Agent
Multimodal LLM
Preproocessing Pipeline

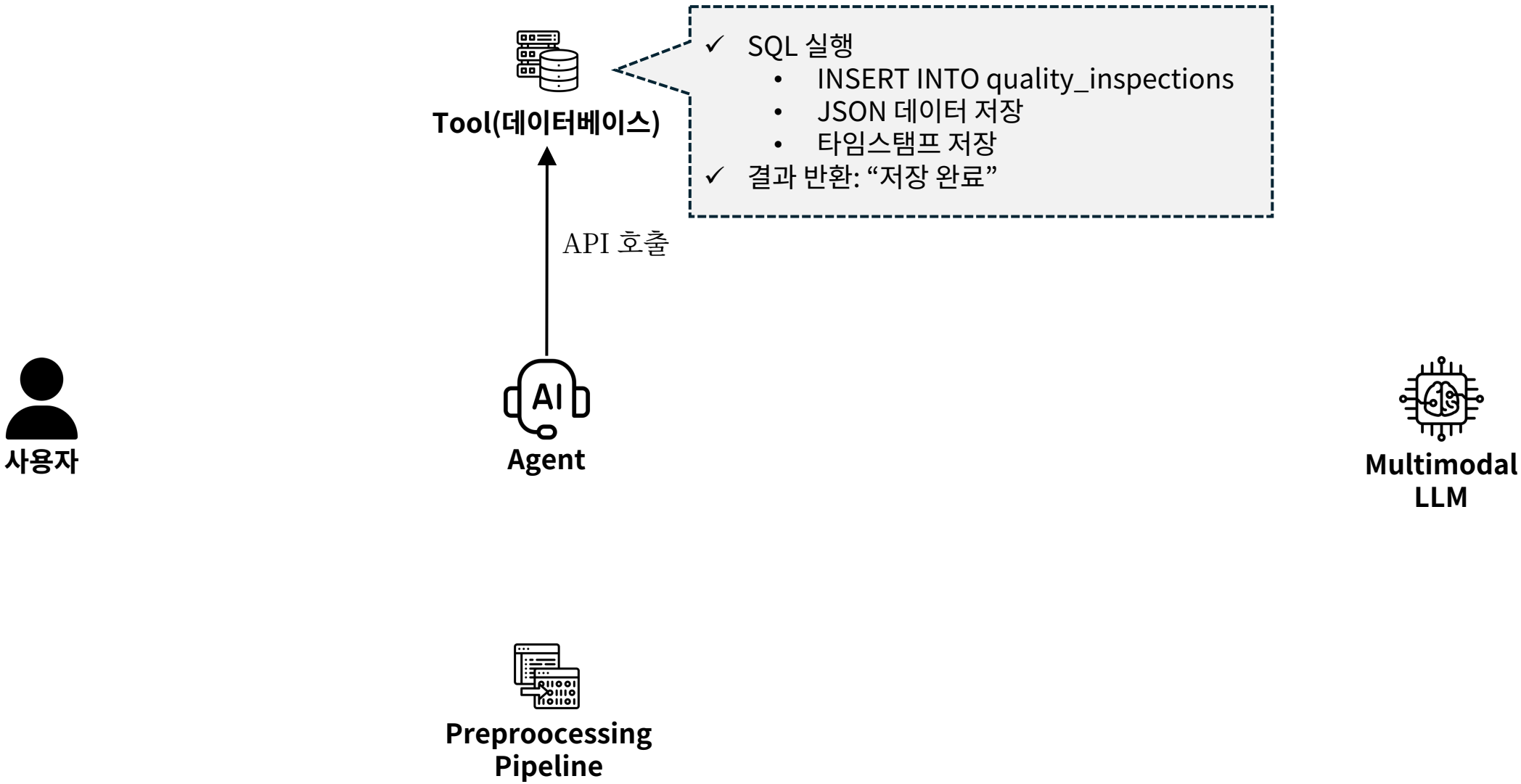
7단계) 실행 연계(선택)


MCP Client
MCP 형식으로 결과 반환
MCP Server
결과 반환
External Server
Agent가 이해 가능한 형태로 결과 반환
사용자
AI
Agent
✓ 결과 확인: 성공
✓ 다음 호출 진행 (알림 시스템)
Multimodal LLM
Preproocessing Pipeline

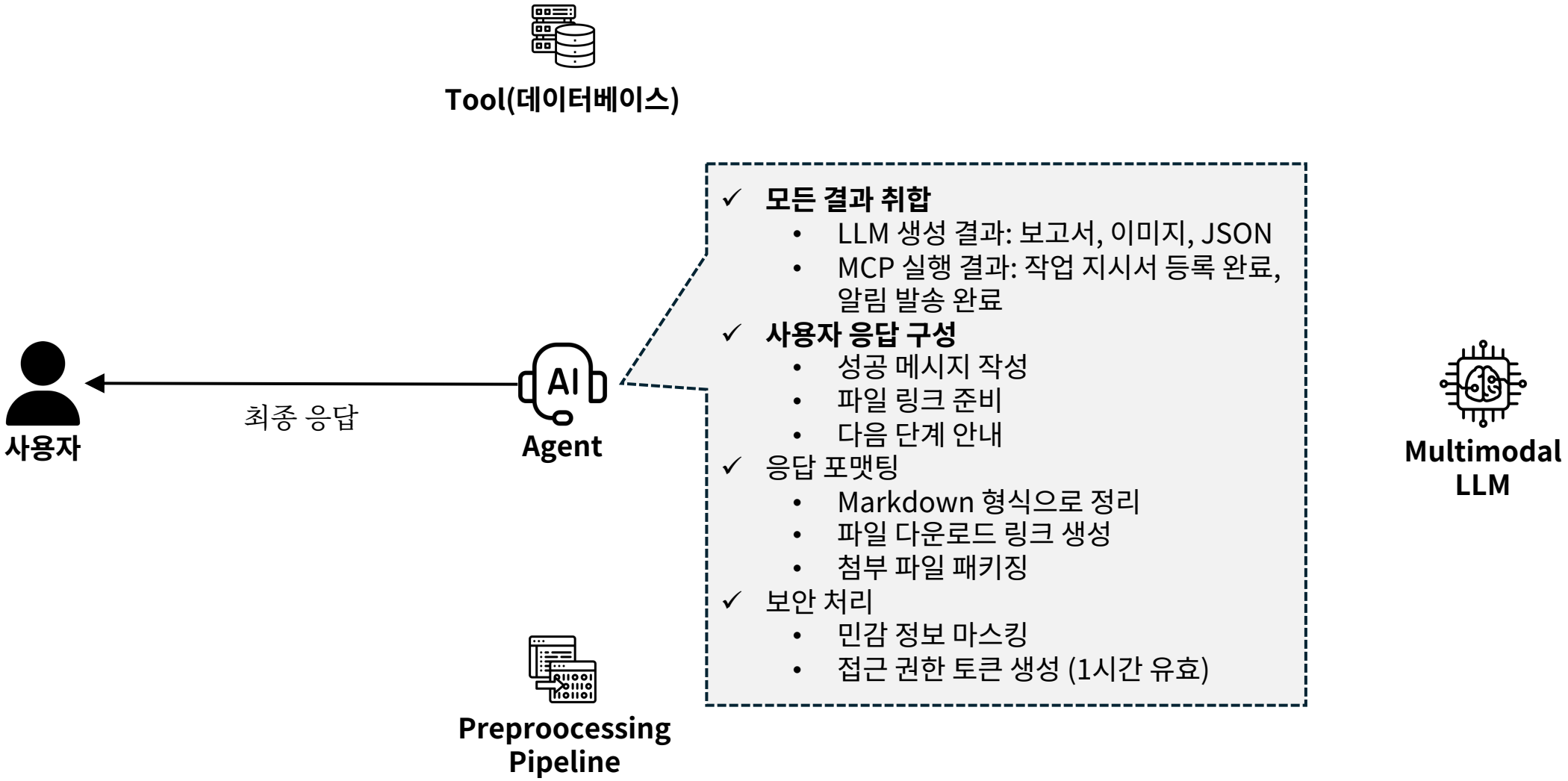
7단계) 실행 연계(선택)


✓ Notification System API
• POST /api/notification
• 이메일 발송: supervisor@factory.com
• 제목: “긴급: PCB 불량 발견”
• 본문: 텍스트 보고서 + 이미지 첨부
✓ 결과 반환: “이메일 발송 완료”
Tool(알림 시스템)
API 호출
사용자
AI
Agent
Multimodal LLM
Preproocessing Pipeline

7단계) 실행 연계(선택)


✓ SQL 실행
• INSERT INTO quality_inspections
• JSON 데이터 저장
• 타임스탬프 저장
✓ 결과 반환: “저장 완료”
Tool(데이터베이스)
API 호출
AI
Agent
사용자
Multimodal LLM
Preproocessing Pipeline

8단계) 최종 응답


Tool(데이터베이스)
✓ 모든 결과 취합
• LLM 생성 결과: 보고서, 이미지, JSON
• MCP 실행 결과: 작업 지시서 등록 완료, 알림 발송 완료
✓ 사용자 응답 구성
• 성공 메시지 작성
• 파일 링크 준비
• 다음 단계 안내
✓ 응답 포맷팅
• Markdown 형식으로 정리
• 파일 다운로드 링크 생성
• 첨부 파일 패키징
✓ 보안 처리
• 민감 정보 마스킹
• 접근 권한 토큰 생성 (1시간 유효)
AI
Agent
최종 응답
사용자
Multimodal LLM
Preproocessing Pipeline